OMRON

産業用PCプラットフォーム

NYシリーズ

IPC マシンコントローラ
産業用パネル型PC／産業用ボックス型PC

ユーザーズマニュアル
ソフトウェア編

形NY532-1500
形NY532-1400
形NY532-1300
形NY532-5400
形NY512-1500
形NY512-1400
形NY512-1300

産業用パネル型PC
産業用ボックス型PC



SYSMAC
always in control

SBCA-436J

おことわり

(1) 本マニュアル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。

(2) 本マニュアルの内容に関しては、改良のため予告なしに仕様などを変更することがあります。
あらかじめご了承ください。

(3) 本マニュアルの内容に関しては、万全を期しておりますが、万一ご不審な点や誤りなど、お気づきのことがありましたら、お手数ですが、弊社支店または営業所までご連絡ください。
その際、巻末記載の Man.No.（マニュアル No.）も併せてお知らせください。

商標

- Sysmac は、オムロン株式会社 FA 機器製品の日本およびその他の国における商標または登録商標です。
- Microsoft、Windows、Excel、Visual Basic、Microsoft Edgeは、米国Microsoft Corporationの米国およびその他の国における登録商標または商標です。
- EtherCAT®は、ドイツ Beckhoff Automation GmbH によりライセンスされた特許取得済み技術であり登録商標です。
- ODVA、CIP、CompoNet、DeviceNet、EtherNet/IP は ODVA の商標です。
- SD、SDHC ロゴは、SD-3C,LLC の商標です。
- Intel、Intel Core は、米国およびその他の国における Intel Corporation の商標です。

その他、本文中に掲載している会社名および製品名は、それぞれ各社の商標または登録商標です。

著作権

- スクリーンショットはマイクロソフトの許諾を得て使用しています。
- 本製品にはサードパーティ製のソフトウェアが組み込まれています。このソフトウェアに関連するライセンスと著作権については、http://www.fa.omron.co.jp/nj_info_j/ をご覧ください。

はじめに

このたびは、NY シリーズ IPC マシンコントローラ 産業用パネル型 PC ／産業用ボックス型 PC をお買い上げいただきまして、ありがとうございます。
本マニュアルでは、対象となる製品の産業用パネル型 PC と産業用ボックス型 PC を総称して「NY シリーズ産業用 PC」と記載する場合があります。また、NY シリーズ産業用 PC のリアルタイム OS に組み込まれたコントローラ機能が直接制御する範囲を「コントローラ」と記載する場合があります。
本マニュアルは、NY シリーズコントローラを使用する上で、必要な情報を記載しています。お使いになる前に本マニュアルをよく読んで、機能・性能などを十分ご理解いただき、システムの構築にご活用ください。
また、お読みになった後も本マニュアルは大切に保管して、いつも手元においてお使いください。

対象となる読者の方々

本マニュアルは、次の方を対象に記述しています。
電気の知識（電気工事士あるいは同等の知識）を有する方で

- FA 機器の導入を担当される方
- FA システムを設計される方
- FA 機器を設置、接続される方
- FA 現場を管理される方

また、プログラミング言語に関して、国際標準規格 IEC 61131-3 もしくは国内標準規格 JIS B 3503 に規定される内容をご理解いただいている方を対象に記述しています。

対象となる製品

本マニュアルは以下の製品を対象にしています。

- NY シリーズ IPC マシンコントローラ 産業用パネル型 PC
  - 形 NY532-15□□
  - 形 NY532-14□□
  - 形 NY532-13□□
  - 形 NY532-5400

- NY シリーズ IPC マシンコントローラ 産業用ボックス型 PC
  - 形 NY512-15□□
  - 形 NY512-14□□
  - 形 NY512-13□□

各製品の仕様の一部または制約事項が、他のマニュアルに記載されている場合があります。🕮「分冊構成 (P.2)」および 🕮「関連マニュアル (P.39)」をご確認ください。

分冊構成

本製品のマニュアルは、次表のとおりに分冊されています。目的に応じてお読みいただき、本製品を活用してください。
本製品の操作には、おもにオートメーションソフトウェア Sysmac Studio を使用します。Sysmac Studio については、📖『Sysmac Studio Version 1 オペレーションマニュアル (SBCA-470)』を参照してください。

使用目的	マニュアル										
	基本情報										
	NY シリーズ IPC マシンコントローラ 産業用パネル型 PC ユーザーズマニュアル ハードウェア編	NY シリーズ IPC マシンコントローラ 産業用ボックス型 PC ユーザーズマニュアル ハードウェア編	NY シリーズ IPC マシンコントローラ 産業用パネル型 PC／産業用ボックス型 PC ユーザーズマニュアル セットアップ編	NY シリーズ IPC マシンコントローラ 産業用パネル型 PC／産業用ボックス型 PC ユーザーズマニュアル ソフトウェア編	NY シリーズ コマンドリファレンスマニュアル 基本編	NY シリーズ IPC マシンコントローラ 産業用パネル型 PC／産業用ボックス型 PC ユーザーズマニュアル モーション制御編	NY シリーズ コマンドリファレンスマニュアル モーション編	NY シリーズ IPC マシンコントローラ 産業用パネル型 PC／産業用ボックス型 PC ユーザーズマニュアル 内蔵 EtherCAT ポート編	NY シリーズ IPC マシンコントローラ 産業用パネル型 PC／産業用ボックス型 PC ユーザーズマニュアル 内蔵 EtherNet/IP ポート編	NJ/NY シリーズ NC 統合コントローラ ユーザーズマニュアル	NY シリーズ トラブルシューティングマニュアル
NY シリーズパネル型 PC の概要を知る	○										
NY シリーズボックス型 PC の概要を知る		○									
装着、設置、ハードウェア設定を行う	○	○									
モーション制御を行う場合						○					
EtherCAT を使用する場合								○			
EtherNet/IP を使用する場合									○		
セットアップを行う[*1]			○								
初期設定を行う場合											
コントローラの使用準備を行う場合											
ソフトウェア設定を行う				○							
モーション制御を行う場合						○					
EtherCAT を使用する場合								○			
EtherNet/IP を使用する場合									○		
数値制御を行う場合										○	
ユーザプログラムを作成する				○	○						
モーション制御を行う場合						○	○				
EtherCAT を使用する場合								○			
EtherNet/IP を使用する場合									○		
数値制御を行う場合										○	
異常処理を組み込む場合											○
動作確認やデバッグを行う				○							
モーション制御を行う場合						○					
EtherCAT を使用する場合								○			
EtherNet/IP を使用する場合									○		
数値制御を行う場合										○	
異常管理機能やトラブル発生時の対処方法を知る[*2]										△	○
保守作業について知る	○	○									
モーション制御を行う場合						○					
EtherCAT を使用する場合								○			
EtherNet/IP を使用する場合									○		

*1. Windows 側のセットアップ方法やユーティリティなどの使い方については、📖『NY シリーズ 産業用パネル型 PC ／産業用ボックス型 PC ユーザーズマニュアル セットアップ編 (SBCA-441)』を参照してください。
*2. 異常管理の考え方と異常項目については、📖『NY シリーズ トラブルシューティングマニュアル (SBCA-438)』を参照してください。ただし、△マークがついているマニュアルの製品に特有の異常の詳細については、△マークがついているマニュアルを参照してください。

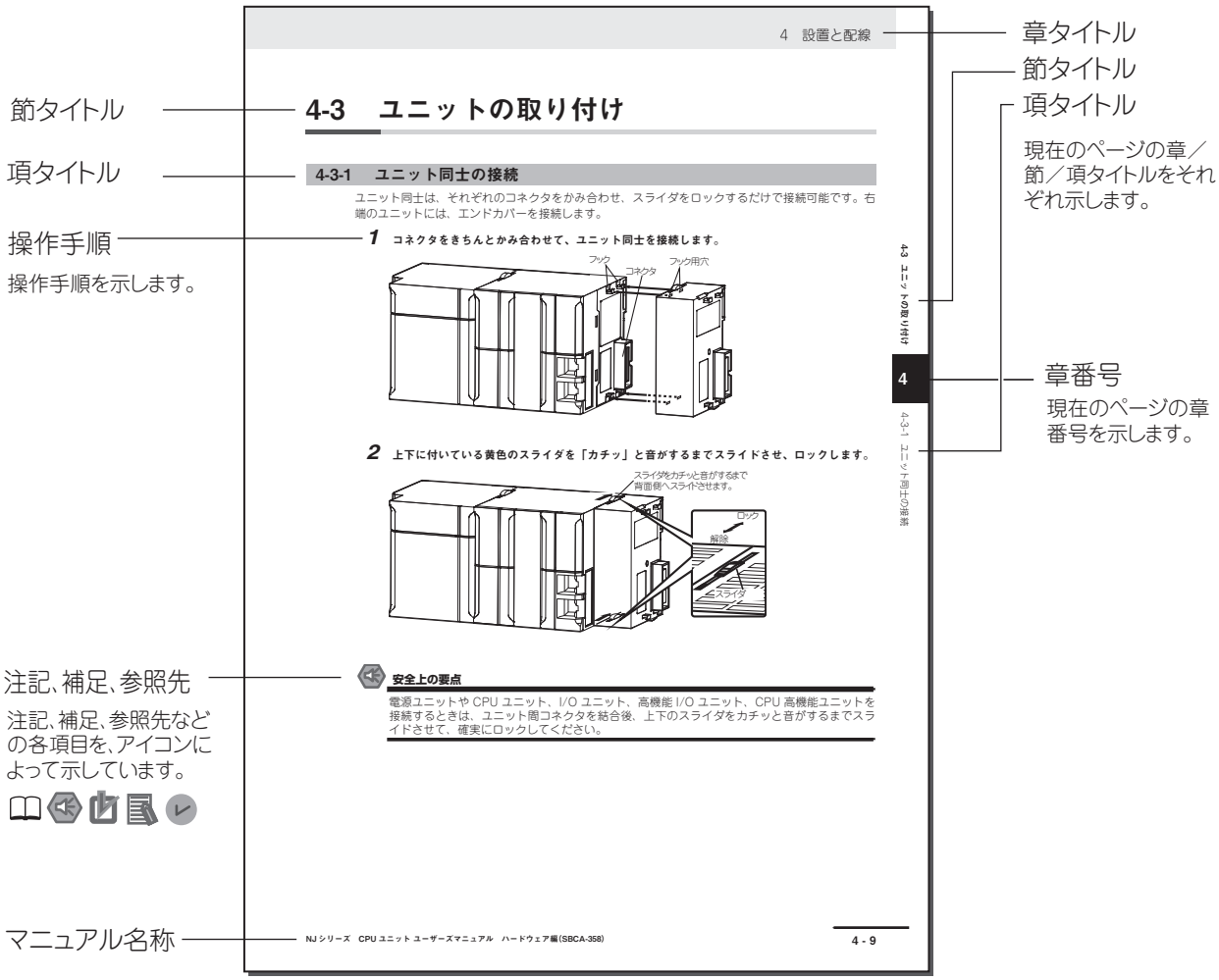
マニュアルの読み方

このマニュアルに記載している機能の説明は、NJ/NXシリーズと共通で書かれている部分があります。そのため、以下の点に注意してください。

- NJ/NX/NY シリーズの共通機能には、同じ機能名称を使用しています。「CPU ユニット名称機能」、「CPU ユニットへの書込プロテクト機能」など、機能名称に「CPU ユニット」を含む場合、NY シリーズでは、「コントローラ」のことを指します。
- 本マニュアルの機能仕様の一覧で、「CPU ユニット」と記載している場合も、NY シリーズでは、「コントローラ」のことを指します。

ページ構成

このマニュアルの各ページの構成は次のとおりです。



このページは、説明用の見本です。実際の内容とは異なります。

アイコン

このマニュアルで使われているアイコンには、次のような意味があります。



安全上の要点

製品を安全に使用するために実施または回避すべきことを示します。



使用上の注意

製品の動作不能、誤動作、または性能･機能への悪影響を予防するために実施または回避すべきことを示します。



参考

必要に応じて読んでいただきたい項目です。
知っておくと便利な情報や、使用する上で参考となる内容について説明しています。



詳しい情報や、関連情報の記載されている場所を示します。



バージョン関連情報

コントローラや Sysmac Studio の、バージョンによる性能や機能の違いについて説明しています。

表記についての留意事項

本マニュアルでは、Sysmac Studio から実機コントローラにデータを転送することを「ダウンロード」、実機コントローラからデータを Sysmac Studio に転送することを「アップロード」と記述しています。
Sysmac Studio では、「ダウンロード」および「アップロード」は、いずれも「同期」機能を使用します。Sysmac Studio の「同期」機能は、パソコン上のデータと実機コントローラのデータを自動照合し、どちらの方向に転送するかをユーザが選択する機能です。

目次構成

1 NYシリーズ コントローラの概要

2 NYシリーズ コントローラの動作

3 I/O ポートと スレーブ／ユニット構成

4 コントローラ設定

5 タスク設計

6 プログラミング

7 動作確認と運転

8 コントローラ内蔵機能

9 バックアップ機能

10 通信設定

11 実際の使用手順例

A 付録

I 索引

目次

はじめに 1
対象となる読者の方々 1
対象となる製品 1

分冊構成 2

マニュアルの読み方 3
ページ構成 3
アイコン 4
表記についての留意事項 4

目次構成 5

ご承諾事項 14

安全上の注意 16
安全に使用していただくための表示と意味 16
図記号の説明 17
警告 18
注意 22

安全上の要点 24

使用上の注意 31

法規と規格 35
海外でのご使用 35
EU 指令への適合 35
ソフトウェアのライセンスと著作権 35

バージョン 36
バージョンの確認方法 36

関連マニュアル 39

用語説明 41

マニュアル改訂履歴 45

第 1 章　NY シリーズコントローラの概要

1-1　NY シリーズコントローラとは 1-2
1-1-1　特長 1-3
1-1-2　システム構成の概要 1-5
1-2　仕様の概要 1-7
1-3　商品形式 1-9
1-4　NY シリーズの使用手順の全体 1-10
1-4-1　手順の概要 1-10
1-4-2　手順の詳細 1-11

第 2 章　NY シリーズコントローラの動作

2-1　NY シリーズコントローラの動作の概要 2-2

2-1-1　NY シリーズコントローラのしくみの概要 2-2
2-1-2　NY シリーズコントローラの各状態における動作の概要 2-3
2-2　ソフトウェアのしくみ 2-4
2-2-1　ソフトウェア構成 2-4
2-2-2　ソフトウェアの動作 2-5
2-3　I/O とのアクセスのしくみ 2-10
2-3-1　変数の種類 2-10
2-3-2　変数と I/O のアクセス 2-13
2-4　シーケンス制御とモーション制御のしくみ 2-16
2-4-1　制御の概要 2-16
2-4-2　シーケンス制御のしくみ 2-18
2-4-3　モーション制御のしくみ 2-19
2-4-4　シーケンス制御とモーション制御の同期 2-20
2-5　コントローラ内のデータの概要 2-21
2-6　コントローラの各状態における動作 2-22
2-6-1　コントローラの状態 2-22
2-6-2　各状態における動作 2-24
2-6-3　動作モード 2-25
2-7　Windows の状態の監視および変更 2-27
2-7-1　Windows の状態 2-27
2-7-2　Windows の状態の監視 2-27
2-7-3　Windows の状態の変更 2-28
2-7-4　Windows の状態の遷移 2-28
2-8　シャットダウン処理 2-29
2-8-1　電源断および電源ボタン押下時などの処理 2-30
2-8-2　シャットダウン処理の設定 2-31
2-8-3　シャットダウン処理の内容 2-32
2-9　コントローラリセット 2-35

第 3 章　I/O ポートとスレーブ／ユニット構成

3-1　スレーブ／ユニット構成の手順 3-2
3-2　EtherCAT スレーブ構成の作成 3-4
3-3　I/O ポートとデバイス変数 3-5
3-3-1　I/O ポート 3-5
3-3-2　I/O ポート名 3-6
3-3-3　デバイス変数 3-6
3-4　ユニットへの変数の割付 3-8
3-4-1　ユニットへの変数の割付方法 3-8
3-4-2　ユニットに割り付けた変数の使用方法 3-9
3-5　「軸」の作成と、サーボドライバ／エンコーダ入力スレーブの割付 3-11
3-5-1　概要 3-11
3-5-2　軸変数／軸グループ変数 3-11
3-5-3　「軸」および軸変数の作成と使用方法 3-13

第 4 章　コントローラ設定

4-1　コントローラ設定の概要 4-2
4-2　PLC 機能モジュールの初期設定 4-4
4-2-1　概要 4-4
4-2-2　コントローラ設定 4-4
4-2-3　タスク設定 4-7
4-3　モーション制御機能モジュールの初期設定 4-12
4-3-1　概要 4-12
4-3-2　設定方法 4-13
4-4　EtherCAT マスタ機能モジュールの初期設定 4-14

4-5 EtherNet/IP 機能モジュールの初期設定 4-15

第 5 章 タスク設計

5-1 タスク設計手順の概要 5-2

5-2 タスクの概要 5-4
5-2-1 タスクとは 5-4
5-2-2 タスク関連の命令 5-6
5-2-3 タスク関連のシステム定義変数 5-6

5-3 NY シリーズのタスクの仕様と基本動作 5-8
5-3-1 NY シリーズのタスクの仕様 5-8
5-3-2 NY シリーズのタスク分割の考え方 5-8
5-3-3 NY シリーズのタスクの基本動作 5-9
5-3-4 NY シリーズのイベントタスクの実行条件 5-15
5-3-5 NY シリーズのイベントタスクの実行タイミング 5-20
5-3-6 イベントタスク完了前に再度実行条件が成立した場合の動作 5-24

5-4 タグデータリンクサービスとシステムサービス 5-25
5-4-1 タグデータリンクサービスとシステムサービスの実行優先度と実行順序 5-26
5-4-2 タグデータリンクサービスの処理内容と実行タイミング 5-28
5-4-3 システムサービスの処理内容と実行タイミング 5-30

5-5 タスク関連の割付と設定 5-31
5-5-1 I/O リフレッシュのタスクへの割付 5-31
5-5-2 プログラムのタスクへの割付 5-37
5-5-3 タスクの設定項目 5-38

5-6 変数の同時性確保 5-40
5-6-1 タスク間での変数の値の同時性を確保する方法 5-40
5-6-2 コントローラ外部からの変数アクセス 5-46

5-7 タスク関連の異常 5-50

5-8 タスク実行状態、タスク実行時間のモニタ 5-52

5-9 タスク設計方法および入出力応答時間 5-56
5-9-1 タスク実行時間の確認 5-56
5-9-2 タスク設計の例 5-57
5-9-3 システムとしての入出力応答時間 5-58

第 6 章 プログラミング

6-1 プログラミング手順の概要 6-3

6-2 POU（プログラム構成単位） 6-5
6-2-1 POU とは 6-5
6-2-2 POU の 3 つのタイプの概要 6-6
6-2-3 プログラム、ファンクションブロック、ファンクションの相違点 6-7
6-2-4 プログラムの詳細 6-8
6-2-5 ファンクションブロック（FB）の詳細 6-9
6-2-6 ファンクション（FUN）の詳細 6-17
6-2-7 ファンクションブロック（FB）、ファンクション（FUN）共通動作 6-23
6-2-8 POU の制限事項 6-25

6-3 変数 6-27
6-3-1 変数とは 6-27
6-3-2 変数の種類 6-27
6-3-3 ユーザ定義変数の POU からみた種類 6-28
6-3-4 変数の属性 6-29
6-3-5 データ型 6-30
6-3-6 派生データ型 6-40
6-3-7 データ型の配列／範囲指定 6-50
6-3-8 変数の各属性の説明 6-58
6-3-9 状態変化に応じた変数の値の変化 6-64
6-3-10 ファンクションブロック（FB）のインスタンス 6-76
6-3-11 変数の値のモニタ 6-76
6-3-12 変数名などのプログラム関連の名称の制限 6-77

6-4 定数（リテラル）..... 6-79
6-4-1 定数とは 6-79
6-4-2 各データ型の記述形式 6-79
6-5 プログラム言語 6-84
6-5-1 プログラム言語とは 6-84
6-5-2 ラダー図言語（LD） 6-84
6-5-3 ST 言語 6-90
6-6 命令 6-128
6-6-1 命令とは 6-128
6-6-2 命令の基本知識 6-128
6-6-3 命令エラー 6-131
6-7 名前空間 6-137
6-7-1 名前空間とは 6-137
6-7-2 名前空間の仕様 6-138
6-7-3 名前空間の使用手順 6-141
6-8 ライブラリ 6-142
6-8-1 ライブラリの概要 6-142
6-8-2 ライブラリの仕様 6-143
6-8-3 ライブラリ部品の仕様 6-144
6-8-4 ライブラリの使用手順 6-145
6-9 ユーザプログラム作成上の留意事項 6-146
6-9-1 入力変数／出力変数／入出力変数が、配列指定の変数の場合 6-146
6-9-2 入力変数／出力変数／入出力変数が、構造体型変数の場合 6-146
6-9-3 マスタコントロール 6-147

第 7 章 動作確認と運転

7-1 動作確認と運転の手順の概要 7-2
7-2 オフラインデバッグ 7-3
7-2-1 シミュレーション機能の特長 7-3
7-2-2 シミュレーション機能の実行 7-3
7-2-3 シミュレーション機能の設定 7-6
7-3 実機での動作確認と運転 7-8
7-3-1 実機での動作確認から運転開始までの手順 7-8
7-3-2 プロジェクトの転送 7-9
7-3-3 I/O の配線確認 7-9
7-3-4 MC 試運転 7-9
7-3-5 ユーザプログラムの動作確認 7-10
7-3-6 本運転の開始 7-10

第 8 章 コントローラ内蔵機能

8-1 データ管理・時計・運転関連機能 8-2
8-1-1 メモリオールクリア 8-2
8-1-2 時計機能 8-2
8-2 SD メモリカード機能 8-5
8-2-1 SD メモリカード機能の一覧 8-5
8-2-2 共有フォルダと仮想 SD メモリカードの設定 8-6
8-2-3 仮想 SD メモリカードの認識と認識解除 8-6
8-2-4 Windows の共有フォルダおよびファイル仕様 8-8
8-2-5 SD メモリカード命令 8-9
8-2-6 FTP クライアント通信命令 8-10
8-2-7 FTP サーバ機能 8-10
8-2-8 Sysmac Studio からのファイル操作 8-10
8-2-9 SD メモリカード関連のシステム定義変数一覧 8-11
8-2-10 仮想 SD メモリカード内のファイルアクセスの排他制御 8-12
8-3 セキュリティ機能 8-13
8-3-1 ユーザプログラム実行用 ID 認証機能 8-14
8-3-2 ユーザプログラム復元情報無し転送機能 8-17

8-3-3 プロジェクトファイル全体のプロテクト機能 8-18
8-3-4 データプロテクト機能 8-19
8-3-5 操作権限の認証機能 8-21
8-3-6 CPU ユニットへの書込プロテクト機能 8-23
8-3-7 CPU ユニット名称機能、およびシリアル ID 機能 8-24

8-4 デバッグ機能 8-27
8-4-1 強制値リフレッシュ 8-27
8-4-2 現在値変更 8-31
8-4-3 オンラインエディット 8-33
8-4-4 データトレース 8-35
8-4-5 微分モニタ 8-40

8-5 イベントログ機能 8-45
8-5-1 概要 8-45
8-5-2 イベントログ機能の詳細 8-46
8-5-3 コントローライベント（コントローラ異常／情報） 8-51
8-5-4 ユーザイベント（ユーザ異常／情報） 8-52

8-6 イベント重要度変更機能 8-58
8-6-1 イベント重要度変更の用途 8-58
8-6-2 イベント重要度変更が可能なイベント 8-58
8-6-3 イベント重要度変更の操作方法 8-59

第 9 章　バックアップ機能

9-1 バックアップ機能の全体 9-3
9-1-1 バックアップ機能の用途 9-3
9-1-2 バックアップ機能の対象データ 9-4
9-1-3 バックアップ機能の種類 9-5
9-1-4 バックアップ機能の種類とデータグループの関係 9-7
9-1-5 バックアップ機能の対象範囲 9-8

9-2 SD メモリカードバックアップ機能 9-10
9-2-1 バックアップ（コントローラから仮想 SD メモリカード） 9-11
9-2-2 照合（コントローラと仮想 SD メモリカード間） 9-15

9-3 SD メモリカードへのバックアップ禁止機能 9-19

9-4 Sysmac Studio コントローラバックアップ機能 9-20
9-4-1 バックアップ（コントローラからパソコン） 9-21
9-4-2 リストア（パソコンからコントローラ） 9-22
9-4-3 照合（コントローラとパソコン間） 9-23

9-5 Sysmac Studio バックアップファイルのインポート / エクスポート機能 9-24

9-6 Sysmac Studio 変数／メモリのバックアップ機能 9-25
9-6-1 Sysmac Studio 変数／メモリのバックアップ機能の対象データ 9-25
9-6-2 Sysmac Studio 変数／メモリのバックアップ機能の操作方法 9-25
9-6-3 NY シリーズコントローラの形式間の互換性 9-25

9-7 EtherCAT スレーブ接続時のバックアップ機能 9-26
9-7-1 EtherCAT スレーブのバックアップ対象データ 9-26
9-7-2 コントローラの各状態でのバックアップ機能実行可否 9-27
9-7-3 EtherCAT スレーブのリストア実行条件 9-28
9-7-4 バックアップ機能の対象 EtherCAT スレーブ 9-29

9-8 EtherCAT スレーブターミナル接続時のバックアップ機能 9-31
9-8-1 EtherCAT スレーブターミナルのバックアップ対象データ 9-31
9-8-2 EtherCAT スレーブターミナルの各状態でのバックアップ機能実行可否 9-32
9-8-3 EtherCAT スレーブターミナルのリストア実行条件 9-32

9-9 バックアップ関連ファイル 9-33
9-9-1 バックアップ関連ファイルの種類 9-33
9-9-2 バックアップファイルの仕様 9-34
9-9-3 リストアコマンドファイルの仕様 9-35
9-9-4 コントローラ照合結果ファイルの仕様 9-38
9-9-5 EtherCAT 照合結果ファイルの仕様 9-39
9-9-6 EtherCAT スレーブターミナル照合結果ファイルの仕様 9-40

9-10 バックアップ機能のファイル間互換性 9-41
9-10-1 バックアップ機能間の互換性 9-41

9-10-2 NY シリーズコントローラの形式間の互換性 9-42
9-10-3 NY シリーズコントローラのユニットバージョンの互換性 9-43
9-11 バックアップ機能と同時に実行できない機能 9-44

第 10 章 通信設定

10-1 通信システムの概要 10-2
10-1-1 概要 10-3
10-2 Sysmac Studio の接続 10-4
10-2-1 オンライン接続が可能な構成 10-4
10-2-2 オンラインできない構成 10-5
10-3 他のコントローラやスレーブとの接続 10-6
10-3-1 コントローラとコントローラ間の接続構成 10-6
10-3-2 コントローラ－スレーブ間の接続構成 10-9
10-4 表示器との接続 10-10

第 11 章 実際の使用手順例

11-1 実際の使用例 11-2
11-1-1 システム構成 11-2
11-1-2 動作 11-2
11-2 操作手順例の概要 11-3
11-2-1 配線・設定 11-3
11-2-2 ソフト設計 11-3
11-2-3 Sysmac Studio でのソフトウェア設定 11-4
11-2-4 Sysmac Studio でのプログラミング 11-8
11-2-5 実機による動作確認と運転 11-9

付録

A-1 仕様 A-3
A-1-1 性能仕様 A-3
A-1-2 機能仕様 A-6
A-2 NY シリーズでのタスク実処理時間の目安の計算 A-12
A-2-1 タスク実処理時間の平均値の計算方法 A-13
A-2-2 タスク実処理時間の平均値の計算とタスク周期の設定の例 A-19
A-3 システム定義変数一覧 A-22
A-3-1 NY シリーズコントローラのシステム全体（カテゴリ名：なし） A-22
A-3-2 PLC 機能モジュール（カテゴリ名：_PLC） A-30
A-3-3 モーション制御機能モジュール（カテゴリ名：_MC） A-31
A-3-4 EtherCAT マスタ機能モジュール（カテゴリ名：_EC） A-32
A-3-5 EtherNet/IP 機能モジュール（カテゴリ名：_EIP） A-37
A-3-6 異常ステータスの各ビットの意味 A-45
A-4 システム定義変数の個別仕様 A-46
A-4-1 NY シリーズコントローラのシステム全体（カテゴリ名：なし） A-47
A-4-2 PLC 機能モジュール（カテゴリ名：_PLC） A-58
A-4-3 モーション制御機能モジュール（カテゴリ名：_MC） A-60
A-4-4 EtherCAT マスタ機能モジュール（カテゴリ名：_EC） A-62
A-4-5 EtherNet/IP 機能モジュール（カテゴリ名：_EIP） A-70
A-5 コントローラ内データの属性 A-80
A-6 変数のメモリ確保の方法 A-83
A-6-1 変数のメモリ確保の規則 A-83
A-6-2 留意するべきケース例 A-93
A-7 CX-Designer の変数テーブル登録方法 A-96
A-8 EtherCAT スレーブや軸の有効／無効切替 A-99

A-8-1 EtherCAT スレーブや軸を使用する場合のプロジェクト設定 A-99
A-8-2 EtherCAT スレーブや軸の有効／無効切替を行う命令 A-99
A-8-3 EtherCAT スレーブや軸の状態を示すシステム定義変数 A-100
A-8-4 プログラムの実行／停止指令 A-101
A-8-5 プログラムの実行／停止の確認 A-101
A-8-6 Sysmac Studio での設定 A-102
A-8-7 EtherCAT スレーブや軸の有効／無効切替のアプリケーション例 A-103

A-9 ユーザプログラムの大きさに関する制限 A-106
A-9-1 ユーザプログラムの各部品の制限 A-106
A-9-2 ユーザプログラムの各部品の数 A-108

A-10 NY シリーズのバージョン関連情報 A-110
A-10-1 ユニットバージョンと Sysmac Studio のバージョンとの関係 A-110
A-10-2 ユニットバージョンの更新によるサポート機能の追加と変更 A-111

索引

ご承諾事項

「当社商品」について特別の合意がない場合には、お客様のご購入先にかかわらず、本ご承諾事項記載の条件を適用いたします。

● 定義

本ご承諾事項中の用語の定義は次のとおりです。

- 「当社商品」:「当社」の FA システム機器、汎用制御機器、センシング機器、電子・機構部品
- 「カタログ等」:「当社商品」に関する、ベスト制御機器オムロン、電子・機構部品総合カタログ、その他のカタログ、仕様書、取扱説明書、マニュアル等であって電磁的方法で提供されるものも含みます。
- 「利用条件等」:「カタログ等」に記載の、「当社商品」の利用条件、定格、性能、動作環境、取り扱い方法、利用上の注意、禁止事項その他
- 「お客様用途」:「当社商品」のお客様におけるご利用方法であって、お客様が製造する部品、電子基板、機器、設備またはシステム等への「当社商品」の組み込み、または利用を含みます。
- 「適合性等」:「お客様用途」での「当社商品」の (a) 適合性、(b) 動作、(c) 第三者の知的財産の非侵害、(d) 法令の遵守および (e) 各種規格の遵守

● 記載事項のご注意

「カタログ等」の記載内容については次の点をご理解ください。

- 定格値および性能値は、単独試験における各条件のもとで得られた値であり、各定格値および性能値の複合条件のもとで得られる値を保証するものではありません。
- 参考データはご参考として提供するもので、その範囲で常に正常に動作することを保証するものではありません。
- 利用事例はご参考ですので、「当社」は「適合性等」について保証いたしかねます。
- 「当社」は、改善や当社都合等により、「当社商品」の生産を中止し、または「当社商品」の仕様を変更することがあります。

● ご利用にあたってのご注意

ご採用およびご利用に際しては次の点をご理解ください。

- 定格・性能ほか「利用条件等」を遵守しご利用ください。
- お客様ご自身にて「適合性等」をご確認いただき、「当社商品」のご利用の可否をご判断ください。「当社」は「適合性等」を一切保証いたしかねます。
- 「当社商品」がお客様のシステム全体の中で意図した用途に対して、適切に配電・設置されていることをお客様ご自身で、必ず事前に確認してください。
- 「当社商品」をご使用の際には、(i) 定格および性能に対し余裕のある「当社商品」のご利用、冗長設計などの安全設計、(ii)「当社商品」が故障しても、「お客様用途」の危険を最小にする安全設計、(iii) 利用者に危険を知らせるための、安全対策のシステム全体としての構築、(iv)「当社商品」および「お客様用途」の定期的な保守、の各事項を実施してください。
- 「当社」は DDoS 攻撃（分散型 DoS 攻撃）、コンピュータウイルスその他の技術的な有害プログラム、不正アクセスにより、「当社商品」、インストールされたソフトウェア、またはすべてのコンピュータ機器、コンピュータプログラム、ネットワーク、データベースが感染したとしても、そのことにより直接または間接的に生じた損失、損害その他の費用について一切責任を負わないものとします。
  お客様ご自身にて、①アンチウイルス保護、②データ入出力、③紛失データの復元、④「当社

商品」またはインストールされたソフトウェアに対するコンピュータウイルス感染防止、⑤「当社商品」に対する不正アクセス防止についての十分な措置を講じてください。

- 「当社商品」は、一般工業製品向けの汎用品として設計製造されています。従いまして、次に掲げる用途での使用は意図しておらず、お客様が「当社商品」をこれらの用途に使用される際には、「当社」は「当社商品」に対して一切保証をいたしません。ただし、次に掲げる用途であっても「当社」の意図した特別な商品用途の場合や特別の合意がある場合は除きます。
  - (a) 高い安全性が必要とされる用途（例：原子力制御設備、燃焼設備、航空・宇宙設備、鉄道設備、昇降設備、娯楽設備、医用機器、安全装置、その他生命・身体に危険が及びうる用途）
  - (b) 高い信頼性が必要な用途（例：ガス・水道・電気等の供給システム、24 時間連続運転システム、決済システムほか権利・財産を取扱う用途など）
  - (c) 厳しい条件または環境での用途（例：屋外に設置する設備、化学的汚染を被る設備、電磁的妨害を被る設備、振動・衝撃を受ける設備など）
  - (d)「カタログ等」に記載のない条件や環境での用途
- 上記の (a) から (d) に記載されている他、「本カタログ等記載の商品」は自動車（二輪車含む。以下同じ）向けではありません。自動車に搭載する用途には利用しないで下さい。自動車搭載用商品については当社営業担当者にご相談ください。

● 保証条件

「当社商品」の保証条件は次のとおりです。

- 保証期間　ご購入後 1 年間といたします。
  （ただし「カタログ等」に別途記載がある場合を除きます。）
- 保証内容　故障した「当社商品」について、以下のいずれかを「当社」の任意の判断で実施します。
  - (a) 当社保守サービス拠点における故障した「当社商品」の無償修理
    （ただし、電子・機構部品については、修理対応は行いません。）
  - (b) 故障した「当社商品」と同数の代替品の無償提供
- 保証対象外　故障の原因が次のいずれかに該当する場合は、保証いたしません。
  - (a)「当社商品」本来の使い方以外のご利用
  - (b)「利用条件等」から外れたご利用
  - (c) 本ご承諾事項「ご利用にあたってのご注意」に反するご利用
  - (d)「当社」以外による改造、修理による場合
  - (e)「当社」以外の者によるソフトウェアプログラムによる場合
  - (f)「当社」からの出荷時の科学・技術の水準では予見できなかった原因
  - (g) 上記のほか「当社」または「当社商品」以外の原因（天災等の不可抗力を含む）

● 責任の制限

本ご承諾事項に記載の保証が、「当社商品」に関する保証のすべてです。
「当社商品」に関連して生じた損害について、「当社」および「当社商品」の販売店は責任を負いません。

● 輸出管理

「当社商品」または技術資料を、輸出または非居住者に提供する場合は、安全保障貿易管理に関する日本および関係各国の法令・規制を遵守ください。お客様が法令・規則に違反する場合には、「当社商品」または技術資料をご提供できない場合があります。

安全上の注意

安全に使用していただくための表示と意味

本マニュアルでは、NY シリーズコントローラを安全に使用していただくために、注意事項を次のような表示と図記号で示しています。ここで示した注意事項は、安全に関する重大な内容を記載しています。必ず守ってください。
表示と意味は次のとおりです。

⚠ 警告	正しい取扱をしなければ、この危険のために、軽傷・中程度の傷害を負ったり万一の場合には重傷や死亡に至ったりする恐れがあります。また、同様に重大な物的損害をもたらす恐れがあります。

⚠ 注意	正しい取扱をしなければ、この危険のために、時に軽傷・中程度の傷害を負ったり、あるいは物的損害を受たりする恐れがあります。

図記号の説明



⃠記号は、禁止を意味しています。
具体的な内容は、⃠の中と文章で示します。
左図の場合は、「分解禁止」を表します。



△記号は、注意（警告を含む）を意味しています。
具体的な内容は、△の中と文章で示します。
左図の場合は、「感電注意」を表します。



△記号は、注意（警告を含む）を意味しています。
具体的な内容は、△の中と文章で示します。
左図の場合は、「一般的な注意」を表します。



●記号は、強制を意味しています。
具体的な内容は、●の中と文章で示します。
左図の場合は、「一般的な強制事項」を表します。



△記号は、注意（警告を含む）を意味しています。
具体的な内容は、△の中と文章で示します。
左図の場合は、「高温注意」を表します。

警告

⚠ 警 告

分解・落下

製品の分解や修理、改造はしないでください。故障や発火の恐れがあります。



設置時

据え付け工事の際には、必ず第 3 種接地をしてください。



製品の設置と設置後の確認は、設置される機械について十分理解されている責任者が行ってください。



フェールセーフ対策

製品の故障や外部要因による異常が発生した場合も、システム全体が安全側に働くように、外部で安全対策を施してください。異常動作により、重大な事故につながる恐れがあります。



非常停止回路、インターロック回路、リミット回路など、安全保護に関する回路は、必ず外部の制御回路で構成してください。



製品は、以下のときに、出力ユニットの全出力を OFF にし、スレーブはスレーブ側の動作に従います。

- 電源部異常が発生したとき
- CPU 異常（WDT 異常）または CPU リセットが発生したとき
- 全停止フォールトレベルのコントローラ異常が発生したとき
- 電源投入から運転モードに移行するまでの起動中
- システム初期化異常が発生したとき

これらのとき、システムが安全側に動作するよう、外部で対策を施してください。



スレーブなどの外部電源が過負荷状態になる、または短絡状態になると電圧が低下し、出力が OFF になったり、入力が取り込めなくなったりする場合があります。このときシステムが安全側に動作するよう、必要によっては外部電源電圧を監視し、制御に取り込むなど外部で対策を施してください。



製品内部のデータの異常時は、意図しない動作をすることがあります。このとき、システムが安全側に動作するよう、外部で対策を施してください。



データリンク通信やリモート I/O 通信において、通信異常や誤動作が発生しても、システム全体が安全側に動作するよう通信システムやユーザプログラムで対策を施してください。



リモート I/O において通信障害が発生した場合や全停止フォールトレベルの異常が発生した場合に、出力状態は使用される製品の仕様によります。
上記通信障害または全停止フォールトレベルの異常発生時にどのような動作仕様であるかを確認の上、安全対策を施してください。
EtherCAT スレーブは各々の「スレーブ設定」を正しく行ってください。



UPS を使用すると瞬時停電発生時でも正常動作を続けるため、瞬時停電により影響を受けた外部機器から誤った信号を受け付けてしまう可能性があります。
外部にてフェールセーフ対策を行い、かつ必要によっては外部機器側の電源電圧自体を監視し、インタロック条件として取り込むなどの対策を施してください。



タッチパネルの入力機能を、人命や重大な損害にかかわるところや、緊急の非常停止スイッチ機能として使用しないでください。



転送

Sysmac Studio からユーザプログラム、「構成／設定」のデータ、デバイス変数を転送するときは、転送先の安全を確認してから行ってください。
製品の動作モードにかかわらず、装置や機械が想定外の動作をする恐れがあります。



本運転時

作成したユーザプログラムおよび各種データ・設定値は、十分な動作確認を行った後、本運転に移行してください。



セキュリティ設定は、十分理解されている技術者が行ってください。
推奨以外の OS のセキュリティ設定は、さらにシステムのセキュリティリスクレベルを悪化させる場合があります。



BIOS 設定の変更は、十分理解されている技術者が行ってください。
製品の動作が、変わる場合があります。



タッチパネルの濡れた部分をタッチした場合、タッチパネルの入力精度が悪くなり誤入力の恐れがあります。事前にタッチパネル上の水分を拭き取ってから操作してください。



セキュリティ対策

アンチウイルス保護
制御システムに接続するパソコンへの最新の商用品質のウイルス対策ソフトの導入および保守を行ってください。



不正アクセスの防止
当社商品に対して、以下の不正アクセス防止対策を行ってください。
- 権限保有者だけを制御システムや装置へアクセス可能とする物理的統制の導入
- 制御システムや装置のネットワーク接続を最小限に抑えることによる、信頼できないデバイスからのアクセス防止
- ファイアウォールの導入による IT ネットワークからの分離（未使用通信ポートの遮断、通信ホストの制限）
- 制御システムや装置へのリモートアクセスが必要な場合における、仮想プライベートネットワーク（VPN）の使用
- 制御システムや装置へのリモートアクセスにおける多要素認証の導入
- 強固なパスワードの採用と頻繁な変更
- 制御システムや装置でUSBメモリなど外部ストレージデバイスを使用する場合の事前ウイルススキャン



データ入出力の保護
制御システムや装置への入出力データの意図せぬ改変に備えた、バックアップや範囲チェックなどの妥当性の確認を行ってください。
- データの範囲チェック
- データの改ざんや異常に備えた、バックアップによる妥当性確認、リストアの準備
- データの改ざんや異常を想定した、緊急停止や縮退運転などの安全設計



紛失データの復元
データ紛失対策として、定期的な設定データのバックアップと保守を行ってください。



グローバルアドレス経由でイントラネット環境を利用する場合、SCADA、HMI などの不正な端末や不正なサーバに接続すると、なりすまし、改ざんなどネットワークセキュリティの問題になる可能性があります。お客様ご自身で、端末へのアクセスの制限、セキュア機能を搭載した端末を使用、設置エリアの施錠などの十分な措置を講じてください。



イントラネットを構築する場合、ケーブル断線や不正ネットワーク機器の影響により通信障害が発生する可能性があります。設置エリアの施錠などの方法によって、ネットワーク機器への物理アクセスを制限するなど、十分な措置を講じてください。



SD メモリカード機能を搭載した機器を使用する場合は、リムーバブルメディアの抜き取り、不正アンマウントなどにより、第三者がリムーバブルメディアの中身のファイルやデータを不正取得、改ざん、差し替えなどを行うセキュリティリスクがあります。
お客様ご自身で、設置エリアの施錠、入室管理などの方法によって、コントローラへの物理アクセスを制限したり、リムーバブルメディアの適切な管理対策を行ったりするなどの十分な措置を講じてください。



注意

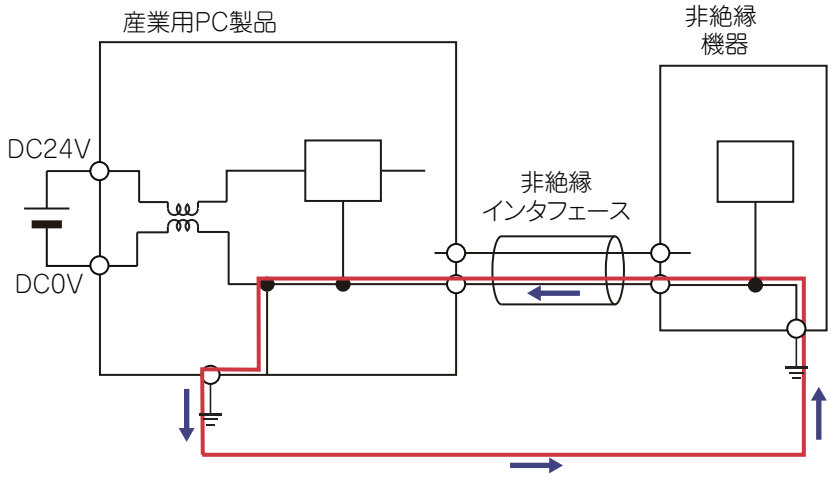
⚠ 注 意

設置時

PCIe カードを着脱する場合、金属薄板フレームのタブの鋭利な部分に触れないよう、カードクリップの側面を確実につかんでください。けがのおそれがあります。

配線時

製品の電源は、非絶縁の DC 電源です。内部の 0V と FG は、直結しています。非絶縁の外部デバイスを接続する場合は、外部で安全対策を施してください。通信異常や製品の故障の可能性があります。



24V 側接地は、絶対に行わないでください。24V-0V 間が短絡します。

オンラインエディット

入出力タイミングが乱れても影響のないことを確認してからオンラインエディットをしてください。オンラインエディットすると、タスク実行時間がタスク周期を上回り、外部との入出力が更新されず、入力信号を読み取れなかったり、出力タイミングが乱れたりする場合があります。

本運転時

複数のタッチパネルが接続されたシステムを使用する場合は、複数ユーザの同時操作によりシステムが想定外の動作を起こさないようにしてください。



安全上の要点

分解、落下、装着時、設置時、保管時

- 製品を落下させたり、異常な振動・衝撃を与えたりしないでください。製品の故障、焼損の可能性があります。
- 開封時に外観チェックを行い、損傷のないことを確認してください。また、製品を軽く振って、異音がないことを確認してください。
- 必ず関連マニュアルに指定された機器をご使用ください。
- 製品の仕様の範囲外の機器を使用しないでください。システムの故障や誤動作の原因になります。
- 設置方向とその周囲温度は、仕様範囲内になるように設置してください。仕様の範囲外で使用すると、製品が正常動作しない恐れがあります。
- 周辺機器を製品に接続する場合は、ノイズ、静電気遮蔽対策を十分に行った上で設置してください。
- 必ず CFast カードスロットカバーを使用して、CFast カードをしっかり挿入してください。CFast カードを指で挿入しようとすると、CFast カードベイ周辺の鋭いエッジによって指にけがをする恐れがあります。
- 製品は必ず制御盤内に設置してください。
- 取り付けできるパネル板の厚さは、1.6 ～ 6.0mm です。防水および防じん能力を保つため、取り付け金具は、0.6N·m のトルクで均等に締めてください。それ以外のトルクで締めたり、均等に締めていなかったりした場合、パネルが変形する恐れがあります。
  汚れやゆがみなく、取り付け強度が十分に保てるパネルを使用してください。
- パネル加工時は、金属クズが製品内部へ侵入しないよう気をつけてください。製品内に配線クズや切粉などの異物が入らないようにしてください。焼損、故障、誤動作の原因となります。特に施工時は覆いを付けるなど対策を行ってください。

配線時

- コネクタの配線及び接続は本マニュアルにしたがって正しく行ってください。配線の順番、コネクタの向きを十分確認してから通電してください。
- コネクタ、ケーブル、PCIe カード、ストレージデバイスなどロック機構のあるものは、必ずロックしていることを確認してからご使用ください。運転中に外れる可能性があります。
- パソコンと製品を接続する際には、パソコンの電源プラグを AC コンセントから抜いておいてください。また、パソコンに FG 端子がある場合は、製品の機能接地端子と同電位になるように接続してください。
  パソコンと製品との間に電位差が発生し、故障や誤動作の原因になる可能性があります。
- ケーブルを無理に曲げたり、引っ張ったりしないでください。
  ケーブルのコード部に重いものを載せないでください。断線する恐れがあります。
- 電圧低下や焼損を防ぐために、十分な大きさの電源の電線を使用してください。電源の配線は、電線の電流容量に注意して行ってください。発熱や発火の可能性があります。端子から端子へ渡り線で配線すると、それぞれの電線には加算された電流が流れます。渡り線で配線するときは、すべての電線の電流容量に注意してください。
- 取り付け金具のねじ、ケーブルのねじは、本マニュアルまたは参照マニュアルで指定したトルクで締め付けてください。ねじが緩むと発火および誤動作の恐れがあります。
- 配線は、圧着端子を付けてください。
- 以下の点を守り、断線しないようご注意ください。
  - 被覆をはがす時、芯線を傷つけないようにしてください。
  - 芯線はよらずに結線してください。
  - 芯線はハンダ上げせずに結線してください。振動により切断する場合があります。
- NY Monitor Link の接続は、本マニュアルまたは参照マニュアルに記載されているケーブル種類や接続方法にしたがってください。通信不良になる恐れがあります。

タスク設計時

- イベントタスクの実行条件が変数の条件式一致で、条件式に以下の変数を指定する場合、条件一致の時にイベントタスクが起動しない、もしくは、条件不一致の時にイベントタスクが起動されることがあります。
  - モーション制御システム変数を除く、データサイズが 16 ビット以上の構造体のメンバ
  - データサイズが 16 ビット以上の配列の要素

  イベントタスクの実行条件については、📖『NY シリーズ IPC マシンコントローラ 産業用パネル型 PC ／産業用ボックス型 PC ユーザーズマニュアル ソフトウェア編 (SBCA-436)』を参照してください。

電源設計時、電源投入時、電源 OFF 時

- システム内で使用する電源は、本マニュアルに記載されている電源の定格範囲で使用してください。
- 耐電圧試験は実施しないでください。
- 製品の電断時のデータ破損やシステムファイルの不整合を防ぐために、UPS を使用してください。さらに、誤操作などによるデータ損失やシステムファイルの不整合を防ぐために、計画的にファイルのバックアップをお取りください。
- 製品を正常にシャットダウンするため、適切なリビジョンナンバーのオムロン製 S8BA UPS を使用してください。
- 電源投入から運転モードに移行するまで10～20秒程度時間がかかります。この間の出力はスレーブやユニットの仕様に従います。外部機器が誤動作しないようにフェールセーフ回路を組んでください。
- 製品とモニタを接続した後、電源を ON してください。
- 電源投入の前に、電源や電源の配線を確認してください。誤った電源や配線は、故障や発火の恐れがあります。
- 人体または導電性があるものがタッチパネル表面に触れているときに製品の電源を投入しないでください。タッチパネル機能の使用が不可能となります。タッチパネルの機能を回復するには、導電性があるものを取り除いて電源を再投入してください。
- 次のことを行うときは、電源を OFF にしてください。
  - PCIe カードを着脱するとき
  - ケーブルを接続、配線するとき
  - システムの配線をするとき
  - コネクタを着脱するとき
  - ストレージデバイスを着脱するとき
  - バッテリの交換
  - ファンユニットの交換

本運転時

- 不正アクセスの防止を考慮した OS のパスワードを設定してください。
- OS のユーザ名やパスワードは、忘れないようにしてください。製品を操作するためには、パスワードが必要です。
- 本運転に移行する前に、適切なソフトウェアがインストールされ、設定されているか確認してください。誤動作の恐れがあります。
- 最新のブラウザが、インストールされていることを確認してください。
- 最新のファイヤーウォールが、インストールされていることを確認してください。
- 装置の OS 環境が、外部からのウイルスや不正アクセスから保護されていることを確認してください。
- 最新のウイルス定義ファイルが、インストールされていることを確認してください。
- 電源が ON している時に、ファンカバーを外さないでください。動作中のファンに触れるとけがの恐れがあります。
- 仮想メモリの設定は、システムの性能に影響する場合があります。ソフトウェアのインストールやアップデートした後にページファイルは、無効に設定してください。
- 配線と設定を正しく行い、UPS からのシャットダウンが実行できることを確認してください。

操作時

- 次の操作は設備に影響がないかを確認したうえで行ってください。
  - 製品の動作モードの変更（電源投入時の動作モード設定含む）
  - ユーザプログラム、設定の変更
  - 設定値 / 現在値の変更
  - 強制値リフレッシュ
- 製品が、USB デバイスや SD カードにアクセス中に、以下の操作をしないでください。
  - 製品の電源を OFF する
  - 電源ボタンを押す
  - USB デバイスや SD カードを抜く
- PCIe コネクタに接続するボードと EtherCAT スレーブのように異なる機能モジュールを連携させて装置を制御する場合は、片方の機能モジュールが停止しても装置に影響を与えないように、ユーザプログラムおよび外部にて対策を行ってください。それぞれの機能モジュールに部分停止フォールトレベルの異常が発生した場合、該当する出力はスレーブや出力ユニットの仕様に従います。
- 通電中や電源 OFF 直後は、ファンユニットの交換を行わないでください。交換を行った場合、高温部によるやけどや回転部によるけがの恐れがあります。
- 電源ボタンを数秒間、長押しをすると、製品は、強制シャットダウンします。それによって、データが破損する場合があります。計画的にファイルのバックアップをお取りください。
- 通電中や電源 OFF 直後は、製品の筐体にふれないでください。やけどの恐れがあります。
- システムの安全を確認してから、タッチパネルを操作してください。
- タッチパネルを高速に連続して押すと、入力が取り込めない場合があります。ひとつの入力を確認した後に、次の入力操作に移ってください。
- バックライトが消灯しているときや、画面が無表示のとき、視認できないほど暗いときは、タッチパネルを不用意に押さないでください。
- 固い物体、または先がとがった物などで表示部に触れたり、擦ったりしないでください。スクリーンの表面が破損する恐れがあります。
- 拡張モードで複数の画面を使うシステムでは、1 つの画面のビデオ信号が何らかの原因で中断した場合、その画面上のすべてのウィンドウがメインディスプレイに移動します。そのような場合でも問題が起きないよう適切に処理してください。

