BA3-CE10

Ethernet PAC

for Building Automation

マニュアル

MG CO., LTD.
www.mgco.jp

NM-7289-B 9版

(このページは空白です)

はじめに

本マニュアルは、Programming Automation Controller(以下、「本製品」および「コントローラ」)に搭載のソフトロジック用専用ファンクション及びファンクションブロックについて説明します。

ご使用になる前に本書をよくお読み頂き、正しくお使い下さい。

なお、本マニュアルは、IEC61131-3仕様を理解している方を前提に作成しています。用語については、それぞれの文献を参照して下さい。

マニュアルについて

本マニュアルに記載されている記号、および共通注意事項は以下のとおりです。

■記号説明

警告

取扱いを誤った場合に危険な状況が起こりえて死亡または重傷を受ける可能性が想定されることを示しています。

注意

取扱いを誤った場合に危険な状況が起こりえて中程度の傷害や軽傷を受ける可能性が想定される場合および物的損害だけの発生が想定されることを示しています。この注意に記載した事項でも状況により重大な結果に結びつく可能性があります。

補足

操作時のヒント、追加情報や補足事項を記載しています。

いずれも重要な内容を記載していますので厳守してください。

本マニュアルは必要なときに読めるよう大切に保管すると共に必ず最終ユーザまでお届けいただくようお願いいたします。

安全上のご注意

(ご使用の前に必ずお読みください)

本製品のご使用の際には本マニュアルおよび関連マニュアルをよくお読みいただき安全に対しての十分な注意と配慮、および正しい取扱いをしていただくようお願いいたします。

■設計上の注意事項

警 告

- フィールドバスを含むネットワークが交信異常になったときの動作状態についてはそのネットワークに関連するマニュアルを参照してください。誤出力や誤動作により事故の恐れがあります。
- インターネット経由の外部機器からの不正アクセスに対してコントローラの安全を保つ必要があるときはユーザによる対策を盛り込んでください。
- 運転中のユーザアプリケーションやデータを変更するときは常時システム全体が安全側に働くようにユーザアプリケーション上でインターロック回路を構成してください。またユーザアプリケーションの変更、パラメータ変更や運転状態の変更を行うときは関連するマニュアルを熟読し十分に安全を確認してから行ってください。
- FLAGSエリアのユーザ使用可能領域以外の領域に対するデータ書込みを行うとコントローラが誤動作する危険性があります。

■取付け上の注意事項

注 意

- 本製品や使用するIOカードはそれぞれに用意されたマニュアルに記載されている環境にて使用してください。それ以外の環境で使用すると感電、火災、誤動作、製品の損傷あるいは劣化の原因になります。
- 本製品やIOカードの着脱は必ずシステムで使用している外部供給電源を全相遮断してから行ってください。全相遮断しないと製品の損傷の恐れがあります。
- 本製品やIOカードの導電部分や電子部品には直接触らないでください。製品のの誤動作や故障の原因になります。

■配線上の注意事項

注 意

- 外部接続用コネクタはメーカ指定の工具で正しく圧着、圧接またはハンダ付けをしてください。接続が不完全な場合は短絡、火災、誤動作の原因になります。
- 本製品やIOカードに接続する通信ケーブルや電源ケーブルはダクトに納めるかクランプにより固定処理を行ってください。ケーブルがダクトに納められなかったりクランプによる固定処理をされないとケーブルのふらつき、移動や不注意の引っ張りなどによる製品やケーブルの破損あるいはケーブルの接続不良による誤動作の原因となります。

■保守時の注意事項

注 意

- 製品の分解や改造はしないでください。故障、誤動作、ケガ、火災の原因になります。
- カードの着脱は必ずシステムで使用している外部供給電源を全相遮断してから行ってください。全相遮断しないとカードの故障や誤動作の原因になります。
- 通電中に端子に触れないでください。誤動作の原因になります。
- 清掃、端子ネジの増し締めは必ずシステムで使用している外部供給電源を全相遮断してから行ってください。全相遮断しないとカードの故障や誤動作の原因になります。
- カードに触れる前には必ず接地された金属などに触れて人体などに帯電している静電気を放電してください。静電気を放電しないとカードの故障や誤動作の原因になります。

■運転時の注意事項

注 意

- 運転中のユーザアプリケーション変更、データ変更や運転状態の変更を行うときは十分に安全を確認してから行ってください。ユーザアプリケーション変更、データ変更、運転状態の変更を誤るとシステムの誤動作や機械の破損や事故の原因になります。

製品の適用について

1. 本製品をご使用にあたり万一本製品に故障・不具合などが発生したとしても重大な事故にいたらない用途であり、故障・不具合発生時にはバックアップやフェールセーフ機能が本製品の外部でシステム的に実施されていることを使用の条件とさせていただきます。
2. 本製品は一般工業などへの用途を対象とした汎用品として設計・製作されています。

依って以下のような機器やシステムなどの特殊用途への適用を除外させていただきます。万一使用された場合は弊社として製品の品質、性能、安全に関る一切の責任(債務不履行責任、瑕疵担保責任、品質保証責任、不法行為責任、製造物責任を含むがそれらに限定されない)を負わないものとさせていただきます。

- 各電力会社の原子力発電所およびその他発電所向けなどの公共への影響が大きい用途
- 鉄道各社および官公庁などの特別な品質保証体制の構築を弊社にご要求になる用途
- 航空宇宙、医療、鉄道、燃焼・燃料装置、乗用移動体、有人搬送装置、娯楽機械、安全機械など生命、身体や財産に大きな影響が予測される用途

注 意

- 本書の内容に関しては、改良のために予告なしに仕様等変更することがありますのでご了承ください。
- 本書の内容の一部または全部を無断で複写、複製、転載することを禁じます。
- 本書の内容に関しては万全を期しておりますが、万一ご不審なな点や誤りなどお気付きのことがありましたら、お手数ですが巻末記載宛てまでご連絡ください。

著作権・商標権について

- Windowsはマイクロソフト社の登録商標です。
- そのた、本文中に掲載しているシステム名および製品名は、それぞれ各社の商標または登録商標です。

目 次

1.コントローラ

1.1.仕様

ここでは次の項目について説明しています。

入出力インタフェース

種類	属性		最大	説明
R3	入出力カード		13	本コントローラ、電源カードを除く
		Analog Input	256	専用のファンクションブロックでアクセス
		Analog Output	256	専用のファンクションブロックでアクセス
		Digital Input	1024	専用のファンクションブロックでアクセス
		Digital Output	1024	専用のファンクションブロックでアクセス

種類	属性	最大	説明
Modbus	Digital Outputs	4096	専用のファンクションブロックでアクセス
	Digital Inputs	4096	
	Input Registers	1024	
	Output Registers	17472	

プログラミング言語

言語	説明
CFC	Continuous Function Chart (FBDに属する)
FBD	Function Block Diagram
IL	Instruction List

言語	説明
LD	Ladder Diagram
SFC	Sequential Flow Chart
ST	Structured Text

IECプログラム(テンプレートを使用しないでプロジェクトを作成した場合)

タスク	範囲	説明
Cyclic	最大10本	定周期実行タスク: 5ms ～ 30000ms周期(デフォルト500ms)
Event	最大10本	イベント実行タスク
Freewheeling	最大10本	フリー実行タスク: 周期毎に10msのウェイトが自動で挿入
総タスク数	最大10本	Cyclic + Event + Freewheeling の合計
監視機能	Watchdog	5ms ～ 5000ms
R3系入出力更新	非周期定周期	周期は10ms以内

ソフトロジックメモリ容量

メモリ領域	最大	説明
アプリケーション(RAM)	768KB	ユーザプログラムのコード格納領域
データ領域(RAM)	512KB	ユーザプログラムのデータ格納領域
ソースファイル制限	800KB	ユーザプログラムソースコード最大サイズ
ストレージ領域(FlashROM)	2048KB	ユーザプログラム(Boot Application)とソースコード格納領域
保持変数領域(不揮発領域)	32KB	RETAIN領域 : 16KB フラグ領域(%M) : 16KB
永続変数領域(不揮発領域)	4KB	PERSISTENT領域
ポイント履歴(不揮発領域)	5000レコード	この数値が「コントローラ内最大ポイント履歴レコード数」とな、最大履歴ユニット数は 1 ～ 50 の範囲で変更可能です。 初期設定は 最大履歴ユニット数 : 50Units ユニット毎の最大履歴レコード数 : 100Records (5000/50units)

IECプログラム(テンプレートを使用してプロジェクトを作成した場合)

項目	仕様	説明
プログラム	PLC_PRG	PLC_PRGはMainTask(Freewheeling)で実行されます。 このPLC_PRGはCPUが低負荷でありユーザプログラムの演算時間が5ms の場合のとき設計上 -65 ～ + 65ms の誤差が生じます。 この誤差はCPU負荷が高くなると更に大きくなります。
プログラム	PLC_PRG_DEF PLC_PRG_500ms PLC_PRG_1 PLC_PRG_5s PLC_PRG_20s	これらのプログラムはタスクに登録されていませんがPLC_PRG内からサブルーチンとして呼び出されます。
タスク	MainTask	Priority:1, Type:Freewheeling, Watchdog: 500ms PLC_PRGを実行します。
プログラム	PLC_PRG_BA3DLink	これはGlobal Data Pointを処理するためのプログラムです。 デバイスデバイスツリーの[BA3DLink]に登録されたデータポイントを処理します。
タスク	SubTask_BA3DLink	Priority:2, Type:Cyclic T#200ms PLC_PRG_BA3Dlinkを実行します。

1.2.通信設定

ここでは次の項目について説明しています。

IP address

IPアドレスの最終バイト値は本体前面にある2つのロータリースイッチで設定できます。また、その値は次の意味を持ちます。

ROTARY SW	内容
0	FlashMemoryに記憶しているアドレスを使用 (出荷時点、メモリ初期化直後は 192.168.1.200 が割り当てられています)
1 - 254	FlashMemoryに記憶しているアドレスを使用し、その最終バイト値をこのスイッチ値としたアドレスを使用
255	DHCPプロトコルにより本体起動時アドレスを構成 (注意:構成に失敗した場合IPアドレスが割り当てられないので通信できなくなります)

設定値範囲

項目	内容	既定値
IP Address	IP address (nnn.nnn.nnn.nnn 最大15文字)	192.168.1.200
Net Mask	IP address (nnn.nnn.nnn.nnn 最大15文字)	255.255.255.0
Default Gateway	IP address (nnn.nnn.nnn.nnn 最大15文字)	0.0.0.0
DNS Server	IP address (nnn.nnn.nnn.nnn 最大15文字)	0.0.0.0
DHCP Server	IP address (nnn.nnn.nnn.nnn 最大15文字)	0.0.0.0
SNTP Server 1,2	IP address / Host name (最大64文字)	ntp.nict.jp ntp.ring.gr.jp
FTP User *1	最大8文字	admin
FTP Password *1	最大8文字	12345

*1: 機能が搭載されている機種のみ有効です

1.3.ModbusTCP Interface

ここでは次の項目について説明しています。

Support Function Code

Function	Code
Read Coils	01
Read Discrete Inputs	02
Read Holding Registers	03
Read Input Registers	04
Write Single Coil	05
Write Single Register	06
Write Multiple Coils	15
Write Multiple Registers	16

Modbus領域割り付け

Modbus Address	領域サイズ	IEC表記	内容	IEC-Program	領域
000001 ～ 002048	2048 bits		R3 Digital Output 領域 (Data:64bits + Status:64bits) x 16slots	-	A *3
002049 ～ 004096	2048 bits	%MX0.0 ～ %MX127.15	Memory Output ビット 領域 Modbus Bit --> IEC BOOL	RW *1	F1 *2
100001 ～ 102048	2048 bits		R3 Digital Input 領域 (Data:64bits + Status:64bits) x 16slots	-	B
102049 ～ 104096	2048 bits	%MX128.0 ～ %MX255.15	Memory Input ビット領域 IEC BOOL --> Modbus Input	RW	F2 *2
300001 ～ 300512	512 words		R3 Analog Input 領域 (Data:16words + Status:16words) x	-	C

Modbus Address	領域サイズ	IEC表記	内容	IEC-Program	領域
			16slots		
300513 ～ 301024	512 words	%MW256 ～ %MW767	Memory Input ワード領域 IEC WORD --> Modbus Register	RW	F3 *2
400001 ～ 400512	512 words		R3 Analog Output 領域 (Data:16words + Status:16words) x 16slots	-	D *3
400513 ～ 401024	512 words	%MW768 ～ %MW1279	Memory Output ワード領域 Modbus Register <--> IEC WORD	RW *1	F4 *2
410001 ～ 418192	8,192 words	RETAIN変数	不揮発(RETAIN) Memory 領域 RETAIN 16kb (8 KWord) システム領域なので書き換えは行わないでください	-	E
420001 ～ 421280 421281 ～ 428192	8,192 words	%MW0 ～ %MW1279 %MW1280 ～ %MW8191	不揮発(Flag) Memory 領域 [%M] FLAG領域 16kb (8 KWord) %MW0 ～ %MW1279 は上記で領域割り当て済み %MW1280 ～ %MW8191 はフリー領域(使用可能)	RW	F *2
430001 ～ 430064	64 words	-	システム領域	-	G

*1) Modbus側(外部)からの書き込みとファンクションブロックによる書き込みで、読み出しファンクションブロックのUpdated Flag が更新されます。

*2) Modbus側(外部)からみてF1～F4の領域はF領域の一部です。異なるModbusアドレスでも%Mの示すアドレスが同じであれば同一領域をアクセスします。ただし、F1, F4の2つの領域への書き込みはファンクションブロックのUpdate Flagが更新されますが、F領域への書き込みでは更新されないことに注意してください。

*3)Data領域は読み書き可能、Status領域は読み込み専用です。Status領域への書き込みはエラーにはなりませんが無視されています。

領域	使用関数
F1	ModbusSlaveGetBit, ModbusSlaveSetBit
F2	ModbusSlaveGetBit, ModbusSlaveSetBit
F3	ModbusSlaveSet16, ModbusSlaveSet32, ModbusSlaveSetREAL, ModbusSlaveSetLREAL
F4	ModbusSlaveGet16, ModbusSlaveGet32, ModbusSlaveSet16,

領域	使用関数
	ModbusSlaveSet32 ModbusSlaveGetREAL, ModbusSlaveGetLREAL, ModbusSlaveSetREAL, ModbusSlaveSetLREAL
F	ModbusSlaveGet16, ModbusSlaveGet32, ModbusSlaveSet16, ModbusSlaveSet32 ModbusSlaveGetREAL, ModbusSlaveGetLREAL, ModbusSlaveSetREAL, ModbusSlaveSetLREAL

注 意

- 全ての領域は、Modbus通信データのバイト順で格納されています。IECプログラムからは各領域を直接%Mとしてアクセスするのではなく、ModbusSlaveGet***, ModbusSlaveSet***ファンクションブロックを使用して適切なバイト順のデータに変換する必要があります。
- IECプログラムとModbus通信の書き込みリクエストや読み出しリクエストへのレスポンスで返されるデータとの同時性はModbusSlaveGet***, ModbusSlaveSet***ファンクションブロックを使用することで確保されます。
- 領域E(RETAIN)は、IECプログラムのRETAIN修飾子で指定されたデータが格納されていますので書き替えを行わないでください。

R3 Digital Output 領域 [000001 ～ 002048]

R3 Degital Output カードが割り当てられる領域です。領域はスロット番号で固定の割り当てとなります。

アドレス	アクセス	カードスロット	内容
000001 ～ 000064	RW	1	デジタル出力データ(64ch)
000065 ～ 000128	R		チャネルステータス(64ch) 0:Deactive, 1:Active
000129 ～ 000192	RW	2	デジタル出力データ(64ch)
000193 ～ 000256	R		チャネルステータス(64ch)
000257 ～ 000320	RW	3	デジタル出力データ(64ch)
000321 ～ 000384	R		チャネルステータス(64ch)
000385 ～ 000448	RW	4	デジタル出力データ(64ch)
000449 ～ 000512	R		チャネルステータス(64ch)
000513 ～ 000576	RW	5	デジタル出力データ(64ch)
000577 ～ 000640	R		チャネルステータス(64ch)

アドレス	アクセス	カードスロット	内容
000641 ～ 000704	RW	6	デジタル出力データ(64ch)
000705 ～ 000768	R		チャネルステータス(64ch)
000769 ～ 000832	RW	7	デジタル出力データ(64ch)
000833 ～ 000896	R		チャネルステータス(64ch)
000897 ～ 000960	RW	8	デジタル出力データ(64ch)
000961 ～ 001024	R		チャネルステータス(64ch)
001025 ～ 001088	RW	9	デジタル出力データ(64ch)
001089 ～ 001152	R		チャネルステータス(64ch)
001153 ～ 001216	RW	10	デジタル出力データ(64ch)
001217 ～ 001280	R		チャネルステータス(64ch)
001281 ～ 001344	RW	11	デジタル出力データ(64ch)
001345 ～ 001408	R		チャネルステータス(64ch)
001409 ～ 001472	RW	12	デジタル出力データ(64ch)
001473 ～ 001536	R		チャネルステータス(64ch)
001537 ～ 001600	RW	13	デジタル出力データ(64ch)
001601 ～ 001664	R		チャネルステータス(64ch)
001665 ～ 001728	RW	14	デジタル出力データ(64ch)
001729 ～ 001792	R		チャネルステータス(64ch)
001793 ～ 001856	RW	15	デジタル出力データ(64ch)
001857 ～ 001920	R		チャネルステータス(64ch)
001921 ～ 001984	RW	16	デジタル出力データ(64ch)
001985 ～ 002048	R		チャネルステータス(64ch)

R3 Digital Input 領域 [100001 ～ 102048]

R3 Degital Input カードが割り当てられる領域です。領域はスロット番号で固定の割り当てとなります。

アドレス	アクセス	カードスロット	内容
100001 ～ 100064	R	1	デジタル入力データ(64ch)
100065 ～ 100128	R		チャネルステータス(64ch) 0:Deactive, 1:Active
100129 ～ 100192	R	2	デジタル入力データ(64ch)
100193 ～ 100256	R		チャネルステータス(64ch)

アドレス	アクセス	カードスロット	内容
100257 ～ 100320	R	3	デジタル入力データ(64ch)
100321 ～ 100384	R		チャネルステータス(64ch)
100385 ～ 100448	R	4	デジタル入力データ(64ch)
100449 ～ 100512	R		チャネルステータス(64ch)
100513 ～ 100576	R	5	デジタル入力データ(64ch)
100577 ～ 100640	R		チャネルステータス(64ch)
100641 ～ 100704	R	6	デジタル入力データ(64ch)
100705 ～ 100768	R		チャネルステータス(64ch)
100769 ～ 100832	R	7	デジタル入力データ(64ch)
100833 ～ 100896	R		チャネルステータス(64ch)
100897 ～ 100960	R	8	デジタル入力データ(64ch)
100961 ～ 101024	R		チャネルステータス(64ch)
101025 ～ 101088	R	9	デジタル入力データ(64ch)
101089 ～ 101152	R		チャネルステータス(64ch)
101153 ～ 101216	R	10	デジタル入力データ(64ch)
101217 ～ 101280	R		チャネルステータス(64ch)
101281 ～ 101344	R	11	デジタル入力データ(64ch)
101345 ～ 101408	R		チャネルステータス(64ch)
101409 ～ 101472	R	12	デジタル入力データ(64ch)
101473 ～ 101536	R		チャネルステータス(64ch)
101537 ～ 101600	R	13	デジタル入力データ(64ch)
101601 ～ 101664	R		チャネルステータス(64ch)
101665 ～ 101728	R	14	デジタル入力データ(64ch)
101729 ～ 101792	R		チャネルステータス(64ch)
101793 ～ 101856	R	15	デジタル入力データ(64ch)
101857 ～ 101920	R		チャネルステータス(64ch)
101921 ～ 101984	R	16	デジタル入力データ(64ch)
101985 ～ 102048	R		チャネルステータス(64ch)

R3 Analog Input 領域 [300001 ～ 300512]

R3 Analog Input カードが割り当てられる領域です。領域はスロット番号で固定の割り当てとなります。

アドレス	アクセス	カードスロット	内容
300001 ～ 300016	R	1	アナログ出力データ(16words)
300017 ～ 300032	R		ステータス(word毎)
300033 ～ 300048	R	2	アナログ出力データ(16words)
300049 ～ 300064	R		ステータス(word毎)
300065 ～ 300080	R	3	アナログ出力データ(16words)
300081 ～ 300096	R		ステータス(word毎)
300097 ～ 300112	R	4	アナログ出力データ(16words)
300113 ～ 300128	R		ステータス(word毎)
300129 ～ 300144	R	5	アナログ出力データ(16words)
300145 ～ 300160	R		ステータス(word毎)
300161 ～ 300176	R	6	アナログ出力データ(16words)
300177 ～ 300192	R		ステータス(word毎)
300193 ～ 300208	R	7	アナログ出力データ(16words)
300209 ～ 300224	R		ステータス(word毎)
300225 ～ 300240	R	8	アナログ出力データ(16words)
300241 ～ 300256	R		ステータス(word毎)
300257 ～ 300272	R	9	アナログ出力データ(16words)
300273 ～ 300288	R		ステータス(word毎)
300289 ～ 300304	R	10	アナログ出力データ(16words)
300305 ～ 300320	R		ステータス(word毎)
300321 ～ 300336	R	11	アナログ出力データ(16words)
300337 ～ 300352	R		ステータス(word毎)
300353 ～ 300368	R	12	アナログ出力データ(16words)
300369 ～ 300384	R		ステータス(word毎)
300385 ～ 300400	R	13	アナログ出力データ(16words)
300401 ～ 300416	R		ステータス(word毎)
300417 ～ 300432	R	14	アナログ出力データ(16words)
300433 ～ 300448	R		ステータス(word毎)
300449 ～ 300464	R	15	アナログ出力データ(16words)
300465 ～ 300480	R		ステータス(word毎)
300481 ～ 300496	R	16	アナログ出力データ(16words)
300497 ～ 300512	R		ステータス(word毎)

R3 Analog ステータス

ステータスの値は、0が正常、0以外が異常ありです。

1カードのデータは16ワードで、それに対応するステータスも1データ1ワードとして16ワード割り当てられています。使用するカードにより1チャネルあたりのデータワード数は異なりますので注意してください。例えば1チャネルが2ワードを使用するカードの場合のステータスは2ワードとなり、そこには同じ値が格納されています。

ステータス	記号	内容
0	NO_ERROR	正常
200	CARD_Empty	カード実装なし
201	CARD_Slot	ー
202	CARD_Addr	ー
203	CARD_TypeMismatch	カード存在するがタイプが異なる
204	CARD_Point	無効なチャネル
205	CARD_StateIsNotValid	カード異常
206	CARD_StateHasError	カード異常
207	CARD_ChHwError	ハード異常
208	CARD_ChInpError	入力異常（バーンアウト、センサーなし、など）
209	CARD_ChInpNotEnabled	チャネル入力無効
210	CARD_Error	カード異常

R3 Analog Output 領域 [400001 ～ 400512]

R3 Analog Output カードが割り当てられる領域です。領域はスロット番号で固定の割り当てとなります。

アドレス	アクセス	カードスロット	内容
400001 ～ 400016	RW	1	アナログ出力データ(16words)
400017 ～ 400032	R		ステータス(word毎)
400033 ～ 400048	RW	2	アナログ出力データ(16words)
400049 ～ 400064	R		ステータス(word毎)
400065 ～ 400080	RW	3	アナログ出力データ(16words)
400081 ～ 400096	R		ステータス(word毎)
400067 ～ 400112	RW	4	アナログ出力データ(16words)
400113 ～ 400128	R		ステータス(word毎)

アドレス	アクセス	カードスロット	内容
400129 ～ 400144	RW	5	アナログ出力データ(16words)
400145 ～ 400160	R		ステータス(word毎)
400161 ～ 400176	RW	6	アナログ出力データ(16words)
400177 ～ 400192	R		ステータス(word毎)
400193 ～ 400208	RW	7	アナログ出力データ(16words)
400209 ～ 400224	R		ステータス(word毎)
400225 ～ 400240	RW	8	アナログ出力データ(16words)
400241 ～ 400256	R		ステータス(word毎)
400257 ～ 400272	RW	9	アナログ出力データ(16words)
400273 ～ 400288	R		ステータス(word毎)
400289 ～ 400304	RW	10	アナログ出力データ(16words)
400305 ～ 400320	R		ステータス(word毎)
400321 ～ 400336	RW	11	アナログ出力データ(16words)
400337 ～ 400352	R		ステータス(word毎)
400353 ～ 400368	RW	12	アナログ出力データ(16words)
400369 ～ 400384	R		ステータス(word毎)
400385 ～ 400400	RW	13	アナログ出力データ(16words)
400401 ～ 400416	R		ステータス(word毎)
400417 ～ 400432	RW	14	アナログ出力データ(16words)
400433 ～ 400448	R		ステータス(word毎)
400449 ～ 400464	RW	15	アナログ出力データ(16words)
400465 ～ 400480	R		ステータス(word毎)
400481 ～ 400496	RW	16	アナログ出力データ(16words)
400497 ～ 400512	R		ステータス(word毎)

ステータスの詳細はここちらを参照してください。⇒R3 Analog ステータス

Modbusシステム領域 [430001 ～ 430064]

記号：R=読み取り専用, RW=読み取り書き込み共に可能

アドレス	アクセス	内容
430001	R	Controller Model ex. 1:BA3-CE10
430002	R	Controller Firmware Version XX.XX

アドレス	アクセス	内容
430003	R	LED ERROR Code
430004	R	Modbus/TCP 現在接続のクライアント数
430005	R	定数 0x0000
430006	R	定数 0xFFFF
430007	R	定数 0x1234
430008 430009	R	定数 float 1.234567
430010	RW	SYSTEM TIME (LOCAL) YEAR (YYYY) ex.2012 (このレジスタの読み込みをトリガにして現在日付時刻をこのレジスタに続く日付、時刻レジスタにコピーを行い値を固定します)
430011	RW	SYSTEM TIME (LOCAL) MONTH (MMDD) ex.12
430012	RW	SYSTEM TIME (LOCAL) DAY (DD) ex.24
430013	RW	SYSTEM TIME (LOCAL) HOUR (hh) ex.23
430014	RW	SYSTEM TIME (LOCAL) MINUTES (mm) ex.59
430015	RW	SYSTEM TIME (LOCAL) SECONDS (ss) ex.59 (このレジスタの書き込みで、事前に設定されている日付、時刻レジスタ値を元に現在の日付時刻を更新します。但し設定された値が日付時刻として範囲外の場合、現在日付時刻の更新は行われません)

1.4.コントローラ・カード

ここでは次の項目について説明しています。

状態表示LED

LED	表示色	状態	動作
RUN	緑	点灯	稼働状態
		消灯	未稼働状態
ERR	赤	点灯	異常状態またはFLASH書込み中
		点滅	異常状態（メモリエラーなどのシステムエラー状態） 10100000 : PLC アプリケーションなし 10101000 : DHCP のIPアドレス構成失敗 11110000 : その他のエラー 桁毎に1=on, 0=off を0.2秒間行い8桁出力の後 5秒間 offし、それを繰り返します。
		消灯	正常状態
READY	赤	点灯	稼働中（6秒点灯1秒消灯の繰り返し）
		消灯	停止
LOGIC	赤	点灯	ユーザプログラムRUN
		消灯	ユーザプログラムSTOP
LINK	赤	点灯	Modbus接続クライアント中
		消灯	Modbus接続クライアントなし
USR	赤	点灯	ユーザアプリケーションによる制御
		消灯	

前面ロータリースイッチ

IPアドレスの最下位値(例 192.168.1.???)は前面のロータリースイッチで設定します。IPアドレスの上位桁はプログラミングツールのデバイス設定で設定されたIPアドレスの上位桁が使用されます。

アドレス = ADR1 (x16) + ADR2 (例 ADR1=1, ADR2=2 の場合 1 x 16 + 2 = 18 となります)

アドレス	動作
00	デバイス設定で設定されたアドレスを使用
01 ～ FE	デバイス設定で設定されたアドレスの最下位値をこのスイッチ値としたアドレスを使用
FF	DHCPプロトコルによりアドレスを構成

前面スナップスイッチ

位置	動作
上	(ソフトロジック) SW2-1がOFFの場合に電源投入時のブートプロジェクトを自動起動する
中	(ソフトロジック) 電源投入時にブートプロジェクトの自動起動をしない
下	(ソフトロジック) ユーザアプリケーションの実行停止

側面DIPスイッチ

SW	機能	動作
2-1	Disable auto boot	(ソフトロジック) 電源投入時のブートプロジェクト自動起動指定 OFF: 自動起動する ON: 自動起動しない
2-2	SRAM Clear	SRAMの検査と IEC Program、データ消去の指定 OFF: 起動時にSRAMデータが不整合であれば自動で実行 (異常を検出時はERR-LEDを2秒周期で点滅、正常時は起動継続) ON: 強制的に検査とデータ消去を行い終了後に停止 (処理中はERR-LEDを点灯し、異常検出時は 100ミリ秒周期で点滅、正常時は 2秒周期で点灯)
2-3	予約	
2-4	予約	
2-5	予約	
2-6	予約	
2-7	予約	
2-8	予約	

注意)出荷時はすべてOFF位置です。

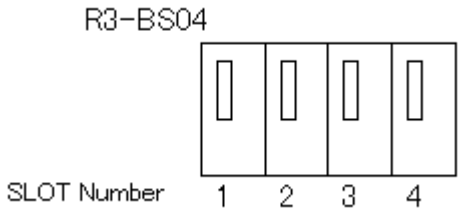
1.5.入出力カード

ここでは入出力カードに関する語句や注意事項を記述しています。

ベースとスロット

入出力カードを使用するためには入出力カードを装着するベースが必要となります。

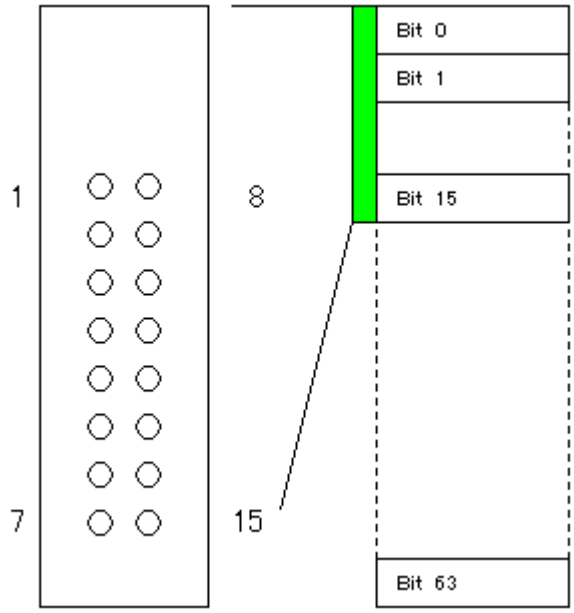
このベースでは入出力カードを装着する位置をスロット番号と呼び、このスロット番号はアドレス可変形ベースを除き左端が1番から始まり最大１６番となります。



デジタル入力(R3-DA16)、出力(R3-DC16)の場合

デジタル入力あるいは出力カードは、データを最大64点分(６４ビット)持ちます。

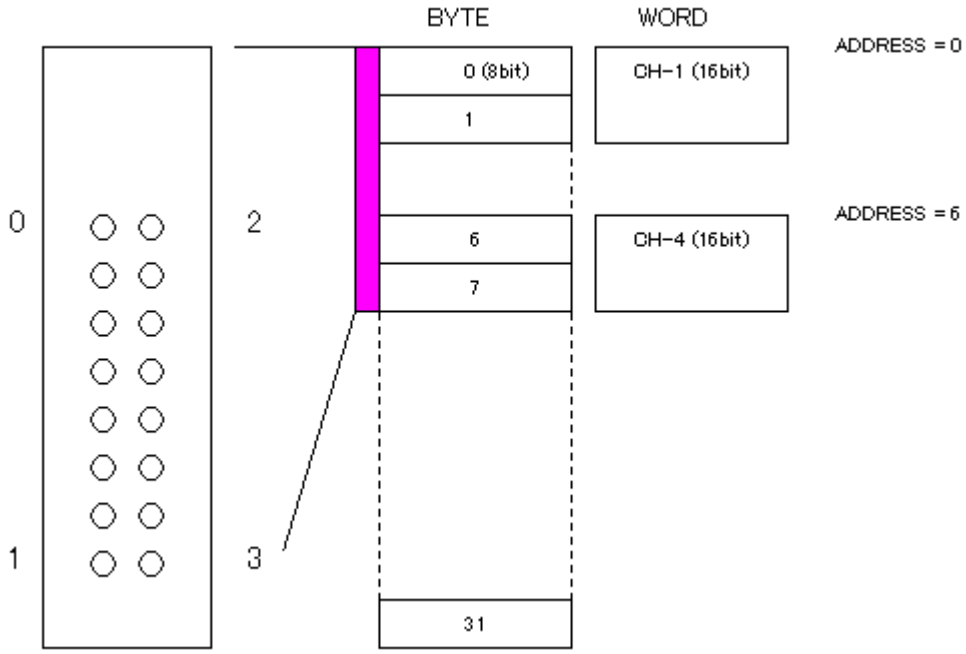
本コントローラで提供するファンクションでは、その指定をビット番号(０～６３)で指定します。



アナログ入力(R3-SV4)、出力(R3-YV4)の場合

アナログ入力あるいは出力カードは、16ビットデータを最大１６点分(32バイト)持ちます。

本コントローラで提供するファンクションでは、その指定をアドレス番号(０～３１)で指定します。



(このページは空白です)

2.IEC61131-3

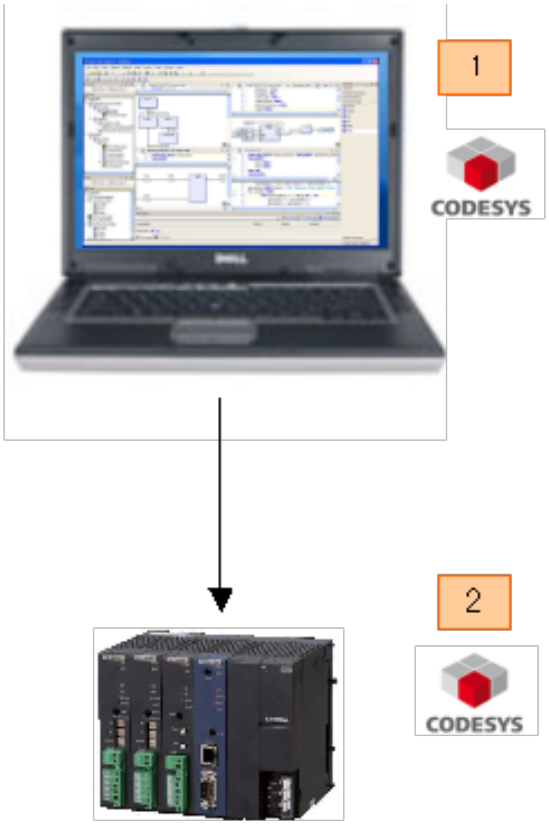
国際規格IEC61131はPLCのハードウェアからプログラミングシステムまでを包含する規格です。

1. 一般情報
2. 装置への要求事項および試験
3. プログラミング言語
4. ユーザガイドライン
5. メッセージングサービス仕様

このような構成で、IEC61131-3は、その第３部を指しています。

2.1.プログラミングツールCODESYSについて

「CODESYS」は国際規格IEC61131-3に準拠したプログラミングツール(プログラミングシステム)です。このツールではプログラミングエディタ、HMI開発環境、オンラインデバッグ機能が統合されています。



1	統合開発環境 (IDE System)	プログラミングエディタ コンパイラ HMIエディタ デバイス設定 オンラインデバッグ
2	ターゲット (Target Runtime System)	ソフトロジック フィールドバス Modbus, BACnet, LonWorks など

2.2.動作環境

CODESYS IDE (Automation Platform) の動作には次の環境を必要とします。

リソース	必要条件
OS	Windows 7/8 (32 / 64 ビット)
RAM	1GB(32ビット) または 2GB(64ビット)以上
ハードディスク空き容量	1GB 以上
画面解像度	1024 x 768 以上
CPU	1GHz以上の32ビット(x86)プロセッサまたは64ビット(x64)プロセッサ

2.3.インストール

CODESYSで開発を行うためには次のソフトウェアをインストールする必要があります。

- 開発環境はエディタなどを統合したCODESYS IDE (CODESYS IDEのインストール)
- 機種別の設定や機能をまとめて提供するパッケージ (PACKAGEのインストール)
- 別途提供されたライブラリ(LIBRARYのインストール) (必須ではありません。機種ごとに必要なライブラリはパッケージで提供されていますので、別途提供されるライブラリに関しては必要に応じてインストールしてください。)

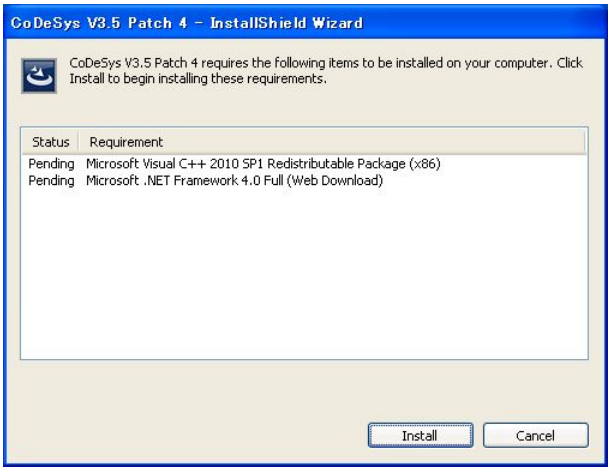
CODESYS IDE のインストール

Setup_CoDeSysV<Version>.exe を実行し、ガイダンスに従いインストールを進めます。

補 足

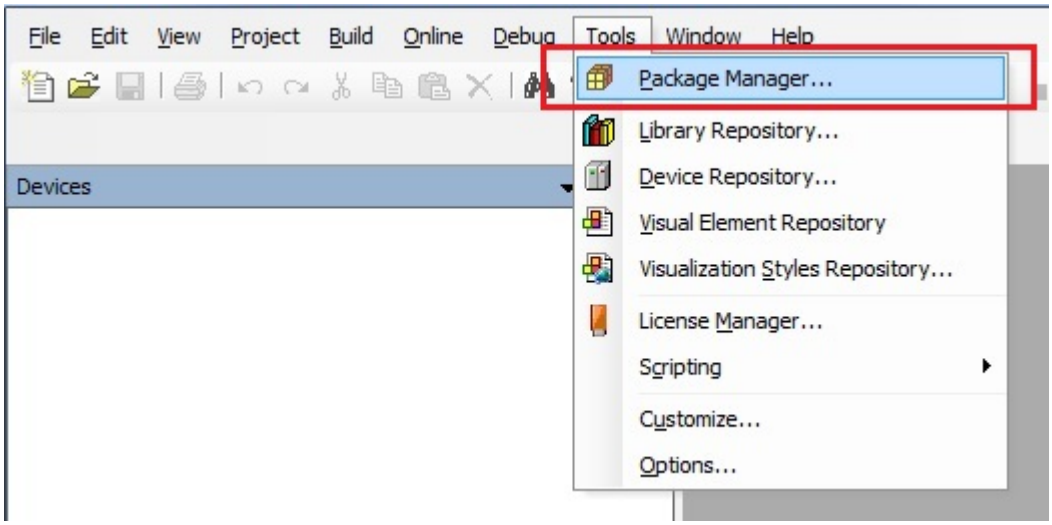
以降の画面例はCoDeSys V3.5 SPatch4 ですが、お使いのバージョンに適宜に読み替えてください。

■次のダイアログが表示される場合があります。この画面ではソフトウェアの動作に必要な追加のソフトウェアをインストールします。「Install」を押してインストールを進めてください。

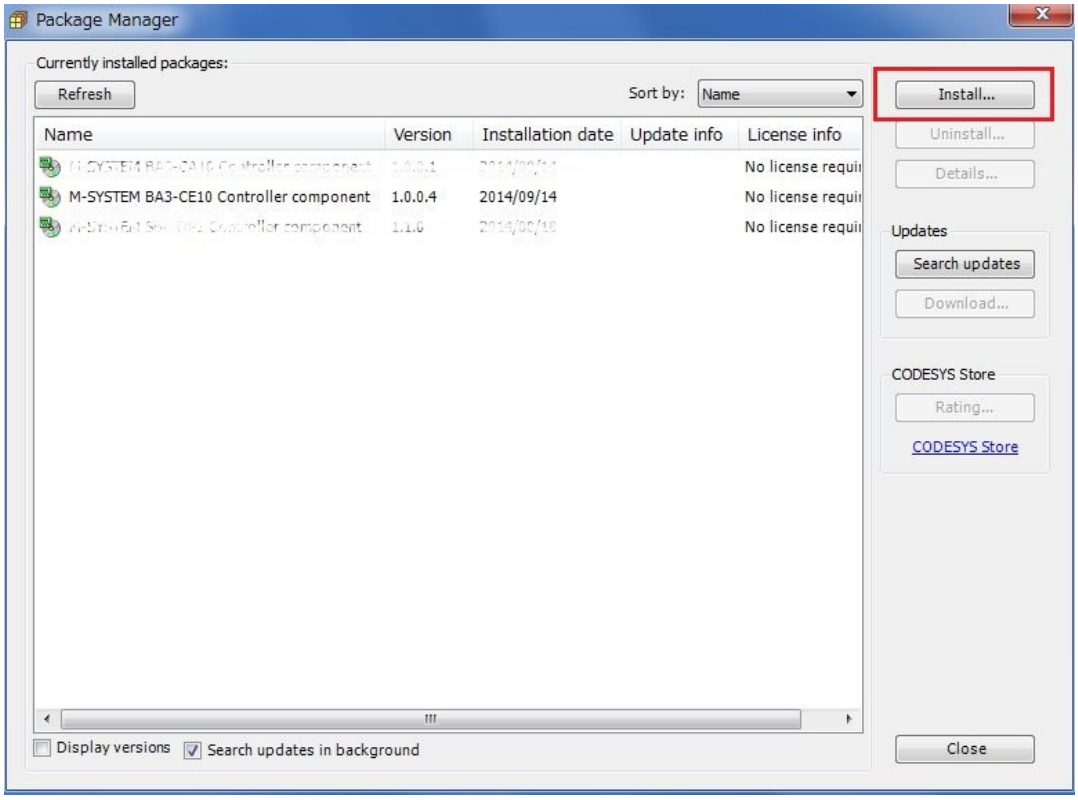


PACKAGEのインストール

CODESYS IDE のメニュー [Tools] [Package Manager...] を選択し、ガイダンスに従いインストールを進めます。



[Package Manager] ダイアログの [Install ...] ボタンを押すとファイル選択ダイアログが表示されます。インストールするパッケージ ファイルを選択してインストールを進めてください。



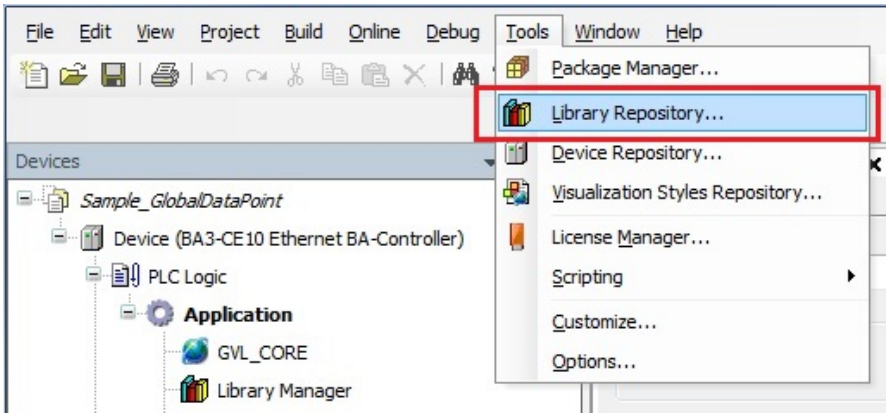
補 足

このソフトウェアには .NET 4.5 が必要です。古い状態の PC (古いバージョンのOSをお使いの場合や Windows Update の行われていない状態の場合) は complete setupを選択して必要なソフトウェアを強制的にインストールすることは可能ですが、すでにインストール済みの他のアプリケーションに悪影響を与える可能性がありますので自己責任

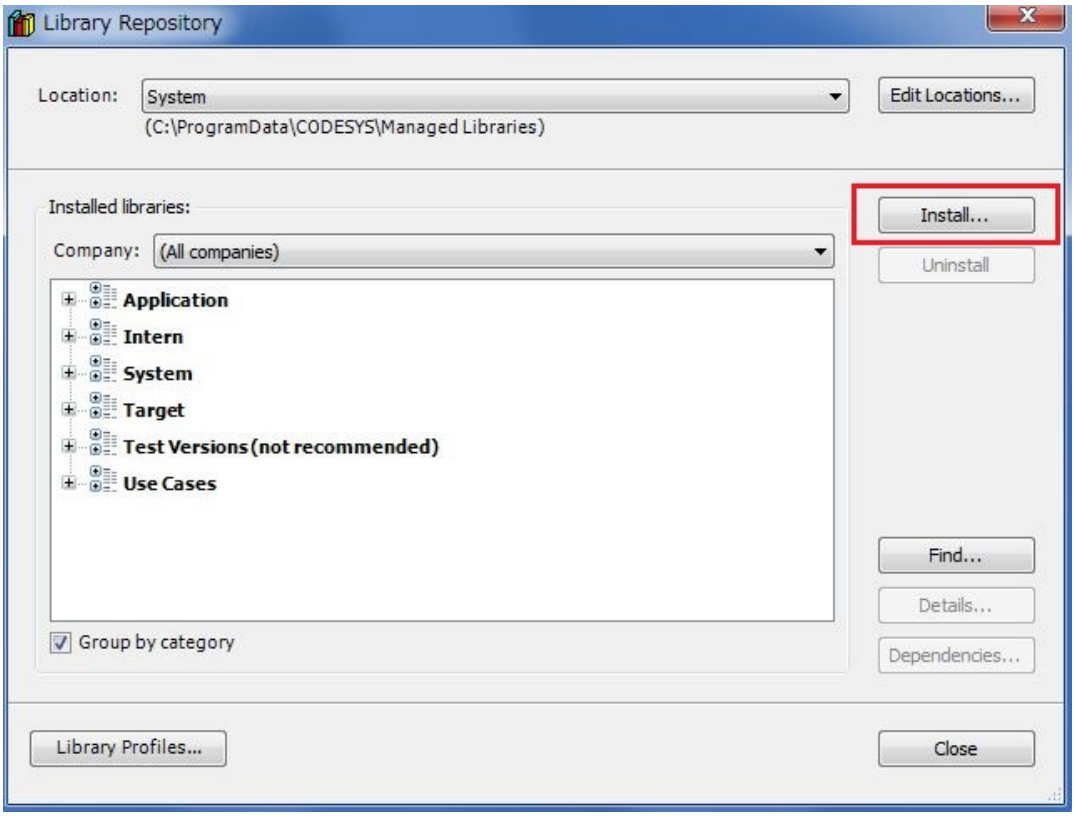
でお願いします。

LIBRARYのインストール

CODESYS IDE のメニュー [Tools] [Library Repository…] を選択し、ガイダンスに従いインストールを進めます。



[Library Repository] ダイアログの [Install …] ボタンを押すとファイル選択ダイアログが表示されます。インストールするライブラリファイルを選択してインストールを進めてください。



2.4.アンインストール

ここではCODESYS IDE 本体やCODESYS IDE にインストールされているソフトウェアのアンインストール方法を説明します。

- CODESYS IDE 本体(CODESYS IDEのアンインストール)
- 機種別の設定や機能をまとめて提供するパッケージ (PACKAGEのアンインストール)
- 別途提供されたライブラリ(LIBRARYのアンインストール)

CODESYS IDE のアンインストール

Setup_CoDeSysV<Version>.exe を実行し、インストールウィザードの最初のダイアログにあるオプション「Remove all installed features」を選択し、「Next」を押します。あるいはWindowsの「プログラムの追加と削除」からもアンインストールを行えます。

アンインストールが進むと、「M-SYSTEM BA3 Controllers component」のアンインストールを問い合わせるダイアログが表示される場合があります。その際は表示に従い「Next」を押しアンインストールしてください。

PACKAGEのアンインストール

CODESYS IDE のメニュー [Tools] [Package Manager...] を選択します。ここで表示されるダイアログの現在インストールされているパッケージ一覧からアンインストールしたいパッケージを選択し [Uninstall...] ボタンを押すと開始されます。

LIBRARYのアンインストール

CODESYS IDE のメニュー [Tools] [Library Repository...] を選択します。ここで表示されるダイアログの現在インストールされているライブラリー覧からアンインストールしたいライブラリを選択し [Uninstall...] ボタンを押すと開始されます。表示されているライブラリー覧は [Company] や [Group by category] でフィルタリングされています。目的のライブラリが見つからない場合は適切なフィルタリングを選択してください。

2.5.スタートと画面説明

ここでは次の項目について説明しています。

CODESYSの起動

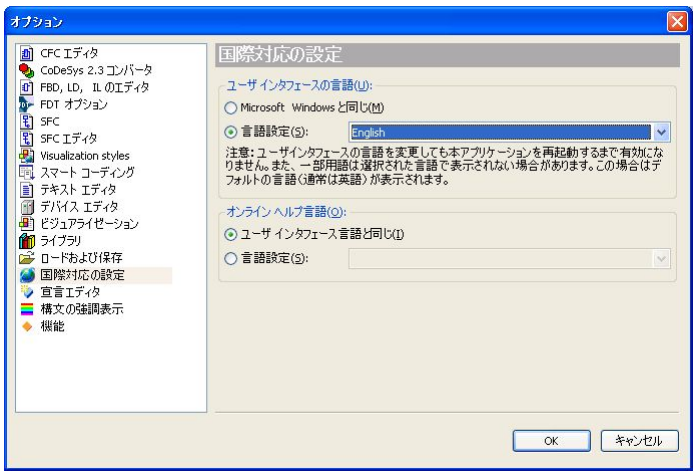
Windowsのスタートメニュー

「M-SYSTEM」→「CODESYS V3 Tools」→「CODESYS」→「CODESYS V3.5 SP4」

を選ぶか、インストール時にデスクトップに作成された以下のショートカットからも起動できます。

CODESYSを初めて起動した際の設定

環境として特殊機能が省略された「Standard」と全ての機能が表示される「Professional」が選択できます。この設定は後で変更ができるので、ここでは標準的な「Standard」を選択しておきます。



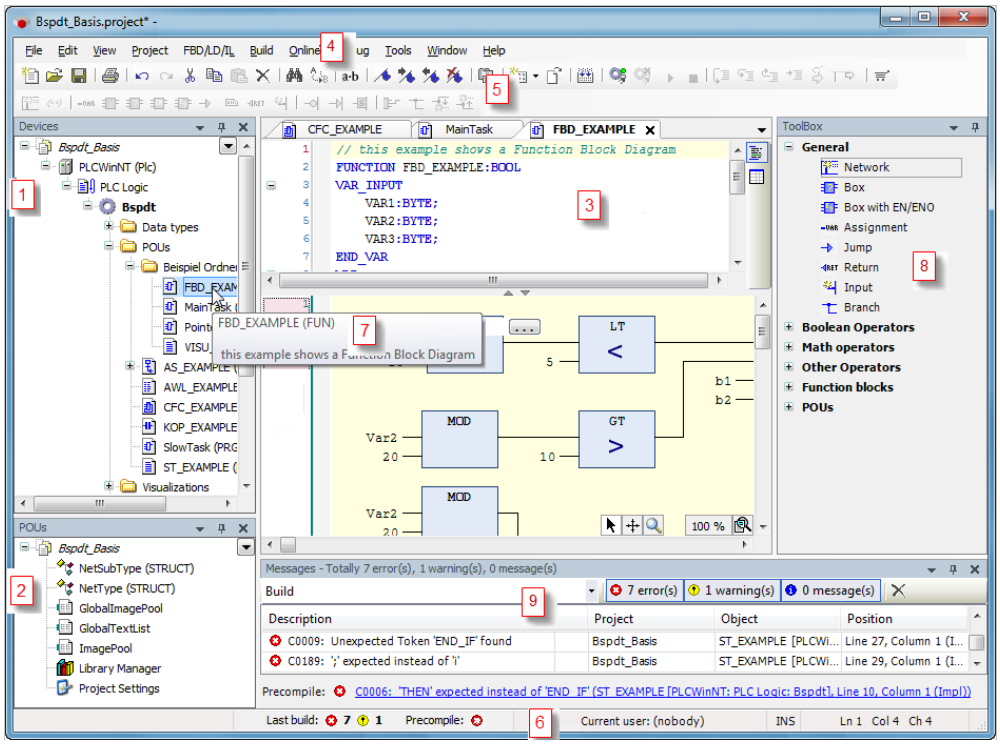
現時点の日本語表示では、ダイアログの中には日本語文字が途中までしか表示されないことや、押しボタンが表示されない場合があります。そのためユーザインタフェースの言語設定を「English」にされることをお勧めします。

変更方法は「ツール」→「オプション」→「国際対応の設定」の「ユーザインタフェースの言語」

言語設定「English」を選択します。

CODESYS再起動でユーザインタフェースの表示が「英語」に替わります。

CODESYSの画面構成



1	デバイス ツリー
2	POUツリー
3	エディタ ウィンドウ
4	メニュー バー
5	ツール バー
6	ステータス行
7	コメントのツールチップ表示
8	ツール ボックス
9	メッセージ ウィンドウ

Deviceビュー(Deviceツリー)

デバイス(コントローラ)で定義されている全てのリソースが表示されます。

代表的な表示内容:

「プロジェクト」: 例 MySample

「デバイス」: 例 Device

「リソース」: 例 PLC Logic

「アプリケーション」: 例 Application

- グローバル変数変数定義 例 GVL_PLC
- 使用ライブラリ定義
- POU(プログラム、ファンクションブロック、ファンクション) 例 PLC_PRG
- DUT(構造体、列挙など)
- タスク定義

プログラムウィンドウの構成(「宣言部」、「命令(ボディ)部」)

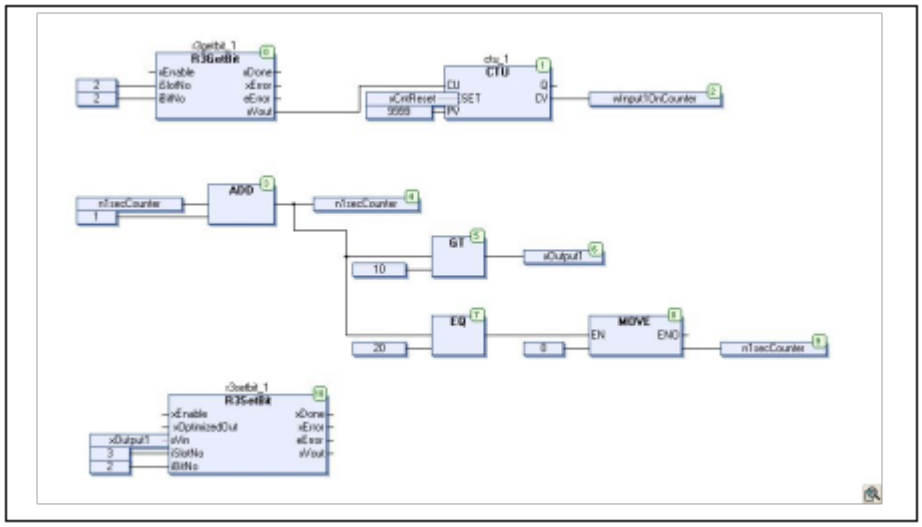
プログラムを作成する際に使用します。プログラムの作成では処理を記述する「ボディ部」とプログラム内で使用される変数を宣言する「宣言部」という2つのウィンドウを使用して行います。

「宣言部」の例

```
VAR
    r3getbit_1          : R3GetBit;
    ctu_1               : CTU;
    xCntReset           : BOOL;
    wInput1OnCounter    : WORD;
    nlsecCounter        : INT;
    xOutput1            : BOOL;
    r3setbit_1          : R3SetBit;
END_VAR
```

「ボディ部」の例

■CFC言語



■ST言語

```
(* Network 1 *)
r3getbit_1(iSlotNo:=2, iBitNo:=1);
ctu_1(CU:=r3getbit_1.xVout, RESET:=xCntReset, PV:=9999, CV=>wInput1OnCounter);
(* Network 2 *)
nlsecCounter := nlsecCounter + 1;
(* Network 3 *)
IF nlsecCounter > 10 THEN
    xOutput1 := TRUE;
ELSE
    xOutput1 := FALSE;
END_IF
(* Network 4 *)
IF nlsecCounter = 20 THEN
    nlsecCounter := 0;
END_IF
(* Network 5 *)
// r3setbit_1(iSlotNo:=3, iBitNo:=1, xVin:=xOutput1);
r3setbit_1.iSlotNo := 3;            // Slot:3, Ch:2
r3setbit_1.iBitNo := 1;
r3setbit_1.xVin:=xOutput1;
r3setbit_1();
```

オンライン モード情報

オンラインモードの情報は、画面の最下位に配置されていますステータスバーに表示されます。

表示される情報は:

[RUN] ：プログラム実行中

[STOP] ：プログラム停止中

[HALT ON BP] ：プログラムがブレイクポイントで停止中

Program loaded	：プログラムはデバイスにロード済み
Program unchanged	：デバイス内のプログラムはプログラムツールのものと一致しています
Program modified(Online Change)	：デバイス内のプログラムはプログラミングツールと異なるためオンライン変更が必要です
Program modified(Full download)	：デバイス内のプログラムはプログラミングツールと異なるため完全なダウンロードが必要です

3.プログラミング言語

IEC61131-3では5つのプログラミング言語の定義、表記と要素を規定しています。

ここでは、次のプログラミング言語について説明します。

- コンティニアス ファンクション チャート「グラフィック」(言語としてはFBD)

  CFC (Continuous Function Chart)

- ファンクション ブロック ダイアグラム「グラフィック」

  FBD (Function Block Diagram)

- 命令リスト「テキスト」

  IL (Instruction List)

- ラダー ダイアグラム「グラフィック」

  LD (Ladder Logic Diagram)

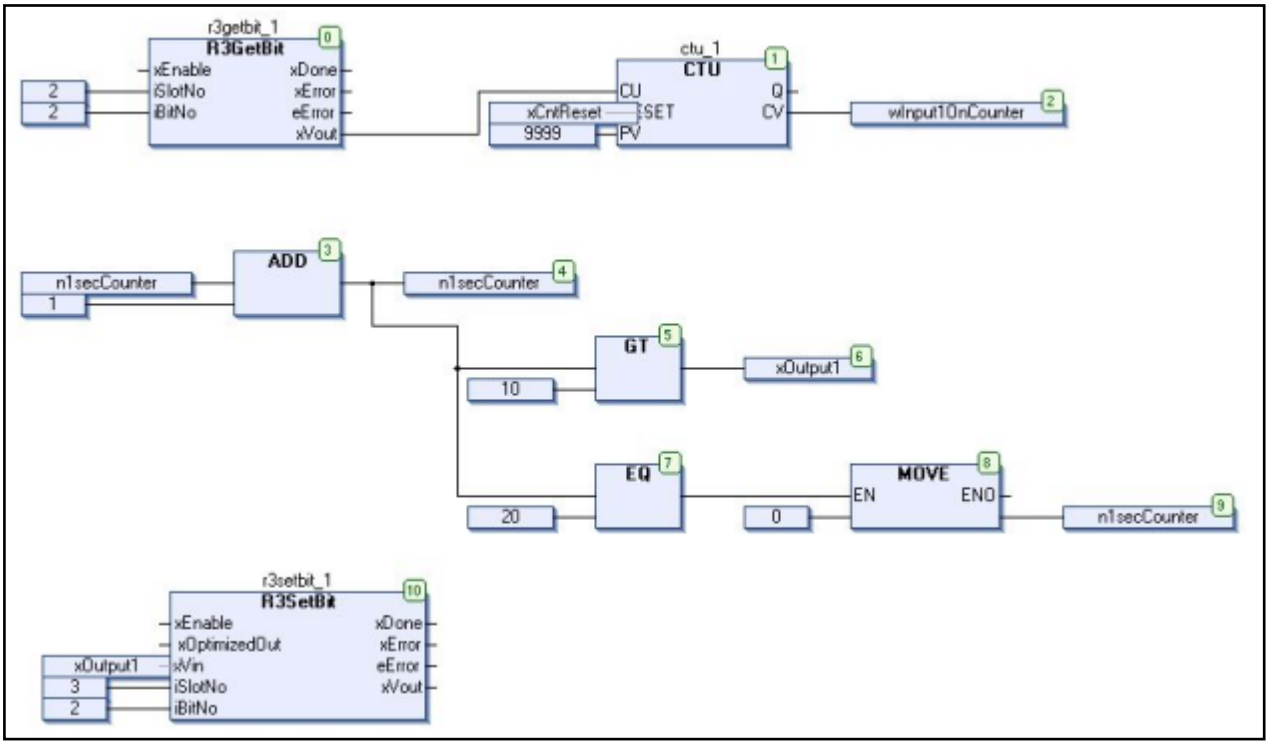
- シーケンシャル ファンクション チャート「グラフィック」

  SFC (Sequential Function Chart)

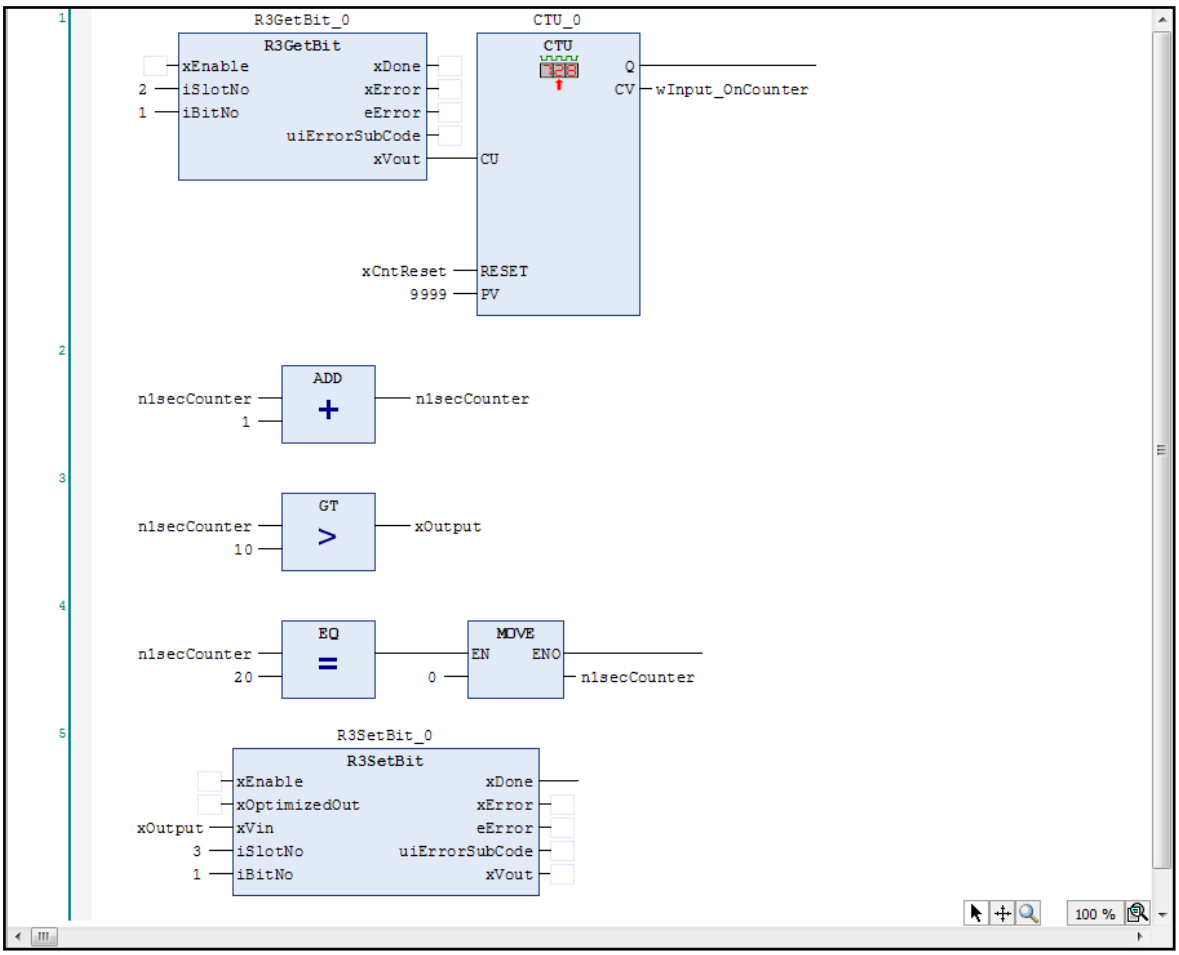
- 構造化テキスト「テキスト」

  ST (Structured Text)

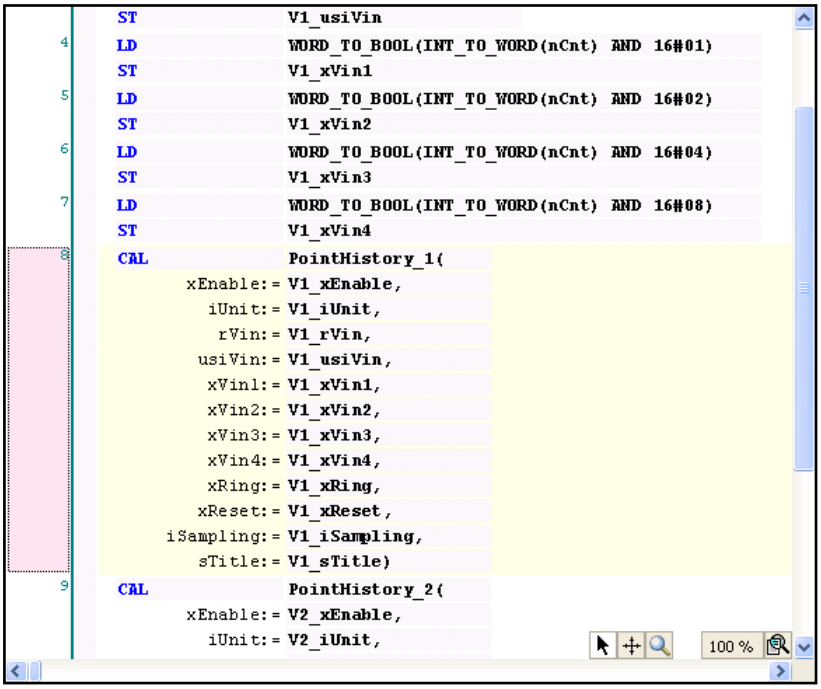
3.1.CFC (Continuous Function Chart)