- デバイス出力保持設定を有効にしている場合で、次の操作をするときは、接続先の設備の安全を十分に確認してください。出力は直前の状態が保持されるため、設備が予期しない動作をするおそれがあります。
  - 製品の動作モードの変更
  - ダウンロード時

通信一般

- 不適切なデータリンクテーブルが設定されていると、設備が予期しない動作をする恐れがあります。また適切なデータリンクテーブルが設定されている場合でも、設備に影響がないことを確認してから、転送を実行してください。転送後に自動的にデータリンクを開始します。
- 装置のネットワークとオフィスネットワークは、分離してください。誤動作の恐れがあります。

EtherNet/IP 通信

- EtherNet/IP は通信距離および接続台数／接続方法を仕様の範囲内でご使用ください。また、EtherCAT など他のネットワークに接続しないでください。過負荷によるネットワークのダウンや誤動作の可能性があります。
- 内蔵 EtherNet/IP の設定（IP アドレスやタグデータリンクの設定）を周辺ツールから転送すると、設定対象すべての EtherNet/IP（ノード）にリセットがかかります。これは、設定内容を読み込んで有効にするためです。リセットがかかっても設備に影響がないことを確認してから、転送を実行してください。
- リピータハブを使用して EtherNet/IP のタグデータリンク通信（サイクリック通信）を行うと、ネットワークの通信負荷が高まるため、コリジョン（衝突）が多数発生し、安定した通信ができなくなります。タグデータリンクを利用するネットワークでは、必ずスイッチングハブを使用してください。

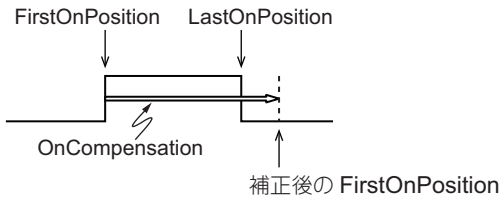
EtherCAT 通信

- EtherCAT のマスタとスレーブのリビジョンの組み合わせによっては誤動作や意図しない動きをする可能性があります。ネットワーク設定のリビジョンチェックの機能を無効にして使用する場合は、Sysmac Studioを使ってマスタの設定上のスレーブリビジョンと実際のスレーブのリビジョンを確認し、スレーブのマニュアル等より機能の互換性を確認した上でご使用ください。なお、実際のスレーブのリビジョンは Sysmac Studio や銘板で確認することができます。
- 「縮退運転設定」を「停止する」と設定している場合、EtherCAT 通信異常がいずれかのスレーブで発生すれば、全スレーブとプロセスデータ通信を停止します。このときのサーボドライバの動作は、サーボドライバの仕様に従います。「縮退運転設定」については、装置の異常発生時に安全な挙動をとるように十分考慮してください。
- ノイズが発生したときや、EtherCAT スレーブからネットワークケーブルを抜いたときに、通信中のフレームが消失することがあります。フレームが消失すると、スレーブの入出力データが伝わらなくなり、意図しない動作をする恐れがあります。スレーブの出力はスレーブの仕様に従います。詳細は、各スレーブのマニュアルを参照してください。ノイズ対策やスレーブの交換が必要な場合は、以下の処置を行ってください。
  - システム定義変数の入力データ無効をインタロック条件としてユーザプログラム作成してください。
  - EtherCAT マスタの設定項目の「PDO 通信連続タイムアウト検出回数」を、必ず 2 回以上に設定してください。詳細は、🕮『NY シリーズ 産業用パネル型 PC ／産業用ボックス型 PC ユーザーズマニュアル 内蔵 EtherCAT® ポート ユーザーズマニュアル(SBCD-368)』を参照してください。
- 電源投入直後、EtherCAT の通信が確立していない場合があります。システム定義変数を用いて通信が確立したことを確認してから、制御に使用するようにプログラミングしてください。

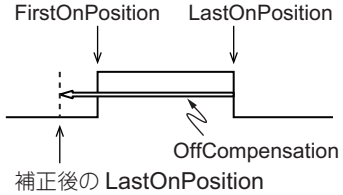
- EtherCAT スレーブを「離脱」や「無効」とした場合、該当スレーブ以降のスレーブとの通信も停止し、出力が制御できなくなります。設備に影響が無いことを確認してから「離脱」、「無効」の指示をしてください。
- 汎用 Ethernet ハブ、リピータハブは使用しないでください。使用した場合、全停止フォールトレベルなどの異常が発生する恐れがあります。
- 必ず指定されている EtherCAT スレーブケーブルを使用してください。指定外のケーブルを使用した場合、EtherCAT マスタまたは EtherCAT スレーブが異常を検出し、以下のような現象が発生する可能性があります。
  - プロセスデータ通信のリフレッシュが、継続的に実行されない。
  - プロセスデータ通信のリフレッシュが、設定周期内に完了しない。

モーション制御

- MC 試運転中にツール - 製品間の通信が途絶えた場合は、モータを強制的に停止させます。通信ケーブルの抜けには注意し、設備への影響がないことを確認してから MC 試運転を実行してください。
- サーボドライバの外部入力に割り付けられた正転側駆動禁止入力（POT）、逆転側駆動禁止入力（NOT）、原点近傍入力（DEC）をそれぞれ正転側限界入力、逆転側限界入力、原点近傍入力として使用します。これらの入力信号に関しては、必ず制御周期以上の信号幅を確保するようにしてください。入力信号が制御周期未満の幅となる場合、入力信号を検出できず、正常な動作ができなくなります。
- 製品のユーザプログラムや Sysmac Studio でカムデータの値を変更した場合、必ずカムテーブル保存命令を実行してください。行わない場合は、次の電源投入時にカムデータの値が変更前の状態に戻ってしまい、装置が意図しない動作をする可能性があります。
- MC 試運転を行う場合、軸番号をよく確認してから行ってください。
- NX_AryDOutTimeStamp（タイムスタンプデジタル出力配列書込）命令は、MC_DigitalCamSwitch（デジタルカムスイッチ有効）命令の InOperation（作動中）が TRUE になっていることを確認してから使用してください。
- MC_DigitalCamSwitch（デジタルカムスイッチ有効）命令は、必ず軸を一定速度で使用してください。
  カウントモードをロータリモードにして、OnCompensation（ON 時間補正）または OffCompensation（OFF 時間補正）を使用し、軸速度が急変すると、以下のような動作となります。
  - OnCompensation（ON 時間補正）または OffCompensation（OFF 時間補正）の値が、軸の半回転以上に相当する時間になる場合は、InOperation（作動中）が FALSE となります。
  - OnCompensation（ON 時間補正）の値が、LastOnPosition（ON 終了位置）を超える場合は、出力タイミングが不定となります。



  - OffCompensation（OFF 時間補正）の値が、FirstOnPosition（ON 開始位置）を超える場合は、出力タイミングが不定となります。



バックアップ時

- 変数の現在値のバックアップは、保持変数が更新されない状態で実施することを推奨します。
  保持変数の値が更新される状態でバックアップすると、正しく保存できないことがあります。
  バックアップ機能、および変数の現在値のバックアップについては、『NY シリーズ IPC マシンコントローラ 産業用パネル型 PC ／産業用ボックス型 PC ユーザーズマニュアル ソフトウェア編(SBCA-436)』を参照してください。

リストア時

- 製品は絶対値エンコーダの情報である「絶対値エンコーダ原点位置オフセット」を不揮発性メモリによって保持しています。以下のいずれかの条件を満たす場合は、絶対値エンコーダ原点位置オフセットをリストア対象からはずしてリストアを実施してください。その後、新たに絶対値エンコーダ原点設定を行ってください。絶対値エンコーダ原点設定をしなかった場合、制御対象が予期しない動作をするおそれがあります。
  - サーボモータとドライバの個体が、バックアップを作成したときの個体と異なる。
  - バックアップを作成した以降に、絶対値エンコーダのセットアップを実行した。
  - 絶対値エンコーダの絶対値データが失われている。
- スレーブやユニットには、設定データのすべてまたは一部がバックアップ、リストア、照合の対象外であるものが存在します。また、無効設定されたスレーブやユニットも、バックアップ、リストア、照合の対象外です。リストア機能の使用時は、十分な動作確認を行った後に、本運転に移行してください。

バッテリ交換時

以下は交換可能なバッテリを備えた商品に対して適用します。

- 床に落下させるなどして強い衝撃をあたえてしまったバッテリは、液漏れする恐れがありますので絶対に使用しないでください。
- バッテリの交換は熟練した技術者によって行われることが、UL 規格で定められています。交換作業は熟練した技術者がご担当ください。
- バッテリは液漏れ、破裂、発熱、発火などの恐れがありますので、＋、－の短絡、充電、分解、加熱、火への投入、強い衝撃を与えることなどは絶対に行わないでください。

ファンユニット交換時

- 無通電での保管が6か月を超えたファンユニットは正常に動作することを確認した上で、使用してください。

製品交換時

- 製品を交換するときは、運転再開に必要なユーザプログラム、「構成／設定」の設定データ、変数を、交換した製品、および外部に接続される機器に転送してから、運転を開始してください。タグデータリンクの設定、ルーチングテーブルも製品に保存されているため転送を行ってください。

清掃と点検、廃棄時

- 製品の掃除に腐食性の薬品を使用しないでください。故障や誤動作の原因になります。
- 油や水がかかる環境で使用する場合は、定期的に設置状態を確認してください。
- ゴムパッキンは使用環境によっては、劣化・収縮および硬化するため定期的な点検をお願いします。
- 製品および不要になったバッテリの廃棄については、地方自治体により規制を受ける場合があります。それぞれの自治体規制に従って廃棄してください。



- リチウム一次電池（過塩素酸含有量が 6ppb 以上）を組み込んだ製品が米国・カリフォルニア州へ輸出・経由される場合、つぎの表示が義務化されています。
  Perchlorate Material - special handling may apply.
  See www.dtsc.ca.gov/hazardouswaste/perchlorate
- 製品にはリチウム一次電池（過塩素酸含有量が 6ppb 以上）を搭載しています。製品が組み込まれた最終製品を米国・カリフォルニア州へ輸出する場合は、個装箱、輸送箱などに上記表示を行ってください。

使用上の注意

保管時、設置時、装着時

- 次のような環境に設置や保管しないでください。運転停止、誤動作する可能性があります。
  - 日光が直接当たる場所
  - 周囲温度や相対湿度が仕様値の範囲を超える場所
  - 温度変化が急激で結露するような場所
  - 腐食性ガス、可燃性ガスのある場所
  - ちり、ほこり、塩分、鉄粉が多い場所
  - 水、油、薬品などの飛まつがかかる場所
  - 製品に直接振動や衝撃が伝わる場所
  - 屋外で風雨が直接当たる場所
  - 紫外線の強い場所
- 発熱体の近傍への設置を避け、通風の確保など正しく設置してください。
- 次のような場所で使用する際は、遮蔽対策を十分に行ってください。
  - 強い高周波ノイズを発生する機器の近く
  - 静電気などによるノイズが発生する場所
  - 強い電界や磁界が生じる場所
  - 放射線を被曝する恐れのある場所
  - 電源線が近くを通る場所
- 接地された金属に触るなどして人体の静電気を放電させてから、製品に触れてください。
- 周辺デバイスは、正しく装着してください。焼損、故障、誤動作の原因になります。
- PCIe コネクタに接続するボードは、そのボードや関連するソフトウェアにより起動不良や動作不良が発生する場合があります。ボート接続前のバックアップの取得と、十分な動作確認を行ってください。
- バックライトには寿命があります。寿命を超えると輝度が低下します。定期的に確認し、必要により交換してください。

配線時

- 製品は、定格電源電圧で使用してください。
- 製品内に配線クズや切粉などが入らないようにしてください。焼損、故障、誤動作の原因となります。特に施工時は覆いを付けるなど対策を行ってください。
- 利用可能なソフトウェアの確認は、利用可能なソフトウェアについて十分理解されている責任者が行ってください。
- 仕様範囲より長いケーブルを使用しないでください。通信不良になる恐れがあります。
- DC 電源コネクタに AC 電源を接続しないでください。

本運転時、操作時

- 製品に搭載される OS のアップデートや、周辺デバイスのドライバ更新後に、製品は更新前と異なる動作をする可能性があります。十分な動作確認を行ってください。
- 製品が動作中に、ファンが動作していることを確認してください。
- ストレージデバイス、SD メモリカード、電源ボタン、ファンユニット、バッテリには、寿命があります。寿命を超えると製品の故障や誤動作の恐れがあります。
- ファンのステータスを監視してください。ファンの寿命を超えて使用すると、ファンの回転数低下の警告メッセージが表示され、製品が、過熱し、停止する恐れがあります。
- バッテリの警告メッセージを監視してください。バッテリがない状態が続くと時計の動作を保持できません。
- パスワード変更するときは、書き込みが終了するまでリセットしたり電源を切ったりしないでください。パスワードの保存に失敗するとプロジェクトが機能しなくなる恐れがあります。
- EtherCAT マスタ機能モジュールに対して、部分停止フォールトレベル以上のコントローラ異常の解除をするときは、接続先の設備の安全を十分に確認してください。
  部分停止フォールトレベル以上のコントローラ異常によりオペレーショナル状態以外（出力無効）に遷移しているスレーブはオペレーショナル状態に遷移し、出力が有効になります。
  異常の全解除やスレーブのリスタートを行うときは、EtherCAT マスタ機能モジュールの部分停止フォールトレベルのコントローラ異常が発生していないか確認してください。
- 製品のユニットバージョンにより、搭載している機能が異なります。バージョンアップで追加された機能を使用するときは、対応しているバージョンの Sysmac Studio を使う必要があります。製品のユニットバージョンの種類と Sysmac Studio のバージョンとの関係、およびユニットバージョンによるサポート機能一覧については、「NY シリーズのバージョン関連情報」を参照してください。
- ツールから各種データを転送中にケーブルを抜いたり、製品の電源を OFF しないでください。データの転送が完了できず、誤動作の恐れがあります。
- Rescue Disk Utility を用いて、製品の復元データを作成してください。これによりストレージデバイス内の設定を復旧することができます。
- タッチパネルは同時に最大５点の入力が可能です。同時に５点を超えた入力を検出することはできません。
- 静電容量方式のタッチパネルは、タッチパネルの表面に導電物が触れると入力されます。不意の接触に注意してください。誤入力の恐れがあります。
- 一部の手袋は装着していてもタッチパネルを操作することができます。本運転の前に、その手袋を装着してタッチパネルを正常に操作できることを確認してください。
- 製品の動作モードを変更するときや、ダウンロードを実行するときは、デバイス出力保持設定を確認してから行ってください。

異常処理

- 異常ステータス取得命令の検出結果を用いるアプリケーションでは、検出時に設備に与える影響を考慮のうえ、プログラミングしてください。たとえば、動作中にバッテリ交換を行った場合など軽微な異常を検出しても、プログラムしだいで動作に影響を与える可能性があります。
- コントローラ異常の重要度を変更した場合、変更によって異常発生時の出力状態が変わる場合があります。安全を確認の上、ご使用ください。

デバッグ時

- 以下の変数をデータトレースのサンプリング対象とする場合、正しいデータをサンプリングできないことがあります。
  - モーション制御システム変数を除く、データサイズが 16 ビット以上の構造体のメンバ
  - データサイズが 16 ビット以上の配列の要素

  データトレースについては、『NYシリーズ IPCマシンコントローラ 産業用パネル型PC／産業用ボックス型 PC ユーザーズマニュアル ソフトウェア編 (SBCA-436)』を参照してください。

リストア時

- 「リストアコマンドファイル」を編集する場合は、選択可能なデータグループの「yes/no」の記述だけにしてください。それ以外の記述を編集してリストアをすると、コントローラが意図しない動作をする場合があります。

タスク設定

- タスク周期オーバーが発生した場合、タスクの周期内に収まるようにユーザプログラムを作成するか、またはタスクの周期を設定してください。

モーション制御

- MC 試運転を開始する際には、動作パラメータが正しいことを確認してからMC試運転動作を開始してください。
- MC 試運転中にモーション設定のダウンロードは行わないでください。

EtherCAT 通信

- コントローラとサーボドライバ間のEtherCAT通信に異常が起こった場合に、動作を停止するようにサーボドライバを設定してください。
- 運転中に EtherCAT スレーブケーブルを抜く場合は、EtherCAT スレーブを離脱する、もしくは、後続する EtherCAT スレーブも含めて無効とした後で抜いてください。
- EtherCAT マスタ機能モジュールでの「ネットワーク構成照合異常」、「プロセスデータ通信異常」、「リンクオフ異常」を解除する場合は、復旧対象のすべてのスレーブが加入状態であることを事前に確認してください。加入していないスレーブがあると、EtherCAT マスタ機能モジュールが、指定と異なるノードアドレスのスレーブにアクセスしたり、正しく異常解除されなかったりする可能性があります。
- EtherCAT は通信距離および接続台数／接続方法を仕様の範囲内でご使用ください。
  また、EtherNet/IP や一般的な構内 LAN など他のネットワークに接続しないでください。過負荷によるネットワークのダウンや誤動作の可能性があります。
- ユーザプログラムを転送すると、製品にリスタートが発生し、EtherCAT スレーブとの通信が途絶えます。その間のスレーブ出力は、スレーブの仕様に従います。EtherCAT ネットワーク構成により、通信が途絶える時間が異なります。ユーザプログラム転送時は装置に影響を与えないことを確認してから行ってください。

バッテリ交換時

以下は交換可能なバッテリを備えた商品に対して適用します。

- 長期間放置された製品はバッテリを交換してから、電源を投入してください。バッテリ交換後、一度も電源投入しない状態で、再び放置しておくと、バッテリ寿命が短くなることがあります。
- バッテリ交換時は指定のバッテリセットをご使用ください。
- バッテリの交換前には5分以上通電し、電源OFF後5分以内に新しいバッテリと交換してください。5 分以上バッテリがない状態が続くと時計情報が消滅することがあります。バッテリの交換後は時計情報を確認してください。

SD メモリカード

- SD メモリカードは奥まで差し込んでください。

清掃と点検

- ベンジン、シンナーなどの揮発性の溶剤や化学雑巾などは使用しないでください。
- 水を使用して清掃するときは、製品をシャットダウンするかタッチパネルの入力を無効にしてください。

法規と規格

海外でのご使用

本製品のうち、外国為替および外国貿易管理法に定める輸出許可、承認対象貨物（または技術）に該当するものを輸出（または非居住者に提供）する場合は、同法に基づく輸出許可、承認（または役務取引許可）が必要です。

EU 指令への適合

適合指令

- EMC 指令

適合の考え方

● EMC 指令

オムロンの商品は、各種機械、製造装置に組み込まれ使用される電気機器であるため、組み込んだ機械・装置がより容易に EMC 規格に適合できるように、商品自身の関連する EMC 規格（*）の適合を図っています。
しかし、お客様の機械・装置は様々であり、かつ EMC の性能は EU 指令適合商品を組み込んだ機器・制御盤の構成、配線状態、配置状態などにより変化するので、お客様の使用状態での適合性は確認できません。したがって、機械・装置全体での最終的な EMC 適合性の確認を、お客様自身で実施していただくようにお願いします。

*EMC（Electro-Magnetic Compatibility：電磁環境両立性）関連規格のうち、
EMS（Electro-Magnetic Susceptibility：電磁感受性）に関しては EN 61131-2
EMI（Electro-Magnetic Interference：電磁妨害）に関しては EN 61131-2
また、Radiated emission は 10m 法によります。

● EU 指令への適合

NY シリーズは、EU 指令に適合しています。しかし、お客様の機械・装置を EU 指令に適合させるにあたり、以下の注意が必要です。

- NY シリーズは、必ず制御盤内に設置してください。
- DC 電源ユニットおよび I/O ユニットに接続する DC 電源は、SELV 仕様の電源を使用してください。
- NY シリーズの EU 指令適合商品は、EMI に関して共通エミッション規格に適合していますが、特に Radiated emission（10m 法）に関しては、ご使用になる制御盤の構成、接続される他の機器との関係、配線等により変化する事があります。
  したがって、EU 指令適合品である NY シリーズをご使用の場合でも、お客様にて機械・装置全体で EU 指令適合性を確認・対応していただく必要があります。

ソフトウェアのライセンスと著作権

本製品にはサードパーティ製のソフトウェアが組み込まれています。このソフトウェアに関連するライセンスと著作権については、http://www.fa.omron.co.jp/nj_info_j/ をご覧ください。

バージョン

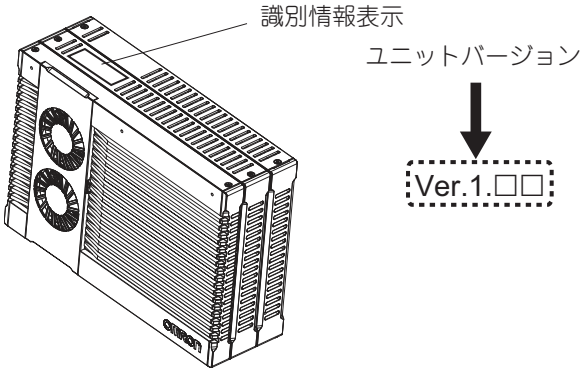
NY シリーズおよび各 EtherCAT スレーブは、ハードウェアやソフトウェアが、ハードウェアリビジョンやユニットバージョンという改変番号でバージョン管理されています。ハードウェアまたはソフトウェアに仕様変更があるごとに、ハードウェアリビジョンまたはユニットバージョンが更新されます。したがって、同じ形式のユニットや EtherCAT スレーブでも、ハードウェアリビジョンまたはユニットバージョンが異なると搭載機能や性能が異なります。

バージョンの確認方法

バージョンは、識別情報表示または Sysmac Studio で確認できます。

識別情報表示による確認

製品背面の識別情報表示で、バージョンが確認できます。
NY シリーズコントローラ形 NY5□2-□□□□ の識別情報表示は、下図のようになります。



Sysmac Studio による確認方法

Sysmac Studio で、バージョンが確認できます。ユニットと EtherCAT スレーブでは、確認方法が異なります。

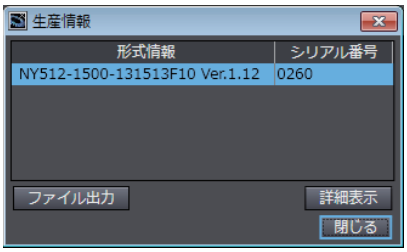
● NY シリーズコントローラでのバージョン確認方法

ユニットのバージョンは、オンライン状態の［生産情報］で確認できます。ただし、バージョンが確認できるユニットは、コントローラだけです。

1 **マルチビューエクスプローラで［構成・設定］–［CPU・増設ラック］–［CPU ラック］を右クリックし、［生産情報表示］を選択します。**
［生産情報］ダイアログボックスが表示されます。

● 生産情報の表示内容の切り替え

1 **［生産情報］ダイアログボックスの右下の、［簡易表示］または［詳細表示］を選択します。**
［生産情報］の簡易表示と詳細表示が、切り替わります。



簡易表示



詳細表示

表示内容は、簡易表示と詳細表示とで異なります。詳細表示では、ユニットバージョン、ハードウェアリビジョン、および各種のバージョンが表示されます。簡易表示では、ユニットバージョンだけが表示されます。

● EtherCAT スレーブのバージョン確認方法

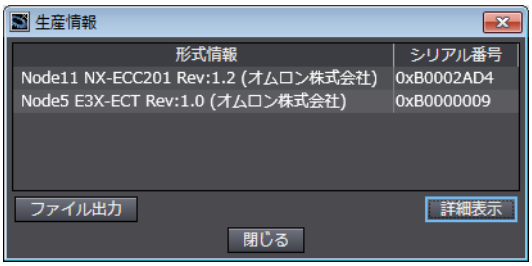
EtherCAT スレーブのバージョンは、オンライン状態の〔 生産情報 〕で確認することができます。
確認方法は、以下のとおりです。

1 **マルチビューエクスプローラで［構成・設定］–［EtherCAT］をダブルクリックします。または、［構成・設定］–［EtherCAT］を右クリックして［編集］を選択します。**
EtherCAT 構成編集画面が表示されます。

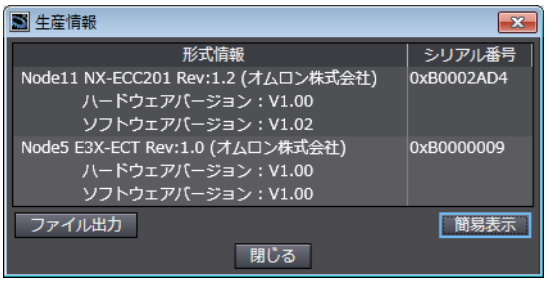
2 **EtherCAT 構成編集画面でマスタを右クリックして［生産情報表示］を選択します。**
生産情報ダイアログボックスが表示されます。
ユニットバージョンは、「Rev」を付けて表示されます。

● 生産情報の表示内容の切り替え

1 **[生産情報] ダイアログボックスの右下の、[簡易表示] または [詳細表示] を選択します。**
〔 生産情報 〕の簡易表示と詳細表示が、切り替わります。



簡易表示



詳細表示

関連マニュアル

本マニュアルに関連するマニュアルは、下表のとおりです。あわせてご覧ください。

マニュアル名称	Man.No.	形式	用途	内容
NY シリーズ IPC マシンコントローラ 産業用パネル型 PC ユーザーズマニュアル ハードウェア編	SBCA-435	形 NY532-□□□□	NY シリーズ産業用パネル型 PC の概要／設計／取付／保守などの基本的な仕様について知りたいとき。 おもにハードウェアに関する情報。	NY シリーズのシステム全体概要、および産業用パネル型 PC に関して、以下の内容を説明します。 • 特長やシステム構成 • 概要 • 各部の名称と機能 • 一般仕様 • 設置と配線 • 保守点検
NY シリーズ IPC マシンコントローラ 産業用ボックス型 PC ユーザーズマニュアル ハードウェア編	SBCA-434	形 NY512-□□□□	NY シリーズ産業用ボックス型 PC の概要／設計／取付／保守などの基本的な仕様について知りたいとき。 おもにハードウェアに関する情報。	NY シリーズのシステム全体概要、および産業用ボックス型 PC に関して、以下の内容を説明します。 • 特長やシステム構成 • 概要 • 各部の名称と機能 • 一般仕様 • 設置と配線 • 保守点検
NY シリーズ IPC マシンコントローラ 産業用パネル型 PC ／産業用ボックス型 PC ユーザーズマニュアル セットアップ編	SBCA-441	形 NY532-□□□□ 形 NY512-□□□□	NY シリーズ産業用 PC の初期設定やコントローラの使用準備について知りたいとき。	NY シリーズのシステム全体概要に関して、以下の内容を説明します。 • 2 つの OS のしくみ • 初期設定 • Industrial PC Support Utility • NYCompolet • Industrial PC API • バックアップとリカバリ
NY シリーズ IPC マシンコントローラ 産業用パネル型 PC ／産業用ボックス型 PC ユーザーズマニュアル ソフトウェア編	SBCA-436	形 NY532-□□□□ 形 NY512-□□□□	NY シリーズ産業用 PC のコントローラ機能のプログラミング／システムの立ち上げについて知りたいとき。	NY シリーズのコントローラ機能に関して、以下の内容を説明します。 • コントローラの動作 • コントローラの機能 • コントローラの設定 • IEC 61131-3ベースの言語仕様とプログラミング
NY シリーズ コマンドリファレンス マニュアル 基本編	SBCA-437	形 NY532-□□□□ 形 NY512-□□□□	NY シリーズ産業用 PC の基本命令仕様の詳細について知りたいとき。	各命令（IEC 61131-3 仕様）の詳細を説明します。
NY シリーズ IPC マシンコントローラ 産業用パネル型 PC ／産業用ボックス型 PC ユーザーズマニュアル モーション制御編	SBCE-379	形 NY532-□□□□ 形 NY512-□□□□	NY シリーズ産業用 PC のモーション制御の設定やプログラミングの考え方について知りたいとき。	モーション制御のためのコントローラの設定や動作、プログラミングの考え方について説明します。
NY シリーズ コマンドリファレンス マニュアル モーション編	SBCE-380	形 NY532-□□□□ 形 NY512-□□□□	NY シリーズ産業用 PC のモーション命令仕様の詳細について知りたいとき。	各モーション制御命令の詳細を説明します。
NY シリーズ IPC マシンコントローラ 産業用パネル型 PC ／産業用ボックス型 PC ユーザーズマニュアル 内蔵 EtherCAT® ポート編	SBCD-368	形 NY532-□□□□ 形 NY512-□□□□	NY シリーズ産業用 PC の内蔵 EtherCAT ポートを使用するとき。	内蔵 EtherCAT ポートに関して説明します。概要、構成、機能、セットアップについて記述しています。

マニュアル名称	Man.No.	形式	用途	内容
NY シリーズ IPC マシンコントローラ 産業用パネル型 PC ／産業用ボックス型 PC ユーザーズマニュアル 内蔵 EtherNet/IP™ ポート編	SBCD-369	形 NY532-□□□□ 形 NY512-□□□□	NY シリーズ産業用 PC の内蔵 EtherNet/IP ポートを使用するとき。	内蔵 EtherNet/IP ポートに関して説明します。基本設定、タグデータリンク、その他の機能について記述しています。
NJ/NY シリーズ NC 統合コントローラ ユーザーズマニュアル	SBCE-428	形 NJ501-5300 形 NY532-5400	NJ/NY シリーズで数値制御を行うとき。	数値制御機能について説明します。プログラミング時に、コマンドリファレンスマニュアル G コード編 (SBCE-429) と併せて使用してください。
NJ/NY シリーズ コマンドリファレンス マニュアル G コード編	SBCE-429	形 NJ501-5300 形 NY532-5400	G コード／ M コード命令仕様の詳細について知りたいとき。	G コード／ M コード命令の詳細を説明します。プログラミング時に、ユーザーズマニュアル (SBCE-428) と併せて使用してください。
NY シリーズ トラブルシューティング マニュアル	SBCA-438	形 NY532-□□□□ 形 NY512-□□□□	NY シリーズ産業用 PC で検出する異常の詳細について知りたいとき。	NY シリーズ システムにて検出する異常管理の考え方と各異常項目について説明します。
Sysmac Studio Version 1 オペレーションマニュアル	SBCA-470	形 SYSMAC -SE2□□□	Sysmac Studio の操作方法、機能について知りたいとき。	Sysmac Studio の操作方法について説明します。
CNC Operator オペレーションマニュアル	SBCE-430	形 SYSMAC -RTNC0 □□□ D	CNC Operator の概要、使用方法について知りたいとき。	CNC Operator の概要、インストール方法、基本操作、接続操作、主な機能の操作方法などについて説明します。
NX シリーズ EtherCAT® カプラユニット ユーザーズマニュアル	SBCD-361	形 NX-ECC□□□	NX シリーズ EtherCAT カプラユニット、および EtherCAT スレーブターミナルの使用方法について知りたいとき。	NX シリーズ EtherCAT カプラユニットと NX ユニットで構成される EtherCAT スレーブターミナルのシステム概要や構成方法、および EtherCAT を介して NX ユニットを設定、制御、モニタするための、EtherCAT カプラユニットのハードウェアや設定方法、機能について説明します。
NX シリーズ セーフティコントロール ユニット ユーザーズマニュアル	SGFM-710	形 NX-SL□□□□ 形 NX-SI□□□□ 形 NX-SO□□□□	NX シリーズ セーフティコントロールユニットの使用方法について知りたいとき。	NX シリーズ セーフティコントロールユニットのハードウェアや設定方法、機能について説明します。
画像処理システム FH/FZ5 シリーズ ユーザーズマニュアル	SDNB-712	形 FH-1□□□ 形 FH-3□□□ 形 FZ5-L35□ 形 FZ5-6□□ 形 FZ5-11□□	画像処理システム FH/FZ5 シリーズの使用方法について知りたいとき。	FH/FZ5 シリーズを使用するためのソフト的な機能、設定、操作について説明します。
視覚センサ FQ-M シリーズ 位置決め専用視覚センサ ユーザーズマニュアル	SDNB-706	形 FQ-MS12□	位置決め専用視覚センサの使用方法について知りたいとき。	位置決め専用視覚センサのハードウェアや設定方法、機能について説明します。
視覚センサ FZ3 シリーズ ユーザーズマニュアル	SCHB-762	形 FZ3-□□□□	視覚センサ FZ3 シリーズの使用方法について知りたいとき。	視覚センサ FZ3 シリーズのソフト的な機能、設定、操作について説明します。
変位センサ ZW シリーズ ファイバ同軸変位センサ ユーザーズマニュアル	SDNE-703	形 ZW-CE1□	変位センサ ZW シリーズの使用方法について知りたいとき。	変位センサ ZW シリーズのハードウェアや設定方法、機能について説明します。
プログラマブルターミナル NA シリーズ ユーザーズマニュアル ソフトウェア編	SBSA-546	形 NA5-□W□□□□	プログラマブルターミナル NA シリーズのページや各オブジェクトの機能について知りたいとき。	プログラマブルターミナル NA シリーズのページや各オブジェクトの機能について説明します。
プログラマブルターミナル NS シリーズ プログラミングマニュアル	SBSA-512	形 NS15-□□□□□ 形 NS12-□□□□□ 形 NS10-□□□□□ 形 NS8-□□□□□ 形 NS5-□□□□□	プログラマブルターミナル NS シリーズの使用方法について知りたいとき。	プログラマブルターミナル NS シリーズの設定方法や機能について説明します。
CX-Designer ユーザーズマニュアル	SBSA-532	ー	プログラマブルターミナル NS シリーズの画面データを作成したいとき	CX-Designer の操作方法について説明します。

用語説明

コントローラ機能に関連する用語を説明します。

用語	説明
AT（割付先）指定	変数の属性のひとつ。 変数の割付先をユーザが指定する属性。I/O ポートを指定できる。
EtherCAT マスタ機能モジュール	機能モジュールのひとつ。EtherCAT マスタとして、EtherCAT スレーブをコントロールする。
EtherNet/IP 機能モジュール	機能モジュールのひとつ。EtherNet/IP ポートをコントロールする。
FB	ファンクションブロックの略。
FUN	ファンクションの略。
I/O ポート	コントローラが外部のデバイス（スレーブ／ユニット）と、データをやりとりするための論理的なインタフェース。
I/O マップ設定	I/O ポートに対して、変数を割り付けた設定。I/O ポートと変数間の割付情報。
I/O リフレッシュ	あらかじめ定められたメモリエリアの、外部とのサイクリックなデータ交換のこと。
MC 試運転	モータの動作や配線の確認を Sysmac Studio 上から行う機能。
NX ユニット	NX シリーズのユニットの中で、外部接続機器の I/O 処理などを行うユニットのこと。通信カプラユニットは、NX ユニットには含まない。
PDO 通信	プロセスデータ通信の略。プロセスデータ通信周期（＝プライマリ定周期タスクのタスク周期）で、サイクリックに、マスタースレーブ間でデータ交換をする通信。
PLC 機能モジュール	機能モジュールのひとつ。ユーザプログラムの実行、モーション制御機能モジュールへの指示などを行う。
POU	Program Organization Unit。IEC 61131-3 におけるプログラムの実行モデルの単位。 ユーザプログラム全体を構成する基本的な単位として、アルゴリズムとローカル変数テーブルを記述する。 プログラム、ファンクションブロック、ファンクションの 3 種類がある。
SDO 通信	EtherCAT 通信の一種で、任意のタイミングで情報伝達を行うサービスデータオブジェクト（Service Data Objects: SDO）を使用した通信。
Sysmac Studio	NY シリーズの、設定、プログラミング、デバッグ、およびトラブルシューティングを行う、パソコン用ソフトウェア。モーション制御用の機能、およびシミュレータ機能を含む。
アップロード	Sysmac Studio の「同期」機能により、コントローラからデータを Sysmac Studio に転送すること。
一般情報	コントローラ情報およびユーザ情報のイベント重要度を示すランクのひとつ。異常ではないが、イベントログに記録してユーザに通知する情報。
イベントログ機能	イベント（異常または事象）を認識し、その記録を残す機能。
イベント設定	ユーザ異常やユーザ情報を定義すること。
イベントタスク	タスクの実行条件が成立したときに、1 回だけユーザプログラムを実行するタスク。
インライン ST	ラダー図プログラムの中に、その一部として ST プログラムを書くこと。または書かれた ST プログラムのこと。
エッジ	変数の属性のひとつ。 BOOL 型の変数の値が FALSE → TRUE に立ち上がったとき、または TRUE → FALSE に立ち下がったときに、ファンクションブロックに TRUE の値を渡す。
仮想 SD メモリカード	コントローラ機能が各種ファイルを扱うための共有フォルダ。 Sysmac のコントローラである NJ/NX シリーズ CPU ユニットでは、SD メモリカードを使用しているが、NY シリーズコントローラでは、Windows との共有フォルダを仮想的な SD メモリカードとして扱う。 そのため、「仮想」を省略して「SD メモリカード」と呼称する場合がある。
カムデータ変数	カムデータを構造体配列化した変数のこと。 位相と変位データからなる、構造体型の配列指定の変数である。

用語	説明
監視情報	コントローラ情報およびユーザ情報のイベント重要度を示すランクのひとつ。 制御に影響がないレベルの軽微な異常であるが、イベントログに記録してユーザに通知する情報。
機能モジュール	コントローラのソフトウェア構成を、機能ごとに分割した単位。
基本データ型	IEC 61131-3 で定義されているデータ型のこと。 ブール型、ビット列型、整数型、実数型、持続時間型、日付型、時刻型、日付時刻型、文字列型がある。 ユーザが定義するデータ型である「派生データ型」の対義語である。
強制値リフレッシュ	ユーザがデバッグ時などに、外部からの入力および外部への出力を強制値でリフレッシュする機能のこと。 強制値リフレッシュ対象のエリアに対しても、ユーザプログラムの命令から上書することができる。
共用体型	派生データ型の一種で、同じデータを異なるデータ型として扱えるようにしたデータ型のこと。
グローバル変数	すべての POU（プログラム、ファンクションブロック、ファンクション）から読み書き可能な変数のこと。
軽度フォールトレベルのコントローラ異常	NY シリーズコントローラの、ある機能モジュールの一部の制御が不能となる異常のこと。 軽度フォールトの発生後も、NY シリーズコントローラは運転を継続する。
構造体型	派生データ型の一種で、複数の異なるデータ型を階層的にひとつにまとめたデータ型のこと。
コンスタント	変数の属性のひとつ。 コンスタント指定を行った変数は、命令、ST 言語の演算子、または CIP メッセージ通信によって値を書き込むことができなくなる。
コントローラ	産業用パネル型 PC ／産業用ボックス型 PC のリアルタイム OS に組み込まれたコントローラ機能が直接制御する範囲を指す。 NY シリーズシステムでは、EtherCAT スレーブ（汎用／サーボドライバ含む）を含めて「コントローラ」と呼称する場合もある。
コントローラ機能	産業用パネル型 PC ／産業用ボックス型 PC のリアルタイム OS に組み込まれた、機械制御を実行するための機能を指す。
コントローラ異常	NY シリーズのシステムで定義した異常のこと。 「全停止フォールトレベル」、「部分停止フォールトレベル」、「軽度フォールトレベル」、「監視情報」のコントローライベントの総称。
コントローライベント	NY シリーズシステムのイベントの一種。システムが、ユーザに通知する必要があると定義する、異常、情報のこと。定義する発生要因をシステムが検知するとコントローライベントが発生する。
コントローラ情報	NY シリーズのシステムで定義した、異常ではない情報のこと。「一般情報」のコントローライベント。
サーボ／エンコーダ入力スレーブ	EtherCAT スレーブの内、「軸」に割り付けることができるスレーブを指す。NY シリーズでは、サーボドライバとエンコーダ入力スレーブを指す。
軸	モーション制御機能モジュール内の機能単位のこと。外部のサーボドライバの駆動機構、エンコーダ入力スレーブの検出機構などを割り付ける。
軸グループ	モーション制御機能モジュール内の、複数の「軸」をまとめた機能単位のこと。
軸グループ変数	軸グループごとの各種ステータス情報や一部の軸グループパラメータ設定情報などを構造体で定義したシステム定義変数。 モーション制御命令の軸グループの指定、および軸グループの指令補間速度や異常情報などのモニタに使用される。
軸変数	軸ごとの各種ステータス情報や一部の軸パラメータ設定情報などを構造体で定義したシステム定義変数。 モーション制御命令の軸の指定、および軸の指令現在位置や異常情報などのモニタに使用される。
システム共通処理	コントローラが、タスク内で I/O リフレッシュやユーザプログラム実行と共に実行するシステム処理のこと。変数のタスク間排他処理、データトレース処理などを行う。
システムサービス	コントローラが、タスク以外に実行する処理のこと。通信処理、仮想 SD メモリカードアクセス、自己診断処理などの処理を実行する。

用語	説明
システム定義変数	すべての属性がシステムによって定義されており、ユーザはそれを一切変更できない変数のこと。
初期値	変数の属性のひとつ。以下のときに設定される変数の値である。 • 電源投入時 • 運転モードへの切替時 • ユーザプログラム転送時で初期化する指定のとき • 全停止フォールトレベルのコントローラ異常発生時
スレーブ	リモート I/O を行う親局（マスタ）に対する子局のこと。
スレーブターミナル	通信カプラユニットに NX ユニット群を装着したビルディングブロック型のリモート I/O ユニットの総称。スレーブターミナルは、スレーブの一種である。
スレーブ／ユニット構成	EtherCAT 構成およびユニット構成の総称。
絶対値エンコーダ原点位置オフセット	絶対値エンコーダ付きのサーボドライバの現在位置を復元するために、コントローラが記憶している、原点確定後の指令位置と、絶対値エンコーダから読み出した絶対値データとの差のこと。
全停止フォールトレベルのコントローラ異常	NY シリーズコントローラ全体の制御が不能となる異常のこと。コントローラは、ユーザプログラムの実行をすぐに停止し、リモート I/O を含むすべてのスレーブ／ユニットに対して負荷を遮断する。
ダウンロード	Sysmac Studio の「同期」機能により、Sysmac Studio からコントローラにデータを転送すること。
タスク	「プログラム実行時の属性」のこと。
タスク周期	プライマリ定周期タスクおよび定周期タスクの実行間隔。
通信カプラユニット	NX ユニット群と上位のネットワークマスタを、ネットワーク上でリモート I/O 通信するためのインタフェースユニットの総称。たとえば、EtherCAT カプラユニットは、EtherCAT 通信ネットワークの通信カプラユニットである。
定周期タスク	周期ごとに、I/O リフレッシュとユーザプログラム実行をするタスク。
デバイス変数	I/O ポートを介して特定のデバイスにアクセスするための変数。
同期	NY シリーズコントローラと Sysmac Studio の情報を自動的に比較を行い、違いの有無、箇所、内容を階層的に表示し、かつ一致させる機能
名前空間	ファンクション、ファンクションブロック定義、データ型の名称を、分割・階層化して管理する仕組み。
ネットワーク公開	変数の属性のひとつ。 CIP メッセージ通信またはタグデータリンク機能によって、その変数を、他のコントローラや上位パソコンなどからも読み書きできるようにする。
配列指定	変数の指定のひとつ。同じデータ型の要素を複数個まとめて、ひとつの変数にしたもの。ひとつひとつの要素は、先頭からの通し番号（添え字）で指定する。
派生データ型	仕様をユーザが定義するデータ型のこと。構造体型、共用体型、列挙型がある。
範囲指定	変数の指定のひとつ。変数を、あらかじめ決められた範囲内の値しかとることができないことを明示する。
汎用スレーブ	EtherCAT スレーブの内、「軸」に割り付けることができないスレーブを指す。
ファンクション	演算処理など、入力に対して出力が一意に決まる部品を作成するときに使用する POU。
ファンクションブロック	タイマやカウンタなど、状態により、同じ入力に対して出力が異なる場合がある部品を作成するときに使用する POU。
部分停止フォールトレベルのコントローラ異常	NY シリーズコントローラの、ある機能モジュール全体の制御が不能となる異常のこと。 部分停止フォールトの発生後も、NY シリーズコントローラは運転を継続する。
プライマリ定周期タスク	最も優先度が高いタスク。
プログラム	ファンクション、ファンクションブロックと並ぶ、3 種類の POU のひとつ。 タスクに割り付けられて動作する。
プロセスデータ通信	EtherCAT 通信の一種で、定周期でリアルタイムの情報交換を行うプロセスデータオブジェクト（Process Data Objects: PDO）を使用した通信。「PDO 通信」とも呼ぶ。
変数	ユーザプログラム中で使用する、任意の数値や文字列を表現する概念のこと。 任意の値を代入することによって、変数の値を変えることができる。常に値が変わらない「定数」の対語である。

用語	説明
変数メモリ	AT（割付先）指定がない変数の現在値。「AT （割付先）指定なし」属性の変数でのみアクセスする。
保持	変数の属性のひとつ。以下のときに、本属性を設定した場合は変数の値を保持し、設定しない場合は初期値になる。 • 電断後の電源投入時 • 運転モードへの切替時 • ユーザプログラム転送時で初期化しない指定のとき
命令	POU のアルゴリズムの中に記載する、オムロンが用意した最小単位の処理要素。命令には、ラダー図（接点、コイル）、FB 型、FUN 型、ST 言語の文がある。
メインメモリ	コントローラが OS、ユーザプログラム実行に使用するコントローラ内部のメモリ。
モーション制御機能モジュール	機能モジュールのひとつ。ユーザプログラム内のモーション命令から与えられた指令に従って、モーション制御を実行する。
モーション制御命令	モーション制御機能を実行するためのファンクションブロック定義の命令。 モーション制御機能モジュールでは、PLCopen® のモーション制御用ファンクションブロックに準拠した命令とモーション制御機能モジュール独自の命令を持つ。
ユーザイベント	NY シリーズシステムのイベントの一種。ユーザが定義するイベント。ユーザ異常、ユーザ情報の総称。
ユーザ定義変数	すべての変数属性がユーザによって定義されており、ユーザによる変更が可能な変数。
ユーザプログラム	複数のプログラムの集合体。
ユニット	通信カプラユニットに装着可能な機器のこと。
ユニット構成	Sysmac Studio で設定したユニットの構成情報で、コントローラに接続されているユニットの種類と接続位置の情報からなる。
リテラル	ユーザプログラム内に使用される定数表現のこと。
列挙型	派生データ型の一種。あらかじめ用意された名前リスト (列挙子) 内のひとつを値とするデータ型。
列挙子	列挙型の変数が取り得る複数の値を、それぞれ文字で表現したもの。 列挙型の変数の値は、列挙子の中のどれかひとつをとる。
ローカル変数	定義された POU 内だけからアクセス可能な変数。複数の POU からアクセス可能な「グローバル変数」の対語である。 ローカル変数には、内部変数、入力変数、出力変数、入出力変数、外部変数が含まれる。

マニュアル改訂履歴

マニュアル改訂記号は、表表紙・裏表紙に記載されている Man.No. の後尾に付記されます。



改訂記号	改訂年月	改訂理由・改訂ページ
A	2016 年 9 月	初版
B	2017 年 4 月	• コントローラ Ver.1.14 以降のサポート機能に関する記述の追記 • 誤記修正
C	2017 年 10 月	• コントローラ Ver.1.16 以降のサポート機能に関する記述の追記 • 誤記修正
D	2018 年 4 月	• コントローラ Ver.1.18 以降のサポート機能に関する記述の追記
E	2019 年 7 月	• コントローラ Ver.1.21 以降のサポート機能に関する記述の追記 • 誤記修正
F	2021 年 3 月	• コントローラ Ver.1.24 以降のサポート機能に関する記述の追記 • 誤記修正
G	2021 年 7 月	• コントローラ Ver.1.26 以降のサポート機能に関する記述の追記
H	2022 年 8 月	• セキュリティに関する安全上の注意追記に伴う改訂
J	2023 年 12 月	• 誤記修正

1

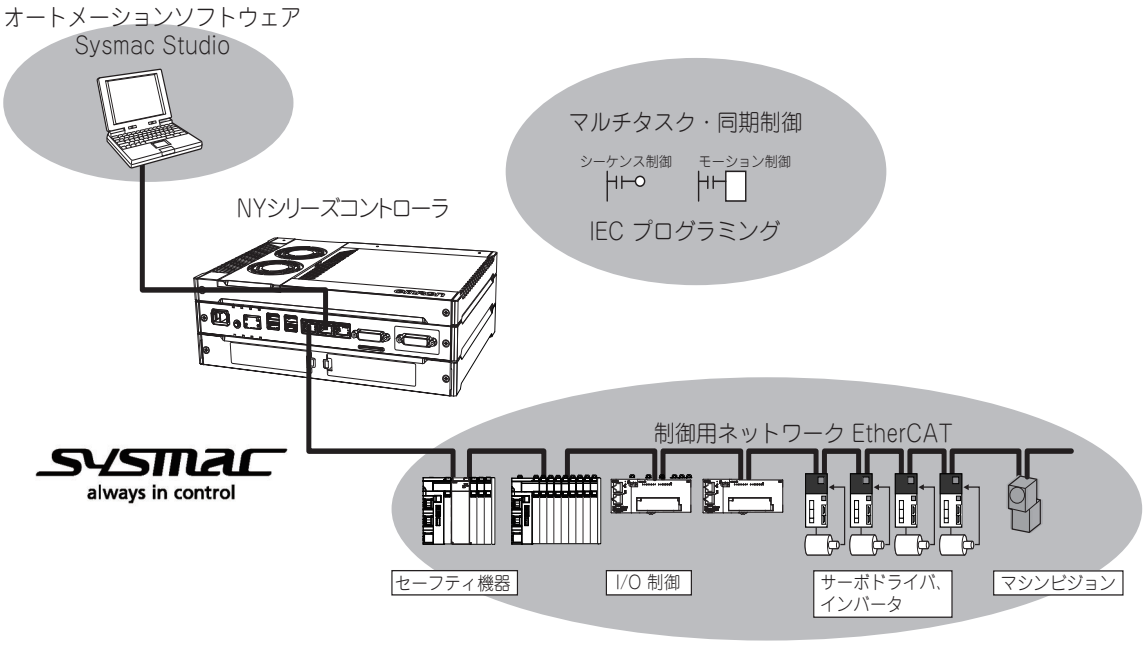
NY シリーズコントローラの概要

本章では、NY シリーズコントローラの特長、システム構成の概要、仕様、使用手順について説明します。

1-1 NY シリーズコントローラとは 1-2
1-1-1 特長 1-3
1-1-2 システム構成の概要 1-5
1-2 仕様の概要 1-7
1-3 商品形式 1-9
1-4 NY シリーズの使用手順の全体 1-10
1-4-1 手順の概要 1-10
1-4-2 手順の詳細 1-11

1-1　NY シリーズコントローラとは

産業用 PC プラットフォーム NY シリーズコントローラは、機械制御に必要な機能・高速性能と、産業用コントローラとしての安全性・信頼性・保守性を両立した、次世代コントローラです。
従来のプログラマブルコントローラの機能を包含し、モーション制御に必要な各種機能を付加した統合型コントローラとして、セーフティ機器、視覚装置、モーション機器、I/O などの入出力機器を、高速な EtherCAT 上で同期して制御できます。
オムロンでは統一した通信仕様やユーザインターフェース仕様で設計されたオムロン制御機器を、Sysmac デバイスとしてご提供しています。産業用 PC プラットフォーム NY シリーズは、EtherCAT スレーブなどの Sysmac デバイス、およびオートメーションソフトウェア Sysmac Studio との組み合わせにおいて、最適な機能、操作性を実現できるように設計されています。
Sysmac デバイスにより構成されたシステムでは、機器どうしが容易につながり、統一した考え方とユーザビリティで操作できます。