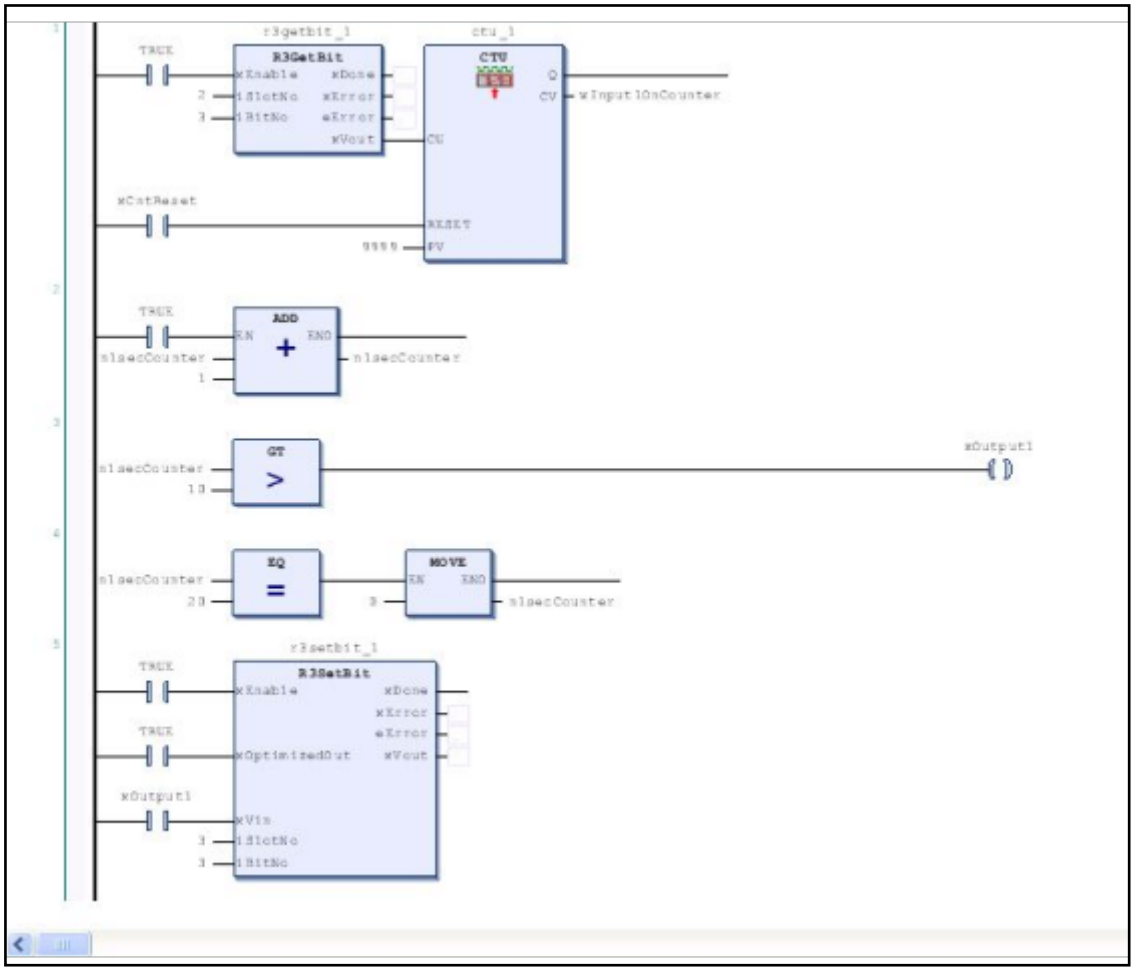
3.2.FBD (Function Block Diagram)



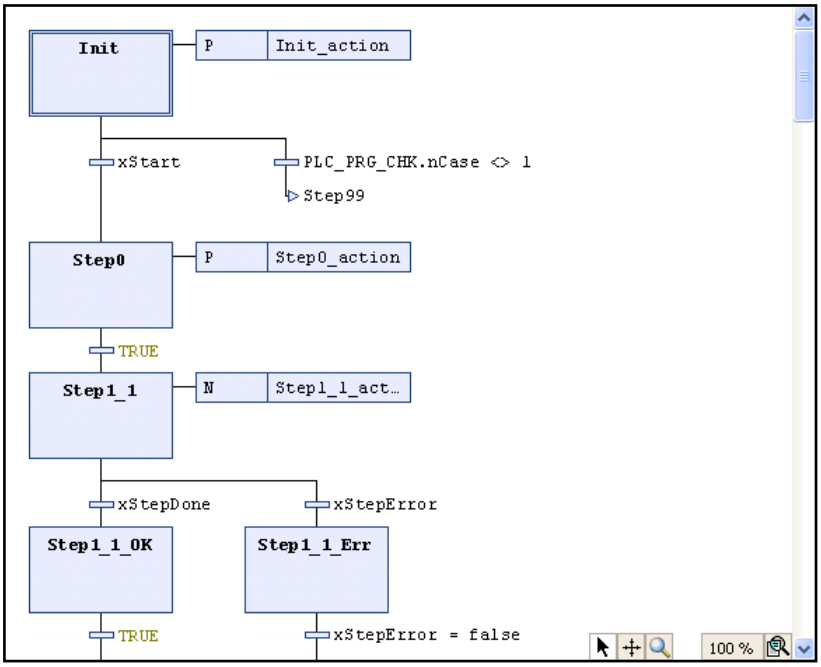
3.3.IL (Instruction List)



3.4.LD (Ladder Logic Diagram)



3.5.SFC (Sequential Function Chart)



3.6.ST (Structured Text)

```
(* Network 1 *)
r3getbit_1(iSlotNo:=2, iBitNo:=1);
ctu_1(CU:=r3getbit_1.xVout, RESET:=xCntReset, PV:=9999, CV=>wInput1OnCounter);
(* Network 2 *)
n1secCounter := n1secCounter + 1;
(* Network 3 *)
IF n1secCounter > 10 THEN
    x0utput1 := TRUE;
ELSE
    x0utput1 := FALSE;
END_IF
(* Network 4 *)
IF n1secCounter = 20 THEN
    n1secCounter := 0;
END_IF
(* Network 5 *)
// r3setbit_1(iSlotNo:=3, iBitNo:=1, xVin:=x0utput1);
r3setbit_1.iSlotNo := 3;            // Slot:3, Ch:2
r3setbit_1.iBitNo := 1;
r3setbit_1.xVin:=x0utput1;
r3setbit_1();
```

(このページは空白です)

4.プログラミング要素

ここでは、プログラミングに必要となる要素について説明します。

4.1.POU (Program Organization Unit)

プログラム構成ユニットと訳されるプログラムの最小単位を指します。これには次の3種類があります。

名称	記号	キーワード
ファンクション	FUN (Function)	FUNCTION
ファンクションブロック	FB (Function Block)	FUNCTION_BLOCK
プログラム	PROG (Program)	PROGRAM

ファンクションと**ファンクションブロック**との主な違いは、**ファンクション**では全ての入力パラメータが等しければ返される結果は常に同じとなります(内部状態の記憶なし)。これに対して**ファンクションブロック**では内部状態を記憶(インスタンス化)するので呼び出し毎に返す結果を変化させることができます。また、**プログラム**はユーザプログラム(アプリケーション)の最上位に位置する唯一のインスタンスを持つファンクションブロックのようなもので、実行権(タスクからの呼び出し)を与えることができます。

POUの典型的な例(プログラミングツールでは個別のウィンドウで表示されます)

```
FUNCTION_BLOCK  MyFB
    VAR_INPUT
        bEN  :  BOOL;
        bReset  : BOOL;
        nCounterLimit  :  INT;
    END_VAR
    VAR_OUTPUT
```

```
        bERROR  :  BOOL;
        nCurrent  :  INT;
    END_VAR

    IF bEN=FALSE THEN
        RETURN;
    END_IF
    IF bReset = TRUE THEN
        nCurrent := 0;
    END_IF
    IF nCurrent <= nCounterLimit THEN
        nCurrent := nCurrent + 1;
    END_IF
END_FUNCTION_BLOCK
```

4.2.FUN (Function)

全ての入力パラメータが等しければ返される結果は常に同じとなります(内部状態の記憶なし)。

種類	可否
入力パラメータ	Yes
出力パラメータ	No
入出力パラメータ	No *1
ファンクションの戻り値	Yes
ローカル変数と出力変数の保持	No

*1): 拡張機能(CODESYS では使用可能)

4.3.FB (Function Block)

内部状態を記憶(インスタンス化)するので呼び出し毎に返す結果を変化させることができます。

種類	可否
入力パラメータ	Yes
出力パラメータ	Yes
入出力パラメータ	Yes
ファンクションの戻り値	No
ローカル変数と出力変数の保持	Yes

4.4.DUT (Data Unit Type)

データユニットタイプはユーザ定義の新しいデータ型を定義できます。基本型や既に定義されたユーザ型から新しいユーザ型を定義することができます。

種類	キーワード
配列	<Array_Name> : ARRAY [<ll1>..<ul1>,<ll2>..<ul2>] OF <elem. Type>
構造体	TYPE <structurename>: STRUCT <declaration of variables 1> ... <declaration of variables n> END_STRUCT END_TYPE
共用体	TYPE <structurename>: UNION <declaration of variables 1> ... <declaration of variables n> END_UNION END_TYPE
列挙	TYPE <identifier> : (<enum_0> ,<enum_1>, ...,<enum_n>) |<base data type>; END_TYPE
参照	<identifier> : REFERENCE TO <data type>
ポインタ	<identifier>: POINTER TO <object>
範囲型	TYPE <name> : <Int type> (<ug>..<og>) END_TYPE

4.5.VAR (Variable)

変数の種類と属性を以下に示します。

種類	キーワード
入力パラメータ	VAR_INPUT
出力パラメータ	VAR_OUTPUT
入出力パラメータ	VAR_IN_OUT
ローカル	VAR
グローバル	VAR_GLOBAL
CONSTANT	VAR CONSTANT / VAR_GLOBAL CONSTANT
RETAIN(不揮発指定)	VAR RETAIN
PERSISTENT(永続指定)	VAR_GLOBAL PERSISTENT RETAIN
静的変数	VAR_STAT
一時変数	VAR_TEMP

4.6.Direct I/O (Variable)

入出力カードやメモリに直接(絶対番地)変数を割り当てる方法があります。

種類	記号	宣言例
入力	%I	xInput1 : BOOL AT %IX0.0; bInput2 : BYTE AT %IB1; wInput3 : WORD AT %IW2;
出力	%Q	xOutput1 : BOOL AT %QX0.0; bOutput2 : BYTE AT %QB1; wOutput3 : WORD AT %QW2;
メモリ	%M	xFlag1 : BOOL AT %MX0.0; bFlag2 : BYTE AT %MB1; wFlag3 : WORD AT %MW2;

補足

- **弊社コントローラでは特別な場合を除いてこの割り当て方法(Direct I/O)を使用する必要はありません。**
- 弊社コントローラでは、入出力カードやメモリにアクセスする専用命令が用意されています。それら専用命令を使用すると複数バイトのデータアクセスをより安全に行うことができます。

4.7.リテラル

数値などの常数を直接表記する場合はリテラルを用います。リテラルは、数字、文字列、時刻の表記に必要です。

また、デバックなどで値を入力するときにも、リテラル同様の表記を使用する必要があります。

リテラルは次のように表記します。

[データ型 #][データ]

- 数字リテラル
- 文字列リテラル
- 持続リテラル
- 日付時刻リテラル

数字リテラル

使用できる数字リテラルを次の表に示します:

型	例
整数リテラル	-12 0 123_456 +986
実数リテラル	-12.0 0.0 0.4560 3.14159_26
指数付実数リテラル	-1.34E-12 -1.34e-12 1.0E+6
2 進リテラル	INT#2#1111_1111
8 進 リテラル	INT#8#377
16 進リテラル	INT#16#FF SINT#16#ff
ブール FALSE と TRUE	FALSE TRUE
ブール 0 と 1	0

型	例
	1

INT やBOOL だけでなく、変数ワークシートで使用するリテラルも、キーワード(データ型)なしで使用できます。

例：

INT#16#FF なら 16#ff を使えます。

BOOL#FALSE なら FALSE を使えます。

文字列リテラル

文字列リテラルは、2つの単引用符で囲まれたゼロまたはそれ以上の連続する文字です。WSTRING (UNICODE) 文字列リテラルでは、2つの2重引用符で囲みます。

型	例
空文字列	''
空白付文字列	' '
空でない文字列	'Hello'
空でないWSTRING文字列	"こんにちは"

持続リテラル

持続時間データは、時間、分、秒、ミリ秒とその組合せで表せます。

型	例
短い接頭語	T#14ms t#14ms t#12m18s3.5ms T#25h_15m t#25h_15m
長い接頭語	TIME#14ms time#14ms TIME#25h_15m time#25h_15m

日付時刻リテラル

型	例
日付	DATE#1996-01-24 date#1996-01-24 D#1996-01-24 d#1996-01-24
時刻	TIME_OF_DAY#15:36:55.36 time_of_day#15:36:55.36 TOD#15:36:55.36 tod#15:36:55.36
日付と時刻	DATE_AND_TIME#1996-01-24-15:36:55.36 date_and_time#1996-01-24-15:36:55.36 DT#1996-01-24-15:36:55.36 dt#1996-01-24-15:36:55.36

4.8.データ型

ここでは次の項目について説明しています。

基本データ型

型	説明	データ範囲
BOOL	ブール	TRUE, FALSE
SINT	単精度（8ビット）整数	-128 ～ +127
USINT	符号なし単精度（8ビット）整数	0 ～ +255
INT	（16ビット）整数	-32,768 ～ +32,767
UINT	符号なし（16ビット）整数	0 ～ +65,535
DINT	倍精度（32ビット）整数	-2,147,483,648 ～ +2,147,483,647
UDINT	符号なし倍精度（32ビット）整数	0 ～ +4294967295
LINT	64ビット整数	-9,223,372,036,854,775,808 ～9,223,372,036,854,775,807
ULINT	符号なし64ビット整数	0 ～18,446,744,073,709,551,615
TIME	持続時間	4,294,976,295ms ～ 4,294,976,295s
BYTE	長さ8のビット列（バイト）	16#00 ～ 16#FF
WORD	長さ16のビット列（ワード）	16#0000 ～ 16#FFFF
DWORD	長さ32のビット列（ダブルワード）	16#00000000 ～ 16#FFFFFFFF
LWORD	長さ64のビット列（クワッドワード）	16#0000000000000000 ～ 16#FFFFFFFFFFFFFFFF
REAL	単精度実数（32ビット浮動小数） (IEEE 754)	±0.0000001 ～ ±9,999,999 有効桁数7桁 (*1)
LREAL	倍精度実数（64ビット浮動小数）	±0.000000000000001 ～ ±999,999,999,999,999

型	説明	データ範囲
	(IEEE 754)	有効桁数15桁 (*1)
STRING	文字列	最大255文字　(初期80文字)
Arrays	配列	
Structures	構造体	
Union	共用体	

*1: 演算結果は誤差(桁落ち、小数点位置の違う2値の演算による有効桁数による情報落ち)が発生することに留意する必要があります。
例えば、REAL型の次の演算では 1.1 + 2.2 = 3.3 を期待するが結果は 3.3000002 となります。

Arrays:配列

配列は要素となるデータ型の1～3次元までの配列を提供します。この配列はPOUやグローバル変数リストの中の宣言部で定義できます。

構文:

<配列名> : ARRAY [<ll1>..<ul1>, <ll2>..<ul2>, <ll3>..<ul3>] OF <要素データ型>
ll1, ll2, ll3 は各次元の下限値で、ul1, ul2, ul3 は各次元の上限値を整数で指定します。

例:

```
Card_game: ARRAY [1..13, 1..4] OF INT;
```

配列を初期化する例:

```
arr1 : ARRAY [1..5] OF INT := [1,2,3,4,5];
arr2 : ARRAY [1..2,3..4] OF INT := [1,3(7)];   (* short for 1,7,7,7 *)
arr3 : ARRAY [1..2,2..3,3..4] OF INT := [2(0),4(4),2,3];   (* short
for 0,0,4,4,4,4,2,3 *)
```

構造体の配列を初期化する例:

```
Structure definition:

    TYPE STRUCT1
    STRUCT
```

```
            p1:int;
            p2:int;
            p3:dword;

        END_STRUCT
        END_TYPE

    Array initialization:

        ARRAY[1..3] OF STRUCT1:= [(p1:=1,p2:=10,p3:=4723),
        (p1:=2,p2:=0,p3:=299),(p1:=14,p2:=5,p3:=112)];
```

配列の部分的な初期化の例：

```
    arr1 : ARRAY [1..10] OF INT := [1,2];
```

Structures：構造体

構造体はプロジェクト内にDUT オブジェクトとして定義できます。

構文：

TYPE <structurename>:
STRUCT

<変数１の宣言>
…
<変数ｎの宣言>

END_STRUCT
END_TYPE

構造体名 Polygonline の初期化の例：

```
    TYPE Polygonline:
    STRUCT
    Start:ARRAY [1..2] OF INT;

        Point1:ARRAY [1..2] OF INT;
        Point2:ARRAY [1..2] OF INT;
        Point3:ARRAY [1..2] OF INT;
```

```
        Point4:ARRAY [1..2] OF INT;
        End:ARRAY [1..2] OF INT;
    END_STRUCT
    END_TYPE
```

構造体の初期化の例：

```
    Poly_1:polygonline := ( Start:=[3,3], Point1:=[5,2], Point2:=[7,3],
    Point3:=[8,5], Point4:=[5,7], End:= [3,5]);
```

構造体メンバーのアクセス:

次の構文を使用して構造体のメンバーにアクセスできます:

<構造体名>.<メンバー名>

次は構造体`polygonline` のメンバー `start` をアクセスする例です。

```
    Poly_1.Start
```

構造体内でのBit アクセス

データ型 BIT は構造体内の宣言にだけ使用できます。これは構造体における1ビットのメモリ空間を消費する名前の付いた単一ビットを宣言できます。

TYPE <構造体名>:
STRUCT

<BIT名 bit1> : BIT;
<BIT名 bit2> : BIT;
<BIT名 bit3> : BIT;
...
<BIT名 bitn> : BIT;

END_STRUCT
END_TYPE

次の構文を使用することで構造体のBITメンバーにアクセスできます:

<構造体名>.<BIT名>

補 足

BIT変数のポインタ指定と参照指定は使用できません。また、BIT変数の配列は許されていません。

5.プログラミング

5.1.オンラインコマンドと保持変数

オンラインコマンド実行後の各変数の状態を以下に示します。

×＝値が保持されます －＝値は初期化されます

オンライン コマンド	オンラインコマンド実行後の状態					
	VAR	VAR RETAIN	VAR PERSISTENT	Application	Boot Application	Downloaded Source
Download	－	－	×	Download	×	×
Online Change	×	×	×	Update	×	×
STOP	×	×	×	×	×	×
Reboot PAC	－	×	×	Loading boot application	×	×
Reset warm	－	×	×	×	×	×
Reset cold	－	－	×	×	×	×
Reset origin	－	－	－	－	－	×

5.2.アプリケーションの構成

ここでは次の項目についての説明を記述します。

テンプレートを使用してアプリケーションの作成

作成されたアプリケーションはプロジェクトとして管理され、最終的にはコントローラに転送して実行されます。

アプリケーションは様々なプログラミングスタイル(制御方法)が考えられますが、弊社コントローラで実行するアプリケーションは1つのタスクで実行されることを推奨します。

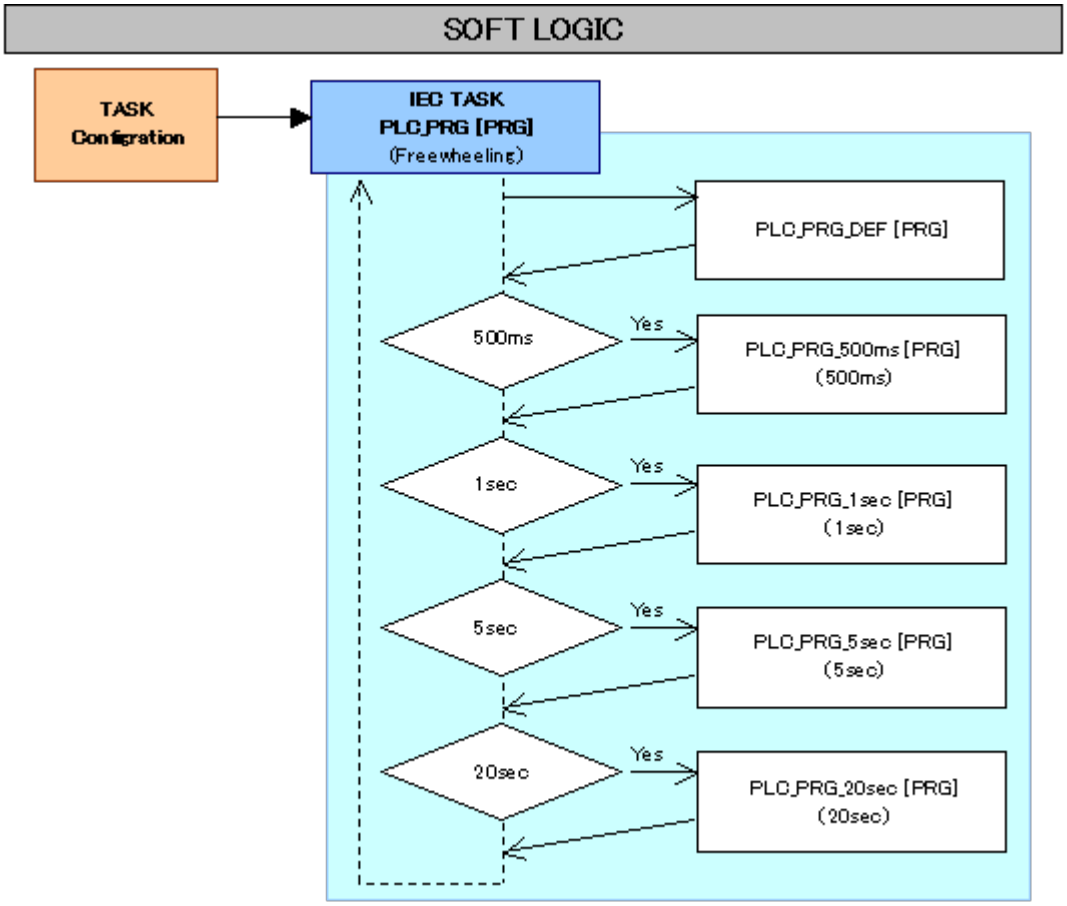
これはグローバル変数や入出力カードへの同時書き込みや指示の競合を防ぐ一つの手法です。

そのため、弊社コントローラ用プロジェクトの新規作成時には下記に示しますプログラム構成をテンプレートとして用意しています。

このテンプレートで作成されるアプリケーションはPLC_PRGタスクで制御され各サブルーチン(PLC_PRG_～)を順次呼び出して、最後は呼び出し元のPLC_PRGに戻り、これを繰り返します。

PLC_PRGは定周期で呼び出されるのではなく常時繰り返し実行(Freewheeling)されています。

ユーザはこのテンプレートで作成される構造のPLC_PRG_DEFを始めとするPLC_PRG_～ の5つのサブルーチンからプログラムを記述していきます。



IECタスク	サブルーチン	周期	備考
PLC_PRG		最速	システム予約処理
	PLC_PRG_DEF	最速	最速実行ユーザ処理
	PLC_PRG_500ms	500ms	高速実行ユーザ処理
	PLC_PRG_1sec	1Sec	中速実行ユーザ処理
	PLC_PRG_5sec	5Sec	低速実行ユーザ処理
	PLC_PRG_20sec	20Sec	最低速実行ユーザ処理

アプリケーションの実行周期

IECタスクのPLC_PRGの実行周期(スキャンタイム)の最大値は、各サブルーチンが必要とする処理時間の合計となります。テンプレートでは安定した動作を確認するために、ウォッチドッグを設定し最大実行許容時間を超過しないかどうかを監視させています。もし、設定されている最大実行許容時間を超過した場合のアプリケーションは「ウォッチドッグ異常」として停止されます。

各サブルーチンの実行を表で表すと次のようになります。

×＝実行される －＝実行されない

タイミング	PLC_PRG_DEF	PLC_PRG_500ms	PLC_PRG_1sec	PLC_PRG_5sec	PLC_PRG_20sec
下記以外	×	－	－	－	－
500ms	×	×	－	－	－
1sec	×	×	×	－	－
5sec	×	×	×	×	－
20sec	×	×	×	×	×

ブートアプリケーションの起動

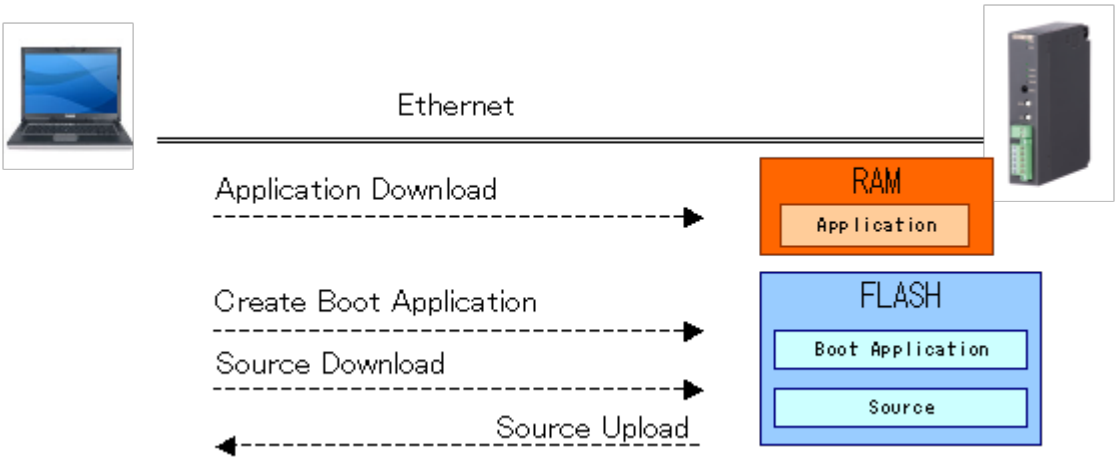
電源投入またはシステムのリブートを行うとコントローラが起動します。

ユーザ作成のアプリケーションがブートアプリケーションとして登録されていると、そのアプリケーションがファイルからメモリへ読み込まれ自動的に実行されます。

5.3.プログラミングツールとの接続

プログラミングツールがインストールされたPCとは、本コントローラの Ethernet ポートで接続します。

プログラミングツールで接続する前に、ping コマンド等により通信が可能かどうか事前に確認されることをお勧めします。



5.4.コントローラを使用する前に行っていただきたいこと

コントローラによりRTC (リアルタイムクロック)機能を搭載している機種があります。該当の機種を**「はじめて使用する」**場合や**「長期間未稼働のものを使用再開する」**場合など、日付時刻情報が正確ではない可能性がある場合には、使用開始前に再度日付時刻の設定を行うことをお勧めします。

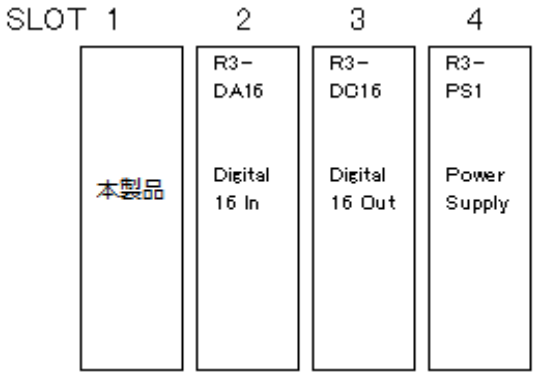
6.サンプルプログラムの作成

本章ではサンプルプログラムの作成を通してプログラミング方法からデバッグまで一連の操作を説明します。ここでは次の動作をおこなうプログラムを作成していきます。

■課題

- デジタル入力に入力されたONの回数をカウントする。
- デジタル出力を20秒周期でON／OFFする(10秒間OFF→10秒間ON→繰り返し)。

■機器構成



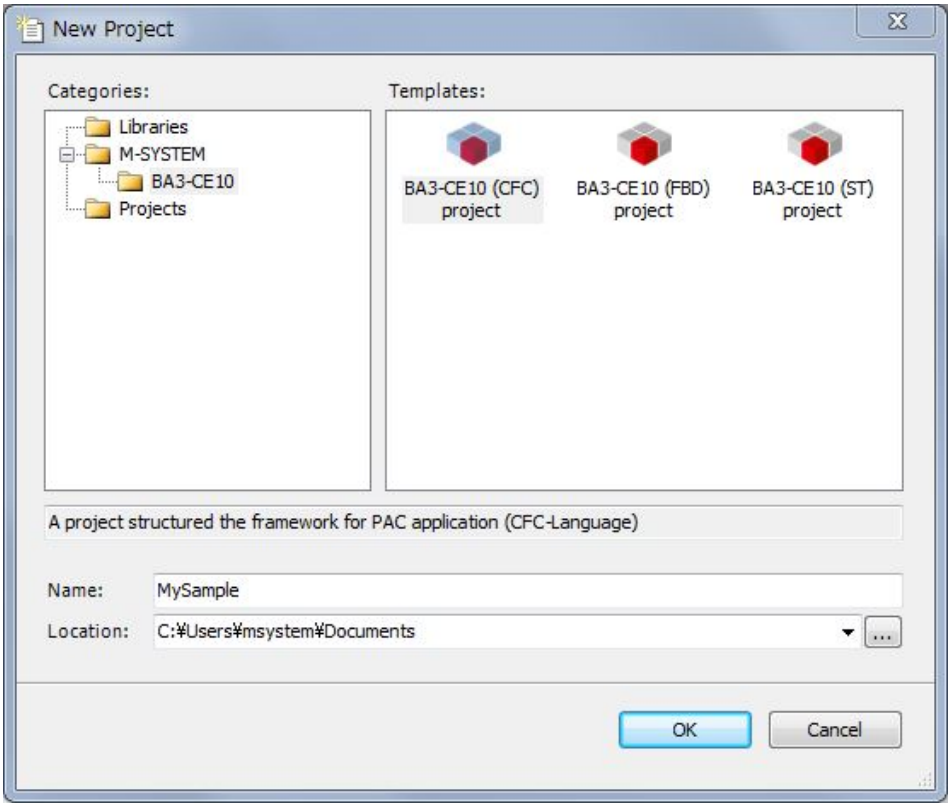
1	BA3-CE10	Controller Card
2	R3-DA16	Digital input 16channels
3	R3-DC16	Digital Output 16channels
4	R3-PS1	Power Supply

6.1.プログラミング手順

画面キャプチャの形式表示が「BA3-CE10」ですが、ご使用の機器の形式に読み替えて読み進めてください。

新規プロジェクトの作成

操作:「File」→「New Project…」



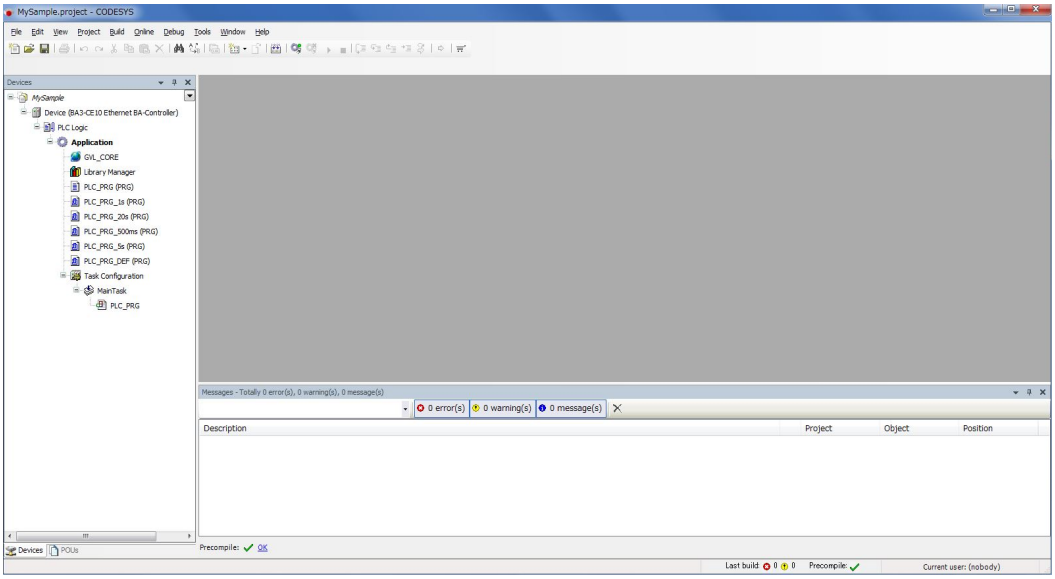
Templates: 使用するコントローラに対応したテンプレートを選択します。

例: コントローラ BA3-CE10 の場合は [M-SYSTEM] [BA3-CE10] [BA3-CE10 (CFC) project] を選択します。

Name: プロジェクトの名称を入力します。

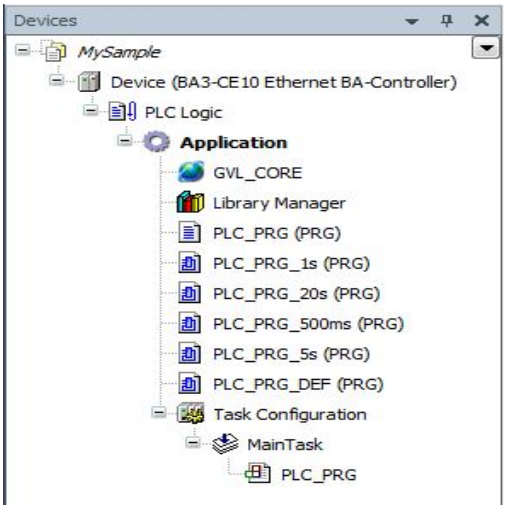
例: MySample

Location: 作成されるプロジェクトファイルを格納するためのフォルダを指定します。



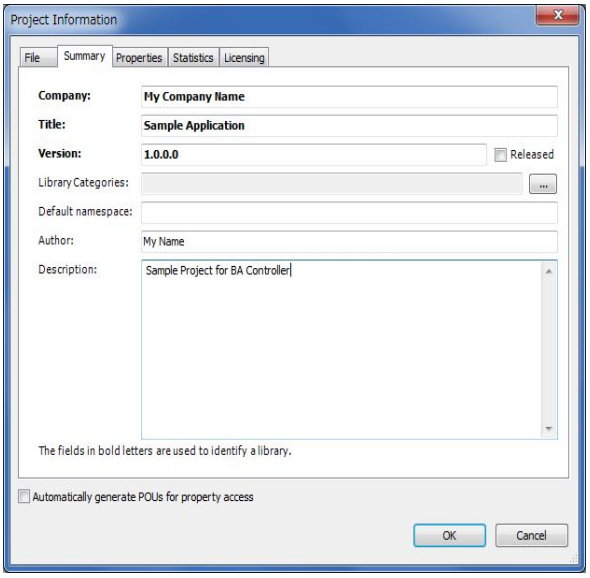
プロジェクトが作成され各ウィンドウが表示されます。

「Deviceツリー」では次のような表示が確認できます。



プロジェクト情報の設定

操作:「Project」→「Project Information...」



ここではプロジェクトに作成者などの情報を記録しておきます。

Company:作成者の会社名など 例：My Company Name

Title:タイトル 例：Sample Application

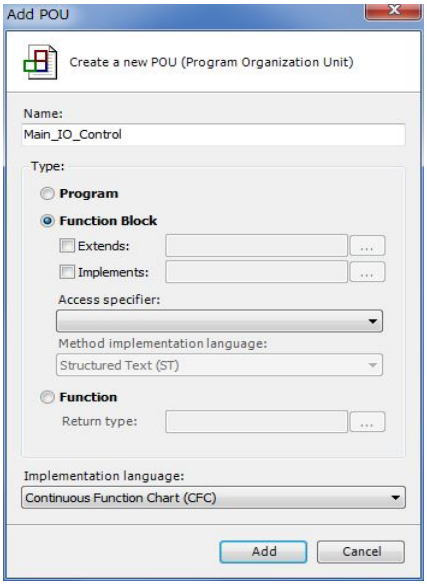
Version:バージョン 例：1.0.0.0

Author:作成者 例：My Name

Description:説明 例：Sample Project for BA controller

ファンクションブロック「Main_IO_Control」の作成(POUの追加)

操作:デバイスツリー内のApplicationを選択した状態で「Project」→「Add Object」→「POU...」



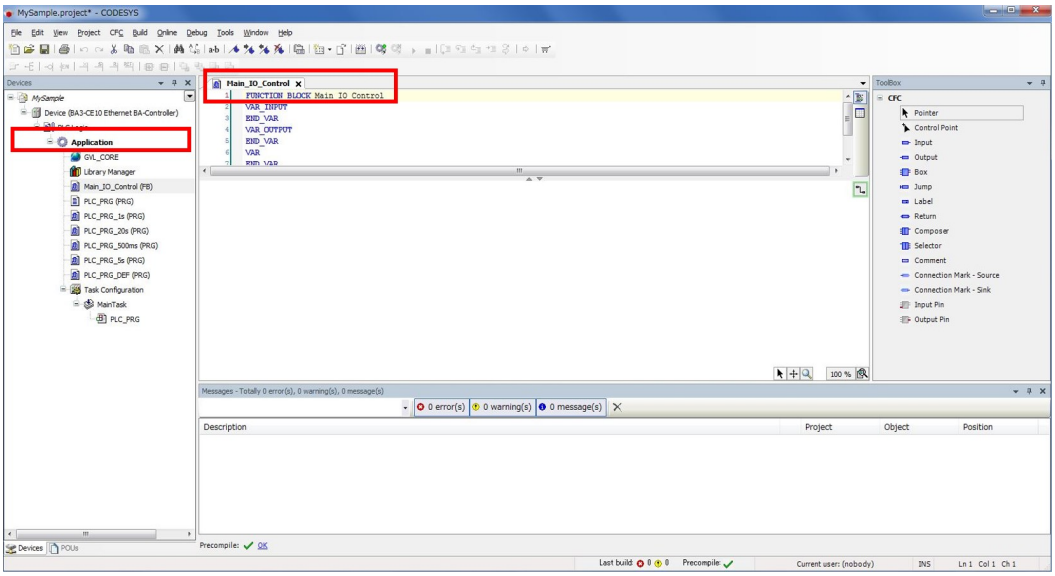
ここでは

Name: “Main_IO_Control”

Type: [Function Block] を選択

Implementation Language: [Continuous Function Chart (CFC)] を選択

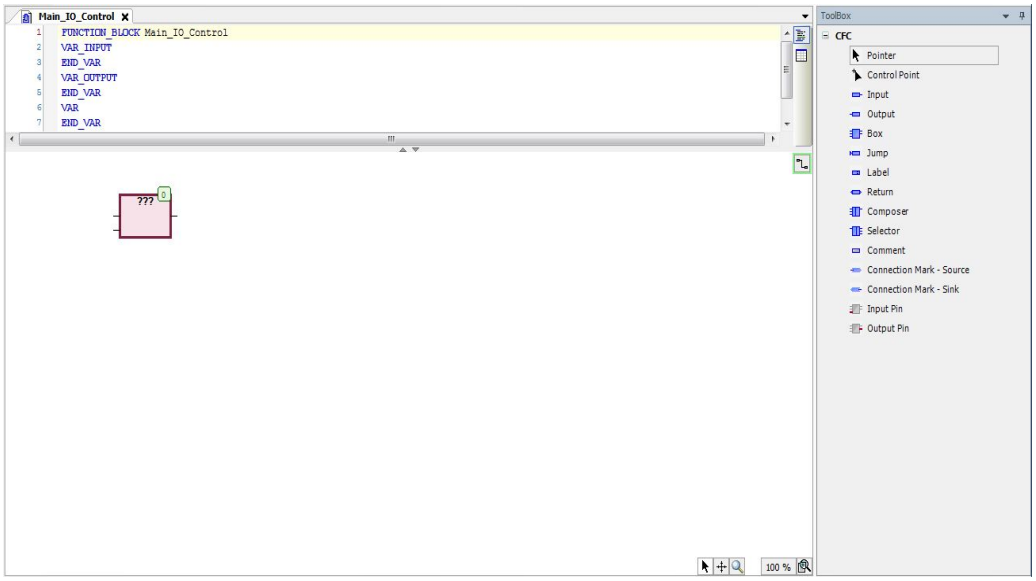
[Add] ボタンを押します。



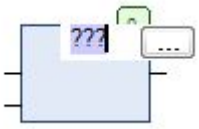
新しいファンクションブロック [Main_IO_Control] が作成され、画面中央ではそのファンクションブロックのタブが追加され「宣言部」、「ボディ部」の各ウィンドウが開いた状態になります。

CFC言語での既存ファンクションブロックの配置

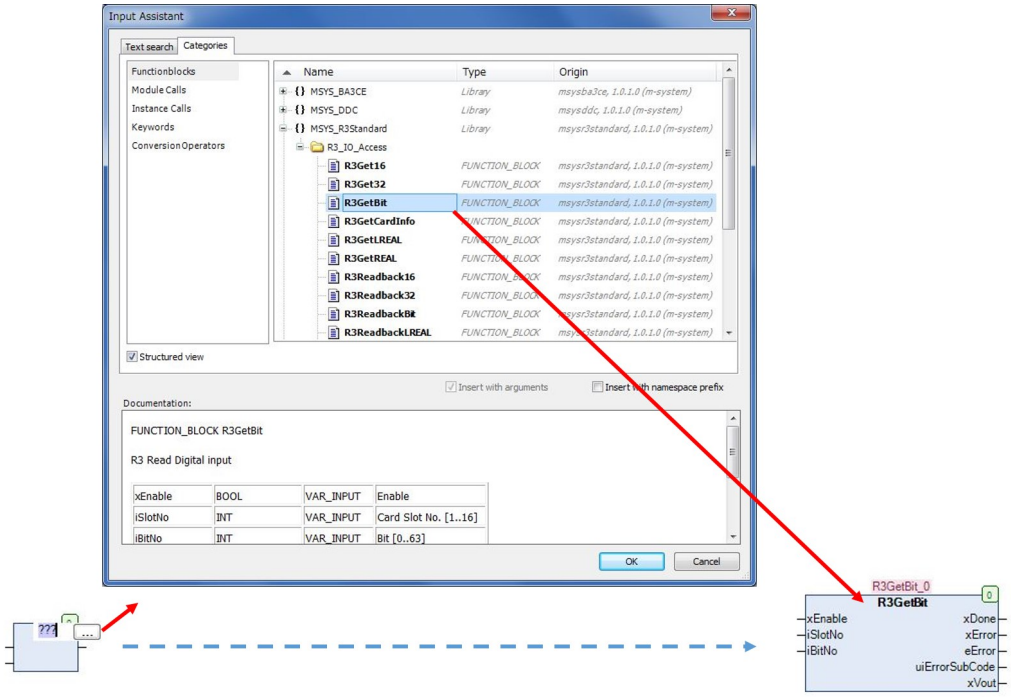
操作：ツールボックスの「Box」を選択した後、命令を配置したい「ボディー部」の場所をクリックする



ここでは右のツールボックスの[Box]を選択して左の「ボディ部」ウィンドウをクリックするとその場所に命令が配置されます。

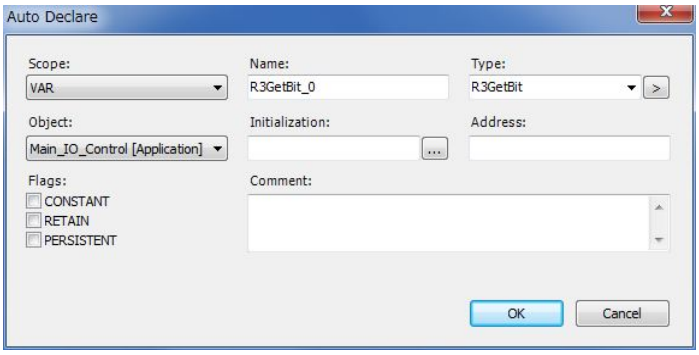


無名"???"のボックスが配置されるので、"???"をクリックして希望する「ファンクションブロック名」を入力(ここでは"???"をクリックすることで表示される「…」ボタンを押し入力アシスタントで表示される一覧から”R3GetBit”を選択)します。ボックスの上には希望する「インスタンス名」を入力(ここでは入力アシスタントから戻ると自動的に”R3GetBit_0”が表示されている)します。



ボックス上部のインスタンス名”R3GetBit_0”を入力後に確定([Enter]キー)させると、その変数(インスタンス)が未宣言であることから、「入力アシスタント」ダイアログが表示され変数宣言を手助けします。

(この未宣言変数の「入力アシスタント」自動表示は、オプションの指定により抑止できます)



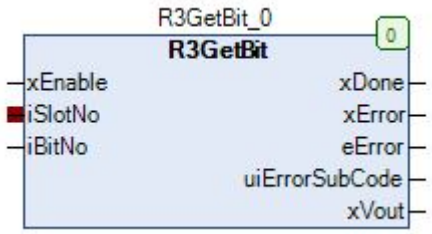
ここで「OK」を押すと、自動的に「宣言部」に変数宣言が追加されます。

続いて次の操作を行います。

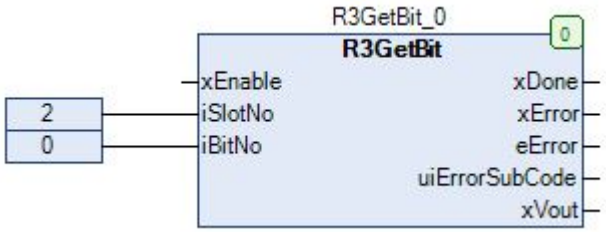
xEenable左の“???”は必要ないので選択後「DEL」キーで削除し確定([Enter]キー)します。

iSlotNo左の”???”は2に置き換えます(下記の図のようにiSlotNo左のピンをクリックし選択状態にして数値を入力します)。

iBitNo左の”???”は0に置き換えます。

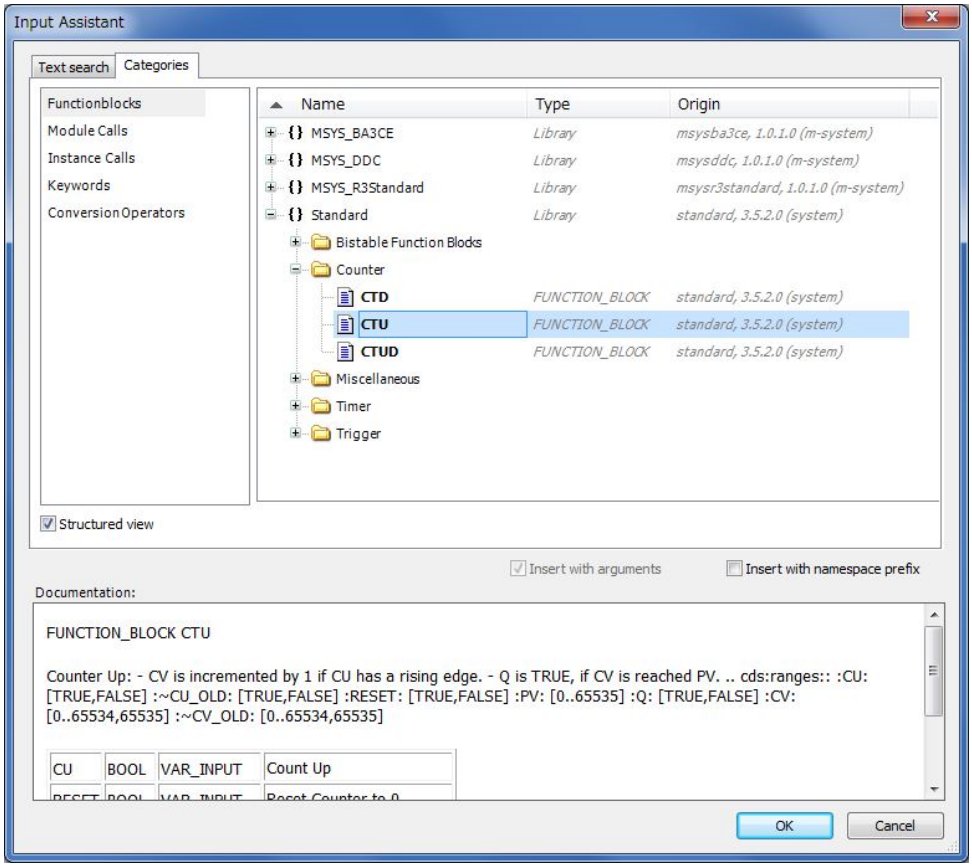


入力後は次の表示になります。



次にR3GetBit_0の出力xVoutにアップカウンタの入力を接続し入力のON回数を数えさせます。

まず新しいBOXを配置します。



ボックス内のファンクションブロック名の入力ではアップカウンタ CTU を選択します。

アップカウンタのインスタンス名を “CTU_0” とします。

次に"R3GetBit_0"のxVoutと"CTU_0"のCUを接続します。操作はxVoutの右をクリックしたままマウスをCUの左まで移動し放すとその間が線で接続されます。

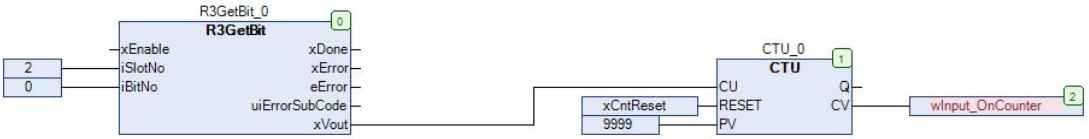
アップカウンタ “CTU_0” のパラメータを次のように設定します。

入力RESETに新たなBOOL型の変数 “xCntReset”

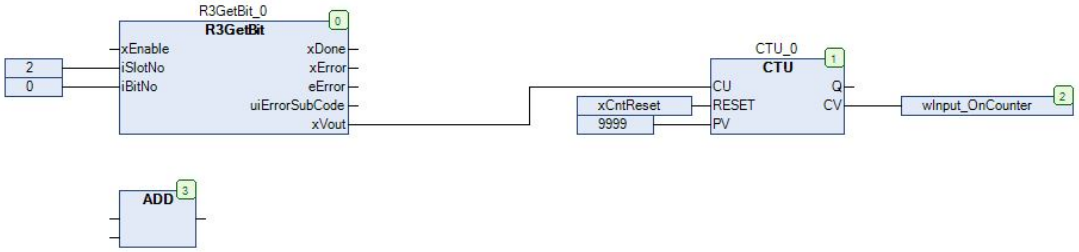
入力PVに数値リテラル “9999”

出力CVに新たなWORD型の変数 “wInput_OnCounter”

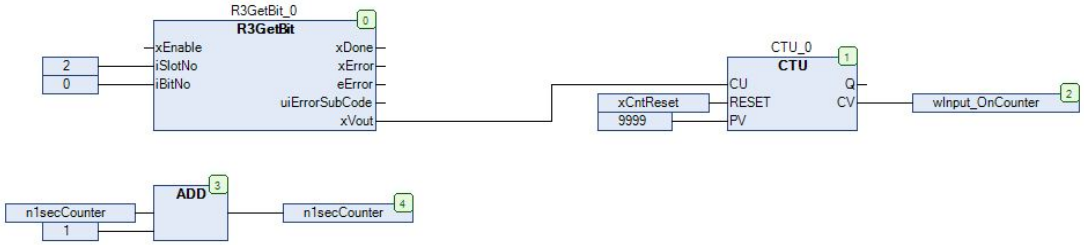
ここまでのプログラムは以下のようになります。



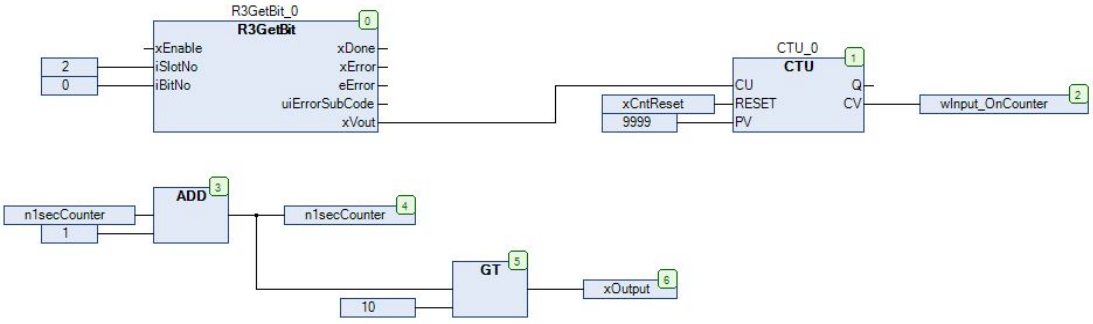
次に20秒周期でON/OFFするためにカウンタを作成します。ここでのカウンタは加算での積算で実現させるため「ADD」命令を配置します。



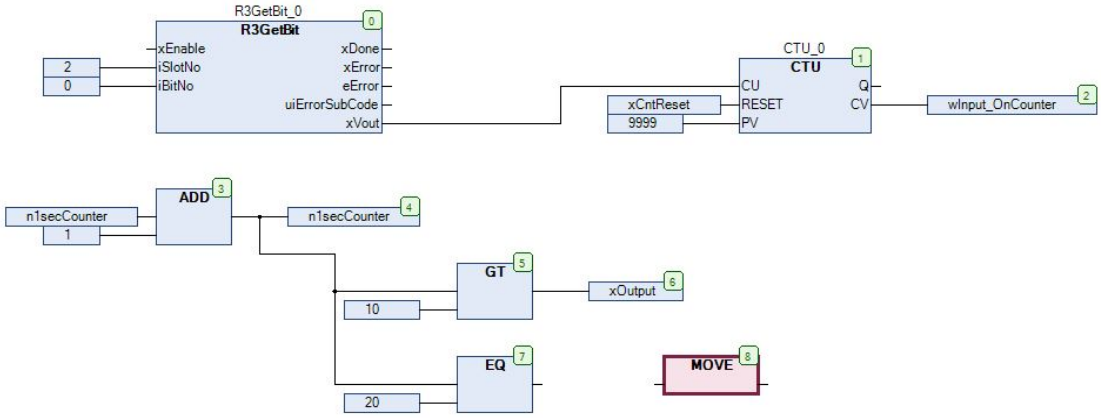
加算値のリテラル値"1"と積算値を格納するINT型の変数 “n1secCounter” を「ADD」命令に割り当てます。



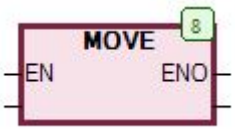
続いてn1SecCounter > 10 の時BOOL変数のxOutputをTRUEとする演算を追加します。命令は比較命令の「GT」を使用します。



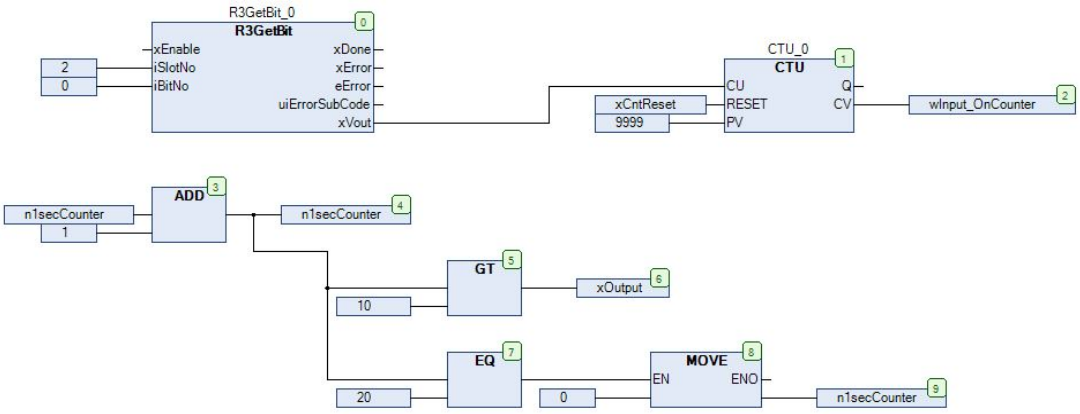
次にn1SecCounterが20と等しい時にn1SecCounterを0にリセットする演算を追加します。命令は比較命令「EQ」と代入命令「MOVE」を使用します。



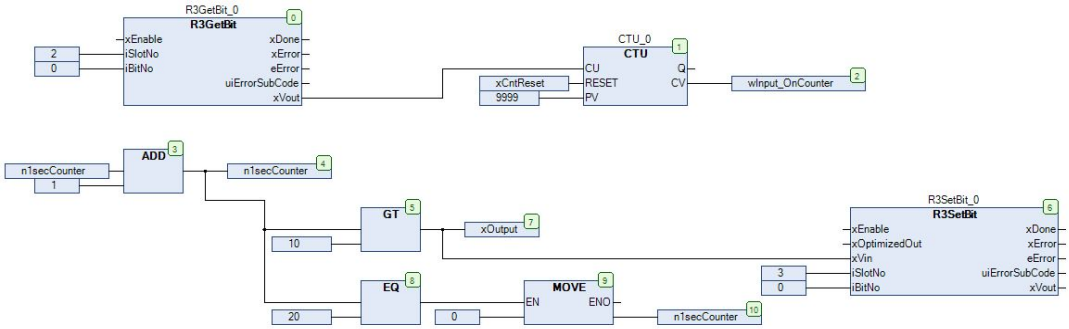
ここで使用する代入命令「MOVE」は常に演算されると不都合なのでEN/ENOで演算の実行を制御します。画面の「MOVE」を選択した状態でメニュー「CFC」の「EN/ENO」を選択してEN/ENO付の命令に変更します。



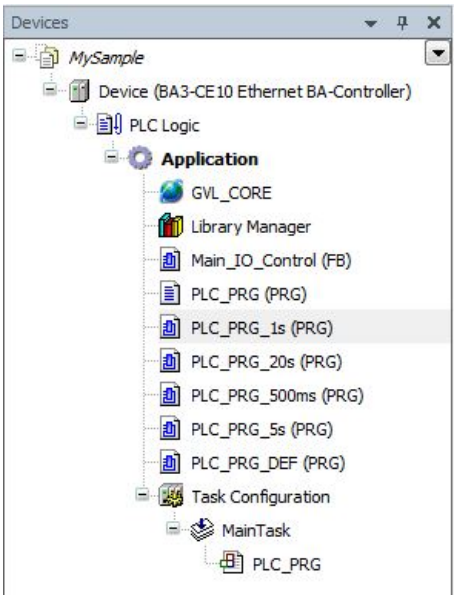
この「MOVE」命令に代入する値"0"と代入される変数"n1SecCounter"を割り付けます。



次に変数 xOutput の値をデジタル出力カードに出力します。ファンクションブロックの “R3SetBit” を使用しスロット番号3(iSlotNo=3)のチャネル番号1(iBitNo=0)に出力します。

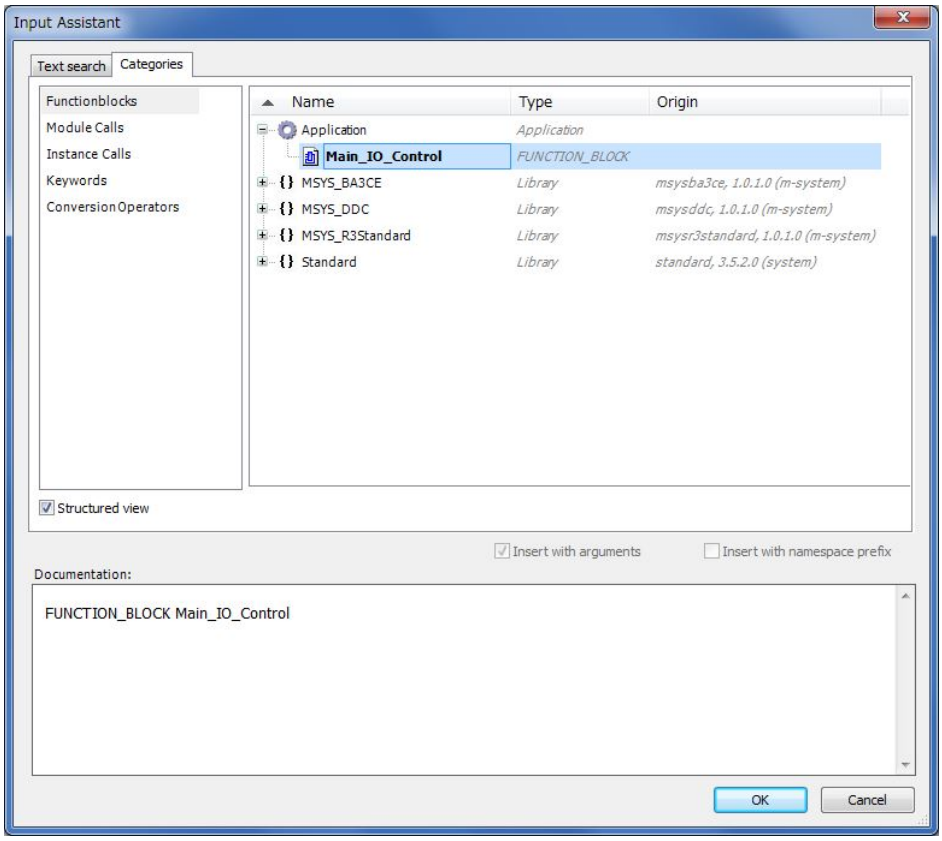


ここで作成したファンクションブロック “Main_IO_Control” の実行には、プログラムから呼び出す必要があります。ここでは1秒周期で処理しているプログラム“PLC_PRG_1s”から呼び出されるようにします。



Deviceツリーから “PLC_PRG_1s” をマウスでダブルクリックします。

呼び出しするために新たに「BOX」を配置します。入力アシスタントを利用して、先ほど作成したMain_IO_Controlファンクションブロックを選択します。



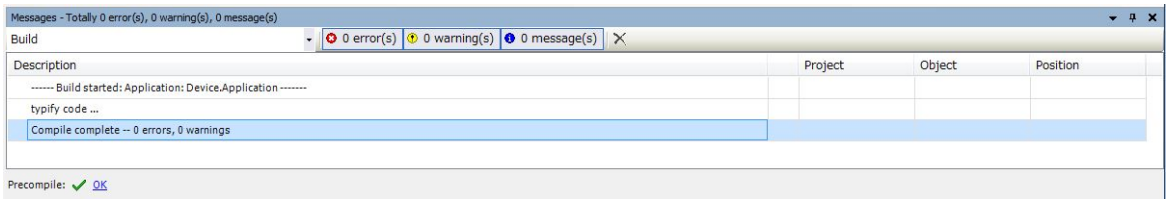
このインスタンス名は"Main_IO_Control_0"を使用します。

これでプログラムの作成は終了しました。

次はプログラムのエラーチェックとコントローラに転送するイメージを作成するためにコンパイルを行います。

コンパイル

操作:「Build」から「Build」を選択



正常にコンパイルが終わると上記のように0 errors, 0 warningsと表示されます。

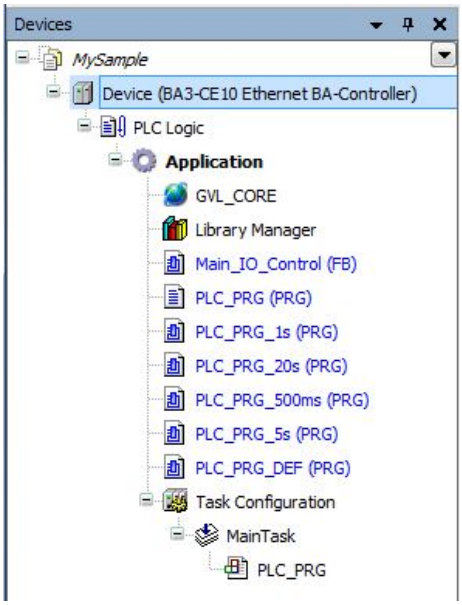
もし、エラーや警告が検出された場合は、出現場所がこのメッセージウィンドウに表示されます。エラーと警告が無くなるまで、問題個所の修正とコンパイル作業を繰り返す必要があります。

補足

インスタンスの追加または削除を含む変更をOnline Change(変更後に継続実行)で行うと問題が発生する場合がありますので、インスタンスの追加または削除を行った際のBuild(明示、暗黙的に関わらず)の前にはCleanを行うようにしてください。

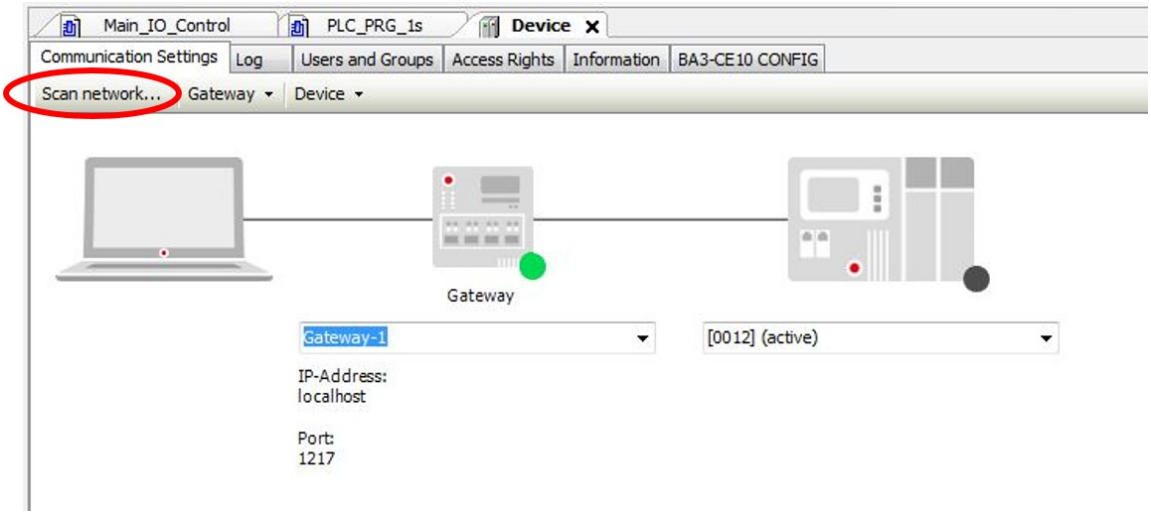
通信ゲートウェイの追加とコントローラの検出

操作:Deviceツリーから「Device」を選択



上記では「Device (BA3-CE10)」をマウスでダブルクリックします。画面中央には選択されたデバイスのDeviceタブが表示されます。

その中で「Communication Setting」タブを選択します。



次に、コントローラを登録します。ここでは現在接続できる(PCが接続されているネットワークに接続されているコントローラ)コントローラを検出します。

手順:

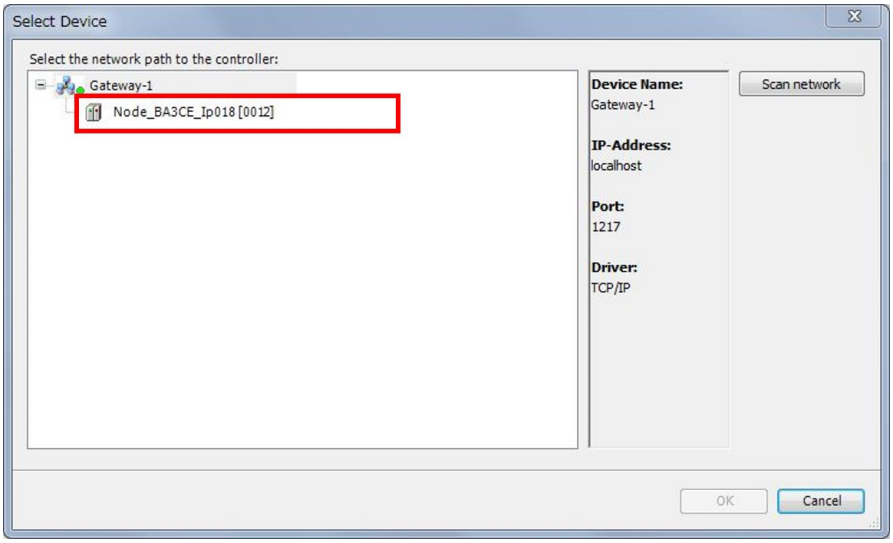
1. 画面左上に表示されている「Scan network」をマウスでクリックし選択します。

   ここでネットワーク上に存在する接続可能なコントローラが検出されリストアップされます。

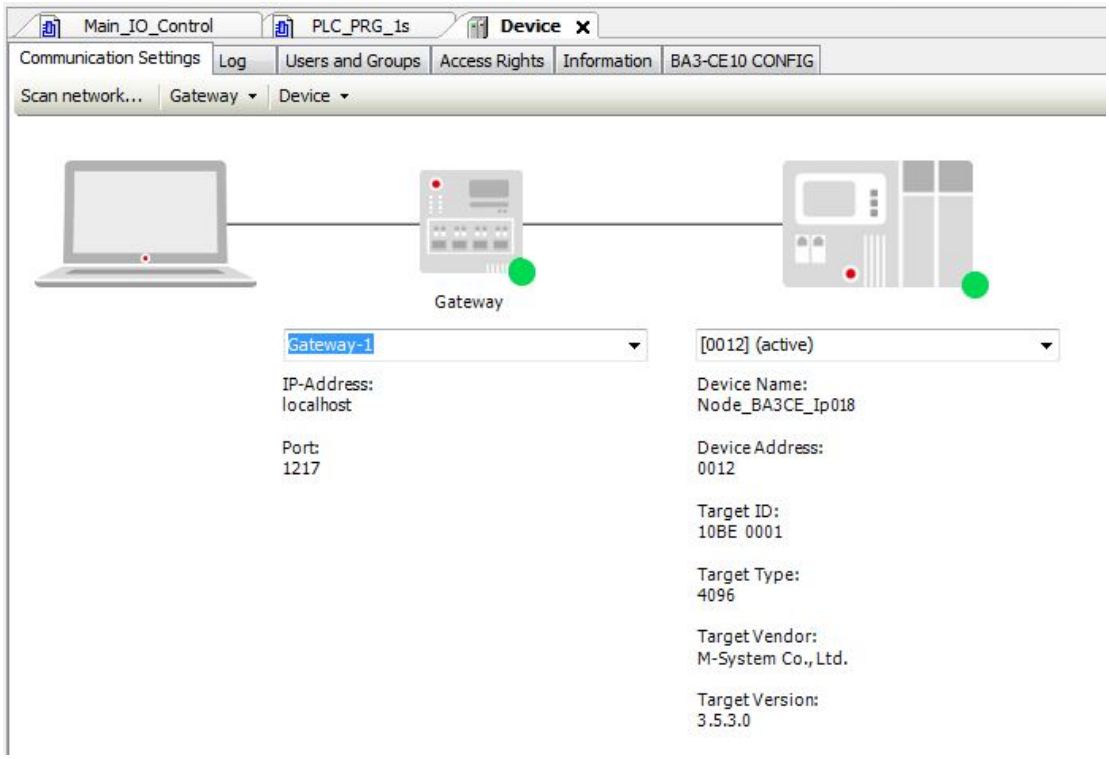
2. 接続するコントローラを選択します。

リストアップされたコントローラからコントローラを1台選び「OK」ボタンを押します。

下記の例では1台のコントローラが検出されているので、マウスで選択して「OK」を押します。



もし、「Scan network」で1台もコントローラが検出されない場合は、ネットワークの設定を確認する必要があります。接続対象のコントローラとPCとが同一のネットワークに存在するようにネットワークパラメータ(IPアドレスなど)が適切であるか確認してください。



検出されたコントローラは次の書式で表示されます。

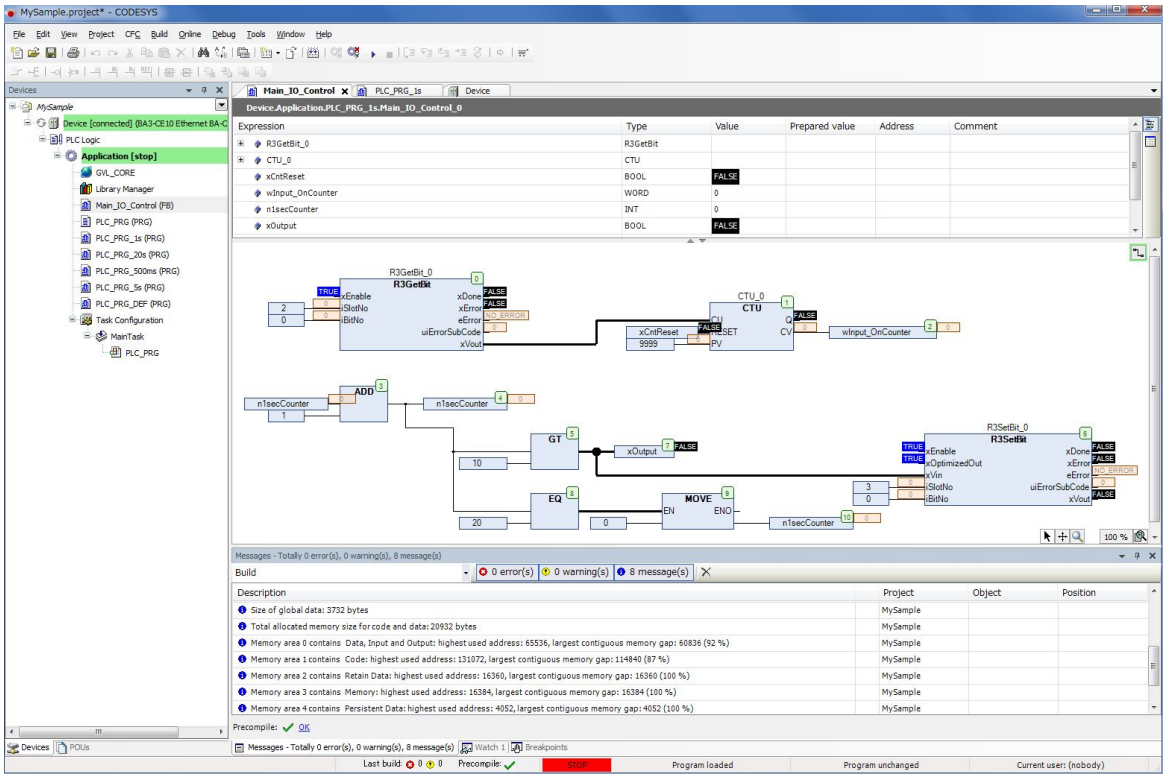
機種	表示形式1	表示形式2
BA3-CE10	Node_BA3CE_IpNNN NNN: IP Address (Rotary SW)	-

コントローラに接続(ログイン)とプログラムの転送

操作:メニュー「Online」から「Login」を選択

接続したコントローラにユーザプログラム(アプリケーション)がない場合に表示されます。「Yes」ボタンを押してプログラムのダウンロード(PCからコントローラへのプログラム転送)を行います。

プログラミングツールは、コントローラにログインすると現在状態をリアルタイムに表示するオンラインモードとなります。



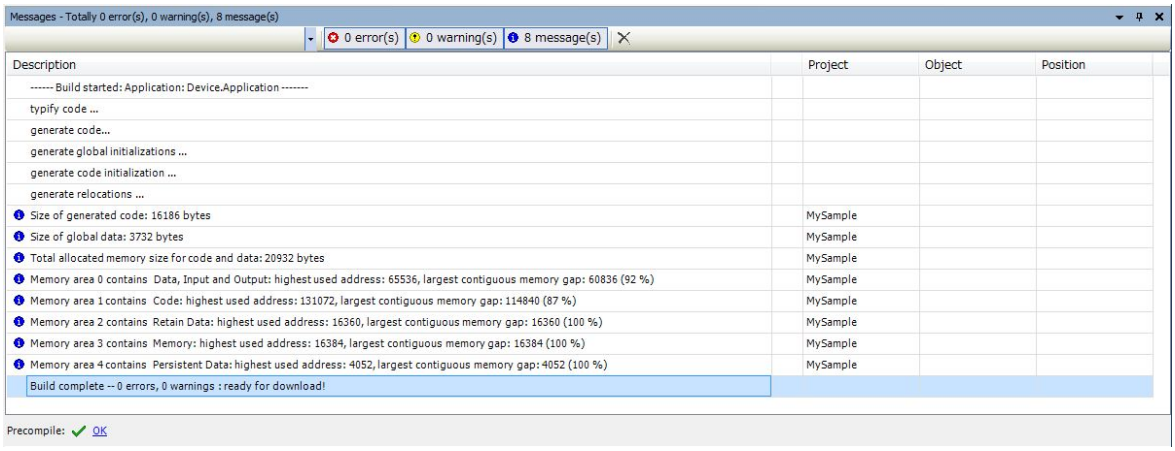
ダウンロードされたプログラムが必要としているメモリ容量や空きメモリ状況が表示されます。

ここで表示される gap は連続したメモリの空き容量です。

プログラムのダウンロードで必要な空きメモリは分断されたメモリの合計ではなく、この連続したメモリの空き容量です。何度もダウンロードを行うと次のダウンロードに必要な連続した空きメモリ容量が確保できなくなる場合があります。この場合のダウンロードは失敗します。連続した空きメモリを確保するためにはアプリケーションの再ビルドを行い完全ダウンロードを実施してください。

手順は

1. ログイン状態ならばログアウトします。
2. メニュー「Build」から「Clean all」を選択します。
3. 同じメニューから「Build」を選択します。
4. 再度コントローラにログインします(アプリケーションのダウンロードを問いかけるダイアログが表示されるので「Yes」とします)。



補 足

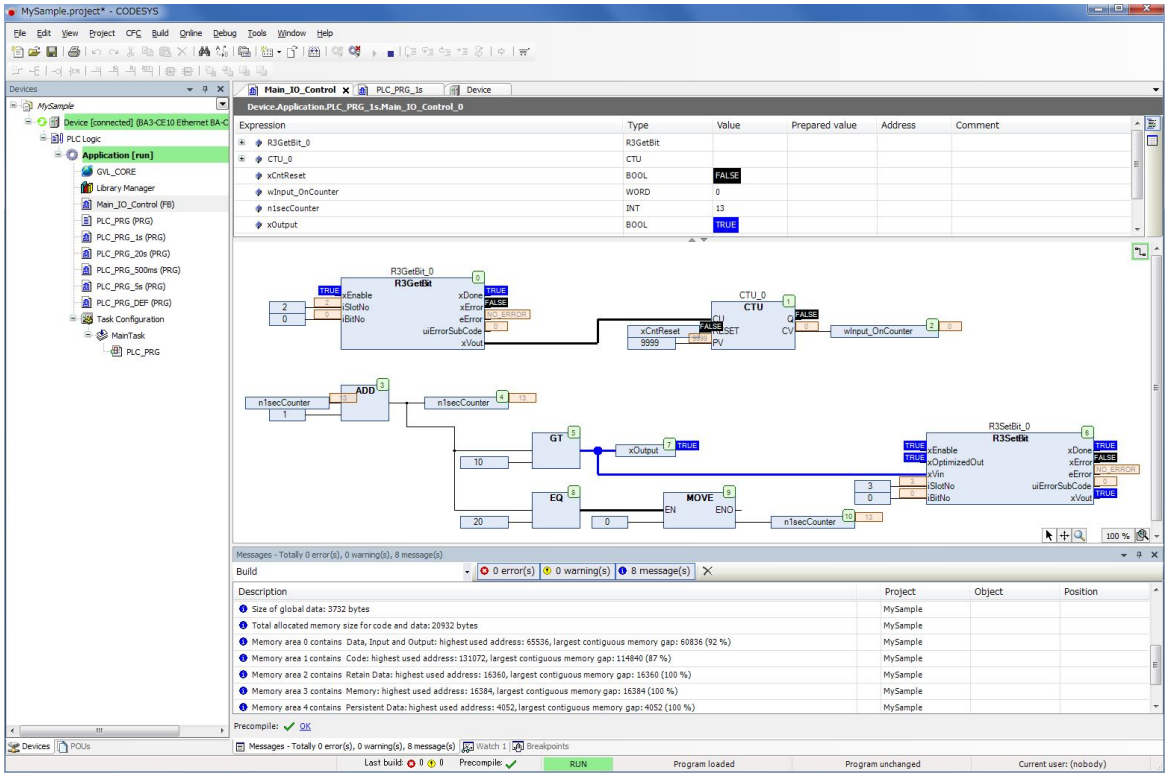
「コード」、「データ」、「RETAIN」、「PERSISTENT」の各領域の使用容量は「Build」から「Generate code」を実行することで事前に知ることもできます。

コントローラ内のプログラムを実行する

操作：プログラムを実行するにはメニュー「Debug」から「Start」を選択し、プログラムを停止するにはメニュー「Debug」から「Stop」を選択

プログラムを新規にダウンロードした直後のプログラム実行状態は「STOP」です。

プログラムの実行には実行状態を「RUN」に移行する必要があります。



コントローラ内にブートアプリケーションを作成する

ブートアプリケーションとは、コントローラが電源を投入され自動的に起動するアプリケーションです。

ここではオンライン(ログイン)状態でのブートアプリケーションの作成手順を示します。

手順:

メニュー「Online」から「Create boot application」を選択します。

この操作で現在プログラミングツールで開いているユーザプログラム(アプリケーション)がブートアプリケーションとしてコントローラ内に登録されます。

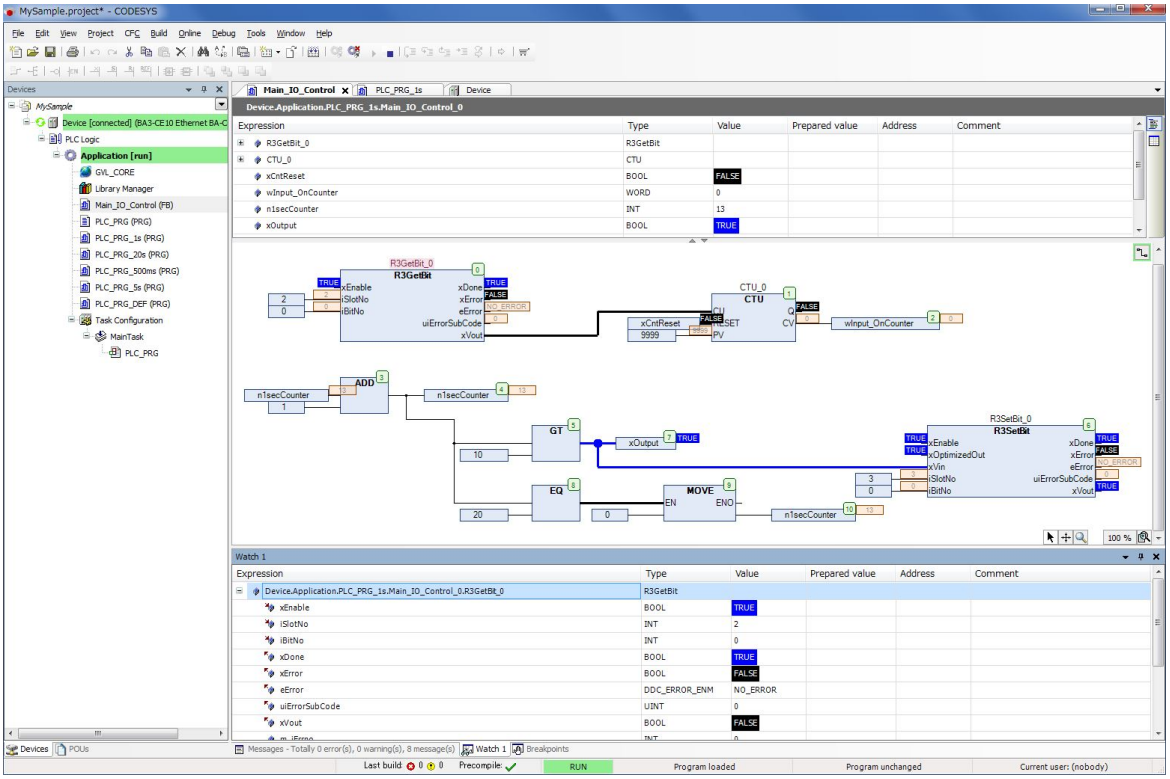
注意

- ログインなどで転送されたアプリケーションは、ブートアプリケーションではありません。そのため次回電源再投入では以前作成したブートアプリケーションが起動されます。別途オプション設定によりログインなどのダウンロード時に自動でブートアプリケーションを作成するように指定できます(Application プロパティの [Boot application]タブ)。
- [online] モード中にメニュー[Online] [Create boot application] で作成されるアプリケーションはフラッシュメモリに

格納されます。この書き込みには数十秒から数分時間を必要としますので、その間は書き込み処理中であることを示すERR LEDを点灯状態にします。このERR LEDが点灯している間は、ファイルの破損を防ぐために「リセット」や「電源のOFF」を行わないようにしてください。もし不慮の事態でファイルが破損した場合、次回電源投入かリセット時にファイルシステムは初期化(すべてのファイルは削除)されます。

オンラインモードで変数の現在値をモニタリング

オンラインモードでは変数の現在値やファンクション、ファンクションブロックの入出力パラメータの現在値をリアルタイムに表示します。また特定の変数をまとめてモニタできるウォッチリスト機能もあります。



オンラインモードで変数の現在値を設定変更

オンラインモードでは必要に応じて変数値を書き替えることができます。

書き替えには次の方法があります。

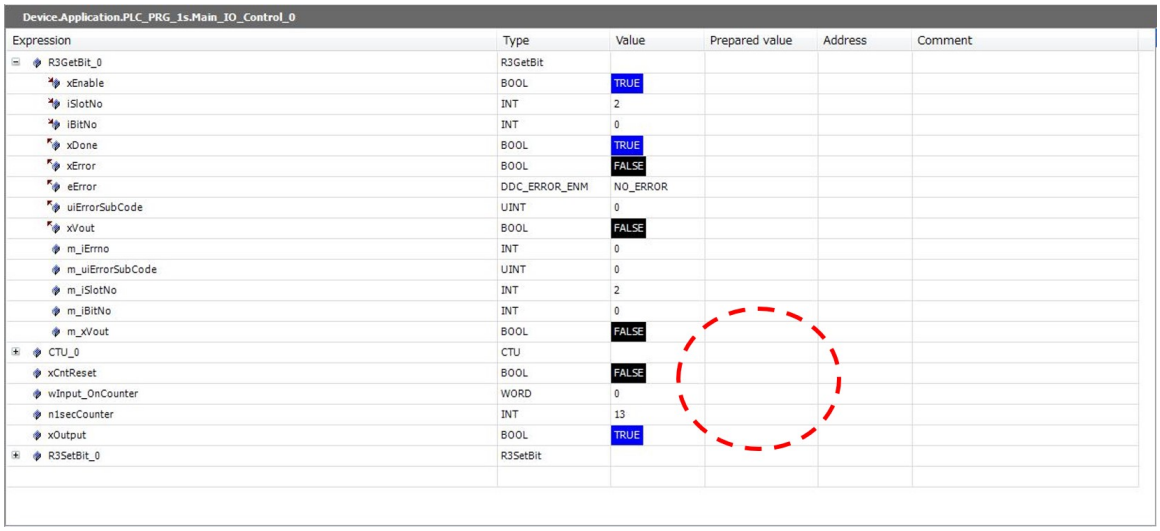
種類	手順	効果
値書き込み	メニュー「Debug」「Write values」あるいは <Ctrl>+<F7>	次の実行周期の最初に指定の値を1度だけ書き込みます。

種類	手順	効果
値の強制	設定： メニュー「Debug」「Force values」 または <F7> 解除： メニュー「Debug」「Unforce values」 または <Alt>+<F7>	実行周期の最初と最後で指定の値を毎周期書き込みます。 実行周期では次順で処理されます。 1．強制値書き込み 2．プログラムコードの実行 3．強制値書き込み

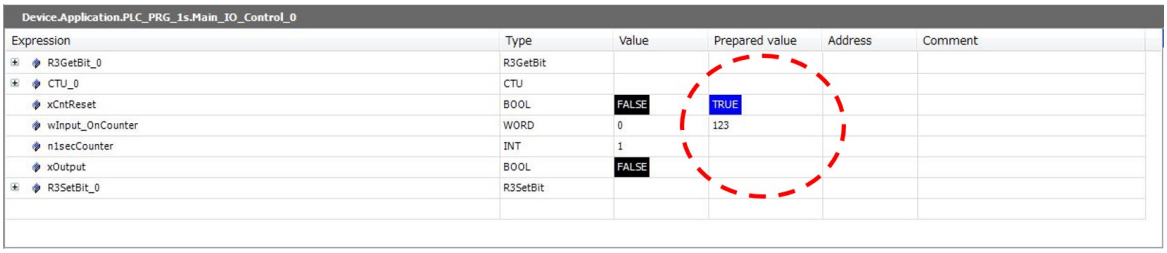
注意

「値書き込み」や「値の強制」は実行中のプログラム動作に予想外の影響を与えることがあります。使用する場合は動作の影響範囲や安全に十分な配慮をしてください。

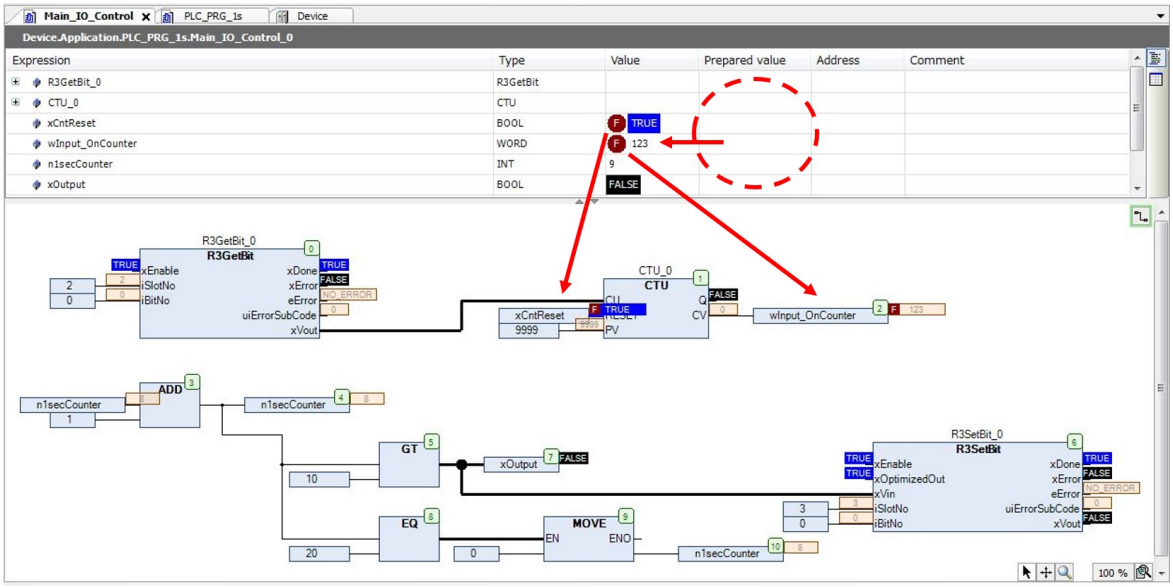
「値書き込み」、「値の強制」には「Prepared value」欄に書き込む値を事前に準備する必要があります。



「prepared value」欄に値を準備出来たら、「値書き込み」の場合は <Ctrl>+<F7> あるいは「値の強制」の場合は <F7> を押します。



以下は「値の強制」を行った場合の表示です。「Value」欄には、現在強制中であることを示す(F)が現在値の左に表示されています。



プログラムコードを任意の位置で停止させる

オンラインモードでは必要に応じてブレークポイントを設定できます。

プログラムコードの実行がブレークポイントに達するとプログラムの実行が一時的に停止されます。

停止位置からプログラムを再開するには次の指定ができます。

1. 実行の再開

   「Run」

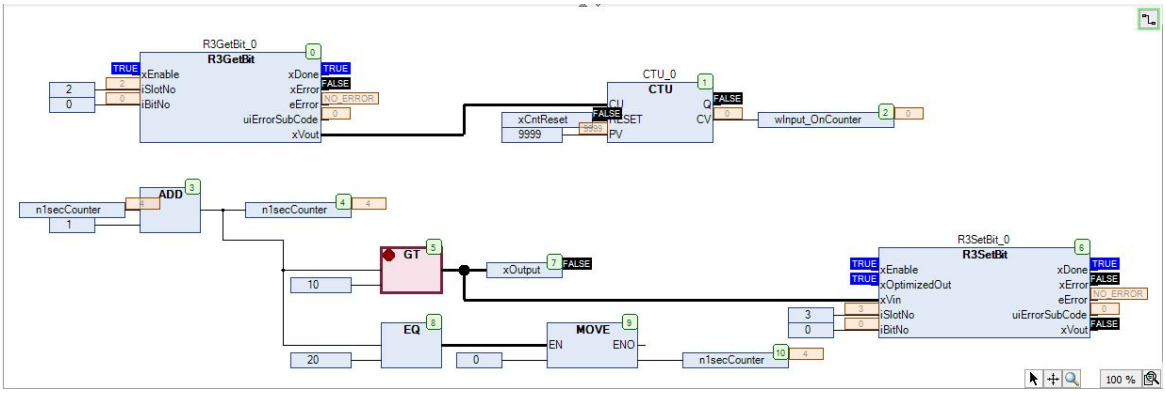
2. ステップ実行

   「Step Over」:現在命令の次の命令に進む
   「Step Into」:現在命令の内部に移動(命令のコードが表示可能な場合)

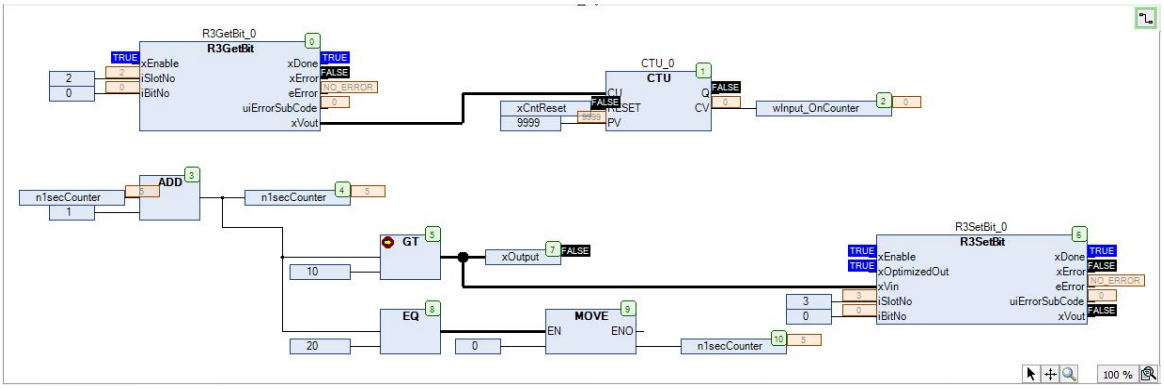
「Step Out」: 現在のファンクションあるいはファンクションブロックから呼び出すもとに戻る

3. カーソル位置まで実行

「Run to Cursor」: 現在停止位置からカーソル位置の命令まで実行する



停止したい命令(ここでは実行番号5の「GT」命令)にブレークポイントを設定した時の表示です。ブレークポイントが設定された命令(またはインスタンス)には命令BOXの左上に  が表示されます。



実行がブレークポイントに到達するとプログラムが一時停止します。ブレークポイントに到達するとブレークポインを示すシンボルが  表示に替わります。

6.2.サンプルコード

サンプルプログラムコード(ST言語)

(宣言部)

```
FUNCTION_BLOCK Main_IO_Control_ST
VAR_INPUT
END_VAR
VAR_OUTPUT
END_VAR
VAR
    R3GetBit_0          : R3GetBit;
    CTU_0               : CTU;
    xCntReset           : BOOL;
    wInput_OnCounter    : WORD;
    n1secCounter        : INT;
    xOutput             : BOOL;
    R3SetBit_0          : R3SetBit;
END_VAR
```

(ボディ部)

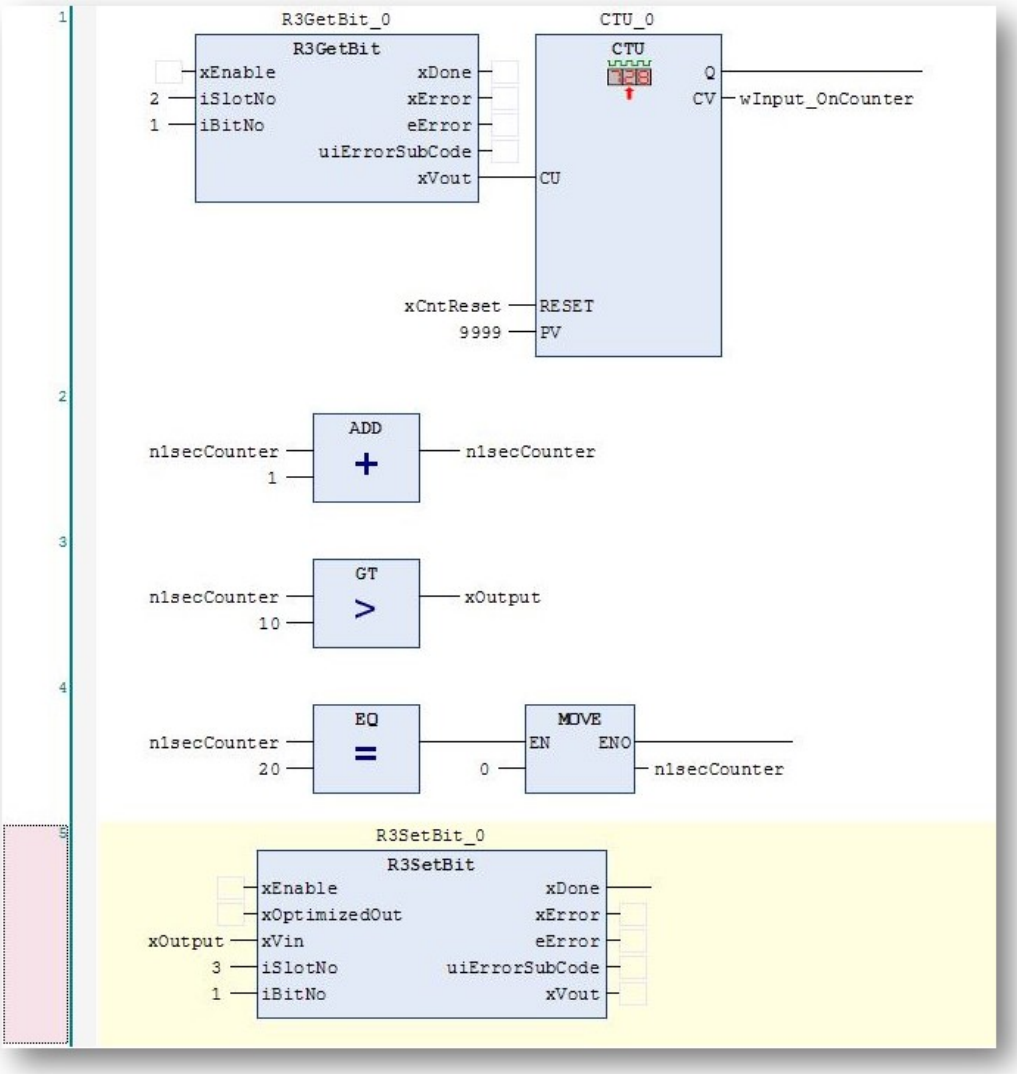
```
(* Network 1 *)
R3GetBit_0(iSlotNo:=2, iBitNo:=3);
CTU_0(CU:=R3GetBit_0.xVout, RESET:=xCntReset, PV:=9999, CV=>wInput_OnCounter);
(* Network 2 *)
n1secCounter := n1secCounter + 1;
(* Network 3 *)
IF n1secCounter > 10 THEN
    xOutput := TRUE;
ELSE
    xOutput := FALSE;
END_IF
(* Network 4 *)
IF n1secCounter = 20 THEN
    n1secCounter := 0;
END_IF
(* Network 5 *)
// R3GetBit_0(iSlotNo:=3, iBitNo:=1, xVin:=xOutput);
R3SetBit_0.iSlotNo := 3;            // Slot:3, Ch:2
R3SetBit_0.iBitNo := 3;
R3SetBit_0.xVin := xOutput;
R3SetBit_0();
```

サンプルプログラムコード(FBD言語)

(宣言部)

```
FUNCTION_BLOCK Main_IO_Control_FBD
VAR_INPUT
END_VAR
VAR_OUTPUT
END_VAR
VAR
    R3GetBit_0          : R3GetBit;
    CTU_0               : CTU;
    xCntReset           : BOOL;
    wInput_OnCounter    : WORD;
    n1secCounter        : INT;
    xOutput             : BOOL;
    R3SetBit_0          : R3SetBit;
END_VAR
```

(ボディ部)

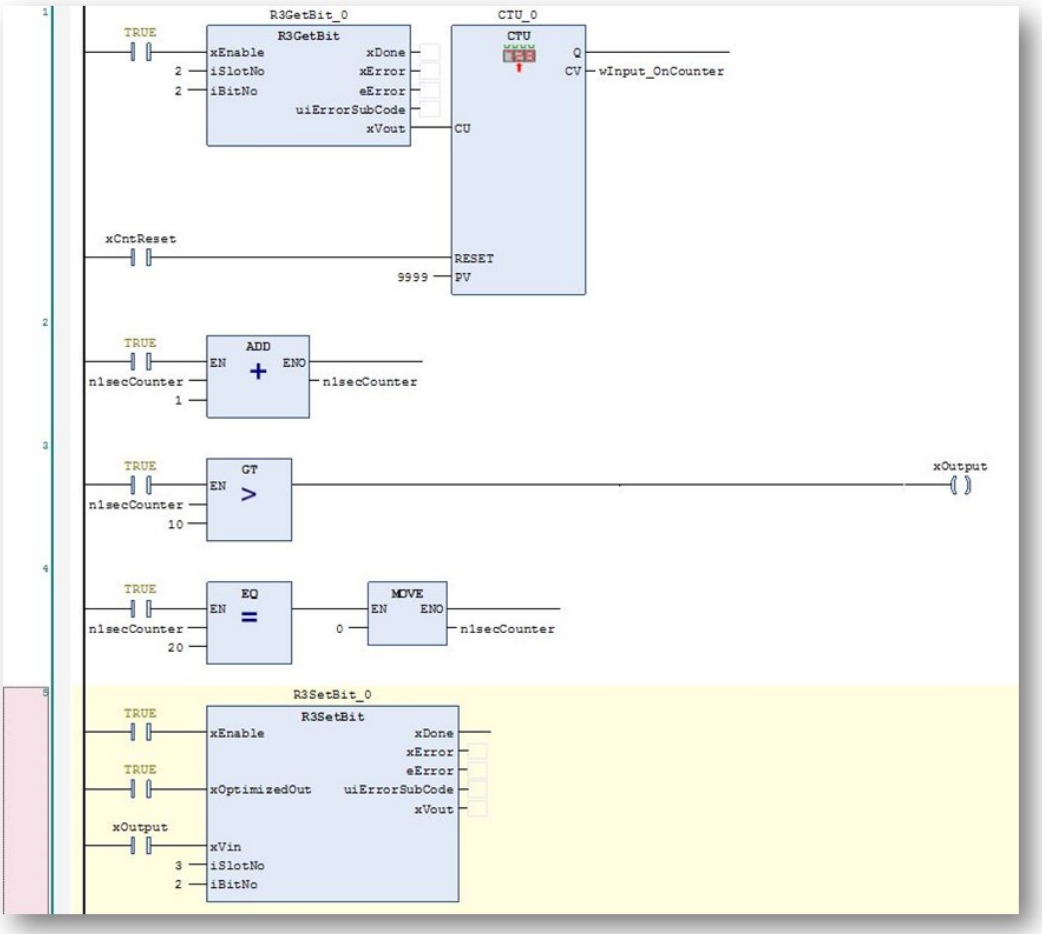


サンプルプログラムコード(LD言語)

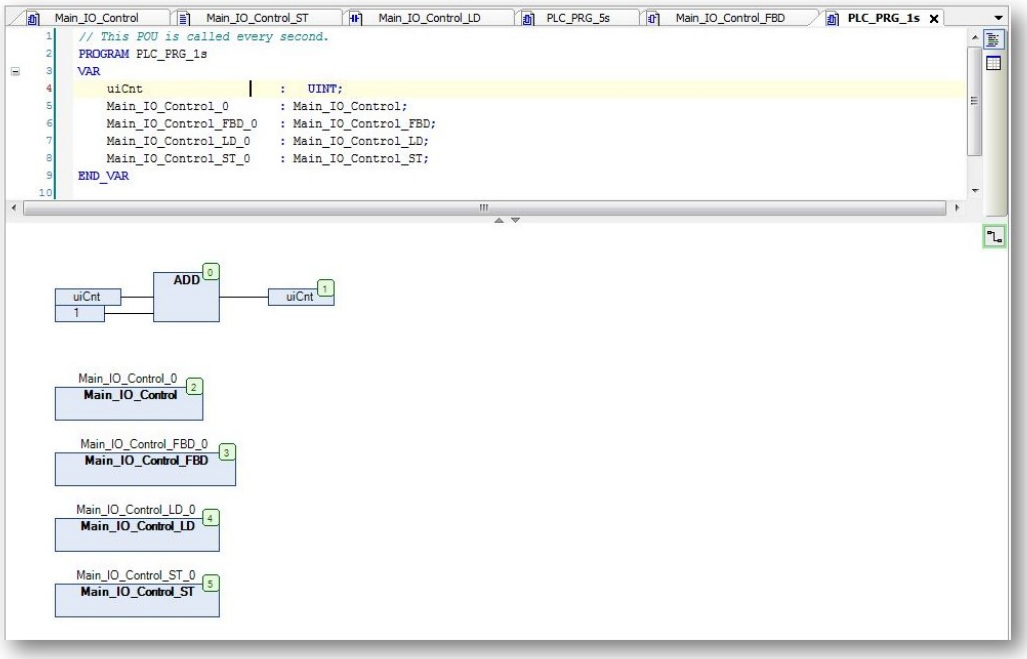
(宣言部)

```
1  FUNCTION_BLOCK Main_IO_Control_LD
2  VAR_INPUT
3  END_VAR
4  VAR_OUTPUT
5  END_VAR
6  VAR
7      R3GetBit_0          : R3GetBit;
8      CTU_0               : CTU;
9      xCntReset           : BOOL;
10     wInput_OnCounter    : WORD;
11     n1secCounter        : INT;
12     xOutput             : BOOL;
13     R3SetBit_0          : R3SetBit;
14 END_VAR
15
```

(ボディ部)



サンプルプログラムコードの呼び出し(PLC_PRG_1s)

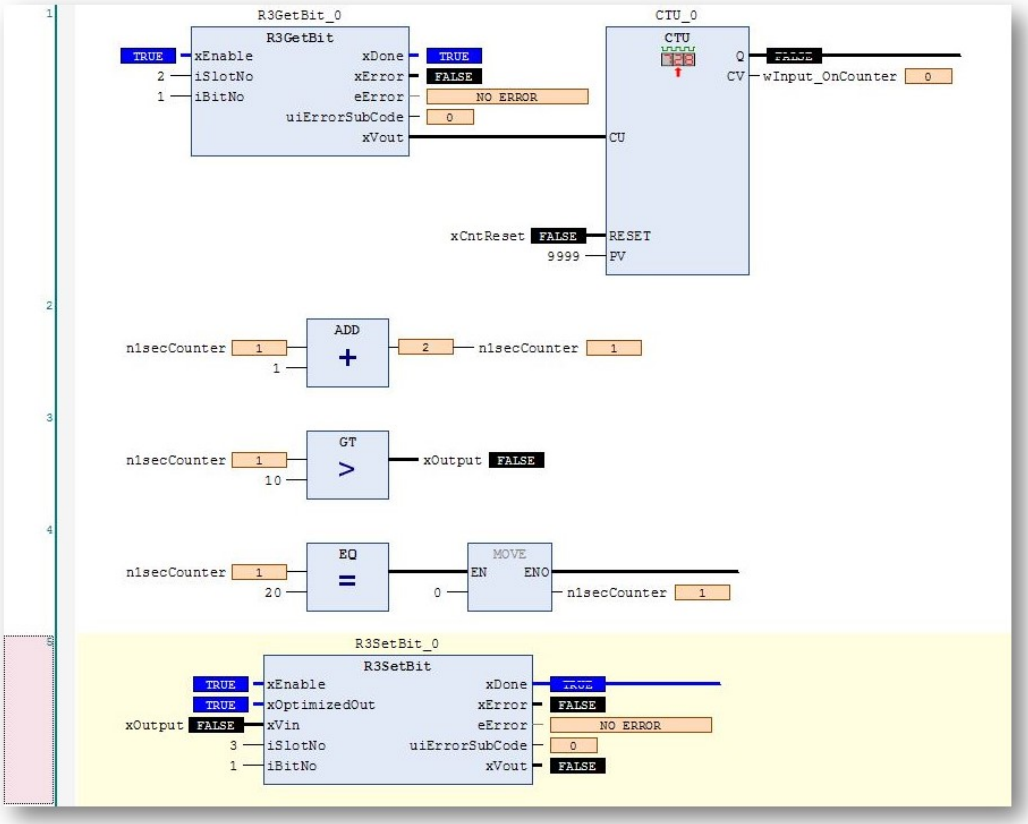


サンプルプログラムのオンラインモニタ表示例

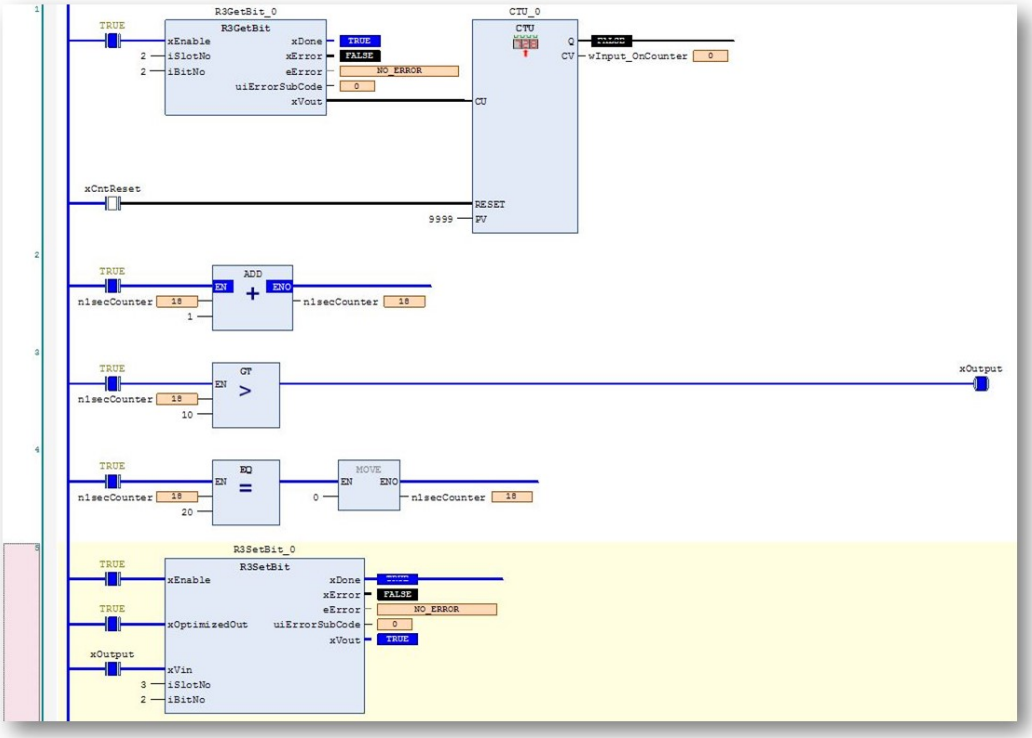
(ST言語)

```
(* Network 1 *)
R3GetBit_0(iSlotNo:=2, iBitNo:=3);
CTU_0(CU:=R3GetBit_0.xVout, RESET:=xCntReset, PV:=9999, CV=>wInput_OnCounter);
(* Network 2 *)
n1secCounter := n1secCounter + 1;
(* Network 3 *)
IF n1secCounter > 10 THEN
    xOutput := TRUE;
ELSE
    xOutput := FALSE;
END_IF
(* Network 4 *)
IF n1secCounter = 20 THEN
    n1secCounter := 0;
END_IF
(* Network 5 *)
// R3GetBit_0(iSlotNo:=3, iBitNo:=1, xVin:=xOutput);
R3SetBit_0.iSlotNo := 3;          // Slot:3, Ch:2
R3SetBit_0.iBitNo := 3;
R3SetBit_0.xVin := xOutput;
R3SetBit_0();
RETURN
```

(FBD言語)



(LD言語)

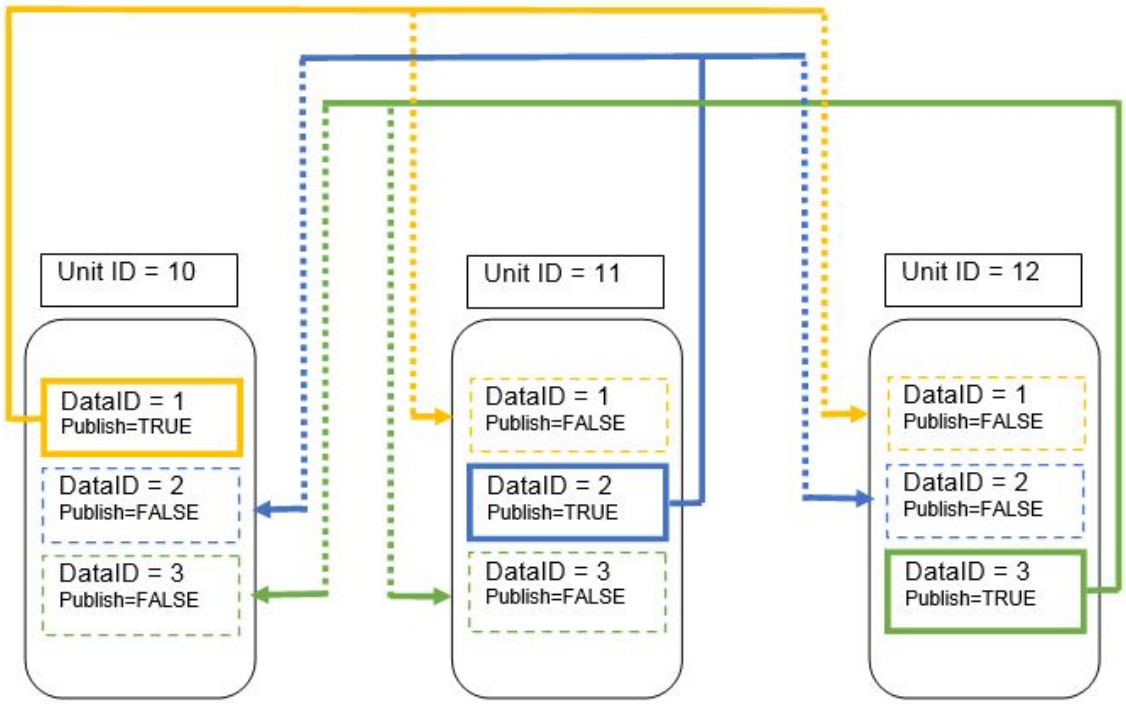

R3GetBit_0
R3GetBit
xEnable
iSlotNo
iBitNo
xDone
xError
eError
uiErrorSubCode
xVout
CTU_0
CTU
CU
RESET
PV
Q
CV
wInput_OnCounter
xCntReset
9999
ADD
nlsecCounter
GT
xOutput
EQ
MOVE
R3SetBit_0
R3SetBit
xOptimizedOut
xVin
TRUE
FALSE
NO_ERROR

7. Global Data Point

ここではGlobal Data Point機能に関する説明、設定や注意事項を記述しています。

機能説明

Global Data Point は、複数コントローラでデータを共有する機能です。
ここで使用されるデータは現在値やデータの品質などの情報をポイントとして管理しています。コントローラはシステムドメインに存在する他コントローラ向けてデータの放送(送出)と放送されたデータの取り込み(受信)を行います。
データの放送は該当ポイントのプロパティにPublish(=TRUE), DataID の指定を行うことで開始できます。また、放送データの取り込みは、該当ポイントにPublish(=FALSE), DataID の設定を行うことで DataID と一致するデータが取り込まれます。そのため DataID は、送信元と受信先のポイントで一致させる必要があります。
ここで使用されている「データ放送」の特徴は同じデータを複数のコントローラが同時に受信できることです。



ポイント番号とDataID

ポイントは登録リストの並び順に1から始まる連続した番号で管理されます。IECプログラムではこのポイント番号を指定してポイントデータにアクセスします。また、このポイント番号とは別に放送されたデータを識別する番号がDataIDです。このDataIDは放送されたデータを識別するための番号なのでシステムドメイン内で管理し採番します。DataIDは連番である必要はありません。

システムドメイン

放送データが到達する(データ交換可能な)ネットワーク範囲(サブネット)がシステムドメインとなります。

書き込み優先度

各ポイントは書き込み優先度を持ちます。優先度に書き込まれる値で非数(NaN)は特別な意味を持ちます。非数(NaN)が設定されている優先度は評価されないで残りの内で最も優先度に設定されている値が現在値(PVAL0)とされます。

以下は優先度の使用法の一例です。各優先度に設定された値の中で最優先度の値が現在値として採用されます。

	用途	設定例1	設定例2	設定例3	設定例4
優先度5 (PVAL5)	スケジュール、自動制御の指示値	1	1	NaN	10
優先度4 (PVAL4)	火災時の指示値	NaN	NaN	2	200
優先度3 (PVAL3)	停電時の指示値	NaN	0	NaN	30
優先度2 (PVAL2)	手動時の指示値	NaN	NaN	NaN	400
優先度1 (PVAL1)	強制時の指示値 (テスト、保守)	NaN	NaN	5	50
現在値 (PVAL0)	-	1	0	5	50

設定

共通設定

パラメータ	型	デフォルト値	説明
Multicast	BOOL	TRUE	TRUEでマルチキャスト、それ以外はブロードキャスト
BroadcastAddress	STRING	'192.168.1.255'	ブロードキャストで使用するIPアドレス
MulticastAddress	STRING	'224.0.1.1'	マルチキャストで使用するIPアドレス
Port	UINT	9898	ポート番号
TTL	INT	1	TTL (マルチキャストの場合に有効)

データポイント設定

パラメータ	型	デフォルト値	説明
Publish	BOOL	FALSE	TRUEで放送側、それ以外は受信側
InhibitCOV	BOOL	FALSE	TRUEで変化時で放送し、それ以外は変化で放送なし
DataID	UINT	1	データの識別番号 (0 ～ 65535)
CycTimeSec	UINT	15	放送側は放送周期間隔、受信側は最大受信監視時間を秒で設定します (0:機能無効, 1 ～ 65535:秒)

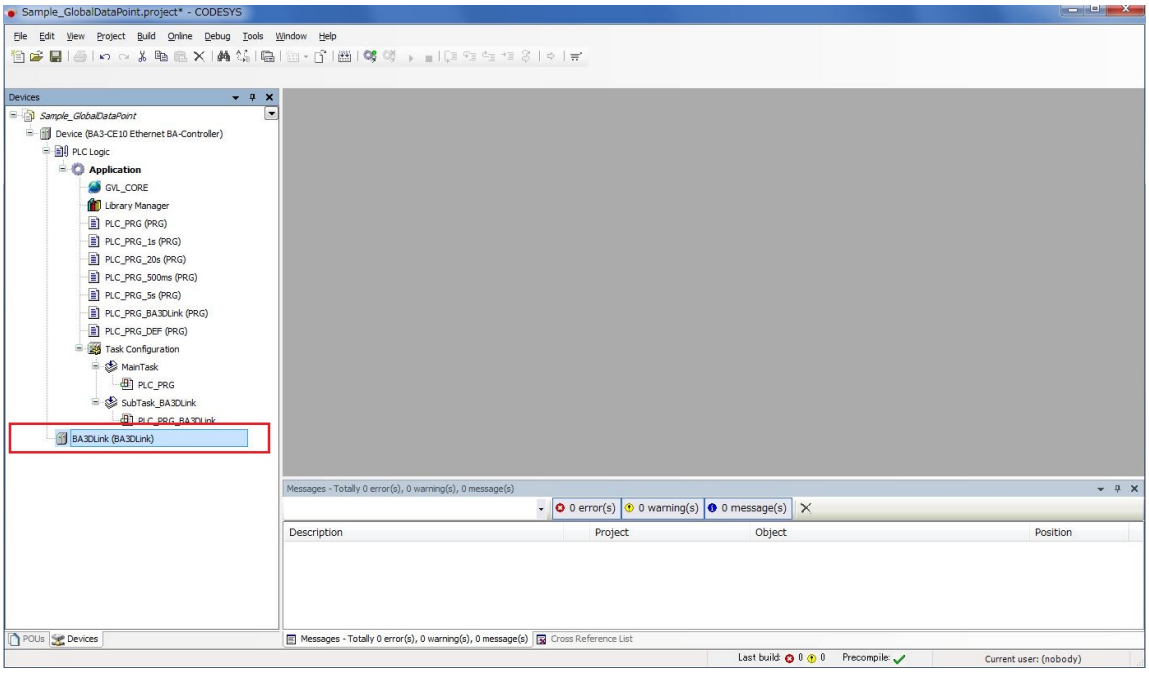
パラメータの組み合わせ動作一覧

方向	動作	Publish	InhibitCOV	CycTimeSec	説明
送信	値更新＋最大送信時間15sec	TRUE	FALSE	15	値更新と更新がなくても15秒に1度の送信が行われます
送信	値更新	TRUE	FALSE	0	値の更新時のみ送信が行われます

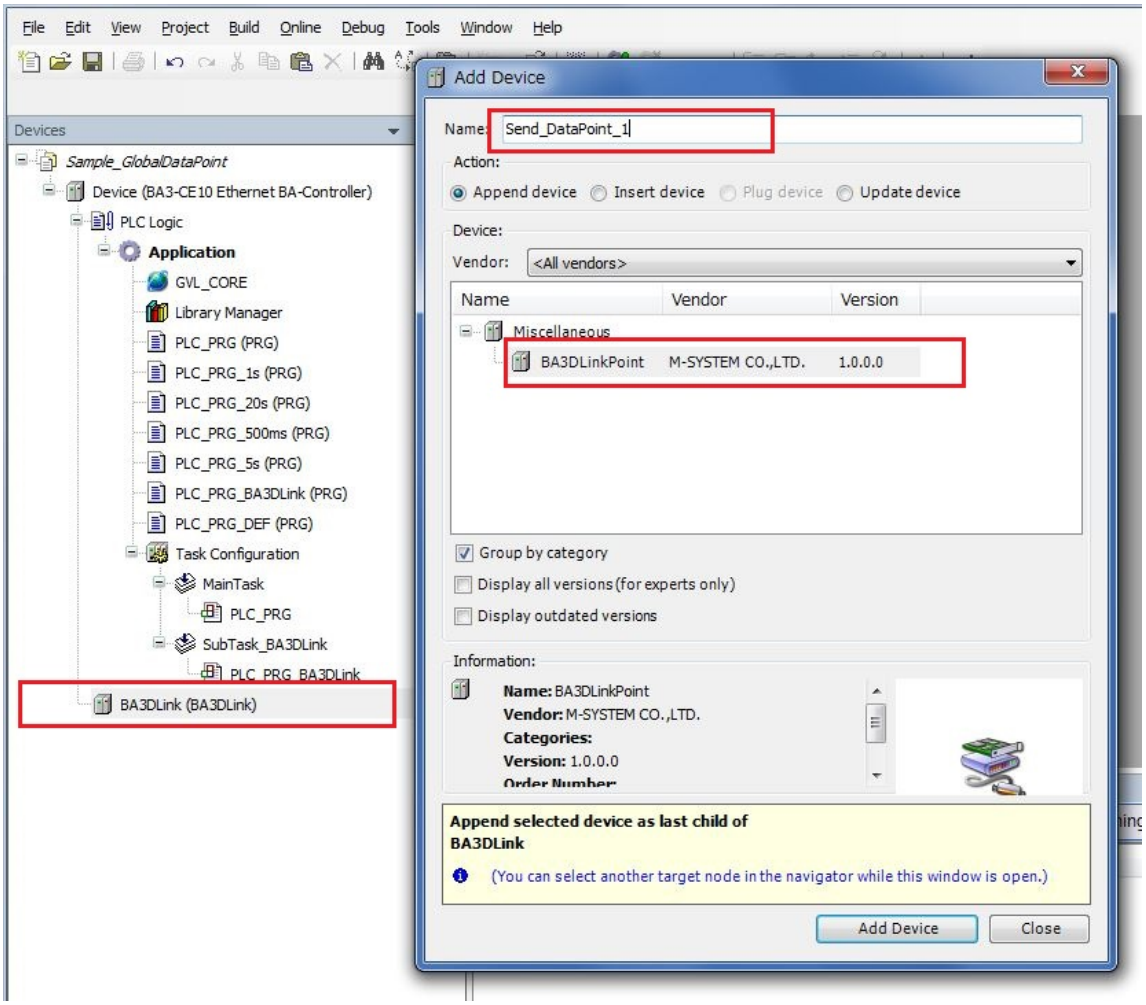
方向	動作	Publish	InhibitCOV	CycTimeSec	説明
送信	最大送信時間 15sec	TRUE	TRUE	15	15秒毎の定周期で送信が行われます
送信	-	TRUE	TRUE	0	-
受信	最小受信時間監視なし	FALSE	-	0	最小受信時間監視は行われません
受信	最小受信時間 15secで監視	FALSE	-	15	15秒以内に受信がなければポイントの Quality が DLINKDWN に変更されます

DLink Configuration イメージ

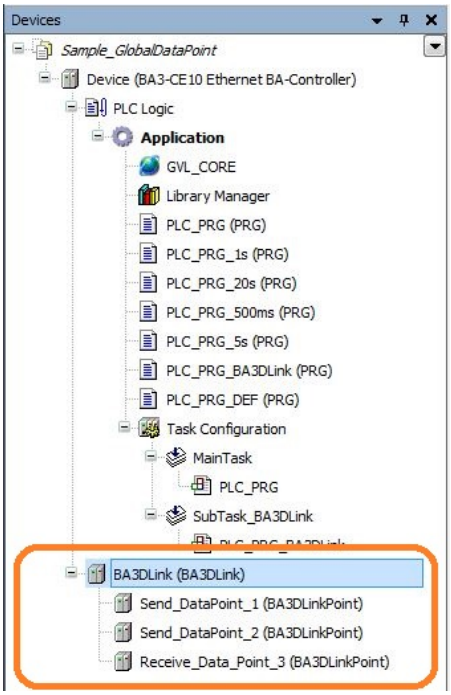
データポイントを追加するには [BA3DLink (BA3DLink)] を選択して、右クリックで表示されるコンテキストメニューから [Add Device ...] を選択します。



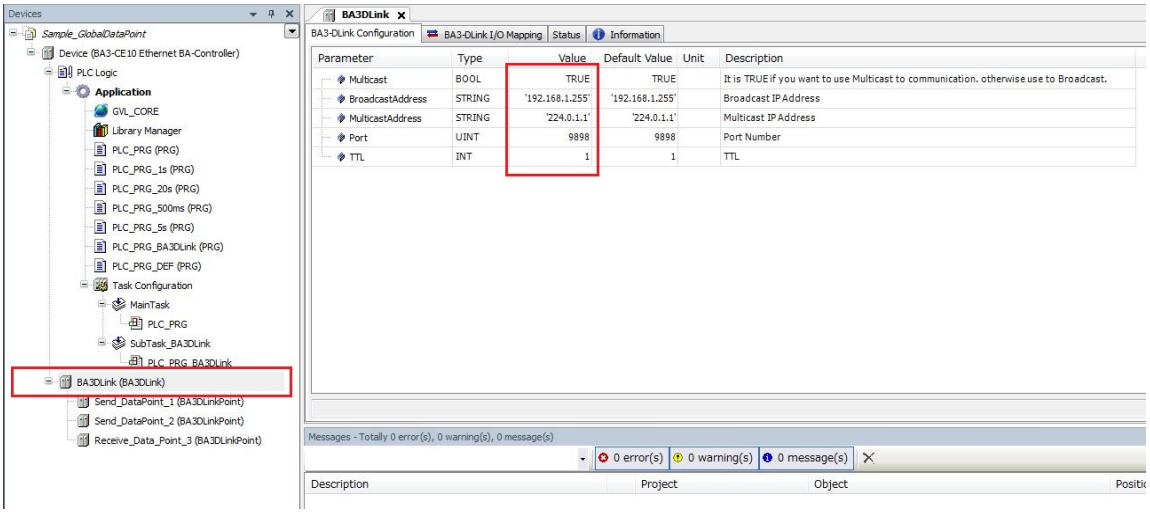
表示された [Add Device] ダイアログで [BA3DLinkPoint] を選択し「名称」(例：Send_DataPoint_1)を入力した後に [Add Device] ボタンを押します。(連続で登録できるようにダイアログは [Close] を押すまで表示されています)



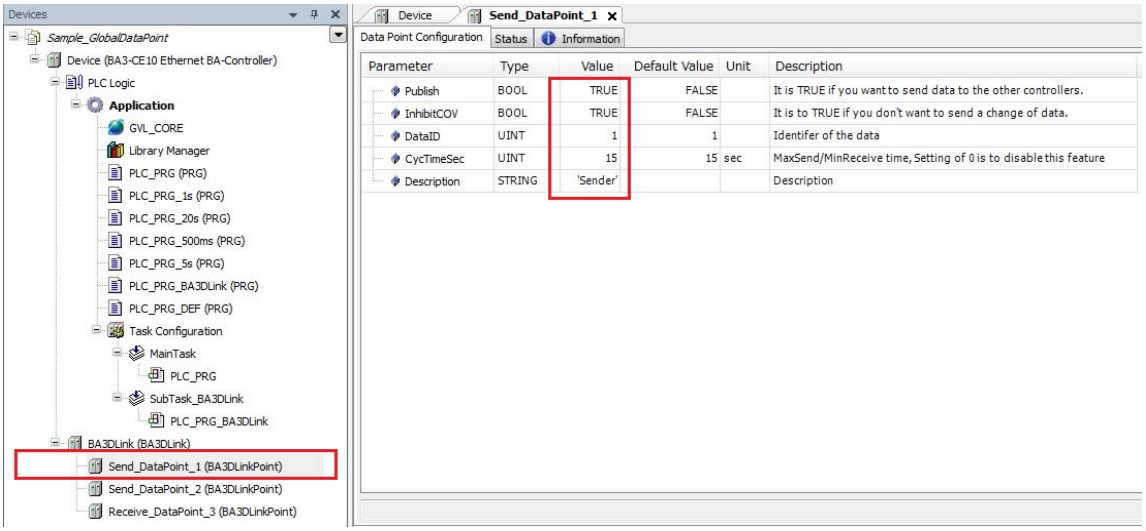
ここでは例として3つのデータポイントを登録しています。



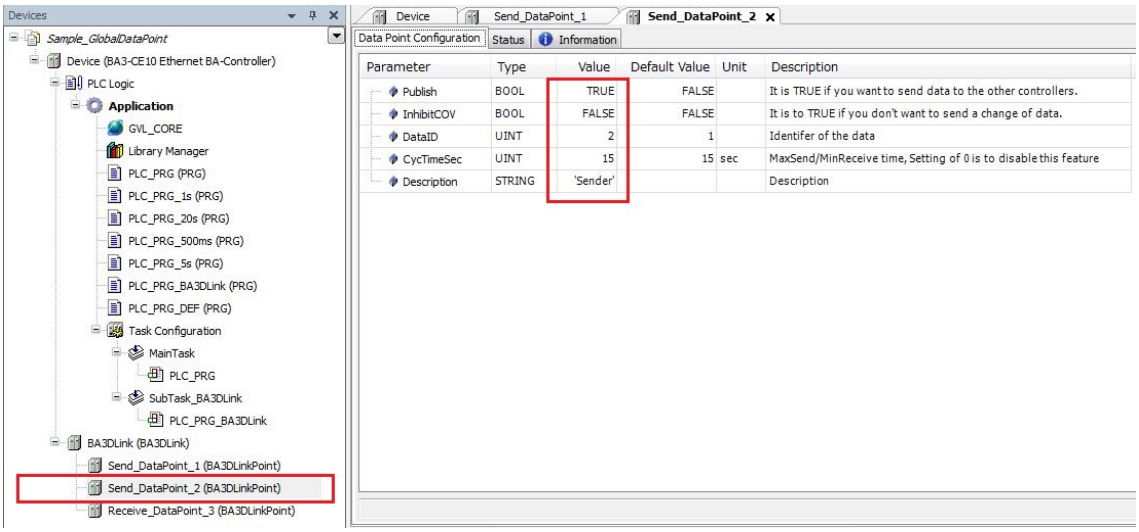
[BA3DLink (BA3DLink)] をダブルクリックすることで右ウィンドウにパラメータ設定ペインが表示されます。この例ではデフォルト値を使用しています。



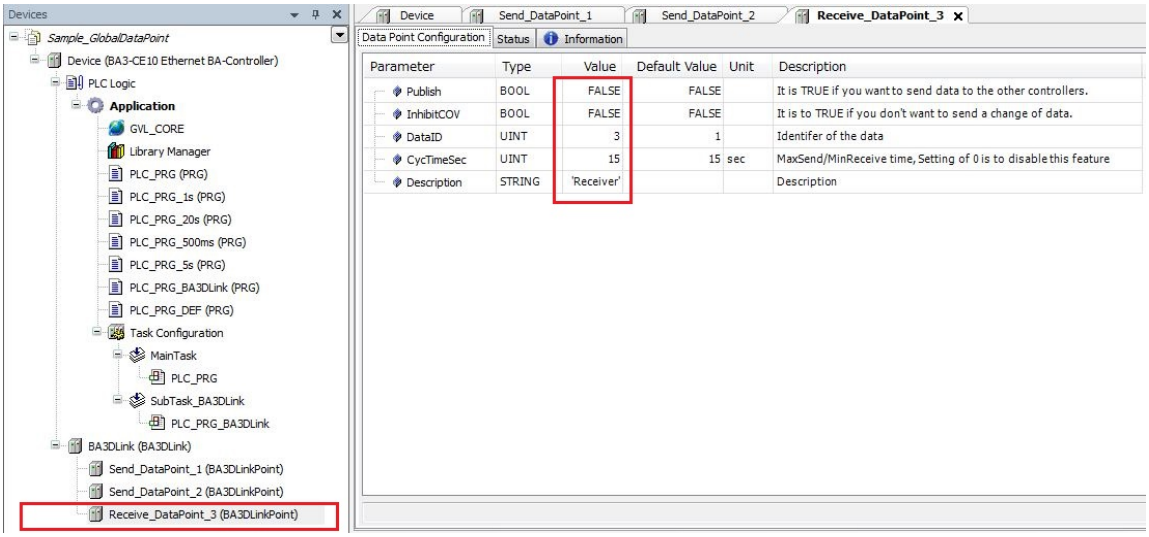
[Send_DataPoint_1 (BA3DLinkPoint)] をダブルクリックすることで右ウィンドウにパラメータ設定ペインが表示されます。ここではポイント番号1 (Send_DataPoint_1)にDataID=1の変化＋15sec定周期を設定しています。



次に、[Send_DataPoint_2 (BA3DLinkPoint)] をダブルクリックすることで右ウィンドウにパラメータ設定ペインを表示させます。ここではポイント番号2(Send_DataPoint_2)にDataID=2の15sec定周期を設定しています。

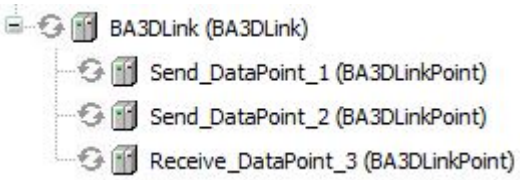


次に、[Receive_DataPoint_3 (BA3DLinkPoint)] をダブルクリックすることで右ウィンドウにパラメータ設定ペインを表示させます。ここではポイント番号3(Receive_DataPoint_3)にDataID=3の最受信監視15secを設定しています。

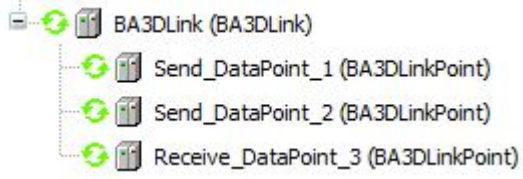


DLink 状態表示

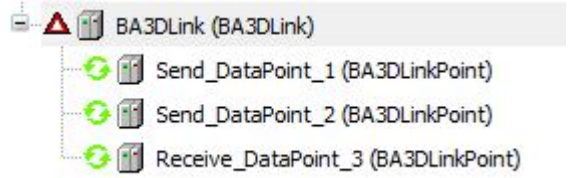
(停止)これはパラメータが正常に読み込まれ停止している状態を示しています。



(正常)これは正常動作している状態を示しています。



(異常)これは異常により動作していない状態を示しています。パラメータに誤りがないか確認が必要です。



注意事項

受信側コントローラは他コントローラが放送したデータを取り込みます。これは受信したデータの送信元 CUNIT_ID と自身の CUNIT_ID が一致するデータは取り込まれないことを意味しています。そのためデータの送信元を識別するために使用される CUNIT_ID は必ず設定する必要があります。

トラブルシューティング

動作しない場合は以下の項目を確認してください。

- [CUNIT_ID] 設定が適切な値に設定されているか。(通常はコントローラ毎にシステム内でユニークな値を割り当てます)
- ブロードキャストを使用する場合は、コントローラの IP アドレスと [DLink Configuration] で指定したブロードキャスト IP アドレスと同一のサブネットであるか。
- [データポイント設定] の [DataID] が適切な値に設定されているか。(通常はデータポイント毎にシステム内でユニークな値を割り当てます)

(このページは空白です)

8.ライブラリ

ライブラリはプログラムを再利用する最も有効な方法です。自身の作成したファンクションやファンクションブロックを他のアプリケーションで使用したり、他の開発者に提供することを容易にします。

ここではユーザライブラリの作成方法から既存のライブラリの説明を行います。

8.1.ユーザライブラリ

ライブラリの作成はテンプレートを使用する場合と新規(空)に作成する方法とがあります。

テンプレートを利用した場合は多くのモジュールが含まれているので完成後には不要なモジュールを削除されることをお勧めします。

CODESYS CAAのガイドラインに準拠したライブラリの作成には次のテンプレートを使用します。

[CODESYS container library], [CODESYS interface library], [CODESYS library]

この中で[External CODESYS library] は使用しません。

この章では、ライブラリとしての最小限の要素を持つライブラリを新規(空)から作成する方法を説明します。

詳細についてはオンラインヘルプの「Guidelines for creating libraries」を参照してください。

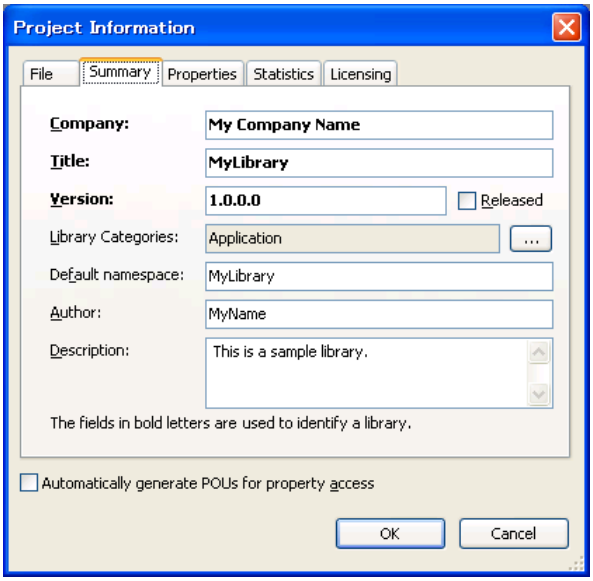
ユーザライブラリの作成

メニュー [File] [New Project...] のダイアログで [Categories]=Library, [Templates]=Empty library を選択します。

ライブラリのプロジェクト情報を設定

左側に表示される「Deviceツリー」のタブ[POUs]を選択しておきます。

メニュー [Project] [Project Information...] を選択します。



Company: <My Company Name>

Title: <MyLibrary>

Version: <1.0.0.0>

Library Categories: 次に説明します。

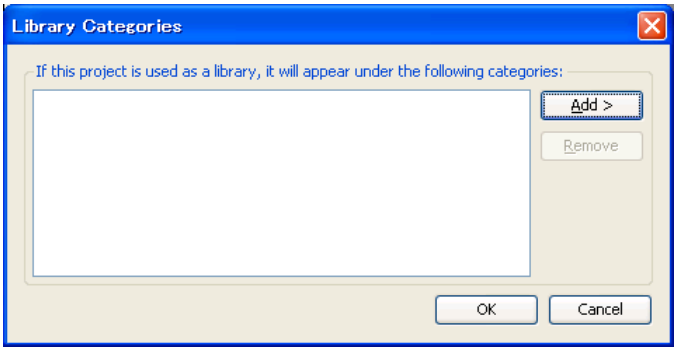
Default namespace: <MyLibrary>

Author: <MyName>

Description: <This is a sample library.>

これらの項目を入力します。

[Library categories]の入力は、右端に位置する [...] ボタンを押します。

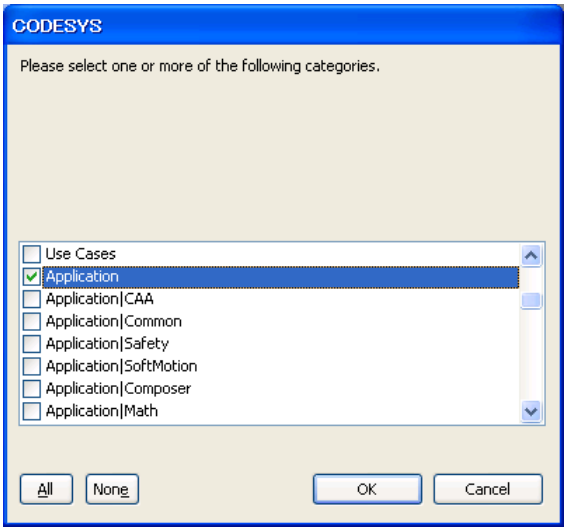


[Add>] [From Description File...]