1-1-1　特長

NY シリーズ IPC マシンコントローラ 産業用パネル型 PC ／産業用ボックス型 PC のコントローラ機能の特長について説明します。

ハードウェア上の特長

● 制御用ネットワーク EtherCAT を標準装備

NY シリーズコントローラは EtherCAT 通信マスタ機能ポートを標準装備しています。
EtherCAT は、Ethernet システムをベースとし、より高速で高効率な通信を実現する高性能な産業用ネットワークシステムです。各ノードが Ethernet フレームを高速で転送することにより、短い定周期な通信サイクルタイムを実現することができます。
制御用ネットワーク EtherCAT を標準採用したことで、マシン制御に必要な I/O システム、サーボドライバ、インバータ、マシンビジョンなどの機器を 1 つのネットワークで接続できます。

● EtherCAT スレーブターミナルをサポート

EtherCAT スレーブターミナルを使用することで省スペース化ができます。また、豊富な種類の NX ユニットによりフレキシブルなシステム構築ができます。

● EtherCAT 上にセーフティサブシステムを実現

NX シリーズ セーフティコントロールユニットを使用することによって、シーケンスコントロール、モーションコントロールのシステムに、セーフティコントロールを EtherCAT 上のサブシステムとして統合することができます。

● EtherNet/IP 通信機能ポートを標準装備

NY シリーズコントローラは EtherNet/IP 通信機能ポートを標準装備しています。
EtherNet/IP は、Ethernet を使用した産業用のマルチベンダネットワークです。コントローラ間のネットワークやフィールドネットワークとして使用することができます。また、標準の Ethernet 技術が使用されているため、さまざまな汎用 Ethernet 機器とも混在させて使用することができます。

● 高いハードウェア信頼性

PLC 並みのハードウェア信頼性と RAS 機能を用意しています。

● マルチコアプロセッサによる並列実行

形 NY532-□□□□ および形 NY512-1□□□ は、マルチコアプロセッサを搭載しているため、タスクの実行とタグデータリンクサービスやシステムサービスを並列に実行することができます。これにより、大規模な装置も高速に制御できます。

ソフトウェア上の特長

● シーケンス制御とモーション制御を統合

1 台のコントローラでシーケンス制御とモーション制御の両方をカバーします。これにより、シーケンス制御と多軸同期制御を同時に実現します。同じ制御周期の中で、シーケンス制御・モーション制御と I/O リフレッシュが実行されます。
また、制御周期は EtherCAT のプロセスデータ通信周期と一致します。これにより、定周期でゆらぎの少ない高精度な、シーケンス制御およびモーション制御を実現できます。

● マルチタスク対応

複数のタスクに I/O リフレッシュやユーザプログラム実行などの一連の処理を割り付け、それぞれに実行条件と実行順序を指定し、それらを組み合わせることでアプリケーションに合わせて柔軟な制御が構築できます。

● 国際標準規格 IEC 61131-3 準拠のプログラミング言語仕様

IEC 61131-3 に準拠した言語仕様を搭載しています。一部、オムロン独自の改良を加えています。
PLCopen® に準拠したモーション制御命令をはじめ、IEC ルールに則った豊富な命令群（POU）を用意しています。

● メモリマップを意識しない、変数によるプログラミング

パソコン上で高級言語を使用するときと同様に、すべてのデータは変数によりアクセスします。作成した変数はコントローラのメモリ上に自動的に割り付けられ、ユーザがそれを意識する必要はありません。

● 豊富なセキュリティ機能

操作権限の設定、ID によるユーザプログラムの実行制限など、豊富なセキュリティ機能を用意しています。

● コントローラ全体の監視機能

コントローラは、装着された EtherCAT スレーブを含めたコントローラ全体の事象（イベント）を監視します。
異常時の対処方法は、Sysmac Studio または NS/NA シリーズ表示器でガイドされます。発生した事象（イベント）は履歴として記憶されます。

● オートメーションソフトウェア Sysmac Studio

Sysmac Studio は、コントローラだけでなく周辺機器や EtherCAT 上の機器までを、1 つのソフトウェアでカバーした統合開発環境です。機器の違いにかかわらず、統一した操作性を提供します。設計から、デバッグ、シミュレーション、立ち上げ、運用開始後の変更までの、すべての工程を支援します。

● 豊富なシミュレーション機能

仮想コントローラの実行機能、デバッグ機能など、豊富なシミュレーション機能を用意しています。

1-1-2　システム構成の概要

NY シリーズのシステム構成は、以下のとおりです。

● 基本構成

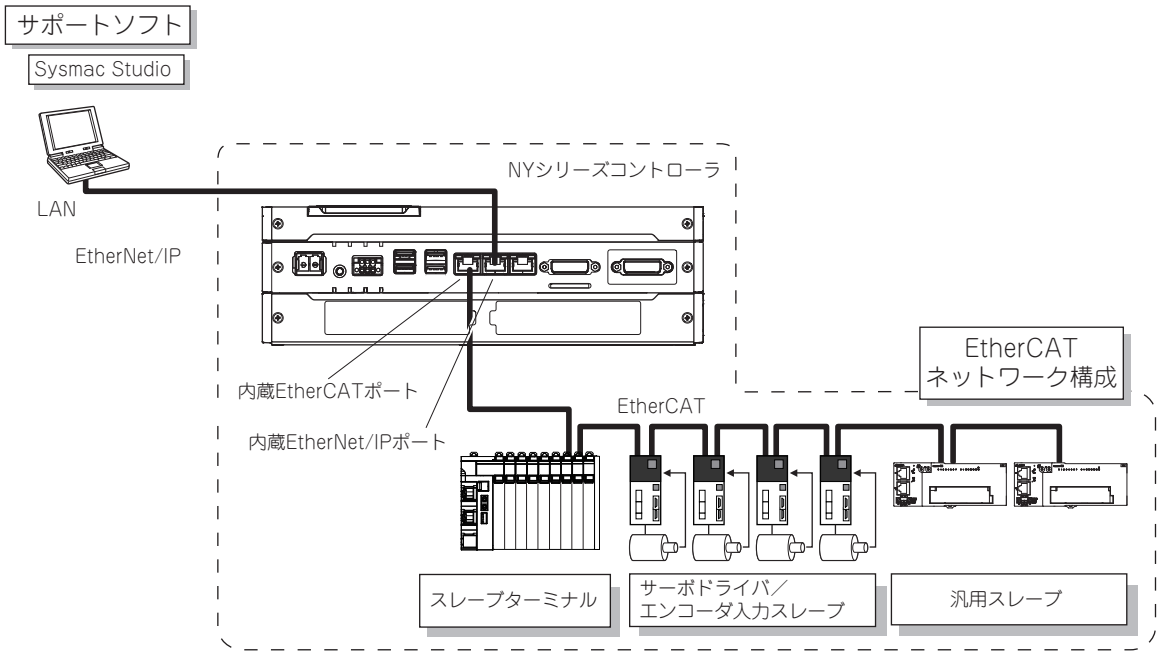
NY シリーズでは、EtherCAT ネットワーク構成とサポートソフトを基本構成とします。

- EtherCAT ネットワーク構成

  内蔵 EtherCAT ポートを使って、EtherCAT スレーブターミナル、デジタル I/O、アナログ I/O などの汎用スレーブ、およびサーボ／エンコーダ入力スレーブを接続できます。この構成を使用することで、定周期かつゆらぎの少ない高精度な、シーケンス制御およびモーション制御を実現できます。
- サポートソフト

  内蔵 EtherNet/IP ポートに Ethernet ケーブルで接続します。

  サポートソフトの接続構成の詳細は、🕮「10-2 Sysmac Studio の接続」(P.10-4) を参照してください。

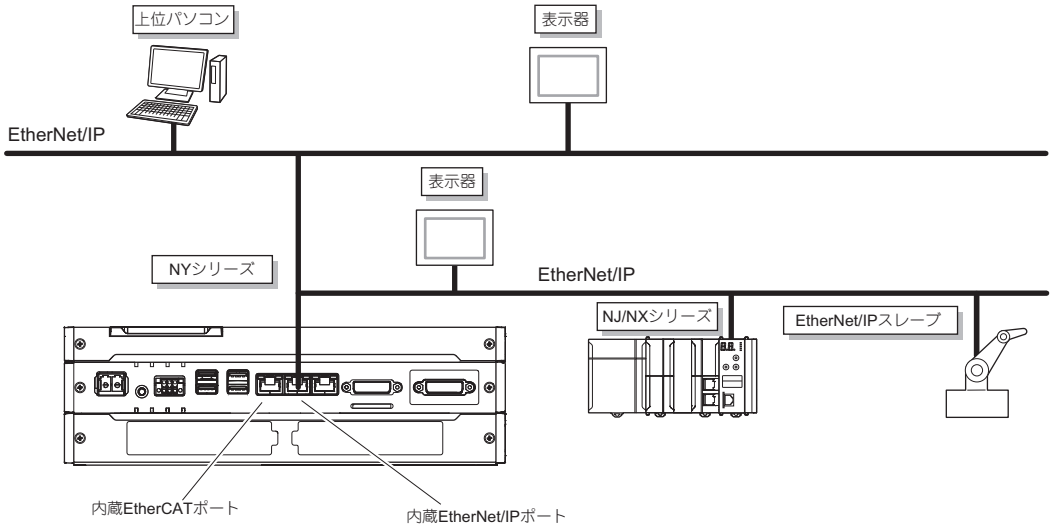


参考

通信カプラユニットに Sysmac Studio を直接接続して、スレーブターミナルの設定を行うことも可能です。詳細は、🕮『NX シリーズ　EtherCAT カプラユニット ユーザーズマニュアル (SBCD-361)』を参照してください。

● ネットワーク構成

- 内蔵 EtherNet/IP ポートに上位パソコン、表示器、NJ/NX/NY シリーズを接続します。



ネットワーク構成の詳細は、🕮「第 10 章 通信設定」(P.10-1) を参照してください。

● サポートソフト

NY シリーズのコントローラ機能の設定／モニタ／デバッグを行う周辺ツールは、以下のとおりです。

- Sysmac Studio

  NY シリーズでは、コントローラ構成・設定、プログラム作成、デバッグ、シミュレーションを行うサポートソフトとして Sysmac Studio を基本として使用します。

- その他のサポートソフト

  Sysmac Studio 以外にも、ソフトウェアパッケージ Sysmac Studio スタンダードエディションには以下のソフトが含まれます。

構成ソフト	用途
Sysmac Studio	シーケンス制御およびモーション制御を含む、以下以外のすべての機能のときに使用します。
Network Configurator	EtherNet/IP ポートやユニットで、タグデータリンクをするときに使用します。*1
CX-Designer	NS シリーズ表示器の画面を作成するときに使用します。

*1. CS/CJ シリーズ PLC がオリジネータデバイスの場合は、Network Configurator で設定します。

NY シリーズのコントローラ機能以外の周辺ツールについては、🕮『NY シリーズ 産業用パネル型 PC ／産業用ボックス型 PC ユーザーズマニュアル セットアップ編 (SBCA-441)』を参照してください。

1-2　仕様の概要

ここでは、NY シリーズのコントローラ機能の主な性能仕様について説明しています。一般仕様、性能仕様、機能仕様は、📖「A-1 仕様」(P.A-3) を参照してください。

項目				形 NY5□2-15□□	形 NY5□2-□4□□	形 NY5□2-13□□
プログラミング	プログラム容量[*1]	サイズ		40MB		
		数	POU 定義数	3,000		
			POU インスタンス数	24,000		
	変数容量	保持属性あり	サイズ	4MB		
			変数の数	40,000		
		保持属性なし	サイズ	64MB		
			変数の数	180,000		
	データ型	データ型の数		4,000		
モーション制御	制御軸数	制御軸最大数[*2]		64 軸	32 軸	16 軸
		使用実軸最大数[*3]		64 軸	32 軸	16 軸
		単軸制御最大数		64 軸	32 軸	16 軸
		直線補間制御最大数		1 軸グループあたり 4 軸		
		円弧補間制御軸数		1 軸グループあたり 2 軸		
	軸グループ最大数			32 グループ		
	モーション制御周期			EtherCAT 通信のプロセスデータ通信周期と同じ		
	カム	カムデータ点数	1 カムテーブルあたりの最大点数	65,535 点		
			全カムテーブルの最大点数	1,048,560 点		
		カムテーブル最大テーブル数		640 テーブル		
内蔵 EtherNet/IP ポート	ポート数			1		
	物理層			10BASE-T/100BASE-TX/1000BASE-T		
	フレーム長			最大 1,514 バイト		
	媒体アクセス方式			CSMA/CD		
	変調方式			ベースバンド		
	トポロジ			スター型		
	伝送速度			1G ビット /s（1000BASE-T）		
	伝送媒体			ツイストペアケーブル（シールド付：STP）：カテゴリ 5、5e 以上		
	伝送距離（ハブとノード間の距離）最大値			100m		
	カスケード接続最大数			スイッチングハブの使用において制限なし		
	CIP サービス：タグデータリンク（サイクリック通信）	コネクション最大数		128		
		パケットインターバル[*4]		コネクションごとに設定可能 1 ～ 10000ms（1ms 単位）		
		ユニット許容通信帯域		20,000 pps[*5]（ハートビート含む）		
		タグセット最大数		128		
		タグ種別		ネットワーク変数		
		1 コネクション（=1 タグセット）あたりのタグ数		8（タグセットにコントローラステータスを含める場合は 7）		
		タグ最大数		256		
		1 ノードあたりの最大リンクデータサイズ（全タグの合計サイズ）		184,832 バイト		
		1 コネクションあたりの最大データサイズ		1,444 バイト		
		登録可能なタグセット最大数		128 （1 コネクション= 1 タグセット）		
		1 タグセットの最大サイズ		1,444 バイト（タグセットにコントローラステータスを含める場合は 2 バイト分を使用）		
		マルチキャストパケットフィルタ機能[*6]		可		

項目			形 NY5□2-15□□	形 NY5□2-□4□□	形 NY5□2-13□□
内蔵 EtherCAT ポート	通信規格		IEC 61158 Type12		
	EtherCAT マスタ仕様		Class B 対応（Feature Pack Motion Control 対応）		
	物理層		100BASE-TX		
	変調方式		ベースバンド		
	伝送速度		100M ビット /s（100BASE-TX）		
	Duplex モード		Auto		
	トポロジ		ライン、ディジーチェーン、分岐		
	伝送媒体		カテゴリ 5 以上　ツイストペアケーブル（アルミテープと編組の二重遮へいシールドケーブル、ストレートを推奨）		
	ノード間距離最大値		100m		
	スレーブ最大数		128		
ユニット構成	装着できるユニット数	システム全体の NX ユニット最大数	4,096 （EtherCAT スレーブターミナル上）		

*1. 実行オブジェクト、変数テーブル（変数名など）の容量です。
*2. すべての軸種別の合計のことです。
*3. 軸種別が [サーボ軸] または [エンコーダ軸] で、かつ軸使用が [使用軸] に設定された軸の合計のことです。
*4. ノード数に依存せず、設定した周期で回線上をデータ更新します。
*5. pps とは Packet Per Second を意味し、1 秒間に処理可能な送受信パケット数を示します。
*6. EtherNet/IP ポートは、IGMP クライアントを実装しているため、IGMP Snooping 対応のスイッチングハブを使用することで不要なマルチキャストパケットのフィルタリングが行えます。

1-3　商品形式

NY シリーズコントローラの形式は、制御軸数やハードウェアの構成を識別できるようになっています。

形 NY 5 3 2 - 1 5 0 0 - 1 1 2 2 1 3 C 2 2
1　2 3 4　5 6 7 8　9 10 11 12 13 14 15 16 17

番号	項目	オプション
1	シリーズ名	NY ： NY シリーズ産業用 PC プラットフォーム
2	コントローラ仕様	5 ： 最大 64 軸までの大規模かつ高速高精度な制御用途
3	モデルタイプ	1 ： 産業用ボックス型 PC 3 ： 産業用パネル型 PC
4	追い番	2 ： 最新版
5	機能モジュール	1 ： 標準 5 ： CNC
6	モーション制御軸数	3 ： 16 軸 4 ： 32 軸 5 ： 64 軸
7	追加機能ソフトウェアモジュール	0 ： ---
8	予約	0 ： ---
9	拡張スロット	0 ： PCIe スロットなし 1 ： PCIe スロット ×1
10	フレームタイプ	1 ： アルミニウムフレーム、黒色、透過型静電容量タッチタイプ 2 ： アルミニウムフレーム、ニッケルめっき X ： ディスプレイなし（産業用ボックス型 PC）
11	ディスプレイサイズ	1 ： 12.1 インチ 2 ： 15.4 インチ 3 ： 18.5 インチ X ： ディスプレイなし（産業用ボックス型 PC）
12	OS	1 ： Windows Embedded Standard 7 32bit 2 ： Windows Embedded Standard 7 64bit 4 ： Windows 10 IoT Enterprise 2019 LTSC 64bit
13	プロセッサ	1 ： Intel® Core ™ i7-4700EQ Processor 4 ： Intel® Core ™ i5-7440EQ Processor
14	メインメモリ	3 ： 8GB Non-ECC 5 ： 32GB Non-ECC
15	ストレージ	6 ： 128GB CFast 7 ： 256GB CFast 8 ： 32GB SSD SLC 9 ： 64GB SSD SLC C ： 320GB HDD K ： 128GB SSD MLC P ： 1TB SSD 3D TLC
16	インタフェースオプション	0 ： なし 1 ： RS-232C 2 ： DVI-D 6 ： NY Monitor Link
17	ロゴマーク	0 ： OMRON 2 ： カスタム X ： ディスプレイなし（産業用ボックス型 PC）

参考

本マニュアルは主にコントローラ機能を説明をしているため、コントローラ機能に影響しない番号 9 以降の形式を省略して記載している場合があります。

1-4　NY シリーズの使用手順の全体

ここでは、NY シリーズのコントローラ機能の全体手順の概要と詳細について、説明しています。

1-4-1　手順の概要

NY シリーズのコントローラ機能は以下の流れで使用します。

STEP 1. ソフト設計

システムの全体構成や、タスク構成／プログラム／変数を設計します。

- STEP1-1 I/O と処理の設計
- STEP1-2 タスク設計
- STEP1-3 プログラムの設計

STEP 2. ソフトウェア設定とプログラミング

STEP1 で設計したシステム構成をサポートソフトにて作成するとともに変数の割付を行います。また、タスクおよびプログラムの作成を行い、シミュレーションなどのデバッグを行います。

- STEP2-1 スレーブ／ユニット構成
- STEP2-2 コントローラ設定
- STEP2-3 プログラミング
- STEP2-4 オフラインデバッグ

STEP 3. 装着とハードウェアの設定

各種ユニットを装着し、必要に応じてハードウェアの設定を行います。

STEP 4. 配線

ネットワークケーブルや I/O の配線を行います。

STEP 5. 実機による動作確認と運転

サポートソフトと実際のシステムに接続し、プロジェクトをダウンロードします。実機にて動作確認を行い、運転を開始します。

1-4-2　手順の詳細

STEP1. ソフト設計

手順	内容	参照
STEP1-1 **I/O と処理の設計**	• 外部 I/O とユニットの構成 • 外部機器とリフレッシュする周期 • プログラム処理の内容	🕮『NY シリーズ 産業用パネル型 PC ユーザーズマニュアル ハードウェア編（SBCA-435）』 🕮『NY シリーズ 産業用ボックス型 PC ユーザーズマニュアル ハードウェア編（SBCA-434）』
STEP1-2 **タスク設計**	• タスク構成 • タスクープログラムの関係 • 各タスクの周期 • スレーブ／ユニットのリフレッシュ周期 • タスク間の変数の排他方法	🕮「4-2-3 タスク設定」（P.4-7）
STEP1-3 **プログラムの設計**		
POU（処理の分割単位）の設計	• プログラム • ファンクション／ファンクションブロック • 各アルゴリズムの言語の選択	🕮「第 6 章 プログラミング」（P.6-1）
変数の設計	• POU 共通で使用する変数と、各 POU だけで使用する変数を仕分け • スレーブ／ユニットにアクセスするためのデバイス変数の変数名を設計 • 変数の名称、保持などの各属性を設計 • 変数のデータ型を設計	🕮「6-3 変数」（P.6-27）

STEP2. ソフトウェア設定とプログラミング

手順	内容	Sysmac Studio での操作	参照
プロジェクトの新規作成	1. Sysmac Studio でプロジェクトを新規作成します。 2. コントローラを挿入します。	［新規プロジェクト］ボタン ［挿入］|［コントローラ］	🕮『Sysmac Studio Version 1 オペレーションマニュアル（SBCA-470）』

以下の「コントローラ構成・設定」および「プログラミング／タスク設定」の順は、いずれも可能です。

STEP2-1 **スレーブ／ユニット構成**			
1）スレーブ／ユニット構成の作成	1. オフラインまたはオンラインで、スレーブ／ユニット構成を作成します。（オンラインの場合、STEP 5 のオンライン接続を実行の上） 2. スレーブターミナルを使用する場合、スレーブターミナル設定を行います。	EtherCAT スレーブ設定エディタ ユニットエディタ	🕮「3-2 EtherCAT スレーブ構成の作成」（P.3-4） 🕮『NX シリーズ EtherCAT カプラユニット ユーザーズマニュアル（SBCD-361）』

2）I/O ポートへのデバイス変数の割付	デバイス変数を変数テーブルに登録 （変数名をユーザ定義または自動生成）	I/O マップ	🕮「3-3 I/O ポートとデバイス変数」（P.3-5）

（モーション制御をするとき）

3）「軸」の作成と、サーボ／エンコーダ入力スレーブへの割付	「軸」を作成し、実軸または仮想軸に割付。補間軸制御のときは「軸グループ」を作成。	構成・設定｜モーション制御設定にて設定	🕮「3-5 「軸」の作成と、サーボドライバ／エンコーダ入力スレーブの割付」（P.3-11）

STEP2-2 コントローラ設定	Sysmac Studio で、以下のパラメータを設定します。		🕮「第 4 章 コントローラ設定」（P.4-1）
	PLC 機能モジュール関連の初期設定	構成・設定｜動作設定にて設定	🕮「4-2 PLC 機能モジュールの初期設定」（P.4-4）
	（モーション制御をするとき） モーション制御機能モジュール関連の初期設定	構成・設定｜モーション制御設定にて設定	🕮「4-3 モーション制御機能モジュールの初期設定」（P.4-12）
	EtherCAT マスタ機能モジュール関連の初期設定	構成・設定｜EtherCAT にて設定	🕮「4-4 EtherCAT マスタ機能モジュールの初期設定」（P.4-14）
	EtherNet/IP 機能モジュール関連の初期設定	構成・設定｜内蔵 EtherNet/IP ポート設定にて設定	🕮「4-5 EtherNet/IP 機能モジュールの初期設定」（P.4-15）

STEP2-3 プログラミング			
1）変数の登録	• Sysmac Studio で、グローバル変数テーブルへ、POU 共通の変数を登録 • 各プログラムのローカル変数テーブルを登録 • 各ファンクションブロック、ファンクションのローカル変数テーブルを登録	グローバル変数テーブルエディタ ローカル変数テーブルエディタ	🕮『Sysmac Studio Version 1 オペレーションマニュアル（SBCA-470）』 🕮「6-3 変数」（P.6-27）
2）POU のアルゴリズムの作成	POU（プログラム／ファンクションブロック／ファンクション）のアルゴリズムを各言語で作成します	プログラミングエディタ	🕮「第 6 章 プログラミング」（P.6-1） 🕮『NY シリーズ　コマンドリファレンスマニュアル　基本編（SBCA-437）』、 🕮『NY シリーズ コマンドリファレンスマニュアル モーション編（SBCE-380）』
3）タスクの設定	タスク関連の設定をします	タスク設定にて設定	🕮「4-2-3 タスク設定」（P.4-7）

STEP2-4 オフラインデバッグ	シミュレータ（仮想コントローラ）で、アルゴリズムをチェック		🕮「第 7 章 動作確認と運転」（P.7-1）

STEP3. 装着とハードウェアの設定

手順	内容	参照
1）装着	• 各部の取り付け • 制御盤への設置	🕮『NY シリーズ 産業用パネル型 PC ユーザーズマニュアル ハードウェア編 (SBCA-435)』 🕮『NY シリーズ 産業用ボックス型 PC ユーザーズマニュアル ハードウェア編 (SBCA-434)』
2）共有フォルダの設定	• 仮想 SD メモリカードとして使用する共有フォルダの Windows 側の設定	🕮『NY シリーズ 産業用パネル型 PC ／産業用ボックス型PC ユーザーズマニュアル セットアップ編 (SBCA-441)』
3）ハードウェアの設定	• EtherCAT スレーブのノードアドレス設定	🕮各 EtherCAT スレーブのユーザーズマニュアル、各高機能ユニットのユーザーズマニュアル

STEP4. 配線

手順	内容	参照
1）Ethernet ケーブルの配線	• 内蔵 EtherCAT ポートの配線 • 内蔵 EtherNet/IP ポートの配線	🕮『NY シリーズ 産業用パネル型 PC ユーザーズマニュアル ハードウェア編 (SBCA-435)』 🕮『NY シリーズ 産業用ボックス型 PC ユーザーズマニュアル ハードウェア編 (SBCA-434)』
2）I/O 配線	• EtherCAT スレーブの I/O 配線	🕮各 EtherCAT スレーブのユーザーズマニュアル 🕮『NY シリーズ 産業用パネル型 PC ユーザーズマニュアル ハードウェア編 (SBCA-435)』 🕮『NY シリーズ 産業用ボックス型 PC ユーザーズマニュアル ハードウェア編 (SBCA-434)』
	• 配線のチェック	🕮『Sysmac Studio Version 1 オペレーションマニュアル（SBCA-470)』
3）パソコン（Sysmac Studio）との配線	• 内蔵 EtherNet/IP ポートの配線	🕮『Sysmac Studio Version 1 オペレーションマニュアル（SBCA-470)』

STEP5. 実機による動作確認と運転			
手順	**内容**	**Sysmac Studio での操作**	**参照**
1）Sysmac Studio とのオンライン接続、プロジェクトのダウンロード	コントローラの電源を ON し、Sysmac Studio とオンライン接続します。そののち、プロジェクトのダウンロードをします。*1 実装ユニットから STEP2-1 の「スレーブ／ユニット構成の作成」をする場合は、STEP2-1 の前に実施してください。	[コントローラ]-[通信設定] [コントローラ]-[同期]	🕮「第 7 章 動作確認と運転」(P.7-1)
2）実機動作確認	1. I/O マップまたはウォッチウィンドウにて、実 I/O への強制値リフレッシュなどをして、配線確認をします。 2. モーション制御をする場合は、プログラムモードにて、MC 試運転画面で、配線チェックを行います。そののち、ジョグ送りでモータの方向確認、相対位置決めで移動量（電子ギアの設定）確認、原点復帰動作確認をします。 3. 運転モードにしてユーザプログラムの動作確認をします。	－	🕮「第 7 章 動作確認と運転」(P.7-1)
3）コントローラの本運転	本運転を開始します。	－	－

*1. ダウンロードは、Sysmac Studio の同期機能を使用します。

2

NY シリーズコントローラの動作

NY シリーズコントローラを使用する上で必要な知識として、コントローラのしくみ、および各状態における動作を説明します。

2-1 NY シリーズコントローラの動作の概要 2-2
2-1-1 NY シリーズコントローラのしくみの概要 2-2
2-1-2 NY シリーズコントローラの各状態における動作の概要 2-3

2-2 ソフトウェアのしくみ 2-4
2-2-1 ソフトウェア構成 2-4
2-2-2 ソフトウェアの動作 2-5

2-3 I/O とのアクセスのしくみ 2-10
2-3-1 変数の種類 2-10
2-3-2 変数と I/O のアクセス 2-13

2-4 シーケンス制御とモーション制御のしくみ 2-16
2-4-1 制御の概要 2-16
2-4-2 シーケンス制御のしくみ 2-18
2-4-3 モーション制御のしくみ 2-19
2-4-4 シーケンス制御とモーション制御の同期 2-20

2-5 コントローラ内のデータの概要 2-21

2-6 コントローラの各状態における動作 2-22
2-6-1 コントローラの状態 2-22
2-6-2 各状態における動作 2-24
2-6-3 動作モード 2-25

2-7 Windows の状態の監視および変更 2-27
2-7-1 Windows の状態 2-27
2-7-2 Windows の状態の監視 2-27
2-7-3 Windows の状態の変更 2-28
2-7-4 Windows の状態の遷移 2-28

2-8 シャットダウン処理 2-29
2-8-1 電源断および電源ボタン押下時などの処理 2-30
2-8-2 シャットダウン処理の設定 2-31
2-8-3 シャットダウン処理の内容 2-32

2-9 コントローラリセット 2-35

2-1　NY シリーズコントローラの動作の概要

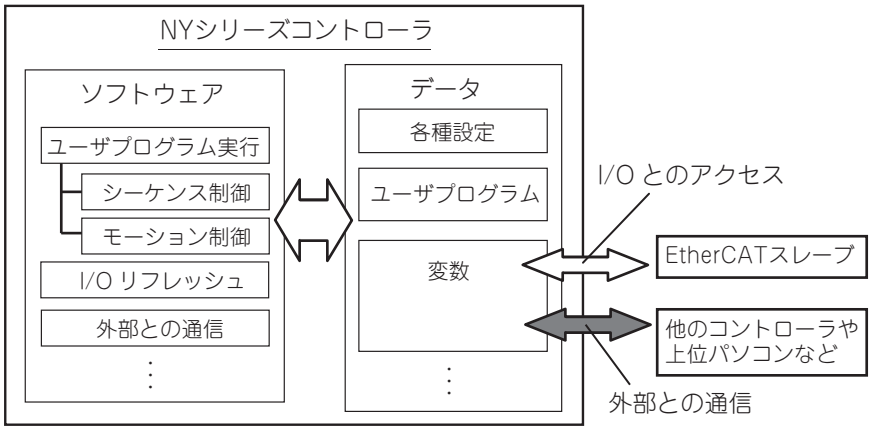
ここでは、NY シリーズコントローラの動作として、コントローラのしくみ、およびコントローラの各状態における動作の概要を説明します。

2-1-1　NY シリーズコントローラのしくみの概要

NY シリーズコントローラは、シーケンス制御およびモーション制御のためのユーザプログラム実行、I/O リフレッシュ、外部との通信などの処理を行います。これらの処理は、コントローラのソフトウェアが行います。

また、コントローラには、各種設定、ユーザプログラム、変数などのデータがあります。コントローラはこれらのデータを使って処理を行います。コントローラと I/O とのアクセス、および外部との通信には、これらのデータのうち、変数を使用します。

コントローラは、ソフトウェアおよび、変数を使った I/O とのアクセスにより、シーケンス制御とモーション制御の両方を実行することができます。



本章では、シーケンス制御、モーション制御、または両方の制御にコントローラを使用する際の基本知識として、以下の内容を説明します。

内容	参照ページ
ソフトウェアのしくみ	P.2-4
I/O とのアクセスのしくみ	P.2-10
シーケンス制御とモーション制御のしくみ	P.2-16
コントローラのデータの概要	P.2-21
コントローラの各状態における動作	P.2-22

参考

- 変数を使った外部との通信のしくみについては、『NY シリーズ 産業用パネル型 PC ／産業用ボックス型 PC ユーザーズマニュアル 内蔵 EtherNet/IP ポート編（SBCD-369）』を参照してください。
- Windows 側から NY シリーズコントローラ側の変数にアクセスすることが可能です。アクセス方法については、『NY シリーズ 産業用パネル型 PC ／産業用ボックス型 PC ユーザーズマニュアル セットアップ編（SBCA-441）』を参照してください。

2-1-2　NY シリーズコントローラの各状態における動作の概要

NY シリーズコントローラは、異常の発生や、動作モードの切替などにより、各状態に遷移します。NY シリーズコントローラの状態遷移により、ユーザプログラム実行、I/O リフレッシュ、外部との通信の処理の動作が変化します。

🕮「2-6　コントローラの各状態における動作」（P.2-22）では、コントローラの各状態における、各処理の動作を説明します。

2-2　ソフトウェアのしくみ

ここでは、NY シリーズのコントローラ機能のソフトウェア構成および、ソフトウェアの動作について説明します。

2-2-1　ソフトウェア構成

コントローラ機能のソフトウェアは、4 つの機能から構成されます。構成する機能の単位を、「機能モジュール」と呼びます。
各機能モジュールと、その処理を以下に示します。

機能モジュール名	処理
PLC 機能モジュール	タスクのスケジューリング、各機能モジュールへの指示、イベントログ、ユーザプログラムの実行、仮想 SD メモリカードサービス、データトレース処理などを行います。
モーション制御機能モジュール	モーション制御処理を行います。*1
EtherCAT マスタ機能モジュール	EtherCAT スレーブの I/O リフレッシュ*2、EtherCAT メッセージ通信*3 など、EtherCAT マスタとして EtherCAT スレーブとの通信を行います。
EtherNet/IP 機能モジュール	タグデータリンク処理、内蔵 EtherNet/IP ポートサービスなど、EtherNet/IP 通信の処理を行います。

*1. モーション制御命令から与えられた目標値（位置、速度）などの指示に従ってモーション演算を実行し、指令値出力、状態管理および情報取得を EtherCAT マスタを介して実行します。
*2. プロセスデータ通信（以降、PDO 通信と省略することがあります。）によって行います。PDO 通信とは、マスタとスレーブの間を、一定の周期でサイクリックにデータ交換する通信方式です。
*3. EtherCAT マスタとして、EtherCAT スレーブとの通信を行います。

2-2-2　ソフトウェアの動作

ソフトウェアが行うコントローラの処理には、処理が実行されるコントローラの状態、および処理の実行条件によって、以下の 4 種類があります。

コントローラの処理の種類	処理が実行されるコントローラの状態	実行条件	処理の例
起動処理	起動中	電源投入時だけ実行されます。	電源投入時の自己診断
タスクで実行される処理	正常運転中および異常状態	割り付けられたタスク内で実行されます。タスクには、周期的に繰り返し実行されるものと、指定した条件が成立したときに 1 回だけ実行されるものがあります。	ユーザプログラム実行、I/O リフレッシュ
タグデータリンクサービス		ハードウェアや外部からの要求があったときにだけ実行されます。	タグデータリンク
システムサービス			仮想 SD メモリカードサービス

コントローラの状態については、📖「2-6-1　コントローラの状態」（P.2-22）を参照してください。

ここでは、各処理の動作を説明します。

起動処理

起動処理は、電源投入時だけ実行されます。
起動処理では、以下の処理を実行します。

処理	内容
電源投入時の自己診断	「電源部異常」、「CPU リセット」、「CPU 異常（WDT 異常）」を検出します。*1
データチェック	電源断保持属性を持つ変数の値が電源断中に保持できなかった場合、システム定義変数「_RetainFail」（電源断保持失敗フラグ）を TRUE とします。
電源投入および電源断のイベント記録	電源投入および電源断のイベントを記録します。

*1.「電源部異常」、「CPU リセット」、「CPU 異常（WDT 異常）」については、📖『NY シリーズ トラブルシューティングマニュアル（SBCA-438）』を参照してください。

タスクで実行される処理

● タスクで実行される処理の種類

タスクでは、以下の処理が実行されます。

処理	内容
I/O リフレッシュ	EtherCAT スレーブとデータ入出力を行います。
ユーザプログラム実行	シーケンス制御のユーザプログラムを実行します。また、モーション制御に指令を発行ます。
モーション制御	ユーザプログラムの指令を受けて、モーション制御を実行します。
システム共通処理	データトレース処理、タグデータリンク処理などを実行します。

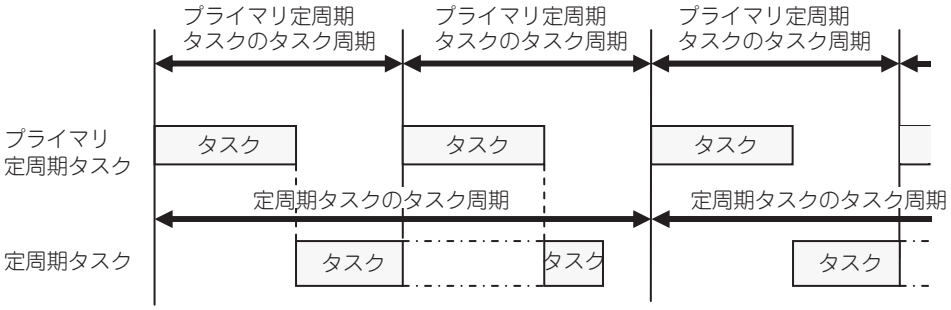
タスクで実行される処理の詳細は、🕮「5-3-3　NY シリーズのタスクの基本動作」（P.5-9）を参照してください。

● タスクの動作

各処理は、タスクに割り付けられます。タスクには、実行優先度と実行条件によって、以下の 3 種類があります。

定周期タスクの種類	実行優先度（値が小さいほど高い）	実行条件	おもな処理内容
プライマリ定周期タスク	4 最も実行優先度が高いタスクです。	周期的に繰り返し実行されます。	I/O リフレッシュ、ユーザプログラム実行、モーション制御
定周期タスク	16,17,18	周期的に繰り返し実行されます。 タスク周期は、プライマリ定周期タスクのタスク周期の整数倍です。	実行優先度により、処理内容が異なります。 実行優先度 16、17、18：ユーザプログラム実行
イベントタスク	8,48	あらかじめ指定した条件が成立したときに 1 回だけ実行されます。	ユーザプログラム実行

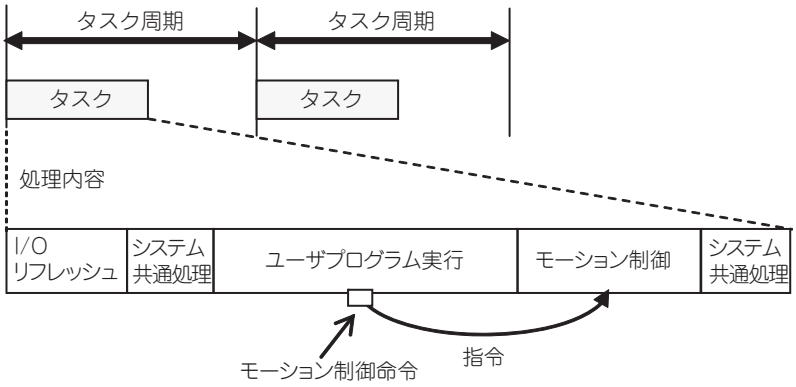
コントローラは、実行優先度が高いタスクを優先的に実行します。下図は、プライマリ定周期タスクと定周期タスクの動作例です。定周期タスクの実行中にプライマリ定周期タスクが実行されると、プライマリ定周期タスクの実行を優先します。



タスク動作の詳細は、🕮「5-2　タスクの概要」（P.5-4）を参照してください。

● タスクで実行される処理の動作

タスクに割り付けられた処理は、タスク内で下図の順番で行われます。プログラムにモーション制御命令が記述されている場合は、タスク内の「ユーザプログラム実行」から、「モーション制御」に指令を発行します。「モーション制御」は指令を元にモーション制御処理を行います。



（注）「モーション制御」の処理は、プライマリ定周期タスクで実行されます。モーション制御命令を記述したプログラムが、定周期タスクに割り付けられている場合、定周期タスクの「ユーザプログラム実行」から、プライマリ定周期タスクの「モーション制御」に指令を発行します。

詳細は、📖「5-9-3　システムとしての入出力応答時間」（P.5-58）を参照してください。

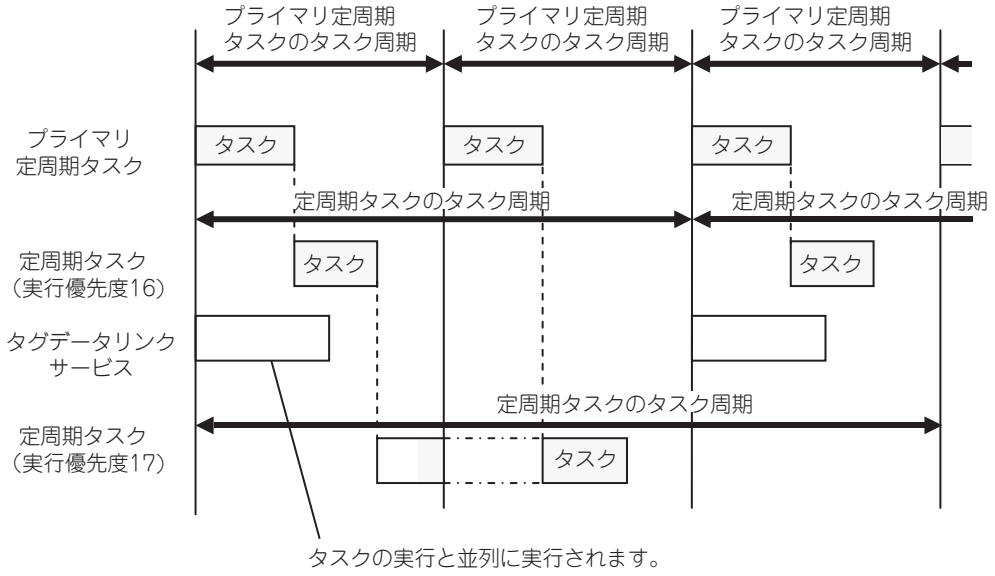
タグデータリンクサービス

● タグデータリンクサービスの処理

タグデータリンクサービスとは、EtherNet/IP ネットワーク上で、他のコントローラやデバイスとタグを用いて通信する処理のことです。コントローラの内蔵 EtherNet/IP ポートを使用します。

● タグデータリンクサービスの動作

タグデータリンクサービスは、一定周期で実行されます。周期や、1 回の処理の実行時間は、コントローラの形式やタグデータリンク設定によって変動します。
また、タグデータリンクサービスは、タスクの実行と並列に実行されます。



タグデータリンクサービスの詳細は、🕮「5-4　タグデータリンクサービスとシステムサービス」（P.5-25）を参照してください。

システムサービス

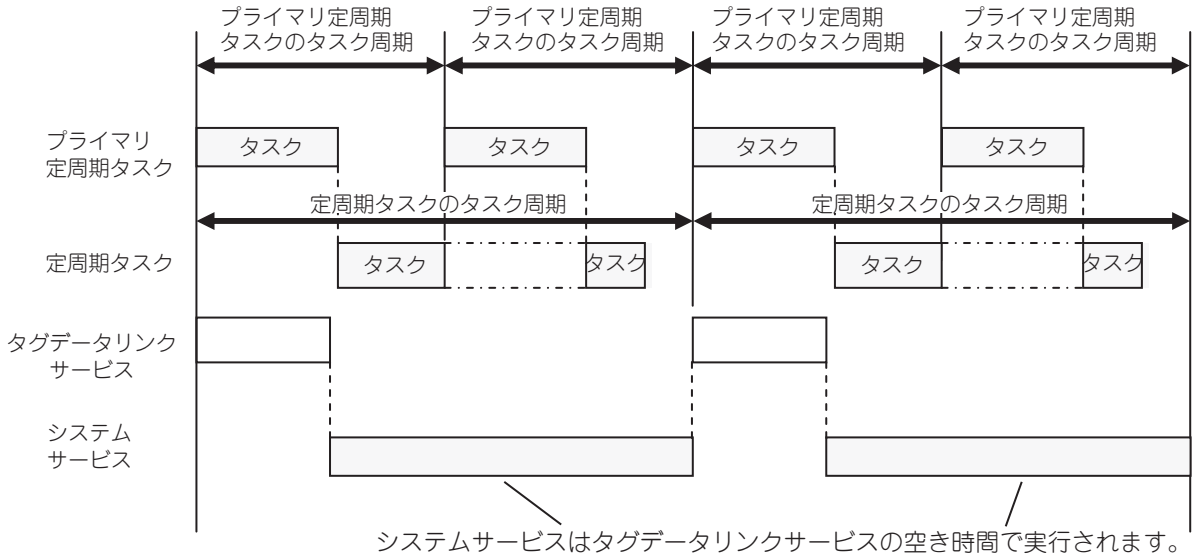
● システムサービスの種類

システムサービスには以下の処理があります。

処理	内容
内蔵 EtherNet/IP ポートサービス	• 周辺ツール、表示器、上位パソコン、または他コントローラなどからのサービス要求（CIP など）に対する処理 • 通信命令（CIP、ソケット）の実行
内蔵 EtherCAT ポートサービス	• EtherCAT メッセージ通信の実行
仮想 SD メモリカードサービス	• FTP クライアントからのアクセス • Sysmac Studio からの SD メモリカード操作 • SD メモリカード命令の実行
自己診断処理	• ハードウェアの異常検知 • Windows のステータス監視

● システムサービスの動作

NY シリーズコントローラでは、ハードウェアやコントローラ外部からの要求があった場合、システムサービスは他の処理と並列して実行されます。タスクの実行などの影響を受けずに、いつでも実行されます。ただし、タグデータリンクサービスの実行中は、システムサービスは実行されません。



システムサービスの詳細は、📖「5-4　タグデータリンクサービスとシステムサービス」（P.5-25）を参照してください。

2-3　I/O とのアクセスのしくみ

NY シリーズコントローラは、変数を使って I/O とアクセスします。ここでは、変数と I/O とのアクセスのしくみについて説明します。
本マニュアルでは、EtherCAT スレーブを I/O として扱います。タグデータリンクを使った、他のコントローラとのアクセスについては、🕮『NY シリーズ 産業用パネル型 PC ／産業用ボックス型 PC ユーザーズマニュアル 内蔵 EtherNet/IP ポート編 （SBCD-369)』を参照してください。

2-3-1　変数の種類

NY シリーズコントローラでは、ユーザプログラムから、変数を介して I/O やコントローラ内部などにアクセスします。
変数は、ユーザによる属性の設定可否およびアクセス対象により以下のように分類されます。

変数			属性の設定可否	アクセス対象
ユーザ定義変数			すべての属性が設定可能	コントローラ内部
準ユーザ定義変数	デバイス変数	EtherCAT スレーブデバイス変数	一部の属性が設定可能[*1]	EtherCAT スレーブ[*2]
	カムデータ変数			サーボドライバ、エンコーダ入力スレーブ、コントローラ内部
システム定義変数	PLC システム変数		すべての属性が設定不可	コントローラ内部
	モーション制御システム変数	MC 共通変数		サーボドライバ、エンコーダ入力スレーブ、コントローラ内部
		軸変数		
		軸グループ変数		
	EtherNet/IP システム変数			内蔵 EtherNet/IP ポート
	EtherCAT マスタシステム変数			内蔵 EtherCAT マスタポート

*1. 設定可能な属性については、🕮「デバイス変数の属性」（P.3-7）を参照してください。
*2. EtherCAT スレーブには、EtherCAT スレーブターミナル上の NX ユニットも含まれます。

ユーザ定義変数

すべての属性がユーザによって定義される変数です。ユーザ定義変数の詳細は、🕮「6-3　変数」（P.6-27）を参照してください。

準ユーザ定義変数

変数の属性の一部が設定可能な変数です。特定の対象のデータにアクセスするために使用します。準ユーザ定義変数は、アクセス対象により、デバイス変数とカムデータ変数に分類されます。

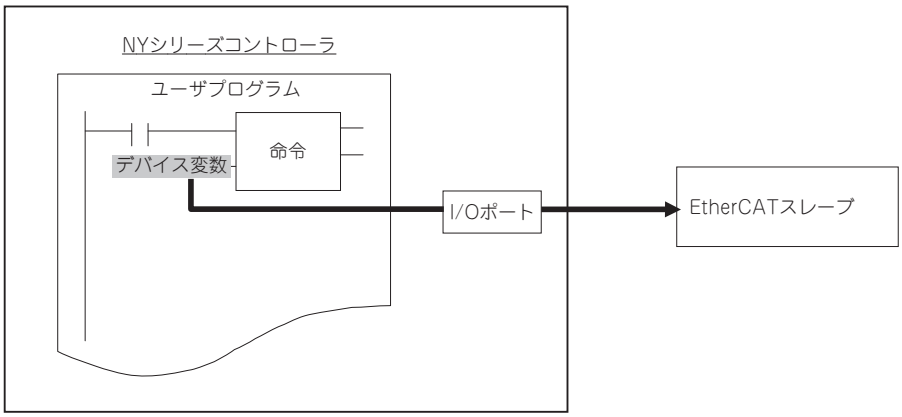
● デバイス変数

デバイス変数とは、デバイスのデータにアクセスする変数です。デバイスとは、コントローラから I/O リフレッシュをする対象の総称です。EtherCAT スレーブを示します。
デバイスおよび、デバイスのアクセス対象のデータにより、以下の種類があります。

デバイス変数の種類	デバイス	アクセス対象のデータ
EtherCAT スレーブデバイス変数	EtherCAT スレーブ	EtherCAT 上のスレーブのプロセスデータ*1

*1. マスタとスレーブ間でプロセスデータ通信周期ごとに交換する I/O データです。

デバイス変数は、I/O ポートを介して EtherCAT スレーブにアクセスします。I/O ポートとは、デバイスとのアクセス用の論理ポートです。



I/O ポートおよびデバイス変数の詳細は、🕮「3-3-1　I/O ポート」（P.3-5）を参照してください。

● カムデータ変数

モーション制御で、カムテーブルのデータにアクセスする変数です。
詳細は、🕮『NY シリーズ 産業用パネル型 PC ／産業用ボックス型 PC ユーザーズマニュアル モーション制御編（SBCE-379）』を参照してください。

システム定義変数

システム定義変数とは、NY システムによって、名称などのすべての属性があらかじめ定義されており、かつ特定の機能が割り付けられている変数です。名称などの属性をユーザが変更することはできません。
特定の機能は機能モジュールごとにあるため、機能モジュールごとにシステム定義変数が存在します。システム定義変数の種類は以下のとおりです。

機能モジュール	システム定義変数の種類
PLC 機能モジュール	PLC システム変数
モーション制御機能モジュール	モーション制御システム変数
EtherNet/IP 機能モジュール	EtherNet/IP システム変数
EtherCAT マスタ機能モジュール	EtherCAT マスタシステム変数

さらに、モーション制御システム変数は、モーション制御機能の内容によって以下に分類されます。

モーション制御システム変数	機能の内容
MC 共通変数	モーション制御機能モジュール共通処理
軸変数	「軸」の制御
軸グループ変数	軸グループ[*1] の制御

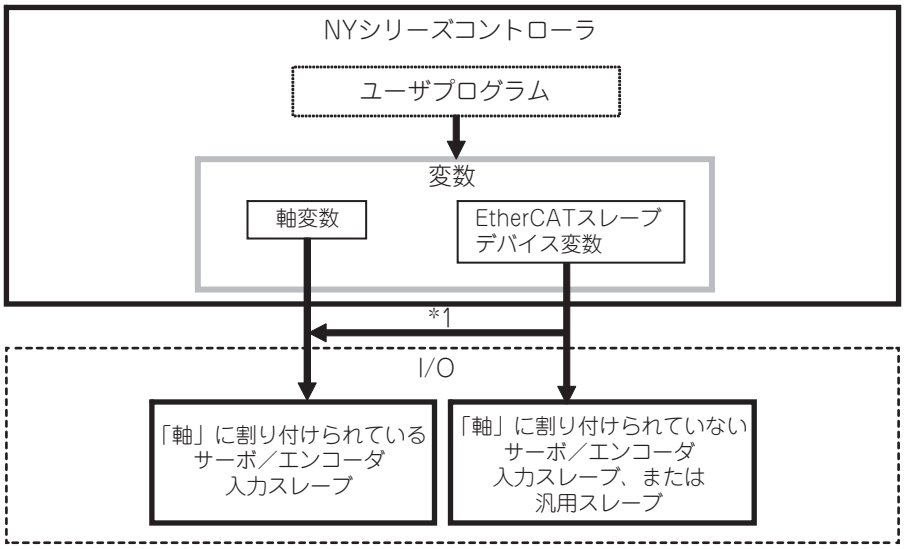
*1. 軸グループとは、複数の「軸」をまとめたグループです。補間動作に使用します。

システム定義変数の詳細については、🕮「A-3　システム定義変数一覧」（P.A-22）を参照してください。

2-3-2　変数と I/O のアクセス

NY シリーズコントローラは、ユーザプログラムで変数を使用します。変数は、割り付けられた I/O のデータとアクセスします。NY シリーズコントローラでは、以下のように I/O と変数を割り付けます。

I/O		変数
EtherCAT スレーブ	EtherCAT スレーブ	EtherCAT スレーブデバイス変数
	「軸」に割り付けられている EtherCAT スレーブ	軸変数または EtherCAT スレーブデバイス変数



EtherCAT スレーブとのアクセス

EtherCAT スレーブとのアクセス方法は、EtherCAT スレーブの種類によって異なります。

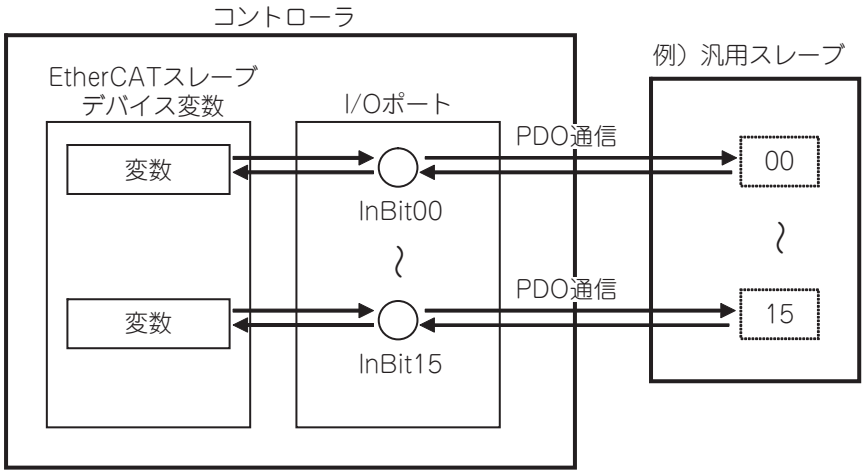
EtherCAT スレーブの種類	アクセス方法
• サーボ／エンコーダ入力スレーブ • 汎用スレーブ	EtherCAT スレーブデバイス変数により I/O ポートを介してアクセス
「軸」に割り付けられているサーボ／エンコーダ入力スレーブ	軸変数からダイレクトにアクセス*1

*1. サーボドライバの場合は 1 台のサーボモータ、エンコーダ入力スレーブの場合は 1 個のカウンタを、1 つの「軸」として軸変数に割り付けて使用します。

（注）「軸」に割り付けることができない EtherCAT スレーブを「汎用スレーブ」、「軸」に割り付けることができる EtherCAT スレーブを「サーボ／エンコーダ入力スレーブ」と呼びます。サーボドライバ／エンコーダ入力スレーブについては、🕮『NY シリーズ 産業用パネル型 PC ／産業用ボックス型 PC ユーザーズマニュアル モーション制御編（SBCE-379）』を参照してください。

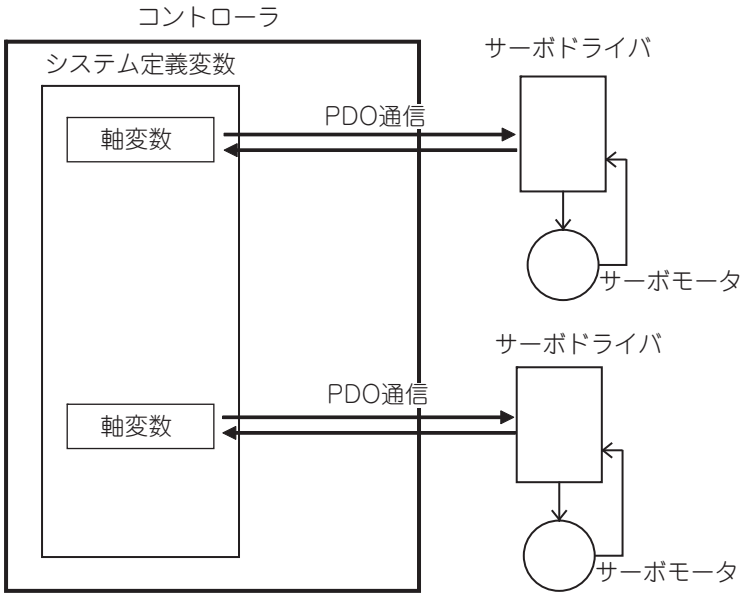
●「軸」に割り付けられていないサーボ／エンコーダ入力スレーブ、または汎用スレーブとのアクセス

EtherCAT スレーブデバイス変数から I/O ポートを介してアクセスします。I/O ポートからのアクセスは、PDO 通信で行われます。



●「軸」に割り付けられているサーボ／エンコーダ入力スレーブとのアクセス

「軸」に割り付けられているサーボ／エンコーダ入力スレーブに、軸変数がダイレクトにアクセスします。軸変数からのアクセスは、PDO 通信で行われます。
たとえば、サーボドライバを使ってサーボモータを制御する場合、軸変数に割り付けられたサーボモータに対する制御の指示が、サーボドライバに送信されます。サーボモータからのフィードバックは、軸変数を使ってサーボドライバからコントローラに送信されます。



軸変数の詳細は、🕮「3-5-2　軸変数／軸グループ変数」（P.3-11）を参照してください。

使用上の注意

「軸」に割り付けられているサーボ／エンコーダ入力スレーブの I/O ポートに、デバイス変数を割り付けることが可能です。
デバイス変数を割付可能な I/O ポートは、以下のいずれかの条件を満たすものです。

- R/W 属性の値が、R（読出専用）である。
- R/W 属性の値が、W（書込専用）であり、Sysmac Studio の［軸基本設定］画面の［詳細設定］で、プロセスデータの設定が＜割付なし＞である。

参考

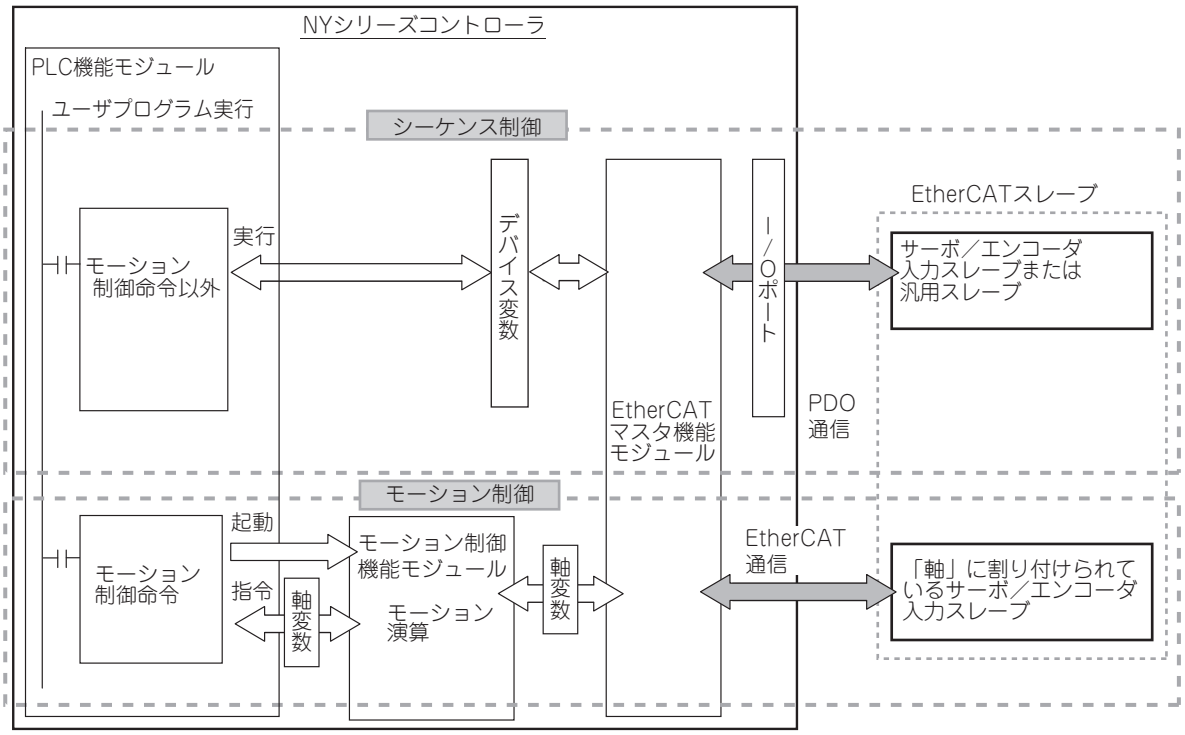
EtherCAT 通信には、PDO 通信と SDO 通信があります。サーボモータの位置制御など、一定の制御周期で入出力データ更新を行う指令に対しては PDO 通信を使用します。パラメータ転送など、指定されたタイミングでデータの読出や書込を行う指令には SDO 通信を使用します。詳細は、🕮『NY シリーズ 産業用パネル型 PC ／産業用ボックス型 PC ユーザーズマニュアル 内蔵 EtherCAT ポート編（SBCD-368）』を参照してください。

2-4　シーケンス制御とモーション制御のしくみ

ここでは、NY シリーズコントローラのシーケンス制御とモーション制御のしくみについて説明します。

2-4-1　制御の概要

NY シリーズコントローラでは、シーケンス制御とモーション制御を実行することができます。
シーケンス制御は、ユーザプログラムに記述されたモーション制御命令以外を実行し、EtherCAT スレーブを制御します。制御には、PLC 機能モジュール、EtherCAT マスタ機能モジュールを用います。
モーション制御は、ユーザプログラムに記述されたモーション制御命令を実行し、「軸」に割り付けられている EtherCAT 上のサーボドライバ／エンコーダ入力スレーブを制御します。制御には、PLC 機能モジュール、モーション制御機能モジュール、EtherCAT マスタ機能モジュールを用います。



参考

制御のしくみからみた命令の分類
制御のしくみ上、命令には大別して以下の 2 種類があります。

命令の分類	定義
モーション制御命令以外（シーケンス制御）	PLC 機能モジュール内のユーザプログラム実行で、処理が完結する命令。
モーション制御命令	PLC 機能モジュール内のユーザプログラム実行により起動され、モーション制御機能モジュールに対して指示を与える命令。 例)MC_Home（原点復帰）、MC_Move（位置決め）、MC_CamIn（カム動作開始）などのモーション制御関連の各種操作

モーション制御命令の詳細は、🕮『NY シリーズ コマンドリファレンスマニュアル モーション編（SBCE-380)』を参照してください。モーション制御命令以外の命令の詳細は、🕮『NY シリーズ コマンドリファレンスマニュアル 基本編（SBCA-437)』を参照してください。

使用上の注意

「軸」に割り付けられているサーボ／エンコーダ入力スレーブの I/O ポートに、デバイス変数を割り付けることが可能です。
デバイス変数を割付可能な I/O ポートは、以下のいずれかの条件を満たすものです。

- R/W 属性の値が、R（読出専用）である。
- R/W 属性の値が、W(書込専用)であり、Sysmac Studio の [軸基本設定] 画面の [詳細設定] で、プロセスデータの設定が < 割付なし > である。

「軸」に割り付けられているサーボ／エンコーダ入力スレーブの I/O ポートにデバイス変数を割り付ける場合、以下の操作を行うとデバイス変数の割付が解除されます。ご注意ください。

(1) Sysmac Studio Ver.1.09 以降で、「軸」に割り付けられているサーボ／エンコーダ入力スレーブの I/O ポートにデバイス変数を割り付ける。

(2) プロジェクトデータを保存する。

(3) そのプロジェクトデータを、Sysmac Studio Ver.1.08 以前で開く。

2-4-2　シーケンス制御のしくみ

シーケンス制御の制御対象のデバイスにより、しくみが異なります。以下に、シーケンス制御のしくみとして、機能モジュールの動作、制御周期を示します。

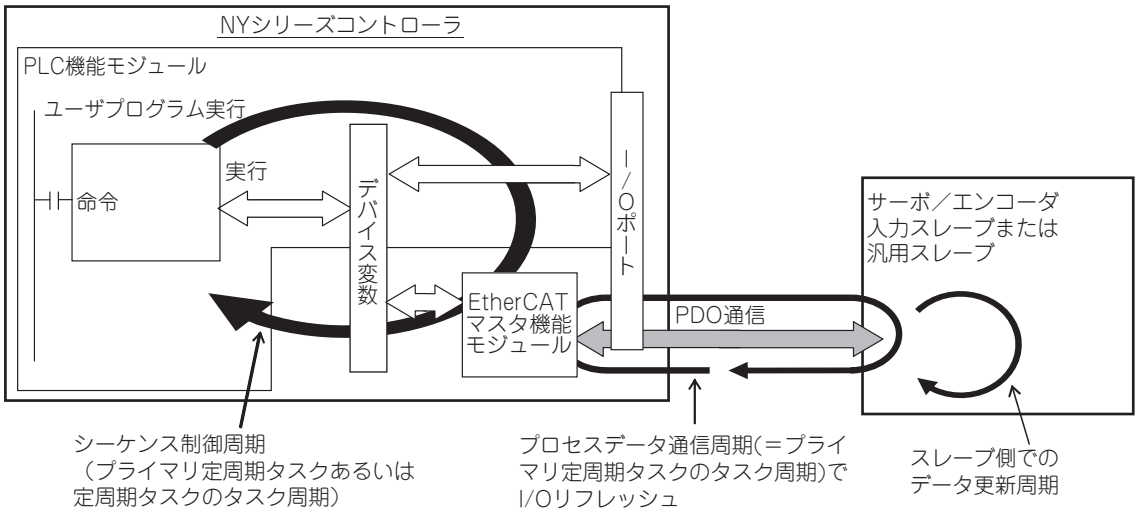
デバイス	シーケンス制御のしくみ	
	機能モジュールの動作	制御周期
サーボ／エンコーダ入力スレーブ、または汎用スレーブ	• PLC 機能モジュールは、ユーザプログラムを実行し、デバイス変数を更新します。 • EtherCAT マスタ機能モジュールは、デバイス変数を使って I/O ポートを介してスレーブとデータ交換します。	プログラムを割り付けたタスクのタスク周期（プライマリ定周期タスクまたは定周期タスクのタスク周期）*1

*1. スレーブ側のデータ更新周期は、スレーブの設定に依存します。

サーボ／エンコーダ入力スレーブ、または汎用スレーブは、プロセスデータ通信周期、つまり、プライマリ定周期タスクのタスク周期で I/O リフレッシュされます。しかし、ユーザプログラムの実行およびデバイス変数の更新は、プログラムを割り付けたタスクのタスク周期で行われます。このため、スレーブの値がデバイス変数に反映され、制御されるのは、プログラムを割り付けたタスクのタスク周期となります。

スレーブをプロセスデータ通信周期で制御したい場合、スレーブを制御するプログラムをプライマリ定周期タスクに割り付ける必要があります。

詳細は、📖「5-9-3　システムとしての入出力応答時間」（P.5-58）を参照してください。



2-4-3　モーション制御のしくみ

モーション制御のしくみとして、機能モジュールの動作、制御周期を示します。

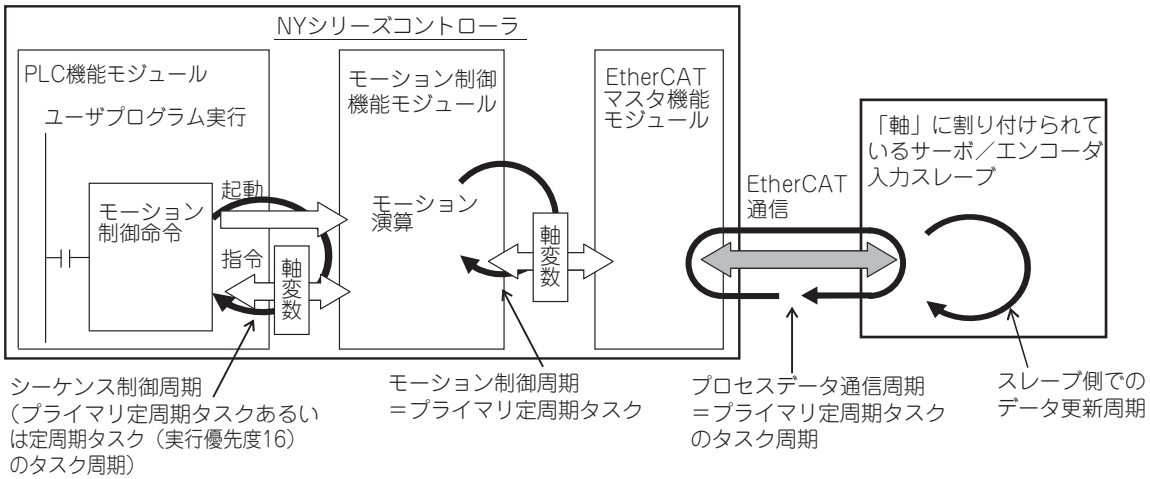
● 機能モジュールの動作

- PLC 機能モジュールは、ユーザプログラム上のモーション制御命令を起動し、モーション制御機能モジュールに対してモーション制御の指令を発行します。この指令には軸変数を使用します。
- モーション制御機能モジュールは、PLC 機能モジュールの指令を元にモーション演算を行い、演算結果の指令値で軸変数の値を更新します。
- EtherCAT マスタ機能モジュールは、軸変数の指令値をサーボドライバなどに EtherCAT 通信で送信します。

● 制御周期

モーション制御周期は、プライマリ定周期タスクのタスク周期となります。
モーション制御機能モジュールのモーション演算の周期は、プライマリ定周期タスクのタスク周期で実行されます。また、制御対象の「軸」に割り付けられているサーボ／エンコーダ入力スレーブとは、プロセスデータ通信周期でデータ交換します。プロセスデータ通信周期は、プライマリ定周期タスクに同期します。
これにより、モーション制御周期は、プライマリ定周期タスクのタスク周期となり、複数の軸を完全に同期して動かすことができます。
ただし、以下の制約があります。

- モーション制御命令の起動およびモーション制御の指令の発行は、シーケンス制御周期で実行されます。
- EtherCAT スレーブ側のデータ更新周期は、スレーブの設定に依存します。



参考

- モーション制御機能モジュールから制御する EtherCAT スレーブは、Sysmac Studio で「軸」への割付設定を行います。これにより、PLC 機能モジュールは、ユーザプログラムの「モーション制御命令」を使用して、モーション制御機能モジュールへ指令を出すことができます。さらに、モーション制御機能モジュールから軸変数を介して情報を取得することができます。
- モーション制御命令が記述されているプログラムを割り付けるタスクの種類によって、モーション制御のシステムとしての入出力応答時間が異なります。詳細は、🕮「5-9-3　システムとしての入出力応答時間」（P.5-58）を参照してください。

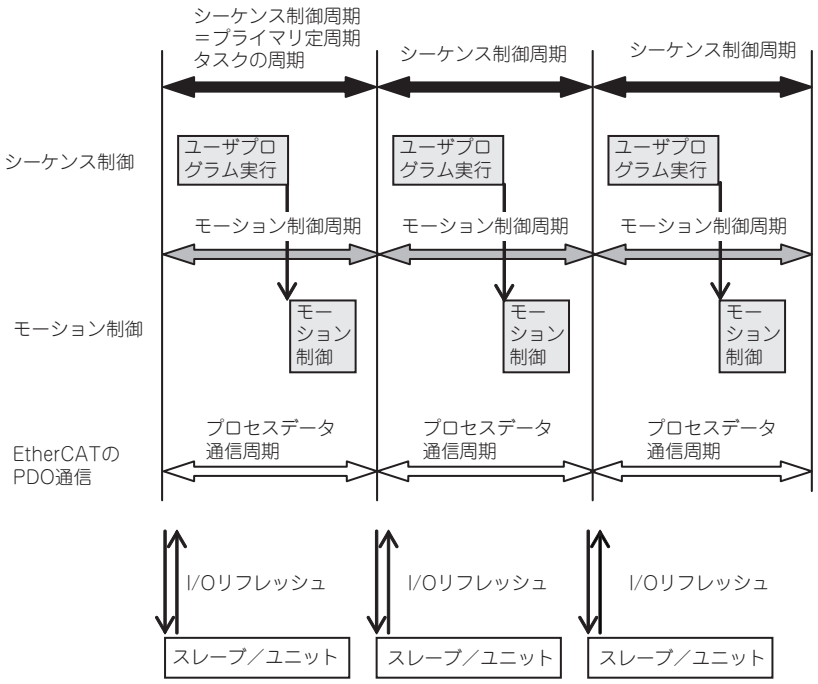
2-4-4　シーケンス制御とモーション制御の同期

シーケンス制御の制御周期は、プログラムや I/O リフレッシュを割り付けたタスクのタスク周期となります。一方、モーション制御はプライマリ定周期タスクのタスク周期で常時実行されます。また、モーション制御に使用する EtherCAT スレーブのプロセスデータ通信周期は、プライマリ定周期タスクに同期します。

NY シリーズコントローラでは、プライマリ定周期タスクの周期で軸および軸グループのモーション制御とプロセスデータ通信が実行されます。

そのため、プライマリ定周期タスクにシーケンス制御のプログラムを割り付けると、シーケンス制御とモーション制御の制御周期、および EtherCAT のプロセスデータ通信周期を、同期させることができます。

プログラムをプライマリ定周期タスクに割り付けた場合の図を以下に示します。下図では、シーケンス制御周期、モーション制御周期、EtherCAT のプロセスデータ通信周期がすべて同期しています。



2-5　コントローラ内のデータの概要

NY シリーズコントローラには、各種設定、ユーザプログラム、現在値などのデータがあります。主なデータとその内容を以下に示します。

NY シリーズコントローラのすべてのデータについては、🕮「A-5　コントローラ内データの属性」（P.A-80）を参照してください。

データ種類			内容
各種設定	EtherCAT 構成	EtherCAT スレーブ構成	EtherCAT スレーブの構成情報。
		EtherCAT マスタ設定	EtherCAT マスタの通信周期などのパラメータ設定 。
	ユニット構成・ユニット設定		コントローラが認識するためのユニットの構成情報、および高機能ユニットの初期設定情報。
	I/O マップ		ユニット構成から自動生成された I/O ポートの変数への割付情報。
	コントローラ設定	動作設定	電源投入時の動作モード、セキュリティなどの設定や、システムサービス設定。
		内蔵 EtherNet/IP ポート設定	EtherNet/IP ポートの以下の設定。 TCP/IP 設定、Ethernet 設定、DHCP 設定、DNS 設定、FTP 設定、SNMP 設定
	モーション制御設定		「軸」／「軸グループ」の軸変数／軸グループ変数、およびモーション制御パラメータ設定。
	カムデータ設定		モーション制御命令のカム動作で使用する位相／変位データから成るカムテーブル。
	イベント設定		ユーザ定義の異常／情報の設定。
	タスク設定		タスクの種類や数、タスクの実行条件、タスク名称、タスク内で実行するプログラムなどの設定。
	データトレース設定		データトレースのトリガ条件などの設定。
	タグデータリンクテーブル		EtherNet/IP でのタグデータリンク設定。
	コントローラ名称		コントローラの名称。
	操作権限認証		Sysmac Studio がコントローラに対して操作するための権限パスワード。
	内蔵時計	現在時刻	コントローラ側の時計の時刻。
		タイムゾーン設定	コントローラ側の時計のタイムゾーン設定。
ユーザプログラム	POU（プログラム構成単位）		プログラム、ファンクション、ファンクションブロックの定義。ローカル変数テーブルおよびその初期値を含む。
	データ	データ型	データ型の定義。
		グローバル変数	グローバル変数の属性情報。初期値属性や保持属性など。
現在値	変数の値		変数の値。
その他	イベントログ		コントローラの異常履歴、および電源 ON/OFF や運転開始といった異常以外の事象の履歴。
	絶対値エンコーダ原点位置オフセット		モーション制御での、絶対値エンコーダ付きのサーボドライバの現在位置を復元するための、原点確定後の指令位置と、絶対値エンコーダから読み出した絶対値データとの差。

2-6　コントローラの各状態における動作

NY シリーズコントローラの各状態における、ユーザプログラム実行、I/O リフレッシュ、外部との通信の処理の動作を説明します。さらに、NY シリーズコントローラの状態の 1 つである「正常運転中」においてユーザプログラムの実行状態を切り替える、動作モードについても説明します。

2-6-1　コントローラの状態

NY シリーズコントローラには、「起動中」、「正常運転中」、「異常状態」の 3 つの状態があります。各状態の定義は以下のとおりです。

状態	定義
起動中	ソフトウェアが起動処理を実行している状態
正常運転中	タスクで実行される処理またはシステムサービスをソフトウェアが実行し、コントローラ異常が発生していない状態
異常状態	タスクで実行される処理またはシステムサービスをソフトウェアが実行し、コントローラ異常が発生している状態

さらに、「正常運転中」には、「プログラムモード動作中」、「運転モード動作中」、「ダウンロード中」の 3 つの状態があります。「正常運転中」のコントローラは、ユーザの操作によって各状態に遷移します。各状態の定義は、以下のとおりです。

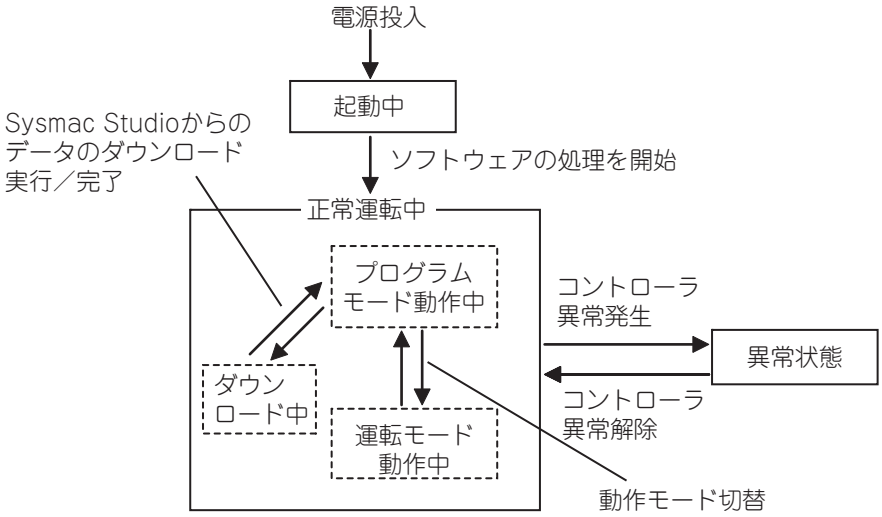
状態	定義
プログラムモード動作中	動作モードがプログラムモードである状態
運転モード動作中	動作モードが運転モードである状態
ダウンロード中	Sysmac Studio からデータのダウンロードを実行している状態

（注） プログラムモードと運転モードについては、「2-6-3　動作モード」（P.2-25）を参照してください。

● コントローラの状態遷移

電源投入後、コントローラは「起動中」となります。コントローラは、「起動中」になってから 10 ～ 20 秒経過後に、ソフトウェアの処理を開始し、「正常運転中」となります。「正常運転中」にコントローラ異常が発生すると「異常状態」となります。コントローラ異常が解除されると、「正常運転中」に戻ります。

「起動中」から「正常運転中」になると、コントローラは、コントローラ設定に設定されている動作モードにしたがって「プログラムモード動作中」または「運転モード動作中」となります。その後、動作モードの切替により、「プログラムモード動作中」の場合は「運転モード動作中」に、「運転モード動作中」の場合は「プログラムモード動作中」となります。「プログラムモード動作中」に、Sysmac Studio からデータのダウンロードを実行すると、「ダウンロード中」となります。ダウンロードが完了すると、「プログラムモード動作中」に戻ります。



参考

- コントローラの状態は、産業用パネル型 PC ／産業用ボックス型 PC の状態表示 LED で確認できます。状態表示 LED による確認方法については、🕮『NY シリーズ トラブルシューティングマニュアル（SBCA-438)』を参照してください。
- コントローラの状態が変化するときのデータの動作については、🕮「A-5　コントローラ内データの属性」（P.A-80）を参照してください。
- コントローラの状態が変化するときの変数の値については、🕮「6-3-9　状態変化に応じた変数の値の変化」（P.6-64）を参照してください。

2-6-2　各状態における動作

NY シリーズコントローラの状態によって、ユーザプログラム、I/O リフレッシュ、外部との通信の動作が変化します。「起動中」および「正常運転中」での、各処理の動作を以下に示します。

「異常状態」については、🕮『NY シリーズ トラブルシューティングマニュアル（SBCA-438）』を参照してください。

コントローラの処理	起動中の動作	正常運転中の動作		
		プログラムモード動作中	運転モード動作中	ダウンロード中
ユーザプログラム	停止します。	停止します。	実行します。	停止します。
EtherCAT スレーブの I/O リフレッシュ	停止します。	実行します。		EtherCAT 通信がセーフオペレーショナル状態となります。*1*2
外部との通信	停止します。	実行します。		実行します。*3

*1. 入力の値だけが更新されます。
*2. システム定義変数「_DeviceOutHoldCfg（デバイス出力保持設定）」で、デバイス出力保持設定を有効（16#A5A5）に設定した場合、I/O リフレッシュを実行します。デバイス出力保持設定については、🕮「デバイス出力保持設定」（P.6-65）を参照してください。システム定義変数「_DeviceOutHoldCfg（デバイス出力保持設定）」は、コントローラ Ver.1.14 以降と Sysmac Studio Ver.1.18 以降の組み合わせで使用できます。
*3. タグデータリンクは継続しますが、タグデータリンクの値は更新されません。出力タグの値は、ダウンロード開始前の値に保持されます。入力タグの値は変数に反映されません。

● I/O の出力接点の値

I/O リフレッシュ処理による、各状態での I/O の出力接点の値は以下のとおりです。

I/O の出力接点	起動中の動作	正常運転中の動作		
		プログラムモード動作中	運転モード動作中	ダウンロード中
EtherCAT スレーブの出力接点	各スレーブの設定に従います。*1	EtherCAT スレーブデバイス変数の値となります。		各スレーブの設定に従います。*2 *3

*1. 電源投入後、EtherCAT 通信が開始するまでのスレーブの設定については、各スレーブのマニュアルを参照してください。
*2. ダウンロード完了後、EtherCAT スレーブは初期化処理を行います。初期化処理中の出力は各スレーブの設定に従います。
*3. システム定義変数「_DeviceOutHoldCfg（デバイス出力保持設定）」で、デバイス出力保持設定を有効（16#A5A5）に設定した場合、動作モード変更時およびダウンロード時でも、デバイス出力を保持します。詳細は、🕮「6-3-9　状態変化に応じた変数の値の変化」（P.6-64）を参照してください。
システム定義変数「_DeviceOutHoldCfg（デバイス出力保持設定）」は、コントローラ Ver.1.14 以降と Sysmac Studio Ver.1.18 以降の組み合わせで使用できます。

変数の初期値属性については、🕮「6-3-8　変数の各属性の説明」（P.6-58）を参照してください。

デバイス出力保持設定の詳細については、🕮「　デバイス出力保持設定」（P.6-65）を参照してください。

参考

動作モード切替によるサーボドライバの動作
モーション制御中に、運転モードからプログラムモードへ切り替えられた場合、各軸パラメータに設定された最大減速度で減速停止します。

EtherCAT スレーブの初期化処理中の動作モード切替
EtherCAT スレーブの初期化処理中に、コントローラの動作モードを運転モードへ変更することができます。この場合、通信が確立したことを確認してからスレーブのデータを制御に使用するようにプログラミングしてください。ユーザプログラムからは、システム定義変数「_EC_PDSlavTbl」（プロセスデータ通信中スレーブテーブル）を使用して、全スレーブのプロセスデータの出力と入力が有効であるかどうかを確認することができます。

2-6-3　動作モード

動作モードとは、動作確認や本運転などの目的にあわせて切り替えるモードです。動作モードには、目的にあわせて、運転モードとプログラムモードがあります。各動作モードでは、ユーザプログラムの実行状態が異なります。以下に、各動作モードの目的とユーザプログラムの実行状態を示します。

動作モード	目的	ユーザプログラムの実行状態
運転モード*1	ユーザプログラムを実行させる試運転や、本運転	実行します。
プログラムモード	I/O 配線チェックなどの、ユーザプログラムを実行させないで行う動作確認	実行しません。

*1. 初期設定では、コントローラが「起動中」から「正常運転中」になると運転モードで動作します。

参考

動作モードの切替により、軸の停止やモーション制御命令の中断などを行います。動作モード切替時のモーション制御機能モジュールの動作は、🕮『NY シリーズ コマンドリファレンスマニュアル モーション編（SBCE-380)』を参照してください。

各動作モードでの Sysmac Studio および表示器からの操作の可否

各動作モードでの、Sysmac Studio および表示器からの主な操作の可否は、以下のとおりです。

操作		運転モード	プログラムモード
Sysmac Studio	同期	不可	可
	オンラインエディット	可	
	強制値リフレッシュ	可	
	変数の値変更	可	
表示器	変数の値変更	可	

動作モード切替時の変数の値の保持

運転モードからプログラムモード、またはプログラムモードから運転モードに切り替えると、変数の保持属性によって、変数の値が以下のようになります。

変数の保持属性	変数の値
非保持	初期値設定ありの場合：　指定した初期値 初期値設定なしの場合：　システム定義の初期値*1
保持	動作モード変更前の値を保持

*1. 変数のデータ型により異なります。🕮「初期値指定が空白のとき」（P.6-61）を参照してください。

コントローラの状態が変化するときの変数の値については、🕮「6-3-9　状態変化に応じた変数の値の変化」（P.6-64）を参照してください。

動作モードの設定および切替

電源投入後の運転開始時、コントローラは、コントローラ設定に従った動作モードで動作します。また、正常運転中は、ユーザが目的にあわせて動作モードを切り替えます。設定および切替は、いずれも Sysmac Studio から行います。

● 電源投入後の運転開始時の動作モードの設定

電源投入後の運転開始時は、「電源投入時モード」に設定した動作モードでコントローラが動作します。コントローラ設定の「動作設定」に「電源投入時モード」として、運転モードまたはプログラムモードを設定します。「電源投入時モード」の設定については、📖「4-2-2　コントローラ設定」（P.4-4）を参照してください。

● 運転中の動作モード切替

動作モードは、Sysmac Studio によって切り替えます。メニューバーの［コントローラ］－［動作モード］から、運転モードまたはプログラムモードを選択します。

使用上の注意

電源投入時の動作モード設定、および動作モードの切替は、設備に影響がないかを確認したうえで行ってください。

動作モードの確認

動作モードは、産業用 PC の RUN LED または Sysmac Studio で確認できます。

● RUN LED での確認

産業用 PC の RUN LED の表示により、動作モードは以下のようになります。

RUN LED の表示	動作モード
消灯	プログラムモード
点灯	運転モード

● Sysmac Studio での確認

動作モードは、Sysmac Studio のコントローラステータスウィンドウで確認できます。以下のコントローラステータスウィンドウでは、「運転モード」を表示しています。



2-7　Windows の状態の監視および変更

NY シリーズでは、コントローラから Windows の状態を監視したり、状態を変更したりすることができます。

2-7-1　Windows の状態

NY シリーズにインストールされている Windows には、以下の 5 つの状態があります。

状態	説明
起動中 (Booting)	Windows が起動処理を実行している状態
稼働中 (Running)	Windows が稼働している状態
シャットダウン中 (ShuttingDown)	Windows がシャットダウン処理を実行している状態
停止中 (Halted)	Windows が停止している状態 (シャットダウン処理が完了し、稼働していない状態)
異常発生中 (Error)	Windows が異常を発生している状態

参考

システム定義変数および命令で、状態を識別することができます。
識別できるのは、「稼働中」、「停止中」、「異常発生中」です。「起動中」、「シャットダウン中」については、状態を識別することはできません。

2-7-2　Windows の状態の監視

Windows の状態の監視には、システム定義変数または命令を使用します。

Windows の状態監視関連のシステム定義変数

システム定義変数では、Windows の状態のうち、「稼働中」、「停止中」、および「異常発生中」の 3 つの状態を表示します。
システム定義変数の詳細は、🕮「機能分類：OS(Windows) 関連」（P.A-54）を参照してください。

変数名	名称	機能	データ型	R/W
_OSRunning	OS 稼働中フラグ	コントローラが OS(Windows) の状態を稼働中と観測した場合、TRUE となります。	BOOL	R
_OSHalted	OS 停止中フラグ	コントローラが OS(Windows) の状態を停止中と観測した場合、TRUE となります。	BOOL	R
_OSErrorState	OS 異常状態フラグ	コントローラが OS(Windows) の状態を異常発生中と判断した場合、TRUE となります。	BOOL	R

Windows の状態監視関連の命令

Windows の状態を監視する命令として、以下のものが用意されています。
命令の詳細は、🕮『NY シリーズ コマンドリファレンスマニュアル 基本編（SBCA-437）』を参照してください。

命令	命令名称	概要
IPC_GetOSStatus	OS ステータス読出	OS(Windows) のステータスを読み出します。読み出す状態は、Windows の状態のうち、「稼働中」、「停止中」、および「異常発生中」の 3 つです。

2-7-3　Windows の状態の変更

Windows の状態を変更するには、命令を使用します。

Windows の状態変更関連の命令

Windows の状態を変更する命令として、以下のものが用意されています。
命令の詳細は、🕮『NY シリーズ コマンドリファレンスマニュアル 基本編（SBCA-437）』を参照してください。

命令	命令名称	概要
IPC_RebootOS	OS 再起動	OS(Windows) を再起動します。
IPC_Shutdown	シャットダウン	OS(Windows) にシャットダウン要求を通知します。

2-7-4　Windows の状態の遷移

産業用 PC を起動すると、Windows は「起動中」となります。起動処理が終了すると「稼働中」となります。
稼働中に IPC_Shutdown 命令、UPS の電断検知などによる給電の停止、または電源ボタンの押下が発生すると「シャットダウン中」となります。シャットダウン処理が終了すると「停止中」となります。
また、稼働中に IPC_RebootOS 命令を実行すると「シャットダウン中」となります。シャットダウン処理が終了すると「起動中」となり、起動処理が終了すると「稼働中」となります。
「異常発生中」から復旧させる場合も、IPC_RebootOS 命令を実行します。

Windows の状態が、「稼働中」、「停止中」、および「異常発生中」に遷移した場合は、遷移したことを示すコントローライベントが登録されます。
ただし、IPC_Shutdown 命令の実行により、コントローラが Windows にシャットダウン要求を通知してから、〔 シャットダウン処理待ち時間設定 〕の〔OS〕に設定した時間を経過しても Windows が「停止中」に遷移しない場合、コントローライベントは登録されません。

2-8　シャットダウン処理

NY シリーズ産業用 PC は、必ず UPS（無停電電源）を接続して使用してください。UPS を接続することにより、停電などによる不意の電源喪失時でも、データやファイルを損失することなく、正しくシャットダウンすることができます。

UPS での電断の検知、あるいは電源ボタンの押下を示すシステム定義変数が TRUE となった場合に、コントローラのユーザプログラムにて、電源断に備えるための処理を実行することができます。
必要により、次のような処理をプログラミングしてください。

- コントローラのユーザプログラムで使用している保持属性なしの変数を、保持属性ありの変数にコピーする。
- コントローラの仮想 SD メモリカードに対するファイル操作を停止する。
- コントローラで実行中のモーション制御の動作を安全に停止する。

ユーザプログラムでの処理の完了後に、コントローラのシステムが、保持属性ありの変数を保持するなどのシャットダウン処理を実行します。コントローラのシステムがシャットダウン処理を完了すると、コントローラのシステムは、Windows 側にシャットダウン要求を通知します。

Windows 側では、コントローラからの通知をトリガとして、アプリケーションの終了などのシャットダウン処理を行うことができます。

シャットダウン処理の詳細は、「シャットダウン処理の詳細」（P.2-33）を参照してください。

使用上の注意

- UPS を接続していない場合や UPS のバッテリ電圧が低下している場合、電源断時にシャットダウン処理が実行されないため、保持属性ありの変数、絶対値エンコーダ原点位置オフセット、およびイベントログを保持することができません。
- コントローラ側および Windows 側のシャットダウン処理時間より、バックアップ時間が長くなるバッテリ容量の UPS を選定してください。バッテリ時間が短い場合、シャットダウン処理が正常に終了しません。

参考

UPS の接続方法については、🕮『NY シリーズ 産業用パネル型 PC ユーザーズマニュアル ハードウェア編(SBCA-435)』、🕮『NY シリーズ 産業用ボックス型 PC ユーザーズマニュアル ハードウェア編（SBCA-434)』を参照してください。

2-8-1　電源断および電源ボタン押下時などの処理

コントローラの動作中に、電源断、電源ボタンの押下、あるいは IPC_Shutdown 命令の実行が発生した場合のコントローラの処理について説明します。UPS の接続有無や電源ボタンの押下時間によって、処理が異なります。

操作	UPS 接続	コントローラの処理
電源断	UPS 接続あり	UPS からの信号をトリガとして、シャットダウン処理を実行します。*1 *2
	UPS 接続なし、または UPS のバッテリ電圧低下	シャットダウン処理を実行しません。*3
電源ボタン押下（5 秒未満）	－	電源ボタンの押下をトリガとして、シャットダウン処理を実行します。*1 *2
IPC_Shutdown 命令実行	－	コントローラのシステムがシャットダウン処理を実行します。*1
電源ボタン押下（5 秒以上）	－	強制シャットダウンとなります。シャットダウン処理を実行しません。*3

*1. シャットダウン処理の内容については、🕮「2-8-3　シャットダウン処理の内容」（P.2-32）を参照してください。
*2. IPC_Shutdown 命令を実行するか、[シャットダウン処理待ち時間設定] の [ユーザプログラム] に設定した時間が経過すると、コントローラのシステムがシャットダウン処理を実行します。
*3. シャットダウン処理が実行されなった場合に次回電源投入すると、監視情報の「イベントログ保存失敗」および全停止フォールトの「保持変数の現在値保存未実行」が発生します。
また、コントローラ内のデータは、以下の状態となります。
- 保持属性ありの変数、および絶対値エンコーダ原点位置オフセットは、前回の電源投入時の値となります。
- イベントログは、前回の起動時からシャットダウン処理が実行されなかった時点までのイベントログが消失します。



参考

コントローラが動作していない状態で電源ボタンを押下すると、産業用 PC は通常モードで起動します。電源コネクタを外した状態で電源ボタンを押下し、押下したままの状態で電源コネクタを接続して 10 秒経過すると、コントローラ設定の [電源投入時モード] の設定にかかわらず、コントローラはプログラムモードで運転を開始します。この場合、監視情報の「セーフモード」が発生します。
起動と終了については、🕮『NY シリーズ 産業用パネル型 PC ／産業用ボックス型 PC ユーザーズマニュアル セットアップ編（SBCA-441）』を参照してください。

2-8-2　シャットダウン処理の設定

コントローラ設定で、以下の項目を設定します。

設定項目	設定中項目	機能説明	設定値	デフォルト	反映タイミング	運転モード中書替
シャットダウン処理待ち時間設定	ユーザプログラム	電断検知を示す UPS からの信号が ON、または電源ボタンによるシャットダウン指示を受け付けてからコントローラがシャットダウン処理を開始するまでの待ち時間を設定します。	1 ～ 30s	5s	設定転送時	不可
	OS	コントローラから Windows 側にシャットダウン要求を通知してから、Windows 側のシャットダウン完了を検知するまでの待ち時間を設定します。	30 ～ 300s	60s	設定転送時	不可

● ユーザプログラム

電断検知を示す UPS からの信号は、コントローラのユーザプログラムで監視します。UPS からの信号をトリガとし、ユーザプログラムで電源断に備えるための処理を実行します。
コントローラのシステムは、［ユーザプログラム］に設定された時間だけ、ユーザプログラムでの処理の完了を待ちます。そのため、［ユーザプログラム］にはユーザプログラムでの処理時間より長い時間を設定してください。設定した時間が短い場合、ユーザプログラムでの処理が中断されます。

● OS

コントローラのシステムがシャットダウン処理を完了すると、コントローラのシステムは、Windows 側にシャットダウン要求を通知します。Windows 側では、コントローラからの通知をトリガとして、アプリケーションの終了などのシャットダウン処理を行うことができます。
コントローラのシステムは、Windows 側の状態を監視することで、Windows 側のシャットダウンが正常に完了したか検知することができます。
［OS］に設定した時間が経過するまでに、Windows 側がシャットダウン処理を完了すると、「停止中」に遷移したことを示すコントローライベントが登録されます。そのため、［OS］には Windows 側での処理時間より長い時間を設定してください。設定した時間が短い場合、Windows 側での処理が中断されるため、Windows のアプリケーションを正常に終了することができません。
「停止中」に遷移したことを示すコントローライベントが登録されていない場合、［OS］に設定した時間が Windows 側での処理時間より短い可能性があります。

2-8-3　シャットダウン処理の内容

UPS が電断を検知した場合、あるいは電源ボタンが押下された場合、それぞれの状態に対応するシステム定義変数が TRUE となります。これらのシステム定義変数の状態を監視することにより、必要なデータを保持属性ありの変数にコピーしたり、実行中のモーション制御の動作を安全に停止させたりすることが可能となります。システム定義変数の監視や電源断に備えるための処理は、コントローラのユーザプログラムで作成しておきます。
ユーザプログラムでの電源断に備えるための処理の終了後に、IPC_Shutdown 命令を実行してください。IPC_Shutdown 命令の実行により、コントローラのシステムがシャットダウン処理を行います。

シャットダウン処理のトリガとして使用するシステム定義変数

ユーザプログラムで、処理を開始するためのトリガとして使用するシステム定義変数は以下です。

変数名	名称	機能	データ型	R/W
_SelfTest_UPSSignal	UPS 信号検知フラグ	UPS からの一時電断信号を検知した場合、TRUE となります。	BOOL	R
_RequestShutdown	シャットダウン要求フラグ	稼働中に電源ボタンが押下された場合、TRUE となります。	BOOL	R

システム定義変数の詳細は、🕮「機能分類：電源管理関連」（P.A-53）を参照してください。

シャットダウン要求を通知する命令

ユーザプログラムでの処理の完了をコントローラのシステムに通知し、コントローラのシステムから Windows 側にシャットダウン要求を通知する命令として、以下のものが用意されています。

命令	命令名称	概要
IPC_Shutdown	シャットダウン	コントローラのシステムでのシャットダウン処理を開始し、完了後に Windows 側にシャットダウン要求を通知します。

命令の詳細は、🕮『NY シリーズ コマンドリファレンスマニュアル 基本編（SBCA-437）』を参照してください。

シャットダウン処理の詳細

コントローラの「運転モード動作中」、「プログラムモード動作中」、および「起動中」の各状態における、シャットダウン処理の詳細を説明します。

● 運転モード動作中

コントローラのユーザプログラムが実行されている状態です。
運転モード動作中では、コントローラ側で管理している装置の稼働データなどを消失させないように、ユーザプログラムで作成しておいた、システム定義変数の監視や電源断に備えるための処理を実行します。ユーザプログラムでの処理の完了後に、コントローラのシステムがシャットダウン処理を行い、その後、Windows 側でアプリケーションの終了などのシャットダウン処理を行います。
運転モード動作中でのシャットダウン処理の詳細について説明します。

1 **ユーザプログラムで、システム定義変数「_SelfTest_UPSSignal（UPS 信号検知フラグ）」、および「_RequestShutdown（シャットダウン要求フラグ）」を監視してください。**

2 **システム定義変数「_SelfTest_UPSSignal（UPS 信号検知フラグ）」、または「_RequestShutdown（シャットダウン要求フラグ）」が TRUE となった場合に、必要によりユーザプログラムで電源断に備えるための処理を実行してください。**

3 **電源断に備えるための処理の終了後に、ユーザプログラムで IPC_Shutdown 命令を実行してください。**

- IPC_Shutdown 命令を実行すると、コントローラのシステムがシャットダウン処理を開始します。
- 保持属性ありの変数、絶対値エンコーダ原点位置オフセット、およびイベントログなどが保持されます。
- コントローラが Windows にシャットダウン要求を通知し、Windows がシャットダウン処理を開始します。
- IPC_Shutdown命令を実行しなかった場合でも、[シャットダウン処理待ち時間設定]の[ユーザプログラム] に設定した時間が経過すると、コントローラのシステムはシャットダウン処理を開始します。
- Windows のシャットダウンが完了すると、Windows が「停止中」に遷移したことを示すコントローライベントが登録されます。
  ただし、コントローラから Windows にシャットダウン要求を通知してから、[シャットダウン処理待ち時間設定] の [OS] に設定した時間を経過しても Windows が「停止中」に遷移しない場合、コントローライベントは登録されません。
- UPS からの信号をトリガとしてシャットダウン処理を開始した場合は、シャットダウン処理の完了後に、産業用 PC の I/O コネクタの「電源状態出力」が ON します。
- 「電源状態出力」が ON すると、UPS は給電を停止します。

● プログラムモード動作中

コントローラのユーザプログラムが実行されていない状態です。
プログラムモード動作中では、コントローラ側は装置の制御を実行していないため、ユーザプログラムでの電源断に備えるための処理などは必要ありません。
コントローラのシステムのシャットダウン処理の完了後に、Windows 側でアプリケーションの終了などのシャットダウン処理を実行します。
プログラムモード動作中でのシャットダウン処理の詳細について説明します。

- システム定義変数「_SelfTest_UPSSignal（UPS 信号検知フラグ）」、または「_RequestShutdown（シャットダウン要求フラグ）」が TRUE となった場合、コントローラのシステムはすぐにシャットダウン処理を開始します。
- 保持属性ありの変数、絶対値エンコーダ原点位置オフセット、およびイベントログなどが保持されます。
- コントローラがWindows側にシャットダウン要求を通知し、Windows側がシャットダウン処理を開始します。
- Windows 側のシャットダウンが完了すると、Windows 側が「停止中」に遷移したことを示すコントローライベントが登録されます。
  ただし、コントローラから Windows 側にシャットダウン要求を通知してから、[シャットダウン処理待ち時間設定] の [OS] に設定した時間を経過しても Windows 側が「停止中」に遷移しない場合、コントローライベントは登録されません。
- UPS からの信号をトリガとしてシャットダウン処理を開始した場合は、シャットダウン処理の完了後に、産業用 PC の I/O コネクタの「電源状態出力」が ON します。
- 「電源状態出力」が ON すると、UPS は給電を停止します。

● 起動中

コントローラが起動処理を実行している状態です。
起動中では、コントローラ側は装置の制御を実行していないため、ユーザプログラムでの電源断に備えるための処理などは必要ありません。
コントローラのシステムのシャットダウン処理の完了後に、Windows 側でアプリケーションの終了などのシャットダウン処理を実行します。
起動中でのシャットダウン処理の詳細について説明します。

- システム定義変数「_SelfTest_UPSSignal（UPS 信号検知フラグ）」、または「_RequestShutdown（シャットダウン要求フラグ）」が TRUE となった場合、コントローラのシステムはすぐにシャットダウン処理を開始します。
- 保持属性ありの変数、絶対値エンコーダ原点位置オフセット、およびイベントログなどが保持されます。
- コントローラがWindows側にシャットダウン要求を通知し、Windows側がシャットダウン処理を開始します。
- Windows 側のシャットダウンが完了すると、Windows 側が「停止中」に遷移したことを示すコントローライベントが登録されます。
  ただし、コントローラから Windows 側にシャットダウン要求を通知してから、[シャットダウン処理待ち時間設定] の [OS] に設定した時間を経過しても Windows 側が「停止中」に遷移しない場合、コントローライベントは登録されません。
- UPS からの信号をトリガとしてシャットダウン処理を開始した場合は、シャットダウン処理の完了後に、産業用 PC の I/O コネクタの「電源状態出力」が ON します。
- 「電源状態出力」が ON すると、UPS は給電を停止します。

2-9　コントローラリセット

Sysmac Studio、または Industrial PC Support Utility からコントローラリセットを実行することで、コントローラ側だけをリセットすることができます。

● Sysmac Studio からのコントローラリセット

コントローラの動作モードがプログラムモードのときに、コントローラリセットを実行できます。
Sysmac Studio からのコントローラリセットについては、🕮『Sysmac Studio Version 1 オペレーションマニュアル（SBCA-470）』を参照してください。

● Industrial PC Support Utility からのコントローラリセット

コントローラの動作モードがプログラムモードのときに、コントローラリセットを実行できます。
Industrial PC Support Utility からのコントローラリセットについては、🕮『NY シリーズ 産業用パネル型 PC ／産業用ボックス型 PC ユーザーズマニュアル セットアップ編（SBCA-441）』を参照してください。

3

I/O ポートとスレーブ／ユニット構成

本章では、I/O ポートの使い方と、スレーブ／ユニット構成の作成と手順、各機能の割付について説明します。

3-1 スレーブ／ユニット構成の手順 3-2

3-2 EtherCAT スレーブ構成の作成 3-4

3-3 I/O ポートとデバイス変数 3-5
3-3-1 I/O ポート 3-5
3-3-2 I/O ポート名 3-6
3-3-3 デバイス変数 3-6

3-4 ユニットへの変数の割付 3-8
3-4-1 ユニットへの変数の割付方法 3-8
3-4-2 ユニットに割り付けた変数の使用方法 3-9

3-5 「軸」の作成と、サーボドライバ／エンコーダ入力スレーブの割付 3-11
3-5-1 概要 3-11
3-5-2 軸変数／軸グループ変数 3-11
3-5-3 「軸」および軸変数の作成と使用方法 3-13