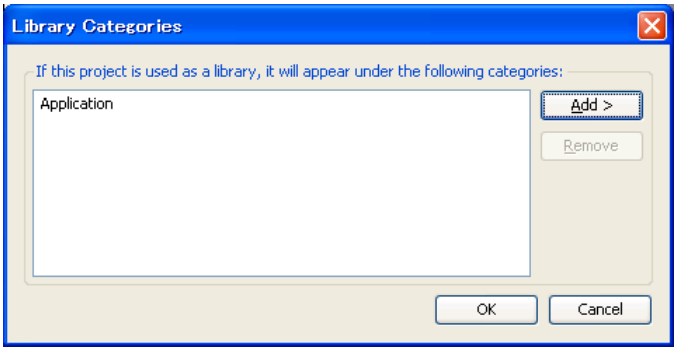
[Windows XP の場合]

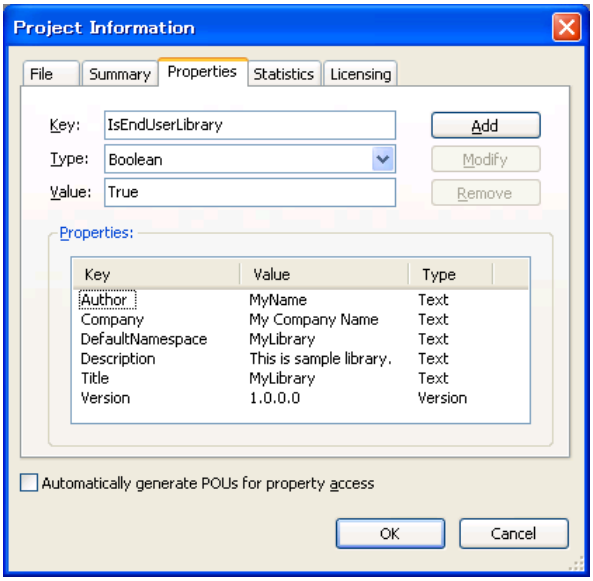
C:\Program Files\m-system\CoDeSysV3 Tools\CODESYS\Templates\Library_Template

LibraryCategoryBase.libcat.xml



全選択状態なので[None]を押し一旦解除しておき、リストの[Application]だけを選択する。





ここで次のプロパティを手動で追加します。

Key: IsEndUserLibrary

Type: Boolean

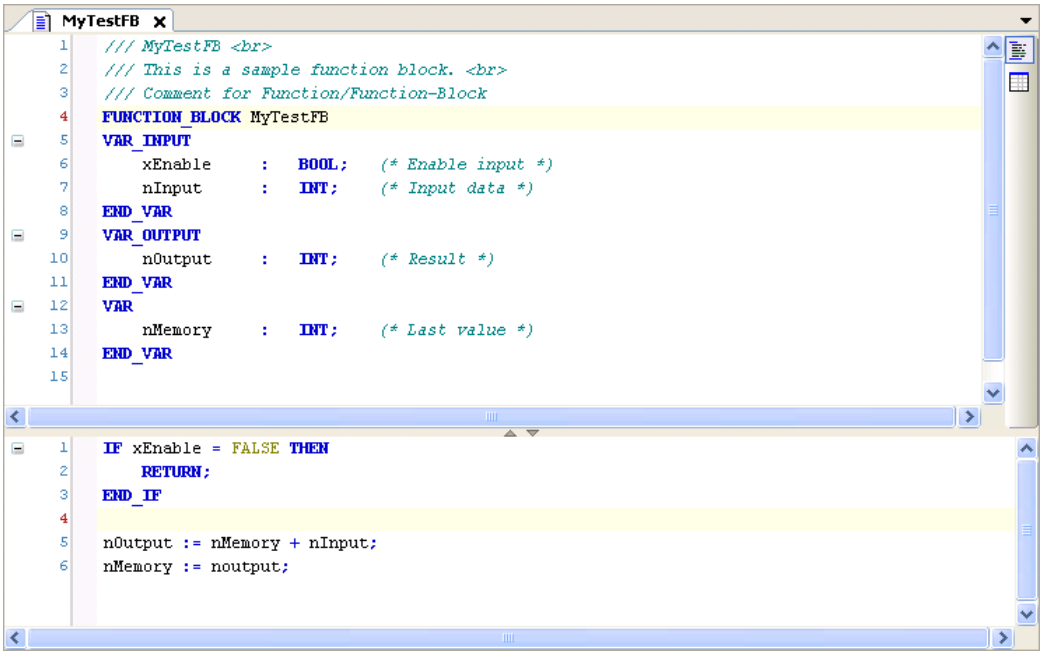
Value: True

これらを入力後に追加[Add]ボタンを押します。

ライブラリに自身のオブジェクトを追加

左側に表示される「Deviceツリー」の最初の項目(ファイル名と同じ文字)を選択します。

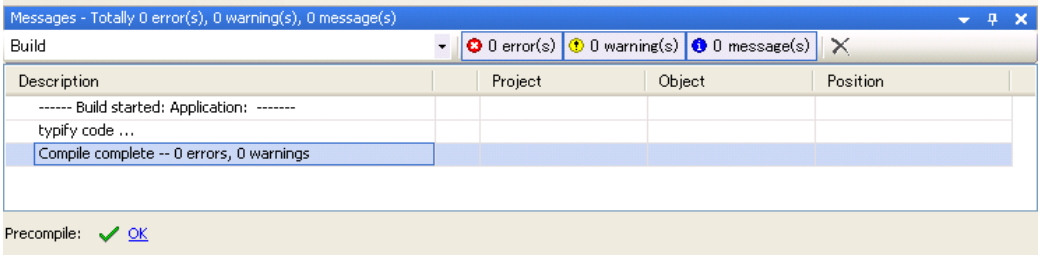
メニュー[Project][Add Object] で表示されるオブジェクトからDUTやPOUなどライブラリにしたいプログラムを追加していきます。



ライブラリのエラー確認

ライブラリの確認は[Build] [Build] ではなく[Build] [Check all Pool Objects] で行います。

もしエラーが検出されたなら修正し解消してください。



ライブラリの種類

ライブラリは作成時に使用しているライブラリプロジェクトのまま使用、配布することもできますが、ソースコードを公開しない「コンパイル済みライブラリ」を作成することができます。

ライブラリには次の形式があります。

- ライブラリプロジェクト (*.library)

  ソースコード参照可能ですが、プロテクトオプションで制御可能

- コンパイル済みライブラリ(*.compiled-library)

  ソースコード参照不可

「コンパイル済みライブラリ」の作成は、開発環境でライブラリプロジェクト(*.library)を開きメニュー [File] [Save Project As Compiled Library...] を押します。

ここで作成された「コンパイル済みライブラリ」は、自動的にリポジトリへの登録は行われていません。

このライブラリを使用するには次の「リポジトリ登録」作業を行う必要があります。

ライブラリの公開(リポジトリ登録)

作成したライブラリや配布されたライブラリをアプリケーションから使用するためにはライブラリ・リポジトリへの登録が必要となります。

ライブラリ・リポジトリへの登録は前述のどちらのライブラリ形式でも可能です。

- **外部から配布されたライブラリをリポジトリに登録するには**

  メニュー [Tool] [Library Repository...] で表示されるダイアログの [Install...] を押します。

  登録したいライブラリ(*.library あるいは *.compiled-library)を選択します。

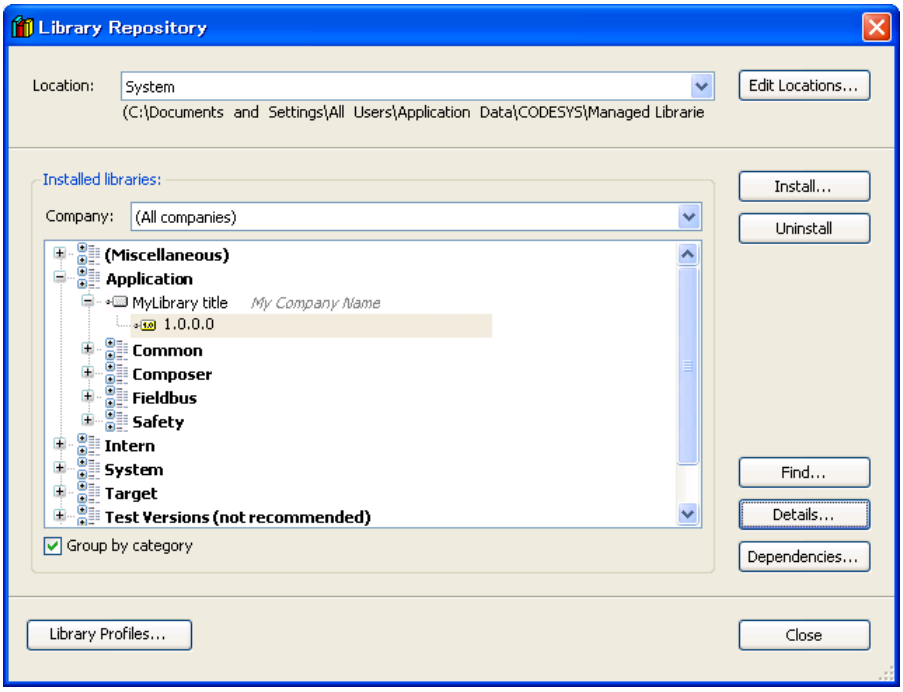
- **開発環境で現在開いているライブラリプロジェクトを登録するには**

  メニュー [File] [Save Project And Install Into Library Repository] を実行します。

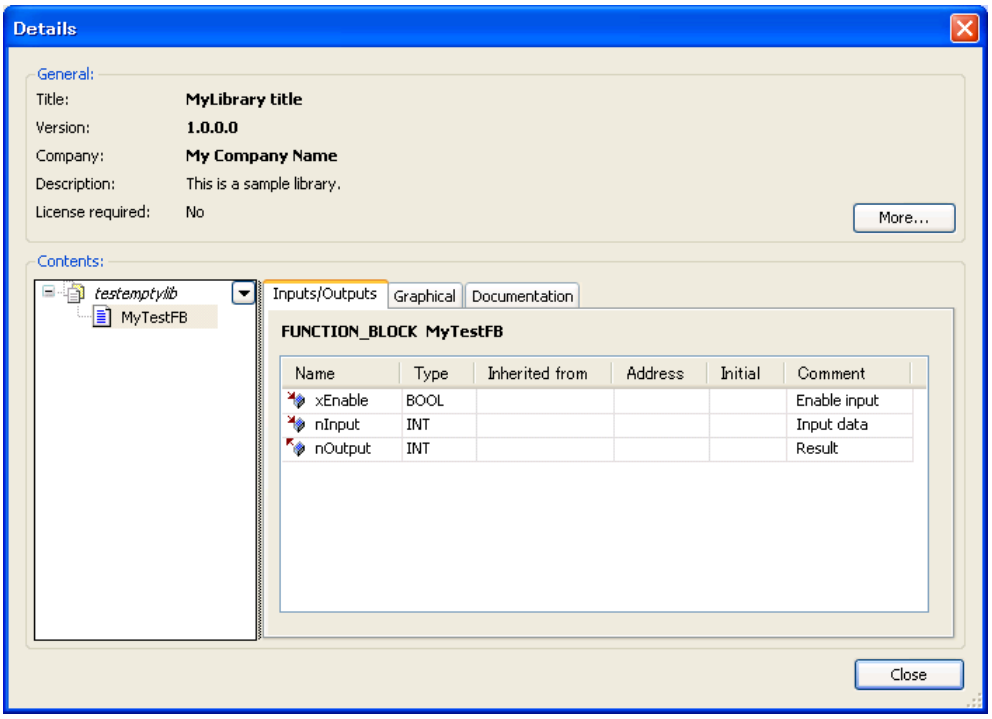
公開されているライブラリの確認

登録済みライブラリの情報は、「ライブラリマネージャ」か「ライブラリ・リポジトリ」画面で確認できます。

ここでは「ライブラリ・リポジトリ」画面での操作を紹介します。



先ほど作成したライブラリを登録した場合は、[Application]の下に[MyLibrary title]が登録されているのが確認できます。このライブラリの詳細情報を表示するためにバージョン[1.0.0.0]を選択し[Details...]ボタンを押します。

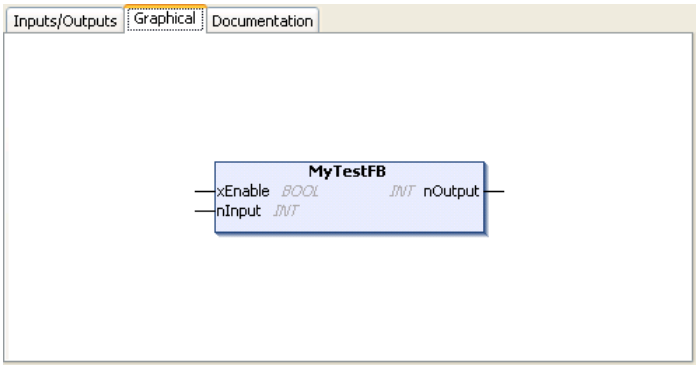


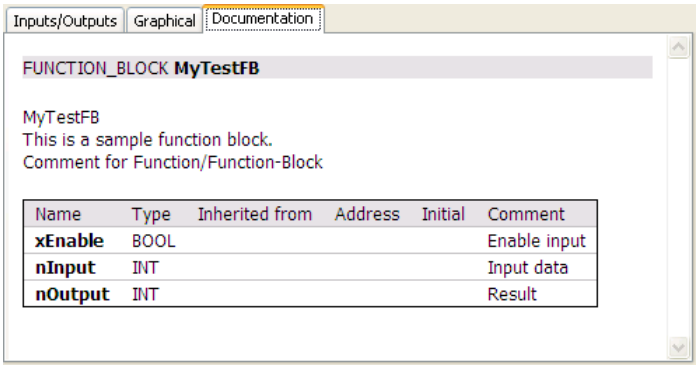
このライブラリの詳細情報が表示されます。

ツリー内にはライブラリ内で公開されているDUTやPOUなどが表示されます。

ここでは先で作成したファンクション・ブロック[MyTestFB]が確認できます。

ツリー内でファンクションブロック名を選択すると、その入力出力定義やグラフィックエディタでの表示や説明を確認できます。





ここではファンクション・ブロックを作成する際に記述したコメントが説明文の一部として現れます。プログラムする際には、この自動ドキュメント化機能を意識したコメントをプログラムと併せて記述することで個別ドキュメントを別途用意する工数の削減とプログラムとドキュメントの一元管理が容易となります。

8.2.演算子

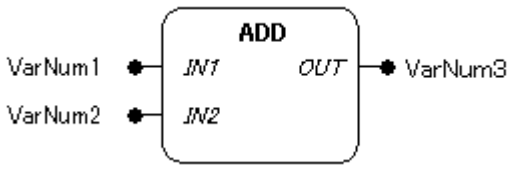
分類	命令	機能
算術演算	ADD	加算
	MUL	乗算
	SUB	減算
	DIV	除算後の商
	MOD	除算後の余り
	MOVE	転送
	INDEXOF *1	インデックスの取得
	SIZEOF *1	変数の占有サイズの取得
論理演算	AND	論理積
	OR	論理和
	XOR	ビットデータ排他的論理和
	NOT	ビットデータ反転
ビットシフト演算	SHL	左ビットシフト
	SHR	右ビットシフト
	ROL	左ビットローテーション
	ROR	右ビットローテーション
データ選択	SEL	データ選択
	MAX	最大値選択
	MIN	最小値選択
	LIMIT	上下限制限
	MUX	マルチプレクサ
比較演算	GT	IN1 ＞ IN2
	LT	IN1 ＜ IN2
	LE	IN1 ≦ IN2
	GE	IN1 ≧ IN2
	EQ	IN1 ＝ IN2
	NE	IN1 ≠ IN2
アドレス演算	ADR *1	アドレス取得
	BITADR *1	DWORD中のビットオフセット取得

分類	命令	機能
数値演算	ABS	絶対値
	SQRT	平方根
	LN	自然対数
	LOG	常用対数
	EXP	e の指数累乗
	SIN	サイン(入力はラジアン)
	COS	コサイン(入力はラジアン)
	TAN	タンジェント(入力はラジアン)
	ASIN	アークサイン(結果はラジアン)
	ACOS	アークコサイン(結果はラジアン)
	ATAN	アークタンジェント(結果はラジアン)
	EXPT	べき乗

*1) IEC61131-3 非準拠

ADD:加算

(FBD の例)



機能

入力パラメータに接続された値の加算結果(IN1 + IN2 = OUT)を出力します。

パラメータで使用可能なデータ型

BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL, LREAL, TIME, TIME_OF_DAY (TOD), DATE, DATE_AND_TIME (DT)

次の組み合わせも可能です: TIME+TIME=TIME, TOD+TIME=TOD, DT+TIME=DT

パラメータ

入力パラメータ	説明	備考
IN1 (VarNum1)	被加数	
IN2 (VarNum2)	加数	IN1と同じデータ型

出力パラメータ	説明	備考
OUT (VarNum3)	結果	IN1と同じデータ型

全てのパラメータは、同じデータ型でなければなりません。

FB/LDエディタにおけるADDオペレータは入力数を拡張できます。

(ILの例)

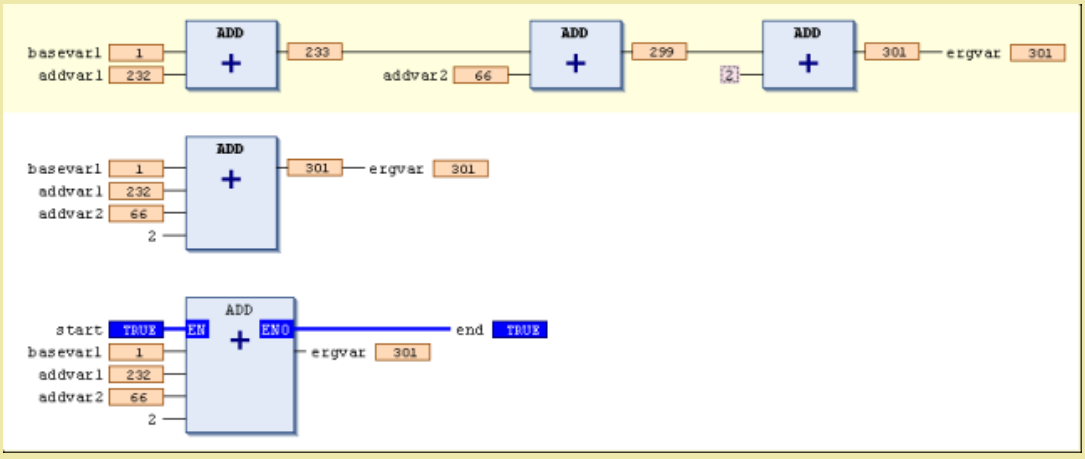
```
LD      7
ADD     2
ADD     4
ADD     7
ST      iVar
```

(STの例)

```
var1 := 7+2+4+7;
```

(FBDの例)

1.連続させたADD, 2.拡張したADD, 3.EN/ENO付のADD



MUL:乗算

(FBD の例)

機能

入力パラメータに接続された値の乗算結果(IN1 × IN2 = OUT)を出力します。

パラメータで使用可能なデータ型

BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL, LREAL, TIME

TIME型変数は整数型変数と乗算できます。

パラメータ

入力パラメータ	説明	備考
IN1 (VarNum1)	被乗数	
IN2 (VarNum2)	乗数	IN1と同じデータ型

出力パラメータ	説明	備考
OUT (VarNum3)	結果	IN1と同じデータ型

全てのパラメータは、同じデータ型でなければなりません。

FB/LDエディタにおけるMULオペレータは入力数を拡張できます。

(ILの例)

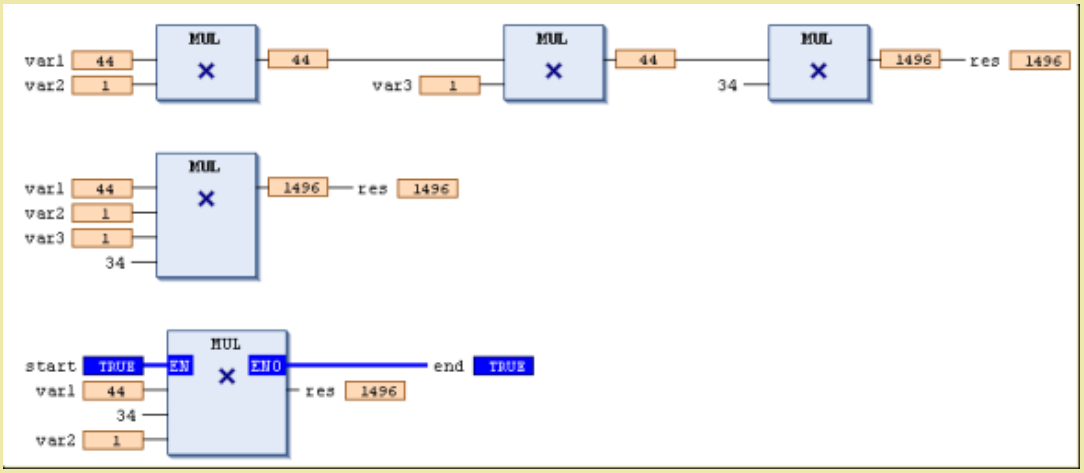
```
LD      7
MUL     2      ,
        4      ,
        7
ST      Var1
```

(STの例)

```
var1 := 7*2*4*7;
```

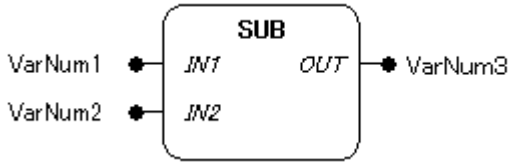
(FBDの例)

1.連続させたMUL, 2.拡張したMUL, 3.EN/ENO付のMUL



SUB:減算

(FBD の例)



機能

入力パラメータに接続された値の減算結果(IN1 - IN2 = OUT)を出力します。

パラメータで使用可能なデータ型

BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL, LREAL, TIME, TIME_OF_DAY (TOD), DATE, DATE_AND_TIME (DT)

次の組み合わせも可能です: TIME-TIME=TIME, DATE-DATE=TIME, TOD-TIME=TOD, TOD-TOD=TIME, DT-TIME=DT, DT-DT=TIME

負のTIME値は未定義と見なされます。

パラメータ

入力パラメータ	説明	備考
IN1 (VarNum1)	被減数	
IN2 (VarNum2)	減数	IN1と同じデータ型

出力パラメータ	説明	備考
OUT (VarNum3)	結果	IN1と同じデータ型

全てのパラメータは、同じデータ型でなければなりません。

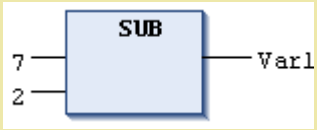
(ILの例)

```
LD   7
SUB  2
ST   Var1
```

(STの例)

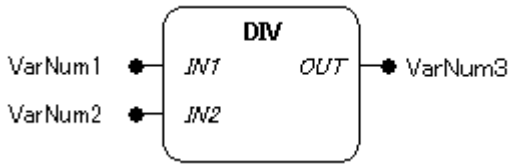
```
var1 := 7-2;
```

(FBDの例)



DIV: 除算

(FBD の例)



機能

入力パラメータに接続された値の除算結果(IN1 ÷ IN2 = OUT)を出力します。

パラメータで使用可能なデータ型

BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL, LREAL, TIME

TIME変数は整数変数で除算できます。

パラメータ

入力パラメータ	説明	備考
IN1 (VarNum1)	被除数	
IN2 (VarNum2)	除数	IN1と同じデータ型

出力パラメータ	説明	備考
OUT (VarNum3)	結果	IN1と同じデータ型

全てのパラメータは、同じデータ型でなければなりません。

補足

(ILの例) (結果Var1 の値は 4)

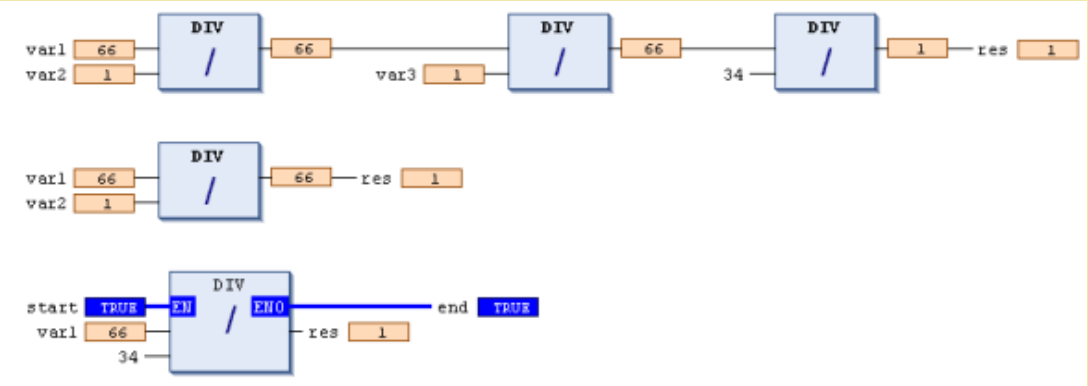
```
LD      8
DIV     2
ST      Var1
```

(STの例)

```
var1 := 8/2;
```

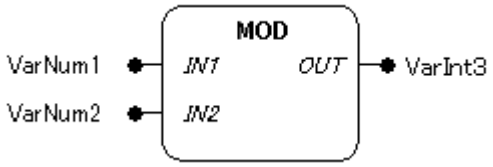
(FBDの例)

1.連続させたDIV, 2.単体でのDIV, 3.EN/ENO付のDIV



MOD:除算の余り

(FBD の例)



機能

入力パラメータに接続された値を除算した余り(IN1 MOD IN2 = OUT)を出力します。

パラメータで使用可能なデータ型

BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT

パラメータ

入力パラメータ	説明	備考
IN1 (VarNum1)	被除数	
IN2 (VarNum2)	除数	

出力パラメータ	説明	備考
OUT (VarInt3)	結果	整数データ型

補 足

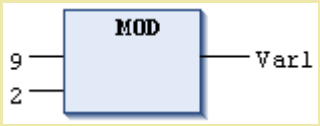
(ILの例) (結果: Var1 の値は 1)

```
LD      9
MOD     2
ST      Var1
```

(STの例)

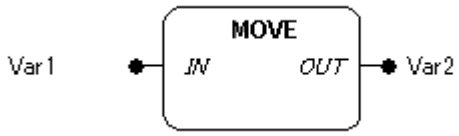
```
Var1 := 9 MOD 2;
```

(FBDの例)



MOVE:転送

(FBD の例)



機能

入力パラメータに接続された値を出力パラメータに接続された変数に転送(IN = OUT)します。

パラメータで使用可能なデータ型

全てのデータ型が使用できます。

パラメータ

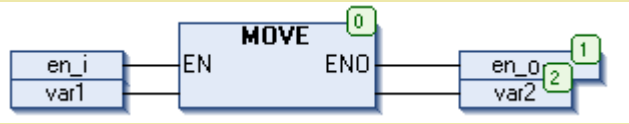
入力パラメータ	説明	備考
IN (Var1)	入力	

出力パラメータ	説明	備考
OUT (Var2)	出力	INと同じデータ型

全てのパラメータは、同じデータ型でなければなりません。

EN/ENO付のCFC 接続例：

en_i が TRUE の場合に var1 の値が var2 に代入されます。



(ILの例) (結果： var2 が var1 の値になります)

```
LD    var1
MOVE
ST    var2
```

次も同じ結果となります:

```
LD    var1
ST    var2
```

(STの例)

```
ivar2 := MOVE(ivar1);
```

(次も同じ結果となります: ivar2 := ivar1;)

INDEXOF：インデックスの取得

機能

与えられたオブジェクトのPOUインデックスを返します。

この演算子は使用されなくなります。代替えとしてADR演算子を使用してください。

SIZEOF：変数の占有サイズの取得

機能

与えられた変数が占有する記憶領域のサイズをバイト単位で返します。

パラメータ

入力パラメータ	説明	備考
-		

出力パラメータ	説明	備考
占有バイトサイズ	出力（返される数によりデータ型が決定）	返されるデータ型
	0 <= size of x < 256	USINT
	256 <= size of x < 65536	UINT
	65536 <= size of x < 4294967296	UDINT
	4294967296 <= size of x	ULINT

(ILの例) (結果 10)

```
arr1:ARRAY[0..4] OF INT;
Var1:INT;
```

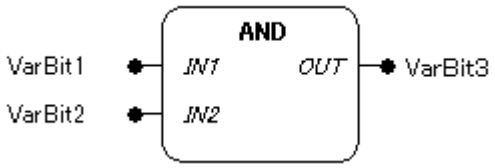
```
LD      arr1
SIZEOF
ST      Var1
```

(STの例)

```
var1 := SIZEOF(arr1);  (* 等価 var1:=USINT#10; *)
```

AND:論理積

(FBD の例)



機能

入力パラメータに接続された値の論理積結果(IN1 AND IN2 = OUT)を出力します。

パラメータで使用可能なデータ型

BOOL, BYTE, WORD, DWORD, LWORD

パラメータ

入力パラメータ	説明	備考
IN1 (VarBit1)	入力１	
IN2 (VarBit2)	入力２	IN1と同じデータ型

出力パラメータ	説明	備考
OUT (VarBit3)	結果	IN1と同じデータ型

全てのパラメータは、同じデータ型でなければなりません。

(ILの例) (結果 Var1 の値は 2#1000_0010)

```
Var1:BYTE;

LD      2#1001_0011
AND     2#1000_1010
ST      var1
```

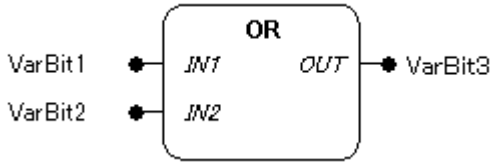
(STの例)

```
var1 := 2#1001_0011 AND 2#1000_1010
```

(FBDの例)

OR:論理和

(FBD での例)



機能

入力パラメータに接続された値の論理和結果(IN1 OR IN2 = OUT)を出力します。

パラメータで使用可能なデータ型

BOOL, BYTE, WORD, DWORD, LWORD

パラメータ

入力パラメータ	説明	備考
IN1 (VarBit1)	入力1	
IN2 (VarBit2)	入力2	IN1と同じデータ型

出力パラメータ	説明	備考
OUT (VarBit3)	結果	IN1と同じデータ型

全てのパラメータは、同じデータ型でなければなりません。

(ILの例) (結果 var1 の値は 2#1001_1011)

```
var1 :BYTE;
```

```
LD      2#1001_0011
OR      2#1000_1010
ST      Var1
```

(STの例)

```
Var1 := 2#1001_0011 OR 2#1000_1010;
```

(FBDの例)

XOR:排他的論理和

(FBD の例)

機能

入力パラメータに接続された値の排他的論理和結果(IN1 XOR IN2 = OUT)を出力します。

パラメータで使用可能なデータ型

BOOL, BYTE, WORD, DWORD, LWORD

パラメータ

入力パラメータ	説明	備考
IN1 (VarBit1)	入力1	

入力パラメータ	説明	備考
IN2 (VarBit2)	入力２	IN1と同じデータ型

出力パラメータ	説明	備考
OUT (VarBit3)	結果	IN1と同じデータ型

全てのパラメータは、同じデータ型でなければなりません。

(ILの例) (結果 var1 の値は 2#0001_1001)

```
LD      2#1001_0011
XOR     2#1000_1010
ST      var1
```

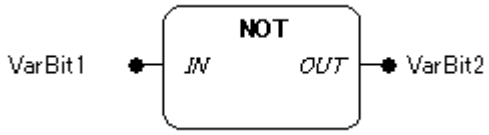
(STの例)

```
Var1 := 2#1001_0011 XOR 2#1000_1010;
```

(FBDの例)

NOT:ビットデータ反転

(FBD の例)



機能

入力パラメータに接続された値のビット単位の反転(NOT:1の補数)結果(IN NOT ＝ OUT)を出力します。

パラメータで使用可能なデータ型

BOOL, BYTE, WORD, DWORD, LWORD

パラメータ

入力パラメータ	説明	備考
IN (VarBit1)	入力	

出力パラメータ	説明	備考
OUT (VarBit2)	結果	INと同じデータ型

全てのパラメータは、同じデータ型でなければなりません。

(ILの例) (結果 Var1 の値は 2#0110_1100)

```
Var1 :BYTE;
```

```
LD      2#1001_0011
NOT
ST      var1
```

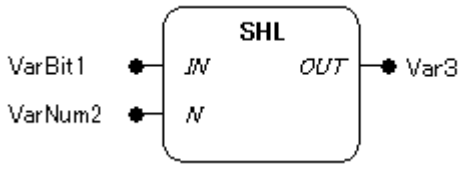
(STの例)

```
Var1 := NOT 2#1001_0011 ;
```

(FBDの例)

SHL: 左ビットシフト

(FBD の例)



機能

入力パラメータINに接続した値を、Nに接続したビット数だけ左シフトした結果を出力します。

パラメータで使用可能なデータ型

BYTE, WORD, DWORD, LWORD

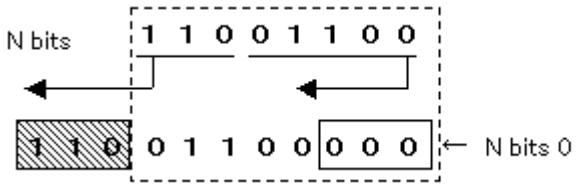
パラメータ

入力パラメータ	説明	備考
IN (VarBit1)	シフトされるデータ	ここに与えるデータ型のサイズが演算の最大ビット数となり、 ここに定数を与えた場合その値が格納できる最小データ型の サイズで演算されます。
N (VarNum2)	シフトするビット数	0から入力パラメータINのデータ型の最大ビット数までが有効であり、 負の値を与えるとシフト方向が逆になります。

出力パラメータ	説明	備考
OUT (Var3)	結果	このデータ型のサイズはシフト演算に影響しません。 INデータ型が符号付きであれば符号付きのデータ型とする

解説

左シフト命令では指定のNビットだけ左へシフトすると共に右のビットに0を補充します。



データ型がWORDの場合

2#0000_0000_0100_1011の2ビット左シフトの結果は2#0000_0001_0010_1100

2#0000_0000_0100_1011の4ビット左シフトの結果は2#0000_0100_1011_0000

データ型がBYTE の場合

2#0100_1011 の2 ビット左シフトの結果は2#0010_1100

2#0100_1011 の4 ビット左シフトの結果は2#1011_0000

(ILの例)

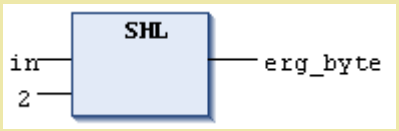
```
LD    in_byte
SHL   2
ST    erg_byte
```

(STの例)

次の例は入力変数(BYTEまたはWORD)のデータ型に対応したerg_byteとerg_wordのそれぞれの結果を16進で表しています。

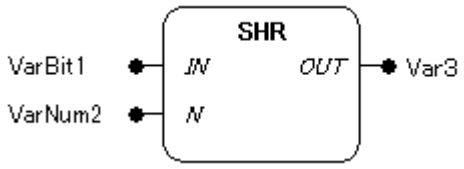
```
PROGRAM shl_st
VAR
    in_byte : BYTE:=16#45; (* 2#01000101 )
    in_word : WORD:=16#0045; (* 2#0000000001000101 )
    erg_byte : BYTE;
    erg_word : WORD;
    n: BYTE :=2;
END_VAR
erg_byte:=SHL(in_byte,n); (* 結果 16#14, 2#00010100 *)
erg_word:=SHL(in_word,n); (* 結果 16#0114, 2#0000000100010100 *)
```

(FBDの例)



SHR:右ビットシフト

(FBD の例)



機能

入力パラメータINに接続した値を、Nに接続したビット数だけ右シフトした結果を出力します。

パラメータで使用可能なデータ型

BYTE, WORD, DWORD, LWORD

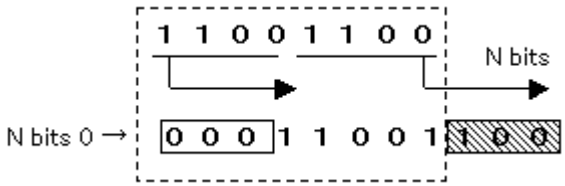
パラメータ

入力パラメータ	説明	備考
IN (VarBit1)	シフトされるデータ	ここに与えるデータ型のサイズが演算の最大ビット数となり、ここに定数を与えた場合その値が格納できる最小データ型のサイズで演算されます。
N (VarNum2)	シフトするビット数	0から入力パラメータINのデータ型の最大ビット数までが有効であり、負の値を与えるとシフト方向が逆になります。

出力パラメータ	説明	備考
OUT (Var3)	結果	このデータ型のサイズはシフト演算に影響しません。INデータ型が符号付きであれば符号付きのデータ型とする

解説

右シフト命令では指定のNビットだけ右へシフトすると共に左のビットに0を補充します。



データ型がWORDの場合

2#0100_1011_0000_0000の2ビット右シフトの結果は2#0001_0010_1100_0000

2#0100_1011_0000_0000 の4 ビット右シフトの結果は2#0000_0100_1011_0000

データ型がBYTE の場合

2#0100_1011 の2 ビット右シフトの結果は2#0001_0010

2#0100_1011 の4 ビット右シフトの結果は2#0000_0100

(ILの例)

```
LD      in_byte
SHR     2
ST      erg_byte
```

(STの例)

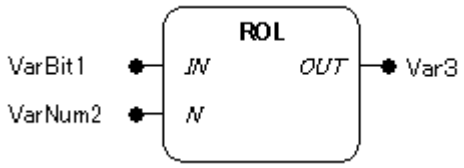
次の例は入力変数(BYTEまたはWORD)のデータ型に対応したerg_byteとerg_wordのそれぞれの結果を16進で表しています。

```
PROGRAM shr_st
VAR
    in_byte : BYTE:=16#45; (* 2#01000101 )
    in_word : WORD:=16#0045; (* 2#0000000001000101 )
    erg_byte : BYTE;
    erg_word : WORD;
    n: BYTE :=2;
END_VAR
erg_byte:=SHR(in_byte,n); (* 結果 16#11, 2#00010001 *)
erg_word:=SHR(in_word,n); (* 結果 16#0011, 2#0000000000010001 *)
```

(FBDの例)

ROL : 左ビットローテーション

(FBD の例)



機能

入力パラメータINに接続した値を、Nに接続したビット数だけ左ローテーションした結果を出力します。

パラメータで使用可能なデータ型

BYTE, WORD, DWORD, LWORD

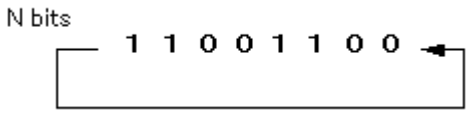
パラメータ

入力パラメータ	説明	備考
IN (VarBit1)	ローテーションされるデータ	ここに与えるデータ型のサイズが演算の最大ビット数となり、ここに定数を与えた場合その値が格納できる最小データ型のサイズで演算されます。
N (VarNum2)	ローテーションするビット数	0から入力パラメータINのデータ型の最大ビット数までが有効であり、負の値を与えるとシフト方向が逆になります。

出力パラメータ	説明	備考
OUT (Var3)	結果	このデータ型のサイズはシフト演算に影響しません。 INデータ型が符号付きであれば符号付きのデータ型とする

解説

左ローテーション命令では指定のNビットだけ左へローテーションします。



データ型がWORD の場合

2#0100_1011_0000_0000 の2 ビット 左ローテーションの結果は 2#0010_1100_0000_0001

2#0100_1011_0000_0000 の4 ビット 左ローテーションの結果は 2#1011_0000_0000_0100

データ型がBYTE の場合

2#0100_1011 の2 ビット 左ローテーションの結果は 2#0010_1101

2#0100_1011 の4 ビット 左ローテーションの結果は 2#1011_0100

(ILの例)

```
LD      in_byte
ROL     n
ST      erg_byte
```

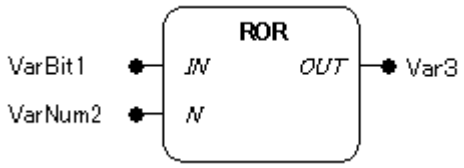
(STの例)

```
PROGRAM rol_st
VAR
    in_byte : BYTE:=16#45;
    in_word : WORD:=16#45;
    erg_byte : BYTE;
    erg_word : WORD;
    n: BYTE :=2;
END_VAR
erg_byte:=ROL(in_byte,n); (* 結果 16#15 *)
erg_word:=ROL(in_word,n); (* 結果 16#0114 *)
```

(FBDの例)

ROR：右ビットローテーション

(FBD の例)



機能

入力パラメータINに接続した値を、Nに接続したビット数だけ右ローテーションした結果を出力します。

パラメータで使用可能なデータ型

BYTE, WORD, DWORD, LWORD

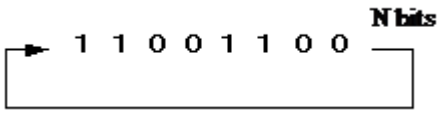
パラメータ

入力パラメータ	説明	備考
IN (VarBit1)	ローテーションされるデータ	ここに与えるデータ型のサイズが演算の最大ビット数となり、 ここに定数を与えた場合その値が格納できる最小データ型の サイズで演算されます。
N (VarNum2)	ローテーションするビット数	0から入力パラメータINのデータ型の最大ビット数までが有効であり、 負の値を与えるとシフト方向が逆になります。

出力パラメータ	説明	備考
OUT (Var3)	結果	このデータ型のサイズはシフト演算に影響しません。 INデータ型が符号付きであれば符号付きのデータ型とする

解説

右ローテーション命令では指定のNビットだけ右へローテーションします。



データ型がWORDの場合

2#0000_0000_0100_1011 の2 ビット右ローテーションの結果は2#1100_0000_0001_0010

2#0000_0000_0100_1011 の4 ビット右ローテーションの結果は2#1011_0000_0000_0100

データ型がBYTE の場合

2#0100_1011 の2 ビット右ローテーションの結果は2#1101_0010

2#0100_1011 の4 ビット右ローテーションの結果は2#1011_0100

(ILの例)

```
LD      in_byte
ROR     n
ST      erg_byte
```

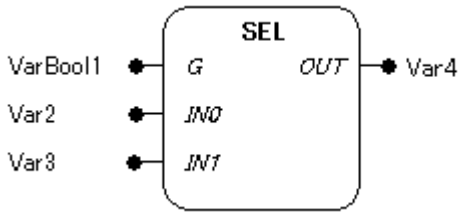
(STの例)

```
PROGRAM ror_st
VAR
    in_byte : BYTE:=16#45;
    in_word : WORD:=16#45;
    erg_byte : BYTE;
    erg_word : WORD;
    n: BYTE :=2;
END_VAR
erg_byte:=ROR(in_byte,n); (* 結果 16#51 *)
erg_word:=ROR(in_word,n); (* 結果 16#4011 *)
```

(FBDの例)

SEL：データ選択

(FBD の例)



機能

選択入力 G の値に従い、2つの入力 IN0、IN1 のどちらかを OUT に出力します。選択入力 G が FALSE なら IN0 が出力され TRUE なら IN1 が出力されます。

パラメータで使用可能なデータ型

IN0, IN1, OUT：全ての型

G：BOOL型

パラメータ

入力パラメータ	説明	備考
G (VarBool1)	選択入力	
IN0 (Var2)	G = FALSEのとき出力する値	
IN1 (Var3)	G = TRUEのとき出力する値	IN0 と同じデータ型

出力パラメータ	説明	備考
OUT (Var4)	結果	IN0 と同じデータ型

入力 IN0、IN1 と出力 OUT は、同じデータ型でなければなりません。

解説

G ＝ FALSE のとき OUT ← IN0

G ＝ TRUE のとき OUT ← IN1

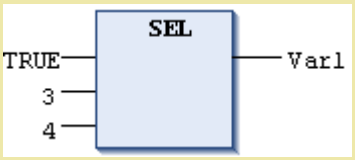
(ILの例)

```
     LD   TRUE
     SEL  3,4     (* IN0 = 3, IN1 =4 *)
     ST   Var1    (* 結果は 4 *)
     LD   FALSE
     SEL  3,4
     ST   Var1    (* 結果は 3 *)
```

(STの例)

```
     Var1:=SEL(TRUE,3,4);  (* 結果は 4 *)
```

(FBDの例)

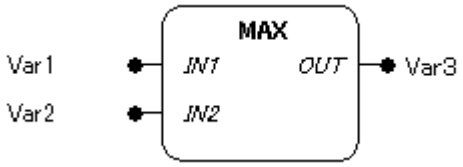


注 意

N0がTRUEの場合はIN0の計算は処理されません。またN0がFALSEの場合はIN1の計算は処理されません。

MAX:最大値選択

(FBD の例)



機能

入力パラメータに接続された値のうちの最も大きな値を出力します。

パラメータで使用可能なデータ型

全ての型

パラメータ

入力パラメータ	説明	備考
IN1 (Var1)	入力１	
IN2 (Var2)	入力２	IN1 と同じデータ型

出力パラメータ	説明	備考
OUT (Var3)	結果	IN1 と同じデータ型

全てのパラメータは、同じデータ型でなければなりません。

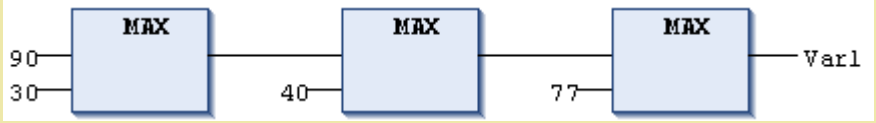
(ILの例) (結果は 90)

```
LD    90
MAX   30
MAX   40
MAX   77
ST    Var1
```

(STの例)

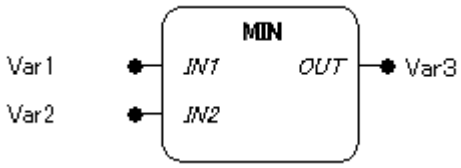
```
Var1:=MAX(30,40);           (* 結果 40 *)
Var1:=MAX(40,MAX(90,30));   (* 結果 90 *)
```

(FBDの例)



MIN:最小値選択

(FBD の例)



機能

入力パラメータに接続された値のうち最も小さな値を出力します。

パラメータで使用可能なデータ型

全ての型

パラメータ

入力パラメータ	説明	備考
IN1 (Var1)	入力１	
IN2 (Var2)	入力２	IN1 と同じデータ型

出力パラメータ	説明	備考
OUT (Var3)	結果	IN1 と同じデータ型

全てのパラメータは、同じデータ型でなければなりません。

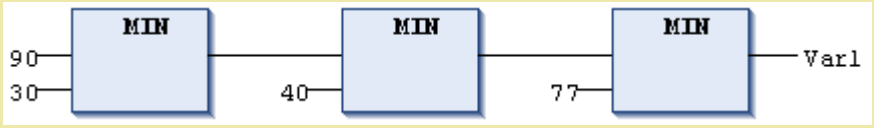
(ILの例) (結果は 30)

```
LD      90
MIN     30
MIN     40
MIN     77
ST      Var1
```

(STの例)

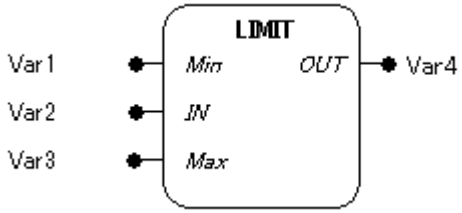
```
Var1:=MIN(90,30);           (* 結果 30 *)
Var1:=MIN(MIN(90,30),40);   (* 結果 30 *);
```

(FBDの例)



LIMIT:上下限制限

(FBD の例)



機能

入力パラメータ IN の値を、入力パラメータ Min(下限値)と Max(上限値)により範囲を制限します。

パラメータで使用可能なデータ型

全ての型

パラメータ

入力パラメータ	説明	備考
Min (Var1)	下限値	INと同じデータ型
IN (Var2)	入力値	
Max (Var3)	上限値	INと同じデータ型

出力パラメータ	説明	備考
OUT (Var4)	結果	INと同じデータ型

全てのパラメータは、同じデータ型でなければなりません。

解説

入力パラメータ IN の値を以下のように制限します。

Min ≦ IN ≦ Max の場合 OUT = IN.

IN ＜ Min の場合 OUT = Min.

IN ＞ Max の場合 OUT = Max.