3-1　スレーブ／ユニット構成の手順

ここでは、スレーブ／ユニット構成の手順を示します。

スレーブ／ユニット構成は、全体の手順の中で、以下の網掛けで示した位置に対応しています。

STEP 1. ソフト設計

- STEP1-1 I/O と処理の設計
- STEP1-2 タスク設計
- STEP1-3 プログラムの設計

↓

STEP 2. ソフトウェア設定とプログラミング

- **STEP2-1 スレーブ／ユニット構成**
- STEP2-2 コントローラ設定
- STEP2-3 プログラミング
- STEP2-4 オフラインデバッグ

↓

STEP 3. 装着とハードウェアの設定

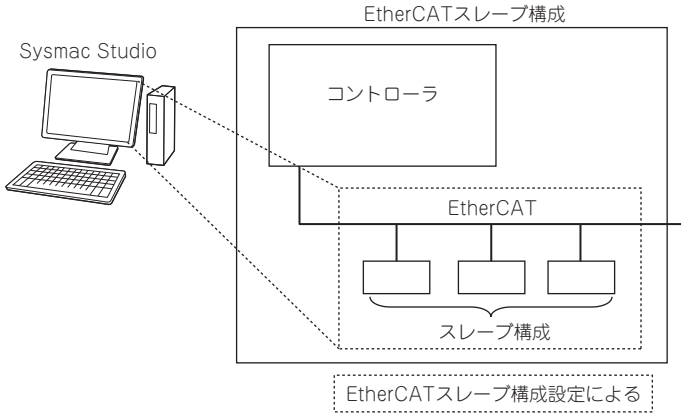
↓

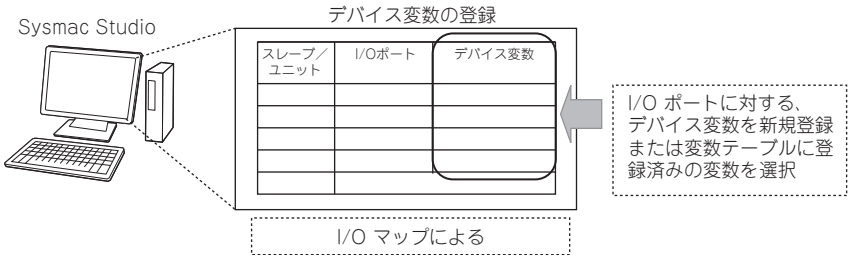
STEP 4. 配線

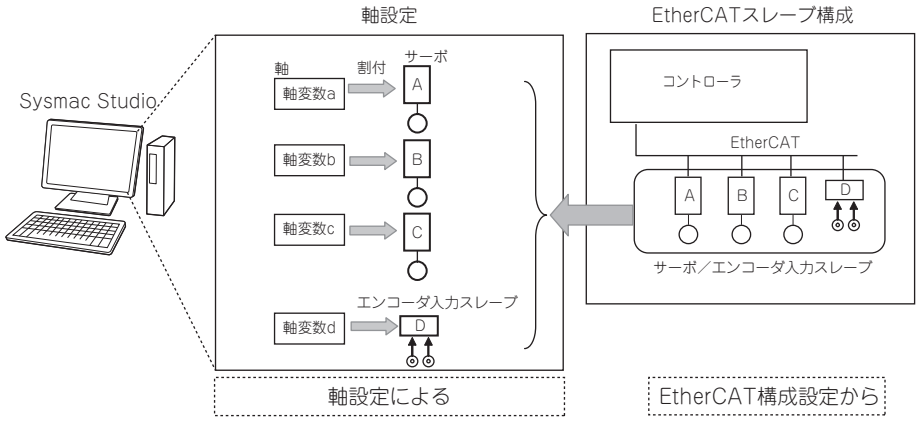
↓

STEP 5. 実機による動作確認と運転

詳細は、🕮「1-4 NY シリーズの使用手順の全体」（P.1-10）を参照してください。

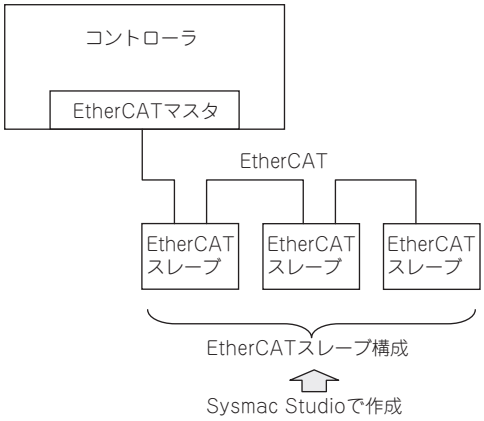
STEP1 EtherCAT スレーブ構成の作成（EtherCAT を使用する場合）	参照
• EtherCAT スレーブ構成の作成 	📖「3-2 EtherCAT スレーブ構成の作成」（P.3-4）

STEP2 I/O ポートへのデバイス変数の割付	参照
• デバイス変数の登録 	📖「2-3-1 変数の種類」（P.2-10） 「3-3 I/O ポートとデバイス変数」（P.3-5）

STEP3 「軸」の作成とサーボ／エンコーダ入力スレーブへの割付（モーション制御をする場合）	参照
1. 「軸」の作成 2. EtherCAT 構成上のサーボ／エンコーダ入力スレーブへの割付 	📖「3-5「軸」の作成と、サーボドライバ／エンコーダ入力スレーブの割付」（P.3-11）

3-2　EtherCAT スレーブ構成の作成

NY シリーズに接続した各スレーブにアクセスするために、Sysmac Studio でスレーブ構成を作成します。NY シリーズが「正」として認識する EtherCAT スレーブ構成を、Sysmac Studio の EtherCAT 構成編集画面で作成します。



これにより、構成スレーブの「I/O ポート」が自動登録されます。「I/O ポート」には、後でユーザが「デバイス変数」を割り付けます。
ユーザプログラム上で「デバイス変数」を指定することで、各スレーブにアクセスすることができます。

EtherCAT スレーブ構成の作成方法は、🕮『Sysmac Studio Version 1 オペレーションマニュアル（SBCA-470）』の、「[EtherCAT] の構成・設定」を参照してください。

参考

EtherCAT スレーブターミナルを接続する場合は、EtherCAT スレーブ構成の作成に加えて、スレーブターミナルの構成と動作設定が必要です。スレーブターミナルの構成と動作設定の詳細については、🕮『NX シリーズ EtherCAT カプラユニット ユーザーズマニュアル（SBCD-361）』を参照してください。

参考

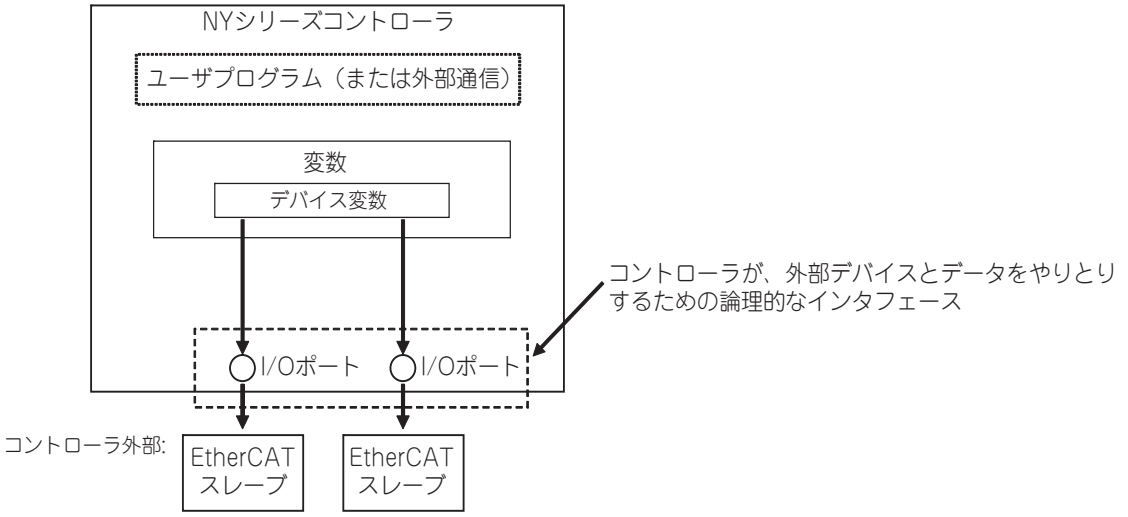
EtherCAT スレーブがサーボ／エンコーダ入力スレーブの場合は、EtherCAT スレーブ構成に登録後、さらに「軸」を作成することで、軸変数を自動登録します。
🕮「3-5 「軸」の作成と、サーボドライバ／エンコーダ入力スレーブの割付」（P.3-11）を参照してください。

3-3　I/O ポートとデバイス変数

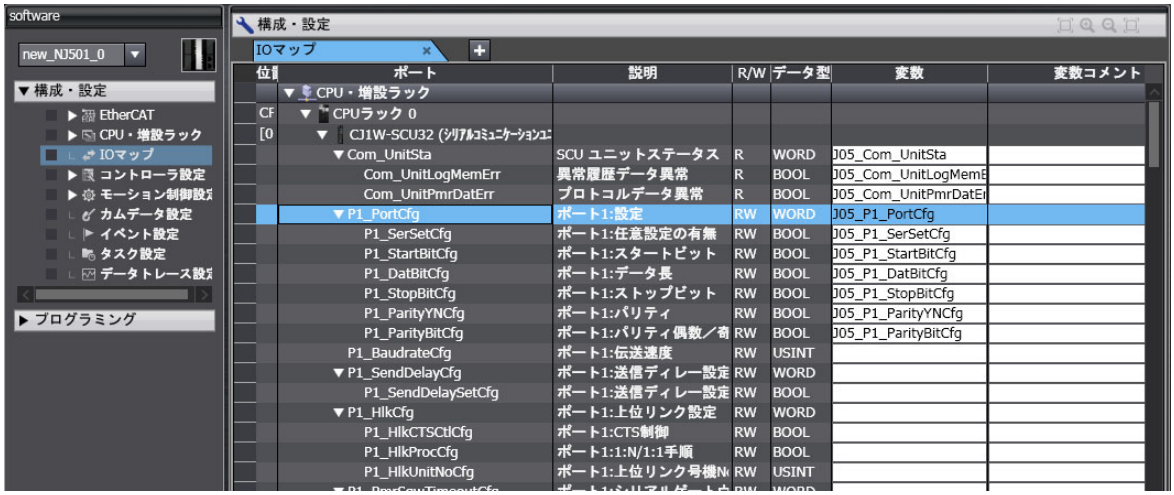
ここでは、NY シリーズコントローラの、EtherCAT スレーブにアクセスするための、I/O ポートおよびデバイス変数について説明します。

3-3-1　I/O ポート

「I/O ポート」とは、NY シリーズコントローラが外部のデバイス（スレーブ／ユニット）と、データをやりとりするための論理的なインタフェースです。
I/O ポートは、Sysmac Studio 上でスレーブ／ユニット構成を作成することで、自動的に生成されます。I/O ポートにデバイス変数を割り付けることにより、ユーザプログラムからアクセスします。



「I/O ポート」は、Sysmac Studio にて、EtherCAT スレーブ構成およびユニット構成を作成、またはそれらを実機コントローラから読み出すと、I/O マップ上に自動的に登録されます。
登録された「I/O ポート」は、Sysmac Studio の「I/O マップ」によって確認することができます。
I/O マップ：



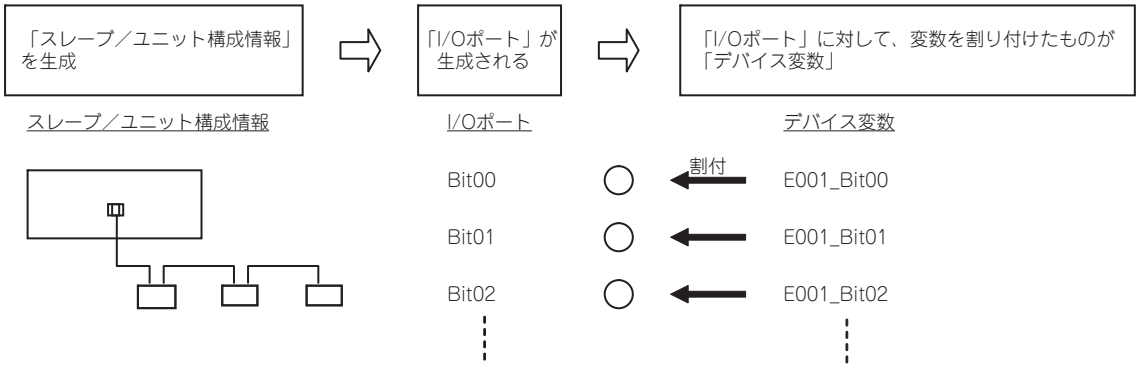
3-3-2　I/O ポート名

I/O ポート名は、自動的に登録されます。デバイス（ユニット／スレーブ）が EtherCAT スレーブの場合、I/O ポート名は以下のようになります。

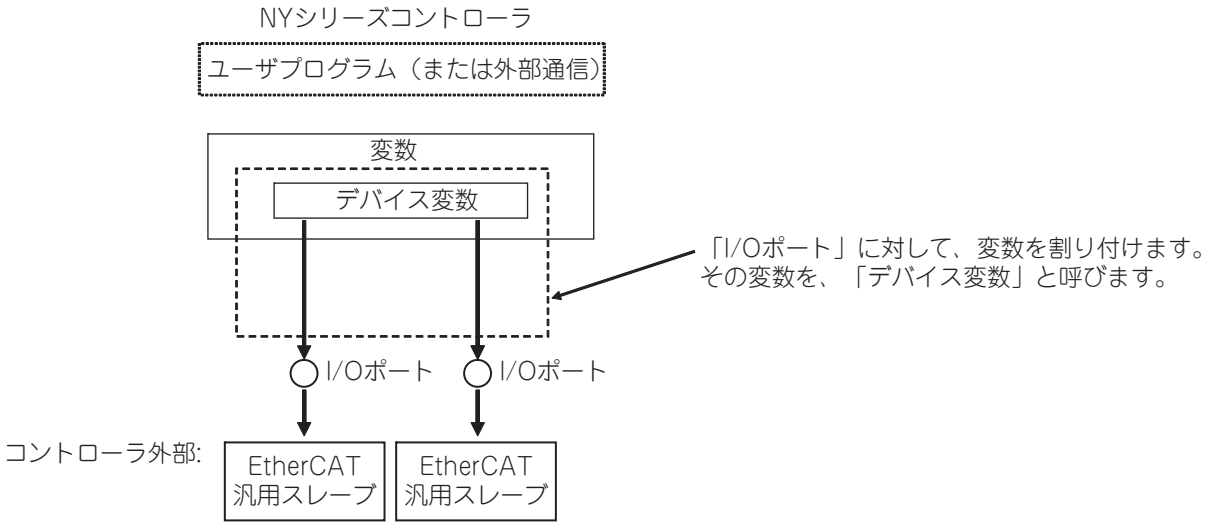
デバイスが EtherCAT スレーブで、かつリモート I/O ターミナルの場合、I/O ポート名は以下のようになります。
例）16 点リモート I/O ターミナルの場合：
　　Bit00 ～ Bit15
その他のスレーブでは、EtherCAT オブジェクトディクショナリにて定義されたオブジェクト名またはその一部となります。
例）アナログ入力ユニットの場合：
　　CH0_input16-bit
例）形 R88D-KN50H-ECT の場合：
　　Position actual value、Digital inputs など

3-3-3　デバイス変数

NY シリーズコントローラでは、コントローラの外部のデバイス（スレーブ／ユニット）が割り付けられるメモリ空間上のアドレスではなく、この「I/O ポート」に対して「変数」を割り付けます。
この変数を、「デバイス変数」と呼びます。



ユーザプログラム（または外部通信）は、この「デバイス変数」によって、デバイス（スレーブ／ユニット）にアクセスすることができます。



変数全体の中でのデバイス変数の位置づけについては、「2-3-1 変数の種類」（P.2-10）を参照してください。

Sysmac Studio でのデバイス変数の登録方法は、『Sysmac Studio Version 1 オペレーションマニュアル（SBCA-470）』を参照してください。

デバイス変数の属性

デバイス変数の各属性は、以下のようになります。また、設定変更が可能な属性と不可の属性があります。

属性	設定される内容	設定変更
変数名	自動生成するとき：[デバイス名 _] + [I/O ポート名] EtherCAT スレーブの「デバイス名」のデフォルトは、「E」＋「001 からの連番」となります。 「I/O ポート名」は、「3-3-2 I/O ポート名」（P.3-6）を参照してください。 手動入力するとき：任意の変数名	可能
データ型	「I/O ポート」のデータ型に従います。	可能
AT（割付先）指定	• NX ユニット（EtherCAT カプラユニット経由） ECAT://node#[EtherCAT カプラユニットノードアドレス ,NX ユニット番号]/I/O ポート名 • EtherCAT スレーブ ECAT://node#[ノードアドレス]/ [I/O ポート名]	不可
保持	EtherCAT スレーブデバイス変数：非保持	不可
初期値	なし	可能
コンスタント	なし	可能
ネットワーク公開	非公開	可能
エッジ	なし	不可

各属性の意味は、「6-3-4 変数の属性」（P.6-29）を参照してください。

参考

- I/O マップ上で、I/O ポートの強制値リフレッシュを行うことができます。これによって、実 I/O を強制的に ON/OFF させ、配線のチェックを行うことができます。
- I/O マップ上では、デバイス変数をどの変数テーブル（グローバル変数テーブル、各 POU のローカル変数テーブル）に登録するかを選択できます。

3-4　ユニットへの変数の割付

命令には、EtherCAT スレーブターミナルの各ユニットを、変数の形で指定する形式のものがあります。したがって、ユーザは、あらかじめ各ユニットに変数を割り付けておく必要があります。いったんユニットに変数を割り付ければ、ユニットの接続位置を変更してもその情報が自動的に変数の値に反映されます。したがって、ユニットの接続位置を変更するたびに、変数の割付をやり直す必要はありません。

参考

スレーブターミナル以外の EtherCAT スレーブにも、変数を割り付けることができます。他社製のEtherCATスレーブも同様です。EtherCATスレーブに変数を割り付ける方法は、EtherCATカプラユニットや NX ユニットに変数を割り付ける方法と同じです。

3-4-1　ユニットへの変数の割付方法

Sysmac Studio で EtherCAT スレーブターミナルの構成設定を行っても、ユニットに割り付ける変数は自動的に作成されません。以下の手順で、ユニットへの変数の割付設定を行ってください。

1. **Sysmac Studio の [構成・設定]–[EtherCAT] で、EtherCAT スレーブターミナルの構成設定を行います。**
2. **[構成・設定]–[I/O マップ] で、I/O マップを表示させます。**
   設定した EtherCAT スレーブターミナルの、各ユニットの I/O マップが表示されます。
3. **変数を割り付けるユニットの形式を右クリックして、[ノード位置ポート表示] を選択します。**
   I/O マップに、「Node location information」というポートが追加されます。
4. **[Node location information] を右クリックして、[デバイス変数生成] を選択します。**
   [Node location information] ポートの [変数] 欄に、変数名が書き込まれます。

ユニットに割り付けられた変数のデータ型は、_sNXUNIT_ID 型の構造体です。_sNXUNIT_ID 型の構造体の詳細は以下のとおりです。

変数		名称	意味	データ型
任意に設定		指定ユニット	指定するユニット	_sNXUNIT_ID
	NodeAdr	ノードアドレス	EtherCAT カプラユニットのノードアドレス	UINT
	IPAdr	IP アドレス*1	EtherCAT カプラユニットの IP アドレス	BYTE[5]
	UnitNo	ユニット番号	指定するユニットのユニット番号	UDINT
	Path	パス*1	指定するユニットまでの経路情報	BYTE[64]
	PathLength	「Path」の有効データ長	「Path」の有効データ長	USINT

*1. コントローラ内部だけで使用する情報です。ユーザが参照や編集をすることはありません。

使用上の注意

ユニットに割り付ける変数の値は、変数登録時に自動的に設定されます。変数の値は、変更しないでください。ユーザが変数の値を変更すると、コントローラが意図どおりに動作しない可能性があります。

参考

スレーブターミナル以外の EtherCAT スレーブに変数を割り付ける場合、変数のデータ型は _sECAT_ID 型の構造体です。詳細は以下のとおりです。

変数	名称	意味	データ型
任意に設定	指定スレーブ	指定するスレーブ	_sECAT_ID
NodeAdr	ノードアドレス	指定スレーブのノードアドレス	UINT

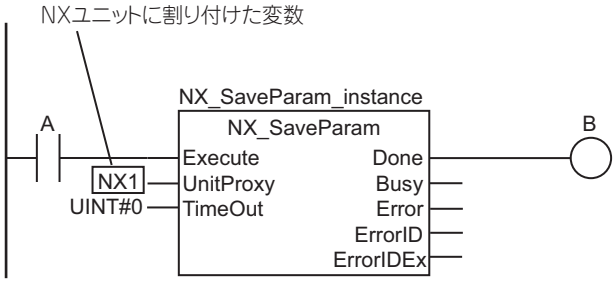
3-4-2 ユニットに割り付けた変数の使用方法

ユニットに割り付けた変数は、ユーザプログラム上で、以下のように使用します。いずれの場合も、I/O マップでユニットに割り付けた変数と同じ名前の変数を、変数テーブルに登録しておく必要があります。その変数のデータ型は、_sNXUNIT_ID 型の構造体です。

ユニットの指定

ユニットを指定する命令には、ユニットに割り付けた変数をそのままパラメータとして渡します。

例）NX_SaveParam 命令の実行
「NX1」という変数を割り付けた NX ユニットを指定して、NX_SaveParam 命令を実行します。変数「UnitProxy」に「NX1」を渡します。



ユニットの属性の指定

ユニットに割り付けた変数のメンバを指定することで、ユニットの一部の属性だけを指定することができます。

例）NX ユニットのユニット番号による命令実行
「NX1」という変数を割り付けた NX ユニットのユニット番号が 2 であれば、そのユニットのデータオブジェクトからデータを読み出します。変数「NX1.UnitNo」が、ユニット番号を表します。

```
IF (NX1.UnitNo = UINT#2) THEN
  NX_ReadObj_instance(Execute:=TRUE, UnitProxy:=NX1, Obj:=S_Obj, ReadDat:=Rdat);
END_IF;
```

複数のユニットの指定

ユニットに割り付ける変数を配列指定にします。ループ処理を使って、複数のユニットに同じ処理を行うことができます。

例）複数の NX ユニットをデータ書込可能なモードにする命令実行
「NX0」、「NX1」、「NX2」が、NX ユニットに割り付けられた変数です。それぞれを配列 NXTable[0..2] に代入してから、順番に NX_ChangeWriteMode 命令を実行します。

- 変数テーブル

名称	データ型
NXTable	ARRAY[0..2] OF_sNXUNIT_ID

- ST プログラム

```
FOR i:= 0 TO 2 DO
   NX_ChangeWriteMode_instance[i](Execute:=FALSE);
END_FOR;

NXTable[0] := NX0;
NXTable[1] := NX1;
NXTable[2] := NX2;

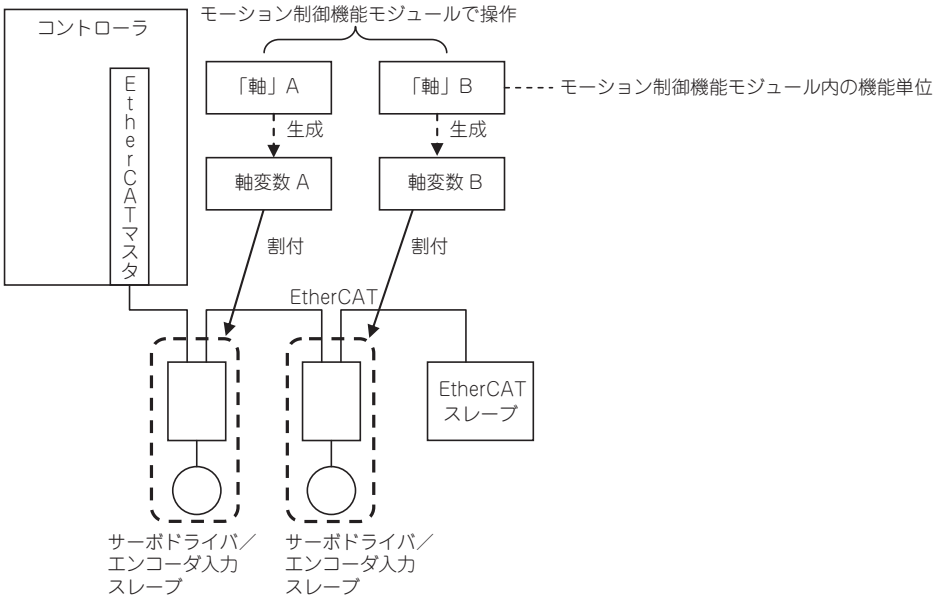
FOR i:= 0 TO 2 DO
   NX_ChangeWriteMode_instance[i](Execute:=TRUE, UnitProxy:=NXTable[i]);
END_FOR;
```

3-5 「軸」の作成と、サーボドライバ／エンコーダ入力スレーブの割付

ここでは、NY シリーズコントローラでの「軸」の作成と、サーボドライバ／エンコーダ入力スレーブの「軸」への割付について、説明します。

3-5-1 概要

EtherCAT スレーブの中のサーボドライバまたはエンコーダ入力スレーブを、モーション制御機能モジュールで操作する場合、Sysmac Studio 上で「軸」を作成し、EtherCAT 上のサーボドライバ、エンコーダ入力スレーブに割り付けます。
これにより、システム定義変数の軸変数が自動的に作成されます。



ユーザプログラム上のモーション制御命令にて軸変数を指定することによって、サーボドライバまたはエンコーダ入力スレーブに対して簡単にアクセスして操作することができます。

3-5-2 軸変数／軸グループ変数

軸変数／軸グループ変数には、以下の種類があります。

変数種類		用途	アクセス対象のデバイス	生成方法
軸変数	システム定義の軸変数	1 軸を単独で制御するとき	「軸」に割り付けている EtherCAT スレーブ（サーボ／エンコーダ入力スレーブ）	標準で用意
	Sysmac Studio で「軸」を作成したときに自動生成される軸変数			Sysmac Studio にて、「軸」を作成し、軸割付
軸グループ変数	システム定義の軸グループ変数	複数軸を多軸協調制御するとき	「軸グループ」に割り付けている EtherCAT スレーブ（サーボ／エンコーダ入力スレーブ）	標準で用意
	Sysmac Studio で「軸グループ」を作成したときに自動生成される軸グループ変数			Sysmac Studio にて、「軸グループ」を作成

軸変数／軸グループ変数の詳細は、🕮『NY シリーズ コマンドリファレンスマニュアル モーション編（SBCE-380）』を参照してください。

• 指定方法

以下の Sysmac Studio 作成のものとシステム定義のもののいずれでも指定できます。

種類	名称	
	軸変数	軸グループ変数
Sysmac Studio 作成のもの	MC_Axis***(*** は作成順に 000 からの昇順) ユーザが任意に変更できます。	MC_Group***（*** は作成順に 000 からの昇順） ユーザが任意に変更できます。
システム定義のもの	_MC_AX[***] （配列要素番号は作成順に 0 からの昇順）	_MC_GRP[***] （配列要素番号は作成順に 0 からの昇順）

• 用途

以下の 2 通りがあります。

(1) モーション制御命令での軸指定または軸グループ指定：

モーション制御命令の入出力変数に指定することで、オムロン製サーボ／エンコーダ入力スレーブに対して操作することができます。

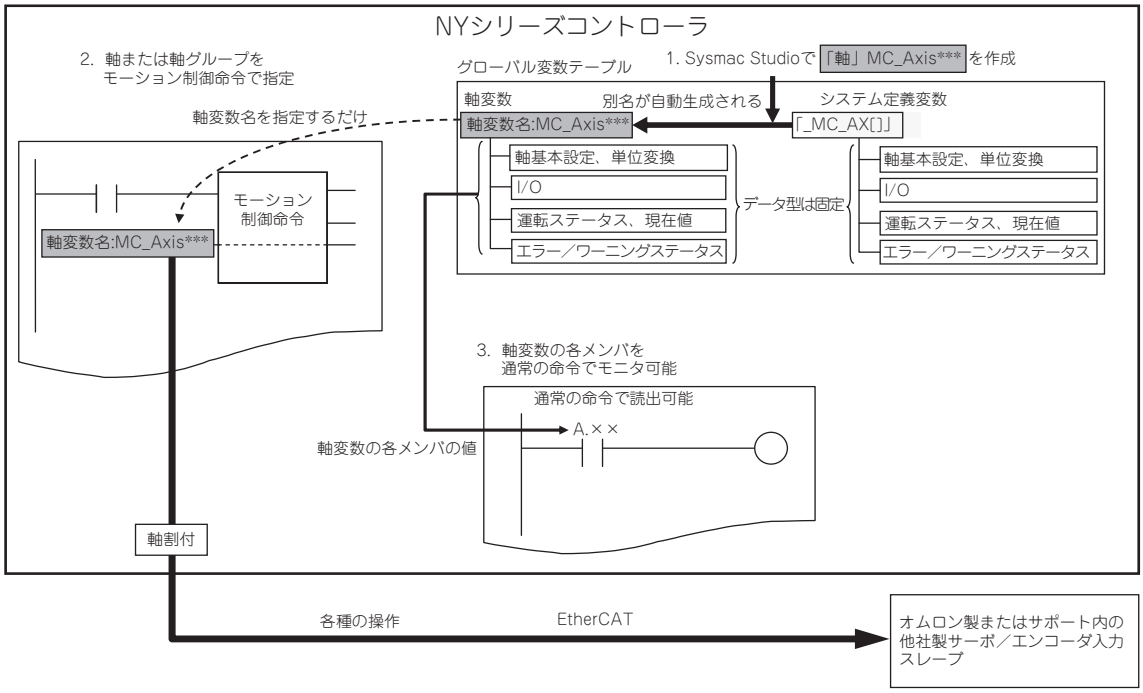
(2) 軸変数の各メンバのモニタ：

通常の命令で各サーボ／エンコーダ入力スレーブの、現在位置や異常情報などをモニタできます。

参考

軸変数の詳細のしくみ

1. Sysmac Studio にて、軸名「A」の「軸」を作成します。
   システム定義変数の軸変数をもとに、変数名「A」の「軸変数」が自動生成されます。軸変数は、軸基本設定、単位変換、I/O、運転ステータス、現在値、エラー／ワーニングステータスから成ります。
2. モーション制御命令の入出力変数で、軸変数名「A」を指定します。オムロン製またはサポート内の他社製サーボ／エンコーダ入力スレーブに対してアクセスして各種の操作ができます。
3. 必要に応じて、軸変数を指定して、通常の命令で、各サーボ／エンコーダ入力スレーブの現在位置や異常情報などをモニタできます。



3-5-3 「軸」および軸変数の作成と使用方法

以下のように、「軸」および軸変数を作成して、使用します。

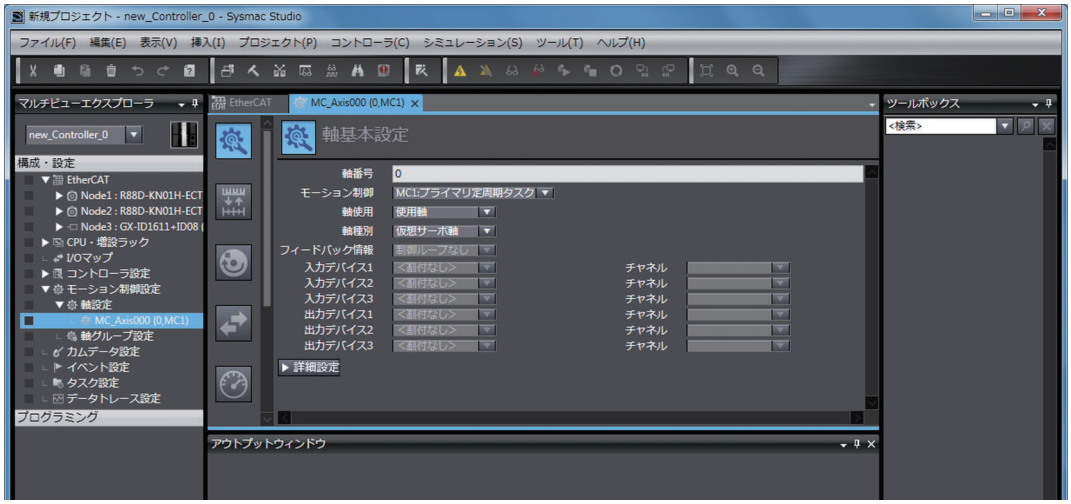
1 **マルチビューエクスプローラの[構成・設定]–[モーション制御設定]–[軸設定]を右クリックし、メニューから [追加]–[軸設定] を選択します。**

必要に応じて軸変数名をデフォルトの「MC_Axis***」（*** は作成順に 000 からの昇順）から変更します。

2 **作成した「軸」に、Sysmac Studio の EtherCAT スレーブ構成上のサーボドライバ／エンコーダ入力スレーブを割り付けます。**

Sysmac Studio から以下の「軸基本設定」を設定します。

区分	パラメータ名	設定内容
軸基本設定	軸番号	軸を作成した順に自動で設定されます。
	軸使用	［使用軸］を選択します。
	軸種別	サーボ軸またはエンコーダ軸を選択します。
	入力／出力デバイス	軸に割り付ける EtherCAT スレーブを選択します。

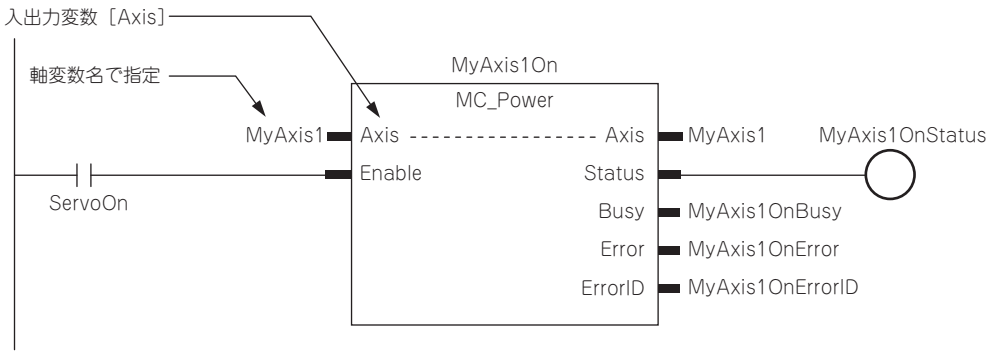


3 **Sysmac Studio から、試運転に必要な設定（単位変換、カウントモード、リミットなど）、および本運転に必要な設定を行い、プロジェクトとしてコントローラに転送します。**

4 **ユーザプログラム上で、モーション制御命令の入出力変数「Axis」に、軸変数名を指定します。**

軸変数名は、「モーション制御設定」にて挿入した「軸名」（「軸変数名」）またはシステム定義変数を指定します。対応するサーボドライバ／エンコーダ入力スレーブに対してモーション制御を実行できます。以下は、軸変数名「MyAxis1」で指定した場合の例です。

例）



軸変数の詳細については、📖「3-5-2 軸変数／軸グループ変数」（P.3-11）を参照してください。

4

コントローラ設定

本章では、各機能モジュールの初期設定について説明します。

4-1 コントローラ設定の概要 4-2

4-2 PLC 機能モジュールの初期設定 4-4
4-2-1 概要 4-4
4-2-2 コントローラ設定 4-4
4-2-3 タスク設定 4-7

4-3 モーション制御機能モジュールの初期設定 4-12
4-3-1 概要 4-12
4-3-2 設定方法 4-13

4-4 EtherCAT マスタ機能モジュールの初期設定 4-14

4-5 EtherNet/IP 機能モジュールの初期設定 4-15

4-1　コントローラ設定の概要

ここでは、コントローラ設定の概要を示します。

全体の手順の中で、以下の網掛けで示した位置に対応しています。



詳細は、📖「1-4 NY シリーズの使用手順の全体」（P.1-10）を参照してください。

コントローラ設定	参照
● **PLC 基本モジュール関連の初期設定：** コントローラ設定： 電源投入時の動作モード、書込プロテクト有無、システムサービス監視設定など	🕮「4-2 PLC 機能モジュールの初期設定」（P.4-4）
●**モーション制御機能モジュール関連の初期設定：** • 軸パラメータ：単軸動作のときの、モーション制御用パラメータ • 軸グループパラメータ： 多軸協調制御動作のときの、モーション制御用パラメータ • カムデータ：カム動作で使用する、位相と変位間の設定テーブル	🕮「4-3 モーション制御機能モジュールの初期設定」（P.4-12）
● **EtherCAT マスタ機能モジュール関連の初期設定：** EtherCAT 構成上の EtherCAT マスタのパラメータ： EtherCAT マスタのプロセスデータ通信周期などのパラメータ設定	🕮「4-4 EtherCAT マスタ機能モジュールの初期設定」（P.4-14）
● **EtherNet/IP 機能モジュール関連の初期設定：** Ethernet ポート設定： EtherNet/IP ポートの TCP/IP 設定、Ethernet 設定など	🕮「4-5 EtherNet/IP 機能モジュールの初期設定」（P.4-15）

4-2　PLC 機能モジュールの初期設定

ここでは、PLC 機能モジュールの初期設定について、説明しています。

4-2-1　概要

PLC 機能モジュール初期設定には以下があります。

- コントローラ設定
- タスク設定

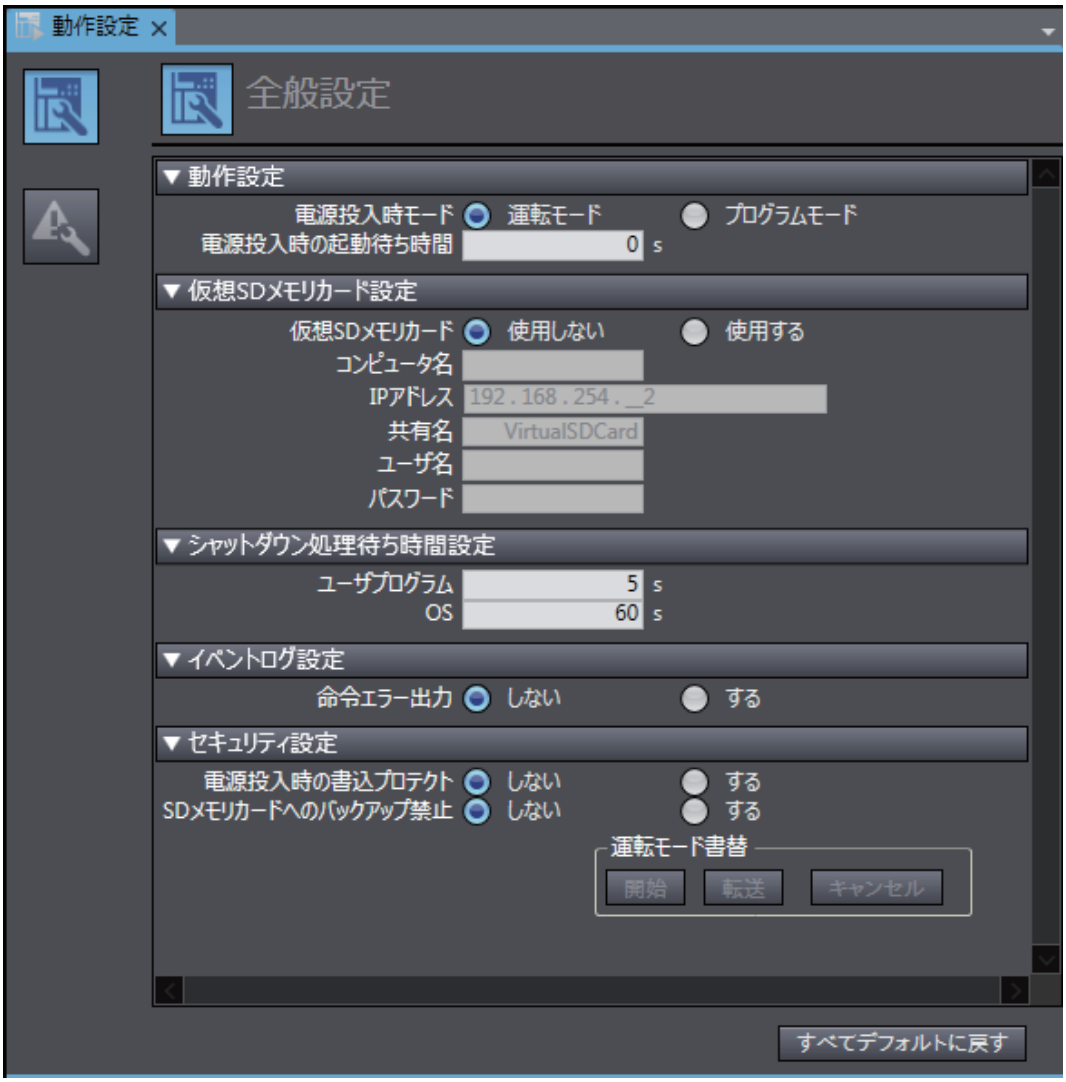
Sysmac Studio から［構成・設定］-［コントローラ設定］、［タスク設定］で各々設定します。

4-2-2　コントローラ設定

動作設定タブ

● 全般設定

電源投入時や動作モード切替時などにおける動作定義など、コントローラがサポートする機能に関する設定です。



設定項目	設定中項目	機能説明	設定値	デフォルト	反映タイミング	運転モード中書替
動作設定	電源投入時モード	電源投入時のコントローラの動作モードを設定します。	運転モード プログラムモード	運転モード	設定転送時	不可
	電源投入時の起動待ち時間 *1	電源投入後の起動中に、NS シリーズ表示器の起動を待ち、タグ照合を優先的に行うための時間を設定します。*2	0 ～ 10s*3	0s	設定転送時	不可
仮想 SD メモリカード設定	仮想 SD メモリカード	仮想 SD メモリカードとして使用する共有フォルダの利用有無を設定します。	使用する 使用しない	無効	設定転送時	不可
	コンピュータ名	Windows 側のコンピュータ名を設定します。	文字列*4	なし	設定転送時	不可
	IP アドレス	Windows 側の内部通信ポートの IP アドレスを設定します。	IPv4*5	192.168.254.2	設定転送時	不可
	共有名	仮想 SD メモリカードとして使用する共有フォルダの共有名を設定します。	文字列*6	VirtualSDCard	設定転送時	不可
	ユーザ名	仮想 SD メモリカードとして使用する共有フォルダの共有ユーザ名を設定します。*7	文字列*8	なし	設定転送時	不可
	パスワード	仮想 SD メモリカードとして使用する共有ユーザのパスワードを設定します。	文字列*9	なし	設定転送時	不可
シャットダウン処理待ち時間設定	ユーザプログラム	電断検知を示す UPS からの信号が ON、または電源ボタンによるシャットダウン指示を受け付けてからコントローラがシャットダウン処理を開始するまでの待ち時間を設定します。	1 ～ 30s	5s	設定転送時	不可
	OS	コントローラから Windows 側にシャットダウン指示をしてから、Windows 側のシャットダウン完了を検知するまでの待ち時間を設定します。	30 ～ 300s	60s	設定転送時	不可
イベントログ設定	命令エラー出力	命令エラーが発生した場合に、それに対応するイベントをイベントログに出力するかしないかを設定します。	しない する	しない	設定転送時	不可

設定項目	設定中項目	機能説明	設定値	デフォルト	反映タイミング	運転モード中書替
セキュリティ設定	電源投入時の書込プロテクト	コントローラの電源投入時に、自動的に書込プロテクトを有効にするかしないかを設定します。	しない する	しない	電源投入時	可
	SD メモリカードへのバックアップ禁止*10	SD メモリカードへのバックアップを禁止するかしないかを設定します。	しない する	しない	設定転送時	可

*1. NS シリーズ表示器をコントローラの内蔵 EtherNet/IP ポートに接続し、NS シリーズ表示器とコントローラを同時に電源投入する場合に有効となる設定です。
*2. 本設定により、NS シリーズ表示器のタグ照合の処理時間を短縮できる場合があります。タグ照合を優先する場合は「10」を、優先しない場合は「0」を設定してください。
「10」を設定した場合、コントローラは、電源投入後の「起動中」に、約 10 秒間 NS シリーズ表示器のタグ照合の処理を優先してから「正常運転中」に移行します。「起動中」にタグ照合の処理の一部を優先することで、タグ照合が完了するまでの時間を短縮できます。
「1 ～ 10」を設定した場合、NS シリーズ表示器の有無によらず、設定した時間だけタグ照合の処理を優先するため、コントローラが「正常運転中」に移行するまでの時間が長くなります。NS シリーズ表示器を接続していない場合、または同時に電源投入しない場合は、「0」を設定してください。
*3. 以下の CPU ユニットと Sysmac Studio の組み合わせでは、「0 ～ 30s」を設定できます。本設定は、CPU ユニットの電源投入から「正常運転中」に移行するまでの時間に影響するため、最大 20 秒を目安に設定してください。

CPU ユニット	Sysmac Studio
形 NY5□2-1□00 の Ver.1.24 以降	1.45 以降

ただし、10 秒を超える値を設定すると、軽度フォールトレベルのコントローラ異常が発生する場合があります。軽度フォールトレベルのコントローラ異常が発生した場合は、異常解除を実行してください。
*4. 使用できる文字は、0～9、A～Z、a～z、- で、文字数は、1～15 文字（NULL 含まず）です。大文字と小文字は区別しません。
*5. 以下に示す IP アドレスは設定できません。
127（10 進数）で始まる IP アドレス
クラス D の IP アドレス（224.0.0.0 ～ 239.255.255.255）
クラス E の IP アドレス（240.0.0.0 ～ 255.255.255.255）
*6. 使用できる文字は、0～9、A～Z、a～z、- で、文字数は、1～32 文字（NULL 含まず）です。大文字と小文字は区別しません。
*7. ドメインを指定する場合、＜ドメイン名＞\＜ユーザ名＞とします。同じ名前のドメインユーザとローカルユーザが混在しない場合は、ドメイン名の指定は省略できます。
*8. 使用できる文字は、0 ～ 9、A ～ Z、a ～ z、および半角記号（` ~ ! # $ ^ & ( ) _ - { } \ ' .）で、文字数は、1 ～ 274 文字（NULL 含まず）です。大文字と小文字は区別しません。
*9. 使用できる文字は、0 ～ 9、A ～ Z、a ～ z、および半角記号（` ~ ! @ # $ % ^ & * ( ) _ - + = { } [ ] \ ¦ : ; " ' < > . ? /）で、文字数は、8 ～ 32 文字（NULL 含まず）です。大文字と小文字は区別します。
*10.NY シリーズコントローラでは、仮想 SD メモリカードが対象となります。

操作項目	機能説明
運転モード書替	運転モード中書替が可能な設定項目の、設定値の書替を実行します。 操作については、『Sysmac Studio Version 1 オペレーションマニュアル（SBCA-470）』を参照してください。 開始　　：設定値の書替が可能な状態になります。 転送　　：設定値をコントローラに転送します。設定値が書き替わります。 キャンセル　：設定値の書替禁止状態になります。

使用上の注意

[イベントログ設定]-[命令エラー出力] を [する] にしている場合、異常がある命令を繰り返し実行すると、命令エラーは毎回出力されます。したがって、イベントログの記録件数が、上限を超える場合があります。その際、古い情報から順に上書きされるので、ご注意ください。

4-2-3　タスク設定

● タスク設定

タスクの追加と各タスクの設定を行います。

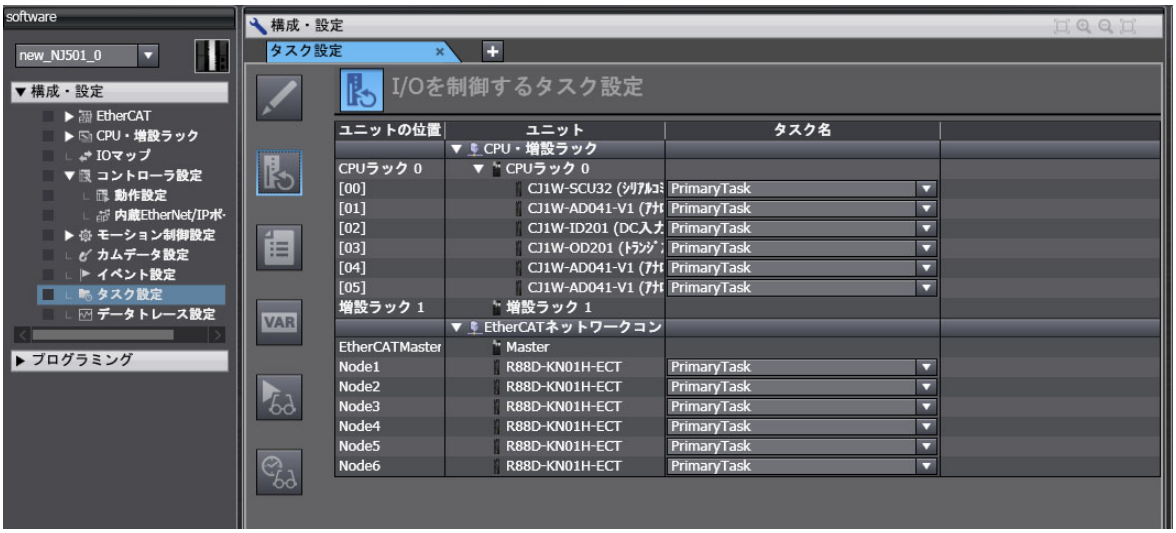


設定項目	設定中項目	機能説明	設定値	デフォルト	反映タイミング	運転モード中書替
タスクタイプ		タスクタイプを選択します。	プライマリ定周期タスク 定周期タスク〔優先度 16〕 定周期タスク〔優先度 17〕 定周期タスク〔優先度 18〕 イベントタスク〔 優先度 8〕 イベントタスク〔 優先度 48〕	プライマリ定周期タスク	設定転送時	不可
	タスクの実行優先度	タスク実行優先度を選択します。	「タスクタイプ」によって自動選択	プライマリ定周期タスク：4	設定転送時	不可
タスク名		タスク名称を設定します。	文字列	プライマリ定周期タスク：PrimaryTask 定周期タスク：PeriodicTask0 イベントタスク：EventTask0	設定転送時	不可
周期／実行条件		タスク周期を選択します。	プライマリ定周期タスク：500μs ～ 8ms（250μs 単位で設定可能） 定周期タスク：1ms ～ 100ms（250μs 単位で設定可能） イベントタスク：命令による実行、変数の条件式一致	プライマリ定周期タスク：1ms 定周期タスク：10ms（優先度 16、17、18） イベントタスク：命令による実行	設定転送時	不可

設定項目	設定中項目	機能説明	設定値	デフォルト	反映タイミング	運転モード中書替
タスク周期オーバ検知		タスク周期超過時にエラーを検知するかを設定します。	• 検知する（軽度フォールトレベルのコントローラ異常発生） • 検知しない（監視情報レベルでログを残す。）	プライマリ定周期タスク、定周期タスク：検知する イベントタスク：検知しないで固定	設定転送時	不可
タスク実行タイムアウト時間		タスク実行タイムアウト時間を設定します。 タイムアウト時間の超過を検出すると、タスク実行タイムアウトが発生します。	プライマリ定周期タスク、定周期タスク： 周期時間 ×1 倍～周期時間 ×5 倍 イベントタスク： 実行優先度 8 の場合 1 ～ 500ms 実行優先度 48 の場合 1ms ～ 10s	プライマリ定周期タスク、定周期タスク： 周期時間 ×5 倍 イベントタスク： 実行優先度 8 の場合 200ms 実行優先度 48 の場合 1s	設定転送時	不可
変数アクセス時間 [%]		コントローラ外部からの変数アクセス時間のタスク周期に対する割合を設定します。	プライマリ定周期タスク、定周期タスク： 1% ～ 50% イベントタスク： なし	3%	設定転送時	不可

● I/O を制御するタスク設定

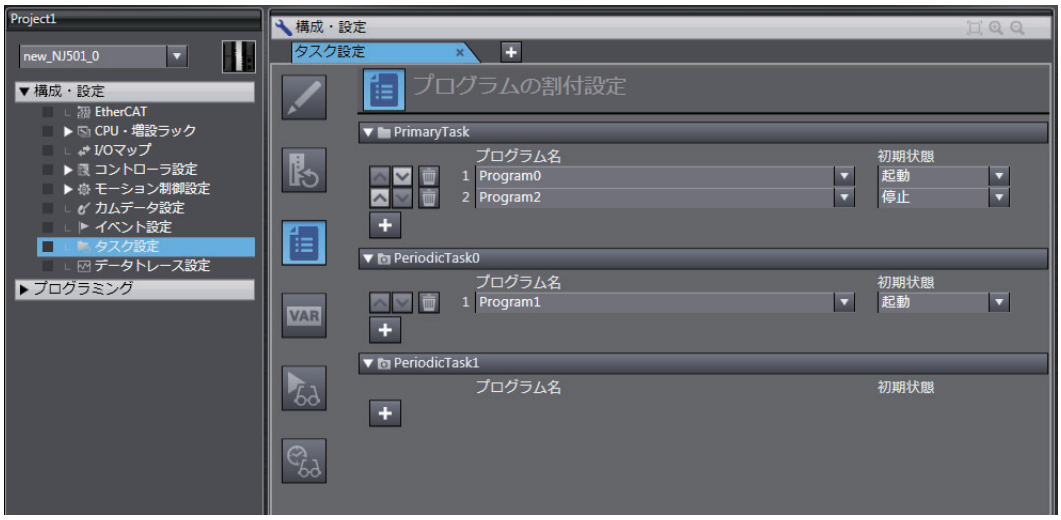
各入出力のリフレッシュ実行タイミングを設定します。



設定項目	機能説明	設定値	デフォルト	反映タイミング	運転モード中書替
タスク名	ユニット／スレーブを、どのタスクで I/O リフレッシュするかを割り付けます。	プライマリ定周期タスク	プライマリ定周期タスク	設定転送時	不可

● プログラムの割付設定

タスクごとのプログラム割付、プログラム実行順、および各プログラムの運転開始時の動作を設定します。



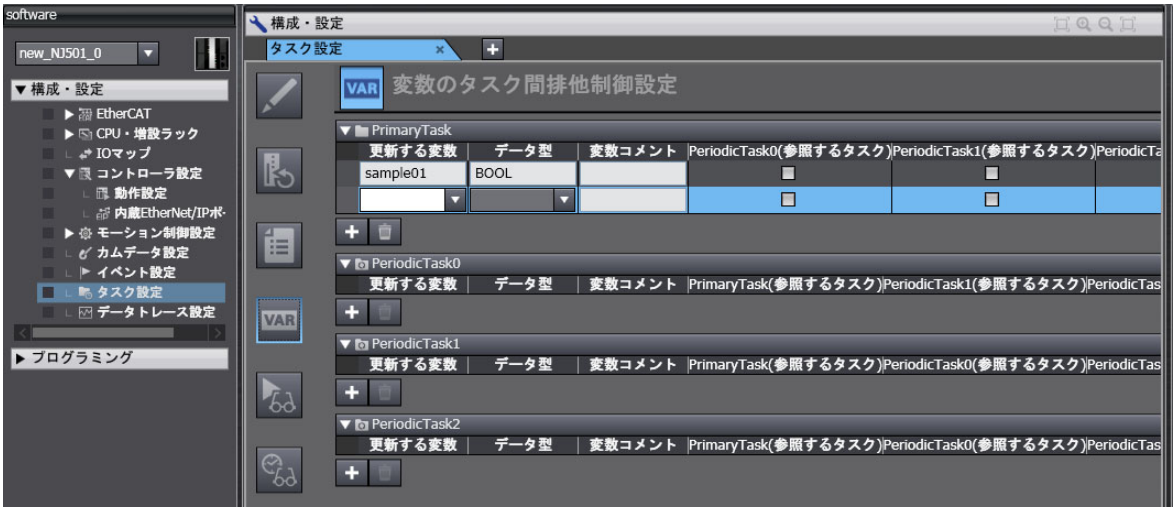
設定項目	機能説明	設定値	デフォルト	反映タイミング	運転モード中書替
プログラム実行順番	タスクに、どのプログラムを割り付けるか、およびタスク内でのそのプログラムの実行順を設定します。	プログラムを実行させたい順で、タスクに上から下に割り付けます。	Program0	設定転送時	不可
プログラムの初期状態	運転開始時に、当該プログラムを起動するかどうかを設定します。	起動 停止	起動	設定転送時	不可

使用上の注意

- スレーブ／ユニットにアクセスするためのデバイス変数を記述したプログラムは、［I/O を制御するタスク設定］で対象のスレーブ／ユニットに割り付けたタスクと同じタスクに割り付けてください。異なるタスクに割り付けた場合、ビルド時にエラーとなります。
- ［I/O を制御するタスク設定］のデフォルトはプライマリ定周期タスクです。プライマリ定周期タスク以外にプログラムを割り付ける場合、あらかじめ［I/O を制御するタスク設定］を再設定しておくことで、ビルド時のエラー発生を防止できます。

● 変数のタスク間排他制御設定

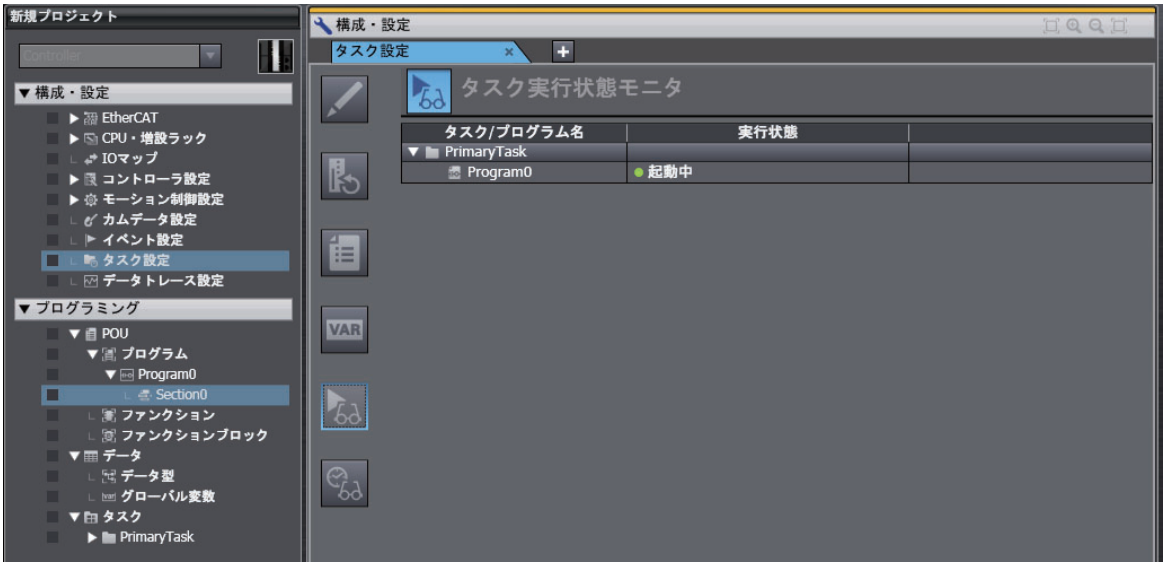
グローバル変数を更新するタスクと参照するタスクを設定します。



設定項目	設定中項目	機能説明	設定値	デフォルト	反映タイミング	運転モード中書替
タスクごと	更新する変数	プライマリ定周期タスクまたは、定周期タスクの中から更新タスクとする変数を選択します。		なし	設定転送時	不可
	データ型	その変数のデータ型	なし			
	変数コメント	その変数の変数コメント	なし			
	参照タスク	その変数の参照タスクを選択します。				

● タスク実行状態モニタ

各プログラムの実行状態を表示します。



● タスク実行時間モニタ

各タスクの実行時間を表示します。



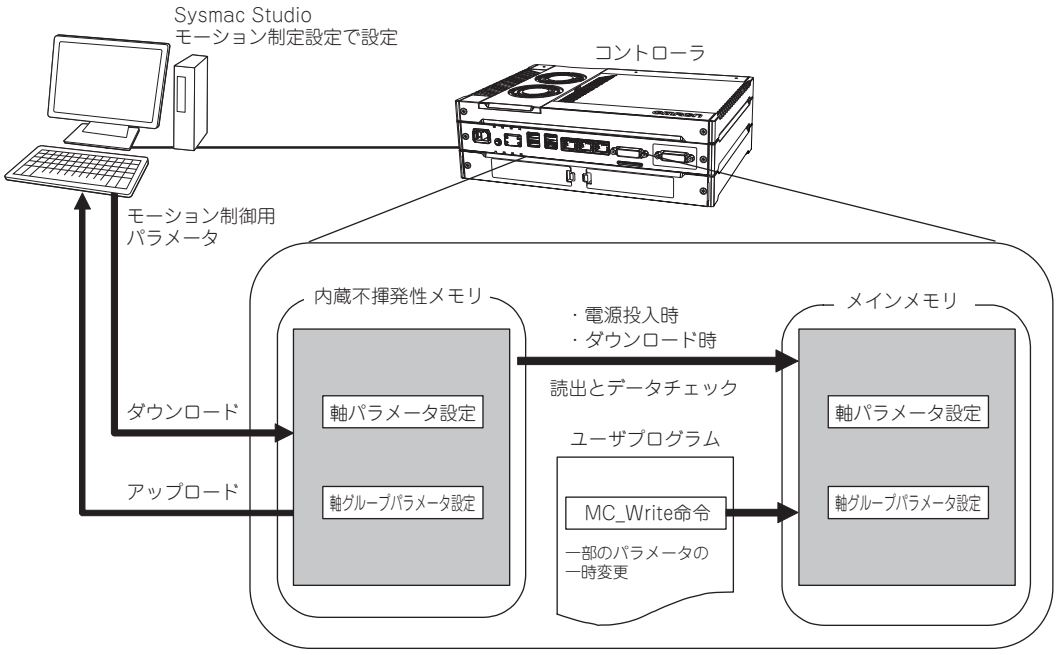
4-3　モーション制御機能モジュールの初期設定

ここでは、モーション制御機能モジュールの初期設定について、説明しています。

4-3-1　概要

モーション制御機能モジュールの初期設定を「モーション制御パラメータ」と呼びます。
「モーション制御パラメータ」には、以下があります。

- 軸パラメータ：単軸制御を行うための設定
- 軸グループパラメータ：多軸協調制御を行うための設定

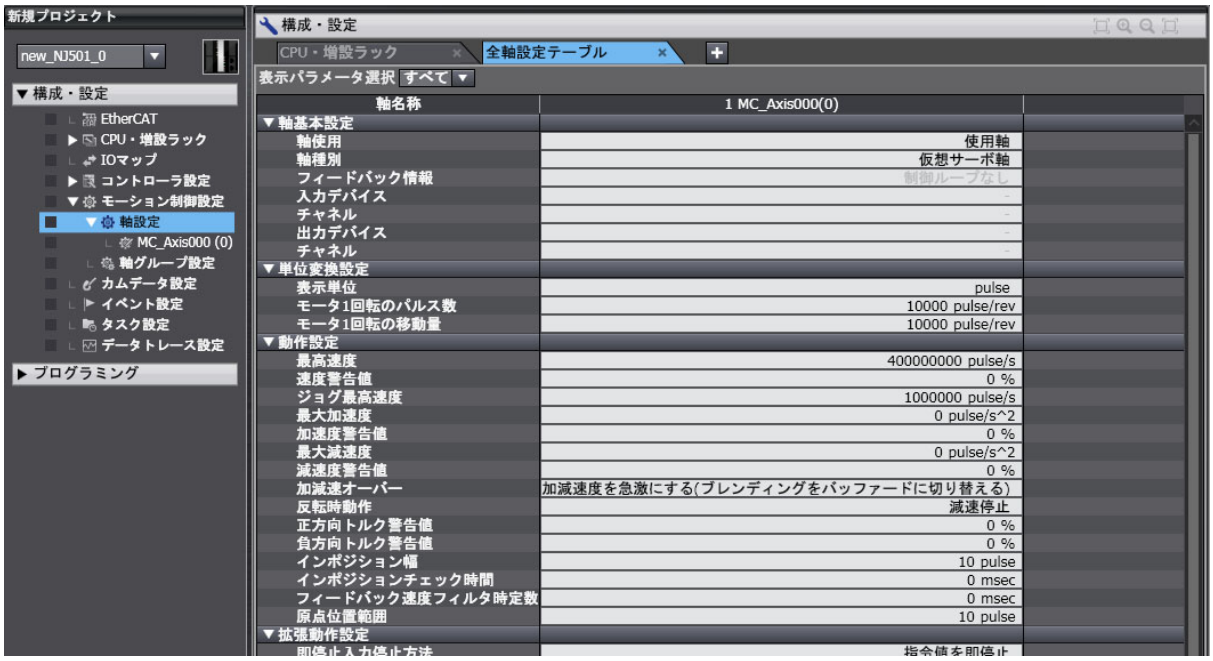


4-3-2　設定方法

モーション制御用パラメータの設定は、以下のいずれかによって設定します。

方法 1：Sysmac Studio のモーション制御設定から設定

Sysmac Studio から以下の [構成・設定]-[モーション制御設定]-[軸設定] を右クリックし [全軸設定テーブル] で設定します。



モーション制御用パラメータをコントローラにダウンロードすると、コントローラ内蔵の不揮発性メモリに保存されます。電源投入時またはダウンロード時に、ダウンロードされた設定データが有効となります。

方法 2：MC_Write 命令から設定

モーション制御用パラメータの一部は、MC_Write 命令から一時的な書き換えが可能です。

詳細は、📖『NY シリーズ コマンドリファレンスマニュアル モーション編（SBCE-380）』を参照してください。

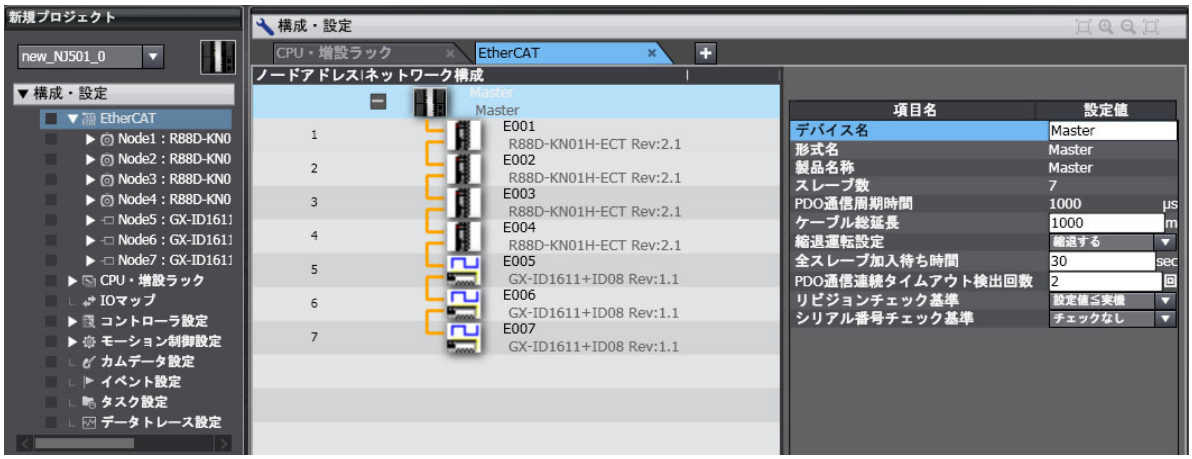
4-4　EtherCAT マスタ機能モジュールの初期設定

ここでは、EtherCAT マスタ機能モジュールの初期設定について、説明しています。

EtherCAT マスタ機能モジュールの初期設定には、以下があります。

- デバイス名
- ケーブル総延長
- 縮退運転設定
- 全スレーブ加入待ち時間
- PDO 通信連続タイムアウト検出回数
- リビジョンチェック基準
- シリアル番号チェック基準

Sysmac Studio から [構成・設定]-[EtherCAT] を選択して、Master を選択すると EtherCAT マスタ機能モジュールの初期設定画面が表示されます。



詳細は、🕮『NY シリーズ 産業用パネル型 PC ／産業用ボックス型 PC ユーザーズマニュアル 内蔵 EtherCAT ポート編（SBCD-368)』を参照してください。

4-5　EtherNet/IP 機能モジュールの初期設定

ここでは、EtherNet/IP 機能モジュールの初期設定について、説明しています。

EtherNet/IP 機能モジュールの初期設定には、以下があります。
- TCP/IP 設定
- LINK 設定
- FTP 設定
- SNMP 設定
- SNMP トラップ設定

Sysmac Studioから［構成･設定］-［コントローラ設定］-［内蔵EtherNet/IPポート設定］で設定します。

詳細は、🕮『NY シリーズ 産業用パネル型 PC ／産業用ボックス型 PC ユーザーズマニュアル 内蔵 EtherNet/IP ポート編（SBCD-369）』を参照してください。

5

タスク設計

本章では、タスクのしくみと種類について説明します。

5-1 タスク設計手順の概要 5-2

5-2 タスクの概要 5-4
5-2-1 タスクとは 5-4
5-2-2 タスク関連の命令 5-6
5-2-3 タスク関連のシステム定義変数 5-6

5-3 NY シリーズのタスクの仕様と基本動作 5-8
5-3-1 NY シリーズのタスクの仕様 5-8
5-3-2 NY シリーズのタスク分割の考え方 5-8
5-3-3 NY シリーズのタスクの基本動作 5-9
5-3-4 NY シリーズのイベントタスクの実行条件 5-15
5-3-5 NY シリーズのイベントタスクの実行タイミング 5-20
5-3-6 イベントタスク完了前に再度実行条件が成立した場合の動作 5-24

5-4 タグデータリンクサービスとシステムサービス 5-25
5-4-1 タグデータリンクサービスとシステムサービスの実行優先度と実行順序 .. 5-26
5-4-2 タグデータリンクサービスの処理内容と実行タイミング 5-28
5-4-3 システムサービスの処理内容と実行タイミング 5-30

5-5 タスク関連の割付と設定 5-31
5-5-1 I/O リフレッシュのタスクへの割付 5-31
5-5-2 プログラムのタスクへの割付 5-37
5-5-3 タスクの設定項目 5-38

5-6 変数の同時性確保 5-40
5-6-1 タスク間での変数の値の同時性を確保する方法 5-40
5-6-2 コントローラ外部からの変数アクセス 5-46

5-7 タスク関連の異常 5-50

5-8 タスク実行状態、タスク実行時間のモニタ 5-52

5-9 タスク設計方法および入出力応答時間 5-56
5-9-1 タスク実行時間の確認 5-56
5-9-2 タスク設計の例 5-57
5-9-3 システムとしての入出力応答時間 5-58

5-1　タスク設計手順の概要

ここでは、タスク設計手順の概要を示します。

全体の手順の中で、以下の網掛けで示した位置に対応しています。

STEP 1. ソフト設計

- STEP1-1 I/O と処理の設計
- **STEP1-2 タスク設計**
- STEP1-3 プログラムの設計

↓

STEP 2. ソフトウェア設定とプログラミング

- STEP2-1 スレーブ／ユニット構成
- STEP2-2 コントローラ設定
- STEP2-3 プログラミング
- STEP2-4 オフラインデバッグ

↓

STEP 3. 装着とハードウェアの設定

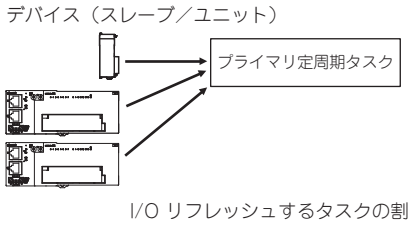
↓

STEP 4. 配線

↓

STEP 5. 実機による動作確認と運転

詳細は、🕮「1-4 NY シリーズの使用手順の全体」（P.1-10）を参照してください。

タスクの設計	参照
● **タスク構成の設計** 制御対象装置の必要な入出力応答性能から、タスク構成を設計します。	🕮「5-3-3 NY シリーズのタスクの基本動作」（P.5-9） 🕮「5-9 タスク設計方法および入出力応答時間」（P.5-56）
● **デバイス（スレーブ／ユニット）の I/O リフレッシュを、タスクに割り付けます。NY シリーズコントローラでは、プライマリ定周期タスクだけに割り付けることができます。** 	🕮「5-5-1 I/O リフレッシュのタスクへの割付」（P.5-31）
● **どのプログラムを、プライマリ定周期タスク、定周期タスク（実行優先度 16 ～ 18）、イベントタスク (実行優先度 8、48）のいずれに割り付けるかを設計**	🕮「5-5-2 プログラムのタスクへの割付」（P.5-37）
● **タスク間の変数の排他方法を設計** 異なるタスク間で同じグローバル変数を使用する場合に、タスク間の変数の排他方法を設計します。	🕮「5-6-1 タスク間での変数の値の同時性を確保する方法」（P.5-40）
● **コントローラ外部から変数をアクセスする時のタスク設計** コントローラ外部からコントローラ内の変数にアクセスする場合に、その変数アクセスと、ある特定タスクのユーザプログラム実行とを同期させるときに設計します。変数アクセスには、EtherNet/IP タグデータリンクを含みます。	🕮「5-6-2 コントローラ外部からの変数アクセス」（P.5-46）

Sysmac Studio でのタスク設定

タスクの設定	参照
● **PLC 基本モジュール関連の初期設定** タスク設定：タスク周期、入出力設定、プログラム割付 タスク間インタフェース設定など	🕮「4-2 PLC 機能モジュールの初期設定」（P.4-4）

Sysmac Studio でのオフラインデバッグ

机上動作確認	参照
● **シミュレータ（仮想 PLC）で、シーケンス制御およびモーション制御を机上デバッグします。**	🕮「第 7 章 動作確認と運転」（P.7-1）

5-2　タスクの概要

ここでは、タスクの概要について説明します。

5-2-1　タスクとは

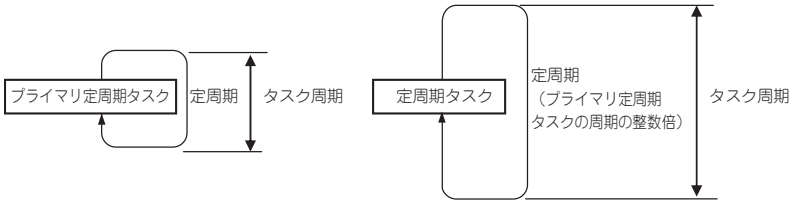
タスクとは、I/O リフレッシュやユーザプログラム実行などの一連の処理に、実行条件と実行順序を指定する機能です。

タスクには、実行条件および実行優先度によって、以下の 3 種類があります。

タスクの種類	タスク数	タスク実行優先度	定義	おもな処理内容
プライマリ定周期タスク	1	4	設定したタスク周期に 1 回実行します。どのタスクよりも最優先に実行します。 プライマリ定周期タスクのモーション制御命令および EtherCAT の通信はプライマリ定周期タスクの周期で実行します。	I/O リフレッシュ、ユーザプログラム実行、モーション制御
定周期タスク	0 ～ 3	16、17、18	設定したタスク周期に 1 回実行します。 実行優先度 16 の定周期タスクのモーション制御命令および EtherCAT の通信はプライマリ定周期タスクの周期で実行します。	タスク実行優先度により、処理内容が異なります。 実行優先度 16、17、18：ユーザプログラム実行
イベントタスク	0 ～ 32	8、48	指定した実行条件が成立したときに 1 回だけ実行します。	ユーザプログラム実行

● プライマリ定周期タスクと定周期タスク

プライマリ定周期タスクおよび定周期タスクは、ともに定周期で実行されます。
プライマリ定周期タスクおよび定周期タスクが定周期で実行される間隔のことを「タスク周期」と呼びます。



各タスクには、1 ～ 128 個のプログラムを割り付けることができます。割り付けられたプログラムは、割付順に実行されます。それを「ユーザプログラム実行」と呼びます。
EtherCAT スレーブとのデータ交換を I/O リフレッシュといいます。
I/O リフレッシュは、スレーブ／ユニットごとにプライマリ定周期タスクだけに割り付けることができます。

● イベントタスク

イベントタスクは、指定した実行条件が成立したときに 1 回だけ実行されます。イベントタスクの実行条件には、以下の 2 種類があります。

実行条件	仕様
命令による実行	イベントタスク実行命令 ActEventTask が実行されると、イベントタスクが実行されます。
変数の条件一致による実行	指定した変数が、あらかじめ設定した条件に一致すると、イベントタスクが実行されます。

各タスクには、1 ～ 128 個のプログラムを割り付けることができます。割り付けられたプログラムは、割付順に実行されます。

使用上の注意

- イベントタスクでは、I/O リフレッシュやモーション制御は実行されません。したがって、I/O 制御を伴うプログラムやモーション制御命令を含むプログラムを、イベントタスクに割り付けることはできません。
- イベントタスクは、タスク周期ごとに繰り返し実行されるものではありません。したがって、複数のタスク周期にまたがって実行される命令を含むプログラムを、イベントタスクに割り付けることはできません。複数のタスク周期にまたがって実行される命令とは、SD メモリカード命令や通信命令など一部の基本命令、すべてのモーション制御命令、すべてのシミュレーション命令です。イベントタスクで使用できない基本命令の詳細は、🕮『NY シリーズ コマンドリファレンスマニュアル 基本編（SBCA-437)』を参照してください。

5-2-2　タスク関連の命令

タスク関連の命令として、以下のものが用意されています。

命令	命令名称	概要	
GetMyTaskStatus	自タスク状態読出	自タスクの以下のタスク状態を読み出します。 前回タスク実行時間、タスク実行時間最大値、タスク実行時間最小値、タスク実行回数、タスク周期超過フラグ、タスク周期超過回数	
GetMyTaskInterval	自タスク設定周期読出	自タスクのタスク周期を読み出します。	
Task_IsActive	タスク実行中判定	指定したタスクが実行中かどうかを返します。	
Lock	タスク間排他ロック	ロック区間を開始します。	他のタスクの同一番号のロック区間は同時に実行できなくなります。
Unlock	タスク間排他ロック解除	ロック区間を終了します。	
ActEventTask	イベントタスク起動	指定したイベントタスクを起動します。	

5-2-3　タスク関連のシステム定義変数

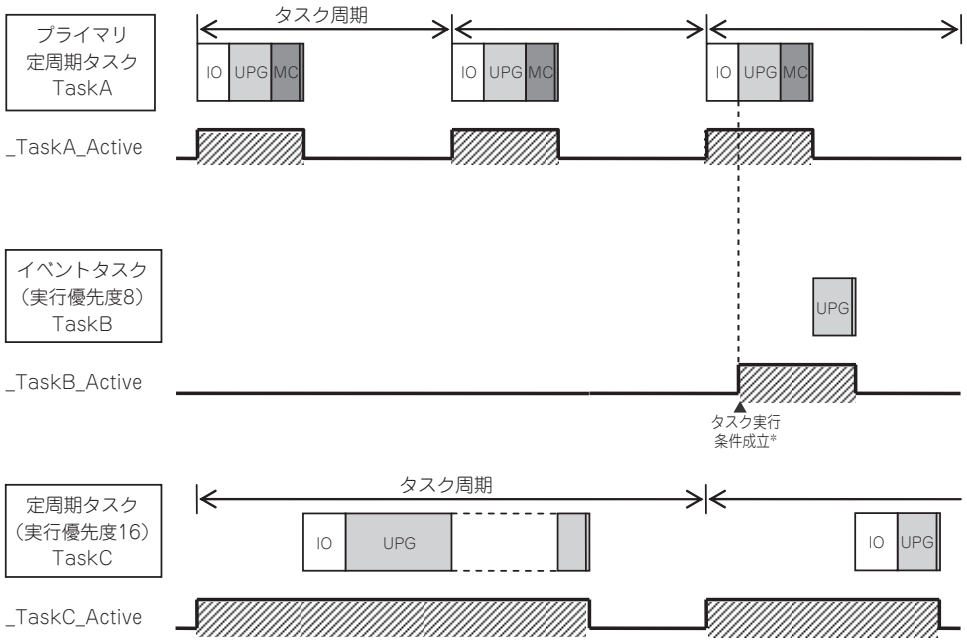
以下のように、タスクの状態を示すシステム定義変数が、タスクごとに用意されています。
ただし、リアルタイム性、および各機能モジュールの異常ステータスとの同時性の観点から、ユーザプログラム上では使用できません。Sysmac Studio のデータトレース機能でタスクの状態をサンプリングするときに使用します。
ユーザプログラムからタスクの状態を参照する場合は、GetMyTaskStatus 命令、Task_IsActive 命令を使用してください。

変数名	名称	機能	データ型	R/W
_＜タスク名称＞_Active	タスク実行中フラグ	タスク実行中は TRUE になります。 タスク実行完了からタスク周期の終わりまで FALSE になります。	BOOL	R
_＜タスク名称＞_LastExecTime	前回タスク実行時間	前回に実行したタスク実行時間を示します。	TIME	R
_＜タスク名称＞_MaxExecTime	タスク実行時間最大値	タスク実行時間の最大値を示します。	TIME	R
_＜タスク名称＞_MinExecTime	タスク実行時間最小値	タスク実行時間の最小値を示します。	TIME	R
_＜タスク名称＞_ExecCount	タスク実行回数	タスクの実行回数を示します。 現在値がデータ型の最大値を超えたら、現在値を 0 に戻して、カウントを継続します。	UDINT	R
_＜タスク名称＞_Exceeded	タスク周期超過フラグ	タスク周期を超過すると、タスク実行の完了時に TRUE となります。 タスク周期以内にタスク実行が完了すると FALSE となります。	BOOL	R
_＜タスク名称＞_ExceedCount	タスク周期超過回数	タスク周期を超過した回数が格納されます。 現在値がデータ型の最大値を超えたら、現在値を 0 に戻して、カウントを継続します。	UDINT	R

（注） 例：タスク名称「MainTask」のタスク周期超過フラグは、“_MainTask_Exceeded” です。

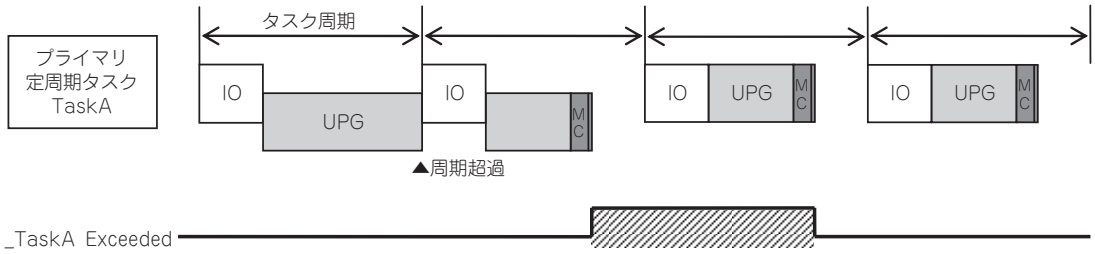
フラグ動作

● タスク実行中フラグ（_＜タスク名称＞_Active）



*ActEventTask命令でイベントタスクを起動する場合は、ActEventTask実行時にタスク実行中フラグがONします。

● タスク周期超過フラグ（_＜タスク名称＞_Exceeded）



5-3　NY シリーズのタスクの仕様と基本動作

ここでは、マルチコアプロセッサを搭載した NY シリーズコントローラのタスクの仕様と基本動作について説明します。

5-3-1　NY シリーズのタスクの仕様

タスクの仕様は以下のとおりです。

項目	仕様
タスクの種類	• プライマリ定周期タスク • 定周期タスク（優先度 16、17、18） • イベントタスク（優先度 8、48）
タスクの数	• プライマリ定周期タスク：1 • 定周期タスク：0 ～ 3*1 • イベントタスク：0 ～ 32*2
1 タスクあたりのプログラムの数	最大 128
プライマリ定周期タスクのタスク周期	500μs ～ 8ms（250μs 単位で設定可能）
定周期タスクのタスク周期	1ms ～ 100ms（250μs 単位で設定可能） 定周期タスクのタスク周期は、プライマリ定周期タスクの整数倍を設定します。 各タスクのタスク周期の最小公倍数が 600ms を超える組み合わせは選択できません。

*1. 実行優先度 16、17、18 が、それぞれ 0 または 1 個です。
*2. 実行優先度 8、48 が、それぞれ 0 ～ 32 個で、両方合わせて最大 32 個です。

5-3-2　NY シリーズのタスク分割の考え方

すべてのプログラムは、いずれかのタスクに割り付ける必要があります。プログラムの特性に応じて、以下のような考え方で割り付けるタスクを決めてください。

タスク	割り付けるのに適したプログラム
プライマリ定周期タスク	• 周期ごとの I/O リフレッシュ、ユーザプログラム実行、モーション制御、システム共通処理の実行周期が、厳密に守られる必要がある。 • 実行優先度が最も高く、高速な応答が必要な制御を含む。 • 実行優先度が最も高いモーション制御命令を含む。
定周期タスク (優先度 16)	• I/O リフレッシュをプライマリ定周期タスクに割り付けた一部のスレーブに対する制御を含む。 • 実行優先度は比較的低いが、周期ごとにユーザプログラム、システム共通処理を実行する必要がある。 • 実行優先度が比較的低いモーション制御命令を含む。 • プライマリ定周期タスクで制御するスレーブ／ユニットに対する処理のうち、比較的実行優先度の低い処理を定周期タスク（優先度 16）に分割して制御する場合に使用する。
定周期タスク (優先度 17,18)	• 実行優先度は比較的低いが、周期ごとにユーザプログラム、システム共通処理を実行する必要がある。 • 高速な応答を必要としない演算処理、通信処理を含む。
イベントタスク	• ある条件が成立しているときだけ実行する。

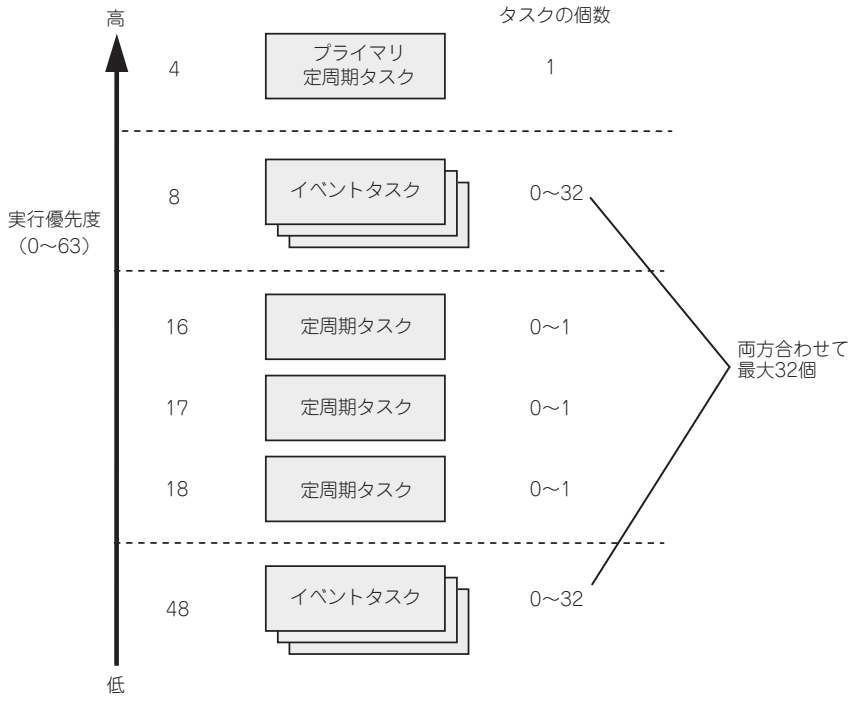
5-3-3　NY シリーズのタスクの基本動作

NY シリーズコントローラは、マルチコアプロセッサの搭載により、タスクの実行とタグデータリンクサービスやシステムサービスを並列に実行することができます。各タスクがどんな順序で実行されるかは、それぞれのタスクに設定された実行優先度によって決まります。

タスク実行優先度

タスクには、その種類によって実行優先度が決められています。NY シリーズコントローラは、実行優先度が高いタスクを優先的に実行します。
あるタスク Ta が実行中に、それよりも実行優先度の高い Tb の実行条件が成立した場合、Tb の処理を優先的にコアに割り当てます。
タスクの種類による実行優先度は、以下のとおりです。実行優先度の値が小さいほど、優先度が高いことを示します。

タスク	実行優先度	実行優先度の重複
プライマリ定周期タスク	4	−
定周期タスク	16、17、18	複数のタスクに、同じ実行優先度を設定することはできません。
イベントタスク	8、48	複数のタスクに、同じ実行優先度を設定することが可能です。 その場合の実行順序については、「5-3-5 NY シリーズのイベントタスクの実行タイミング」（P.5-20）を参照してください。



プライマリ定周期タスクと定周期タスクのタスク周期

プライマリ定周期タスクと定周期タスクは、周期的に繰り返し実行されるタスクです。
定周期タスクのタスク周期は、必ずプライマリ定周期タスクのタスク周期（「プライマリ周期」と呼びます）の整数倍になるように設定します。したがって、数回に一度、両者の実行タイミングがそろいます。
たとえば、プライマリ周期が 1ms、定周期タスク（実行優先度 16）のタスク周期が 4ms の場合は、プライマリ定周期タスクが 4 回実行されるごとに、プライマリ定周期タスクと定周期タスクの実行タイミングがそろいます。

参考

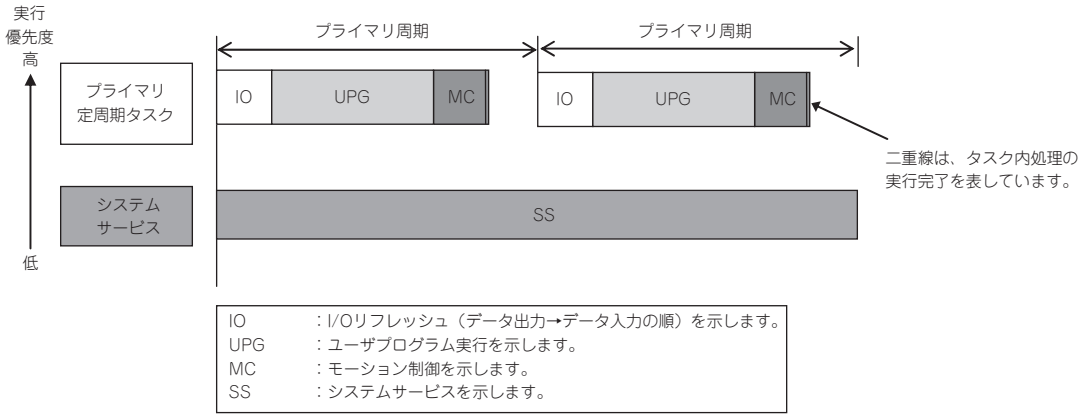
イベントタスクは、周期的に実行されるのではなく、指定した実行条件が成立したときに 1 回だけ実行されるタスクです。したがって、イベントタスクの実行タイミングは、イベントタスクの実行条件が成立したタイミングと、イベントタスクの実行優先度によって決まります。

タスクの実行順序の例

プライマリ定周期タスクと定周期タスクについて、実行順序の例をいくつか示します。
イベントタスクの実行順序については、「5-3-5 NY シリーズのイベントタスクの実行タイミング」（P.5-20）を参照してください。

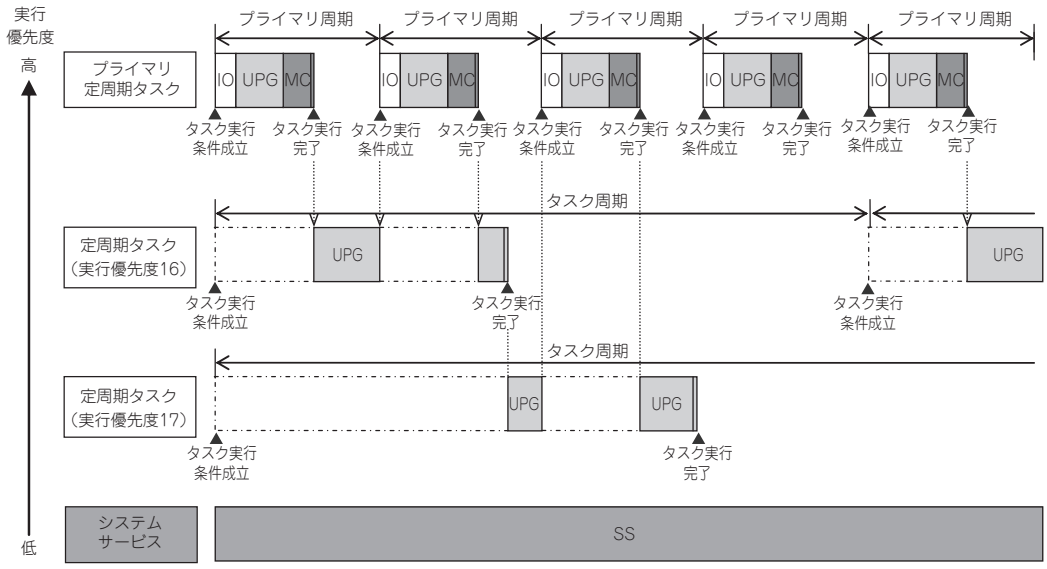
● プライマリ定周期タスクだけの場合

プライマリ周期ごとに、プライマリ定周期タスクが実行されます。
図中のシステムサービスとは、通信処理など、コントローラがタスク以外に実行する処理のことです。システムサービスの詳細は、「 システムサービスの処理内容」（P.5-30）を参照してください。



● **プライマリ定周期タスク、定周期タスク（実行優先度 16）、定周期タスク（実行優先度 17）が混在する場合**

- プライマリ定周期タスクは、実行優先度が最も高いため、確実にプライマリ周期で実行されます。
- 定周期タスク（実行優先度 16）は、プライマリ定周期タスクが完了してから実行されます。
- 定周期タスク（実行優先度 17）は、実行優先度がさらに低いので、上記の 2 つのタスクのいずれも実行されていない時間に実行されます。
- この例では、定周期タスク（実行優先度 16）のタスク周期は、プライマリ周期の 4 倍としています。したがって、プライマリ周期の 4 回に一度、プライマリ定周期タスクと定周期タスク（実行優先度 16）の実行開始のタイミングが同期します。
- システムサービスは、タスクの影響を受けずにいつでも実行されます。



使用上の注意

複数のタスク間で、同じ変数の読み書きをする可能性がある場合は、必ず「変数のタスク間排他制御機能」を使用してください。そうしないと、あるタスクの実行中に、別のタスクによって変数の値が変更されてしまう場合があります。

詳細は、📖「5-6-1 タスク間での変数の値の同時性を確保する方法」（P.5-40）を参照してください。

タスクと動作モード

NY シリーズの動作モードとタスクには、以下の関係があります。

タスク	仕様
プライマリ定周期タスク	• 運転モードでもプログラムモードでも実行されます。 • ただし、ユーザプログラム実行が行われるのは運転モードだけです。
定周期タスク	
イベントタスク	運転モードだけで実行されます。

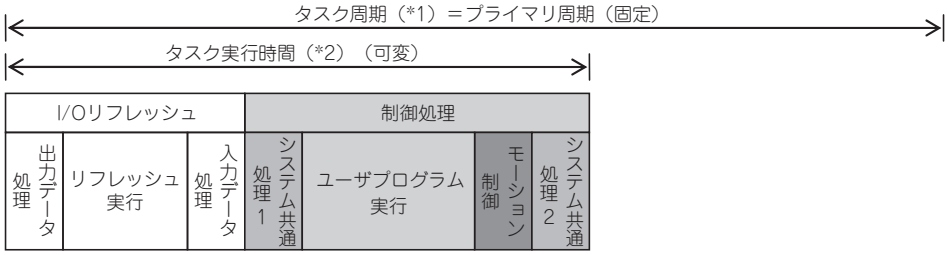
使用上の注意

- 動作モードをプログラムモードから運転モードに切り替えた時点で、イベントタスクの実行条件がすでに成立している場合、イベントタスクは実行されません。イベントタスクが実行されるのは、動作モードが運転モードである状態で、イベントタスクの実行条件が不成立から成立に変化したときです。
- 動作モードが運転モードでも、全停止フォールトレベルの異常の発生中は、イベントタスクは実行されません。

各タスク種類の処理内容

● プライマリ定周期タスク

タスクの実行優先度が最も高く、高速・高精度に実行することができるタスクです。
指定した周期で、システム共通処理、I/O リフレッシュ、ユーザプログラム、モーション制御が実行されます。

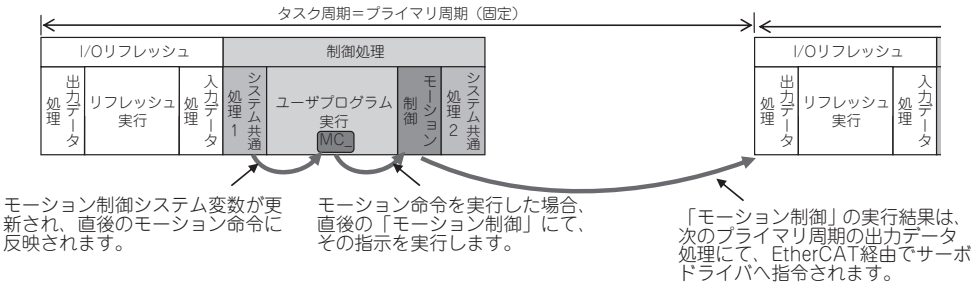


*1 ：タスク周期：　　タスクが定周期で実行される、設定された固定時間です。
*2 ：タスク実行時間：　　タスクの実行条件成立から実行完了までに、実際にかかった時間です。

処理		処理内容
I/O リフレッシュ	出力データ処理	• I/O リフレッシュを実行する I/O（出力）への OUT リフレッシュデータを生成します。 • 強制値リフレッシュが設定されている場合は、OUT リフレッシュデータに強制値（出力）を反映します。
	リフレッシュ実行	• I/O とのデータ交換を実行します。
	入力データ処理	• イベントタスクの条件式一致判定。 • I/O リフレッシュを実行した I/O（入力）ユニットの IN リフレッシュデータを取り込みます。 • 強制値リフレッシュが設定されている場合は、取り込んだ IN リフレッシュデータに強制値（入力）を反映します。
システム共通処理 1		• 変数のタスク間排他制御処理（参照タスクの場合） • モーション入力処理[*1] • データトレース処理（サンプリング、トリガ判定）
ユーザプログラム実行		• タスクに割り付けられたプログラムを割付順に実行します。
モーション制御[*2]		• プライマリ定周期タスク、および定周期タスク（実行優先度 16）に割り付けられたユーザプログラム上のモーション制御命令からのモーション制御の指示を実行します。 • 次のプライマリ定周期タスクの「I/O リフレッシュ」へのモーション出力処理
システム共通処理 2		• 変数のタスク間排他制御処理（更新タスクの場合） • タスク実行との同時性を確保させるためのコントローラ外部からの変数アクセス処理（[タスク設定] にて設定した「変数アクセス時間」分） • EtherNet/IP タグデータリンク処理の場合は、タグ（ネットワーク公開属性の変数）に対して更新タスクを設定すると、上記変数アクセス処理として、実行します。

*1. モーション制御システム変数の軸現在値（位置、速度、トルク）、およびモーション制御システム変数のサーボドライバ状態が、更新されます。
*2. プライマリ定周期タスクのユーザプログラム上のモーション制御命令の場合、次の図のように、直後の「モーション制御」でその命令からの指示が実行され、次のプライマリ定周期タスクの「I/O リフレッシュ」にて、サーボドライバへ出力されます。

（注）上表の各セルの処理は、記載された順に実行されます。

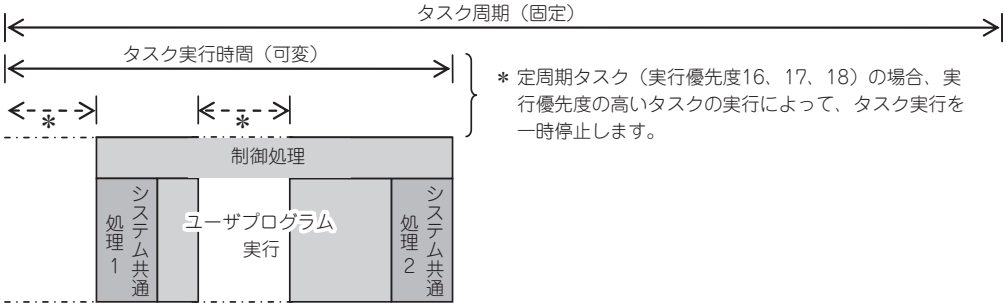


なお、定周期タスク（実行優先度 16）のユーザプログラム上のモーション制御命令の場合は、その定周期（実行優先度 16）の次のプライマリ定周期タスクの「モーション制御（MC）」のタイミングで、その命令からの指示が実行されます。

詳細は、🕮「5-9-3 システムとしての入出力応答時間」（P.5-58）を参照してください。

● 定周期タスク（実行優先度 16、17、18）

プライマリ周期の整数倍に指定されたタスク周期で、ユーザプログラムを実行するタスクです。タスク全体で 0 ～ 3 個割り付けることができます。



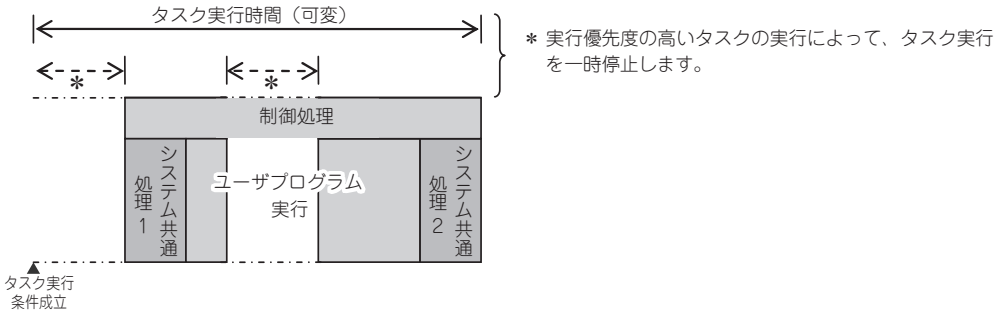
処理	処理内容
システム共通処理 1	• 変数のタスク間排他制御処理（参照タスクの場合） • データトレース処理（サンプリング、トリガ判定）
ユーザプログラム実行	• タスクに割り付けられたプログラムを割付順に実行します。
システム共通処理 2	• 変数のタスク間排他制御処理（更新タスクの場合） • タスク実行との同時性を確保させるためのコントローラ外部からの変数アクセス処理（[タスク設定] にて設定した「変数アクセス時間」分） • EtherNet/IP タグデータリンク処理の場合は、タグ（ネットワーク公開属性の変数）に対して更新タスクを設定すると、上記変数アクセス処理として実行します。

(注)上表の各セルの処理は、記載された順に実行されます。

● イベントタスク

指定した実行条件が成立したときに 1 回だけ実行されるタスクです。タスク全体で 0 ～ 32 個割り付けることができます。

イベントタスクの処理内容は、以下のとおりです。



処理	処理内容
システム共通処理 1	• 変数のタスク間排他制御処理（参照タスクの場合）*1
ユーザプログラム実行	• タスクに割り付けられたプログラムを割付順に実行します。
システム共通処理 2	• 変数のタスク間排他制御処理（更新タスクの場合）*1

*1. 変数のタスク間排他制御処理の詳細は、📖「5-6-1　　タスク間での変数の値の同時性を確保する方法」（P.5-40）を参照してください。

5-3-4　NY シリーズのイベントタスクの実行条件

イベントタスクは、指定した実行条件が成立したときに 1 回だけ実行されるタスクです。イベントタスクの実行条件には、つぎの 2 種類があります。

実行条件	イベントタスクが実行されるタイミング	使用する場合
ActEventTask 命令による実行	ActEventTask 命令実行時	• ユーザプログラム内で、どのイベントタスクを実行するかを明示したい場合 • イベントタスクの実行条件が、変数の条件式一致の判定までに変化する可能性がある場合
変数の条件式一致による実行	指定した変数の値が、指定した条件式に一致したとき*1	イベントタスク実行のためのユーザプログラムを省き、ユーザプログラムを簡略化したい場合

*1. 指定した変数の値が、指定した条件式に一致したかどうかを判定するタイミングについては、📖「実行条件が変数の条件式一致による場合の実行タイミング」（P.5-21）を参照してください。

安全上の要点

イベントタスクの実行条件が変数の条件式一致で、条件式に以下の変数を指定する場合、条件一致の時にイベントタスクが起動しない、もしくは、条件不一致の時にイベントタスクが起動されることがあります。

- モーション制御システム変数を除く、データサイズが 16 ビット以上の構造体のメンバ
- データサイズが 16 ビット以上の配列の要素

上記の変数を指定して、一致判定を行う場合は、以下のいずれかを行ってください。

- 64 ビット以外の基本データ型の内部変数にコピーし、コピーした内部変数を参照する。
- 変数のタスク間排他制御設定機能を使用し、プライマリ定周期タスクを更新タスクとする。