OUT := LIMIT(Min, IN, Max) とOUT := MIN (MAX (IN, Min), Max) は同じ意味です。

(ILの例) (結果 80)

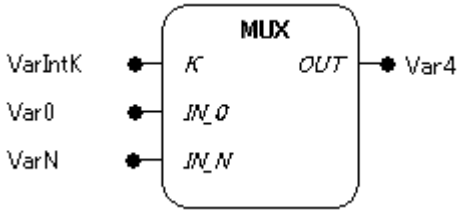
```
LD      90
LIMIT   30 ,
        80
ST      Var1
```

(STの例)

```
Var1 := LIMIT(30,90,80);  (* 結果 80 *)
```

MUX：マルチプレクサ

(FBD の例)



機能

入力パラメータ IN_0 から IN_N の値を入力パラメータ K により選択し出力します。

パラメータで使用可能なデータ型

K：BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, LINT, ULINT, UDINT

IN_0, IN_N, OUT：全ての型

パラメータ

入力パラメータ	説明	備考
K (VarIntK)	選択番号	0 ～ N
IN_0 (Var0)	入力0	
IN_N (VarN)	入力N	IN_0 と同じデータ型

出力パラメータ	説明	備考
OUT (Var4)	結果	IN_0 と同じデータ型

(ILの例) (結果 30)

```
LD      0
MUX     30  ,
        40  ,
        50  ,
        60  ,
        70  ,
        80
ST      Var1
```

(STの例)

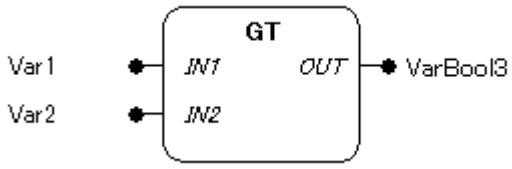
```
Var1:=MUX(0,30,40,50,60,70,80);  (* 結果30 *)
```

注 意

入力パラメータKとKにより選択されている入力パラメータ以外の入力パラメータ式は、実行時間を節約するために処理されません！ しかしシミュレーションモードだけは、すべての式が実行されます。

GT : 比較＞(Grater Than)

(FBD の例)



機能

入力パラメータの比較結果(IN1 > IN2)が成立ならTRUE、それ以外はFALSEを出力します。

パラメータで使用可能なデータ型

IN1, IN2: 全ての基本型

OUT: BOOL

パラメータ

入力パラメータ	説明	備考
IN1 (Var1)	入力1	
IN2 (Var2)	入力2	IN1と同じデータ型

出力パラメータ	説明	備考
OUT (VarBool3)	結果	TRUE: IN1 ＞ IN2 FALSE:IN1 ≦ IN2

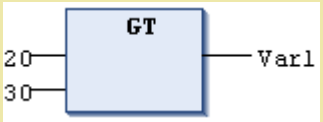
(ILの例) (結果は FALSE)

```
LD    20
GT    30
ST    Var1
```

(STの例)

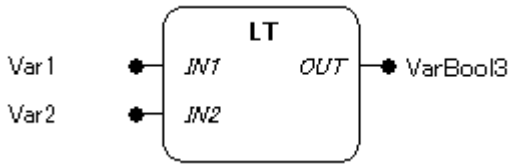
```
Var1 := 20 > 30;
```

(FBDの例)



LT: 比較＜(Less Than)

(FBD の例)



機能

入力パラメータの比較結果(IN1 < IN2)が成立ならTRUE、それ以外はFALSEを出力します。

パラメータで使用可能なデータ型

IN1, IN2: 全ての基本型

OUT: BOOL

パラメータ

入力パラメータ	説明	備考
IN1 (Var1)	入力1	
IN2 (Var2)	入力2	IN1と同じデータ型

出力パラメータ	説明	備考
OUT (Var3Bool)	結果	TRUE: IN1 ＜ IN2 FALSE:IN1 ≧ IN2

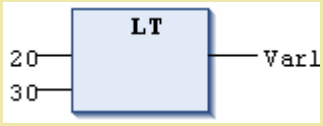
(ILの例) (結果は TRUE)

```
LD      20
LT      30
ST      Var1
```

(STの例)

```
Var1 := 20 < 30;
```

(FBDの例)



LE:比較≦(Less or Equal)

(FBD の例)

機能

入力パラメータの比較結果(IN1 ≦ IN2)が成立ならTRUE、それ以外はFALSEを出力します。

パラメータで使用可能なデータ型

IN1, IN2: 全ての基本型

OUT: BOOL

パラメータ

入力パラメータ	説明	備考
IN1 (Var1)	入力1	
IN2 (Var2)	入力2	IN1と同じデータ型

出力パラメータ	説明	備考
OUT (VarBool3)	結果	TRUE: IN1 ≦ IN2 FALSE:IN1 > IN2

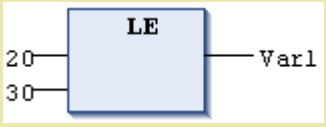
(ILの例) (結果は TRUE)

```
LD      20
LE      30
ST      Var1
```

(STの例)

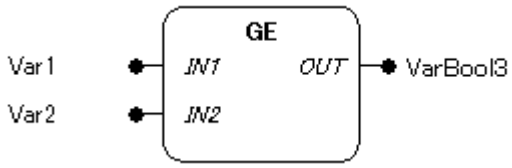
```
Var1 := 20 <= 30;
```

(FBDの例)



GE: 比較≧(Grater or Equal)

(FBD の例)



機能

入力パラメータの比較結果(IN1 ≧ IN2)が成立ならTRUE、それ以外はFALSE を出力します。

パラメータで使用可能なデータ型

IN1, IN2: 全ての基本型

OUT: BOOL

パラメータ

入力パラメータ	説明	備考
IN1 (Var1)	入力１	
IN2 (Var2)	入力２	IN1と同じデータ型

出力パラメータ	説明	備考
OUT (VarBool3)	結果	TRUE: IN1 ≧ IN2 FALSE:IN1 ＜ IN2

(ILの例) (結果は TRUE)

```
LD      60
GE      40
ST      Var1
```

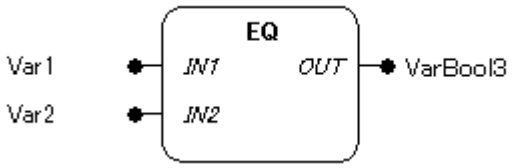
(STの例)

```
Var1 := 60 >= 40;
```

(FBDの例)

EQ: 比較＝(Equal)

(FBD の例)



機能

入力パラメータの比較結果(IN1 = IN2)が成立なら TRUE、それ以外は FALSE を出力します。

パラメータで使用可能なデータ型

IN1, IN2: 全ての基本型

OUT: BOOL

パラメータ

入力パラメータ	説明	備考
IN1 (Var1)	入力1	
IN2 (Var2)	入力2	IN1と同じデータ型

出力パラメータ	説明	備考
OUT (VarBool3)	結果	TRUE: IN1 = IN2 FALSE:IN1 ≠ IN2

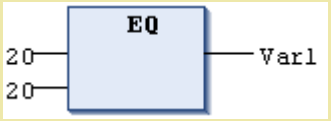
(ILの例) (結果は TRUE)

```
LD    40
EQ    40
ST    Var1
```

(STの例)

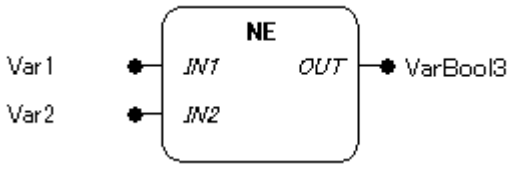
```
Var1 := 40 = 40;
```

(FBDの例)



NE: 比較≠(Not Equal)

(FBD の例)



機能

入力パラメータの比較結果(IN1≠IN2)が成立ならTRUE、それ以外はFALSEを出力します。

パラメータで使用可能なデータ型

IN1, IN2: 全ての基本型

OUT: BOOL

パラメータ

入力パラメータ	説明	備考
IN1 (Var1)	入力1	
IN2 (Var2)	入力2	IN1と同じデータ型

出力パラメータ	説明	備考
OUT (VarBool3)	結果	TRUE: IN1 ≠ IN2 FALSE:IN1 ＝ IN2

(ILの例) (結果は FALSE)

```
LD    40
NE    40
ST    Var1
```

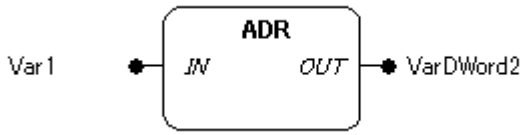
(STの例)

```
Var1 := 40 <> 40;
```

(FBDの例)

ADR：アドレス取得

(FBD の例)



機能

入力パラメータに接続された変数、プログラム、ファンクション、ファンクションブロック名やメソッド名を指定でき、それぞれのオブジェクトが配置されたメモリアドレスをDWORD型で返します。

パラメータで使用可能なデータ型

IN：変数、プログラム、ファンクション、ファンクションブロック名やメソッド名

OUT：DWORD

パラメータ

入力パラメータ	説明	備考
IN (Var1)	変数、インスタンス	

出力パラメータ	説明	備考
OUT (VarDWord2)	結果	POINTER

(STの例)

```
VAR
     pt:POINTER TO INT;   (* ポインタptの宣言 *)
     var_int1:INT := 5;   (* 変数 var_int1, var_int2 の宣言 *)
```

```
        var_int2:INT;

    END_VAR
    pt := ADR(var_int1); (* var_int1のアドレスがポインタptに代入されます *)
    var_int2:= pt^;      (*ポインタptの参照によりvar_int1の値5がvar_int2に代入されます *)
```

注意

- オンライン変更の場合には、変数がメモリ内の別の場所に移動される可能性があります。

  (コピーが必要な場合、オンライン変更時に表示があります)

  ポインタ変数は、そのような後に無効なメモリを指すことがあります。

  この問題を回避するには、サイクルを超えて保持しないよう各サイクルでポインタの新しい値を取得します。

- 関数やメソッドのPOINTER TO変数は、呼び元の関数に返したり、グローバル変数に渡すべきではありません。

BITADR:ビットオフセット取得

(FBDの例)

機能

入力パラメータに接続された変数のメモリ領域(セグメント)内に配置された位置のビットオフセットをDWORD型で返します。オフセット値はターゲット設定のバイトアドレス指定がされているかどうかに依存します。返される値のDWORD上位4ビットはメモリ領域を示しています。

パラメータで使用可能なデータ型

IN: BOOL型変数
OUT: DWORD

パラメータ

入力パラメータ	説明	備考
IN (Var1)	BOOL型変数	

出力パラメータ	説明	備考
OUT (VarDWord2)	結果	ビットオフセット

補 足

DWORD上位４ビットで返されるメモリ領域 :

Memory:16#40000000

Input:16#80000000

Output:16#C0000000

(STの例)

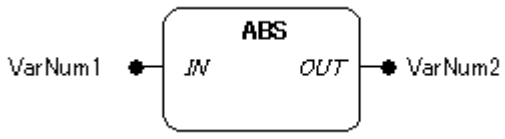
```
VAR
     var1 AT %IX2.3:BOOL;
     bitoffset: DWORD;
END_VAR

bitoffset:=BITADR(var1); (* 結果 BYTE Addresssing=TRUE: 16#80000013 *)
                         (* BYTE Addressing=FALSE: 16#80000023 *)
```

注 意

オンライン変更後の値は変更があるかもしれません。アドレスのポインタとして使用している場合は注意が必要となります。

ABS:絶対値

(FBD の例)



機能

入力パラメータに接続された値 IN の絶対値(｜IN｜ = OUT)を計算します。

パラメータで使用可能なデータ型

IN, OUT: 数値型

パラメータ

入力パラメータ	説明	備考
IN (VarNum1)	入力	

出力パラメータ	説明	備考
OUT (VarNum2)	結果	INと同じデータ型

(ILの例) (結果 2)

```
LD      -2
ABS
ST      i
```

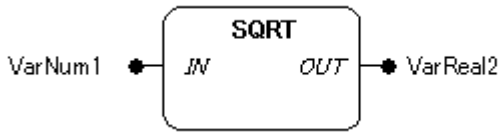
(STの例)

```
i:=ABS(-2);
```

(FBDの例)

SQRT: 平方根(Square Root)

(FBD の例)



機能

入力パラメータに接続された値 IN の平方根を計算します。

パラメータで使用可能なデータ型

IN: 数値型

OUT: REAL, LREAL

パラメータ

入力パラメータ	説明	備考
IN (VarNum1)	入力	

出力パラメータ	説明	備考
OUT (VarReal2)	結果	

(ILの例) (結果q の値は 4)

```
LD      16
SQRT
ST      q
```

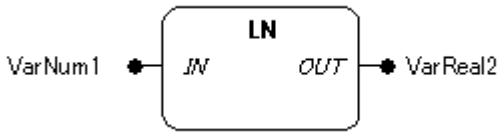
(STの例)

```
q:=SQRT(16);
```

(FBDの例)

LN : 自然対数 (Natural Logarithm)

(FBD の例)



機能

入力パラメータに接続された値 IN の自然対数(底＝e)を計算します。

パラメータで使用可能なデータ型

IN: 数値型

OUT: REAL, LREAL

パラメータ

入力パラメータ	説明	備考
IN (VarNum1)	入力	

出力パラメータ	説明	備考
OUT (VarReal2)	結果	

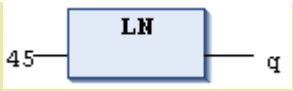
(ILの例) (結果 q は 3.80666)

```
LD      45
LN
ST      q
```

(STの例)

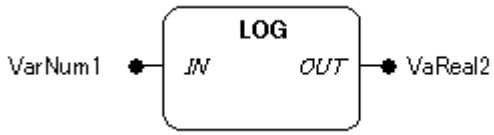
```
q:=LN(45);
```

(FBDの例)



LOG: 常用対数(Logerithm)

(FBD の例)



機能

入力パラメータに接続された値 IN の常用対数(底＝10)を計算します。

パラメータで使用可能なデータ型

IN: 数値型

OUT: REAL, LREAL

パラメータ

入力パラメータ	説明	備考
IN (VarNum1)	入力	

出力パラメータ	説明	備考
OUT (VarReal2)	結果	

(ILの例) (結果 q は 2.49762)

```
LD      314.5
LOG
ST      q
```

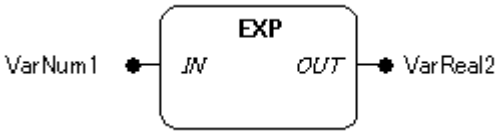
(STの例)

```
q:=LOG(314.5);
```

(FBDの例)

EXP:e の指数累乗(Exponential)

(FBD の例)



機能

入力パラメータに接続された値 IN の自然指数関数を計算します。

パラメータで使用可能なデータ型

IN: 数値型

OUT: REAL, LREAL

パラメータ

入力パラメータ	説明	備考
IN (VarNum1)	入力	ベースeの指数

出力パラメータ	説明	備考
OUT (VarReal2)	結果	

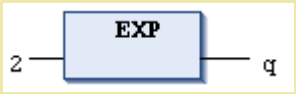
(ILの例) (結果 q は 7.389056099)

```
LD    2
EXP
ST    q
```

(STの例)

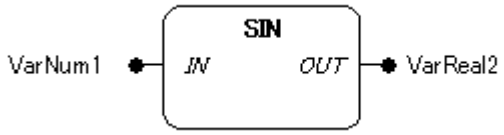
```
q:=EXP(2);
```

(FBDの例)



SIN : サイン(Sine)

(FBD の例)



機能

入力パラメータに接続された値 IN (ラジアン単位)のサイン(正弦)を計算します。

パラメータで使用可能なデータ型

IN: 数値型

OUT: REAL, LREAL

パラメータ

入力パラメータ	説明	備考
IN (VarNum1)	入力	ラジアン

出力パラメータ	説明	備考
OUT (VarReal2)	結果	

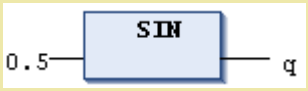
(ILの例) (結果 q は 0.479426)

```
LD    0.5
SIN
ST    q
```

(STの例)

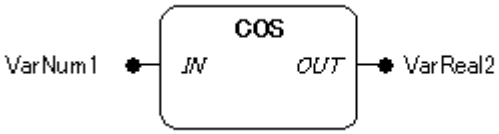
```
q:=SIN(0.5);
```

(FBDの例)



COS:コサイン(Cosine)

(FBD の例)



機能

入力パラメータに接続された値 IN (ラジアン単位)のコサイン(余弦)を計算します。

パラメータで使用可能なデータ型

IN: 数値型

OUT: REAL, LREAL

パラメータ

入力パラメータ	説明	備考
IN (VarNum1)	入力	ラジアン

出力パラメータ	説明	備考
OUT (VarReal2)	結果	

(ILの例) (結果 q は 0.877583)

```
LD      0.5
COS
ST      q
```

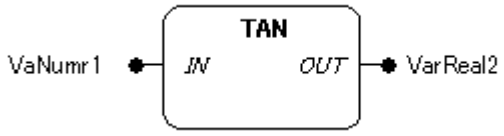
(STの例)

```
q:=COS(0.5);
```

(FBDの例)

TAN : タンジェント(Tangent)

(FBD の例)



機能

入力パラメータに接続された値 IN (ラジアン単位)のタンジェント(接線)を計算します。

パラメータで使用可能なデータ型

IN: 数値型

OUT: REAL, LREAL

パラメータ

入力パラメータ	説明	備考
IN (VarNum1)	入力	ラジアン

出力パラメータ	説明	備考
OUT (VarReal2)	結果	

(ILの例) (結果 q は 0.546302)

```
LD      0.5
TAN
ST      q
```

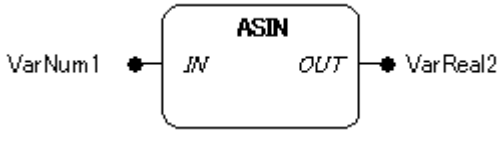
(STの例)

```
q:=TAN(0.5);
```

(FBDの例)

ASIN: アークサイン(Arc Sine)

(FBD の例)



機能

入力パラメータに接続された値 IN のアークサインの主値(ラジアン単位)を計算します。

パラメータで使用可能なデータ型

IN: 数値型

OUT: REAL, LREAL

パラメータ

入力パラメータ	説明	備考
IN (VarNum1)	入力	-1.0 ～ 1.0

出力パラメータ	説明	備考
OUT (VarReal2)	結果	ラジアン

(ILの例) (結果 q は 0.523599)

```
LD      0.5
ASIN
ST      q
```

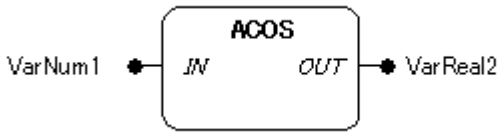
(STの例)

```
q:=ASIN(0.5);
```

(FBDの例)

ACOS: アークコサイン(Arc Cosine)

(FBD の例)



機能

入力パラメータに接続された値 IN のアークコサインの主値(ラジアン単位)を計算します。

パラメータで使用可能なデータ型

IN: 数値型

OUT: REAL, LREAL

パラメータ

入力パラメータ	説明	備考
IN (VarNum1)	入力	-1.0 ～ 1.0

出力パラメータ	説明	備考
OUT (VarReal2)	結果	ラジアン

(ILの例) (結果 q は 1.0472)

```
LD      0.5
ACOS
ST      q
```

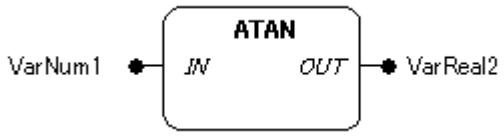
(STの例)

```
q:=ACOS(0.5);
```

(FBDの例)

ATAN：アークタンジェント(Arc Tangent)

(FBD の例)



機能

入力パラメータに接続された値 IN のアークタンジェントの主値(ラジアン単位)を計算します。

パラメータで使用可能なデータ型

IN: 数値型

OUT: REAL, LREAL

パラメータ

入力パラメータ	説明	備考
IN (VarNum1)	入力	-1.0 ～ 1.0

出力パラメータ	説明	備考
OUT (VarReal2)	結果	ラジアン

(ILの例) (結果 q は 0.463648)

```
LD      0.5
ATAN
ST      q
```

(STの例)

```
q:=ATAN(0.5);
```

(FBDの例)

EXPT: べき乗算

(FBD の例)

機能

入力パラメータ IN1 に接続された値を IN2 に接続された整数値によるべき乗算結果を出力します。

パラメータで使用可能なデータ型

IN1, IN2: 数値型

OUT: REAL, LREAL

パラメータ

入力パラメータ	説明	備考
IN1 (VarNum1)	ベース	
IN2 (VarInt2)	指数	

出力パラメータ	説明	備考
OUT (VarReal3)	結果	

(ILの例) (結果 49)

```
LD      7
EXPT    2
ST      Var1
```

(STの例)

```
Var1 := EXPT(7,2);
```

(FBDの例)

8.3.呼び出し演算子

ファンクションブロックの呼び出しで使用されます。

分類	命令	機能
呼び出し演算	CAL	IL言語ファンクションブロック呼び出し

CAL:呼び出し

機能

IL言語で指定されたファンクション・ブロックのインスタンスを呼び出します。

(ILの例)

```
CAL     SRinst  (
    SET1:= VarBool1 ,
  RESET:= VarBool2 )
LD      SRinst.Q1
ST      VarBool3
```

8.4.型変換演算子

大きな型から小さな型への変換は明示的な型変換が必要となります。例えばDINTからINTや、WORDからBYTEへの変換です。

サイズや負号の異なるデータ型へ変換する場合は、情報を失う恐れがありますので注意が必要です。

ある基本型から別の基本型へ変換するには次の構文を使用します。

<変換前の型>_TO_<変換後の型>

例：DINT_TO_INT

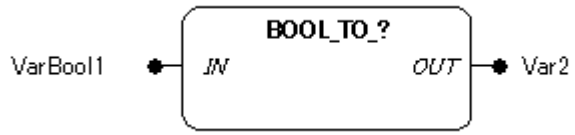
また、～TO_STRING変換の結果は、左詰めの文字列が生成されます。この際に変換後の文字列変数定義が変換前の変数定義よりも短い場合は結果文字列の右側(終端側)が切られます。

分類	命令	機能
BOOL型変換演算子	BOOL_TO_?	BOOL型の変換
	?_TO_BOOL	BOOL型への変換
数値型変換演算子	SINT_TO_? INT_TO_? DINT_TO_? LINT_TO_?	整数型の変換
	?_TO_SINT ?_TO_INT ?_TO_DINT ?_TO_LINT	整数型への変換
	BYTE_TO_? WORD_TO_? DWORD_TO_? LWORD_TO_?	ビット列型の変換
	?_TO_BYTE ?_TO_WORD ?_TO_DWORD ?_TO_LWORD	ビット列型への変換

分類	命令	機能
	USINT_TO_? UINT_TO_? UDINT_TO_? ULINT_TO_?	符号なし数値型の変換
	?_TO_USINT ?_TO_UINT ?_TO_UDINT ?_TO_ULINT	符号なし数値型への変換
実数型変換演算子	REAL_TO_?	実数の変換
	?_TO_REAL	実数型への変換
BCD変換関数	BCD_TO_BYTE BCD_TO_WORD BCD_TO_DWORD BCD_TO_INT	BCD値の変換
	BYTE_TO_BCD WORD_TO_BCD DWORD_TO_BCD INT_TO_BCD	BCD値への変換
時間型変換演算子	TIME_TO TIME_OF_DAY	時間型の変換
日付型変換演算子	DATE_TO DT_TO	日付型の変換
実数の整数変換演算子	TRUNC TRUNC_INT	小数点以下の値の切り捨て

BOOL_TO_?変換

(FBD の例)



機能

BOOL データ型の入力値を他のデータ型へ変換します。

構文は

BOOL_TO_<データ型>

数値型への変換は、入力パラメータが TRUE の時の結果は 1、FALSE の時の結果は 0 が返ります。
文字列型への変換は、入力パラメータが TRUE の時の結果は 'TRUE'、FALSE の時の結果は 'FALSE' が返ります。

パラメータで使用可能なデータ型

IN: BOOL
OUT: 数値型, STRING

パラメータ

入力変数	解説	備考
IN (VarBool1)	入力	

出力変数	解説	備考
OUT (Var2)	結果	指定したデータ型

解説

入力値が FALSE のとき、出力値は 0 に、入力値が TRUE のとき、出力値は 1 に変換されます。

(ILの例)

```
LD              TRUE
BOOL_TO_INT
ST              i
```
(*結果は 1 *)

```
LD              TRUE
BOOL_TO_STRI...
ST              str
```
(* 結果は 'TRUE' *)

```
LD              TRUE
BOOL_TO_TIME
ST              t
```
(* 結果は T#1ms *)

```
LD              TRUE
BOOL_TO_TOD
ST              tof
```
(* 結果は TOD#00:00:00.001 *)

```
LD              FALSE
BOOL_TO_DATE
ST              dandt
```
(* 結果は D#1970-01-01 *)

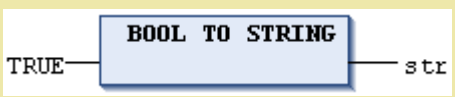
```
LD              TRUE
BOOL_TO_DT
ST              dandt
```
(* 結果は DT#1970-01-01-00:00:01 *)

(STの例)

```
i:=BOOL_TO_INT(TRUE);        (* 結果は 1 *)
str:=BOOL_TO_STRING(TRUE);   (* 結果は "TRUE" *)
t:=BOOL_TO_TIME(TRUE);       (* 結果は T#1ms *)
tof:=BOOL_TO_TOD(TRUE);      (* 結果は TOD#00:00:00.001 *)
dat:=BOOL_TO_DATE(FALSE);    (* 結果は D#1970 *)
dandt:=BOOL_TO_DT(TRUE);     (* 結果は DT#1970-01-01-00:00:01 *)
```

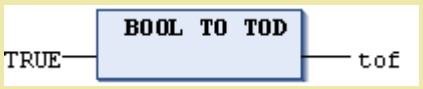
(FBDの例)

(* 結果は 1 *)

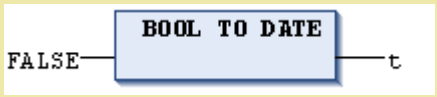


(* 結果は 'TRUE' *)

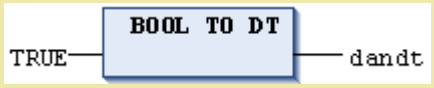
(* 結果は T#1ms *)



(* 結果は TOD#00:00:00.001 *)



(* 結果は D#1970-01-01 *)



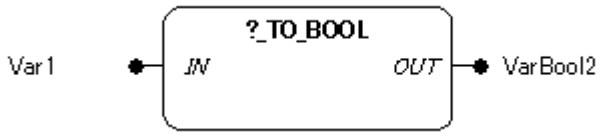
(* 結果は DT#1970-01-01-00:00:01 *)

補 足

STRING変換は文字列変換ファンクションの項を参照して下さい。

?_TO_BOOL 変換

(FBD の例)



機能

他のデータ型の入力値を BOOL データ型へ変換します。

構文は

<データ型>_TO_BOOL

パラメータで使用可能なデータ型

IN: 数値型, STRING

OUT: BOOL

パラメータ

入力変数	解説	備考
IN (Var1)	入力	指定したデータ型

出力変数	解説	備考
OUT (VarBool2)	結果	

解 説

入力が0以外の場合、変換結果はTRUE、入力が0と等しい場合はFALSEになります。

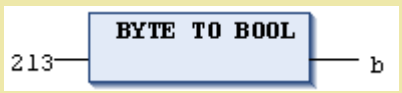
(ILの例)

```
LD          213
BYTE_TO_BOOL
ST          b          (* 結果はTRUE *)

LD          0
INT_TO_BOOL
ST          b          (*結果は FALSE *)

LD          T#5ms
TIME_TO_BOOL
ST          b          (*結果は TRUE *)

LD          'TRUE'
STRING_TO_BOOL
ST          b          (*結果は TRUE *)
```

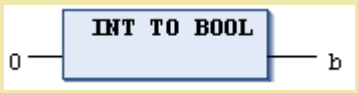
(STの例)

```
b := BYTE_TO_BOOL(2#11010101);  (*結果は TRUE *)
b := INT_TO_BOOL(0);            (*結果は FALSE *)
b := TIME_TO_BOOL(T#5ms);       (*結果は TRUE *)
b := STRING_TO_BOOL('TRUE');    (*結果は TRUE *)
```

(FBDの例)



(*結果は TRUE *)



(*結果は FALSE *)

(*結果は TRUE *)

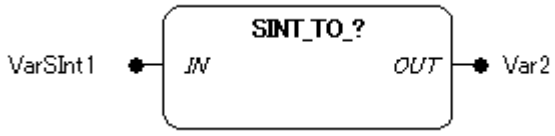
(*結果は TRUE *)

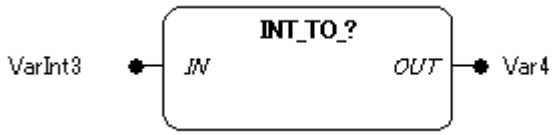
補 足

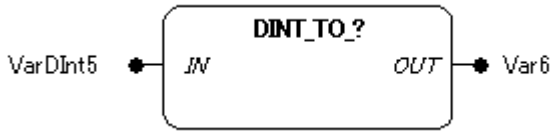
STRING変換は文字列変換ファンクションの項を参照して下さい。

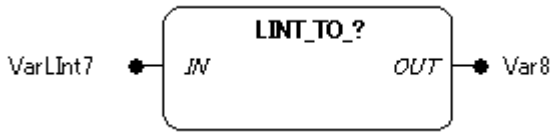
SINT_TO_? / INT_TO_? / DINT_TO_? / LINT_TO_? 変換

(FBD の例)









機能

整数型のデータ型入力値を他のデータ型へ変換します。

構文は

SINT_TO_<データ型>
INT_TO_<データ型>
DINT_TO_<データ型>
LINT_TO_<データ型>

パラメータで使用可能なデータ型

IN: SINT, INT, DINT, LINT
OUT: 数値型

パラメータ

入力変数	解説	備考
IN (VarSInt1, VarInt3, VarDInt5, VarLInt7)	入力	指定したデータ型

出力変数	解説	備考
OUT (Var2, Var4, Var6, Var8)	結果	

解 説

出力データ型が入力データ型より大きいときは、符号適合拡張が行われます。(例:SINT#-1 → DINT#-1)
小さいデータ型へ変換を行う場合、下位データビットに合わせるため、情報を失う恐れがあります。
符号無し型(USINT、UINT、UDINT)へ変換する場合、出力は常に正の値となります。

(STの例)

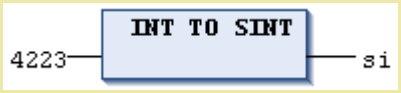
```
si := INT_TO_SINT(4223); (* 結果127 *)
```

ここで整数値 4223 (16進表記で16#107f) をSINT 変数に保存したならば127 (16進表記で16#7f)が現れます。

(ILの例)

```
LD          4223
INT_TO_SINT
ST          si
```

(FBDの例)

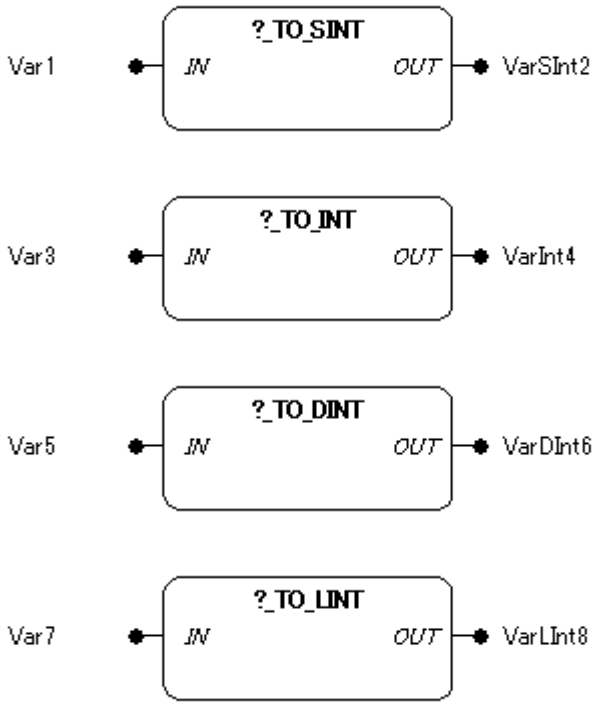


補 足

BOOL、REAL、BCD、STRING変換は各ファンクションの項を参照して下さい。

?_TO_SINT / ?_TO_INT / ?_TO_DINT / ?_TO_LINT 変換

(FBD の例)



機能

他のデータ型入力値を整数型のデータ型へ変換します。

構文は

<データ型>_TO_SINT

<データ型>_TO_INT

<データ型>_TO_DINT

<データ型>_TO_LINT

パラメータで使用可能なデータ型

IN: 数値型, STRING

OUT: SINT, INT, DINT, LINT

パラメータ

入力変数	解説	備考
IN (Var1, Var3, Var5, Var7)	入力	指定したデータ型

出力変数	解説	備考
OUT (VarSInt2, VarInt4, VarDInt6, VarLInt8)	結果	

解 説

出力データ型が入力データ型より大きいときは、符号適合拡張が行われます。(例 : 16#FF → -1)

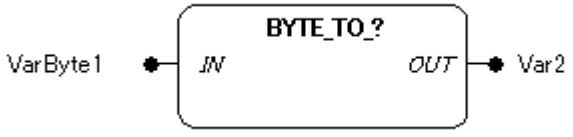
小さいデータ型へ変換を行う場合、下位データビットに合わせるため、情報を失う恐れがあります。

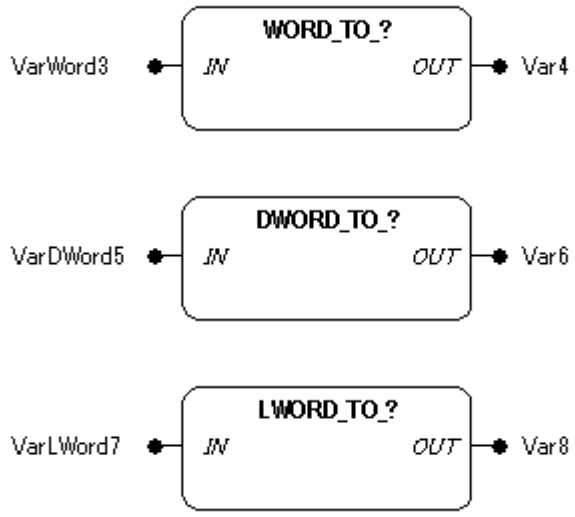
補 足

BOOL、REAL、BCD、STRING変換は各ファンクションの項を参照して下さい。

BYTE_TO_? / WORD_TO_? / DWORD_TO_? / LWORD_TO_? 変換

(FBD の例)





機能

ビットストリーム型のデータ型入力値を他のデータ型へ変換します。

構文は

BYTE_TO_<データ型>
WORD_TO_<データ型>
DWORD_TO_<データ型>
LWORD_TO_<データ型>

パラメータで使用可能なデータ型

IN: BYTE, WORD, DWORD, LWORD
OUT: 数値型, STRING

パラメータ

入力変数	解説	備考
IN (VarByte1, VarWord3, VarDWord5, VarLWord7)	入力	

出力変数	解説	備考
OUT (Var2, Var4, Var6, Var8)	結果	指定したデータ型

解 説

出力データ型が入力データ型より大きいときは、符号適合拡張が行われます。(例 : 16#FF → -1)
小さいデータ型へ変換を行う場合、下位データビットに合わせるため、情報を失う恐れがあります。
出力データ型のMSB (最上位ビット) がセットされている場合は、負の出力値へ変換されます。

(IL の例)

```
LD          16#80   (* 16#80 をアキュムレータにロードします *)
BYTE_TO_INT         (* BYTE を INT に変換します *)
ST          Var2    (* -128 を Var2 に格納します *)


LD           16#7FFF   (* 16#7FFF をアキュムレータにロードします *)
WORD_TO_UINT           (* WORD を UINT に変換します *)
ST           Var4      (* 32767 を Var4 に格納します *)


LD            16#FFFFFF7F   (* 16#FFFFFF7F をアキュムレータにロードします *)
DWORD_TO_BYTE               (* DWORD を BYTE に変換します *)
ST            Var6          (* 16#7F を Var6 に格納します *)
```

(STの例)

```
iVar := BYTE_TO_INT(16#80);  (* 結果 -128 *)
```

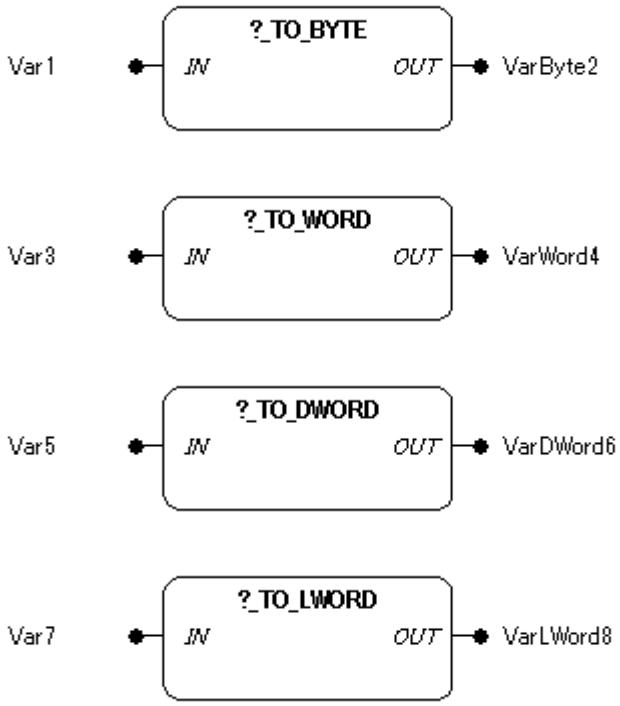
ここで整数値 128 (16進表記で16#80) をINT 変数に保存したならば-128 (16進表記で16#ff80)が現れます。

補 足

BOOL、REAL、BCD、STRING変換は各ファンクションの項を参照して下さい。

?_TO_BYTE / ?_TO_WORD / ?_TO_DWORD / ?_TO_LWORD 変換

(FBD の例)



機能

他のデータ型入力値をビットストリーム型のデータ型へ変換します。

構文は

<データ型>_TO_BYTE

<データ型>_TO_WORD

<データ型>_TO_DWORD

<データ型>_TO_LWORD

パラメータで使用可能なデータ型

IN: 数値型, STRING

OUT: BYTE, WORD, DWORD, LWORD

パラメータ

入力変数	解説	備考
IN (Var1, Var3, Var5, Var7)	入力	指定したデータ型

出力変数	解説	備考
OUT (VarByte2, VarWord4, VarDWord6, VarLWord8)	結果	

解 説

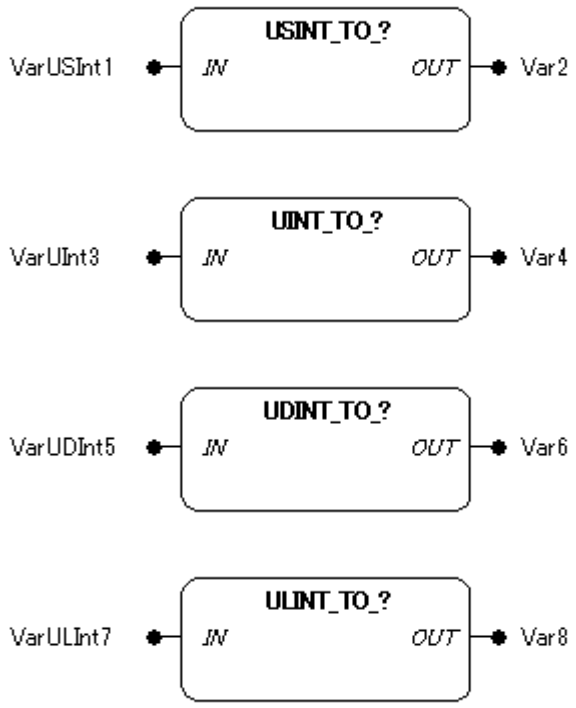
小さいデータ型へ変換を行う場合、下位データビットに合わせるため、情報を失う恐れがあります。

補 足

BOOL、REAL、BCD、STRING変換は各ファンクションの項を参照して下さい。

USINT_TO_? / UINT_TO_? / UDINT_TO_? / ULINT_TO_? 変換

(FBD の例)



機 能

符号無し整数型のデータ型入力値を他のデータ型へ変換します。

構文は

USINT_TO_<データ型>

UINT_TO_<データ型>

UDINT_TO_<データ型>

ULINT_TO_<データ型>

パラメータで使用可能なデータ型

IN: USINT, UINT, UDINT, ULINT

OUT: 数値型, STRING

パラメータ

入力変数	解説	備考
IN (VarUSInt1, VarUInt3, VarUDInt5, VarULInt7)	入力	

出力変数	解説	備考
OUT (Var2, Var4, Var6, Var8)	結果	指定したデータ型

解 説

小さいデータ型へ変換を行う場合、下位データビットに合わせるため、情報を失う恐れがあります。
出力データ型のMSB (最上位ビット) がセットされている場合は、負の出力値へ変換されます。

(STの例)

```
siVar := USINT_TO_SINT(254);  (* 結果 -2 *)
```

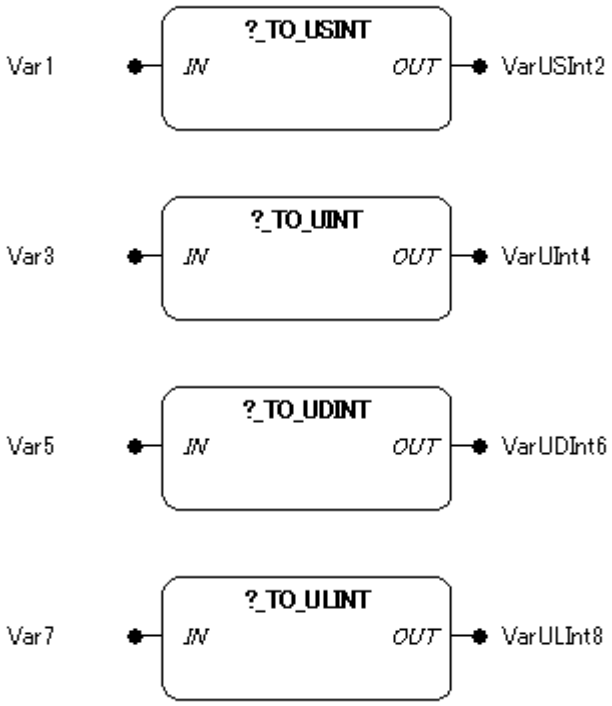
ここで整数値 254 (16進表記で16#fe) をSINT 変数に保存したならば-2 (16進表記で16#fe)が現れます。

補 足

BOOL、REAL、BCD、STRING変換は各ファンクションの項を参照して下さい。

?_TO_USINT / ?_TO_UINT / ?_TO_UDINT / ?_TO_ULINT 変換

(FBD の例)



機能

他のデータ型入力値を符号無し整数型のデータ型へ変換します。

構文は

<データ型>_TO_USINT

<データ型>_TO_UINT

<データ型>_TO_UDINT

<データ型>_TO_ULINT

パラメータで使用可能なデータ型

IN: 数値型, STRING

OUT: USINT, UINT, UDINT, ULINT

パラメータ

入力変数	解説	備考
IN (Var1, Var3, Var5, Var7)	入力	指定したデータ型

出力変数	解説	備考
OUT (VarUSInt2, VarUInt4, VarUDInt6, VarULInt8)	結果	

解 説

出力は常に正の値となります。

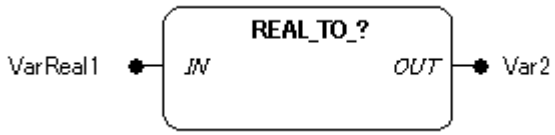
小さいデータ型へ変換を行う場合、下位データビットに合わせるため、情報を失う恐れがあります。

補 足

BOOL、REAL、BCD、STRING変換は各ファンクションの項を参照して下さい。

REAL_TO_? 変換

(FBD の例)



機能

REAL データ型の入力値を他のデータ型の出力値に変換します。

構文は

REAL_TO_<データ型>

パラメータで使用可能なデータ型

IN: REAL

OUT: 数値型, STRING

パラメータ

入力変数	解説	備考
IN (VarReal1)	入力	

出力変数	解説	備考
OUT (Var2)	結果	指定したデータ型

解 説

REAL 入力値は小数点第1位で四捨五入された後、他のデータ型に変換されます。

小さいデータ型へ変換を行う場合、下位データビットに合わせるため、情報を失う恐れがあります。

(STの例)

```
i := REAL_TO_INT(1.5);   (* 結果は 2 *)
j := REAL_TO_INT(1.4);   (* 結果は 1 *)
i := REAL_TO_INT(-1.5);  (* 結果は -2 *)
j := REAL_TO_INT(-1.4);  (* 結果は -1 *)
```

(ILの例)

```
LD          2.7
REAL_TO_INT
ST          i
```

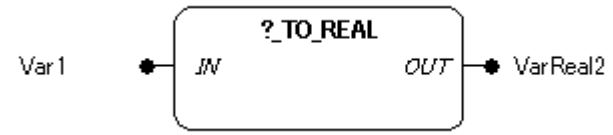
(FBDの例)

補 足

BOOL、REAL、BCD、STRING変換は各ファンクションの項を参照して下さい。

?_TO_REAL 変換

(FBD の例)



機能

他のデータ型の入力値を REAL データ型の出力値に変換します。

構文は

<データ型>_TO_REAL

パラメータ

入力変数	解説	備考
IN (Var1)	入力	指定したデータ型

出力変数	解説	備考
OUT (VarReal2)	結果	

(STの例)

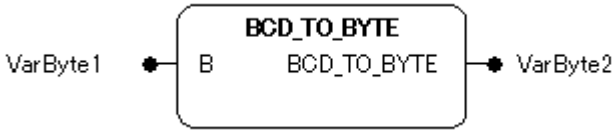
```
VarReal2 := INT_TO_REAL(-32768);  (* 結果は -3.2768000E+04 *)
```

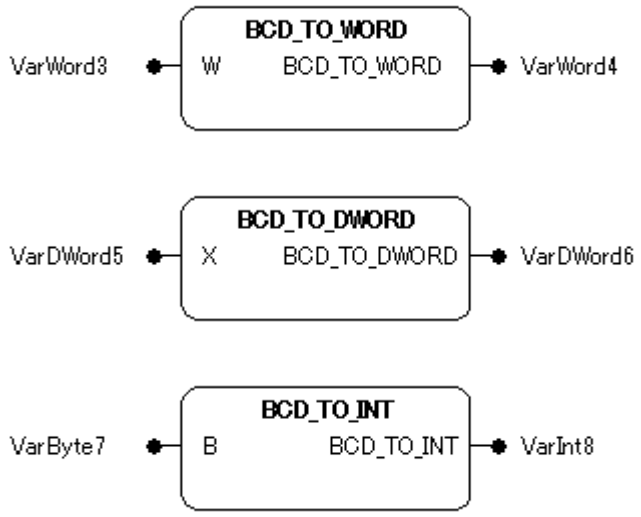
補足

BOOL、REAL、BCD、STRING変換は各ファンクションの項を参照して下さい。

BCD_TO_BYTE / BCD_TO_WORD / BCD_TO_DWORD / BCD_TO_INT 変換【FUN】

(FBD の例)





機能

BCD データ型の入力値 (バイナリコードの 10 進数) を他のデータ型の戻り値に変換します。

構文は

BCD_TO_BYTE
BCD_TO_WORD
BCD_TO_DWORD
BCD_TO_INT

パラメータで使用可能なデータ型

B: BYTE
W: WORD
X: DWORD

パラメータ

入力変数	解説	備考
B (VarByte1, VarByte7)	入力	16#00～16#99
W (VarWord3)	入力	16#0000～16#9999
X (VarDWord5)	入力	16#00000000～16#99999999

戻り値	解説	備考
BCD_TO_BYTE	結果	BYTEデータ型

戻り値	解説	備考
BCD_TO_WORD	結果	WORDデータ型
BCD_TO_DWORD	結果	DWORDデータ型
BCD_TO_INT	結果	BCD書式でなければ -1 を返します

(ST の例)

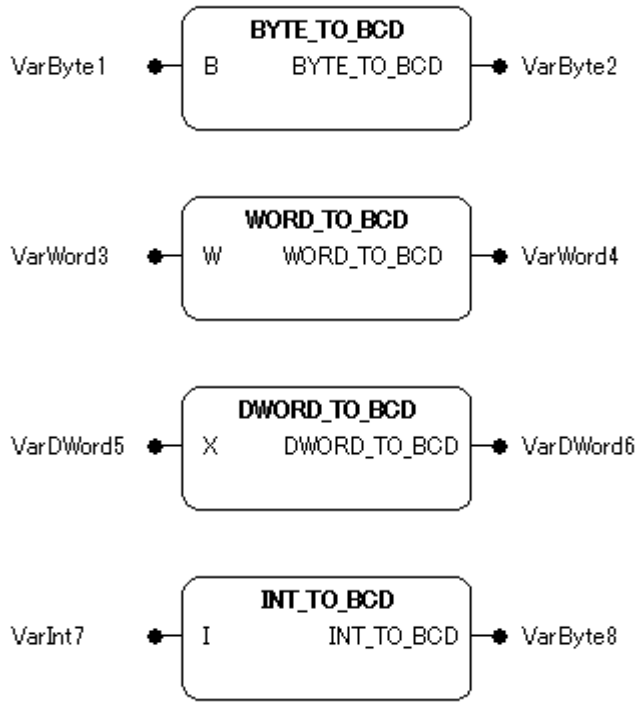
```
i:=BCD_TO_INT(73);   (* 結果は 49 *)
k:=BCD_TO_INT(151);  (* 結果は 97 *)
l:=BCD_TO_INT(15);   (* 値がBCDではないので、結果は -1 *)
```

補 足

- 不正な入力値 (使用可能な値は0から9まで) のときは、出力値は-1に変換されます。たとえば、入力値が16#0A0Bのとき、出力値は-1になります。
- 出力データ型の有効範囲を超えている値の場合は、オーバーフローになることを考慮する必要があります。

BYTE_TO_BCD / WORD_TO_BCD / DWORD_TO_BCD / INT_TO_BCD 変換【FUN】

(FBD の例)



機能

他のデータ型の入力値を BCD データ型 (バイナリコードの 10 進数) の出力値に変換します。

構文は

BYTE_TO_BCD
WORD_TO_BCD
DWORD_TO_BCD
INT_TO_BCD

パラメータで使用可能なデータ型

B: BYTE
W: WORD
X: DWORD
I: INT

パラメータ

入力変数	解説	備考
B (VarByte1)	入力	0～99
W (VarWord3)	入力	0～9999
X (VarDWord5)	入力	0～99999999
I (VarInt7)	入力	0～99

戻り値	解説	備考
BYTE_TOBCD	結果	
WORD_TOBCD	結果	
DWORD_TOBCD	結果	
INT_TO_BCD	結果	0～99以外は 255 を返します

(STの例)

```
i:=INT_TO_BCD(49);   (* 結果は 73 *)
k:=INT_TO_BCD(97);   (* 結果は 151 *)
```

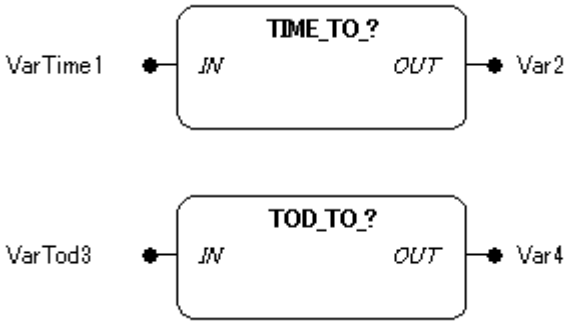
```
l:=INT_TO_BCD(100);  (* エラーなので 255 *)
```

補 足

- 負の入力値は常に16#FFFFFFFF へ変換されます。たとえば、入力値が-128 のとき出力値は16#FFFFFFFF になります。
- 出力データ型 (BCD) の有効範囲を超えた入力値の場合は、出力値は常に16#FFFFFFFF へ変換されます。BCD 値の最大有効範囲は99999999 です。

TIME_TO_? / TIME_OF_DAY_? 変換

(FBD の例)



機能

TIME あるいはTIME_OF_DAY データ型入力値 IN を他のデータ型の出力値に変換します。

構文は

TIME_TO_<データ型>

TOD_TO_<データ型>

パラメータで使用可能なデータ型

IN: TIME, TOD

OUT: 数値型, STRING

パラメータ

入力変数	解説	備考
IN (VarTime1)	入力	TIME データ型
IN (VarTod3)	入力	TIME_OF_DAY

出力変数	解説	備考
OUT (Var2, Var4)	結果	指定したデータ型

解 説

時間は内部でDWORD型にミリ秒で格納され、TIME_OF_DAY 変数では12:00A.M.を開始とする値が格納されます。STRING データ型への変換は時間コンスタント値が返されます。大きな値をもつ入力変数から小さなデータ型へ変換を行うと情報を失う恐れがあります。

(ILの例)

```
LD          T#12ms
TIME_TO_STTI...
ST          str
```
(* 結果は 'T#12ms' *)

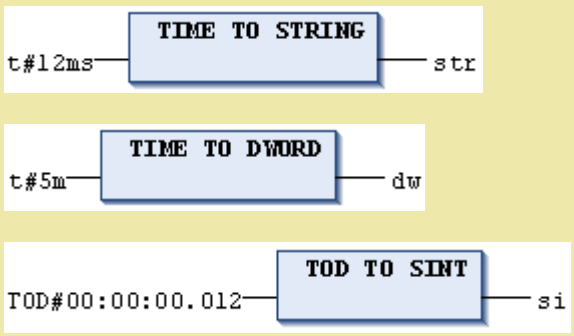
```
LD          T#300000ms
TIME_TO_DWORD
ST          dw
```
(*結果は 300000 *)

```
LD          TOD#00:00:00.012
TIME_TO_SINT
ST          si
```
(*結果は 12 *)

(STの例)

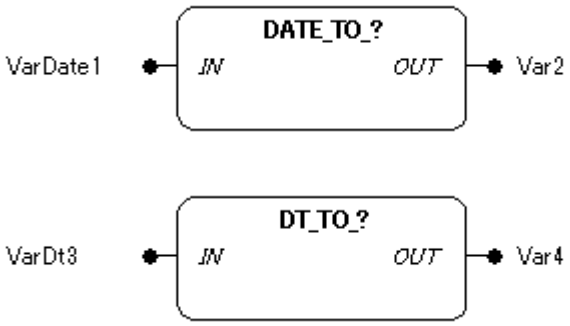
```
str :=TIME_TO_STRING(T#12ms);          (*結果は T#12ms *)
dw:=TIME_TO_DWORD(T#5m);               (*結果は 300000 *)
si:=TOD_TO_SINT(TOD#00:00:00.012);     (*結果は 12 *)
```

(FBDの例)



DATE_TO_? / DATE_AND_TIME_TO_? 変換

(FBD の例)



機能

DATE あるいは DATE_AND_TIME データ型入力値を他のデータ型の出力値に変換します。

構文は

DATE_TO_<データ型>

DT_TO_<データ型>

パラメータで使用可能なデータ型

IN: DATE, DATE_AND_TIME

OUT: 数値型, STRING

パラメータ

入力変数	解説	備考
IN (VarDate1)	入力	DATE データ型
IN (VarDt3)	入力	DATE_AND_TIME

出力変数	解説	備考
OUT (Var2, Var4)	結果	指定したデータ型

解 説

時刻は内部でDWORD型に1970/1/1からの通算秒とする値が格納されます。STRING データ型への変換は時刻コンスタント値が返されます。大きな値をもつ入力変数から小さなデータ型へ変換を行うと情報を失う恐れがあります。

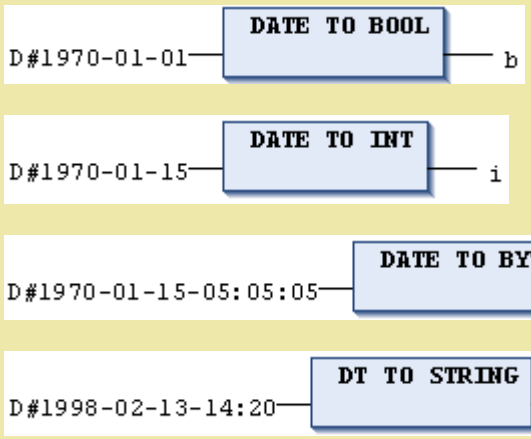
(ILの例)

```
LD          D#1970-01-01
DATE_TO_BOOL
ST          b                    (* 結果は FALSE *)

LD          D#1970-01-01
DATE_TO_INT
ST          i                    (*結果は 29952 *)

LD          D#1970-01-15-05:05:.
DATE_TO_BYTE
ST          byt                  (*結果は 129 *)

LD          D#1998-02-13-14:20
DATE_TO_STRI...
ST          str                  ( *結果は 'DT#1998-02-13-14:20' *)
```

(STの例)

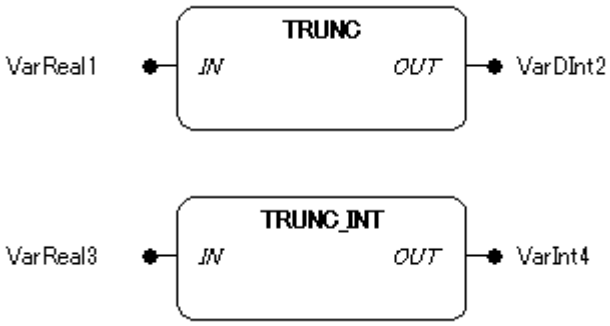
```
b :=DATE_TO_BOOL(D#1970-01-01);          (*結果は FALSE *)
i :=DATE_TO_INT(D#1970-01-15);           (*結果は 29952 *)
byt :=DT_TO_BYTE(DT#1970-01-15-
05:05:05);                               (*結果は 129 *)
str:=DT_TO_STRING(DT#1998-02-13-         (*結果は 'DT#1998-02-13-14:20'
14:20);                                  *)
```

(FBDの例)



TRUNC / TRUNC_INT 変換

(FBD の例)



機能

TRUNCはREAL 値から小数点以下の値は切り捨ててDINT 値へ変換します。

TRUNC_INTはREAL 値から小数点以下の値は切り捨ててINT 値へ変換します。

パラメータ

入力変数	解説	備考
IN (VarReal, VarReal3)	入力	REAL データ型

出力変数	解説	備考
OUT (VarDInt2)	結果	DINT データ型
OUT (VarInt4)	結果	INT データ型

(ILの例)

```
LD      1.9
TRUNC
ST      diVar
```

```
LD      1.9
TRUNC_INT
ST      iVar
```

(STの例)

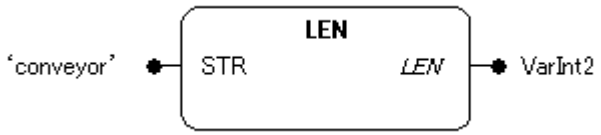
```
diVar:=TRUNC(1.9);   (* 結果は 1 *)
diVar:=TRUNC(-1.4);  (* 結果は -1 *)


iVar:=TRUNC_INT(1.9);   (* 結果は 1 *)
iVar:=TRUNC_INT(-1.4);  (* 結果は -1 *)
```

8.5.文字列操作ファンクション

分類	命令	機能
文字列操作	LEN	文字列の長さ
	LEFT	左端からの文字列抽出
	RIGHT	右端からの文字列抽出
	MID	中間からの文字列抽出
	CONCAT	文字列の連結
	INSERT	文字の挿入
	DELETE	文字列の削除
	REPLACE	文字の置換
	FIND	文字の検索

LEN:文字列長さ【FUN】

(FBD の例)



機能

文字列の長さを返します。

パラメータで使用可能なデータ型

STR: STRING

LEN: INT

パラメータ

入力変数	解説	備考
STR ('conveyor')	入力	

出力変数	解説	備考
LEN (VarInt2)	結果	

解 説

入力パラメータ STR に接続された文字列の長さを返します。

(ILの例) (結果は "4")

```
LD      'SUSI'
LEN
ST      VarINT1
```

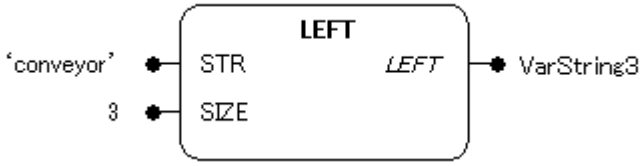
(STの例)

```
VarINT1 := LEN ('SUSI');
```

(FBDの例)

LEFT:左文字列抽出 [FUN]

(FBD の例)



機 能

文字列入力の左から指定長さの文字列を抽出します。

パラメータで使用可能なデータ型

STR: STRING

SIZE: INT

LEFT: STRING

パラメータ

入力パラメータ	説明	備考
STR ('conveyor')	文字列入力	
SIZE (3)	左から抽出する長さ(文字数)	0 以上

出力パラメータ	説明	備考
LEFT (VarString3)	結果 ('con')	

解 説

入力 STR に接続された文字列の左端から入力 SIZE に接続された長さの文字列を抽出し返します。

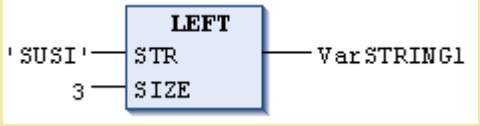
(ILの例) (結果は 'SUS')

```
LD      'SUSI'
LEFT    3
ST      VarSTRING1
```

(STの例)

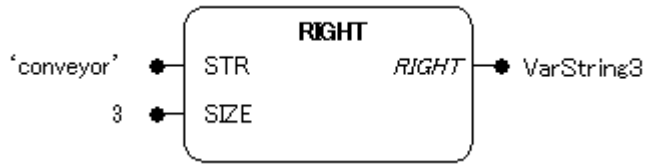
```
VarSTRING1 := LEFT ('SUSI',3);
```

(FBDの例)



RIGHT:右文字列抽出 [FUN]

(FBD の例)



機能

文字列入力の右から指定長さの文字列を抽出します。

パラメータで使用可能なデータ型

STR: STRING

SIZE: INT

RIGHT: STRING

パラメータ

入力パラメータ	説明	備考
STR ('conveyor')	文字列入力	
SIZE (3)	右から抽出する長さ(文字数)	1 以上

出力パラメータ	説明	備考
RIGHT (VarString3)	結果 ('yor')	

解説

入力 STR に接続された文字列の右端から入力 SIZE に接続された長さの文字列を抽出し返します。

(ILの例) (結果は 'USI')

```
LD      'SUSI'
RIGHT   3
ST      VarSTRING1
```

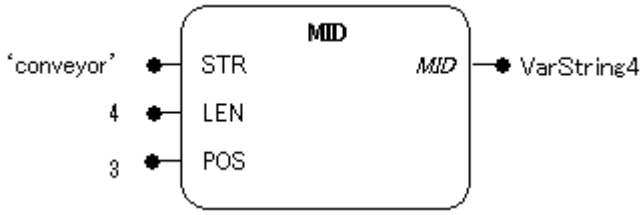
(STの例)

```
VarSTRING1 := RIGHT ('SUSI',3);
```

(FBDの例)

MID:中間文字列抽出 [FUN]

(FBD の例)



機能

文字列入力の中間位置から、指定長さの文字列を抽出します。

パラメータで使用可能なデータ型

STR: STRING

LEN, POS: INT

MID: STRING

パラメータ

入力パラメータ	説明	備考
STR ('conveyor')	文字列入力	
LEN (4)	抽出する長さ(文字数)	0 以上

入力パラメータ	説明	備考
POS (3)	STRの先頭を1とした抽出開始位置	

出力パラメータ	説明	備考
MID (VarString4)	結果 ('nvey')	

解 説

入力 STR に接続された文字列の入力 POS で指定される中間位置から、入力 LEN で指定される長さの文字列を抽出し結果として返します。

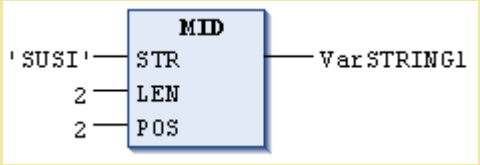
(ILの例) (結果は 'US')

```
LD      'SUSI'
MID     2,
        2
ST      VarSTRING1
```

(STの例)

```
VarSTRING1 := MID ('SUSI',2,2);
```

(FBDの例)

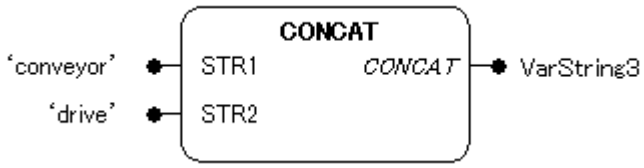


補 足

- POS は 0 とすることはできません。文字列の最初の位置は 1 です。

CONCAT:文字列連結 [FUN]

(FBD の例)



機能

文字列の連結をします。

パラメータで使用可能なデータ型

STR1, STR2: STRING

CONCAT: STRING

パラメータ

入力パラメータ	説明	備考
STR1 ('conveyor')	文字列1入力	
STR2 ('drive')	文字列2入力	

出力パラメータ	説明	備考
CONCAT (VarString3)	結果 ('conveyordrive')	

解説

STR1入力文字列の後尾にSTR2入力文字列を付加し2つの文字列を結合した文字列を返します。

入力文字列と結果として返される文字列の最大は255文字となります。

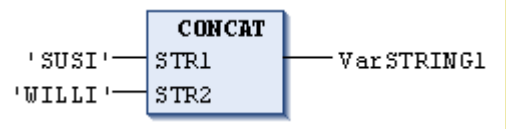
(ILの例) (結果は 'SUSIWILLI')

```
LD      'SUSI'
CONCAT  'WILLI'
ST      VarSTRING1
```

(STの例)

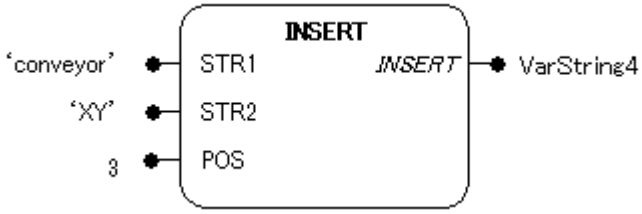
```
VarSTRING1 := CONCAT ('SUSI','WILLI');
```

(FBDの例)



INSERT:文字列挿入 [FUN]

(FBD の例)



機能

文字列の挿入をします。

パラメータで使用可能なデータ型

STR1, STR2: STRING

POS: INT

INSERT: STRING

パラメータ

入力パラメータ	説明	備考
STR1 ('conveyor')	挿入される文字列	
STR2 ('XY')	挿入する文字列	
POS (3)	STRの先頭を1とした位置	この位置の後に挿入

出力パラメータ	説明	備考
INSERT (VarString4)	結果 ('conXYveyor')	

解 説

与えられた文字列 STR1 の指定位置へ文字列 STR2 を挿入します。
STR1 の文字位置 POS の後ろにSTR2 が挿入されます。

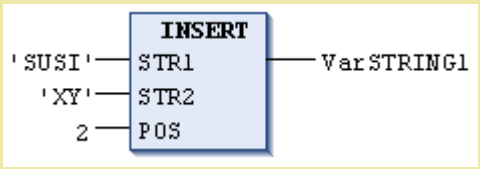
(ILの例) (結果は 'SUXYSI')

```
LD      'SUSI'
INSERT  'XY'    ,
        2
ST      VarSTRING1
```

(STの例)

```
VarSTRING1 := INSERT ('SUSI','XY',2);
```

(FBDの例)

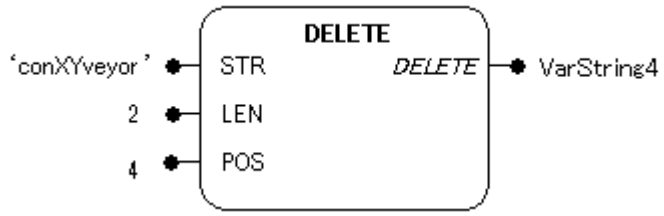


補 足

- POS は 0 とすることはできません。文字列の最初の位置は 1 です。
- 文字列を他の文字列の前に挿入したいときは、ファンクション CONCAT を使用してください。

DELETE:文字列削除 [FUN]

(FBD の例)



機能

文字列の削除をします。

パラメータで使用可能なデータ型

STR: STRING

LEN, POS: INT

DELETE: STRING

パラメータ

入力パラメータ	説明	備考
STR ('conXYveyor')	削除箇所を含む文字列	
LEN (2)	削除する文字数	0 以上
POS (4)	STRの先頭を1とした削除開始位置	この位置から削除

出力パラメータ	説明	備考
DELETE (VarString4)	結果 ('conveyor')	

解説

与えられた文字列 STR の文字位置 POS で始まる文字数 LEN 部分が削除されます。

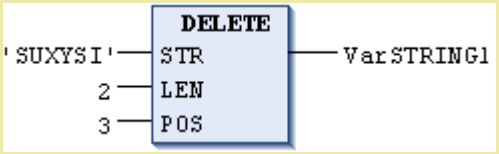
(ILの例) (結果は 'SUSI')

```
LD      'SUXYSI'
DELETE  2       ,
        3
ST      VarSTRING1
```

(STの例)

```
Var1 := DELETE ('SUXYSI',2,3);
```

(FBDの例)

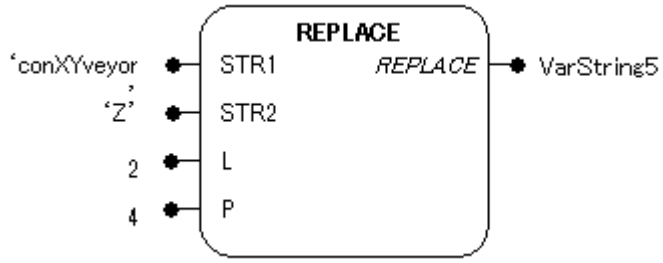


補 足

- POS は 0 とすることはできません。文字列の最初の位置は 1 です。

REPLACE:文字列置換 [FUN]

(FBD の例)



機能

文字列の置換を行います。

パラメータで使用可能なデータ型

STR1, STR2: STRING

L, P: INT

REPLACE: STRING

パラメータ

入力パラメータ	説明	備考
STR1 ('conXYveyor')	置換箇所を含む文字列	
STR2 ('Z')	置換後の文字列	
L (2)	置換する文字数	0 以上
P (4)	STR1の先頭を1とした置換位置	この位置から L 文字置換

出力パラメータ	説明	備考
REPLACE (VarString5)	結果 ('conZveyor')	

解 説

文字列中のある文字列を他のものと置き換えます。

文字列 STR1 の文字位置 P から始まる文字数 L 分を STR2 に置き換えます。

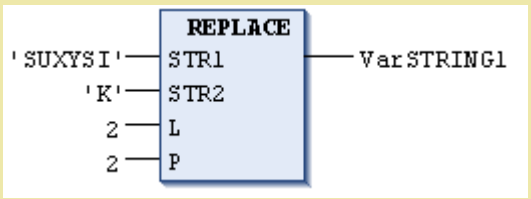
(ILの例) (結果は 'SKYSI')

```
LD      'SUXYSI'
REPLACE 'K'     ,
        2       ,
        2
ST      VarSTRING1
```

(STの例)

```
VarSTRING1 := REPLACE ('SUXYSI','K',2,2);
```

(FBDの例)

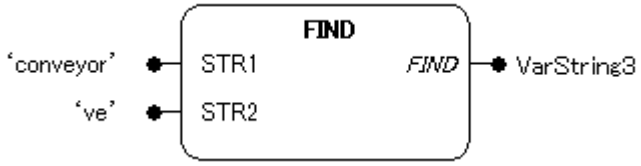


補 足

- P は 0 とすることはできません。文字列の最初の位置は 1 です。

FIND:文字列検索 [FUN]

(FBD の例)



機能

文字列の検索をします。

パラメータで使用可能なデータ型

STR1, STR2: STRING

FIND: INT

パラメータ

入力パラメータ	説明	備考
STR1 ('conveyor')	文字列1	
STR2 ('ve')	STR1内で検索する文字列2	

出力パラメータ	説明	備考
FIND (VarString3)	結果 (4)	STR1の先頭は1

解 説

与えられた文字列 STR1 内で文字列 STR2 の位置を検出します。

STR1 の中で STR2 が最初に現れる位置を結果として返します。

STR1 内にSTR2 が無ければ結果に0を返します。

(ILの例) (結果は '4')

```
LD      'abcdef'
FIND    'de'
ST      VarSTRING1
```

(STの例)

```
arINT1 := FIND ('abcdef','de');
```

(FBDの例)

補 足

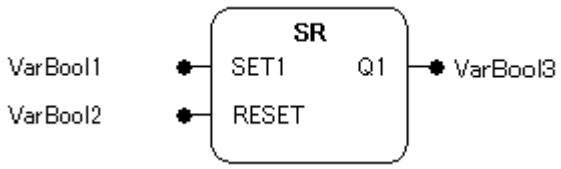
- 文字列の最初の位置は1です。

8.6.標準ファンクションブロック

分類	命令	機能
バイステーブル	SR	セット優先ラッチ
	RS	リセット優先ラッチ
カウンタ	CTU	アップカウンタ
	CTD	ダウンカウンタ
	CTUD	アップダウンカウンタ
タイマ	TON	オンディレイタイマ
	TOF	オフディレイタイマ
	TP	パルス幅出力
エッジ検出型	R_TRIG	立ち上がり検出
	F_TRIG	立ち下がり検出

SR：セット優先ラッチ [FB]

(FBD の例)



機能

SET1が優先するラッチです。SET1とRESETの両方の信号がTRUEならば出力Q1はTRUEになります。

パラメータで使用可能なデータ型

SET1, RESET：BOOL

Q1：BOOL

パラメータ

入力パラメータ	説明	備考
SET1 (VarBool1)	入力	
RESET (VarBool2)	リセット入力	

出力パラメータ	説明	備考
Q1 (VarBool3)	結果	

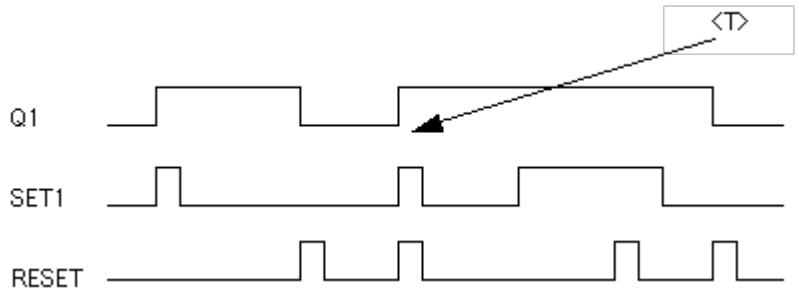
解 説

出力Q1のラッチを行います。

入力SET1 = TRUEの場合に出力Q1はセットされTRUEとなり、その後SET1がFALSEになってもQ1はTRUE状態が残ります。入力RESET = TRUEの場合にQ1がリセットされFALSEとなります。

SET1とRESETの両方の入力がTRUEの場合はSET1が優先されて出力Q1はセットされます。

初めてこのファンクションブロックが呼び出される際のQ1はFALSEです。



<T>: 同時の場合はセットが優先

宣言の例:

SRInst : SR ;

(ILの例)

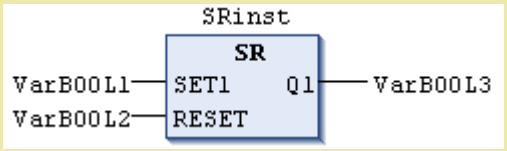
```
CAL          SRinst      (
        SET1:= VarBool1  ,
      RESET:= VarBool2   )
LD           SRinst.Q1
ST           VarBool3
```