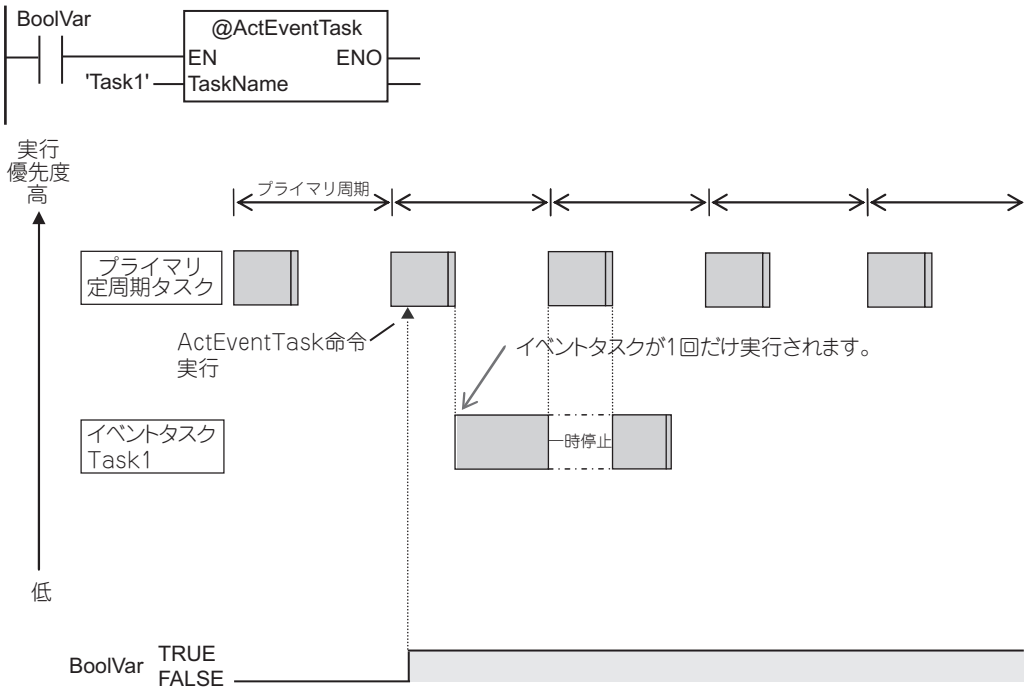
ActEventTask 命令によるイベントタスク実行

ユーザプログラム内で ActEventTask 命令が実行されると、指定したイベントタスクが 1 回実行されます。ActEventTask 命令の仕様の詳細は、📖『NY シリーズ コマンドリファレンスマニュアル 基本編（SBCA-437）』を参照してください。

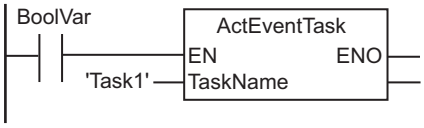
ActEventTask 命令による方法を使うと、ユーザプログラムを見たときに、どのイベントタスクを起動するかがよくわかります。また、イベントタスクの実行条件が、変数の条件式一致の判定までに変化する可能性がある場合にも有効です。

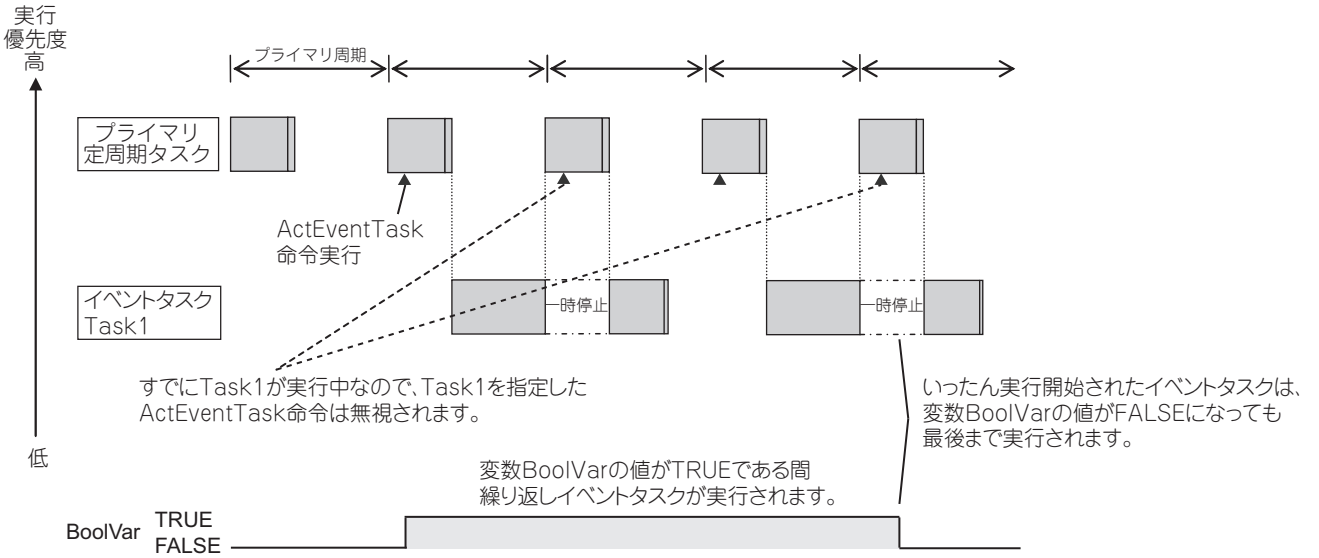
● ActEventTask 命令を使ったユーザプログラム例

例 1）変数の値が変化した際にイベントタスクを 1 回だけ実行する場合
下図のように、ActEventTask 命令に入力立ち上がり微分の動作オプションを付けると、BOOL 型変数 BoolVar の値が FALSE から TRUE に変化した際に、イベントタスク Task1 が 1 回だけ実行されます。



例 2）変数が指定した値である間、繰り返しイベントタスクを実行する場合
下図のように、ActEventTask 命令に入力立ち上がり微分の動作オプションを付けないと、BOOL 型変数 BoolVar の値が TRUE である間、繰り返しイベントタスク Task1 が実行されます。ただし、Task1 が実行中に、Task1 を指定した ActEventTask 命令を実行しても、無視されます。また、いったん実行開始されたイベントタスクは、実行途中に BoolVar の値が FALSE になっても、最後まで実行されます。





変数の条件式一致によるイベントタスク実行

変数の条件式一致による方法は、Sysmac Studio で指定した変数の値が、指定した条件式に一致した場合に、イベントタスクが 1 回実行されるというものです。その変数の値が条件式に一致している間、イベントタスクが実行され続けるのではなく、変数の値が条件式に不一致から一致に変化したときにイベントタスクが 1 回だけ実行されます。
変数の条件一致による方法では、イベントタスク実行のためのユーザプログラムが不要になります。

● 条件式が指定できる変数の種類

条件式が指定できる変数の種類は、以下のとおりです。

変数の種類		指定可否
システム定義変数		可*1
準ユーザ定義変数		可
ユーザ定義変数	グローバル変数	可
	プログラム内で使用する変数	可
	ファンクションブロック内で使用する変数	可*2
	ファンクション内で使用する変数	不可

*1. 以下の変数は除きます。
EN,ENO,P_Off,P_CY,P_First_RunMode,P_First_Run,P_PRGER
*2. 入出力変数は除きます。

● 条件式が指定できる変数のデータ型

条件式が指定できる変数のデータ型は、以下のとおりです。

データ型の分類	データ型		指定可否
基本データ型	ブール型、ビット列型、整数型、実数型		可
	持続時間型、日付型、時刻型、日付時刻型、文字列型		不可
データ型の指定	配列指定	配列全体	不可
		要素	可*1
派生データ型	構造体型	構造体全体	不可
		メンバ	可*2
	共用体型	共用体全体	不可
		メンバ	可*2
	列挙型		可

*1. 配列の要素のデータ型が、ブール型、ビット列型、整数型、実数型のいずれかの場合に限ります。
*2. メンバのデータ型が、ブール型、ビット列型、整数型、実数型のいずれかの場合に限ります。

● 指定できる条件式

指定できる条件式は、条件式を指定する変数のデータ型によって、以下のように決められています。条件式を指定する変数のデータ型がビット列型、整数型、実数型の場合は、変数の値と比較する比較定数を設定する必要があります。

データ型	指定できる条件式
ブール型、ブール型の配列要素、ブール型の構造体メンバ、ブール型の共用体メンバ	立ち上がり FALSE → TRUE
	立ち下がり TRUE → FALSE
ビット列型、整数型、実数型、 または、それらをデータ型とする配列要素、構造体メンバ、共用体メンバ	変数 = { 比較定数 }
	変数 ≠ { 比較定数 }
	変数 > { 比較定数 }
	変数 ≧ { 比較定数 }
	変数 < { 比較定数 }
	変数 ≦ { 比較定数 }

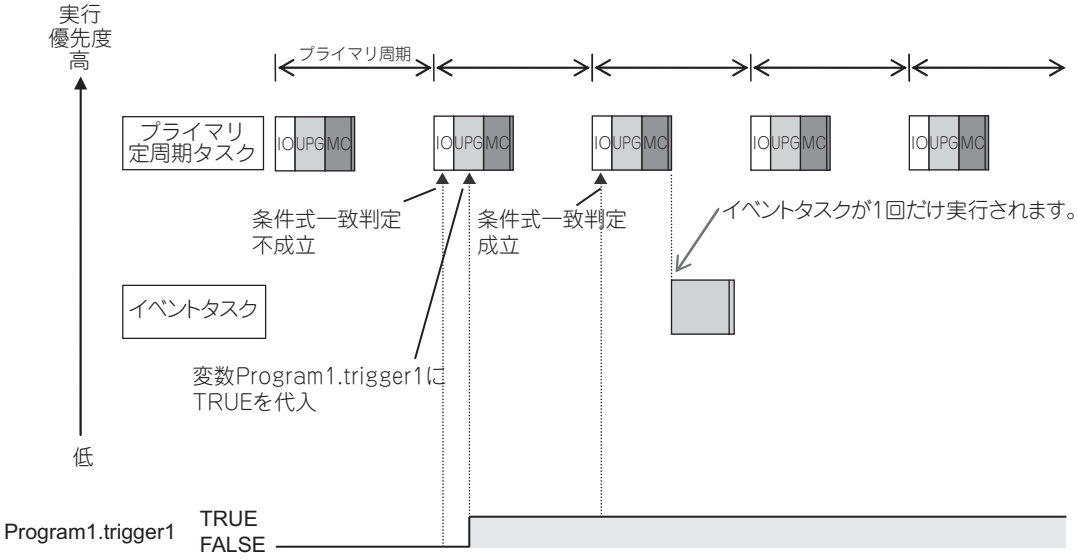
● 比較定数の値の有効範囲

条件式を指定する変数のデータ型がビット列型、整数型、実数型の場合は、変数の値と比較する比較定数を設定する必要があります。比較定数の値の有効範囲は、条件式を指定する変数のデータ型の値の有効範囲と同じです。

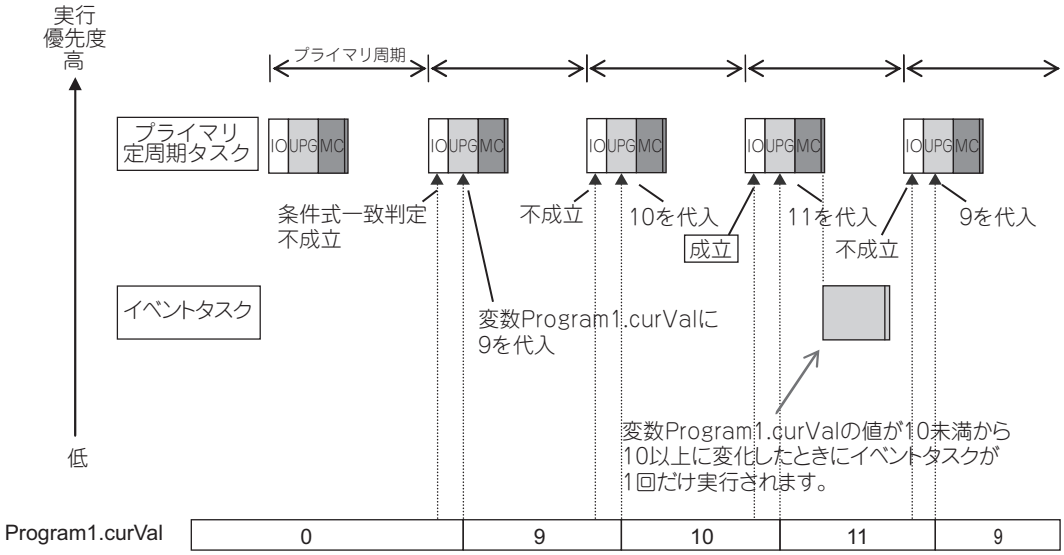
各データ型の値の有効範囲は、🕮「基本データ型と派生データ型」(P.6-30) を参照してください。たとえば、条件式を指定する変数のデータ型が BYTE の場合、比較定数の値の有効範囲は、BYTE#16#00 ～ BYTE#16#FF になります。

● 変数の条件式一致によるイベントタスク実行例

例 1）イベントタスクの実行条件を、BOOL 型変数 Program1.trigger1 の立ち上がりとした場合
Program1.trigger1 の値が、FALSE から TRUE に変化したときに、イベントタスクが 1 回だけ実行されます。



例 2) イベントタスクの実行条件を、INT 型変数 Program1.curVal ≧ 10 とした場合
Program1.curVal の値が、10 未満から 10 以上に変化したときに、イベントタスクが 1 回だけ実行されます。



使用上の注意

プライマリ定周期タスク内で、指定した変数の値が変化した場合、つぎのプライマリ定周期タスクで条件式との一致が判定されます。したがって、イベントタスクが実行されるのは、変数の値が条件式を満たした周期のつぎのプライマリ定周期タスクで、条件式一致の判定が行われた後です。

5-3-5　NY シリーズのイベントタスクの実行タイミング

イベントタスクの実行優先度は、8 または 48 です。ほかのタスクが実行中にイベントタスクの実行条件が成立した場合、実行優先度の高いタスクが先に実行され、実行優先度が低いタスクの実行は中断されます。これは、プライマリ定周期タスクや定周期タスクと同じです。

複数のタスク間で同じ変数の読み書きをする可能性がある場合は、必ず次の機能を使用し、プライマリ定周期タスク、定周期タスク、あるいは実行優先度の異なるイベントタスクとの実行を制御してください。

目的	使用する機能
イベントタスクと他のタスクで同じグローバル変数にアクセスする場合	変数のタスク間排他制御機能 • 変数のタスク間排他制御設定機能 • Lock（タスク間排他ロック）命令 • Unlock（タスク間排他ロック解除）命令
他のタスクの実行状態を確認する場合	Task_IsActive（タスク実行中判定）命令

詳細は、📖「5-6-1 タスク間での変数の値の同時性を確保する方法」（P.5-40）を参照してください。

イベントタスクの実行条件によってイベントタスクの実行タイミングが異なります。
また、イベントタスクは、複数のタスクに同じ実行優先度を割り当てることができます。同じ実行優先度の複数のイベントタスクの実行条件が成立した場合には、注意が必要です。

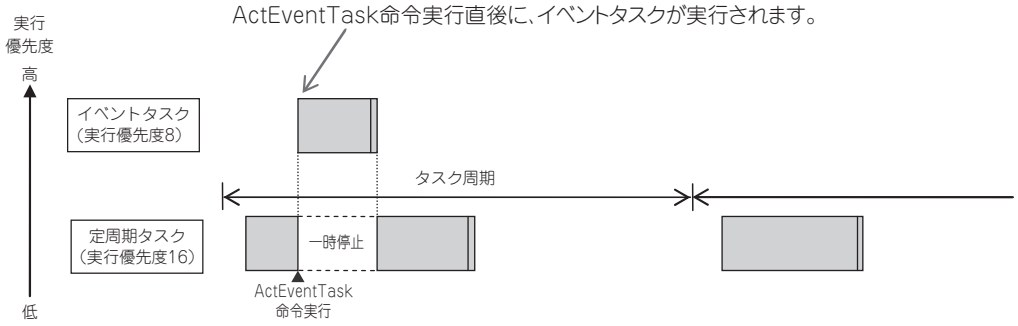
イベントタスクの実行条件による実行タイミングの違い

イベントタスクの実行条件が、ActEventTask 命令による場合と、変数の条件式一致による場合とでは、実行条件の成立タイミングが異なります。

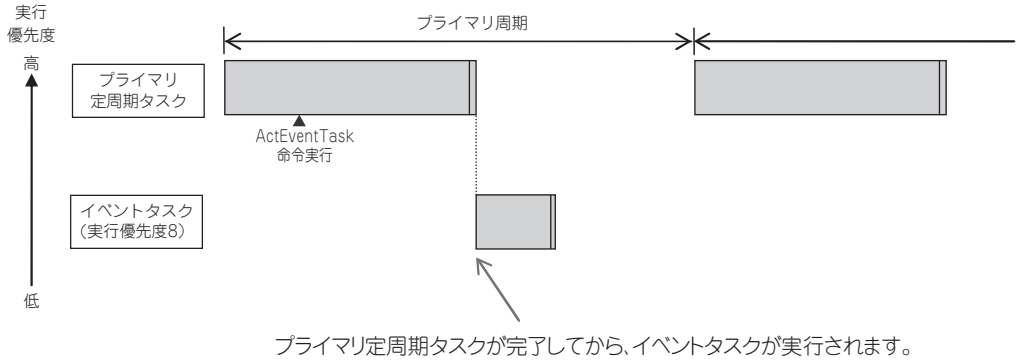
● 実行条件が ActEventTask 命令による場合の実行タイミング

イベントタスクの実行条件が、ActEventTask 命令による場合は、ActEventTask 命令の実行直後に、イベントタスクの実行条件が成立します。実行条件が成立したイベントタスクは、タスクの実行優先度に従って実行が開始されます。

例 1）命令を実行するタスクより実行優先度の高いイベントタスクを起動する場合



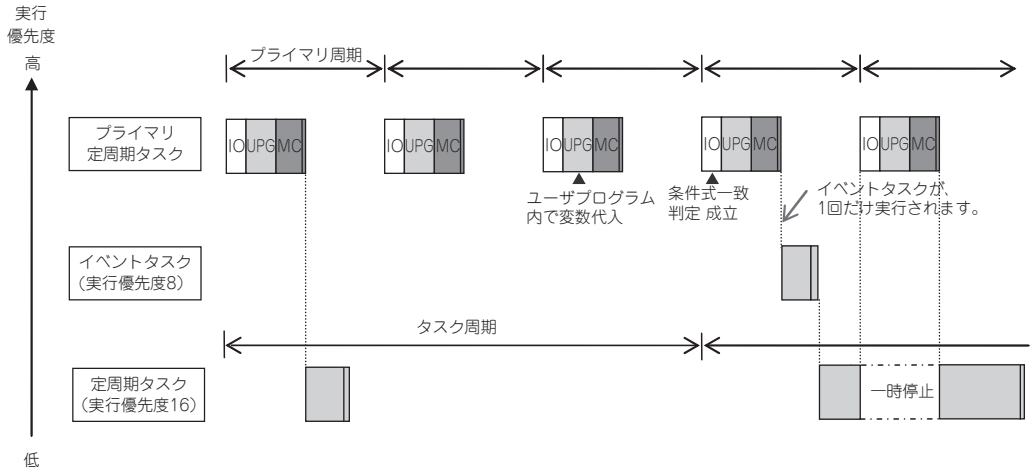
例 2）命令を実行するタスクより実行優先度の低いイベントタスクを起動する場合



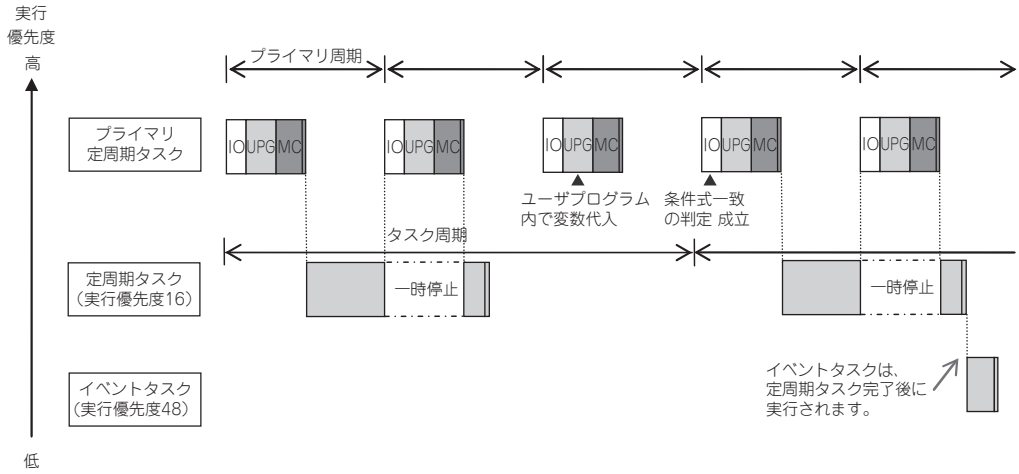
● 実行条件が変数の条件式一致による場合の実行タイミング

条件式一致の判定は、プライマリ定周期タスクの中で行われます。条件式の判定処理で条件が一致したときに、イベントタスクの実行条件が成立します。実行条件が成立したイベントタスクは、タスクの実行優先度に従って実行が開始されます。

例 1）定周期タスク（実行優先度 16）とイベントタスク（実行優先度 8）が存在する場合
イベントタスク（実行優先度 8）の方が定周期タスク（実行優先度 16）より実行優先度が高いので、定周期タスク（実行優先度 16）は、イベントタスクの後に実行されます。



例 2）定周期タスク（実行優先度 16）とイベントタスク（実行優先度 48）が存在する場合
イベントタスクの方が定周期タスク（実行優先度 16）より実行優先度が低いので、定周期タスク（実行優先度 16）が完了してからイベントタスクが実行されます。



使用上の注意

- NY シリーズコントローラでは、条件式一致が、プライマリ定周期タスクの I/O リフレッシュによって発生する場合、およびプライマリ定周期タスクに割り付けられたユーザプログラム実行で発生する場合とも、イベントタスクの実行条件が成立するタイミングは以下のように同じです。

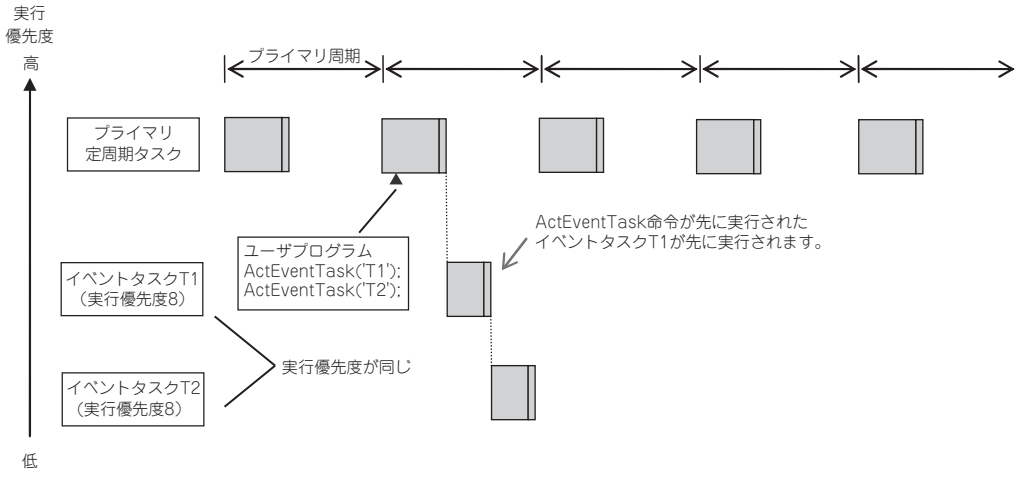
条件式一致発生要因	イベントタスクの実行条件が成立するタイミング
プライマリ定周期タスクの I/O リフレッシュ	つぎのプライマリ定周期タスクの判定処理
プライマリ定周期タスクのユーザプログラム実行	つぎのプライマリ定周期タスクの判定処理

- イベントタスクが実行されるのは、1 回前の判定時に条件式不一致で、つぎの判定時に条件式一致が起こった場合です。したがって、不一致から一致への変化があったとしても、つぎの判定時までにまた不一致に戻った場合、イベントタスクは実行されません。

同じ実行優先度のイベントタスクの実行タイミング

イベントタスクは、複数のタスクに同じ実行優先度を割り当てることができます。ActEventTask 命令により、同じ実行優先度の複数のイベントタスクの実行条件が成立した場合は、命令を実行した順にイベントタスクが実行されます。

例 1）2 つの ActEventTask 命令が実行された場合
下図のように、先に実行された ActEventTask 命令によって指定されたイベントタスク T1 が、イベントタスク T2 よりも先に実行されます。



例 2）変数の条件式一致と ActEventTask 命令が混在する場合
イベントタスク T1、T2、T3 の実行条件が、以下のように設定されているとします。

- T1：変数 VarA の条件式一致
- T2：変数 VarB の条件式一致
- T3：ActEventTask 命令

下図のように、同じプライマリ周期中に、I/O リフレッシュによる VarA の条件式一致、ユーザプログラムでの ActEventTask 命令実行、ユーザプログラムでの VarB の条件式一致が発生した場合は、つぎの順番で動作します。

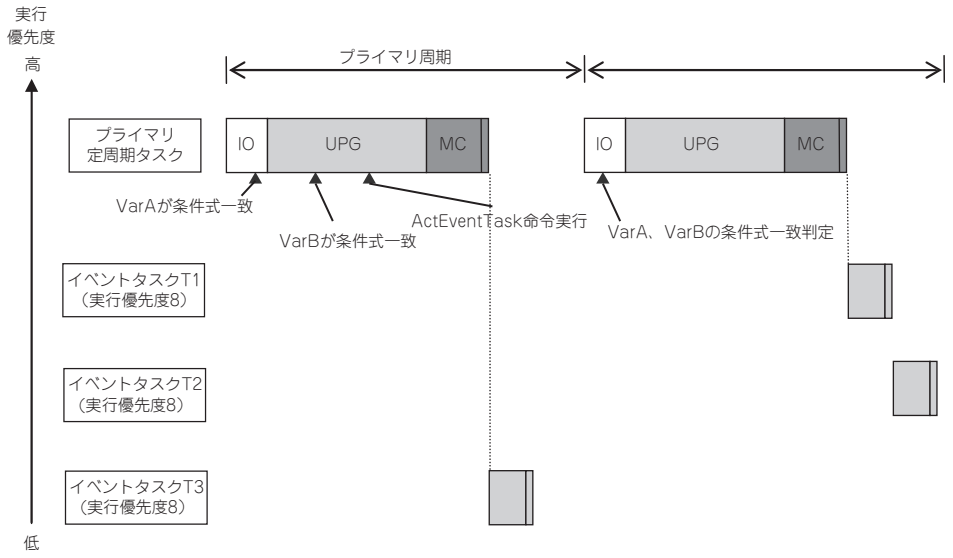
(1) I/O リフレッシュで、VarA が条件式に一致します。

(2) ユーザプログラムで、VarB が条件式に一致します。

(3) この時点では、まだ条件式の一致判定が行なわれていないため、T1 と T2 は実行されません。

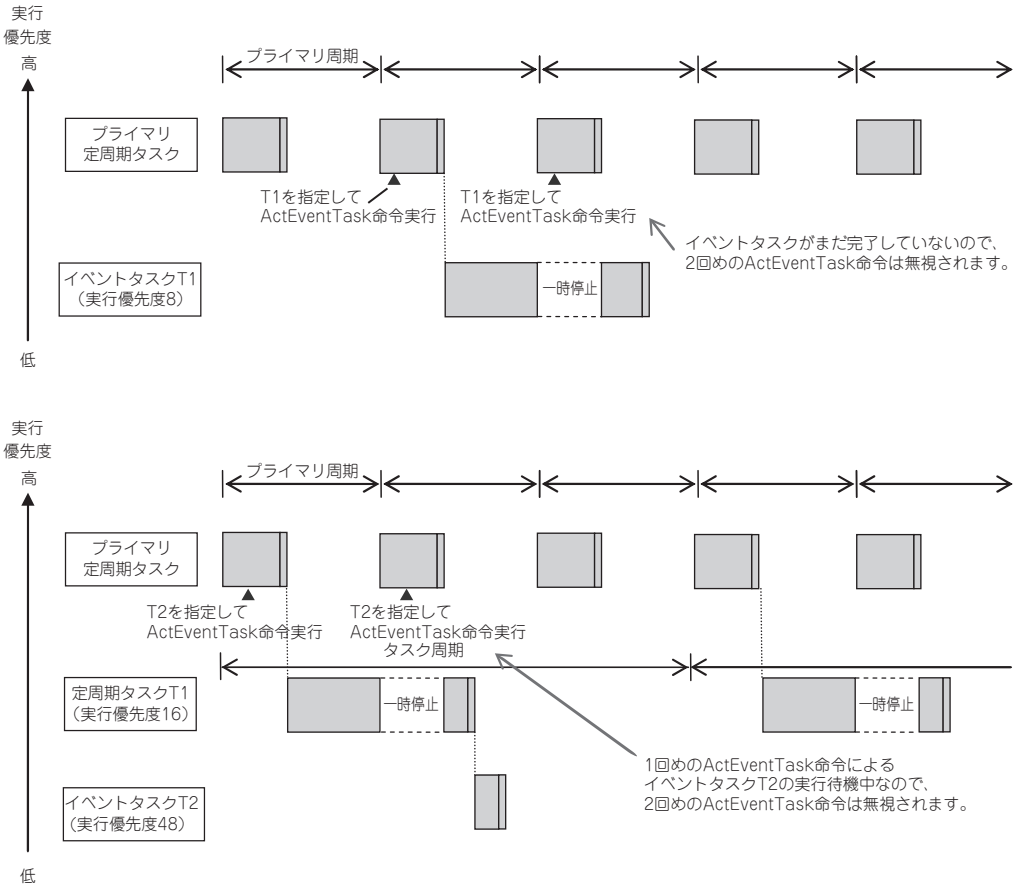
(4) ユーザプログラムで、ActEventTask 命令が実行され、T3 が実行されます。

(5) つぎのプライマリ定周期タスクのリフレッシュ実行で、VarA と VarB が条件式に一致していると判定され、T1 と T2 が実行されます。
同じ周期に、複数の条件式が一致していると判定された場合、イベントタスクの実行順序は不定となります。下図は、T1 が先に実行されたときです。



5-3-6　イベントタスク完了前に再度実行条件が成立した場合の動作

イベントタスク完了前に、再度そのイベントタスクの実行条件が成立した場合、2 回めの実行条件成立は無視されます。イベントタスク完了前とは、イベントタスク実行中およびイベントタスク実行待機中の両方を含みます。また、いったん実行開始されたイベントタスクは、実行条件が不成立になっても、最後まで実行されます。



5-4　タグデータリンクサービスとシステムサービス

NY シリーズコントローラは、プライマリ定周期タスク、定周期タスク、イベントタスク以外にも、処理を行います。これらの処理には、タグデータリンクサービスとシステムサービスの 2 種類があります。この 2 つの、処理内容、実行優先度、実行タイミングは、以下のとおりです。

項目	タグデータリンクサービス	システムサービス
実行優先度	定周期タスク（実行優先度 16）と定周期タスク（実行優先度 17）の間です。	すべてのタスクより実行優先度が低いです。
処理内容	EtherNet/IP ネットワーク上で、他のコントローラやデバイスと、タグを用いてデータ交換を行います。NY シリーズの内蔵 EtherNet/IP ポートを使用します。*1	通信処理など、実行優先度が低い処理の総称です。
実行タイミング	実行間隔や、1 回の処理の実行時間は、NY シリーズの形式やタグデータリンク設定によって変動します。*2	NY シリーズコントローラでは、タスクなどの影響を受けずに、いつでも実行されます。

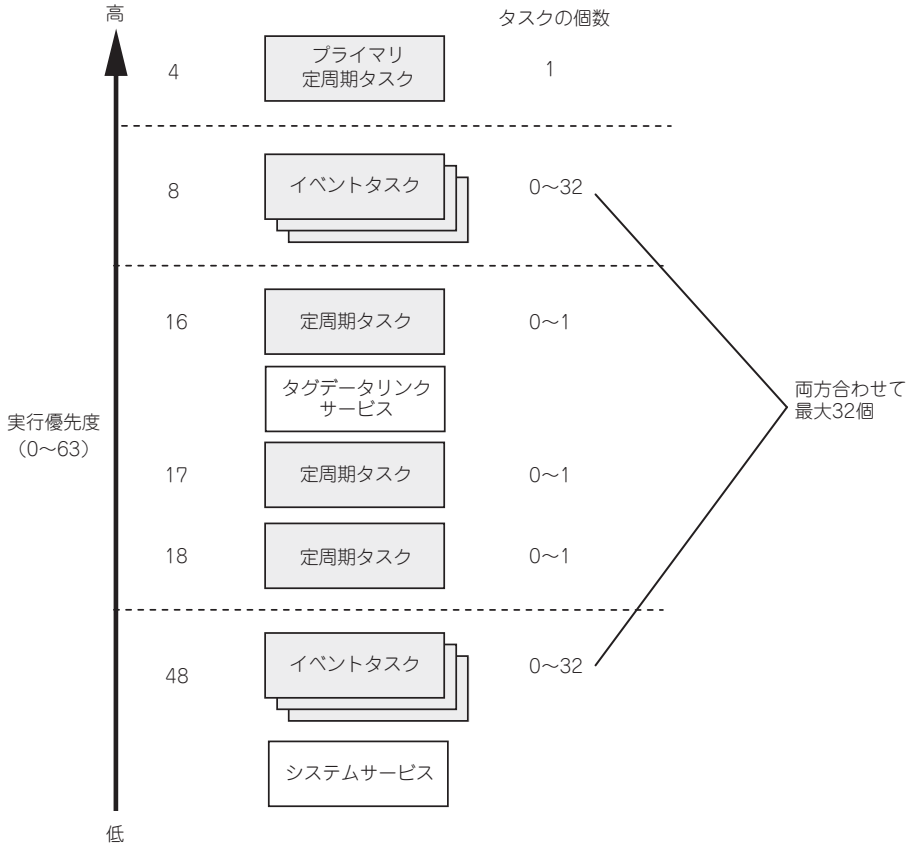
*1. タグデータリンクの処理内容の詳細は、🕮『NYシリーズ 産業用パネル型PC／産業用ボックス型PC ユーザーズマニュアル 内蔵 EtherNet/IP ポート編 （SBCD-369)』を参照してください。
*2. タグデータリンクの実行間隔と実行時間の詳細は、🕮「5-4-2 タグデータリンクサービスの処理内容と実行タイミング」（P.5-28）を参照してください。

5-4-1　タグデータリンクサービスとシステムサービスの実行優先度と実行順序

ここでは、タグデータリンクサービスとシステムサービスの実行優先度と実行順序の例を説明します。

タグデータリンクサービスとシステムサービスの実行優先度

タグデータリンクサービスとシステムサービスの実行優先度は、下図のとおりです。タグデータリンクサービスの実行優先度は、定周期タスク（実行優先度 16）と定周期タスク（実行優先度 17）の間です。システムサービスの実行優先度は、すべてのタスクよりも低いです。

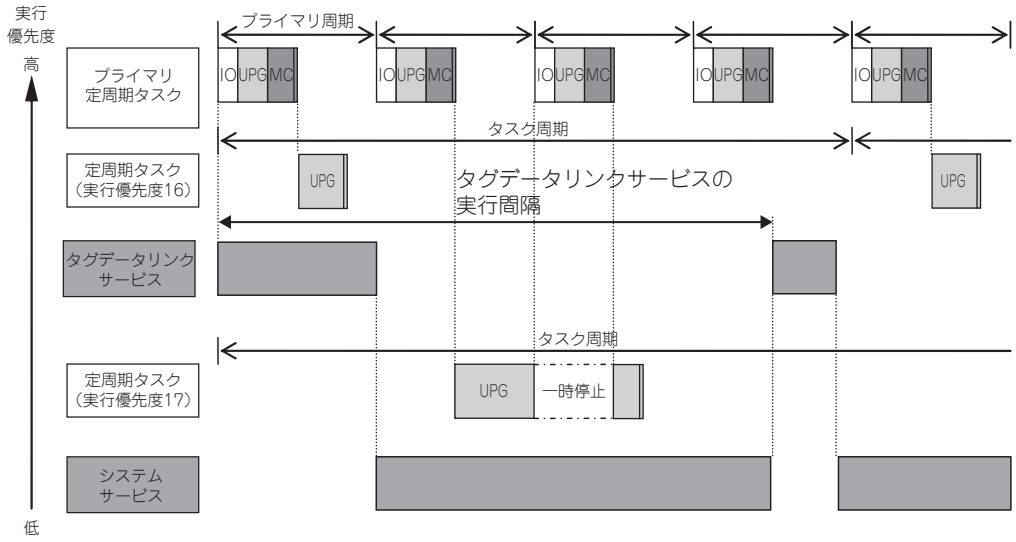


タグデータリンクサービスとシステムサービスの実行順序の例

プライマリ定周期タスク、定周期タスク（実行優先度 16）、定周期タスク（実行優先度 17）、タグデータリンクサービス、システムサービスが混在する場合の、それぞれの実行順序の例を以下に示します。

● NY シリーズコントローラの場合

- タグデータリンクサービスとシステムサービスは、タスクの実行と並列に実行することができます。
- タグデータリンクサービスの実行中は、システムサービスを実行しません。



タグデータリンクサービスの実行間隔や 1 回の実行時間は、NY シリーズの形式やタグデータリンク設定によって変動します。詳細は、🕮「5-4-2 タグデータリンクサービスの処理内容と実行タイミング」（P.5-28）を参照してください。

5-4-2　タグデータリンクサービスの処理内容と実行タイミング

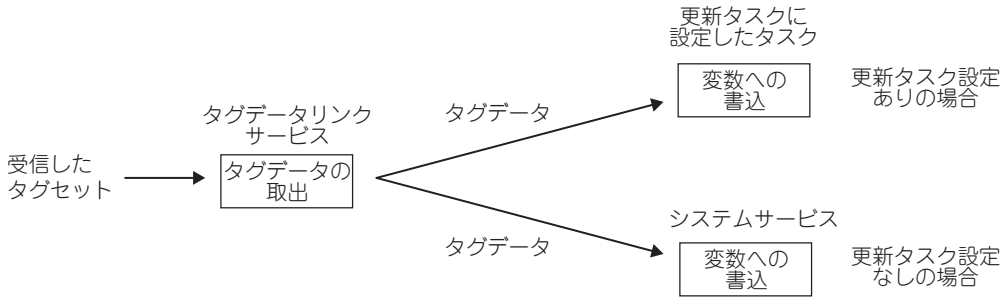
ここでは、タグデータリンクサービスの処理内容と実行タイミングを説明します。

タグデータリンクサービスの処理内容

タグデータリンク処理は、複数の処理に細分化されています。それらの処理は、タグデータリンクサービス、システムサービス、タスクによって実行されます。NY シリーズ内蔵 EtherNet/IP ポートを使用してタグデータリンク処理を行う場合の例を、つぎに示します。

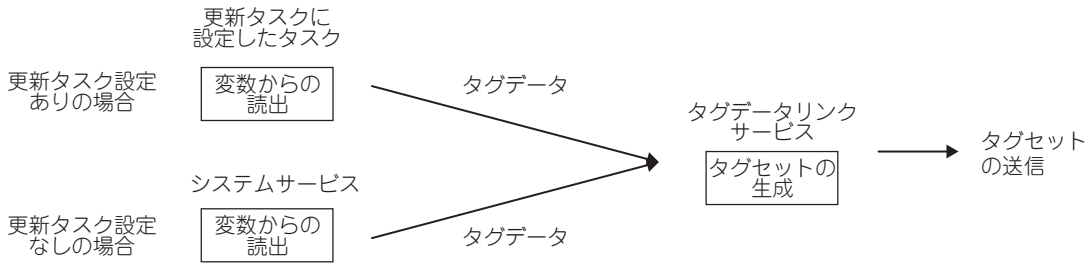
● タグデータ受信処理の流れ

- タグデータリンクサービスによって、受信したタグセットからタグデータが取り出されます。
- 更新タスクを設定している場合、そのタスクによって、タグデータの値がタグに割り付けた変数に書き込まれます。
- 更新タスクを設定していない場合、システムサービスによって、タグデータの値がタグに割り付けた変数に書き込まれます。



● タグデータ送信処理の流れ

- 更新タスクを設定している場合、そのタスクによって、タグデータの値がタグに割り付けた変数から読み出されます。
- 更新タスクを設定していない場合、システムサービスによって、タグデータの値がタグに割り付けた変数から読み出されます。
- その後、タグデータリンクサービスによって、タグデータからタグセットが生成され、送信されます。



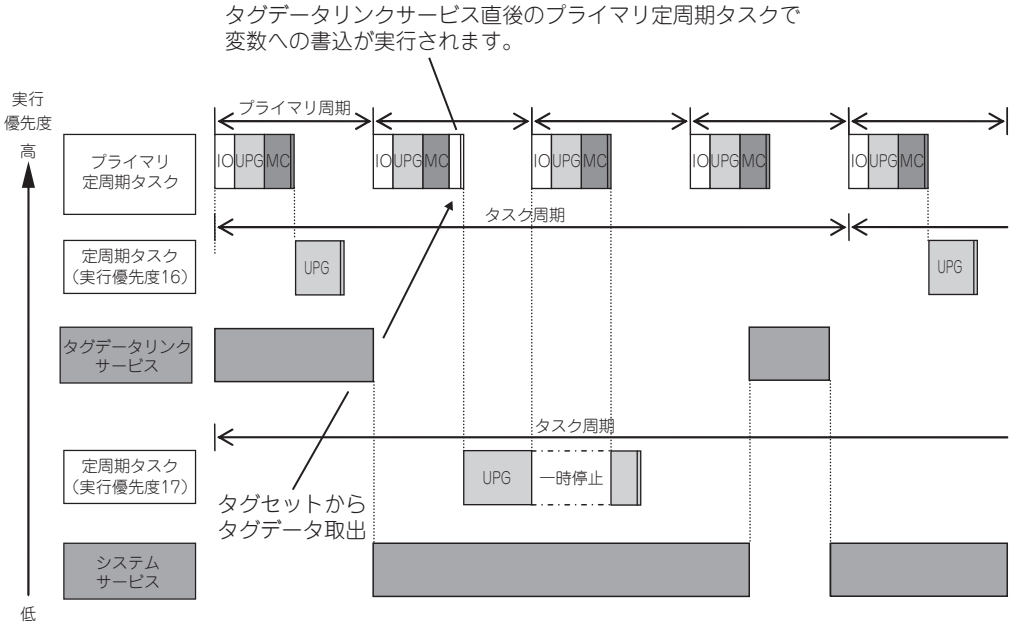
参考

更新タスク設定ありの場合となしの場合の、タグデータリンク処理のタイミングの違い

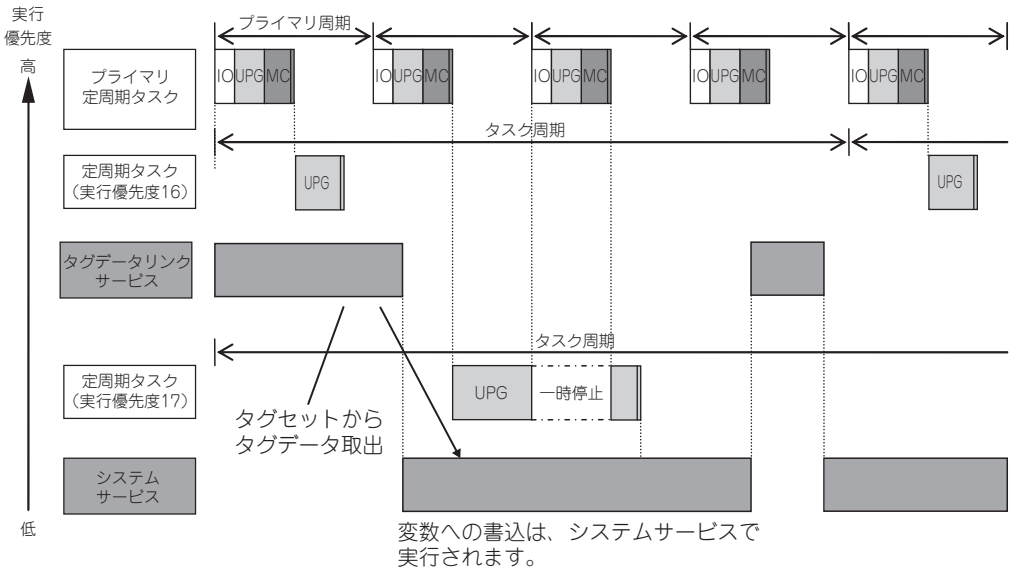
更新タスク設定ありの場合となしの場合とでは、変数への値の読み書きを行う処理が異なります。更新タスク設定ありの場合はそのタスク、なしの場合はシステムサービスによって、値の読み書きが行われます。したがって、タグデータリンク処理の実行タイミングにも違いが発生します。

以下は、NY シリーズコントローラでのタグデータリンク受信の場合の、実行タイミングの違いです。

- 更新タスク設定ありの場合
  プライマリ定周期タスクを更新タスクに設定している場合の例です。変数への書込は、タグデータリンクサービス直後のプライマリ定周期タスクで実行されます。



- 更新タスク設定なしの場合
  変数への書込は、システムサービスで実行されます。



5-4-3　システムサービスの処理内容と実行タイミング

ここでは、システムサービスの処理内容と実行タイミングを説明します。

システムサービスの処理内容

システムサービスの処理には、以下のものがあります。

システムサービスの処理名	内容
内蔵 EtherNet/IP ポートサービス	• 周辺ツール、表示器、上位パソコン、または他コントローラなどからのサービス要求（CIP など）に対する処理 • 通信命令（CIP、ソケット）の実行
内部通信ポートサービス	• NY シリーズ内部の Windows からのサービス要求（CIP、HTTP）に対する処理
内蔵 EtherCAT ポートサービス	• EtherCAT メッセージ通信の実行
仮想 SD メモリカードサービス	• FTP クライアントからのアクセス • Sysmac Studio からの SD メモリカード操作 • SD メモリカード命令の実行
自己診断処理	• ハードウェアの異常検知

システムサービスの実行タイミング

NY シリーズコントローラのシステムサービスは、タスクの影響を受けません。ただし、タグデータリンクサービスの実行中は、システムサービスを実行しません。

システムサービスの処理どうしに、優先順位はありません。各処理はタイムスライスされて、並列に実行されます。

5-5　タスク関連の割付と設定

ここでは、タスクに関する割付と設定について説明します。

5-5-1　I/O リフレッシュのタスクへの割付

EtherCAT スレーブの I/O リフレッシュを、タスクに割り付けます。I/O リフレッシュ対象によって、割付単位が異なります。割付単位とは、ひとつの I/O リフレッシュに割り付けることができる、I/O リフレッシュ対象のまとまりのことです。たとえば、割付単位がスレーブターミナルの場合、1 台の通信カプラユニットに複数の NX ユニットを接続していても、I/O リフレッシュを割り付けられるタスクはひとつだけです。

I/O リフレッシュを割り付けていないタスクで入出力を行いたい場合は、🕮「I/O リフレッシュを割り付けていないタスクの入出力」（P.5-32）を参照してください。

I/O リフレッシュ対象、割付可能なタスク、割付単位の関係は、次のとおりです。

I/O リフレッシュ対象	割付可能なタスク	割付単位
通信カプラユニット	プライマリ定周期タスク	スレーブターミナル
EtherCAT スレーブ	プライマリ定周期タスク	スレーブ

Sysmac Studio での設定操作

軸に割り付けていない各スレーブ／ユニットは、Sysmac Studio の [タスク設定]-[I/O を制御するタスク設定] で I/O リフレッシュするタスクを設定します。

詳細は、🕮「I/O を制御するタスク設定」（P.4-8）を参照してください。

軸に割り付けている各スレーブ／ユニットは、Sysmac Studio の［モーション制御設定］で、使用するモーション制御を選択すると、I/O リフレッシュするタスクが設定されます。

詳細は、🕮『NY シリーズ 産業用パネル型 PC ／産業用ボックス型 PC ユーザーズマニュアル モーション制御編（SBCE-379)』を参照してください。

I/O リフレッシュのタイミング

EtherCAT スレーブの I/O リフレッシュのタイミングは、次のとおりです。

I/O リフレッシュ対象	I/O を制御するタスク	I/O リフレッシュの周期
EtherCAT スレーブ	プライマリ定周期タスク	プライマリ定周期タスクのタスク周期

ユーザプログラムから I/O にアクセスする方法

ユーザプログラムから I/O ポートにアクセスするには、デバイス変数を使用します。

［I/O を制御するタスク設定］に設定したタスクのプログラムからデバイス変数にアクセスしてください。

I/O リフレッシュを割り付けていないタスクの入出力

I/O リフレッシュを割り付けていないタスクからデータを出力しようとすると、Sysmac Studio のプログラムチェック時にエラーになります。たとえば、スレーブターミナルで、各 NX ユニットの処理を異なるタスクに割り付けている場合、I/O リフレッシュを割り付けているタスクからしか、データを出力できません。このような場合は、I/O リフレッシュを割り付けていないタスクから I/O リフレッシュを割り付けたタスクにデータを渡し、そのうえで I/O リフレッシュを割り付けたタスクからデータを出力する必要があります。
次に、I/O リフレッシュを割り付けていないタスクから I/O リフレッシュを割り付けたタスクにデータを渡して出力するサンプルプログラムを示します。
なお、本サンプルプログラムでは、I/O リフレッシュを割り付けていないタスクで、外部データを入力する処理を行っています。このような処理を行おうとすると Sysmac Studio のプログラムチェック時に警告が出ますが、実行は可能です。

● ユニット構成

スレーブターミナルを使用します。スレーブターミナルのユニット構成は、以下のとおりです。

形式	製品名称
NX-ECC20□	EtherCAT カプラユニット
NX-ID3317	DC 入力ユニット
NX-OD3256	トランジスタ出力ユニット

● I/O マップ

I/O マップは、以下のとおりです。表には、サンプルプログラムで使用する接点だけを示します。

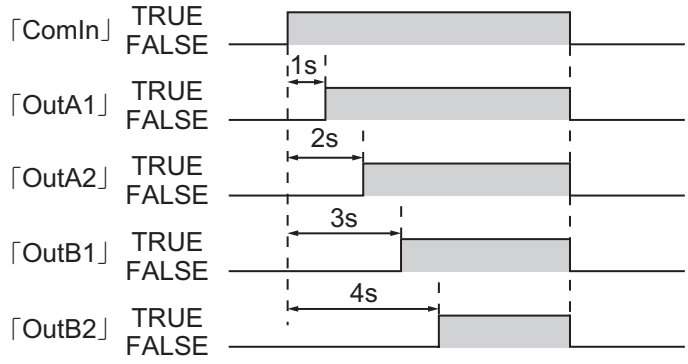
位置	ポート	説明	R/W	データ型	変数	変数コメント	変数種別
Unit1	NX-ID3317						
	Input Bit 00	入力接点 00	R	BOOL	ComIn	共通入力値	グローバル変数
Unit2	NX-OD3256						
	Output Bit 00	出力接点 00	W	BOOL	OutA1	出力値 A1	グローバル変数
	Output Bit 01	出力接点 01	W	BOOL	OutA2	出力値 A2	グローバル変数
	Output Bit 02	出力接点 02	W	BOOL	OutB1	出力値 B1	グローバル変数
	Output Bit 03	出力接点 03	W	BOOL	OutB2	出力値 B2	グローバル変数

● 入出力仕様

入出力仕様は、以下のとおりです。

- 「ComIn」の値が FALSE から TRUE に変化してから、1 秒後に「OutA1」、2 秒後に「OutA2」、3 秒後に「OutB1」、4 秒後に「OutB2」の値が、それぞれ FALSE から TRUE に変化します。
- 「ComIn」の値が TRUE から FALSE に変化したら、「OutA1」、「OutA2」、「OutB1」、「OutB2」のすべての値が、TRUE から FALSE に変化します。

タイミングチャートを、以下に示します。



● タスクの処理内容

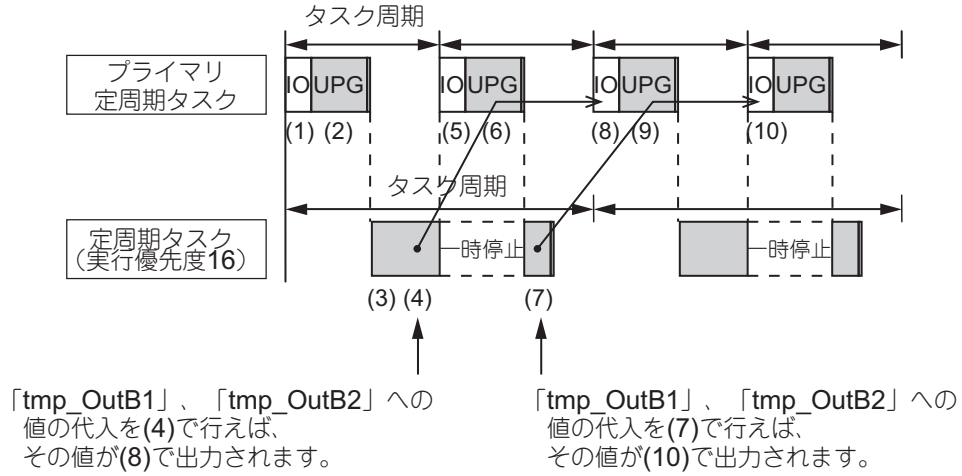
タスクには、プライマリ定周期タスクと定周期タスク（実行優先度 16）を使用します。プライマリ定周期タスクに、I/O リフレッシュを割り付けます。