(STの例)

```
SRInst(SET1:= VarBOOL1 , RESET:=VarBOOL2 );
```

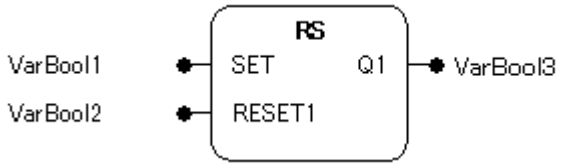
```
VarBOOL3 := SRInst.Q1 ;
```

(FBDの例)



RS:リセット優先ラッチ [FB]

(FBD の例)



機能

RESET1が優先するラッチです。SETとRESET1の両方の信号がTRUEならば出力Q1はFALSEになります。

パラメータで使用可能なデータ型

SET, RESET1: BOOL

Q1: BOOL

パラメータ

入力パラメータ	説明	備考
SET (VarBool1)	入力	
RESET1 (VarBool2)	リセット入力	

出力パラメータ	説明	備考
Q1 (VarBool3)	結果	

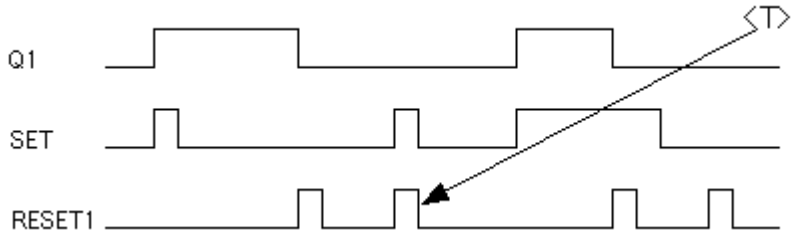
解 説

出力Q1のラッチを行います。

入力SET = TRUEの場合に出力Q1はセットされTRUEとなり、その後SETがFALSEになっても、Q1はTRUE状態が残ります。入力RESET1 = TRUEの場合にQ1はリセットされFALSEとなります。

SETとRESET1の両方の入力がTRUEの場合はRESET1が優先されて出力Q1はリセットされます。

初めてこのファンクションブロックが呼び出される際のQ1はFALSEです。



<T>: 同時の場合はリセットが優先

宣言の例:

RSInst : RS ;

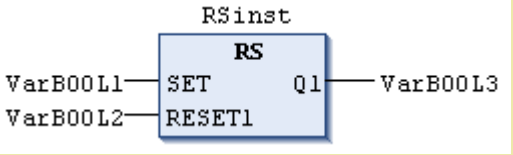
(ILの例)

```
CAL     RSinst  (
    SET:= VarBool1  ,
  RESET1:= VarBool2  )
LD      RSinst.Q1
ST      VarBool3
```

(STの例)

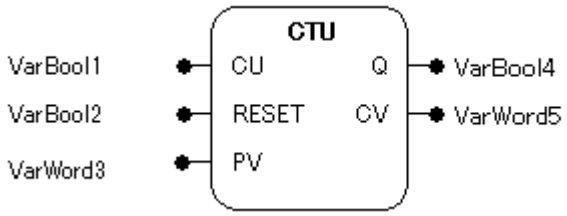
```
RSInst(SET:= VarBOOL1 , RESET1:=VarBOOL2 );
VarBOOL3 := RSInst.Q1 ;
```

(FBDの例)



CTU : アップカウンタ [FB]

(FBD の例)



機能

カウント値をカウントアップしプリセット値(最大値)に達したことを知らせるアップカウンタです。

パラメータで使用可能なデータ型

CU, RESET : BOOL

PV : WORD

Q : BOOL

CV : WORD

パラメータ

入力パラメータ	説明	備考
CU (VarBool1)	カウントトリガ入力	立ち上がりで1加算
RESET (VarBool2)	リセット入力	
PV (VarWord3)	目標値	

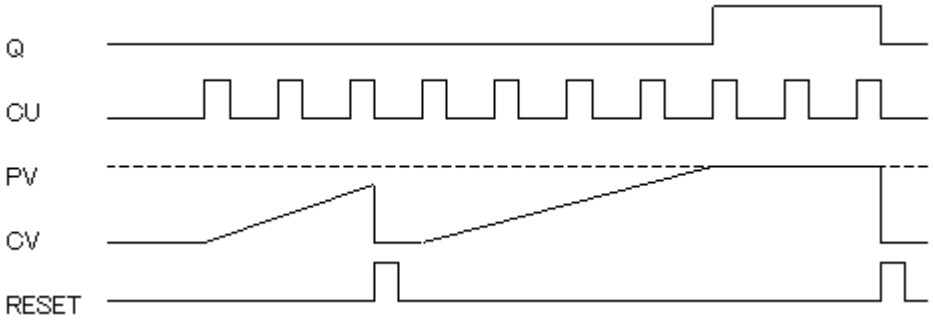
出力パラメータ	説明	備考
Q (VarBool4)	結果 (CV >= PV の場合 TRUE)	
CV (VarWord5)	カウント値	

解説

カウント値CVのカウントアップを行います。

RESET = FALSEの場合は入力CUの立ち上がりエッジでCVを1増加します。CVがプリセット値PVに到達すると出力Q = TRUEが出力され、このファンクションブロックはカウントを停止します。

RESET = TRUEの場合は出力Q = FALSE、カウンタCV = 0で初期化されます。



宣言の例:

CTUInst : CTU ;

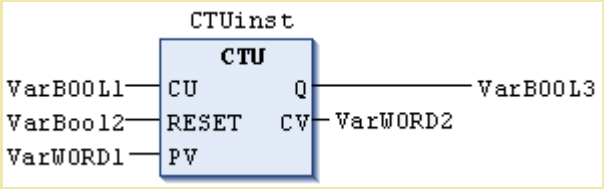
(ILの例)

```
CAL     CTUinst(
        CU:= VarBOOL1,
        Reset:= VarBOOL2,
        PV:= VarWORD1,
        CV=> VarWORD2)
LD      CTUinst.Q
ST      VarBOOL3
```

(STの例)

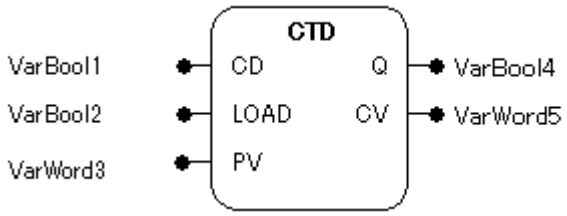
```
CTUInst(CU:= VarBOOL1, RESET:=VarBOOL2 , PV:= VarWORD1);
VarBOOL3 := CTUInst.Q ;
VarWORD2 := CTUInst.CV;
```

(FBDの例)



CTD：ダウンカウンタ [FB]

(FBD の例)



機能

プリセット値からカウントダウンしてカウント値が0に達したことを知らせるダウンカウンタです

パラメータで使用可能なデータ型

CD, LOAD：BOOL

PV：WORD

Q：BOOL

CV：WORD

パラメータ

入力パラメータ	説明	備考
CD (VarBool1)	カウントトリガ入力	立ち上がりで1減算
LOAD (VarBool2)	ロード入力	
PV (VarWord3)	開始値	

出力パラメータ	説明	備考
Q (VarBool4)	結果 (CV = 0 の場合 TRUE)	
CV (VarWord5)	カウント値	

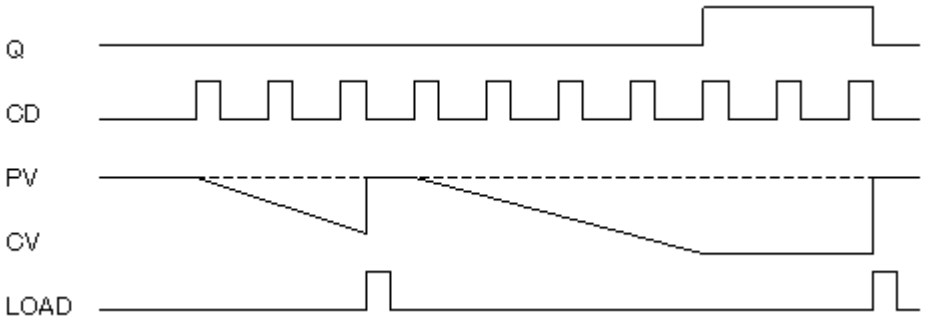
解説

カウント値CVのカウントダウンを行います。

LOAD ＝ FALSE の場合は入力CDの立ち上がりエッジでCV値を1減少させます。CV ＝ 0に到達すると出力 Q ＝ TRUEを出力し、このファンクションブロックはカウントを停止します。

LOAD ＝ TRUEの場合は出力Q ＝ FALSE、カウンタCV ＝ PV で初期化されます。

初期状態でCV ＝ 0の時もQ ＝ TRUEとなります。



宣言の例:

CTDInst : CTD ;

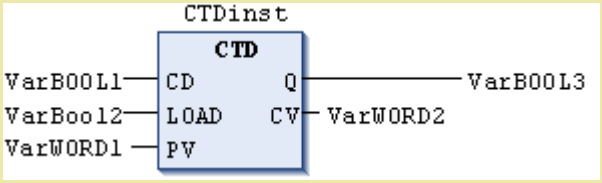
(ILの例)

```
CAL     CTDinst(
          CD:= VarBOOL1,
        LOAD:= VarBOOL2,
          PV:= VarWORD1,
         CV=> VarWORD2)
LD      CTDinst.Q
ST      VarBOOL3
```

(STの例)

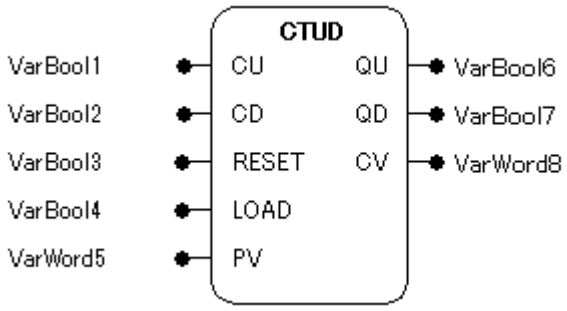
```
CTDInst(CD:= VarBOOL1, LOAD:=VarBOOL2 , PV:= VarWORD1);
VarBOOL3 := CTDInst.Q ;
VarWORD2 := CTDInst.CV;
```

(FBDの例)



CTUD : アップダウンカウンタ [FB]

(FBD の例)



機能

CUでカウントアップ、CDでカウントダウンし、カウント値が0あるいはプリセット値に達したことを知らせるカウンタです。

パラメータで使用可能なデータ型

CU, CD, RESET, LOAD: BOOL

PV: WORD

QU, QD: BOOL

CV: WORD

パラメータ

入力パラメータ	説明	備考
CU (VarBool1)	カウントトリガ入力	立ち上がりで1加算
CD (VarBool2)	カウントトリガ入力	立ち下がりで1減算
RESET (VarBool3)	リセット入力	
LOAD (VarBool4)	ロード入力	
PV (VarWord5)	カウンタ最大値	

出力パラメータ	説明	備考
QU (VarBool6)	結果 (加算してCV >= PV の場合 TRUE)	
QD (VarBool7)	結果 (減算してCV = 0 の場合 TRUE)	
CV (VarWord8)	カウント値	

解 説

入力CUの立ち上がりエッジでCVが1増加しCV = PVに達すると出力QU = TRUEが出力されます。
入力CDの立ち下がりエッジでCVが1減少しCV = 0に達すると出力 QD = TRUEが出力されます。
RESET = TRUE の場合はCV = 0、QU = FALSE、(QD = TRUE)で初期化されます。
LOAD = TRUE の場合はCV = PV、QD = FALSE、(QU = TRUE)で初期化されます。
RESETおよびLOADがFALSEでなければカウントは行われません。

宣言の例:

CTUDInst : CUTD ;

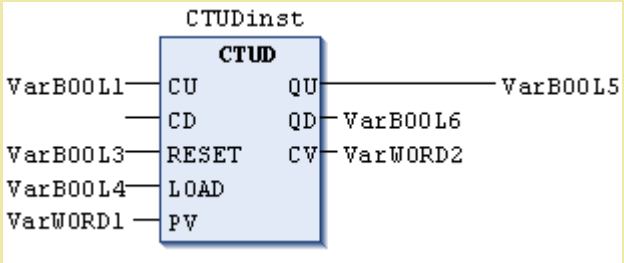
(ILの例)

```
CAL          CTUDinst(
          CU:= VarBOOL1,
       RESET:= VarBOOL3,
        LOAD:= VarBOOL4,
          PV:= VarWORD1,
          QD=> VarBOOL6,
          CV=> VarWORD2)
LD           CTUDinst.QU
ST           VarBOOL5
```

(STの例)

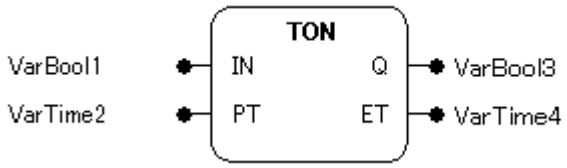
```
CTUDInst(CU := VarBOOL1, CD:= VarBOOL2, RESET := VarBOOL3,
LOAD:=VarBOOL4 , PV:= VarWORD1);
VarBOOL5 := CTUDInst.QU ;
VarBOOL6 := CTUDInst.QD ;
VarWORD2 := CTUDInst.CV;
```

(FBDの例)



TON：オンディレイタイマ [FB]

(FBD の例)



概要

入力がTRUE となってから指定の時間が経過した後に出力をTRUE にするオンディレイタイマです。

パラメータで使用可能なデータ型

IN: BOOL

PT: TIME

Q: BOOL

ET: TIME

パラメータ

入力パラメータ	説明	備考
IN (VarBool1)	開始入力	立ち上がりでET=0
PT (VarTime2)	プリセット時間	

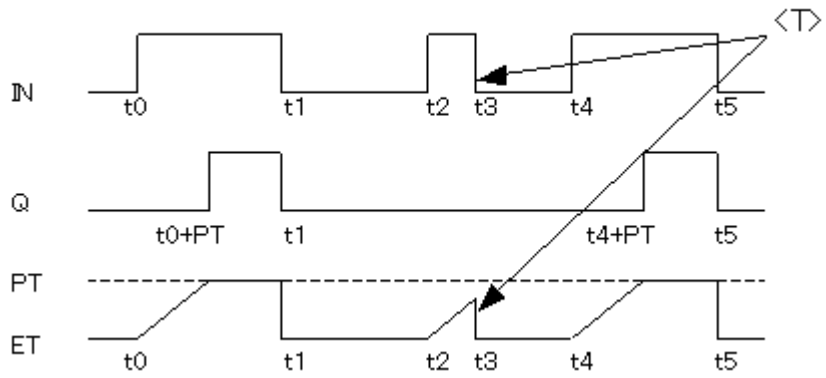
出力パラメータ	説明	備考
Q (VarBool3)	結果	IN＝TRUEかつ ET≧PTの時 TRUE
ET (VarTime4)	経過時間	

解 説

入力がOFFからONになった後に決められた時間の間出力をONすることを遅延します。

入力 IN がFALSEからTRUEに変化すると経過時間ETのカウントを開始します。経過時間ETが遅延用プリセット時間PTに達すると出力Q = TRUEが出力され、この出力は入力IN = TRUEの間保持されます。

入力INがFALSEに戻ると出力Q = FALSEが出力され、経過時間ET = 0に初期化されます。



<T>: 遅延時間に達する前に入力がFALSEとなった場合のQはTRUEにならない

宣言の例:

TONInst : TON ;

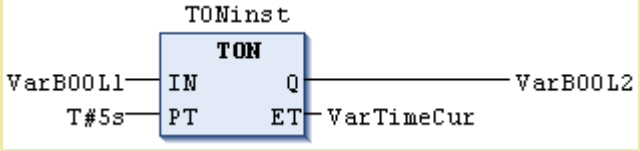
(ILの例)

```
CAL     TONinst (
        IN:= VarBOOL1 ,
        PT:= T#5s ,
        ET=> VarTimeCur )
LD      TONinst.Q
ST      VarBOOL2
```

(STの例):

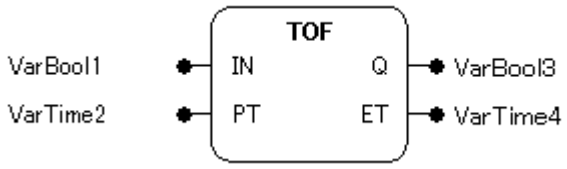
```
TONInst(IN := VarBOOL1, PT:= T#5s);
```

(FBDの例)



TOF : オフディレイタイマ [FB]

(FBD の例)



機能

入力がFALSEとなってから指定の時間が経過するまで出力をFALSEとしないオフディレイタイマです。

パラメータで使用可能なデータ型

IN: BOOL

PT: TIME

Q: BOOL

ET: TIME

パラメータ

入力パラメータ	説明	備考
IN (VarBool1)	開始入力	立ち上がりでET=0
PT (VarTime2)	プリセット時間	

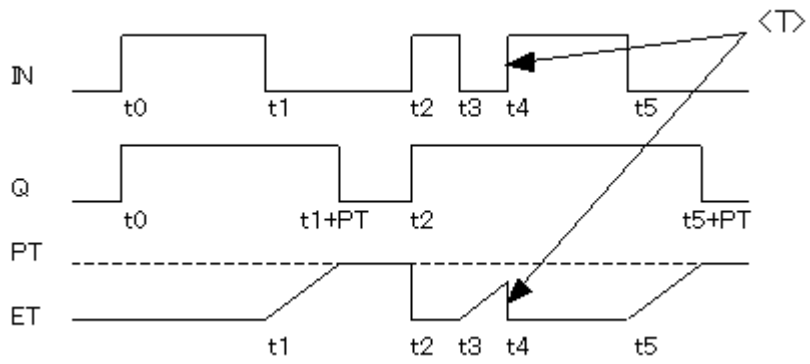
出力パラメータ	説明	備考
Q (VarBool3)	結果	IN＝FALSEかつ ET≧PTの時 FALSE
ET (VarTime4)	経過時間	

解説

入力がONからOFFになった後に決められた時間の間出力をOFFすることを遅延します。

入力 IN がTRUEからFALSEに変化すると経過時間ETのカウントを開始します。経過時間ETが遅延用プリセット時間PTに達すると出力Q = FALSEが出力され、この出力は入力IN = FALSEの間保持されます。

入力INがTRUEに戻ると出力Q = TRUEが出力され、経過時間ET = 0に初期化されます。



<T>: 遅延時間に達する前に入力がTRUEとなった場合のQはFALSEにならない

宣言の例:

TOFInst : TOF ;

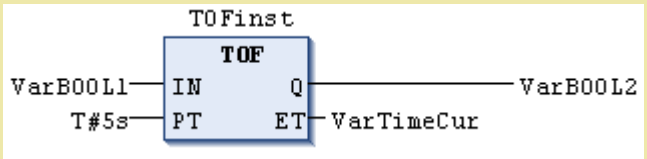
(ILの例)

```
CAL     TOFinst (
        IN:= VarBOOL1 ,
        PT:= T#5s ,
        ET=> VarTimeCur )
LD      TOFinst.Q
ST      VarBOOL2
```

(STの例)

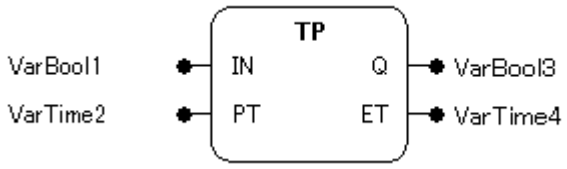
```
TOFInst(IN := VarBOOL1, PT:= T#5s);
VarBOOL2 :=TOFInst.Q;
```

(FBDの例)



TP: パルス幅出力 [FB]

(FBD の例)



機能

指定の持続時間を持ったパルスを発生するタイマです。

パラメータで使用可能なデータ型

IN: BOOL

PT: TIME

Q: BOOL

ET: TIME

パラメータ

入力パラメータ	説明	備考
IN (VarBool1)	開始入力	立ち上がりでET=0
PT (VarTime2)	プリセット時間	

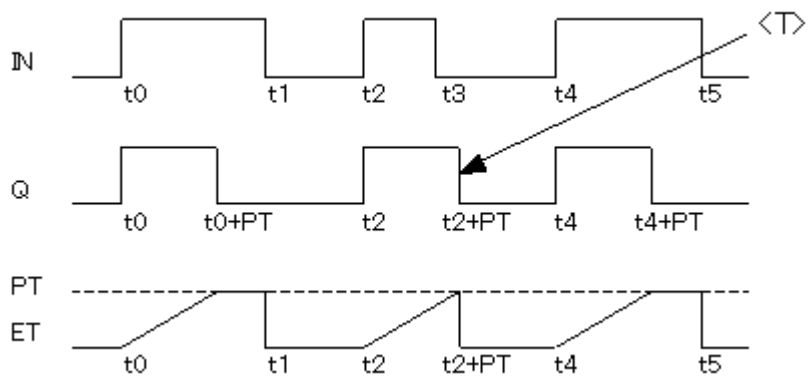
出力パラメータ	説明	備考
Q (VarBool3)	結果	IN＝TRUE後ET＜PTの間TRUE
ET (VarTime4)	経過時間	

解説

パルスを作成します。

入力INがFALSEからTRUEに変化すると出力Qにパルス用プリセット時間PTの長さでパルスが作成されます。PT時間経過前のパルス持続中に入力INが変化しても出力Qのパルス持続には影響しません。

すでに経過した時間は経過時間ETに表示されます。



<T>: 入力の立ち上がりでスタートし、入力の変化に関わらずPT幅のパルスを出力

宣言の例:

TPInst : TP ;

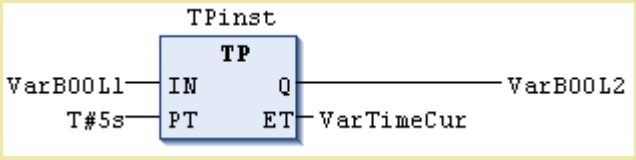
(ILの例)

```
CAL     TPinst  (
        IN:= VarBOOL1   ,
        PT:= T#5s       ,
        ET=> VarTimeCur )
LD      TPinst.Q
ST      VarBOOL2
```

(STの例)

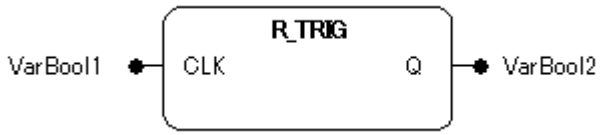
```
TPInst(IN := VarBOOL1, PT:= T#5s);
VarBOOL2 :=TPInst.Q;
```

(FBDの例)



R_TRIG: 立ち上がりエッジ検出 [FB]

(FBD の例)



機能

立ち上がりエッジ(微分)を検出します。エッジを検出したときに単一のパルスを発生します。

パラメータで使用可能なデータ型

CLK: BOOL

Q: BOOL

パラメータ

入力パラメータ	説明	備考
CLK (VarBool1)	入力	

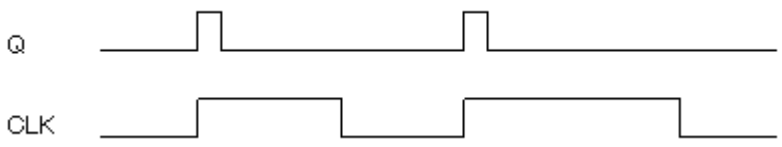
出力パラメータ	説明	備考
Q (VarBool2)	結果	

解説

立ち上がりエッジを検出します。

入力CLKで立ち上がりエッジが検出されると出力QはFALSEからTRUEに変化します。Qはファンクションブロックの次の実行(通常プログラムの1周期)までTRUEの状態を維持します。

初めてファンクションブロックが呼び出される場合は最初のエッジが検出されるまでのQはFALSEとなります。



宣言の例:

RTRIGInst : R_TRIG ;

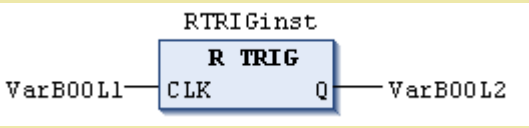
(ILの例)

```
CAL     RTRIGinst   (
    CLK:= VarBool1  )
LD      RTRIGinst.Q
ST      VarBool2
```

(STの例)

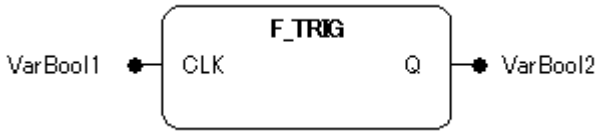
```
RTRIGInst(CLK:= VarBOOL1);
VarBOOL2 := RTRIGInst.Q;
```

(FBDの例)



F_TRIG:立ち下がりエッジ検出 [FB]

(FBD の例)



機能

立ち下がりエッジ(微分)を検出します。エッジを検出したときに単一のパルスを発生します。

パラメータで使用可能なデータ型

CLK: BOOL

Q: BOOL

パラメータ

入力パラメータ	説明	備考
CLK (VarBool1)	入力	

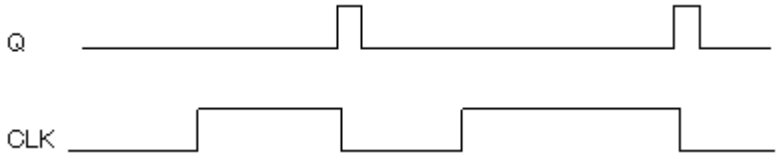
出力パラメータ	説明	備考
Q (VarBool2)	結果	

解 説

立ち下がりエッジを検出します。

入力CLKで立ち下がりエッジが検出されると出力QはFALSEからTRUEに変化します。Qはファンクションブロックの次の実行(通常プログラムの1周期)までTRUEの状態を維持します。

初めてファンクションブロックが呼び出される場合は最初のエッジが検出されるまでのQはFALSEとなります。



宣言の例:

FTRIGInst : F_TRIG ;

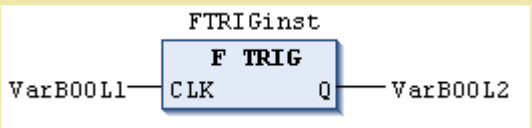
(ILの例)

```
CAL     FTRIGinst   (
    CLK:= VarBOOL1  )
LD      FTRIGinst.Q
ST      VarBOOL2
```

(STの例)

```
FTRIGInst(CLK:= VarBOOL1);
VarBOOL2 := FTRIGInst.Q;
```

(FBDの例)



(このページは空白です)

9.コントローラ専用ライブラリ

コントローラ専用ライブラリー覧

特別な演算や入出力カードのアクセスに必要なファンクション、ファンクションブロックをライブラリとして提供しています。これらのライブラリは機種別に提供されるパッケージに含まれており、パッケージをインストールすることで適切なライブラリがインストールされます。

ライブラリ	機能	Namespace
MsysLibsDDC_Cnt	Container library (MsysDefine, MsysDDC, MsysSystem, MsysUtility)	
MsysDefine	Definitions	MSYS
MsysDDC	DDC Functions	MSYS_DDC
MsysR3Standard	R3 Standard Functions	MSYS_R3Standard
MsysSystem	System Functions	MSYS_System
MsysUtility	Utility Functions	MSYS_Utility
MsysBA3CE	BA3-CE10 Functions	MSYS_BA3CE

MsysBA3CE

MsysBA3CE POUs

OP(Operand), FUN(Function), FB(Function Block), DUT(Data Unit Type)

名称	属性	機能	サポート Library *1
MODBUSSLAVE_ERROR_Enm	DUT	Modbus Slave エラーコード 列挙型	
ModbusSlaveGetInfo	FB	MODBUS SLAVE 情報取得	
ModbusSlaveGetBit	FB	MODBUS SLAVE Discrete input/Coil output からの読み込み	
ModbusSlaveGet16	FB	MODBUS SLAVE Input registerHolding register からの16bit読み込み	
ModbusSlaveGet32	FB	MODBUS SLAVE Input register/Holding register からの32bit読み込み	
ModbusSlaveGetREAL	FB	MODBUS SLAVE Input register/Holding register からのREAL(32bit)読み込み	
ModbusSlaveGetLREAL	FB	MODBUS SLAVE Input register/Holding register からのLREAL(64bit)読み込み	
ModbusSlaveSetBit	FB	MODBUS SLAVE Discrete input/Coil output への書き込み	
ModbusSlaveSet16	FB	MODBUS SLAVE Input register/Holding registerへの16bit書き込み	
ModbusSlaveSet32	FB	MODBUS SLAVE Input register/Holding register への32bit書き込み	
ModbusSlaveSetREAL	FB	MODBUS SLAVE Input register/Holding register へのREAL(32bit)書き込み	
ModbusSlaveSetLREAL	FB	MODBUS SLAVE Input register/Holding register へのLREAL(64bit)書き込み	

*1) サポートLibrary欄は、そのファンクションあるいはファンクションブロックのサポートを開始したライブラリのバージョンを記述しています。この欄が空の場合はv1.0.0以降でサポートしていることを示します。

MODBUSSLAVE_ERROR_Enm [DUT]

Modbus Slave エラーコード 列挙型

識別子	値	説明
NO_ERROR	0	正常（エラーなし）
SYS_STRUCT_SIZE	1	System error
MATH_DivByZero	11	演算で0割が発生
PARA_ARG	100	入力パラメータが範囲外
PARA_IsNaN	101	必須入力パラメータが設定されていないか NaN値が設定されている
PARA_IsNullPointer	102	必須入力パラメータが設定されていないか Nullポインタが設定されている
PARA_Raw_Range	110	上下限設定が範囲外
PARA_Scale_Range	111	スケール設定が範囲外
PARA_MinMax_Range	113	上下限値が範囲外
PARA_ByteOrder_Range	114	バイト順指定が範囲外
PARA_MODBUS_Address	400	要求の Modbus Address が範囲外
PARA_MODBUS_Address_Range	401	要求の Address はサポート範囲外
PARA_MODBUS_ApiError	409	MODBUS (see uiErrorSubCode)

ModbusSlaveGetInfo [FB]

MODBUS SLAVE 通信情報取得

(INPUT)

記号	パラメータ	型	説明
xEnable	ENABLE	BOOL	FALSE: 演算スキップ, TRUE: 演算 Default (TRUE)
xReset	リセット（統計情報0リセット）	BOOL	立ち上がりでリセット Default (FALSE)

(OUTPUT)

記号	パラメータ	型	説明
xDone	PROCESSING STATUS	BOOL	完了でTRUE
xError	ERROR STATUS	BOOL	エラー検出でTRUE
eError	ERROR CODE	MODBUSSLAVE ERROR Enm	エラーコード 0:No Error
uiState	状態	UINT	
uiConnection	接続クライアント数	UINT	
uiFC1	統計情報: Read Coils 要求数	UINT	
uiFC2	統計情報: Read Discrete Inputs 要求数	UINT	
uiFC3	統計情報: Read Holding Registers 要求数	UINT	
uiFC4	統計情報: Read Input Registers 要求数	UINT	
uiFC5	統計情報: Write Single Coil 要求数	UINT	
uiFC6	統計情報: Write Single Register 要求数	UINT	
uiFC15	統計情報: Write Multiple Coils 要求数	UINT	
uiFC16	統計情報: Write Multiple Registers 要求数	UINT	

解 説

Modbus/TCP スレーブ通信の統計情報を返します。

ModbusSlaveGetBit [FB]

MODBUS SLAVE Discrete input/Coil output からの読み込み

(INPUT)

記号	パラメータ	型	説明
xEnable	ENABLE	BOOL	FALSE: 演算スキップ, TRUE: 演算 Default (TRUE)
diAddress	Modbusアドレス 002049 ～ 004096 (xUpdated 有効) 102049 ～ 104096 (xUpdated 無効)	DINT	Default (0)

(OUTPUT)

記号	パラメータ	型	説明
xDone	PROCESSING STATUS	BOOL	完了でTRUE
xError	ERROR STATUS	BOOL	エラー検出でTRUE
eError	ERROR CODE	MODBUSSLAVE ERROR Enm	エラーコード 0:No Error
xUpdated	値更新	BOOL	更新検出時にTRUE (*1)
xVout	出力	BOOL	xEnable=FALSE 時は FALSE

(*1) xUpdated はファンクションブロック実行毎に更新があった場合のみ TRUE を返します。これは次のファンクションブロックの呼び出しまでに更新がなければ次回の呼び出しで返される xUpdated は FALSE になります。

解 説

Modbus通信バッファの Discrete input または Coil output から BIT(1bit) 入力した結果を xVout に出力します。

ModbusSlaveGet16 [FB]

MODBUS SLAVE Input register/Holding register からの16bit読み込み

(INPUT)

記号	パラメータ	型	説明
xEnable	ENABLE	BOOL	FALSE: 演算スキップ, TRUE: 演算 Default (TRUE)
diAddress	Modbusアドレス 400513 ～ 401024 (xUpdated 有効) 421281 ～ 428192 (xUpdated 無効)	DINT	Default (0)
eRawByteOrder	Modbus dataのバイト順	MSYS ByteOrder Enm	16bit用 Default (R_INT16)
rBase	Modbus dataのベース値	REAL	Default (0.0)
rRawL	Modbus dataの下限値	REAL	Default (0.0) (*1)
rRawH	Modbus dataの上限値	REAL	Default (0.0) (*1)
rScaleL	Modbus dataの下限に割り当てる値	REAL	Default (0.0) (*2)
rScaleH	Modbus dataの上限に割り当てる値	REAL	Default (0.0) (*2)
rOffset	内部計算結果に対してこのオフセットを加算し出力(rVout)とします。	REAL	Default (0.0)

(OUTPUT)

記号	パラメータ	型	説明
xDone	PROCESSING STATUS	BOOL	完了でTRUE
xError	ERROR STATUS	BOOL	エラー検出でTRUE
eError	ERROR CODE	MODBUSSLAVE ERROR Enm	エラーコード 0:No Error
xUpdated	値更新	BOOL	更新検出時にTRUE (*3)
wVout	変換前出力	WORD	エラー時は0
rVout	変換後出力	REAL	エラー時はNaN xEnable=FALSE 時は NaN

(*1) rRawL = rRawH の場合は、スケール変換を行いません。

(*2) rScaleL = rScaleH の場合は、上下限制限を行いません。

(*3) xUpdated はファンクションブロック実行毎に更新があった場合のみ TRUE を返します。これは次のファンクションブロックの呼び出しまでに更新がなければ次回の呼び出しで返される xUpdated は FALSE になります。

解 説

Modbus通信バッファ(Input registerまたはHolding register)から16bit入力し下記の計算式で結果をwVout,rVoutに出力します。

rVout = Limit2((Limit1(X + b) - rRawL) * a + rScaleL) + of)

X: ハードウェア入力値

a: (rScaleH – rScaleL) / (rRawH – rRawL)

b: rBase

of: rOffset

Limit1: rRawH ～ rRawL

Limit2: rScaleH ～ rScaleL

[rRawH == rRawL の場合]

rVout = Limit2(X + b + of)

[rRawH == rRawL AND rScaleH == rScaleLの場合]

rVout = X + b + of

■「指定可能 MSYS_ByteOrder_Enm」

INT16

UINT16

R_INT16

R_UINT16

(例 1)次 のModbusデータを値 -1234 (0xFB2E)として取 得 する場 合

アドレス	データ (byte)	LITTLE-ENDIAN
0	0xFB	H
1	0x2E	L

eRawByteOrder にはR_INT16 を設定する

(例２)次 のModbusデータを値 0x1234として取 得 する場 合

アドレス	データ (byte)	LITTLE-ENDIAN
0	0x12	H
1	0x34	L

eRawByteOrder にはR_UINT16 を設定する

補 足

変換前出力が符号ありの場合はWORD_TO_INT( ) で変換してください。

ModbusSlaveGet32 [FB]

MODBUS SLAVE Input register/Holding register からの32bit読み込み

(INPUT)

記号	パラメータ	型	説明
xEnable	ENABLE	BOOL	FALSE: 演算スキップ, TRUE: 演算 Default (TRUE)
diAddress	Modbusアドレス 400513 ～ 401023 (xUpdated 有効) 421281 ～ 428191 (xUpdated 無効)	DINT	Default (0)
eRawByteOrder	Modbus dataのバイト順	MSYS ByteOrder Enm	32bit用 Default (R_INT32WS)
lrBase	Modbus dataのベース値	LREAL	Default (0.0)

記号	パラメータ	型	説明
lrRawL	Modbus dataの下限値	LREAL	Default (0.0) (*1)
lrRawH	Modbus dataの上限値	LREAL	Default (0.0) (*1)
lrScaleL	Modbus dataの下限に割り当てる値	LREAL	Default (0.0) (*2)
lrScaleH	Modbus dataの上限に割り当てる値	LREAL	Default (0.0) (*2)
lrOffset	内部計算結果に対してこのオフセットを加算し出力(rVout)とします。	LREAL	Default (0.0)

(OUTPUT)

記号	パラメータ	型	説明
xDone	PROCESSING STATUS	BOOL	完了でTRUE
xError	ERROR STATUS	BOOL	エラー検出でTRUE
eError	ERROR CODE	MODBUSSLAVE ERROR Enm	エラーコード 0:No Error
xUpdated	値更新	BOOL	更新検出時にTRUE (*3)
dwVout	変換前出力	DWORD	エラー時は0
lrVout	変換後出力	LREAL	エラー時はNaN xEnable=FALSE 時は NaN

(*1) lrRawL = lrRawH の場合は、スケール変換を行いません。

(*2) lrScaleL = lrScaleH の場合は、上下限制限を行いません。

(*3) xUpdated はファンクションブロック実行毎に更新があった場合のみ TRUE を返します。これは次のファンクションブロックの呼び出しまでに更新がなければ次回の呼び出しで返される xUpdated は FALSE になります。

解 説

Modbus通信バッファ(Input registerまたはHolding register)から32bit入力し下記の計算式で結果をdwVout,lrVoutに出力します。

lrVout = Limit2((Limit1(X + b) - lrRawL) * a + lrScaleL) + of)

X: ハードウェア入力値

a: (lrScaleH – lrScaleL) / (lrRawH – lrRawL)

b: lrBase

of: lrOffset

Limit1: lrRawH ～ lrRawL

Limit2: lrScaleH ～ lrScaleL

[lrRawH == lrRawL の場合]

lrVout = Limit2(X + b + of)

[lrRawH == lrRawL AND lrScaleH == lrScaleLの場合]

lrVout = X + b + of

■「指定可能 MSYS_ByteOrder_Enm」

INT32
UINT32
FLOAT32
R_INT32
R_UINT32
R_FLOAT32
INT32WS
UINT32WS
FLOAT32WS
R_INT32WS
R_UINT32WS
R_FLOAT32WS

(例１)次のModbusデータを値 -12345678 (0xFF439EB2)として取得する場合

アドレス	データ (byte)	LITTLE-ENDIAN	
0	0x9E	H	L-WORD
1	0xB2	L	
2	0xFF	H	H-WORD
3	0x43	L	

eRawByteOrder にはR_INT32WS を設定する

(例２)次のModbusデータを値0x12345678として取得する場合

アドレス	データ (byte)	LITTLE-ENDIAN	
0	0x56	H	L-WORD
1	0x78	L	
2	0x12	H	H-WORD
3	0x34	L	

eRawByteOrder にはR_UINT32WS を設定する

補 足

変換前出力が符号ありの場合はDWORD_TO_DINT( ) で変換します。

ModbusSlaveGetREAL [FB]

MODBUS SLAVE Input register/Holding register からのREAL(32bit)読み込み

(INPUT)

記号	パラメータ	型	説明
xEnable	ENABLE	BOOL	FALSE: 演算スキップ, TRUE: 演算 Default (TRUE)
diAddress	Modbusアドレス 400513 ～ 401023 (xUpdated 有効) 421281 ～ 428191 (xUpdated 無効)	DINT	Default (0)
eRawByteOrder	Modbus dataのバイト順	MSYS_ByteOrder_Enm	32bit用 Default (R_FLOAT32WS)

(OUTPUT)

記号	パラメータ	型	説明
xDone	PROCESSING STATUS	BOOL	完了でTRUE
xError	ERROR STATUS	BOOL	エラー検出でTRUE
eError	ERROR CODE	MODBUSSLAVE_ERROR_Enm	エラーコード 0:No Error
xUpdated	値更新	BOOL	更新検出時にTRUE (*1)
rVout	出力	REAL	エラー時はNaN xEnable=FALSE 時は FALSE

(*1) xUpdated はファンクションブロック実行毎に更新があった場合のみ TRUE を返します。これは次のファンクションブロックの呼び出しまでに更新がなければ次回の呼び出しで返される xUpdated は FALSE になります。

解 説

Modbus通信バッファの Input register または Holding register からREAL(32bit)入力した結果を rVout に出力します。

■「指定可能 MSYS_ByteOrder_Enm」

FLOAT32

R_FLOAT32

FLOAT32WS

R_FLOAT32WS

ModbusSlaveGetLREAL [FB]

MODBUS SLAVE Input register/Holding register からのLREAL(64bit)読み込み

(INPUT)

記号	パラメータ	型	説明
xEnable	ENABLE	BOOL	FALSE: 演算スキップ, TRUE: 演算 Default (TRUE)
diAddress	Modbusアドレス 400513 ～ 401021 (xUpdated 有効) 421281 ～ 428189 (xUpdated 無効)	DINT	Default (0)
eRawByteOrder	Modbus dataのバイト順	MSYS_ByteOrder_Enm	64bit用 Default (R_FLOAT64WS)

(OUTPUT)

記号	パラメータ	型	説明
xDone	PROCESSING STATUS	BOOL	完了でTRUE
xError	ERROR STATUS	BOOL	エラー検出でTRUE
eError	ERROR CODE	MODBUSSLAVE_ERROR_Enm	エラーコード 0:No Error
xUpdated	値更新	BOOL	更新検出時にTRUE (*1)
lrVout	出力	LREAL	エラー時はNaN xEnable=FALSE 時は NaN

(*1) xUpdated はファンクションブロック実行毎に更新があった場合のみ TRUE を返します。これは次のファンクションブロックの呼び出しまでに更新がなければ次回の呼び出しで返される xUpdated は FALSE になります。

解 説

Modbus通信バッファの Input register または Holding register から64bit入力した結果を lrVout に出力します。

■「指定可能 MSYS_ByteOrder_Enm」

FLOAT64
R_FLOAT64
FLOAT64WS
R_FLOAT64WS

ModbusSlaveSetBit [FB]

MODBUS SLAVE Discrete input/Coil output への書き込み

(INPUT)

記号	パラメータ	型	説明
xEnable	ENABLE	BOOL	FALSE: 演算スキップ, TRUE: 演算 Default (TRUE) (*1)
xVin	出力値	BOOL	Default (FALSE) (*1)
diAddress	Modbusアドレス 002049 ～ 004096 (xUpdated 有効) 102049 ～ 104096 (xUpdated 無効)	DINT	Default (0) (*1)

(OUTPUT)

記号	パラメータ	型	説明
xDone	PROCESSING STATUS	BOOL	完了でTRUE
xError	ERROR STATUS	BOOL	エラー検出でTRUE
eError	ERROR CODE	MODBUSSLAVE ERROR Enm	エラーコード 0:No Error
xVout	出力値のコピー	BOOL	xEnable=FALSE とエラー時はFALSE

(*1) xEnable は立ち上がり、それ以外は値変化で実行されます。

解 説

Modbus通信バッファの Discrete input または Coil output へBIT(1bit)を出力します。

ModbusSlaveSet16 [FB]

MODBUS SLAVE Input register/Holding registerへの16bit書き込み

(INPUT)

記号	パラメータ	型	説明
xEnable	ENABLE	BOOL	FALSE: 演算スキップ, TRUE: 演算 Default (TRUE) (*1)
rVin	出力値	REAL	Default (NaN) (*1) NaNの場合は0.0とする
diAddress	Modbusアドレス 300513 ～ 301024 (xUpdated 無効) 400513 ～ 401024 (xUpdated 有効) 421281 ～ 428192 (xUpdated 無効)	DINT	Default (0) (*1)
eRawByteOrder	Modbus dataのバイト順	MSYS_ByteOrder_Enm	16bit用 Default (R_INT16)
rBase	Modbus dataのベース値	REAL	Default (0.0)
rRawL	Modbus dataの下限値	REAL	Default (0.0) (*2)
rRawH	Modbus dataの上限値	REAL	Default (0.0) (*2)
rScaleL	Modbus dataの下限に割り当てる値	REAL	Default (0.0) (*3)
rScaleH	Modbus dataの上限に割り当てる値	REAL	Default (0.0) (*3)
rOffset	内部計算結果に対して加算されていたオフセット	REAL	Default (0.0)

(OUTPUT)

記号	パラメータ	型	説明
xDone	PROCESSING STATUS	BOOL	完了でTRUE
xError	ERROR STATUS	BOOL	エラー検出でTRUE
eError	ERROR CODE	MODBUSSLAVE_ERROR_Enm	エラーコード 0:No Error
wVout	出力値のコピー	WORD	xEnable=FALSE とエラー時は0

(*1) xEnable は立ち上がり、それ以外は値変化で実行されます。

(*2) rRawL = rRawH の場合は、スケール変換を行いません。

(*3) rScaleL = rScaleH の場合は、上下限制限を行いません。

解 説

下記の計算式で演算した結果をModbus通信バッファのInput registerまたはHolding registerへ16bitで出力します。

wVout = Limit1((Limit2(rVin – of) - rScaleH) / a + rRawL + b)

a: (rScaleH – rScaleL) / (rRawH – rRawL)

b: rBase

of: rOffset

Limit1: rRawH ～ rRawL

Limit2: rScaleH ～ rScaleL

[rRawH == rRawL の場合]

wVout = Limit2(rVi - of + b)

[rRawH == rRawL AND rScaleH == rScaleLの場合]

wVout = rVi - of + b

■「指定可能 MSYS_ByteOrder_Enm」

INT16

UINT16

R_INT16

R_UINT16

(例1)次のModbusデータを値 -1234 (0xFB2E)として取得する場合

アドレス	データ (byte)	LITTLE-ENDIAN
0	0xFB	H
1	0x2E	L

eRawByteOrderにはR_INT16を設定する

(例)次のModbusデータを値0x1234として取得する場合

アドレス	データ (byte)	LITTLE-ENDIAN
0	0x12	H
1	0x34	L

eRawByteOrder にはR_UINT16 を設定する

ModbusSlaveSet32 [FB]

MODBUS SLAVE Input register/Holding register への32bit書き込み

(INPUT)

記号	パラメータ	型	説明
xEnable	ENABLE	BOOL	FALSE：演算スキップ，TRUE：演算 Default (TRUE) (*1)
lrVin	出力値	LREAL	Default (NaN) (*1)
diAddress	Modbusアドレス 300513 ～ 301023 (xUpdated 無効) 400513 ～ 401023 (xUpdated 有効) 421281 ～ 428191 (xUpdated 無効)	DINT	Default (0) (*1)
eRawByteOrder	Modbus dataのバイト順	MSYS_ByteOrder_Enm	32bit用 Default (R_INT32WS)
lrBase	Modbus dataのベース値	LREAL	Default (0.0)
lrRawL	Modbus dataの下限値	LREAL	Default (0.0) (*2)
lrRawH	Modbus dataの上限値	LREAL	Default (0.0) (*2)
lrScaleL	Modbus dataの下限に割り当てる値	LREAL	Default (0.0) (*3)
lrScaleH	Modbus dataの上限に割り当てる値	LREAL	Default (0.0) (*3)
lrOffset	内部計算結果に対して加算されていたオフセット	LREAL	Default (0.0)

(OUTPUT)

記号	パラメータ	型	説明
xDone	PROCESSING STATUS	BOOL	完了でTRUE
xError	ERROR STATUS	BOOL	エラー検出でTRUE
eError	ERROR CODE	MODBUSSLAVE_ERROR_Enm	エラーコード 0:No Error
dwVout	出力値のコピー	DWORD	xEnable=FALSE とエラー時は0

(*1) xEnable は立ち上がり、それ以外は値変化で実行されます。

(*2) lrRawL = lrRawH の場合は、スケール変換を行いません。

(*3) lrScaleL = lrScaleH の場合は、上下限制限を行いません。

解 説

下記の計算式で演算した結果をModbus通信バッファの Input register または Holding register へ32bitで出力します。

dwVout = Limit1((Limit2(lrVin – of) - lrScaleL) / a + lrRawL + b)

a: (lrScaleH – lrScaleL) / (lrRawH – lrRawL)

b: lrBase

of: lrOffset

Limit1: lrRawH ～ lrRawL

Limit2: lrScaleH ～ lrScaleL

[lrRawH == lrRawL の場合]

dwVout = Limit2(lrVin - of + b)

[lrRawH == lrRawL AND lrScaleH == lrScaleLの場合]

dwVout = lrVin - of + b

■「指定可能 MSYS_ByteOrder_Enm」

INT32
UINT32
FLOAT32
R_INT32
R_UINT32
R_FLOAT32
INT32WS
UINT32WS
FLOAT32WS
R_INT32WS
R_UINT32WS
R_FLOAT32WS

(例１)次のModbusデータを値 -12345678 (0xFF439EB2)として取得する場合

アドレス	データ (byte)	LITTLE-ENDIAN	
0	0x9E	H	L-WORD
1	0xB2	L	
2	0xFF	H	H-WORD
3	0x43	L	

eRawByteOrder にはR_INT32WS を設定する

(例２)次のModbusデータを値0x12345678として取得する場合

アドレス	データ (byte)	LITTLE-ENDIAN	
0	0x56	H	L-WORD
1	0x78	L	
2	0x12	H	H-WORD
3	0x34	L	

eRawByteOrder にはR_UINT32WS を設定する

ModbusSlaveSetREAL [FB]

MODBUS SLAVE Input register/Holding register へのREAL(32bit)書き込み

(INPUT)

記号	パラメータ	型	説明
xEnable	ENABLE	BOOL	FALSE: 演算スキップ, TRUE: 演算 Default (TRUE) (*1)
rvVn	出力値	REAL	Default (NaN) (*1)
diAddress	Modbusアドレス 300513 ～ 301023 (xUpdated 無効) 400513 ～ 401023 (xUpdated 有効) 421281 ～ 428191 (xUpdated 無効)	DINT	Default (0) (*1)
eRawByteOrder	Modbus dataのバイト順	MSYS_ByteOrder_Enm	32bit用 Default (R_FLOAT32WS)

(OUTPUT)

記号	パラメータ	型	説明
xDone	PROCESSING STATUS	BOOL	完了でTRUE
xError	ERROR STATUS	BOOL	エラー検出でTRUE
eError	ERROR CODE	MODBUSSLAVE ERROR Enm	エラーコード 0:No Error

(*1) xEnable は立ち上がり、それ以外は値変化で実行されます。

解 説

Modbus通信バッファの Input register または Holding register にREAL(32bit)を出力します。

■「指定可能 MSYS_ByteOrder_Enm」

FLOAT32
R_FLOAT32
FLOAT32WS
R_FLOAT32WS

ModbusSlaveSetLREAL [FB]

MODBUS SLAVE Input register/Holding registerへのLREAL(64bit)書き込み

(INPUT)

記号	パラメータ	型	説明
xEnable	ENABLE	BOOL	FALSE: 演算スキップ, TRUE: 演算 Default (TRUE) (*1)
lrvVn	出力値	LREAL	Default (NaN) (*1)
diAddress	Modbusアドレス 300513 ～ 301021 (xUpdated 無効) 400513 ～ 401021 (xUpdated 有効) 421281 ～ 428189 (xUpdated 無効)	DINT	Default (0) (*1)
eRawByteOrder	Modbus dataのバイト順	MSYS ByteOrder Enm	64bit用 Default (R_FLOAT64WS)

(OUTPUT)

記号	パラメータ	型	説明
xDone	PROCESSING STATUS	BOOL	完了でTRUE
xError	ERROR STATUS	BOOL	エラー検出でTRUE
eError	ERROR CODE	MODBUSSLAVE ERROR Enm	エラーコード 0:No Error

(*1) xEnable は立ち上がり、それ以外は値変化で実行されます。

解 説

Modbus通信バッファの Input register または Holding register にLREAL(64bit)を出力します。

■「指定可能 MSYS_ByteOrder_Enm」

FLOAT64

R_FLOAT64

FLOAT64WS

R_FLOAT64WS

MsysIoDrvBA3DLink

MsysBA3DLink POUs

OP(Operand), FUN(Function), FB(Function Block), DUT(Data Unit Type)

名称	属性	機能	サポート Library *1
BA3DLINK_ERROR_Enm	DUT	BA3DLINK エラーコード 列挙型	
BA3DlinkPointGetValue	FUN	グローバルデータポイントからデータの取得	
BA3DLinkPointSetValue	FUN	グローバルデータポイントへのデータ設定	

*1) サポートLibrary欄は、そのファンクションあるいはファンクションブロックのサポートを開始したライブラリのバージョンを記述しています。この欄が空の場合はv1.0.0以降でサポートしていることを示します。

BA3DLINK_ERROR_Enm [DUT]

BA3DLINK エラーコード 列挙型

識別子	値	説明
NO_ERROR	0	正常（エラーなし）
SYS_STRUCT_SIZE	1	System error
MATH_DivByZero	11	演算で0割が発生
PARA_ARG	100	入力パラメータが範囲外
API_NOT_READY	500	API が初期化されていない
API_SYNCOBJ_NOT_READY	501	APIの初期化失敗
PARA_DLINK_PointNo	502	ポイント番号が範囲外
PARA_DLINK_PriorityNo	503	優先度値が範囲外
GENERIC_ERROR	32767	その他エラー

MsysBA3DLinkPointGetValue [FUN]

グローバルデータポイントから値の取得

(INPUT)

記号	パラメータ	型	説明
nPointNo	ポイント番号	INT	1 ～ 256
nPriorityNo	優先度	INT	0 ～ 5
plrVAL	取得した値を格納する変数	POINTER TO LREAL	
pnQTY	取得した品質値を格納する変数	POINTER TO INT	
pxUpdated	取得した更新状態を格納する変数	POINTER TO BOOL	TRUE: 更新あり, FALSE:更新なし

(OUTPUT)

記号	パラメータ	型	説明
(RETURN)	結果	BA3DLINK_ERROR_Enm	成功はNO_ERROR

解 説

グローバルデータポイントから最終に更新された値(現在値)を取得します。

データが無効な場合やエラーが発生した場合はデータの品質値に反映されます。

MsysBA3DLinkPointSetValue [FUN]

グローバルデータポイントへ値の設定

(INPUT)

記号	パラメータ	型	説明
nPointNo	ポイント番号	INT	1 ～ 256
nPriorityNo	優先度	INT	-1, 0 ～ 5 (-1 は lrVAL を無効とし、0 は 5 を指定したものとして動作します。)
lrVAL	取得した値を格納する変数	POINTER TO LREAL	非数(NaN)値の書き込みは指定の優先度の設定値を解除します。
nQTY	取得した品質値を格納する変数	POINTER TO INT	品質値-1の書き込みはQTYの更新を行いません。

(OUTPUT)

記号	パラメータ	型	説明
(RETURN)	結果	BA3DLINK_ERROR_Enm	成功はNO_ERROR

解 説

グローバルデータポイントの指定優先度に値と品質値を設定します。
この関数が完了しても他のコントローラのポイントデータが瞬時に更新されるわけではありません。本関数で書き込まれた値が他コントローラのデータポイントに反映されるまでにはSubGlobalDataPointTaskにより他コントローラ向けに放送され、相手側のSubGlobalDataPointTaskにより受信されるまでの時間遅れが生じます。

DDC関連

MsysDDC POUs

OP(Operand), FUN(Function), FB(Function Block), DUT(Data Unit Type)

名称	属性	機能	サポート Library *1
DdcAnaLinear	FB	折れ線リニアライザ	
DdcCalorie	FB	熱量演算	
DdcCore	FB	システム基本処理（システム関数）	
DdcCycTimer	FB	サイクリックタイマ	
DdcDualDelayTimer	FB	デュアルディレイタイマ	
DdcEnthalpy	FB	エンタルピ演算	
DdcFilter	FB	一次遅れフィルタ	
DdcF_Compare, DdcR_Compare	FB	ヒステリシス付き比較	
DdcLoadReset	FB	給気温度最適化制御	
DdcLoopSingle	FB	PID演算	
DdcMomentaryOutput	FB	モメンタリ出力	
DdcMvLimit	FB	変化量制限	
DdcPointHistory	FB	変数値の履歴書き込み	
DdcPulseCounter	FB	パルスカウント	
DdcRtcNow	FB	現在日付時刻の取得	
DdcWeightedAverage	FB	加重平均	
DdcSetRealNaN	FUN	REAL(32bit)型変数値をNaNに設定	
DdcSetLRealNaN	FUN	LREAL(64bit)型変数値をNaNに設定	
Ddc_IsRealNaN	FUN	REAL(32bit)型変数値がNaNであるか判定	
Ddc_IsLRealNaN	FUN	LREAL(64bit)型変数値がNaNであるか判定	

*1) サポートLibrary欄は、そのファンクションあるいはファンクションブロックのサポートを開始したライブラリのバージョンを記述しています。この欄が空の場合はv1.0.0以降でサポートしていることを示します。

Ddc_ERROR_Enm [DUT]

DDC エラーコード 列挙型

識別子	値	説明
NO_ERROR	0	正常（エラーなし）
MATH_DivByZero	11	演算で0割が発生
PARA_ARG	100	入力パラメータが範囲外
PARA_IsNaN	101	必須入力パラメータが設定されていないか NaN値が設定されている
PARA_IsNullPointer	102	必須入力パラメータが設定されていないか Nullポインタが設定されている
PARA_Raw_Range	110	上下限設定が範囲外
PARA_Scale_Range	111	スケール設定が範囲外
PARA_Vin_Range	112	Vin入力値が範囲外
PARA_MinMax_Range	113	上下限値が範囲外
PARA_ByteOrder_Range	114	バイト順指定が範囲外
PARA_UNIT	115	UNIT値が範囲外
PARA_PID_PB_IsZero	116	PB値がゼロ
PARA_PID_Vout_Empty	117	演算エラーのためVoutに結果は格納されていない
PARA_PID_Vout_Range	118	演算結果は規定の範囲を超えている
TIME_DATA	119	時刻値が範囲外 RTCNOW

DdcAnaLinear [FB]

折れ線リニアライザ

(INPUT)

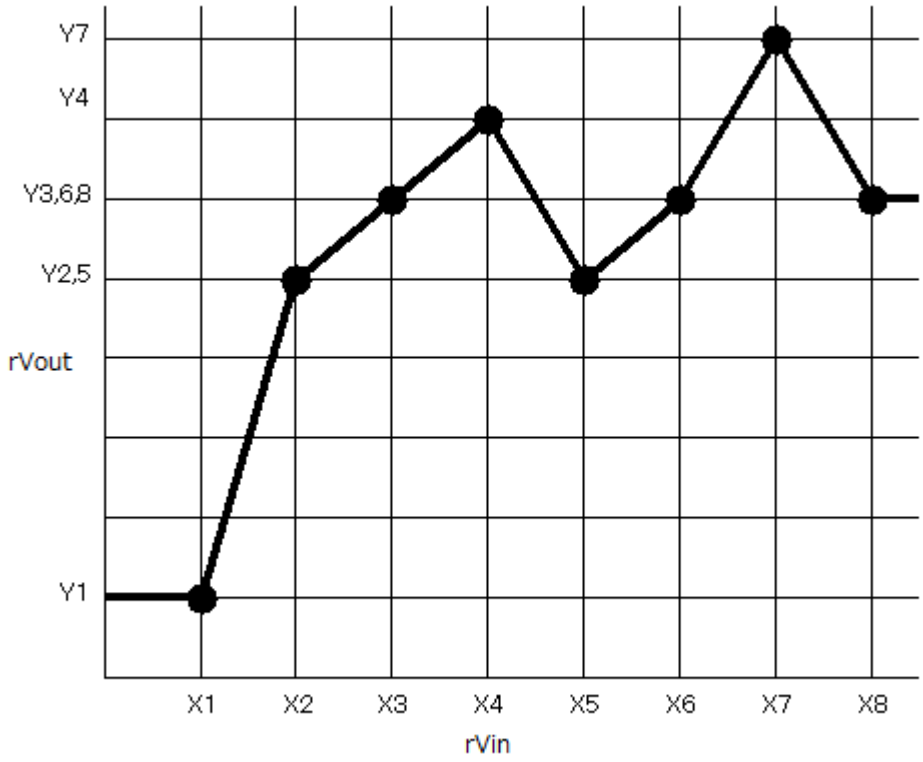
記号	パラメータ	型	説明
xEnable	Enable	BOOL	FALSE: 演算スキップ, TRUE: 演算 Default (TRUE)
rVin	入力	REAL	Default (NaN)
rX1 .. 8	X座標	REAL	Default (NaN)
rY1 .. 8	Y座標	REAL	Default (NaN)

(OUTPUT)

記号	パラメータ	型	説明
xDone	PROCESSING STATUS	BOOL	完了でTRUE
xError	ERROR STATUS	BOOL	エラー検出でTRUE
eError	ERROR CODE	DDC ERROR Enm	内部エラーコード 0:No Error
rVout	出力	REAL	折れ線補正演算結果、または NaN xEnable=FALSE 時は NaN

解 説

入力rVinを折れ線補正し出力します。



■rXn <= rVin < rXn+1 の場合

$$rVout = \frac{rY_{n+1} - rY_n}{rX_{n+1} - rX_n}(rVin - rX_n) + rY_n$$

補 足

1. Y座標は昇順である必要はないがX座標は昇順であること。

   X座標の昇順が崩れた場合はその直前までが有効とします。

   例えば、X1 <= X2 <= X3 > X4 の場合はX3までが有効となります。

2. 入力rVinが設定座標の範囲外の場合は両端の値に固定されます。

   rVin < X1 の場合はrVout = Y1

   rVin > Xn の場合はrVout = Yn 但しnは最終有効値

3. 同じX座標に複数の定義がある場合は最終の値を有効とします。

   (X1, Y1)=(m,n1), (X2, Y2)=(m, n2) で rVin=m であればrVout=n2 となります。

4. X座標あるいはY座標値がNaNの場合はその定義は無視されます。

5. 有効な座標定義がない場合はrVout = NaN となります。

DdcCalorie [FB]

熱量演算

(INPUT)

記号	パラメータ	型	説明
xEnable	Enable	BOOL	FALSE: 演算スキップ, TRUE: 演算 Default (TRUE)
rF	流量(風量)入力	REAL	Default (NaN)
xHC	冷暖切換入力	BOOL	FALSE: 暖房, TRUE: 冷房 Default (FALSE)
rPC	冷房時カロリー演算定数	REAL	Default (NaN)
rPH	暖房時カロリー演算定数	REAL	Default (NaN)
rTR	還水(還気)温度入力	REAL	Default (NaN)
rTS	往水(給気)温度入力	REAL	Default (NaN)

(OUTPUT)

記号	パラメータ	型	説明
xDone	PROCESSING STATUS	BOOL	完了でTRUE
xError	ERROR STATUS	BOOL	エラー検出でTRUE
eError	ERROR CODE	DDC ERROR Enm	エラーコード 0:No Error
rVout	出力	REAL	演算結果、またはNaN xEnable = FALSE 時は NaN

解 説

往水温度・還水温度・流量または給気温度・還気温度・風量

より熱量を演算し出力します。

冷暖切換入力xHCの値により下記の演算を行います。

xHC=FALSE (暖房)

結果 = (rTS – rTR) * rF * rPH

xHC=TRUE (冷房)

結果 = (rTR – rTS) * rF * rPC

補 足

1. 演算に必要な入力にNaN値が含まれる場合は結果をNaN値で返します。

DdcCore [FB]

システム基本処理(システム関数)

(INPUT)

記号	パラメータ	型	説明
xEnable	Enable	BOOL	FALSE:演算スキップ, TRUE:演算 Default (TRUE)
xIO_OutEn	出力制御	BOOL	FALSE: Disable, TRUE: Enable Default (TRUE)

(OUTPUT)

記号	パラメータ	型	説明
xDone	PROCESSING STATUS	BOOL	完了でTRUE
xError	ERROR STATUS	BOOL	エラー検出でTRUE
eError	ERROR CODE	DDC ERROR Enm	エラーコード 0:No Error
xHW_Rdy	H/W Ready	BOOL	FALSE: Not ready, TRUE: Ready
xHW_Err	H/W Config Error	BOOL	FALSE: No error, TRUE: Error
xTRG_100ms	Trigger 100ms	BOOL	100ms 毎 1scan on

記号	パラメータ	型	説明
xTRG_500ms	Trigger 500ms	BOOL	500ms 毎1scan on
xTRG_1s	Triggrt 1sec	BOOL	1sec 毎1scan on
xPULSE_100ms	Pulse 100ms	BOOL	50ms off / 50ms on
xPULSE_500ms	Pulse 500ms	BOOL	250ms off / 250ms on
xPULSE_1s	Pulse 1sec	BOOL	500ms off / 500ms on

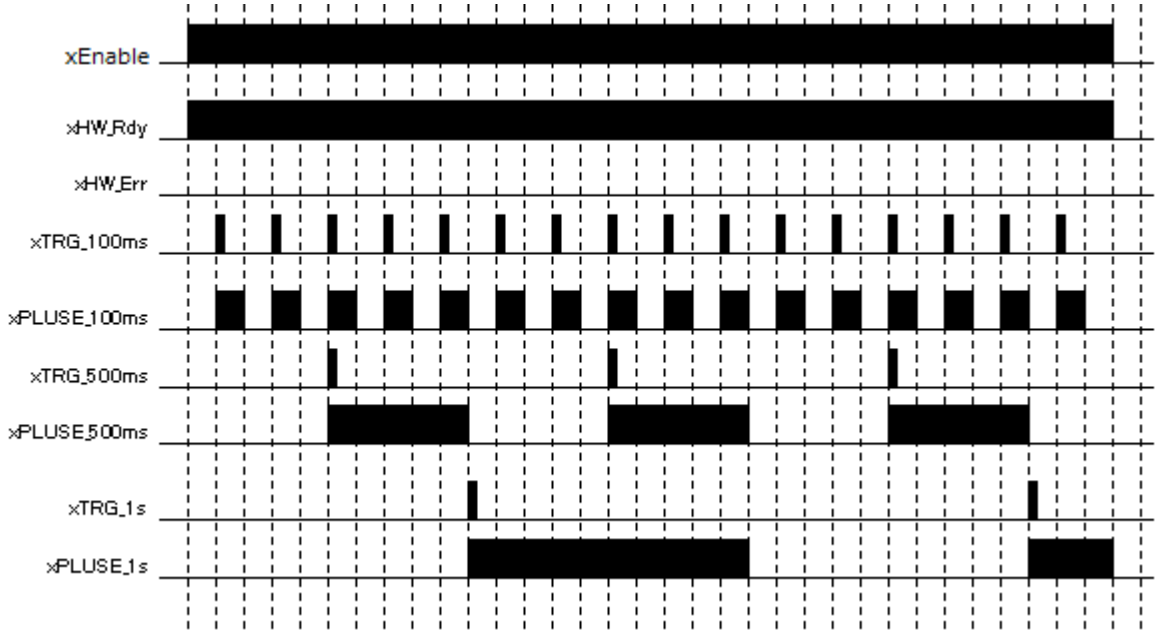
解 説

システムチェック、時間管理を行い結果を返します。

このファンクションブロックはシステムで予約されテンプレートの利用でインスタンスが作成されるのでユーザが明示的にインスタンスを作成する必要はありません。

この関数はシステム関数です。通常テンプレートで用意されたPOU [PLC_PRG] 内で使用されます。

xTRG_***は次の呼び出しでFALSEになりますのでPOU [PLC_DEFAULT_PRG] のみで利用できます。



DdcCycTimer [FB]

サイクリックタイマ

(INPUT)

記号	パラメータ	型	説明
xEnable	Enable	BOOL	FALSE: 演算スキップ, TRUE: 演算 Default (TRUE)

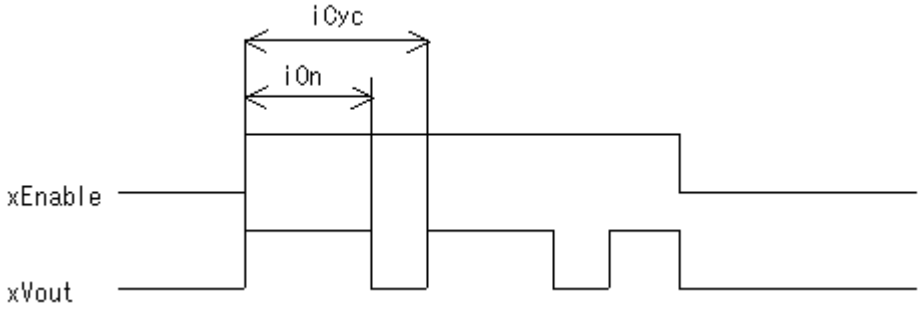
記号	パラメータ	型	説明
iCyc	サイクル時間	INT	[sec] 0 ～ 9999 上下限で丸め Default (0)
iOn	ON時間	INT	[sec] 0 ～ 9999 上下限で丸め Default (0)

(OUTPUT)

記号	パラメータ	型	説明
xDone	PROCESSING STATUS	BOOL	完了でTRUE
xError	ERROR STATUS	BOOL	エラー検出でTRUE
eError	ERROR CODE	DDC ERROR Enm	エラーコード 0:No Error
xVout	出力	BOOL	xEnable=FALSE 時は FALSE

解 説

iCycの周期毎にiOn時間だけ出力xVoutをON(TRUE)にします。



DdcDualDelayTimer [FB]

デュアルディレイタイマ

(INPUT)

記号	パラメータ	型	説明
xEnable	Enable	BOOL	FALSE: 演算スキップ, TRUE: 演算 Default (TRUE)
xAct	ディレイ動作選択	BOOL	TRUE: ディレイ動作有効 FALSE: ディレイ動作無効 (xVout = xVin) Default (FALSE)
xVin	入力値	BOOL	Default (FALSE)
iOnDly	ONディレイ時間	INT	[sec] 0 ～ 9999 上下限で丸め (0: ONディレイ無効)

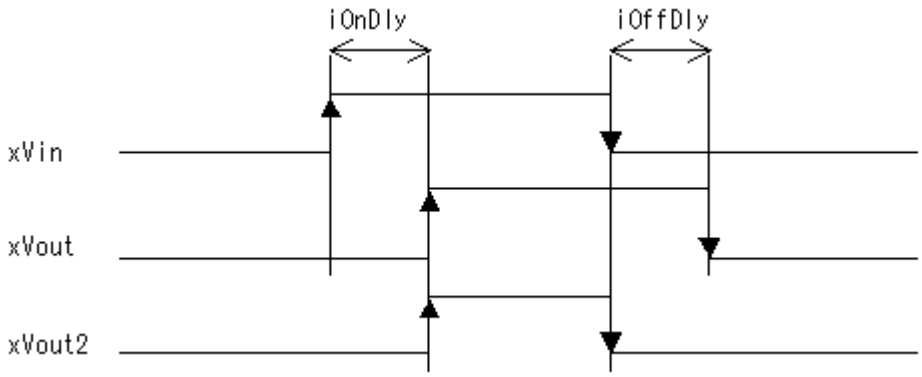
記号	パラメータ	型	説明
			Default (0)
iOffDly	OFFディレイ時間	INT	[sec] 0 ～ 9999 上下限で丸め (0: OFFディレイ無効) Default (0)

(OUTPUT)

記号	パラメータ	型	説明
xDone	PROCESSING STATUS	BOOL	完了はTRUE，処理中はFALSE
xError	ERROR STATUS	BOOL	成功はFALSE，失敗はTRUE
eError	ERROR CODE	DDC ERROR Enm	エラーコード 0:No Error
xVout	出力	BOOL	xEnable=FALSE 時はFALSE
xVout2	出力2	BOOL	xEnable=FALSE 時は FALSE

解 説

入力iVinに対して設定したＯＮディレイまたはＯＦＦディレイした結果をxVoutに出力します。



補 足

1. 初回(xEnable=FALSE→TRUEまたはxEnable=TRUEかつxAct=FALSE→TRUE)で既にxVin=TRUEであった場合はFALSE→TRUEに変化したものとみなします。
2. iOnDly中にxVinがTRUE→FALSEに変化したらxVout=FALSEとします。また、iOffDly中にxVinがFALSE→TRUEに変化した場合はxVout=TRUEとします。
3. xEnableがFALSEである場合はxVout,xVout2はFALSEとなります。

DdcEnthalpy [FB]

エンタルピ演算

(INPUT)

記号	パラメータ	型	説明
xEnable	Enable	BOOL	FALSE: 演算スキップ, TRUE: 演算 Default (TRUE)
rDB	乾球温度	REAL	[℃] NaN, -20.00 ～ +99.00 上下限で丸め Default (NaN)
rRH	相対湿度	REAL	[%] NaN, 0 ～ 100 上下限で丸め Default (NaN)
rDPT	露点温度	REAL	[CDP] NaN, -20.00 ～ +99.00 上下限で丸め Default (NaN)

(OUTPUT)

記号	パラメータ	型	説明
xDone	PROCESSING STATUS	BOOL	完了でTRUE
xError	ERROR STATUS	BOOL	エラー検出でTRUE
eError	ERROR CODE	DDC ERROR Enm	エラーコード 0:No Error
rVout	計算結果（エンタルピ）	REAL	[kcal/kg] xEnable=FALSE 時は NaN

(STATE)

rDB	rRH	rDPT	rVout
NaN	-	-	NaN
-	NaN	-	rDB, rDPTで演算
-	-	NaN	rDB, rRHで演算
-	-	-	rDB, rRHで演算 rDPTは未使用

NaNには範囲外も含みます

解 説

乾球温度／相対湿度、又は乾球温度／露点温度からエンタルピを算出します。

ENTHALPYの算出は

- 乾球温度(rDB)、相対湿度(rRH)
- 乾球温度(rDB)、露点温度(rDPT)

パラメータ指定により上記２式のうちどちらかを使用します。

DdcFilter [FB]

一次遅れフィルタ

(INPUT)

記号	パラメータ	型	説明
xEnable	Enable	BOOL	FALSE: 演算スキップ, TRUE: 演算 Default (TRUE)
rVin	入力値	REAL	Default (NaN)
iT1	遅れ時間	INT	[sec] 0 ～ 100 上下限で丸め Default (0)
xRST	リセット	BOOL	TRUE: rVout = rVin, FALSE: rVout = 計算結果 Default (FALSE)

(OUTPUT)

記号	パラメータ	型	説明
xDone	PROCESSING STATUS	BOOL	完了でTRUE
xError	ERROR STATUS	BOOL	エラー検出でTRUE
eError	ERROR CODE	DDC ERROR Enm	エラーコード 0:No Error
rVout	出力	REAL	演算結果、またはNaN xEnable=FALSE 時は rVin

解 説

下記の計算式で結果を返します。

rVout = voLast + TS / (TS + iT1) * (rVin – voLast)

voLast：前回出力値

TS：実行周期

補 足

1. 初回の実行結果はrVout = voLast = rVin とします。
2. rVin がNaNであると結果rVoutはNaNとなります。
3. 実行周期(TS)＞遅れ時間(iT1)であるときはiT1=TSとして計算します。

DdcR_Compare / DdcF_Compare [FB]

比較

(INPUT)

記号	パラメータ	型	説明
xEnable	Enable	BOOL	FALSE: 演算スキップ, TRUE: 演算 Default (TRUE)
rVin	入力値	REAL	Default (0.0)
rRef	比較設定値	REAL	Default (0.0)
rHys	ヒステリシス	REAL	NaN, 値 >= 0 下限以下は下限値 Default (0.0)

(OUTPUT)

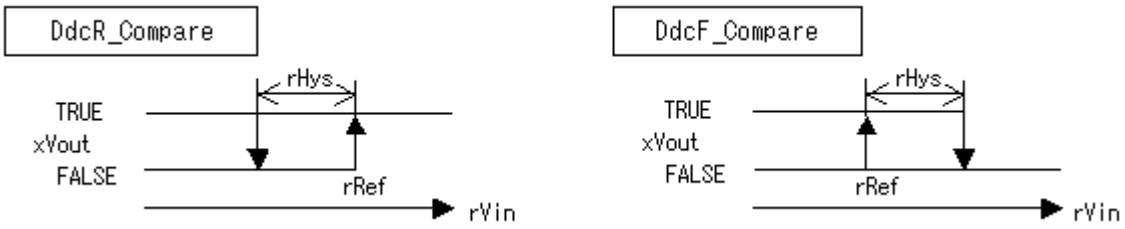
記号	パラメータ	型	説明
xDone	PROCESSING STATUS	BOOL	完了でTRUE
xError	ERROR STATUS	BOOL	エラー検出でTRUE
eError	ERROR CODE	DDC_ERROR_Enm	エラーコード 0:No Error
xVout	出力	BOOL	xEnable=FALSE 時は FALSE

解 説

入力を比較した結果を返します。各関数は下記の演算結果を返します。

DdcR_CompareはrVin >= rRefの時点からrVin < (rRef – rHys)までの間xVout = TRUE

DdcF_CompareはrVin <= rRefの時点からrVin > (rRef + rHys)までの間xVout = TRUE



DdcLoadReset [FB]

給気温度最適化制御

(INPUT)

記号	パラメータ	型	説明
xEnable	Enable	BOOL	FALSE: 演算スキップ, TRUE: 演算 Default (TRUE)
xMX	UP/DOWNコマンド	BOOL	Default (FALSE)
xMN	DOWN/UPコマンド	BOOL	Default (FALSE)
xHC	冷暖切換入力	BOOL	FALSE: 暖房, TRUE: 冷房 Default (FALSE)
xRST	リセット	BOOL	TRUE: リセット Default (FALSE)
rVin	初期値設定	REAL	[%] NaN, -999.9 ～ 999.9 上下限で丸め Default (NaN)
iCYC	実行周期	INT	[s] 0 ～ 9999 上下限で丸め Default (600)
rSTP	増加ステップ幅	REAL	[%] NaN, -999.9 ～ 999.9 上下限で丸め Default (1.0)
rMAX	出力最大値	REAL	[%] NaN, -999.9 ～ 999.9 上下限で丸め Default (100.0)
rMIN	出力最小値	REAL	[%] NaN, -999.9 ～ 999.9 上下限で丸め Default (-100.0)

(OUTPUT)

記号	パラメータ	型	説明
xDone	PROCESSING STATUS	BOOL	完了でTRUE
xError	ERROR STATUS	BOOL	エラー検出でTRUE
eError	ERROR CODE	DDC_ERROR_Enm	エラーコード 0:No Error
rVout	出力	REAL	NaN, -999.9 ～ 999.9 xEnable=FALSE 時は NaN

解 説

入力(xMX、xMN)により一定周期(iCyc)毎に設定値を一定値(rStp)だけUP/DOWNさせ出力します。

1. 実行周期(iCYC)毎にUP/DOWNコマンド(xMX,xMN)の指令により出力を指定のステップ幅(rSTP)だけ増減します。

   ●暖房時 (xHC = FALSE)

xMX	xMN	出力
TRUE	FALSE	iCYC毎にrSTP分だけ出力を増加する。 出力 = 前回出力 + ステップ幅(rSTP)

xMX	xMN	出力
FALSE	TRUE	iCYC毎にrSTP分だけ出力を減少する。 出力 = 前回出力 - ステップ幅(rSTP)
TRUE	TRUE	前回出力を保持
FALSE	FALSE	〃

●冷房時 (xHC = TRUE)

xMX	xMN	出力
TRUE	FALSE	iCYC毎にrSTP分だけ出力を減少する。 出力 = 前回出力 - ステップ幅(rSTP)
FALSE	TRUE	iCYC毎にrSTP分だけ出力を増加する。 出力 = 前回出力 + ステップ幅(rSTP)
TRUE	TRUE	前回出力を保持
FALSE	FALSE	〃

2. セット(xRST = TRUE)の時

   初期値設定(rVin)をそのまま出力します。

3. パワーオンリセット時

   初期値設定(rVin)をそのまま出力します。

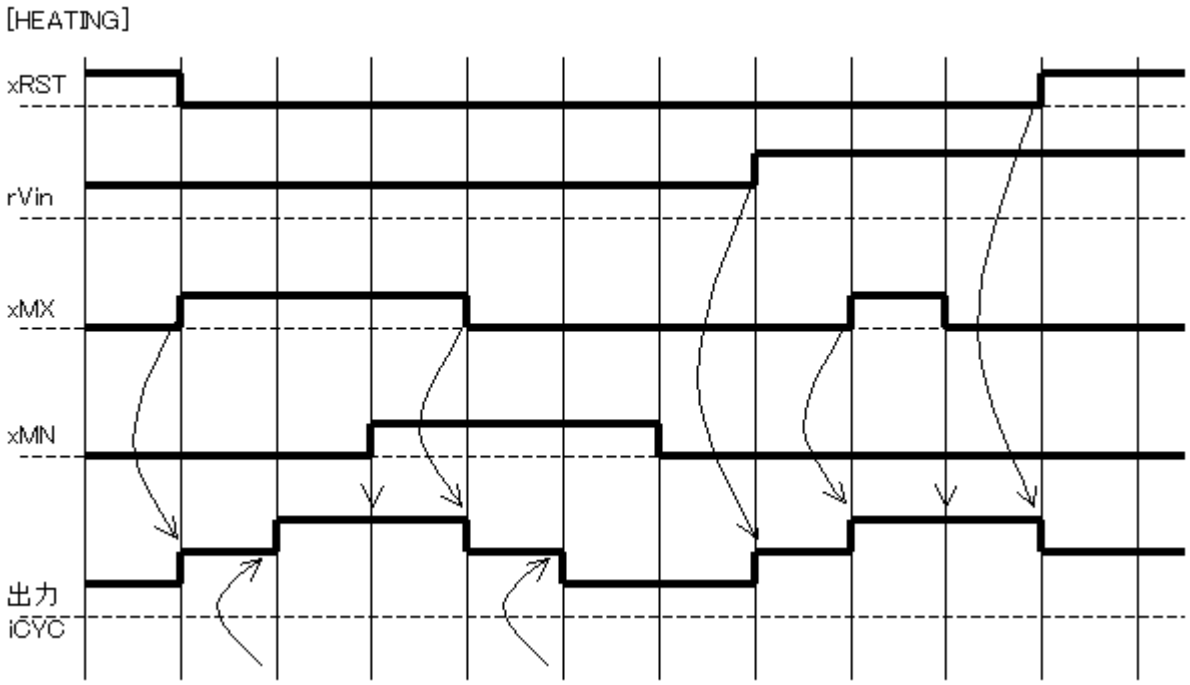
4. 初期値設定(rVin)の変化時

   初期値設定(rVin)をそのまま出力します。

5. 出力は最大値(rMAX)、最小値(rMIN)により値が制限されます。

6. rVin,rSTP,rMAX,rMINの何れかがNaNなら出力(rVout)にNaNが出力されます。

7. 上記2,3,4,6は実行周期に関係なく評価されます。また実行周期の計測はその時点から再計算(リセット)されます。



DdcLoopSingle [FB]

PID演算

(INPUT)

記号	パラメータ	型	説明
xEnable	Enable	BOOL	FALSE: 演算スキップ, TRUE: 演算 Default (TRUE)
rPV	プロセス入力	REAL	[℃] NaN, -100.00 ～ +100.00 上下限で丸め Default (NaN)
rSP	設定入力	REAL	[℃] NaN, -100.00 ～ +100.00 上下限で丸め Default (NaN)
rTR	トラッキング入力	REAL	[%] NaN:Disable, 0 ～ 100.0 上下限で丸め Default (NaN)
xST	インタロック ロックアウト入力	BOOL	TRUE: ロックアウト状態 (rVout=比例制御演算), FALSE:rVout=演算出力 Default (FALSE)
rPB	比例帯	REAL	[%] NaN, 0 ～ 999.9 上下限で丸め (0 または NaN:Disable) Default (5)
iTI	積分時間	INT	[sec] 0 ～ 9999 上下限で丸め (0:Disable, 動作は位置比例制御となる) Default (900)

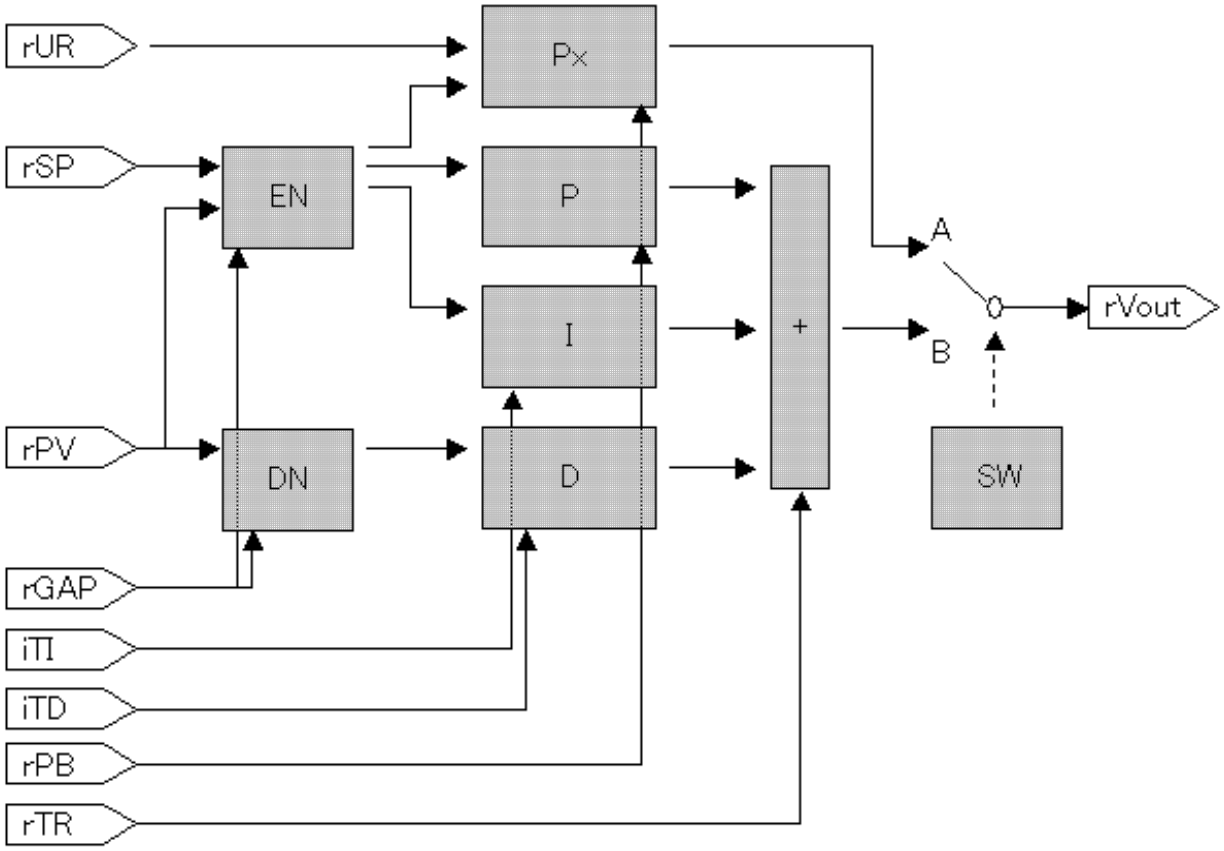
記号	パラメータ	型	説明
iTD	微分時間	INT	[sec] 0 ～ 9999 上下限で丸め (0:Disable) Default (0)
rUR	リセット値	REAL	[%] NaN, 0 ～ 100.0 上下限で丸め Default (50)
rGAP	偏差ギャップ	REAL	[%] NaN, 0 ～ 999.9 上下限で丸め Default (NaN)
xDIR	正逆動作選択	BOOL	FALSE:逆, TRUE:正 Default (FALSE)
iSKP	演算スキップ回数	INT	[回] 0 ～ 9999 上下限で丸め Default (0)

(OUTPUT)

記号	パラメータ	型	説明
xDone	PROCESSING STATUS	BOOL	完了でTRUE
xError	ERROR STATUS	BOOL	エラー検出でTRUE
eError	ERROR CODE	DDC_ERROR_Enm	エラーコード 0:No Error
rVout	計算結果	REAL	[%] 0 ～ 100.0 xEnable=FALSE 時は 0.0

解 説

プロセス入力(rPV)と設定入力(rSP)との偏差によりPID演算を行います。



EN, DN: 偏差, Px:位置比例, P: 比例項, I: 積分項, D: 微分項, SW: 条件

■演算エラー時のxError

以下の条件でxError=TRUE、rVout=0.0に設定されます。

1. rPV,rSPがNaNの場合
2. 比例帯(rPB)が0.0%の場合
3. 演算結果rVoutが±999.9%を超えた場合

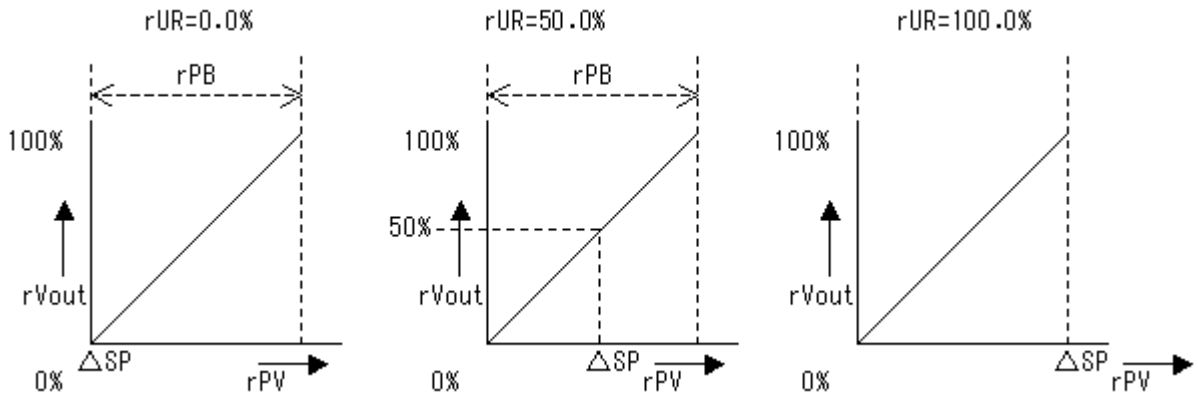
■条件SW

以下のいづれかの条件でA(位置比例)を選択します。

1. イニシャル時(初回演算またはrPV,rSPの前回値がNaNである時)
2. 積分時間iTIが0(ゼロ)
3. インターロック入力xSTがTRUE

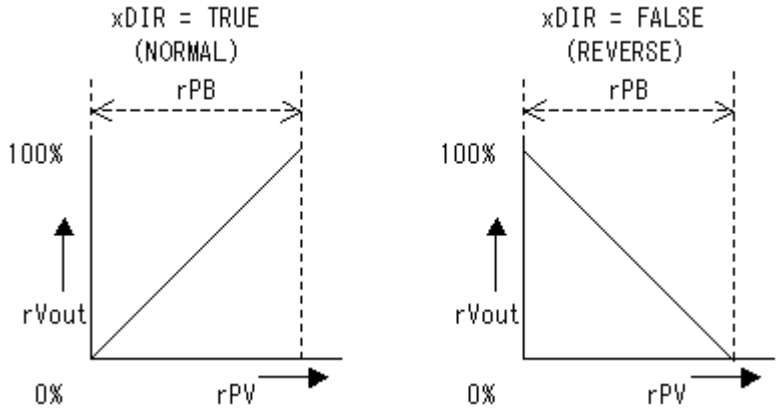
■リセット値rURについて

位置比例制御を行うときの出力オフセットを指定します。



■正逆動作選択xDIRについて

PIDおよび位置比例制御の動作方向を指定します。



DdcMomentaryOutput [FB]

モメンタリ出力

(INPUT)

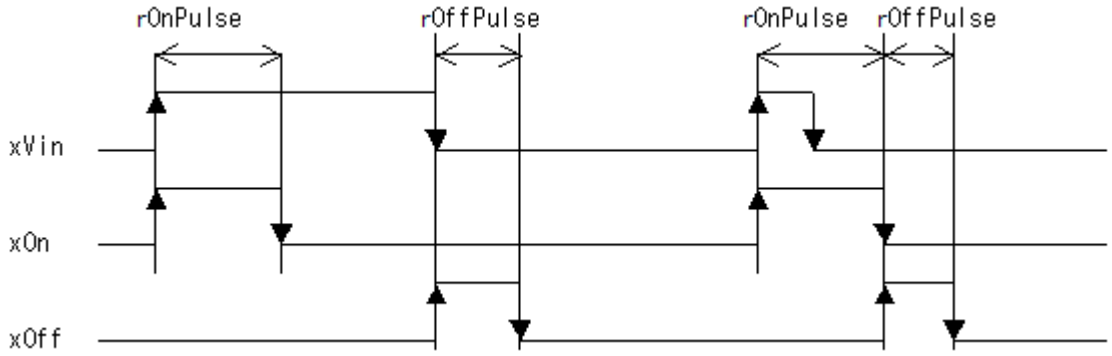
記号	パラメータ	型	説明
xEnable	ENABLE	BOOL	FALSE: 演算スキップ, TRUE: 演算 Default (TRUE)
xVin	入力	BOOL	Default (FALSE)
rOnPulse	On パルス時間	REAL	[sec] NaN, 0.1 ～ 10.0 上下限で丸め (NaN:no start pulse) Default (NaN)
rOffPulse	Off パルス時間	REAL	[sec] NaN, 0.1 ～ 10.0 上下限で丸め (NaN:no stop pulse) Default (NaN)

(OUTPUT)

記号	パラメータ	型	説明
xDone	PROCESSING STATUS	BOOL	完了でTRUE
xError	ERROR STATUS	BOOL	エラー検出でTRUE
eError	ERROR CODE	DDC ERROR Enm	エラーコード 0:No Error
xOn	Start 出力	BOOL	xEnable=FALSE 時は FALSE
xOff	Stop 出力	BOOL	xEnable=FALSE 時は FALSE

解 説

入力をモメンタリ出力に変換します。



補 足

1. 初回(xEnable=FALSE→TRUE)で既にxVin=TRUEであった場合はFALSE→TRUEに変化したものとみなします。
2. rOnPulse中にxVinがTRUE→FALSEあるいはrOffPulse中にxVinがFALSE→TRUEに変化してもPulse出力は継続します。このときrOnPulse出力完了時点のxVinがFALSEであれば続いてrOffPulseが出力されます。また、rOffPulse出力完了時点のxVinがTRUEであれば続けてrOnPulseが出力されます。
3. xEnableがFALSEである場合はxOn,xOffを即FALSEにします。

DdcMvLimit [FB]

変化量制限

(INPUT)

記号	パラメータ	型	説明
xEnable	Enable	BOOL	FALSE: 演算スキップ, TRUE: 演算 Default (TRUE)
rVin	入力値	REAL	Default (NaN)
rCR	変化量	REAL	NaN, 0.0 ～ 99999 上下限で丸め

記号	パラメータ	型	説明
			Default (NaN)

(OUTPUT)

記号	パラメータ	型	説明
xDone	PROCESSING STATUS	BOOL	完了でTRUE
xError	ERROR STATUS	BOOL	エラー検出でTRUE
eError	ERROR CODE	DDC ERROR Enm	エラーコード 0:No Error
rVout	結果	REAL	演算結果、あるいはNaN xEnable=FALSE 時は rVin

解 説

入力rVinの変化に対して変化量制限を行いrVoutに出力します。

入力rVinの変化量(前回出力値voLast – 入力rVin)が±rCR以上の場合は次の示す値を出力します。

上限値 : voLast + rCR

下限値 : voLast - rCR

voLast : 前回出力値

補 足

1. 初回(xEnableがFALSE→TRUE)の実行結果はrVout = voLast = rVin とします。
2. rVinかrCRがNaNであると結果rVoutはNaNとなります。

DdcPointHistory [FB]

変数値の履歴書き込み

(INPUT)

記号	パラメータ	型	説明
xEnable	Enable	BOOL	FALSE: 演算スキップ, TRUE: 演算 Default (TRUE)
iUnit	履歴領域番号	INT	1 ～ 最大履歴UNIT数 Default (0)
rVin	対象の値(REAL)	REAL	Default (NaN)
usiVin	対象の値(USINT)	USINT	Default (0)

記号	パラメータ	型	説明
xVin1	対象の値(BOOL)	BOOL	Default (FALSE)
xVin2	対象の値(BOOL)	BOOL	Default (FALSE)
xVin3	対象の値(BOOL)	BOOL	Default (FALSE)
xVin4	対象の値(BOOL)	BOOL	Default (FALSE)
xVin5	対象の値(BOOL)	BOOL	Default (FALSE)
xVin6	対象の値(BOOL)	BOOL	Default (FALSE)
xVin7	対象の値(BOOL)	BOOL	Default (FALSE)
xVin8	対象の値(BOOL)	BOOL	Default (FALSE)
xRing	Enable RingBuffer	BOOL	Default (FALSE) TRUE:Ring Buffer
xReset	リセット(履歴削除)	BOOL	立ち上がりでリセット Default (FALSE)
iSampling	サンプリング間隔(秒)	INT	0 ～ 1000 (0:Program Scan) Default (0)
sTitle	履歴タイトル	STRING	max. 32 characters Default ('')

(OUTPUT)

記号	パラメータ	型	説明
xDone	PROCESSING STATUS	BOOL	完了でTRUE
xBusy	PROCESSING STATUS	BOOL	処理中はTRUE
xError	ERROR STATUS	BOOL	エラー検出でTRUE
eError	ERROR CODE	DDC_ERROR_Enm	エラーコード 0:No Error
iCount	履歴現在件数	INT	0 ～ 最大履歴レコード数 最大履歴レコード数は設定に依存

解 説

指定の値をコントローラ内の履歴領域に記録します。

最大履歴UNIT数と最大履歴レコード数は設定で決定されます。

初期値は、最大履歴UNIT数(50)で各UNITの最大履歴レコード数(100)です。

■ xRing 指定による xDone, xBusy 状態

xRing	エラー時	iCount < 履歴最大件数	iCount = 履歴最大件数
FALSE	xDone = FALSE	xDone = FALSE	xDone = TRUE

xRing	エラー時	iCount < 履歴最大件数	iCount = 履歴最大件数
	xBusy = FALSE	xBusy = TRUE 記録中	xBusy = FALSE 記録継続
TRUE	xDone = FALSE xBusy = FALSE	xDone = FALSE xBusy = TRUE 記録中	xDone = FALSE xBusy = TRUE 記録停止

DdcPulseCounter [FB]

パルスカウント

(INPUT)

記号	パラメータ	型	説明
xEnable	ENABLE	BOOL	FALSE: 演算スキップ, TRUE: 演算 Drfault (TRUE)
xVin	パルス入力	BOOL	Default (FALSE)
xReset	カウンタリセット	BOOL	Default (FALSE)
rPulse	パルスカウント定数	REAL	Default (NaN)

(OUTPUT)

記号	パラメータ	型	説明
xDone	PROCESSING STATUS	BOOL	完了でTRUE
xError	ERROR STATUS	BOOL	エラー検出でTRUE
eError	ERROR CODE	DDC ERROR Enm	エラーコード 0:No Error
rVout	レート出力	REAL	[Pulse / sec] xEnable=FALSE 時は 0
udiCount	カウント出力	UDINT	0 ～ 999,999,999 上限を超えると0に戻る xEnable=FALSE 時は 0

解 説

パルスを入力し演算結果をrVoutに出力します。

rVout = rPulse * (1 / (t1 – t2))

t1: 今回パルスFALSE→TRUE時刻[sec]

t2: 前回パルスFALSE→TRUE時刻[sec]

補 足

- パルスは 50% duty サイクルであるものとして演算します。
- 最大パルスレートは (このファンクションブロックの置かれたプログラムのスキャン周期に依存します)

  例えば100msスキャン周期であれば
  1sec / (100ms * 1/(50%)) = 5 [pulse / sec]
  が最大パルスレートとなります。
- 最小パルスレートは (このファンクションブロックの置かれたプログラムのスキャン周期に依存します)

  1pulse / 5min (300sec) です。
  この最小パルスを超える場合はvolに 0 が出力されます。
- xResetはTRUEの間カウント出力を0に設定します。
- カウント出力(udiCount)は、パルス入力のFALSE→TRUEの検出で1加算されます。

DdcRtcNow [FB]

現在日付時刻の読み出し

(INPUT)

記号	パラメータ	型	説明
xEnable	Enable	BOOL	FALSE: 演算スキップ, TRUE: 演算 Default (TRUE)

(OUTPUT)

記号	パラメータ	型	説明
xDone	PROCESSING STATUS	BOOL	完了でTRUE
xError	ERROR STATUS	BOOL	エラー検出でTRUE
eError	ERROR CODE	DDC ERROR Enm	エラーコード 0:No Error
iYear	年	INT	0, 1900 ～ 2070 (*1)
iMonth	月	INT	0, 1 ～ 12 (*1)
iDay	日	INT	0, 1 ～ 31 (*1)
iHour	時	INT	0 ～ 23 (*1)
iMinute	分	INT	0 ～ 59 (*1)
iSecond	秒	INT	0 ～ 59 (*1)
iDayOfWeek	曜日	INT	0 ～ 6 (0:Sunday, 6:Saturday) (*1)
iDayOfYear	年間積算日	INT	0, 1 ～ 366 (*1)

(*1) xEnable=FALSE 時は 0 を返します。

解 説

リアルタイムクロックより現在時刻を取得します。

DdcWeightedAverage [FB]

加重平均

(INPUT)

記号	パラメータ	型	説明
xEnable	Enable	BOOL	FALSE: 演算スキップ, TRUE: 演算 Default (TRUE)
rV1 .. 4	入力1～4	REAL	Default (NaN)
rW1 .. 4	加重1～4	REAL	Default (NaN)

(OUTPUT)

記号	パラメータ	型	説明
xDone	PROCESSING STATUS	BOOL	完了でTRUE
xError	ERROR STATUS	BOOL	エラー検出でTRUE
eError	ERROR CODE	DDC ERROR Enm	エラーコード 0:No Error
rVout	結果	REAL	演算結果、またはNaN xEnable=FALSE 時は NaN

解 説

入力rV1～4の加重平均演算を行います。入力rV1に対する加重はrW1に入力します。対応する入力あるいは加重がNaN値の場合は演算対象から除かれます。

$$rVout = \frac{\sum_{n=1}^{4} rVn \times rWn}{\sum_{n=1}^{4} rWn}$$

補 足

1. 演算対象が無い場合は結果としてNaN値を返します。
2. 加重合計が0の場合は結果として0を返します。

DdcSetLRealNaN [FUN]

LREAL(64bit)型変数値をNaNに設定

(INPUT)

記号	パラメータ	型	説明
なし			

(OUTPUT)

記号	パラメータ	型	説明
(RETURN)	結果	LREAL	NaN値

解 説

指定LREAL(64bit)型変数をNaN値に設定します。

DdcSetRealNaN [FUN]

REAL(32bit)型変数値をNaNに設定

(INPUT)

記号	パラメータ	型	説明
なし			

(OUTPUT)

記号	パラメータ	型	説明
(RETURN)	結果	REAL	NaN値

解 説

指定REAL(32bit)型変数をNaN値に設定します。

Ddc_IsLRealNaN [FUN]

LREAL(64bit)型変数値がNaNであるか判定

(INPUT)

記号	パラメータ	型	説明
lrVin	判定対象変数	LREAL	

(OUTPUT)

記号	パラメータ	型	説明
(RETURN)	判定結果	BOOL	FALSE (Not NaN), TRUE (NaN)

解 説

指定LREAL(64bit)型変数がNaN値であるか判定し結果を返します。

Ddc_IsRealNaN [FUN]

REAL(32bit)型変数値がNaNであるか判定

(INPUT)

記号	パラメータ	型	説明
rVin	判定対象変数	REAL	

(OUTPUT)

記号	パラメータ	型	説明
(RETURN)	判定結果	BOOL	FALSE (Not NaN), TRUE (NaN)

解 説

指定REAL(32bit)型変数がNaN値であるか判定し結果を返します。

DEFINE関連

MsysDefine POUs

OP(Operand), FUN(Function), FB(Function Block), DUT(Data Unit Type)

名称	属性	機能	サポート Library *1
MSYS_ByteOrder_Enm	DUT	バイト順 列挙型	
MSYS_ERROR_Enm	DUT	エラーコード 列挙型	

*1) サポートLibrary欄は、そのファンクションあるいはファンクションブロックのサポートを開始したライブラリのバージョンを記述しています。この欄が空の場合はv1.0.0以降でサポートしていることを示します。

MSYS_ByteOrder_Enm [DUT]

バイト順 列挙型

識別子	値	説明
NONE	0	NONE
INT8	1	Integer [Byte order: 0]
INT16	2	Integer [Byte order: 0,1]
INT32	3	Integer [Byte order: 0,1,2,3]
UINT8	4	Unsigned Integer [Byte order: 0]
UINT16	5	Unsigned Integer [Byte order: 0,1]
UINT32	6	Unsigned Integer [Byte order: 0,1,2,3]
FLOAT32	7	IEEE Float [Byte order: 0,1,2,3]
FLOAT64	8	IEEE Float [Byte order: 0,1,2,3,4,5,6,7]
R_INT16	9	Integer [Byte order: 1,0]
R_INT32	10	Integer [Byte order: 3,2,1,0]
R_UINT16	11	Unsigned Integer [Byte order: 1,0]
R_UINT32	12	Unsigned Integer [Byte order: 3,2,1,0]
R_FLOAT32	13	IEEE Float [Byte order: 3,2,1,0]
R_FLOAT64	14	IEEE Float [Byte order: 7,6,5,4,3,2,1,0]

識別子	値	説明
INT32WS	15	Integer [Byte order: 2,3,0,1]
UINT32WS	16	Unsigned Integer [Byte order: 2,3,0,1]
FLOAT32WS	17	IEEE Float [Byte order: 2,3,0,1]
FLOAT64WS	18	IEEE Float [Byte order: 6,7,4,5,2,3,0,1]
R_INT32WS	19	Integer [Byte order: 1,0,3,2]
R_UINT32WS	20	Unsigned Integer [Byte order: 1,0,3,2]
R_FLOAT32WS	21	IEEE Float [Byte order: 1,0,3,2]
R_FLOAT64WS	22	IEEE Float [Byte order: 1,0,3,2,5,4,7,6]

識別子の型とバイト順はLITTLE-ENDIANを基準としています。

もしコントローラのCPUがLITTLE-ENDIANと異なる場合でも指定は同じです。

対象データがLITTLE-ENDIANのバイト順(L,H)であればINT16のように指定します。

また、対象データがBIG-ENDIANのバイト順(H,L)であればR_INT16のように指定します。

MSYS_ERROR_Enm [DUT]

エラーコード 列挙型

識別子	値	説明
NO_ERROR	0	正常（エラーなし）
SYS_STRUCT_SIZE	1	System
MATH_DivByZero	11	演算で0割が発生
PARA_ARG	100	入力パラメータが範囲外
PARA_IsNaN	101	必須入力パラメータが設定されていないか NaN値が設定されている
PARA_IsNullPointer	102	必須入力パラメータが設定されていないか Nullポインタが設定されている
PARA_ByteOrder_Range	114	バイト順指定が範囲外

R3入出力カード関連

MsysR3Standard POUs

OP(Operand), FUN(Function), FB(Function Block), DUT(Data Unit Type)

名称	属性	機能	サポート Library *1
R3_CARD_INFO_Typ	DUT	R3 カード情報 構造体	
R3GetCardInfo	FB	R3 カード情報取得	
R3GetBit	FB	R3 デジタル入力読み込み	
R3Get16	FB	R3 アナログ(16bit)入力読み込み	
R3Get32	FB	R3 アナログ(32bit)入力読み込み	
R3GetREAL	FB	R3 アナログ(REAL)入力読み込み	
R3GetLREAL	FB	R3 アナログ(LREAL)入力読み込み	
R3ReadbackBit	FB	R3 デジタル出力読み込み	
R3Readback16	FB	R3 アナログ(16bit)出力読み込み	
R3Readback32	FB	R3 アナログ(32bit)出力読み込み	
R3ReadbackREAL	FB	R3 アナログ(REAL)出力読み込み	
R3ReadbackLREAL	FB	R3 アナログ(LREAL)出力読み込み	
R3SetBit	FB	R3 デジタル出力書き込み	
R3Set16	FB	R3 アナログ(16bit)出力書き込み	
R3Set32	FB	R3 アナログ(32bit)出力書き込み	
R3SetREAL	FB	R3 アナログ(REAL)出力書き込み	
R3SetLREAL	FB	R3 アナログ(LREAL)出力書き込み	

*1) サポートLibrary欄は、そのファンクションあるいはファンクションブロックのサポートを開始したライブラリのバージョンを記述しています。この欄が空の場合はv1.0.0以降でサポートしていることを示します。

R3_ERROR_Enm [DUT]

R3 エラーコード 列挙型

識別子	値	説明
NO_ERROR	0	正常(エラーなし)
SYS_STRUCT_SIZE	1	System error
MATH_DivByZero	11	演算で0割が発生
PARA_ARG	100	入力パラメータが範囲外
PARA_IsNaN	101	必須入力パラメータが設定されていないか NaN値が設定されている
PARA_IsNullPointer	102	必須入力パラメータが設定されていないか Nullポインタが設定されている
PARA_Raw_Range	110	上下限設定が範囲外
PARA_Scale_Range	111	スケール設定が範囲外
PARA_Vin_Range	112	Vin入力値が範囲外
PARA_MinMax_Range	113	上下限値が範囲外
PARA_ByteOrder_Range	114	バイト順指定が範囲外
CARD_Empty	200	指定のR3 I/Oカードスロットにカードなし
CARD_Slot	201	指定のR3 I/Oカードスロット番号は範囲外
CARD_Addr	202	指定のR3 I/Oアドレスは範囲外
CARD_TypeMismatch	203	指定のR3 I/Oカードの入出力タイプとFBが不一致
CARD_Point	204	指定のR3 I/Oアドレスは実ポイント数の範囲外
CARD_StateIsNotValid	205	R3 I/Oの該当カードにカード入出力情報が未確定
CARD_StateHasError	206	R3 I/Oの該当カードにデータエラーあるいはハードエラーが発生
CARD_ChHwError	207	R3 I/Oの該当チャネルにハードウェアエラーが発生
CARD_ChInpError	208	R3 I/Oの該当チャネルに入力データエラーが発生
CARD_ChInpNotEnabled	209	R3 I/Oの該当チャネルの入力データステータスが無効
CARD_Error	210	エラー (ErrorSubCode 参照)

R3_CARD_INFO_Typ [DUT]

R3カード情報 構造体型

メンバー	型	説明
byState	BYTE	CARD io status の state情報
bySystem	BYTE	CARD io status の system情報
byConnType	BYTE	CARD io status の conn_type
byPoint	BYTE	CARD io status の point

メンバー	型	説明
sCardName	STRING(8)	CARD io status の card_name
byIocardStatus	BYTE	CARD io status の iocard_status

R3GetCardInfo [FB]

R3カード情報取得

(INPUT)

記号	パラメータ	型	説明
xExecute	実行	BOOL	立ち上がりで実行 Default(TRUE)
iSlotNo	カードスロット番号	INT	0,1 ～ 16 (0は予約) Default (0)

(IN_OUT)

記号	パラメータ	型	説明
stDatat	カード情報格納領域	R3_CARD_INFO_Typ	Default (0)

(OUTPUT)

記号	パラメータ	型	説明
xDone	PROCESSING STATUS	BOOL	完了でTRUE
xError	ERROR STATUS	BOOL	エラー検出でTRUE
eError	ERROR CODE	R3_ERROR_Enm	エラーコード 0:No Error
uiErrorSubCode	ERROR SUB-CODE	UINT	内部エラー補助コード

解 説

ベースの指定スロットに配置されたカード情報を取得します。

R3カード情報 stDataは、機種により設定されないメンバーがあります。

機種	byState	bySystem	byConnType	byPoint	sCardName	byIocardStatus
BA3-CE10	yes	yes	yes	yes	yes	yes

R3GetBit [FB]

R3 デジタル入力読み込み

(INPUT)

記号	パラメータ	型	説明
xEnable	ENABLE	BOOL	FALSE: 演算スキップ, TRUE: 演算 Default (TRUE)
iSlotNo	カードスロット番号	INT	0,1 ～ 16 (0は予約) Default (0)
iBitNo	ビット番号	INT	0 ～ 63 Default (0)

(OUTPUT)

記号	パラメータ	型	説明
xDone	PROCESSING STATUS	BOOL	完了でTRUE
xError	ERROR STATUS	BOOL	エラー検出でTRUE
eError	ERROR CODE	R3 ERROR Enm	エラーコード 0:No Error
uiErrorSubCode	ERROR SUB-CODE	UINT	内部エラー補助コード
xVout	出力	BOOL	xEnable=FALSE か、エラー時は FALSE

解 説

ハードウェアからBIT(1bit)入力し結果をxVoutに出力します。

R3Get16 [FB]

R3 アナログ(16bit)入力読み込み

(INPUT)

記号	パラメータ	型	説明
xEnable	ENABLE	BOOL	FALSE: 演算スキップ, TRUE: 演算 Default (TRUE)
iSlotNo	カードスロット番号	INT	0,1 ～ 16 (0は予約) Default (0)
iAddrNo	アドレス番号 (byte)	INT	0 ～ 30 (*1) Default (0)

記号	パラメータ	型	説明
eRawByteOrder	ハードウェア入力のバイト順	MSYS_ByteOrder_Enm	16bit用 Default (INT16)
rBase	ハードウェア入力のベース値	REAL	Default (0.0)
rRawL	ハードウェア入力の下限値	REAL	Default (0.0) (*2)
rRawH	ハードウェア入力の上限値	REAL	Default (10000.0) (*2)
rScaleL	ハードウェア入力の下限に割り当てる値	REAL	Default (0.0) (*3)
rScaleH	ハードウェア入力の上限に割り当てる値	REAL	Default (100.0) (*3)
rOffset	内部計算結果に対してこのオフセットを加算し出力(wVout)とします。	REAL	Default (0.0)
iT1	ハードウェア入力の一次遅れフィルタの遅れ時間	INT	[sec] 0 ～ 100 上下限で丸め Default (0)

(OUTPUT)

記号	パラメータ	型	説明
xDone	PROCESSING STATUS	BOOL	完了時にTRUE (*4)
xError	ERROR STATUS	BOOL	エラー検出時にTRUE
eError	ERROR CODE	R3_ERROR_Enm	エラーコード 0:No Error
uiErrorSubCode	ERROR SUB-CODE	UINT	内部エラー補助コード
wVout	ハードウェア値出力	WORD	xEnable=FALSE か、エラー時は0 (*5)
rVout	変換後出力	REAL	xEnable=FALSE か、エラー時はNaN (*5)

(*1) 例えばチャネル1のアドレスは0、チャネル2のアドレスは2、チャネル3のアドレスは4、チャネル4のアドレスは6を指定します。

(*2) rRawL = rRawH の場合は、スケール変換を行いません。

(*3) rScaleL = rScaleH の場合は、上下限制限を行いません。

(*4) 演算が完了しエラー eMsysERR_PARA_CARDChHwError, eMsysERR_PARA_CARDChInpError, eMsysERR_PARA_CARDChInpNotEnabled のいずれかである場合は xDone = TRUE となります。それ以外のエラーの場合は xDone = FALSE, xError = TRUE となります。

(*5) xDone = FALSE, xError = TRUE の場合は wVout = 0, rVout = NaN を返します。

解 説

下記計算式の演算結果を wVout, rVout に出力します。

rVout = Limit2((Limit1(X + b) - rRawL) * a + rScaleL) + of)

X: ハードウェア入力値

a: (rScaleH – rScaleL) / (rRawH – rRawL)

b: rBase

of: rOffset

Limit1: rRawH ～ rRawL

Limit2: rScaleH ～ rScaleL

[rRawH == rRawL の場合]

rVout = Limit2(X + b + of)

[rRawH == rRawL AND rScaleH == rScaleLの場合]

rVout = X + b + of

■「指定可能 eRawByteOrder」

INT16, UINT16, R_INT16, R_UINT16

■温度データ(R3-TS, R3-RSなど)

温度データは単位が摂氏(℃)の場合、温度を10倍した値がデータとなります。
例えば23.4℃の場合には234(10進)がデータ(Raw)となります。

■R3カード(上記以外)からの入力データ

入力レンジに対して0 ～ 100% が0 ～ 10000 (10進)が対応したデータ(Raw)となります。
例えば、この入力をプログラムで%値として使用するときは
rRawL = 0.0 , rRawH = 10000.0 , rScaleL = 0.0 , rScaleH = 100.0 または
rRawL = -1500.0 , rRawH = 11500.0 , rScaleL = -15.0 , rScaleH = 115.0 と指定します。

■入力チャネル異常(R3-TS, R3-RSなど)

バーンアウトなど入力異常となったチャネルの値は、xDone=TRUE, xError=TRUE, eError=CARD_ChInpErrorとなりますがwVoutとrVoutの値が正常時と同じく更新されます。

R3Get32 [FB]

R3 アナログ(32bit)入力読み込み

(INPUT)

記号	パラメータ	型	説明
xEnable	ENABLE	BOOL	FALSE: 演算スキップ, TRUE: 演算 Default (TRUE)
iSlotNo	カードスロット番号	INT	0,1 ～ 16 (0は予約) Default (0)
iAddrNo	アドレス番号 (byte)	INT	0 ～ 28 (*1) Default (0)
eRawByteOrder	ハードウェア入力のバイト順	MSYS_ByteOrder_Enm	32bit用 Default (INT32)
lrBase	ハードウェア入力のベース値	LREAL	Default (0.0)
lrRawL	ハードウェア入力の下限値	LREAL	Default (0.0) (*2)
lrRawH	ハードウェア入力の上限値	LREAL	Default (10000.0) (*2)
lrScaleL	ハードウェア入力の下限に割り当てる値	LREAL	Default (0.0) (*3)
lrScaleH	ハードウェア入力の上限に割り当てる値	LREAL	Default (100.0) (*4)
lrOffset	内部計算結果に対してこのオフセットを加算し出力 (rVout) とします。	LREAL	Default (0.0)

(OUTPUT)

記号	パラメータ	型	説明
xDone	PROCESSING STATUS	BOOL	完了でTRUE (*4)
xError	ERROR STATUS	BOOL	エラー検出でTRUE
eError	ERROR CODE	R3_ERROR_Enm	エラーコード 0:No Error
uiErrorSubCode	ERROR SUB-CODE	UINT	内部エラー補助コード
dwVout	ハードウェア値出力	DWORD	xEnable=FALSE か、エラー時は0 (*5)
lrVout	変換後出力	LREAL	xEnable=FALSE か、エラー時はNaN (*5)

(*1) 例えばチャネル1のアドレスは0、チャネル2のアドレスは4、チャネル3のアドレスは8、チャネル4のアドレスは12を指定します。

(*2) lrRawL = lrRawH の場合は、スケール変換を行いません。

(*3) lrScaleL = lrScaleH の場合は、上下限制限を行いません。

(*4) 演算が完了しエラー eMsysERR_PARA_CARDChHwError, eMsysERR_PARA_CARDChInpError, eMsysERR_PARA_CARDChInpNotEnabled のいずれかである場合は xDone = TRUE となります。それ以外のエラーの場合は xDone = FALSE, xError = TRUE となります。
(*5) xDone = FALSE, xError = TRUE の場合は dwVout = 0,lrVout = NaN を返します。

解 説

下記の計算式で結果をdwVout,lrVoutに出力します。

lrVout = Limit2((Limit1(X + b) - lrRawL) * a + lrScaleL) + of)

X: ハードウェア入力値
a: (lrScaleH – lrScaleL) / (lrRawH – lrRawL)
b: lrBase
of: lrOffset
Limit1: lrRawH ～ lrRawL
Limit2: lrScaleH ～ lrScaleL

[lrRawH == lrRawL の場合]

lrVout = Limit2(X + b + of)

[lrRawH == lrRawL AND lrScaleH == lrScaleLの場合]

lrVout = X + b + of

■「指定可能 eRawByteOrder」

INT32, UINT32, FLOAT32, R_INT32, R_UINT32, R_FLOAT32,
INT32WS, UINT32WS, FLOAT32WS, R_INT32WS, R_UINT32WS, R_FLOAT32WS

■入力チャネル異常

バーンアウトなど入力異常となったチャネルの値は、xDone=TRUE, xError=TRUE, eError=CARD_ChInpErrorとなりますがdwVoutとlrVoutの値が正常時と同じく更新されます。

R3GetREAL [FB]

R3 アナログREAL(32bit)入力読み込み

(INPUT)

記号	パラメータ	型	範囲	説明
xEnable	ENABLE	BOOL		FALSE: 演算スキップ, TRUE: 演算 Default (TRUE)
iSlotNo	カードスロット番号	INT		0,1 ～ 16 (0は予約) Default (0)
iAddrNo	アドレス番号 (byte)	INT		0 ～ 28 (*1) Default (0)
eRawByteOrder	ハードウェア入力のバイト順	MSYS_ByteOrder_Enm		32bit用 Default (FLOAT32)

(OUTPUT)

記号	パラメータ	型	説明
xDone	PROCESSING STATUS	BOOL	完了でTRUE (*2)
xError	ERROR STATUS	BOOL	エラー検出でTRUE
eError	ERROR CODE	R3_ERROR_Enm	エラーコード 0:No Error
uiErrorSubCode	ERROR SUB-CODE	UINT	内部エラー補助コード
lrVout	ハードウェア値出力	LREAL	xEnable=FALSE か、エラー時は NaN (*3)

(*1) 例えばチャネル1のアドレスは0、チャネル2のアドレスは4、チャネル3のアドレスは8、チャネル4のアドレスは12を指定します。

(*2) 演算が完了しエラー eMsysERR_PARA_CARDChHwError, eMsysERR_PARA_CARDChInpError, eMsysERR_PARA_CARDChInpNotEnabled のいずれかである場合は xDone = TRUE となります。それ以外のエラーの場合は xDone = FALSE, xError = TRUE となります。

(*3) xDone = FALSE, xError = TRUE の場合は lrVout = NaN を返します。

解 説

ハードウェアからの取得結果をlrVoutに出力します。

■「指定可能 MSYS_ByteOrder_Enm」

FLOAT32, R_FLOAT32, FLOAT32WS, R_FLOAT32WS

■入力チャネル異常

バーンアウトなど入力異常となったチャネルの値は、xDone=TRUE, xError=TRUE, eError=CARD_ChInpErrorとなりますがlrVoutの値が正常時と同じく更新されます。

R3GetLREAL [FB]

R3 アナログLREAL(64bit)入力読み込み

(INPUT)

記号	パラメータ	型	説明
xEnable	ENABLE	BOOL	FALSE: 演算スキップ, TRUE: 演算 Default (TRUE)
iSlotNo	カードスロット番号	INT	0,1 ～ 16 (0は予約) Default (0)
iAddrNo	アドレス番号 (byte)	INT	0 ～ 24 (*1) Default (0)
eRawByteOrder	ハードウェア入力のバイト順	MSYS_ByteOrder_Enm	64bit用 Default (FLOAT64)

(OUTPUT)

記号	パラメータ	型	説明
xDone	PROCESSING STATUS	BOOL	完了でTRUE (*2)
xError	ERROR STATUS	BOOL	エラー検出でTRUE
eError	ERROR CODE	R3_ERROR_Enm	エラーコード 0:No Error
uiErrorSubCode	ERROR SUB-CODE	UINT	内部エラー補助コード
lrVout	ハードウェア値出力	LREAL	xEnable=FALSE か、エラー時は NaN (*3)

(*1) 例えばチャネル1のアドレスは0、チャネル2のアドレスは8、チャネル3のアドレスは16、チャネル4のアドレスは24を指定します。

(*2) 演算が完了しエラー eMsysERR_PARA_CARDChHwError, eMsysERR_PARA_CARDChInpError, eMsysERR_PARA_CARDChInpNotEnabled のいずれかである場合は xDone = TRUE となります。それ以外のエラーの場合は xDone = FALSE, xError = TRUE となります。

(*3) xDone = FALSE, xError = TRUE の場合は lrVout = NaN を返します。

解 説

ハードウェアからの取得結果をlrVoutに出力します。

■「指定可能 eRawByteOrder」

FLOAT64, R_FLOAT64, FLOAT64WS, R_FLOAT64WS

■入力チャネル異常

バーンアウトなど入力異常となったチャネルの値は、xDone=TRUE, xError=TRUE, eError=CARD_ChInpErrorとなりますがrVoutの値が正常時と同じく更新されます。

R3ReadbackBit [FB]

R3 デジタル出力読み込み

(INPUT)

記号	パラメータ	型	説明
xEnable	ENABLE	BOOL	FALSE: 演算スキップ, TRUE: 演算 Default (TRUE)
iSlotNo	カードスロット番号	INT	0,1 ～ 16 (0は予約) Default (0)
iBitNo	ビット番号	INT	0 ～ 63 Default (0)

(OUTPUT)

記号	パラメータ	型	説明
xDone	PROCESSING STATUS	BOOL	完了でTRUE
xError	ERROR STATUS	BOOL	エラー検出でTRUE
eError	ERROR CODE	R3 ERROR Enm	エラーコード 0:No Error
uiErrorSubCode	ERROR SUB-CODE	UINT	内部エラー補助コード
xVout	出力	BOOL	xEnable=FALSE か、エラー時は FALSE

解 説

ハードウェアからBIT(1bit)入力し結果をxVoutに出力します。

R3Readback16 [FB]

R3 アナログ(16bit)出力読み込み

(INPUT)

記号	パラメータ	型	説明
xEnable	ENABLE	BOOL	FALSE: 演算スキップ, TRUE: 演算 Default (TRUE)
iSlotNo	カードスロット番号	INT	0,1 ～ 16 (0は予約) Default (0)
iAddrNo	アドレス番号 (byte)	INT	0 ～ 30 (*1) Default (0)
eRawByteOrder	ハードウェア入力のバイト順	MSYS_ByteOrder_Enm	16bit用 Default (INT16)
rBase	ハードウェア入力のベース値	REAL	Default (0.0)
rRawL	ハードウェア入力の下限値	REAL	Default (0.0) (*2)
rRawH	ハードウェア入力の上限値	REAL	Default (10000.0) (*2)
rScaleL	ハードウェア入力の下限に割り当てる値	REAL	Default (0.0) (*3)
rScaleH	ハードウェア入力の上限に割り当てる値	REAL	Default (100.0) (*3)
rOffset	内部計算結果に対してこのオフセットを加算し出力(wVout)とします。 (現物合わせ調整用)	REAL	Default (0.0)
iT1	ハードウェア入力の一次遅れフィルタの遅れ時間	INT	[sec] 0 ～ 100 上下限で丸め Default (0)

(OUTPUT)

記号	パラメータ	型	説明
xDone	PROCESSING STATUS	BOOL	完了でTRUE (*4)
xError	ERROR STATUS	BOOL	エラー検出でTRUE
eError	ERROR CODE	R3_ERROR_Enm	エラーコード 0:No Error
uiErrorSubCode	ERROR SUB-CODE	UINT	内部エラー補助コード
wVout	ハードウェア値出力	WORD	xEnable=FALSE か、エラー時は 0 (*5)
rVout	変換後出力	REAL	xEnable=FALSE か、エラー時は NaN (*5)

(*1) 例えばチャネル1のアドレスは0、チャネル2のアドレスは2、チャネル3のアドレスは4、チャネル4のアドレスは6を指定します。

(*2) rRawL = rRawH の場合は、スケール変換を行いません。

(*3) rScaleL = rScaleH の場合は、上下限制限を行いません。

(*4) 演算が完了しエラー eMsysERR_PARA_CARDChHwError, eMsysERR_PARA_CARDChInpError, eMsysERR_PARA_CARDChInpNotEnabled のいずれかである場合は xDone = TRUE となります。それ以外のエラーの場合は xDone = FALSE, xError = TRUE となります。
(*5) xDone = FALSE, xError = TRUE の場合は wVout = 0, rVout = NaN を返します。

解 説

下記の計算式で結果をwVout,rVoutに出力します。

rVout = Limit2((Limit1(X + b) - rRawL) * a + rScaleL) + of)

X: ハードウェア入力値
a: (rScaleH – rScaleL) / (rRawH – rRawL)
b: rBase
of: rOffset
Limit1: rRawH ～ rRawL
Limit2: rScaleH ～ rScaleL

[rRawH == rRawL の場合]

rVout = Limit2(X + b + of)

[rRawH == rRawL AND rScaleH == rScaleLの場合]

rVout = X + b + of

■「指定可能 eRawByteOrder」

INT16, UINT16, R_INT16, R_UINT16

■温度データ(R3-TS, R3-RSなど)

温度データは単位が摂氏(℃)の場合、温度を10倍した値がデータとなります。
例えば23.4℃の場合には234(10進)がデータ(Raw)となります。

■R3カード(上記以外)からの入力データ

入力レンジに対して0 ～ 100% が0 ～ 10000 (10進)が対応したデータ(Raw)となります。
例えば、この入力をプログラムで%値として使用するときは
rRawL = 0.0 , rRawH = 10000.0 , rScaleL = 0.0 , rScaleH = 100.0 または

rRawL = -1500.0 , rRawH = 11500.0 , rScaleL = -15.0 , rScaleH = 115.0 と指定します。

R3Readback32 [FB]

R3 アナログ(32bit)出力読み込み

(INPUT)

記号	パラメータ	型	説明
xEnable	ENABLE	BOOL	FALSE: 演算スキップ, TRUE: 演算 Default (TRUE)
iSlotNo	カードスロット番号	INT	0,1 ～ 16 (0は予約) Default (0)
iAddrNo	アドレス番号 (byte)	INT	0 ～ 28 (*1) Default (0)
eRawByteOrder	ハードウェア入力のバイト順	MSYS_ByteOrder_Enm	32bit用 Default (INT32)
lrBase	ハードウェア入力のベース値	LREAL	Default (0.0)
lrRawL	ハードウェア入力の下限値	LREAL	Default (0.0) (*2)
lrRawH	ハードウェア入力の上限値	LREAL	Default (10000.0) (*2)
lrScaleL	ハードウェア入力の下限に割り当てる値	LREAL	Default (0.0) (*3)
lrScaleH	ハードウェア入力の上限に割り当てる値	LREAL	Default (100.0) (*3)
lrOffset	内部計算結果に対してこのオフセットを加算し出力(rVout)とします。	LREAL	Default (0.0)

(OUTPUT)

記号	パラメータ	型	説明
xDone	PROCESSING STATUS	BOOL	完了でTRUE (*4)
xError	ERROR STATUS	BOOL	エラー検出でTRUE
eError	ERROR CODE	R3_ERROR_Enm	エラーコード 0:No Error
uiErrorSubCode	ERROR SUB-CODE	UINT	内部エラー補助コード
dwVout	ハードウェア値出力	DWORD	xEnable=FALSE か、エラー時は 0 (*5)
lrVout	変換後出力	LREAL	xEnable=FALSE か、エラー時は NaN (*5)

(*1) 例えばチャネル1のアドレスは0、チャネル2のアドレスは4、チャネル3のアドレスは8、チャネル4のアドレスは12を指定します。

(*2) lrRawL = lrRawH の場合は、スケール変換を行いません。

(*3) lrScaleL = lrScaleH の場合は、上下限制限を行いません。

(*4) 演算が完了しエラー eMsysERR_PARA_CARDChHwError, eMsysERR_PARA_CARDChInpError, eMsysERR_PARA_CARDChInpNotEnabled のいずれかである場合は xDone = TRUE となります。それ以外のエラーの場合は xDone = FALSE, xError = TRUE となります。

(*5) xDone = FALSE, xError = TRUE の場合は dwVout = 0,lrVout = NaN を返します。

解 説

下記の計算式で結果をdwVout,lrVoutに出力します。

lrVout = Limit2((Limit1(X + b) - lrRawL) * a + lrScaleL) + of)

X: ハードウェア入力値

a: (lrScaleH – lrScaleL) / (lrRawH – lrRawL)

b: lrBase

of: lrOffset

Limit1: lrRawH ～ lrRawL

Limit2: lrScaleH ～ lrScaleL

[lrRawH == lrRawL の場合]

lrVout = Limit2(X + b + of)

[lrRawH == lrRawL AND lrScaleH == lrScaleLの場合]

lrVout = X + b + of

■「指定可能 eRawByteOrder」

INT32, UINT32, FLOAT32, R_INT32, R_UINT32, R_FLOAT32

INT32WS, UINT32WS, FLOAT32WS, R_INT32WS, R_UINT32WS, R_FLOAT32WS

R3ReadbackREAL [FB]

R3 アナログ(REAL)出力読み込み

(INPUT)

記号	パラメータ	型	説明
xEnable	ENABLE	BOOL	FALSE: 演算スキップ, TRUE: 演算 Default (TRUE)
iSlotNo	カードスロット番号	INT	0,1 ～ 16 (0は予約) Default (0)
iAddrNo	アドレス番号 (byte)	INT	0 ～ 28 (*1) Default (0)
eRawByteOrder	ハードウェア入力のバイト順	MSYS_ByteOrder_Enm	32bit用 Default (FLOAT32)

(OUTPUT)

記号	パラメータ	型	説明
xDone	PROCESSING STATUS	BOOL	完了でTRUE (*2)
xError	ERROR STATUS	BOOL	エラー検出でTRUE
eError	ERROR CODE	R3_ERROR_Enm	エラーコード 0:No Error
uiErrorSubCode	ERROR SUB-CODE	UINT	内部エラー補助コード
lrVout	ハードウェア値出力	LREAL	xEnable=FALSE か、エラー時は NaN (*3)

(*1) 例えばチャネル1のアドレスは0、チャネル2のアドレスは4、チャネル3のアドレスは8、チャネル4のアドレスは12を指定します。

(*2) 演算が完了しエラー eMsysERR_PARA_CARDChHwError, eMsysERR_PARA_CARDChInpError, eMsysERR_PARA_CARDChInpNotEnabled のいずれかである場合は xDone = TRUE となります。それ以外のエラーの場合は xDone = FALSE, xError = TRUE となります。

(*3) xDone = FALSE, xError = TRUE の場合は lrVout = NaN を返します。

解 説

ハードウェアからの取得結果をlrVoutに出力します。

■「指定可能 eRawByteOrder」

FLOAT32, R_FLOAT32,
FLOAT32WS, R_FLOAT32WS

R3ReadbackLREAL [FB]

R3 アナログ(LREAL)出力読み込み

(INPUT)

記号	パラメータ	型	説明
xEnable	ENABLE	BOOL	FALSE: 演算スキップ, TRUE: 演算 Default (TRUE)
iSlotNo	カードスロット番号	INT	0,1 ～ 16 (0は予約) Default (0)
iAddrNo	アドレス番号 (byte)	INT	0 ～ 24 (*1) Default (0)
eRawByteOrder	ハードウェア入力のバイト順	MSYS_ByteOrder_Enm	64bit用 Default (FLOAT64)

(OUTPUT)

記号	パラメータ	型	説明
xDone	PROCESSING STATUS	BOOL	完了でTRUE (*2)
xError	ERROR STATUS	BOOL	エラー検出でTRUE
eError	ERROR CODE	R3_ERROR_Enm	エラーコード 0:No Error
uiErrorSubCode	ERROR SUB-CODE	UINT	内部エラー補助コード
lrVout	ハードウェア値出力	LREAL	xEnable=FALSE か、エラー時は NaN (*3)

(*1) 例えばチャネル1のアドレスは0、チャネル2のアドレスは8、チャネル3のアドレスは16、チャネル4のアドレスは24を指定します。

(*2) 演算が完了しエラー eMsysERR_PARA_CARDChHwError, eMsysERR_PARA_CARDChInpError, eMsysERR_PARA_CARDChInpNotEnabled のいずれかである場合は xDone = TRUE となります。それ以外のエラーの場合は xDone = FALSE, xError = TRUE となります。

(*3) xDone = FALSE, xError = TRUE の場合は lrVout = NaN を返します。

解 説

ハードウェアからの取得結果をlrVoutに出力します。

■「指定可能 eRawByteOrder」

FLOAT64, R_FLOAT64,
FLOAT64WS, R_FLOAT64WS

R3SetBit [FB]

R3 デジタル出力書き込み

(INPUT)

記号	パラメータ	型	説明
xEnable	ENABLE	BOOL	FALSE: 演算スキップ, TRUE: 演算 (*1) Default (TRUE)
xOptimizedOut	出力モード	BOOL	FALSE: 常時, TRUE: 変化時 Default (TRUE)
xVin	出力値	BOOL	Default (FALSE) (*1)
iSlotNo	カードスロット番号	INT	0,1 ～ 16 (0は予約) Default (0)
iBitNo	ビット番号	INT	0 ～ 63 Default (0)

(OUTPUT)

記号	パラメータ	型	説明
xDone	PROCESSING STATUS	BOOL	完了でTRUE
xError	ERROR STATUS	BOOL	エラー検出でTRUE
eError	ERROR CODE	R3_ERROR_Enm	エラーコード 0:No Error
uiErrorSubCode	ERROR SUB-CODE	UINT	内部エラー補助コード
xVout	ハードウェア出力値のコピー	BOOL	xEnable=FALSE か、エラー時はFALSE

(*1) xEnable は立ち上がり、それ以外は値変化で実行されます。

解 説

ハードウェアにBIT(1bit)出力します。

R3Set16 [FB]

R3 アナログ(16bit)出力書き込み

(INPUT)

記号	パラメータ	型	説明
xEnable	ENABLE	BOOL	FALSE: 演算スキップ, TRUE: 演算 (*1) Default (TRUE)
xOptimizedOut	出力モード	BOOL	FALSE: 常時, TRUE: 変化時 Default (TRUE)
rVin	出力値	REAL	Default (NaN) (*1)
iSlotNo	カードスロット番号	INT	0,1 ～ 16 (0は予約) (*1) Default (0)

記号	パラメータ	型	説明
iAddrNo	アドレス番号 (byte)	INT	0 ～ 30 (*1) (*2) Default (0)
eRawByteOrder	ハードウェア入力のバイト順	MSYS_ByteOrder_Enm	16bit用 Default (INT16)
rBase	ハードウェア出力のベース値	REAL	Default (0.0)
rRawL	ハードウェア出力の下限値	REAL	Default (0.0) (*3)
rRawH	ハードウェア出力の上限値	REAL	Default (10000.0) (*3)
rScaleL	ハードウェア出力の下限に割り当てる値	REAL	Default (0.0) (*4)
rScaleH	ハードウェア出力の上限に割り当てる値	REAL	Default (100.0) (*4)
rOffset	内部計算結果に対して加算されていたオフセット	REAL	Default (0.0)

(OUTPUT)

記号	パラメータ	型	説明
xDone	PROCESSING STATUS	BOOL	完了でTRUE
xError	ERROR STATUS	BOOL	エラー検出でTRUE
eError	ERROR CODE	R3_ERROR_Enm	エラーコード 0:No Error
uiErrorSubCode	ERROR SUB-CODE	UINT	内部エラー補助コード
wVout	ハードウェア出力値のコピー	WORD	xEnable=FALSE か、エラー時は0

(*1) xEnable は立ち上がり、それ以外は値変化で実行されます。

(*2) 例えばチャネル1のアドレスは0、チャネル2のアドレスは2、チャネル3のアドレスは4、チャネル4のアドレスは6を指定します。

(*3) rRawL = rRawH の場合は、上下限制限を行いません。

(*4) rScaleL = rScaleH の場合は、スケール変換を行いません。

解 説

下記の計算式で結果をハードウェアに出力します。

wVout = Limit1((Limit2(rVin – of) - rScaleL) / a + rRawL + b)

a: (rScaleH – rScaleL) / (rRawH – rRawL)

b: rBase

of: rOffset

Limit1: rRawH ～ rRawL

Limit2: rScaleH ～ rScaleL

[rRawH == rRawL の場合]

wVout = Limit2(rVin - of + b)

[rRawH == rRawL AND rScaleH == rScaleLの場合]

wVout = rVin - of + b

■「指定可能 eRawByteOrder」

INT16, UINT16, R_INT16, R_UINT16

■R3カードへの出力データ

出力レンジに対して0 ～ 100% が0 ～ 10000 (10進)が対応したデータ(Raw)となります。
例えば、プログラムから%値で出力するときは
rRawL = 0.0 , rRawH = 10000.0 , rScaleL = 0.0 , rScaleH = 100.0 と指定します。

R3Set32 [FB]

R3 アナログ(32bit)出力書き込み

(INPUT)

記号	パラメータ	型	説明
xEnable	ENABLE	BOOL	FALSE: 演算スキップ, TRUE: 演算 (*1) Default (TRUE)
xOptimizedOut	出力モード	BOOL	FALSE: 常時, TRUE: 変化時 Default (TRUE)
lrVin	出力値	LREAL	Default (NaN) (*1)
iSlotNo	カードスロット番号	INT	0,1 ～ 16 (0は予約) (*1) Default (0)
iAddrNo	アドレス番号 (byte)	INT	0 ～ 28 (*1) (*2) Default (0)

記号	パラメータ	型	説明
eRawByteOrder	ハードウェア入力のバイト順	MSYS_ByteOrder_Enm	32bit用 Default (INT32)
lrBase	ハードウェア出力のベース値	LREAL	Default (0.0)
lrRawL	ハードウェア出力の下限値	LREAL	Default (0.0) (*3)
lrRawH	ハードウェア出力の上限値	LREAL	Default (10000.0) (*3)
lrScaleL	ハードウェア出力の下限に割り当てる値	LREAL	Default (0.0) (*4)
lrScaleH	ハードウェア出力の上限に割り当てる値	LREAL	Default (100.0) (*4)
lrOffset	内部計算結果に対して加算されていたオフセット	LREAL	Default (0.0)

(OUTPUT)

記号	パラメータ	型	説明
xDone	PROCESSING STATUS	BOOL	完了でTRUE
xError	ERROR STATUS	BOOL	エラー検出でTRUE
eError	ERROR CODE	R3_ERROR_Enm	エラーコード 0:No Error
uiErrorSubCode	ERROR SUB-CODE	UINT	内部エラー補助コード
dwVout	ハードウェア出力値のコピー	DWORD	xEnable=FALSE か、エラー時は0

(*1) xEnable は立ち上がり、それ以外は値変化で実行されます。

(*2) 例えばチャネル1のアドレスは0、チャネル2のアドレスは4、チャネル3のアドレスは8、チャネル4のアドレスは12を指定します。

(*3) lrRawL = lrRawH の場合は、上下限制限を行いません。

(*4) lrScaleL = lrScaleH の場合は、スケール変換を行いません。

解 説

下記の計算式で結果をハードウェアに出力します。

dwVout = Limit1((Limit2(lrVin – of) - lrScaleL) / a + lrRawL + b)

a: (lrScaleH – lrScaleL) / (lrRawH – lrRawL)

b: lrBase

of: lrOffset

Limit1: lrRawH ～ lrRawL

Limit2: lrScaleH ～ lrScaleL

[lrRawH == lrRawL の場合]

dwVout = Limit2(lrVin - of + b)

[lrRawH == lrRawL AND lrScaleH == lrScaleLの場合]

dwVout = lrVin - of + b

■「指定可能 eRawByteOrder」

INT32, UINT32, FLOAT32, R_INT32, R_UINT32, R_FLOAT32
INT32WS, UINT32WS, FLOAT32WS, R_INT32WS, R_UINT32WS, R_FLOAT32WS

R3SetREAL [FB]

R3 アナログ(REAL)出力書き込み

(INPUT)

記号	パラメータ	型	説明
xEnable	ENABLE	BOOL	FALSE: 演算スキップ, TRUE: 演算 (*1) Default (TRUE)
xOptimizedOut	出力モード	BOOL	FALSE: 常時, TRUE: 変化時 Default (TRUE)
rVin	出力値	REAL	Default (NaN) (*1)
iSlotNo	カードスロット番号	INT	0,1 ～ 16 (0は予約) (*1) Default (0)
iAddrNo	アドレス番号 (byte)	INT	0 ～ 28 (*1) (*2) Default (0)
eRawByteOrder	ハードウェア入力のバイト順	MSYS ByteOrder Enm	32bit用 Default (FLOAT32)

(OUTPUT)

記号	パラメータ	型	説明
xDone	PROCESSING STATUS	BOOL	完了でTRUE
xError	ERROR STATUS	BOOL	エラー検出でTRUE

記号	パラメータ	型	説明
eError	ERROR CODE	R3 ERROR Enm	エラーコード 0:No Error
uiErrorSubCode	ERROR SUB-CODE	UINT	内部エラー補助コード

(*1) xEnable は立ち上がり、それ以外は値変化で実行されます。

(*2) 例えばチャネル1のアドレスは0、チャネル2のアドレスは4、チャネル3のアドレスは8、チャネル4のアドレスは12を指定します。

解 説

指定の出力値をハードウェアに出力します。

■「指定可能 eRawByteOrder」

FLOAT32, R_FLOAT32

FLOAT32WS, R_FLOAT32WS

R3SetLREAL [FB]

R3 アナログ(LREAL)出力書き込み

(INPUT)

記号	パラメータ	型	説明
xEnable	ENABLE	BOOL	FALSE: 演算スキップ, TRUE: 演算 (*1) Default (TRUE)
xOptimizedOut	出力モード	BOOL	FALSE: 常時, TRUE: 変化時 Default (TRUE)
lrVin	出力値	LREAL	Default (NaN) (*1)
iSlotNo	カードスロット番号	INT	0,1 ～ 16 (0は予約) (*1) Default (0)
iAddrNo	アドレス番号 (byte)	INT	0 ～ 24 (*1) (*2) Default (0)
eRawByteOrder	ハードウェア入力のバイト順	MSYS ByteOrder Enm	64bit用 Default (FLOAT64)

(OUTPUT)

記号	パラメータ	型	説明
xDone	PROCESSING STATUS	BOOL	完了でTRUE
xError	ERROR STATUS	BOOL	エラー検出でTRUE
eError	ERROR CODE	R3 ERROR Enm	エラーコード 0:No Error
uiErrorSubCode	ERROR SUB-CODE	UINT	内部エラー補助コード

(*1) xEnable は立ち上がり、それ以外は値変化で実行されます。

(*2) 例えばチャネル1のアドレスは0、チャネル2のアドレスは8、チャネル3のアドレスは16、チャネル4のアドレスは24を指定します。

解 説

指定の出力値をハードウェアに出力します。

■「指定可能 eRawByteOrder」

FLOAT64, R_FLOAT64

FLOAT64WS, R_FLOAT64WS

SYSTEM関連

MsysSystem POUs

OP(Operand), FUN(Function), FB(Function Block), DUT(Data Unit Type)

名称	属性	機能	サポート Library *1
MsysDebugFootprint	FUN	「デバッグ」軌跡設定	
MsysDebugPrint	FUN	「デバッグ」文字列設定	
MsysSysGetSw	FB	設定スイッチ値の読み込み	
MsysSysSetLed	FB	前面LEDへの出力指示	
MsysSysTimeSpanNow	FUN	「システム」現在のチックカウント値を取得	
MsysSysTimeSpanSplit	FUN	「システム」チックカウント経過値の取得	
MsysSysSleep	FUN	「システム」時間遅延	

*1) サポートLibrary欄は、そのファンクションあるいはファンクションブロックのサポートを開始したライブラリのバージョンを記述しています。この欄が空の場合はv1.0.0以降でサポートしていることを示します。

MsysCnvByteOrderFromLE [FUN]

Convert the Src-data(Little-Endian) to Dest-data(Host byte-order) by specific byte-order

(INPUT)

記号	パラメータ	型	説明
eRawByteOrder	ByteOrder of Source data	MSYS_ByteOrder_Enm	変換前データのバイト順
pSrcData	0 or Pointer of Source data	DWORD	変換前データの格納された領域 (Little Endian) (*1)
pDestData	0 or Pointer of Destination data	DWORD	変換結果を格納する領域 (CPU Endian) (*1)
nBytes	Bytes of Data	INT	[bytes] データサイズ (1, 2, 4, 8)
pnDataType	Pointer of Result Data type	POINTER TO MSYS_ ByteOrderDataType_Enm	結果として返されるデータタイプを格納する領域 (*2) (0:none, 1:signed,

記号	パラメータ	型	説明
			2:unsigned, 3:float/double)

(OUTPUT)

記号	パラメータ	型	説明
(RETURN)	結果	MSYS_ERROR_Enm	成功は0

(*1) pSrcData, pDestData に0を指定した場合は、バイト順変換は行われず pnDataType の結果だけ返されます。

(*2) pnDataType に0を指定すると pnDataType への結果は返されません。

解 説

変換元 (Little-endian) のデータを指定のバイト順に変換します。

MsysCnvByteOrderToLE [FUN]

Convert the Src-data(Host byte-order) to Dest-data(Little-Endian) by specific byte-order

(INPUT)

記号	パラメータ	型	説明
eRawByteOrder	ByteOrder of Destination data	MSYS_ByteOrder_Enm	変換後のバイト順
pSrcData	Pointer of Source data	DWORD	変換前データの格納された領域 (CPU Endian) (*1)
pDestData	Pointer of Destination data	DWORD	変換結果を格納する領域 (Little Endian) (*1)
nBytes	Bytes of Data	INT	[bytes] データサイズ (1, 2, 4, 8)
pnDataType	Pointer of Result Data type	POINTER TO MSYS_ ByteOrderDataType_Enm	結果として返されるデータタイプを格納する領域 (*2) (0:none, 1:signed, 2:unsigned, 3:float/double)

(OUTPUT)

記号	パラメータ	型	説明
(RETURN)	結果	MSYS_ERROR_Enm	成功は0

(*1) pSrcData, pDestData に0を指定した場合は、バイト順変換は行われず pnDataType の結果だけ返されます。

(*2) pnDataType に0を指定すると pnDataType への結果は返されません。

解 説

変換元 (host byte order) のデータを指定のバイト順 (Little-endian) に変換します。

MsysDebugFootprint [FUN]

「デバッグ」軌跡設定

(INPUT)

記号	パラメータ	型	説明
diKeyCode	キーコードとして使用する任意の数値	DINT	
sCheckPoint	チェックポイントで記録する文字列	STRING	

(OUTPUT)

記号	パラメータ	型	説明
(RETURN)	結果	BOOL	完了はTRUE，未実装はFALSE

解 説

デバッグ時に使用できる軌跡追跡の情報を記録します。

記録された内容はCODESYS IDE にて確認できます。

MsysDebugPrint [FUN]

「デバッグ」文字列設定

(INPUT)

記号	パラメータ	型	説明
diKeyCode	キーコードとして使用する任意の数値	DINT	
sMessage	記録する文字列	STRING	

(OUTPUT)

記号	パラメータ	型	説明
(RETURN)	結果	BOOL	完了はTRUE，未実装はFALSE

解 説

デバッグ時に使用できるメッセージを記録します。
記録された内容はCODESYS IDEにて確認できます。

MsysSysGetSw [FB]

本体設定スイッチ状態の読み込み

(INPUT)

記号	パラメータ	型	説明
xExecute	実行	BOOL	立ち上がりで実行 Default(TRUE)

(OUTPUT)

記号	パラメータ	型	説明
xDone	PROCESSING STATUS	BOOL	完了でTRUE
xError	ERROR STATUS	BOOL	エラー検出でTRUE
eError	ERROR CODE	MSYS_ERROR_Enm	内部エラーコード 0:No Error
uiErrorSubCode	ERROR SUB-CODE	UINT	
wDipSw1	ディップスイッチ1現在値	WORD	0x00 ～ 0xff
wDipSw2	ディップスイッチ2現在値	WORD	0x00 ～ 0xff
wDipSw3	ディップスイッチ3現在値	WORD	0x00 ～ 0xff
wToggleSw	トグルスイッチ現在値	WORD	0:中央, 1:下, 2:上
wRotarySw	ロータリースイッチ現在値	WORD	0x00 ～ 0xff

解 説

設定スイッチの現在値を返します。

■各スイッチからの入力が有効かどうかは機種に依存します。実装のないスイッチの情報は0が返ります。

機種	DipSw1	DipSw2	DipSw3	ToggleSw	RotarySw
BA3-CE10	yes	yes	no	yes	yes

MsysSysSetLed [FB]

本体LEDへの出力指示

(INPUT)

記号	パラメータ	型	説明
xEnable	ENABLE	BOOL	FALSE: 演算スキップ, TRUE: 演算 Default (TRUE)
iLedRUN	RUN -LED出力状態	INT	0:OFF, 1:ON Default (0)
iLedERR	ERR -LED出力状態	INT	0:OFF, 1:ON Default (0)
iLed1	LED1出力状態	INT	0:OFF, 1:ON Default (0)
iLed2	LED2出力状態	INT	0:OFF, 1:ON Default (0)
iLed3	LED3出力状態	INT	0:OFF, 1:ON Default (0)
iLed4	LED4出力状態	INT	0:OFF, 1:ON Default (0)

(OUTPUT)

記号	パラメータ	型	説明
xDone	PROCESSING STATUS	BOOL	完了でTRUE
xError	ERROR STATUS	BOOL	エラー検出でTRUE
eError	ERROR CODE	MSYS_ERROR_Enm	内部エラーコード 0:No Error
uiErrorSubCode	ERROR SUB-CODE	UINT	

解 説

LEDの出力状態を書き込みます。

■各LEDへの出力が有効かどうかは機種に依存します(有効であってもLEDへ出力を行うためのに別途設定が必要な場合があります)。

機種	RUN	ERR	Led-1 (READY)	Led-2 (LOGIC)	Led-3 (LNK)	Led-4 (USR)
BA3-CE10	no	no	yes	yes	yes	yes

MsysSysSleep [FUN]

「システム」時間遅延

(INPUT)

記号	パラメータ	型	説明
dwMs	遅延させる時間	DWORD	ミリ秒

(OUTPUT)

記号	パラメータ	型	説明
(RETURN)	結果	BOOL	完了はTRUE，失敗はFALSE

解 説

この関数内で指定の時間が経過するまで遅延します。

MsysSysTimeSpanNow [FUN]

「システム」現在のチックカウント値を取得

(INPUT)

記号	パラメータ	型	説明
-			

(OUTPUT)

記号	パラメータ	型	説明
(RETURN)	結果	DWORD	チックカウント値

解 説

チックカウントの現在値を返します。

MsysSysTimeSpanSplit [FUN]

「システム」チックカウント経過値の取得

(INPUT)

記号	パラメータ	型	説明
dwStartTickCounter	MsysSysTimeSpanNow()で取得した値	DWORD	
pdwNowTickCounter	現在値	POINTER TO DWORD	変数のポインタが指定されれば(ポインタ値が0でなければ)

記号	パラメータ	型	説明
			MsysSysTimeSpanNow() で取得した値が返されます

(OUTPUT)

記号	パラメータ	型	説明
(RETURN)	経過時間	DWORD	ミリ秒

解 説

指定のチックカウント値から現在までの経過をミリ秒で返します。

経過チックカウント = dwStartTickCounter - now

この関数では経過チックカウント値をミリ秒に変換して返します。

UTILITY関連

MsysUtility POUs

OP(Operand), FUN(Function), FB(Function Block), DUT(Data Unit Type)

名称	属性	機能	サポート Library *1
MsysUtilASCIIbyteToString	FUN	ASCII byte の STRING 変換	
MsysUtilStringToASCIIbyte	FUN	STRING の ASCII byte 変換	

*1) サポートLibrary欄は、そのファンクションあるいはファンクションブロックのサポートを開始したライブラリのバージョンを記述しています。この欄が空の場合はv1.0.0以降でサポートしていることを示します。

MsysUtilASCIIbyteToString [FUN]

ASCII byte の STRING 変換

(INPUT)

記号	パラメータ	型	説明
byASCIIbyte	ASCII code	BYTE	

(OUTPUT)

記号	パラメータ	型	説明
(RETURN)	結果	STRING	変換結果は1文字

解 説

指定の文字コードに対応する ASCII 文字を文字列として返します。

MsysUtilStringToASCIIbyte [FUN]

STRING の ASCII byte 変換

(INPUT)

記号	パラメータ	型	説明
strASCII	ASCII string	STRING	先頭の1文字が対象となる

(OUTPUT)

記号	パラメータ	型	説明
(RETURN)	結果	BYTE	ASCII code

解 説

指定文字列の先頭1文字の文字コード(ASCII code) を返します。

(このページは空白です)

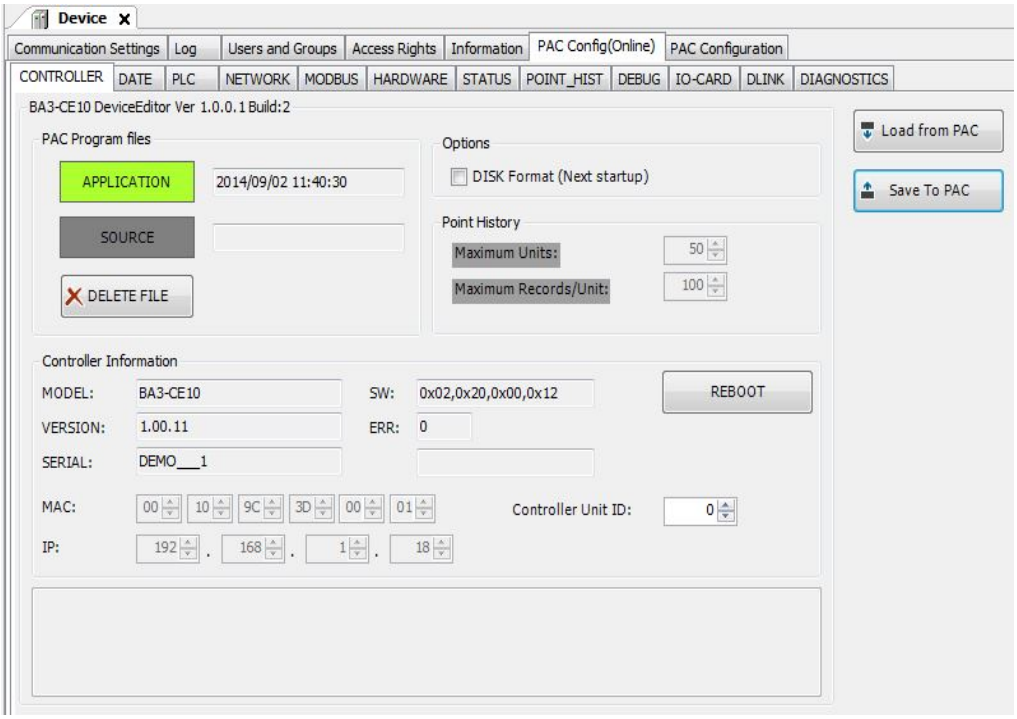
10.CODESYS IDE

10.1.BA3-CE10コントローラ設定画面

コントローラの設定は、CODESYS IDE のデバイス画面の機種別設定タブ(タブ名[BA3-CE10])で行います。

タブ名	説明
CONTROLLER	コントローラ情報の表示、ユーザアプリケーションの操作
DATE	コントローラ時刻の設定
PLC	起動遅延、NV設定、ハードウェアに関する設定
NETWORK	ネットワークに関する設定
MODBUS	MODBUS通信に関する設定
HARDWARE	ハードウェア情報の表示
STATUS	各機能の動作状況の表示
POINT_HIST	IECプログラムで蓄積されたポイント履歴情報の操作
DEBUG	デバッグ情報の確認
IO-CARD	接続されているIOカード情報の操作

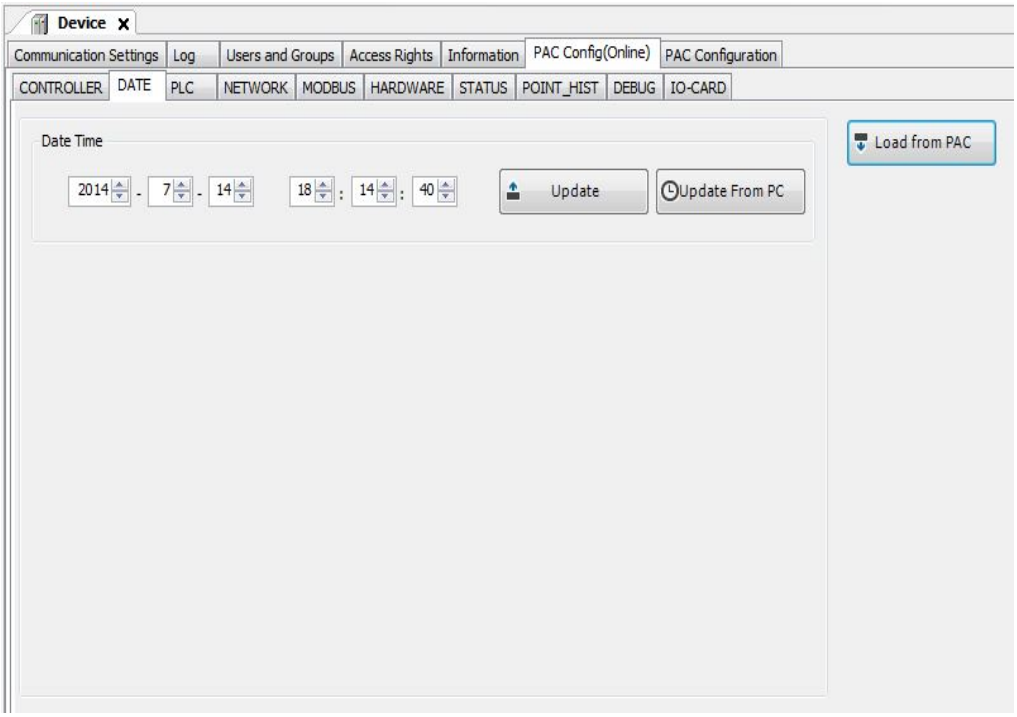
タブ: CONTROLLER



項目	説明
「最新読込み」ボタン	ボタンを押すと最新情報をコントローラから取得します。
「設定書込み」ボタン	この画面で設定された情報をコントローラに設定します。
PACプログラムファイル	コントローラ内に存在するユーザアプリケーションの存在が表示されます。 「アプリケーション」： ブートアプリケーションの存在を表示 「ソース」： ソースダウンロードで転送されたソースの存在を表示 表示色： グレー＝存在なし 赤＝異常（不完全） 緑＝存在
「ファイル削除」ボタン	コントローラ内に存在しているユーザアプリケーション、転送済みソースを削除します。
「最大履歴ユニット数」	ポイント履歴のユニット（管理領域）数を指定します。 履歴データ（レコード）は、履歴ユニット毎に蓄積されます。 履歴ユニット毎に蓄積可能なデータ数は、「コントローラ内最大ポイント履歴レコード数」÷「最大履歴ユニット数」で決定されます。
「DISK フォーマット」 チェックボックス	コントローラ内のファイル記憶領域を初期化します。 通常運用では使用しない機能です。 この機能はユーザアプリケーション転送中の電源断などによりファイル情報が不正となった場合に使用します。
ポイント履歴	FBのDdcPointHistoryで使用できる最大ユニット数とユニットあたりの最大レコード数が表示されます。

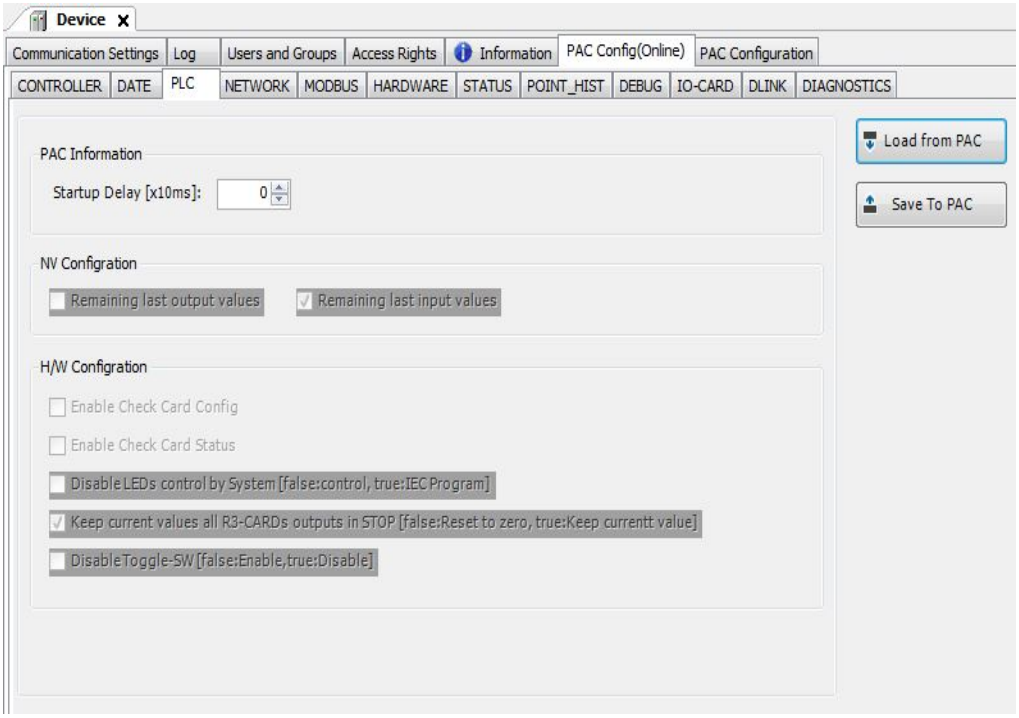
項目	説明
Controller Unit ID	コントローラ固有の識別番号を指定します。 この識別番号は Global Data point のデータ放送でも使用されますが、個体を識別する必要の有るIECプログラムで利用できます。
コントローラ情報	型式、ファームウェアバージョン、現在有効なIPアドレスなどが表示されます。
「リブート」ボタン	コントローラを再起動します。 この操作を行う際は再起動しても安全な状況であることを確認する必要があります。

タブ:DATE



項目	説明
「最新読込み」ボタン	ボタンを押すと最新情報をコントローラから取得します。
日付時刻	「最新読込み」を押すとコントローラの現在時刻が表示されます。
「設定」ボタン	手動で「日付時刻」に設定した時刻をコントローラに設定します。
「PCと同期」ボタン	PCの現在時刻をコントローラに設定します。

タブ: PLC

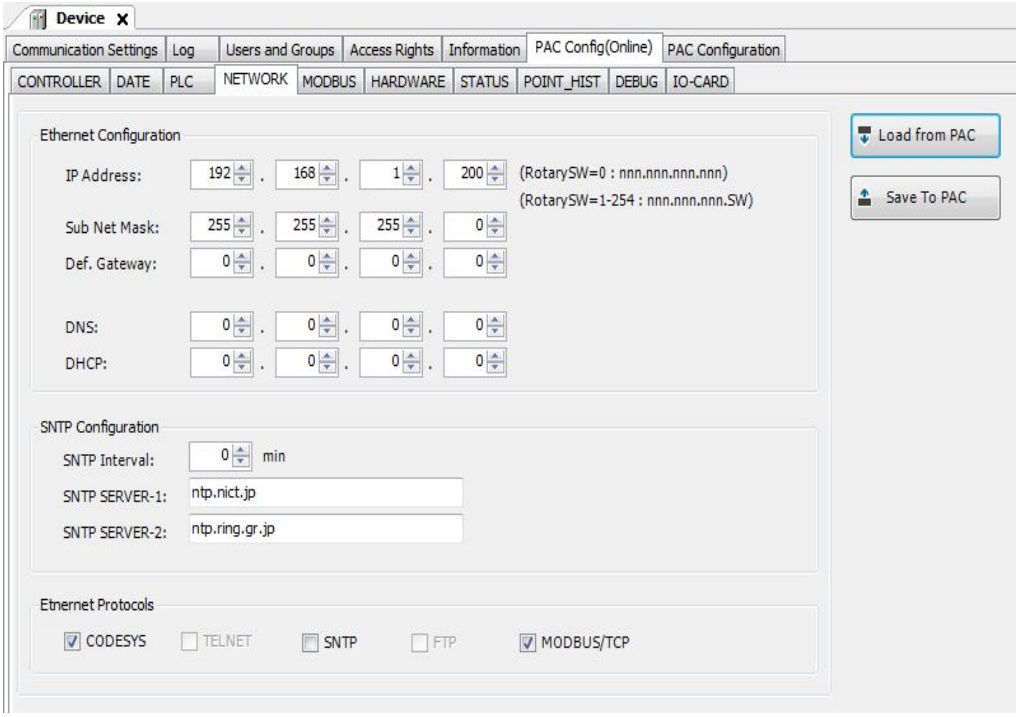


項目	説明
「最新読込み」ボタン	ボタンを押すと最新情報をコントローラから取得します。
「設定書込み」ボタン	この画面で設定された情報をコントローラに設定します。
「起動遅延時間」	電源ONからコントローラを動作させるまでの時間を 10ms 単位で設定します。 例 100 を設定すると 1秒遅延します。
「最終の出力値を保持する」 チェックボックス (*1)	コントローラ起動時、次のMODBUS領域の初期値を決定します。 001025 - 003328, 400513 - 400768 TRUEの場合は前回最終値となります。 FALSEの場合は0を設定します。
「最終の入力値を保持する」 チェックボックス (*1)	コントローラ起動時、次のMODBUS領域の初期値を決定します。 101025 - 102048, 104097 - 105120, 105377 - 105632, 300257 - 300512 TRUEの場合は前回最終値となります。 FALSEの場合は0を設定します。
「Disable Led 1-4 control by System」 チェックボックス (*1)	前面のLED 1 - 4 表示制御方法を指定します。 TRUEの場合はユーザアプリケーションが行います。 FALSEの場合はシステムが行います(既定の表示)。
「RUN->STOP」 (*1)	RUN -> STOP 切替後の同一ベースR3 IO-CARD 出力値の状態を指定します。 TRUEの場合はデジタル、アナログの出力値を最終値のまま保持します。

項目	説明
	FALSEの場合はデジタル、アナログの出力値を0に設定します。
「Disable Toggle-SW」 (*1)	前面のトグルスイッチの動作を指定します。 TRUEの場合はトグルスイッチを無効にします(「RUN」として動作)。 FALSEの場合はトグルスイッチ規定の動作を行います。

*1) これらの項目は読み取り専用でありコントローラに設定されている値を表示します。これらの項目への設定は「タブ:PAC Configuration」で行うことができます。

タブ:NETWORK

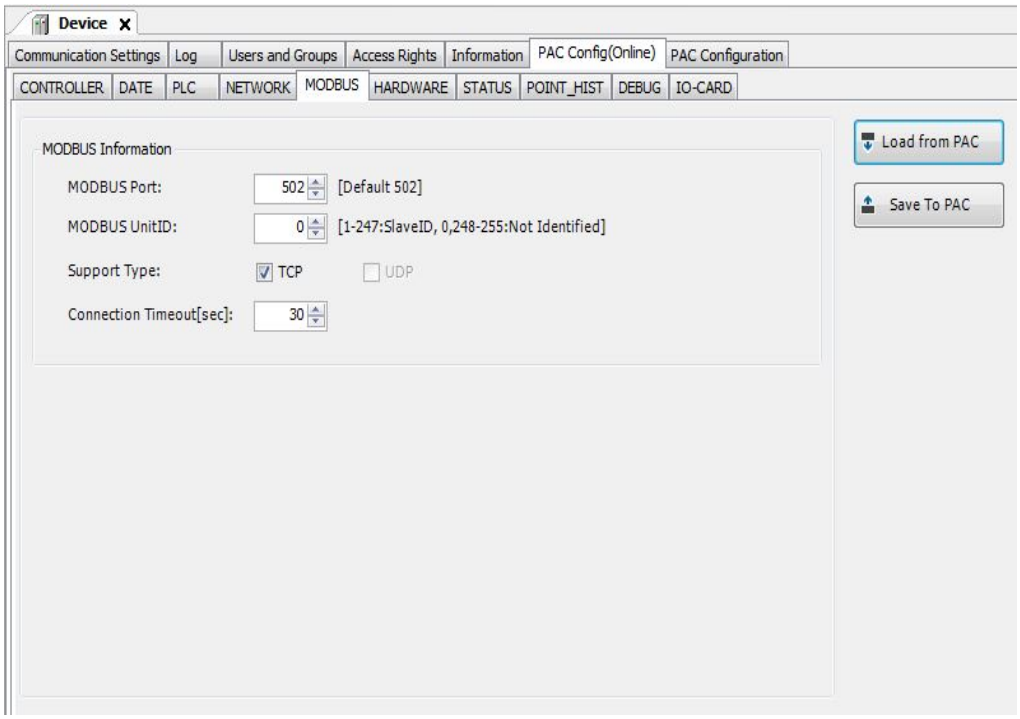


項目	説明
「最新読込み」ボタン	ボタンを押すと最新情報をコントローラから取得します。
「設定書込み」ボタン	この画面で設定された情報をコントローラに設定します。
「IP Address」	IP アドレスを設定します。(初期値 192.168.1.200) 有効となるアドレスは前面ロータリスイッチの設定に依存します。 ロータリスイッチが0の場合: ＜このIPアドレス＞ ロータリスイッチが1～254の場合: ＜このIPアドレス上位3桁＞.＜ロータリスイッチ値＞
「Sub Net Mask」	サブネットマスクを設定します。(初期値 255.255.255.0)
「Def. Gateway」	デフォルトゲートウェイを設定します。(初期値 0.0.0.0 = なし)
「DNS」	DNSサーバーアドレスを設定します。(初期値 0.0.0.0 = なし)
「DHCP」	DHCPサーバーアドレスを設定します。(初期値 0.0.0.0 = なし)

項目	説明
「SNTP Interval」	SNTPリクエスト周期を設定します。（初期値 0 = 時刻同期しない） 時刻同期は、ここで指定された間隔で実行され取得した時刻に更新します。
「SNTP SERVER-1」	SNTP第一サーバーを示すURLを設定します。
「SNTP SERVER-2」	SNTP第二サーバーを示すURLを設定します。 第一サーバーが使用できない場合に第二サーバーを試します。
「Ethernet Protocols」	このコントローラで使用するプロトコルを指定します。 [COESYS]: Softlogic 通信をサポートします。(必須 *1) [TELNET]: 選択不可 [SNTP]: SNTPをサポートします。(SNTP使用時には有効にする必要があります) [FTP]: 選択不可 [MODBUS/TCP]: MODBUS通信をサポートします。(推奨)

*1) [CODESYS]通信を無効にするとEthernet 上からこれらの設定やプログラミングができなくなります。この設定を有効に戻すためにはコントローラ本体の初期化(出荷時設定状態)を行う必要があります。

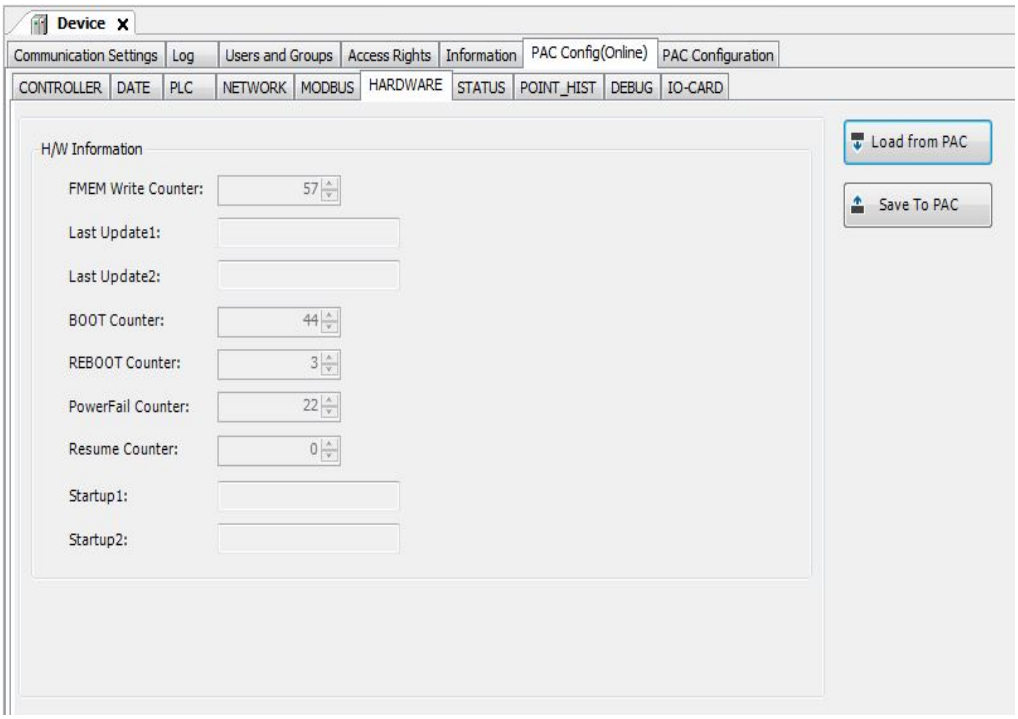
タブ: MODBUS



項目	説明
「最新読込み」ボタン	ボタンを押すと最新情報をコントローラから取得します。
「設定書込み」ボタン	この画面で設定された情報をコントローラに設定します。

項目	説明
「MODBUS Port」	使用するポート番号を設定します。（初期値 502）
「MODBUS UnitID」	ユニットIDを設定します。（初期値 0 = スレーブID無視） 設定可能な値を示します。 1～247：要求中のスレーブIDを認識しこの値と一致しない要求にはエラーを返します。 0，248～255：要求中のスレーブIDを無視します。全ての要求を処理します。
「Support Type」	サポートする通信タイプを指定します。（TCPのみ指定可能）
「Connection Timeout」	接続タイムアウトを設定します。（初期値 30秒） この機能は接続してから一定（この値）時間に新たな要求が無ければ自動的に接続を切断します。

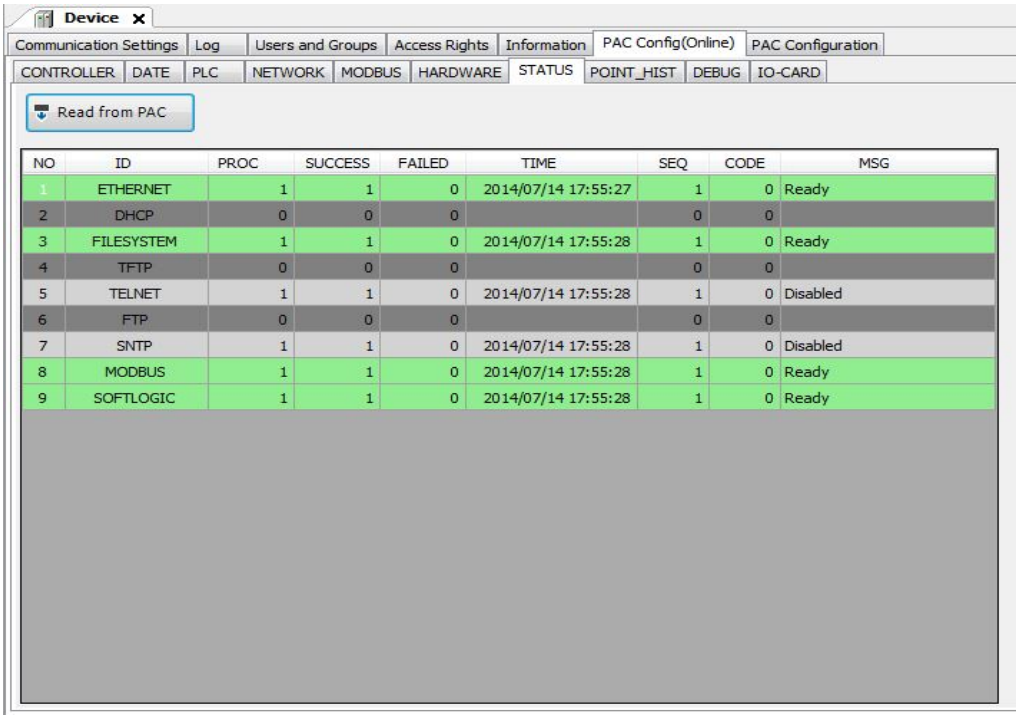
タブ：HARDWARE



項目	説明
「最新読込み」ボタン	ボタンを押すと最新情報をコントローラから取得します。
「設定書込み」ボタン	この画面で設定された情報をコントローラに設定します。
「ハードウェア情報」	ハードウェアに関する情報を表示します。 [FMEM Write Counter]：フラッシュメモリ書換回数 [Last Update1]：前回フラッシュメモリ書換日付 [Last Update2]：前々回フラッシュメモリ書換日付 [BOOT Counter]：起動回数 [REBOOT Counter]：再起動回数

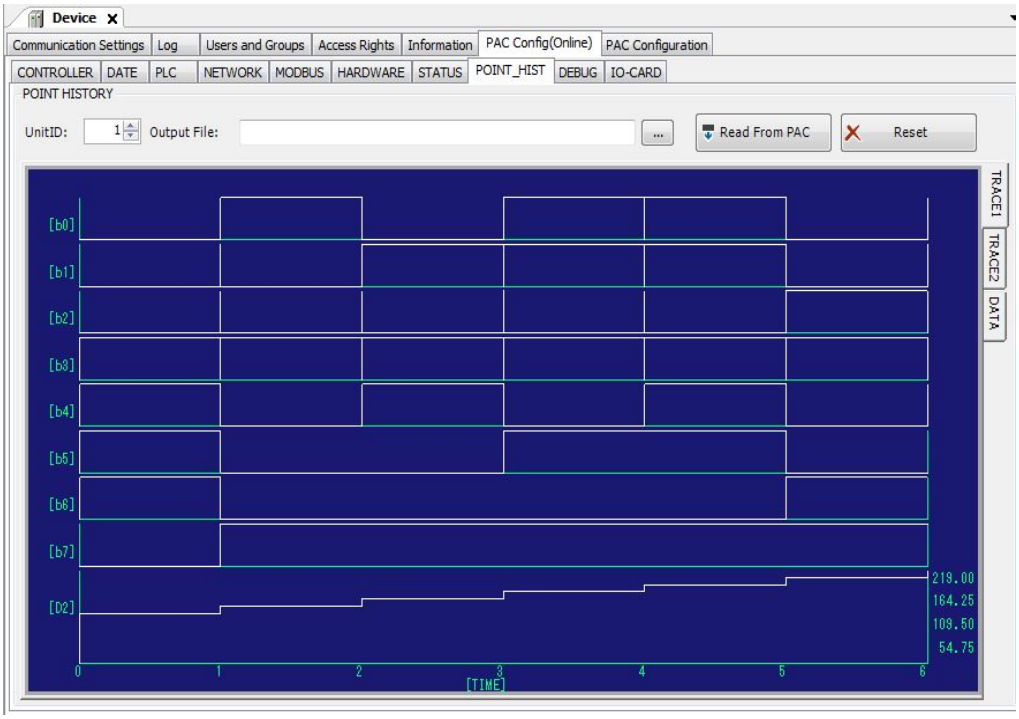
項目	説明
	[PowerFail Counter]: 電源異常検出数 [Resume Counter]: 電源異常回復数 [Startup1]: 起動日付 [Startup2]: 前回起動日付

タブ: STATUS



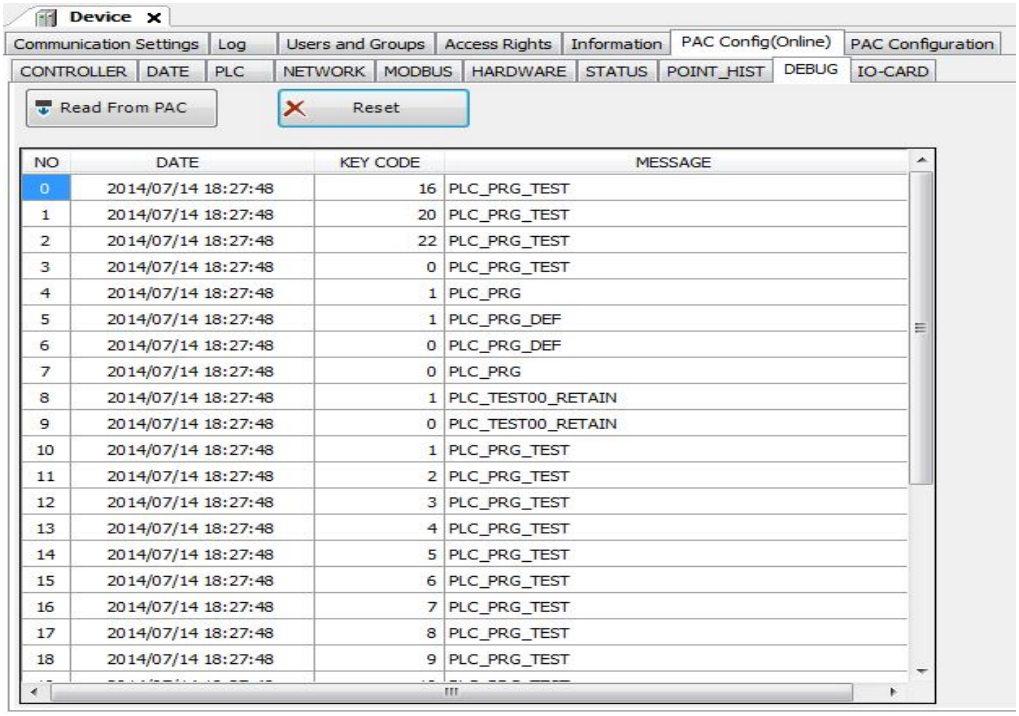
項目	説明
「最新読込み」ボタン	ボタンを押すと最新情報をコントローラから取得します。
モジュール一覧	各モジュールに関する情報を表示します。 [NO]: 連番 [ID]: モジュール記号 [実行]: 実行回数 [成功]: 成功回数 [失敗]: 失敗回数 [時刻]: 状態変化 [SEQ]: 連番 (オーバーフロー時は1に戻る) [CODE]: エラーコード [MSG]: 状態を表す文字列

タブ: POINT_HIST



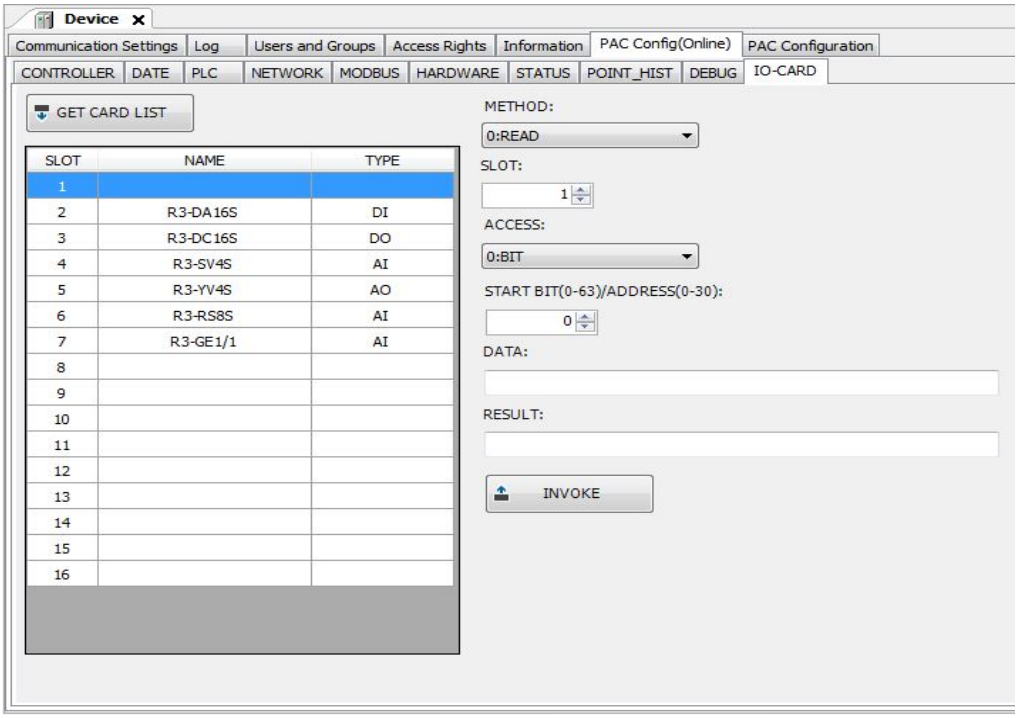
項目	説明
「履歴取得」ボタン	ボタンを押すとコントローラ内に記録されている指定の[UnitID]の履歴情報を取得し出力ファイルへ書き出します。
「履歴リセット」	ボタンを押すとコントローラ内に記録されている指定の[UnitID]の履歴情報をリセット(クリア)します。
「UnitID」	履歴ユニット番号を指定します。(範囲は 1 ～ 最大履歴ユニット数)
「出力ファイル名」	出力ファイル名を指定します。 ファイル名の拡張子が省略されている場合は自動的に '.csv' が付加されます。 また、ファイルへの出力が必要でない場合は空欄としておきます。 ファイルは、タイトル行とデータ行(カンマ区切り)で書き出されます。
データ表示タブ	「TRACE1」: xVin1 ～ xVin8, usiVin をグラフ描画します。 「TRACE2」: rVin をグラフ描画します。 「DATA」: 表形式でデータを表示します。

タブ: DEBUG



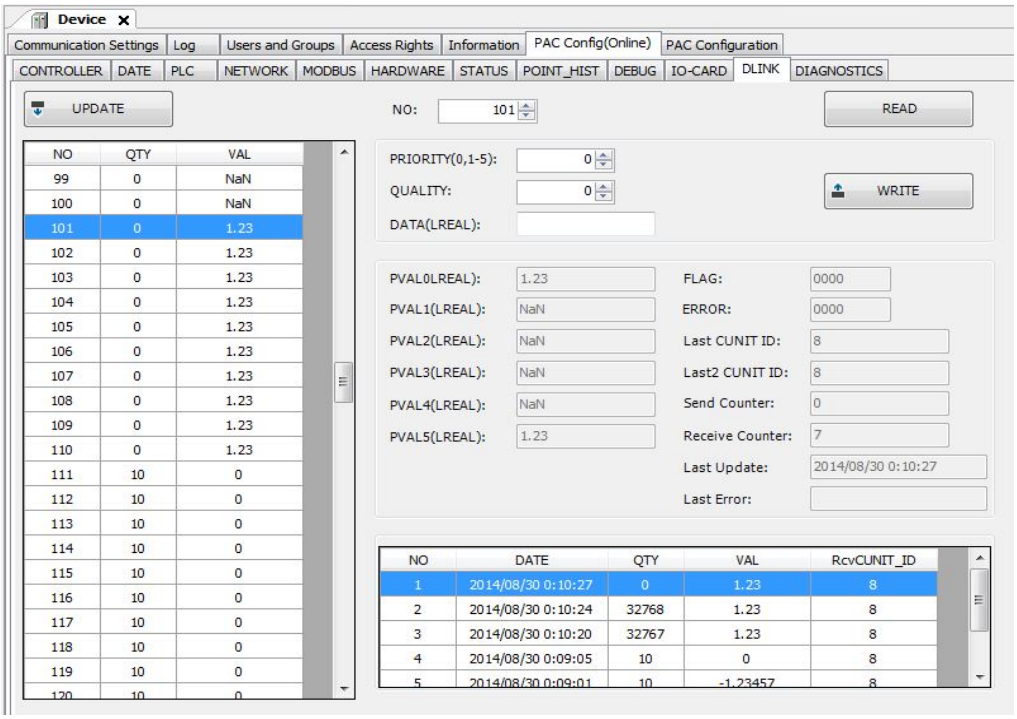
項目	説明
「最新読込み」ボタン	ボタンを押すとコントローラ内に記録されているデバッグ出力情報を取得します。
「情報のリセット」	ボタンを押すとコントローラ内に記録されているデバッグ出力情報をリセット(クリア)します。
デバッグ出力表示	デバッグ出力を表示します。 [NO]: 連番 [DATE]: 出力日付 [KEY CODE]: デバッグ出力で指定された選択(フィルタ)用数値 [MESSAGE]: デバッグ出力文字列

タブ:IO-CARD



項目	説明
「カード情報取得」ボタン	ボタンを押すとコントローラと同一ベースに存在するIOカード情報を取得します。 取得された結果は一覧で表示されます。
「実行」	ボタンを押すとパラメータで指定された要求を実行します。 要求:[0:READ]の場合 パラメータは [SLOT],[ACCESS],[START BIT/ADDRESS] 結果は [データ],[結果] 要求:[1:READBACK]の場合 パラメータ [SLOT],[ACCESS],[START BIT/ADDRESS], [データ] 結果は [結果] 要求:[2:WRITE]の場合 パラメータ [SLOT],[ACCESS],[START BIT/ADDRESS], [データ] 結果は [結果]
パラメータ	[要求]: 要求を指定します。 [SLOT]: スロット番号(1～16)を指定します。 [ACCESS]: データアクセス方法を指定します。 (0:BIT, 1:BYTE, 2:WORD, 4:DWORD/REAL, 8:LWORD/LREAL) [START BIT/ADDRESS]: 開始位置を指定します。 (BIT:0～63, BIT以外:0～30バイト目) [データ]: 書き込むデータを指定します。
「結果」	実行結果コードを表示します。(RSLT, RSLT2 共に 0 が正常)

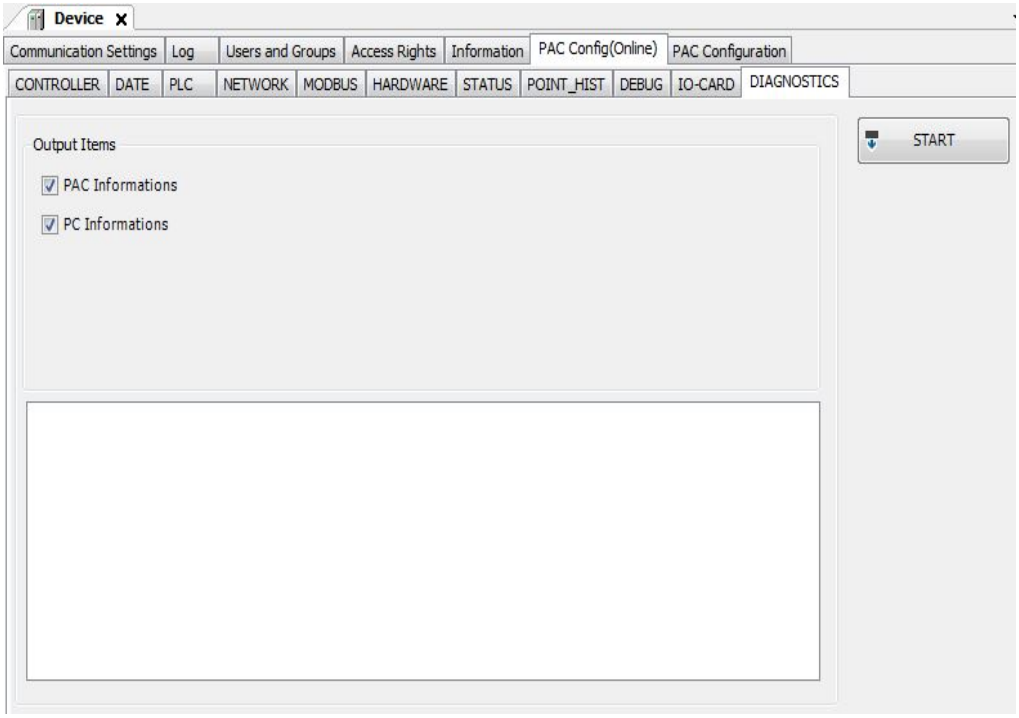
タブ: DLINK



項目	説明
「更新」ボタン	ボタンを押すと最新情報をコントローラから取得してデータポイント一覧表を更新します。
「読み込み」ボタン	「番号」で指定されているデータポイントの情報をコントローラから取得します。
「書き込み」ボタン	この画面で設定された情報をコントローラの「No」で指定されているデータポイントに設定します。
データポイント一覧	データポイント番号、品質、値を一覧で表示します。
「番号」 スピンボックス	データポイント番号を指定します。
「優先度」スピンボックス	データポイントの値には5つの書き込み優先度を持ち、優先番号1が最高優先度です。 通常操作は優先度5を使用します。 優先する値を解除するためには該当の優先度の値としてNaNを書き込みます。
「品質」 スピンボックス	読み込みではデータの品質が表示され、書き込みでは値を設定できます。
「データ入力」 テキストボックス	読み込みでは指定の優先度の現在値が表示されます。 書き込みでは指定の優先度に書き込む値を入力します。 指定の優先度に設定されている値を解除するには非数(NaN)値を設定します。 このNaN値の設定は入力テキストボックスを空にするか文字列'NaN'の3文字を入力して「書き込み」ボタンを押します。
PVAL0	現在値が表示されます。
PVAL1 ～	優先度1～5に設定されている値 非数(NaN)が設定されている優先度は評価されないで残りの内で最も優先度に設定されている

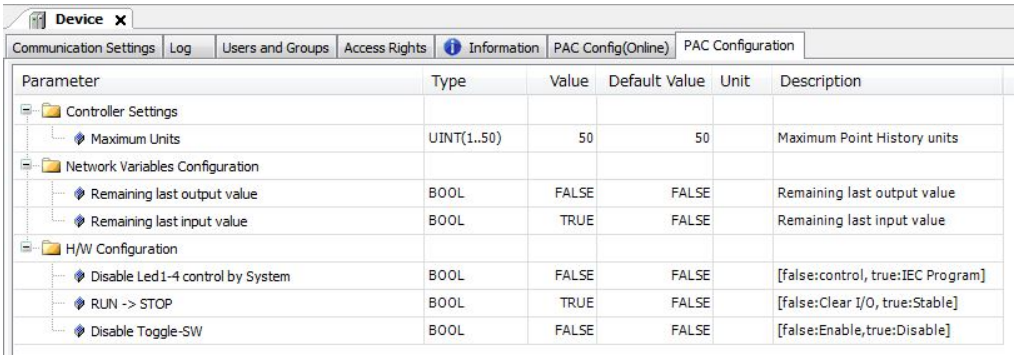
項目	説明
PVAL5	値が現在値(PVAL0)とされます。
フラグ	システムで使用
エラー	エラー発生時にエラー番号が表示されます。
最新UNITID	最終データの送信元UNITIDが表示されます。
前回UNITID	前回データの送信元UNITIDが表示されます。
送信回数	Publish:TRUEの場合の統計情報で送信回数が表示されます。
受信回数	統計情報で受信回数が表示されます。
最新更新時刻	最終更新時刻が表示されます。
最新エラー発生時刻	最終エラー発生時刻が表示されます。

タブ: DIAGNOSTICS



項目	説明
「開始」ボタン	ボタンを押すと出力項目で指定された情報をコントローラ、PCから取得してファイルに出力します。
出力情報	出力する項目を指定します。 ・PAC情報 ： コントローラから情報を取得します。 ・PC情報 ： 現在使用PCの動作環境を取得します。

タブ: PAC Configuration



項目	説明
Maximum Units	DdcPointHistory ファンクションブロックで使用するユニット（管理領域）の最大数を指定します。 履歴データ（レコード）は、自動的に算出されます。 計算式： ユニットあたりの最大履歴データ数 ＝ 履歴データ最大数（固定） / ユニット数
Remaining last output value	ネットワーク(Modbus Read) Memory Input 領域の最終状態を記憶するかどうかを指定します。 FALSE: 記憶しない（初期値）, TRUE: 記憶する
Remaining last input value	ネットワーク(Modbus Write) Memory Output 領域の最終状態を記憶するかどうかを指定します。 FALSE: 記憶しない（初期値）, TRUE: 記憶する
Disable Led1-4 control by System	本体前面に配置されている 4つのLED の制御をシステムで行うかどうかを指定します。 FALSE: システムで制御（初期値）, TRUE: IEC プログラムで制御 (MsysSysSetLed ファンクションブロック)
RUN -> STOP	実行状態から停止状態に移行した際の挙動を指定します。 FALSE: 入出力状態をクリアする（初期値）, TRUE: 変更しない
Disable Toggle-SW	前面のトグルスイッチを無効にするかどうかを指定します。 FALSE: 有効（初期値）, TRUE: 無効

索 引