「OutA1」、「OutA2」の制御は、プライマリ定周期タスクで行います。「ComIn」の入力から演算、「OutA1」、「OutA2」の出力までの一連の処理を、すべてプライマリ定周期タスクで行います。

「OutB1」、「OutB2」の制御は、「ComIn」の入力から演算までを定周期タスクで行います。定周期タスクでの演算結果は、一時変数のグローバル変数「tmp_OutB1」、「tmp_OutB2」に代入します。プライマリ定周期タスクでは、「tmp_OutB1」、「tmp_OutB2」の値を「OutB1」、「OutB2」に代入した後、「OutB1」、「OutB2」が出力されます。

定周期タスクは、プライマリ定周期タスクの実行によって、一時停止する場合があります。定周期タスクでの演算結果を「tmp_OutB1」、「tmp_OutB2」に代入する処理を、定周期タスクの一時停止の前に行うか後に行うかによって、定周期タスクの演算結果がプライマリ定周期タスクによって出力されるタイミングが変わります。

処理の流れの例を、下図に示します。定周期タスクのタスク周期は、プライマリ定周期タスクのタスク周期の 2 倍であるとします。



(1) プライマリ定周期タスクの I/O リフレッシュで、「ComIn」の入力が行われます。

(2) プライマリ定周期タスクのユーザプログラム実行で演算を行い、「OutA1」、「OutA2」に値を代入します。
また、「tmp_OutB1」、「tmp_OutB2」の値を、「OutB1」、「OutB2」に代入します。しかし、まだ定周期タスクで「tmp_OutB1」、「tmp_OutB2」に値を代入していないため、「tmp_OutB1」、「tmp_OutB2」の値は初期値となっています。したがって、「OutB1」、「OutB2」の値も初期値になります。

(3) 定周期タスクで、「ComIn」の入力が行われます。

(4) 定周期タスクのユーザプログラム実行で演算を行います。
定周期タスクの実行中にプライマリ定周期タスクの実行が開始されると、定周期タスクは一時停止します。「tmp_OutB1」、「tmp_OutB2」への値の代入を、定周期タスクの一時停止の前に行うか後に行うかは、定周期タスク内の処理のタイミングによって変わります。

(5) プライマリ定周期タスクの I/O リフレッシュで、「OutA1」、「OutA2」、「OutB1」、「OutB2」の出力が行われます。
また、2 回めの「ComIn」の入力が行われます。

(6) プライマリ定周期タスクのユーザプログラム実行で演算を行い、「OutA1」、「OutA2] に値を代入します。
また、「tmp_OutB1」、「tmp_OutB2」の値を、「OutB1」、「OutB2」に代入します。(4) で「tmp_OutB1」、「tmp_OutB2」への値の代入を行っていた場合は、定周期タスクの演算結果が「OutB1」、「OutB2」に反映されます。(4) ではまだ「tmp_OutB1」、「tmp_OutB2」への値の代入を行っていなかった場合は、「tmp_OutB1」、「tmp_OutB2」の値は初期値となっています。したがって、「OutB1」、「OutB2」の値も初期値になります。

(7) 定周期タスクのユーザプログラム実行で、(4) の続きの演算を行います。
(4) で「tmp_OutB1」、「tmp_OutB2」への値の代入を行っていなかった場合、ここで「tmp_OutB1」、「tmp_OutB2」への値の代入を行います。

(8) プライマリ定周期タスクの I/O リフレッシュで、「OutA1」、「OutA2」、「OutB1」、「OutB2」の出力が行われます。
(4) で「tmp_OutB1」、「tmp_OutB2」への値の代入を行った場合は、定周期タスクの演算結果が「OutB1」、「OutB2」として出力されます。
(7) で「tmp_OutB1」、「tmp_OutB2」への値の代入を行った場合は、「OutB1」、「OutB2」の初期値が出力されます。

(9) プライマリ定周期タスクのユーザプログラム実行で演算を行い、「OutA1」、「OutA2] に値を代入します。
また、「tmp_OutB1」、「tmp_OutB2」の値を、「OutB1」、「OutB2」に代入します。(7) で「tmp_OutB1」、「tmp_OutB2」への値の代入を行った場合は、ここで定周期タスクの演算結果が「OutB1」、「OutB2」に反映されます。

(10) プライマリ定周期タスクの I/O リフレッシュで、「OutA1」、「OutA2」、「OutB1」、「OutB2」の出力が行われます。
(7) で「tmp_OutB1」、「tmp_OutB2」への値の代入を行った場合は、ここで定周期タスクの演算結果が「OutB1」、「OutB2」として出力されます。

また、「tmp_OutB1」、「tmp_OutB2」の値が、プライマリ定周期タスク内で参照される前に定周期タスクによって上書きされることを防ぐために、Lock、Unlock 命令を使用して、タスク間の排他制御を行います。タスク間の排他制御の詳細については、📖「5-6-1 タスク間での変数の値の同時性を確保する方法」（P.5-40）を参照してください。

● グローバル変数テーブル

グローバル変数は、以下のとおりです。

グローバル変数の変数テーブル

名称	データ型	初期値	割付先	コメント
ComIn	BOOL	FALSE	ECAT://node#[1,1]/Input Bit 00	共通入力値
OutA1	BOOL	FALSE	ECAT://node#[1,2]/Output Bit 00	出力値 A1
OutA2	BOOL	FALSE	ECAT://node#[1,2]/Output Bit 01	出力値 A2
OutB1	BOOL	FALSE	ECAT://node#[1,2]/Output Bit 02	出力値 B1
OutB2	BOOL	FALSE	ECAT://node#[1,2]/Output Bit 03	出力値 B2
tmp_OutB1	BOOL	FALSE		B1 の一時変数
tmp_OutB2	BOOL	FALSE		B2 の一時変数

● プライマリ定周期タスクのラダー図

プライマリ定周期タスクのラダー図を、以下に示します。

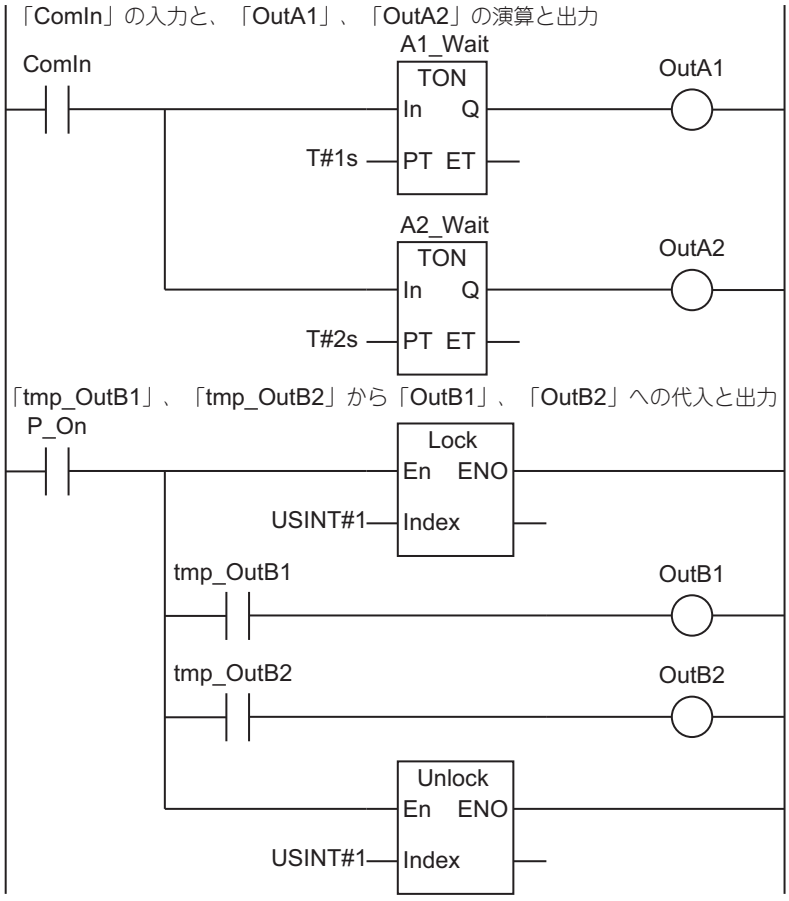
内部変数テーブル

名称	データ型
A1_Wait	TON
A2_Wait	TON

外部変数テーブル

名称	データ型	コメント
ComIn	BOOL	共通入力値
OutA1	BOOL	出力値 A1
OutA2	BOOL	出力値 A2
OutB1	BOOL	出力値 B1
OutB2	BOOL	出力値 B2
tmp_OutB1	BOOL	B1 の一時変数
tmp_OutB2	BOOL	B2 の一時変数

アルゴリズム部



● 定周期タスクのラダー図

定周期タスクのラダー図を、以下に示します。

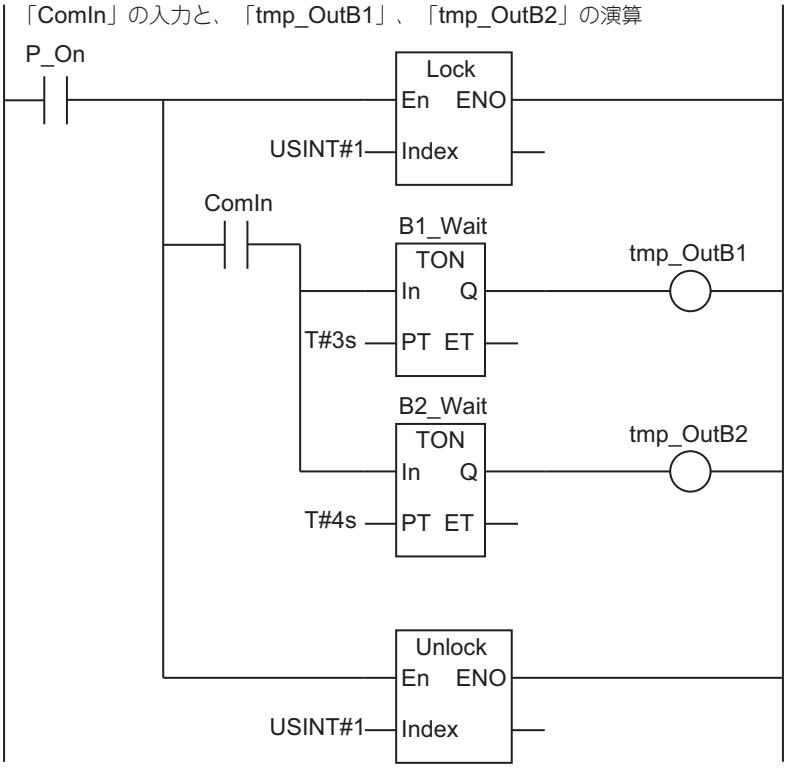
内部変数テーブル

名称	データ型
B1_Wait	TON
B2_Wait	TON

外部変数テーブル

名称	データ型	コメント
ComIn	BOOL	共通入力値
tmp_OutB1	BOOL	B1 の一時変数
tmp_OutB2	BOOL	B2 の一時変数

定周期タスクのアルゴリズム部



5-5-2　プログラムのタスクへの割付

実行するプログラムを、タスクに割り付けます（1 タスクあたり最大 128 個）。また、各プログラムの運転開始時の動作を設定します。

プログラムの実行順

タスク内でのプログラムの実行順序を、Sysmac Studio で設定します。

プログラムの運転開始時初期状態

各プログラムの運転開始時の動作を設定します。運転開始時初期状態とは、コントローラの動作モードがプログラムモードから運転モードに変更された後、当該プログラムが割り付けられたタスクが最初に実行された際に、当該プログラムを実行させるかどうかの設定のことです。設定範囲は、「起動」、「停止」のいずれかです。
初期状態が「停止」の場合、PrgStart 命令で実行を指令すると、つぎに当該プログラムが実行されるタイミングから、実行されます。初期状態が「起動」の場合、PrgStop 命令で停止を指令すると、つぎに当該プログラムが実行されるタイミングから、停止します。

● Sysmac Studio での設定操作

Sysmac Studio から、[構成・設定]-[タスク設定]-[プログラムの割付設定] にて、タスクへのプログラムの割付、タスク内でのプログラムの実行順序、および各プログラムの初期状態を設定します。詳細は、🕮「プログラムの割付設定」（P.4-9）を参照してください。

タスクに割り付けることができる POU

タスクには、0 ～ 128 個のプログラムを割り付けることができます。
割付可能な POU はプログラムだけです。FB インスタンスおよび、FUN を直接タスクに割り付けることはできません。同じプログラムを、複数のタスクに割り付けることはできません。

5-5-3　タスクの設定項目

各タスクの設定項目は、以下のとおりです。

● プライマリ定周期タスクの設定項目

設定項目		設定範囲	デフォルト	反映タイミング	運転モード中書替
タスクタイプ		「プライマリ定周期タスク」を選択します。	－	設定転送時	不可
	実行優先度	「4」に固定	－		
タスク名		文字列	PrimaryTask		
周期／実行条件	タスク周期[*1]	500μs～8ms（250μs単位）	1ms		
タスク周期オーバー検知		タスクの実行時間が設定したタスク周期を超えた場合にエラーを検知するか否かを設定します。 • 検知する（軽度フォールトレベルのコントローラ異常発生） • 検知しない（監視情報としてイベントログに残す） 詳細は、🕮「タスク周期オーバー」（P.5-50）を参照してください。	検知する		
タスク実行タイムアウト時間		無限ループなどによりタスクの実処理が完了しないときのタイムアウト検知時間を設定します。 タスク周期の倍数で設定します。 1倍～5倍 詳細は、🕮「タスク実行タイムアウト」（P.5-51）を参照してください。	5倍		
変数アクセス時間［%］		変数アクセス時間のタスク周期に対する割合を設定します。 1%～50% 🕮「変数アクセス時間設定」（P.5-48）を参照してください。	3%		

*1. EtherCAT 設定のプロセスデータ通信周期（プロセスデータ通信周期）は、この周期と同値となります。

● 定周期タスク（実行優先度 16、17、18）の設定項目

設定項目		設定範囲	デフォルト	反映タイミング	運転モード中書替
タスクタイプ		以下のいずれかを選択します。 「定周期タスク（優先度 16）」 「定周期タスク（優先度 17）」 「定周期タスク（優先度 18）」	−	設定転送時	不可
	実行優先度	「16」「17」「18」のいずれかが自動的に選択されます。	−		
タスク名		文字列	Periodic Task0		
周期／実行条件	タスク周期	🕮「5-3-1 NY シリーズのタスクの仕様」（P.5-8）参照	10ms		
タスク周期オーバー検知		プライマリ定周期タスクと同じです。	プライマリ定周期タスクと同じです。		
タスク実行タイムアウト時間					
変数アクセス時間［%］					

● イベントタスクの設定項目

設定項目	設定範囲	デフォルト	反映タイミング	運転モード中書替
タスクタイプ	以下のいずれかを選択します。 「イベントタスク（優先度 8）」 「イベントタスク（優先度 48）」	−	設定転送時	不可
タスク名	文字列	Event Task0		
実行条件	「命令による実行」「変数の条件式一致」のいずれかを選択します。	命令による実行		
タスク実行タイムアウト時間	無限ループなどによりタスクの実処理が完了しないときのタイムアウト検知時間を設定します。設定単位は 1ms です。 • 実行優先度「8」の場合：1 ～ 500ms • 実行優先度「48」の場合：1ms ～ 10s	• 実行優先度「8」の場合：200ms • 実行優先度「48」の場合：1s		

● Sysmac Studio での設定操作

Sysmac Studioから、[構成･設定]-[タスク設定]にて、タスクの追加と各タスクの設定を行います。

詳細は、🕮「タスク設定」（P.4-7）を参照してください。

5-6　変数の同時性確保

ここでは、タスク間で変数の値の同時性を確保する方法とコントローラ外部からの変数アクセスについて説明します。

5-6-1　タスク間での変数の値の同時性を確保する方法

異なる複数のタスクから、同じグローバル変数に読み書きする場合に、タスク間でグローバル変数の値の同時性を確保する方法には、以下の 2 つがあります。これらを総称して、「変数のタスク間排他制御機能」と呼びます。

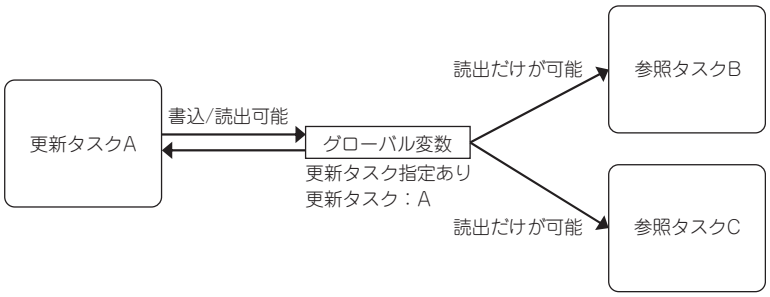
方法 1：1 つのタスクからグローバル変数に書込をし、ほかの複数タスクから読出をする場合
　　変数のタスク間排他制御設定機能を使用します。
方法 2：複数タスクからグローバル変数に書込をしたい場合
　　タスク間排他制御命令を使用します。

方法 1）変数のタスク間排他制御設定機能

● **概要**

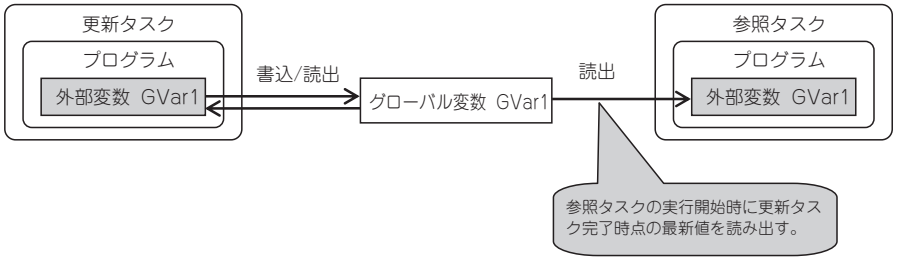
グローバル変数を更新するタスクと参照するタスクを指定することで、参照する側のタスクからみたグローバル変数の値の同時性を確保する機能です。

グローバル変数に読み書きを行うタスク（更新タスク）を 1 つ指定します。そのグローバル変数に対して、読出だけを行うタスク（参照タスク）を指定します。これにより、グローバル変数の値の同時性を確保します。



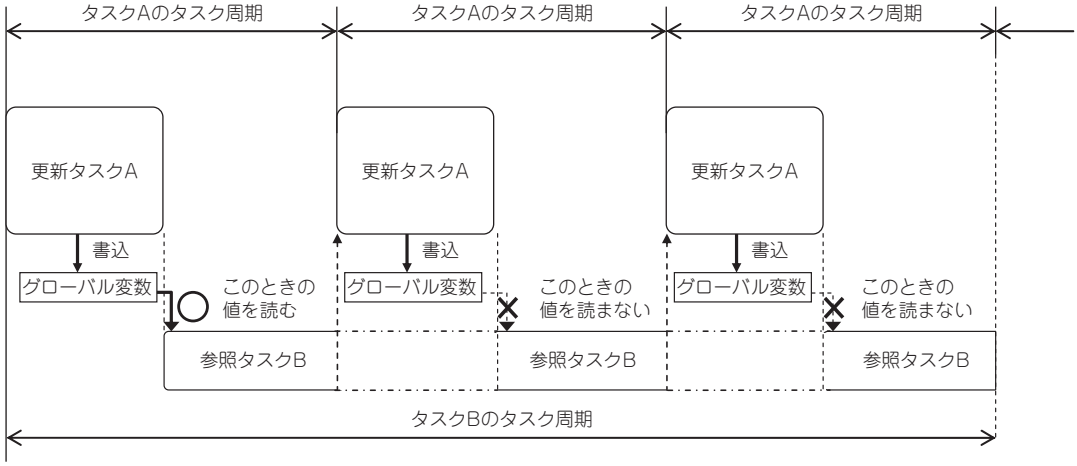
● **用途例**

プライマリ定周期タスクによって書込がされるグローバル変数があり、その変数の値を定周期タスク側で読み出すときに、その定周期タスク内で変数の値の同時性を確保するときに使用します。



● **しくみ**

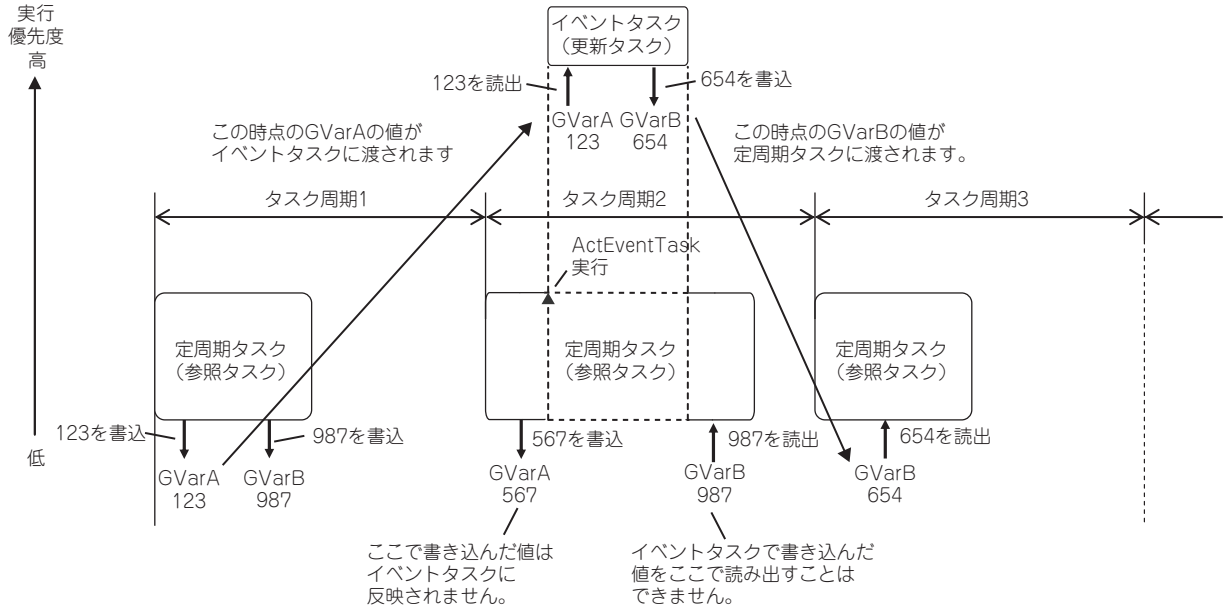
更新タスクが設定されているグローバル変数に対しては、参照タスクは、自身のタスク実行開始時に、更新タスクが実行完了し書き込んだ値を読み出します。



これにより、特別なユーザプログラムなしでタスク内でのグローバル変数の値の同時性を、確保することができます。
参照タスクからグローバル変数に値を書き込む命令を使用すると、Sysmac Studio のプログラムチェックでエラーになります。

使用上の注意

ActEventTask 命令を使用する場合、2 つのタスク間で、グローバル変数のいつの時点の値が読み出されるのか、いつの時点で値が更新されるのかには、十分に注意してください。
たとえば下図の場合、イベントタスクが読み出すグローバル変数 GVarA の値は、タスク周期 1 の終了時点のものです。したがって、タスク周期 2 で定周期タスクが GVarA の値を書き込んでも、それはイベントタスクには反映されません。
また、イベントタスクが書き込んだグローバル変数 GVarB の値が定周期タスクに渡されるのは、タスク周期 3 の開始時点です。したがって、タスク周期 2 で定周期タスクが GVarB の値を読み出しても、イベントタスクが書き込んだ値を読み出すことはできません。



このように、ActEventTask 命令を使用してイベントタスクを実行する場合、グローバル変数の値の受け渡しには、「変数のタスク間排他制御機能」は使用しないでください。
ActEventTask 命令使用時にグローバル変数の同時性を確保するためには、タスク実行中判定命令 Task_IsActive が有効です。Task_IsActive 命令は、指定したタスクが実行中または待機中かどうかを判定する命令です。この命令を使って、イベントタスク実行中には、イベントタスクが書込を行う変数を、別タスクから参照しないようにしてください。Task_IsActive 命令の詳細は、🕮『NY シリーズ コマンドリファレンスマニュアル 基本編（SBCA-437）』を参照してください。

● 制限

- グローバル変数1つに対して更新タスクは1つだけ指定します。同じグローバル変数に対して複数のタスクが書込可能としたいときは、後述のタスク間排他制御命令を使用してください。
- 構造体／共用体変数に更新タスクを設定する場合は、構造体／共用体変数全体で、1 つの更新タスクを設定します。構造体／共用体メンバごとに更新タスクを設定することはできません。
- 配列変数に更新タスクを設定する場合は、配列変数全体で、1 つの更新タスクを設定します。配列要素ごとに更新タスクを設定することはできません。

使用上の注意

値の同時性を確保したいグローバル変数には、参照タスクでも更新タスクでもないタスクから値を書き込まないでください。更新タスクを設定したグローバル変数に、更新タスクでも参照タスクでもないタスクのユーザプログラムから読み書きした場合は、このグローバル変数の値の同時性は保証できません。このようなプログラムを作成すると、プログラムチェック時に警告が出ます。

参考

- 変数のタスク間排他制御設定機能を使用しているグローバル変数は、そのグローバル変数を示す外部変数をデータトレースのサンプリング対象に設定できます。これにより、データトレースで、更新タスクと参照タスク、それぞれのグローバル変数の値をサンプリングすることができます。データトレースについては、🕮「8-4-4 データトレース」（P.8-35）を参照してください。

● Sysmac Studio での設定操作

Sysmac Studio から、[タスク設定]-[変数のタスク間排他制御設定] にて、更新タスク指定をするグローバル変数（「更新する変数」）、およびそれに対する参照タスク（「参照するタスク」）を設定します。

詳細は、🕮「変数のタスク間排他制御設定」（P.4-10）を参照してください。

方法 2）タスク間排他制御命令

複数タスクから同じグローバル変数に書込する場合、タスク間排他制御命令（Lock/Unlock 命令）を使用して、グローバル変数の同時性を確保します。

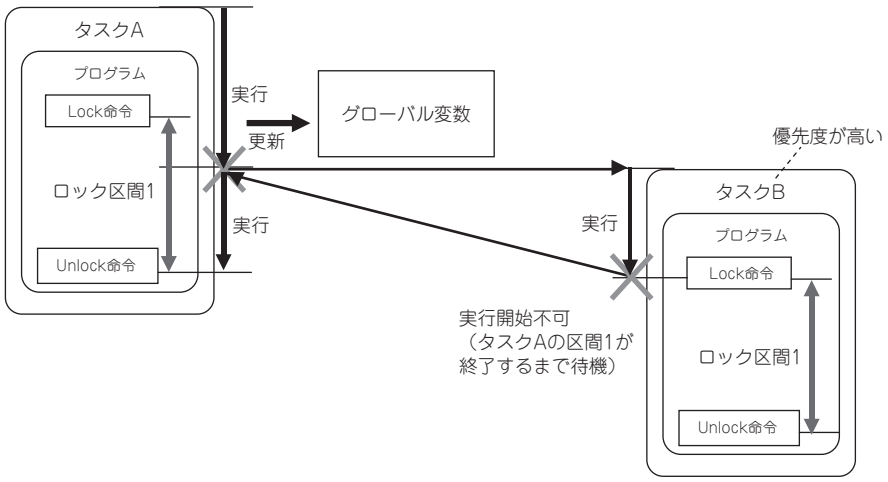
タスク間排他制御命令は、Lock 命令から Unlock 命令までの区間（ ロック区間）を実行中に、ほかのタスクの同じロック番号のロック区間を実行できなくします。グローバル変数への書込をロック区間にすることで、複数のタスクからグローバル変数の書込をする場合も、値の同時性を確保できます。

詳細は、🕮『NY シリーズ コマンドリファレンスマニュアル 基本編（SBCA-437）』の Lock（タスク間排他ロック）命令、Unlock（タスク間排他ロック解除）命令を参照してください。

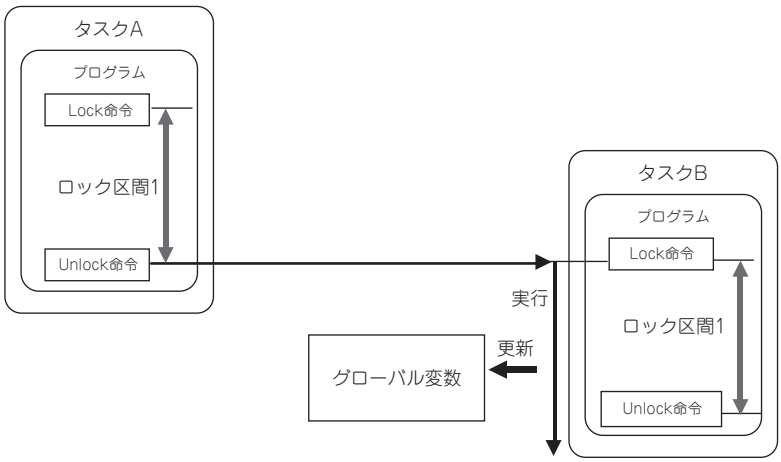
例）

タスク A、タスク B は、どちらもロック区間 1 を持つタスクで、タスク A よりタスク B の優先度が高いとします。

タスク A のロック区間 1 の実行中にタスク B の実行条件が成立すると、タスク A はロック区間 1 の途中で一時停止され、タスク B が実行されます。しかし、タスク A でロック区間 1 が終了されていないため、タスク B はロック区間 1 の処理の前に一時停止されます。タスク B が一時停止されると、タスク A のロック区間 1 が再開されます。



タスク A のロック区間 1 が終了すると、タスク A は再度一時停止され、タスク B のロック区間 1 以降が実行されます。グローバル変数の書込処理をタスク間で排他することで、値の同時性を確保します。



使用上の注意

- ロック区間は、必要最小限の長さにしてください。ロック区間が長いと、タスクの実行周期を超過する可能性があります。
- Lock 命令と Unlock 命令は、同一 POU の同一セクション内で、必ず対にしてお使いください。

タスク間で同じグローバル変数にアクセスする場合の例

ここでは、複数のグローバル変数を使用し、別のタスクに処理を依頼する場合について説明します。次のサンプルプログラムでは、ひとつのグローバル変数「gReq」を処理の排他フラグとして、ユーザプログラムで排他制御しています。このような場合でも、タスク間で使用する複数のグローバル変数「gPar1」、「gPar2」、「gReq」にアクセスするロジックに対して、「変数のタスク間排他制御機能」による排他制御が必要です。

次に、タスク間排他制御命令（Lock/Unlock 命令）を使用して排他制御するサンプルプログラムを示します。

● グローバル変数

名称	データ型	コメント
gReq	BOOL	要求フラグ
gPar1	ULINT	パラメータ 1
gPar2	DATA_AND_TIME	パラメータ 2

● 処理を依頼する側のタスク (MainTask)

• 内部変数

名称	データ型	コメント
ReqTrg	BOOL	要求トリガ
cCnt	ULINT	100ms カウンタ値
cTime	DATA_AND_TIME	現在時刻

• ST プログラム

```
cCnt:=Get100msCnt();                          (*100ms カウンタ値の取得 *)
cTime:=GetTime();                             (* 現在時刻の取得 *)

Lock(1);                                      (* タスク間排他ロック開始 *)
IF ReqTrg=TRUE AND gReq=FALSE THEN            (* 排他フラグの参照 *)
  gPar1:=cCnt;                                (*SubTask で処理するパラメータのセット *)
  gPar2:=cTime;                               (*SubTask で処理するパラメータのセット *)
  gReq:=TRUE;
  ReqTrg:=FALSE;
END_IF;
Unlock(1);                                    (* タスク間排他ロック解除 *)
```

● 処理を依頼される側のタスク (SubTask)

• 内部変数

名称	データ型	コメント
ReqBusy	BOOL	−
UserDefFB_ins	UserDefFB	処理を実行するユーザ定義 FB のインスタンス

• ST プログラム

```
Lock(1);                                      (* タスク間排他ロック開始 *)
IF gReq=TRUE AND ReqBusy=FALSE THEN           (* 排他フラグの参照 *)
  ReqBusy:=TRUE;
  UserDefFB_ins.PutData:=gPar1;               (*MainTask からのパラメータの読み出し *)
  UserDefFB_ins.PutDate:=gPar2;               (*MainTask からのパラメータの読み出し *)
  gReq:=FALSE;                                (* 排他フラグのリセット *)

  UserDefFB_ins.Execute:=TRUE;
END_IF;
Unlock(1);                                    (* タスク間排他ロック解除 *)

UserDefFB_ins();

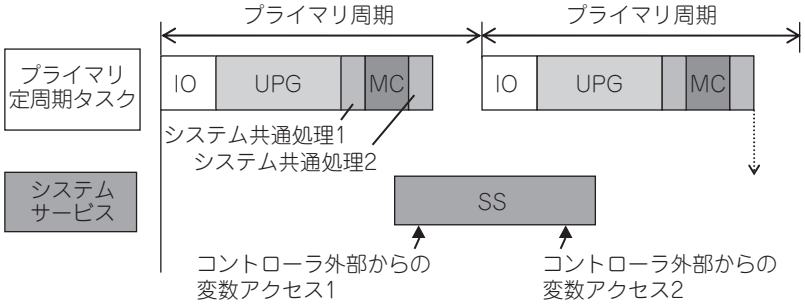
IF UserDefFB_ins.Done:=TRUE THEN
  UserDefFB_ins.Execute:=FALSE;
  ReqBusy:=FALSE;
END_IF;
```

5-6-2　コントローラ外部からの変数アクセス

コントローラ外部からの変数アクセスは、システムサービス内で実行されます。システムサービスは、タスク実行よりも低い優先度で実行します。したがって、コントローラ外部から複数の変数にアクセスする場合、各変数の値の更新が、同一タスク周期内に実行されない可能性があります。ユーザプログラム内に、値が更新されている変数と更新されていない変数が混在すると、コントローラが予期せぬ動作をする場合があります。
これを防ぐために、コントローラ外部からの変数アクセスを、タスクの「システム共通処理 2」で実行させることで、複数の変数の値の更新を、確実に同一タスク周期内に実行させることが可能です。

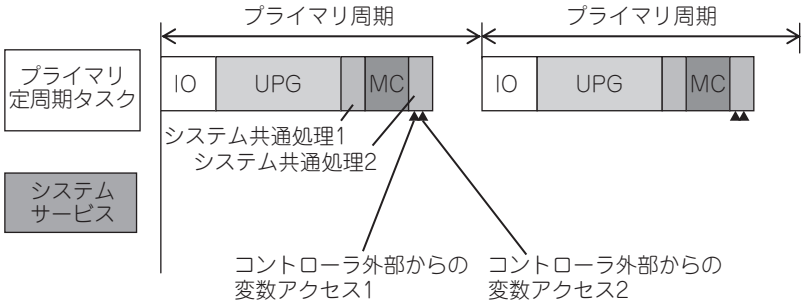
● コントローラ外部からの変数アクセスを、システムサービスで実行する場合

複数の変数アクセスが、同一タスク周期で実行されない可能性があります。



● コントローラ外部からの変数アクセスを、システム共通処理 2 で実行する場合

複数の変数アクセスが、同一タスク周期で実行されます。



ここでは、コントローラ外部からの変数アクセスを、タスクの「システム共通処理 2」で実行させる方法を説明します。

コントローラ外部からの変数アクセス方法

コントローラ外部からの変数アクセス方法には、以下の 4 種類があります。

- Sysmac Studio
- NS/NA シリーズ表示器
- EtherNet/IP のタグデータリンク
- 上位パソコンなどからの CIP 通信命令

ただし、変数アクセス方法が Sysmac Studio の場合、タスクの「システム共通処理 2」で実行させることができるのは、値の更新処理だけです。値の参照処理は、システムサービスで実行されます。

変数アクセスをシステム共通処理 2 で実行するタスク

コントローラ外部からの変数アクセスを、「システム共通処理 2」で実行するタスクは、変数の種類によって以下のように決まっています。

変数	値を更新するタスク
変数のタスク間排他制御設定で指定したグローバル変数	Sysmac Studio の [構成・設定]-[タスク設定]-[変数のタスク間排他制御設定] で指定した更新タスク
EtherCAT スレーブ用デバイス変数	Sysmac Studio の [構成・設定]-[タスク設定]-[I/O を制御するタスク設定] で指定したタスク

変数アクセスをシステム共通処理 2 で実行するための設定

コントローラ外部からの変数アクセスを、タスクの「システム共通処理 2」で実行するためには、Sysmac Studio で以下の 2 つの設定を行うことが必要です。

- 変数のタスク間排他制御設定（対象変数がグローバル変数の場合）
- 変数アクセス時間設定

変数のタスク間排他制御設定

コントローラ外部からの変数アクセスを、タスクの「システム共通処理 2」で実行する対象変数がグローバル変数の場合、変数のタスク間排他制御設定が必要です。変数のタスク間排他制御機能とは、対象グローバル変数を更新するタスクを指定することで、それ以外のタスクやコントローラ外部のアクセス元からは、対象グローバル変数の値を更新できなくする機能のことです。

変数のタスク間排他制御機能の詳細は、📖「変数のタスク間排他制御設定」（P.4-10）を参照してください。

使用上の注意

EtherNet/IP のタグデータリンクを使用する場合は、同じタグセット内のタグ（ネットワーク公開属性の変数）に対して、必ず同じタスクを更新タスクとしてください。そうしないと、タグセット内の複数のタグの更新タイミングが、異なるタスク周期にまたがってしまう可能性があります。

変数アクセス時間設定

コントローラ外部からの変数アクセスを、タスクの「システム共通処理 2」で実行すると、タスク実行時間が延びる可能性があります。ユーザは、Sysmac Studio で、変数へのアクセスにかかる処理時間の上限値を設定する必要があります。変数へのアクセスにかかる処理時間の上限値のことを、「変数アクセス時間」と呼びます。

● 変数へのアクセス時間の算出方法

変数へのアクセス時間は、次式で算出します。

変数へのアクセス時間 [μs]= 変数の合計サイズ [バイト]*a + 変数の数 *b + アクセス数 *c + d

上式中の定数 a ～ d の値は、NY シリーズコントローラでは以下となります。

コントローラの形式	定数の値 [μs]			
	a	b	c	d
形 NY5□2-□□□□	0.0006	0.100	1.40	6.60

● 変数アクセス時間の設定方法

変数アクセス時間は、Sysmac Studio の [構成・設定]-[タスク設定] で設定します。設定はタスクごとに行い、タスク周期に対する割合を入力します。デフォルト値は 3% です。設定方法の詳細は、🕮『Sysmac Studio Version 1 オペレーションマニュアル（SBCA-470)』を参照してください。

● 変数アクセス時間の設定例

変数アクセス時間の設定例を、以下に示します。
形 NY5□2-□□□□ で動作するタスクに対して、以下の 3 種類の、コントローラ外部からの変数アクセスがあるとします。

アクセス No.	変数アクセス元	アクセスする変数の合計サイズ [バイト]	アクセスする変数の数	アクセス数
1	EtherNet/IP のタグデータリンク	600	8	1
2	EtherNet/IP のタグデータリンク	200	4	1
3	CIP 通信命令	1,000	1	1

アクセス No.1 の、変数へのアクセス時間は、算出式より以下の値になります。
アクセス No.1 の変数へのアクセス時間 =600*0.0006+8*0.100+1*1.40+6.60
=9.16[μs]

同様に、各アクセス No. の変数へのアクセス時間は、以下の値になります。

アクセス No.	変数へのアクセス時間 [μs]
1	9.16
2	8.52
3	8.7

1 タスク周期中に、これらのアクセスの中のどれか 1 つしか発生しないのであれば、変数へのアクセス時間が最も長いアクセス No.1 に合わせて、変数アクセス時間を設定します。
アクセス No.1 の変数へのアクセス時間が 9.16μs ですから、タスク周期が 500μs だとすると、変数アクセス時間は 9.16/500 ≒ 2% と設定します。

1 タスク周期中に、これらのアクセスが 1 回ずつ発生するのであれば、変数へのアクセス時間は、算出式より以下の値になります。
変数へのアクセス時間 =(600+200+1000)*0.0006+(8+4+1)*0.100+(1+1+1)*1.40+6.60
=13.18[μs]

タスク周期が 500μs だとすると、変数アクセス時間は 13.18/500 ≒ 3% と設定します。

● 変数アクセス時間の設定値より実際のアクセス時間が長くなった場合の処理

変数アクセス時間の設定値より実際のアクセス時間が長くなった場合、1 タスク周期中の変数アクセス回数によって以下のように処理されます。
複数の変数アクセスが同一タスク周期に収まるように、変数アクセス時間は十分に余裕をもたせて設定してください。

1 タスク周期中の変数アクセス回数	処理
複数回	変数アクセス時間の設定値に収まるだけの変数アクセスが実行されます。実行できなかった変数アクセスは、つぎのタスク周期に実行されます。 したがって、複数の変数アクセスが同一タスク周期に収まりません。
1 回	変数アクセス時間の設定値を超えて、変数アクセスが実行されます。したがって、タスク実行時間が延びます。

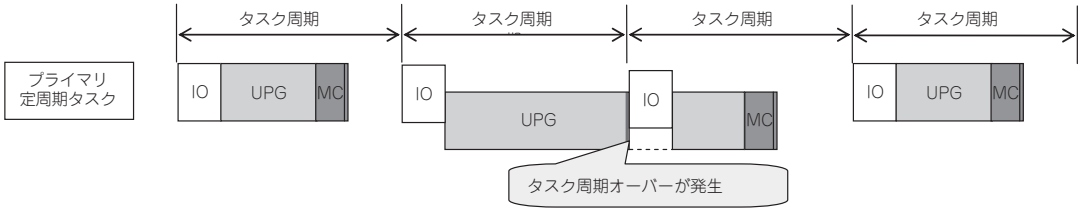
5-7　タスク関連の異常

ここでは、以下の異常について説明します。
- タスク周期オーバー
- モーション制御周期オーバー
- タスク実行タイムアウト
- I/O リフレッシュ周期オーバー

タスク周期オーバー

タスクの実行時間が設定したタスク周期を超えた場合に、「タスク周期オーバー」が発生します。
軽度フォールトレベルのコントローラ異常です。エラー発生時でも運転は継続します。
プライマリ定周期タスク、定周期タスクが対象です。

なお、設定により、タスク周期オーバーを発生させないことも可能です。Sysmac Studio の「タスク設定」－「タスク設定」の「タスク周期オーバー検知」にて設定します。デフォルトは「検知する」となっています。



異常名	異常レベル	処置
タスク周期オーバー	軽度フォールト	タスク設定およびプログラムを見直し、再ダウンロードします。 Sysmac Studio から異常解除してください。

「タスク周期オーバー検知」設定にかかわらず、周期内にタスク内の処理が完了しなかった場合は、システム定義変数の「_< タスク名称 >_Exceeded」（タスク周期超過フラグ）、「_< タスク名称 >_ExceedCount」（タスク周期超過回数）、「_ErrSta」（コントローラ異常ステータス）およびイベントログに情報が出力されます。
周期内にタスクの処理が完了しない場合の I/O リフレッシュ動作は、入出力の対象によって以下のように異なります。

入出力の対象	周期内にタスクの処理が完了しない場合の I/O リフレッシュ動作
EtherCAT スレーブ*1	出力：前回の周期の値を出力します。 入力：I/O リフレッシュは行いますが、入力データは実行中のユーザプログラムに反映されません。

*1. EtherCAT スレーブターミナル内の NX ユニットを含みます。

使用上の注意

「タスク周期オーバー」が発生した場合、タスク周期内に収まるようにプログラムを作成するか、またはタスク周期を変更してください。

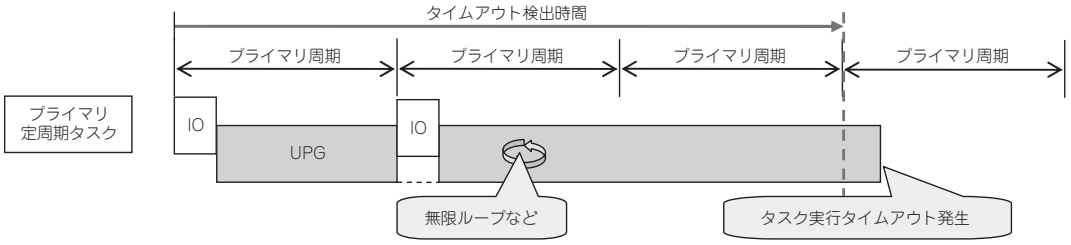
モーション制御周期オーバー

「モーション制御（MC）」の処理がプライマリ周期（＝モーション制御周期）連続で完了しなかった場合に発生します。モーション制御機能モジュールの部分停止フォールトレベルのコントローラ異常となります。このとき、「タスク周期オーバー」も同時に発生します。

異常名	異常レベル	処置
モーション制御周期オーバー	部分停止フォールト	プログラムの処理内容を少なくするか、動作に問題が発生しない範囲で制御周期をより大きな値に設定してください。

タスク実行タイムアウト

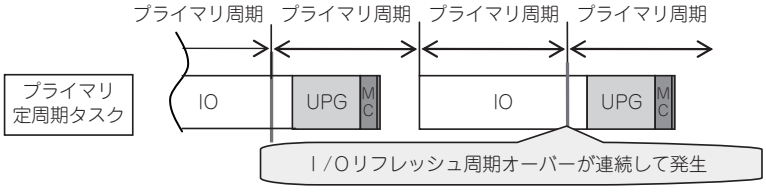
指定した時間（「タイムアウト検出時間」）を経過してもタスクの実処理が完了しない場合に、「タスク実行タイムアウト」が発生します。
全停止フォールトレベルのコントローラ異常です。エラー発生時に、ユーザプログラムの実行を停止します。
ユーザプログラム内の無限ループなど、ロジック誤りによってタスクが正常動作できない場合にも発生します。



異常名	異常レベル	処置
タスク実行タイムアウト	全停止フォールト	タスク設定を見直し、ユーザプログラムを再ダウンロードします。 電源再投入またはリセット操作が必要です。

I/O リフレッシュ周期オーバー

プライマリ定周期タスクで、タスク周期内に I/O リフレッシュが完了しない状態が連続すると、「I/O リフレッシュ周期オーバー」が発生します。
全停止フォールトレベルのコントローラ異常です。エラー発生時に、ユーザプログラムの実行を停止します。



異常名	異常レベル	処置
I/O リフレッシュ周期オーバー	全停止フォールト	タスク設定を見直し、再ダウンロードします。 電源再投入またはリセット操作が必要です。

5-8　タスク実行状態、タスク実行時間のモニタ

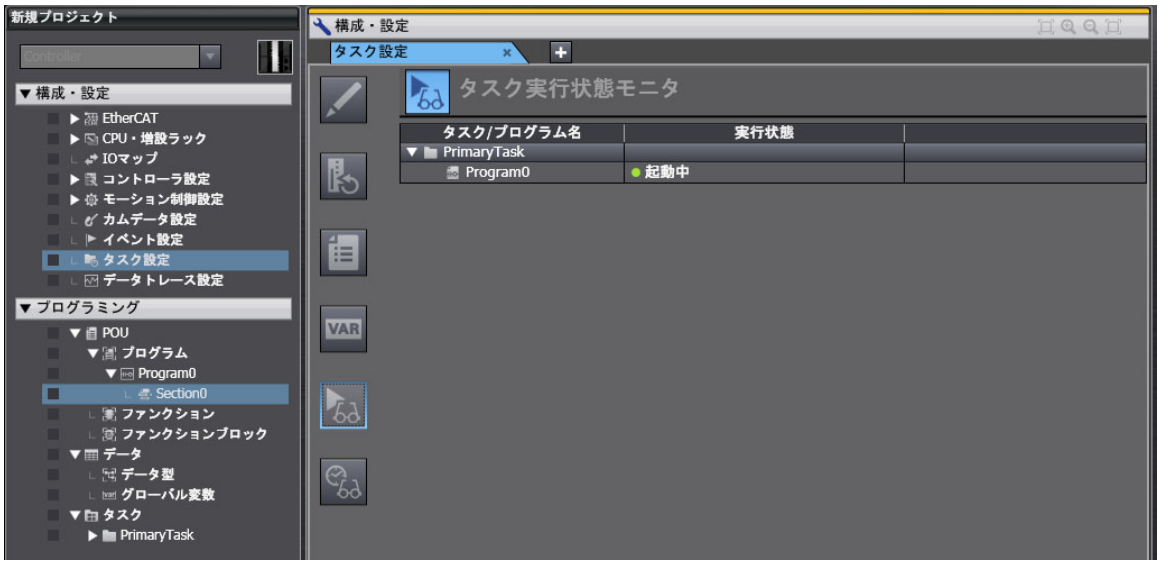
Sysmac Studio からのオンライン操作で、タスクの実行状態、タスクの実行時間をモニタすることができます。

タスクの実行状態のモニタ

各タスク内のプログラムの実行状態（起動中／停止中）を、Sysmac Studio からモニタすることができます。

● Sysmac Studio での操作

Sysmac Studio から、コントローラにオンライン接続して、[構成・設定]-[タスク設定] を選択します。タスク実行状態モニタボタンをクリックすると、以下の画面が表示されます。

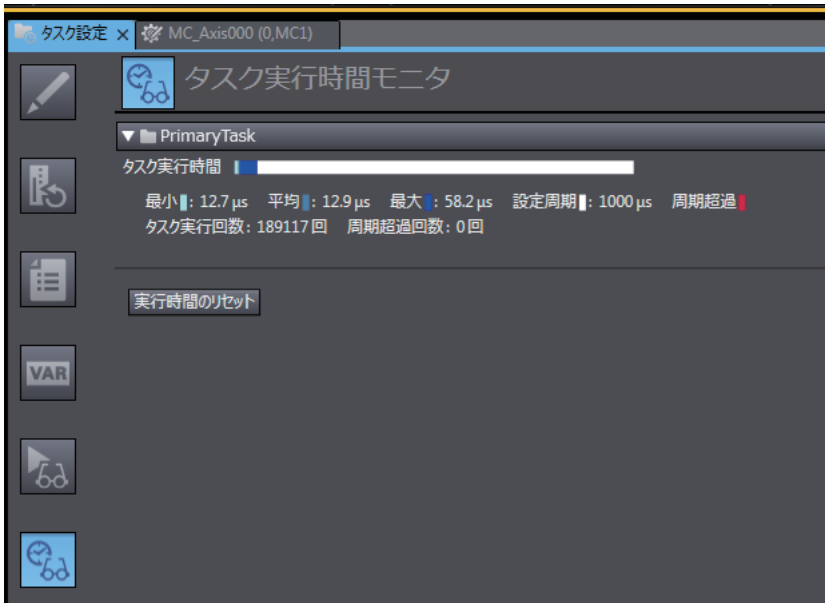


タスクの実行時間のモニタ

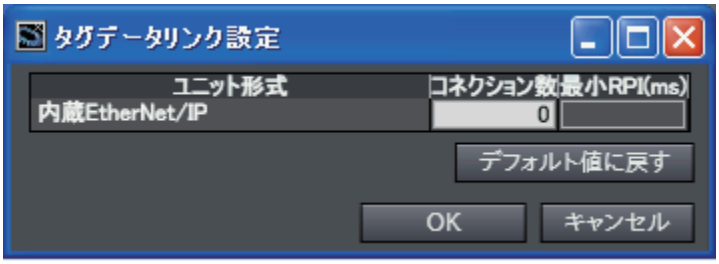
各タスクの実行時間を Sysmac Studio からモニタすることができます。

● Sysmac Studio でモニタできる値

コントローラ接続時



シミュレータ接続の内蔵 EtherNet/IP 画面



設定項目は以下のとおりです。

使用するポート	設定項目	機能説明	設定値	デフォルト
NY シリーズ内蔵 EtherNet/IP ポート	コネクション数	タグデータリンクのコネクション数です。	0 ～ 128	0
	最小 RPI	タグデータリンクのコネクションのうち、最小のパケットインターバル (RPI) の設定値です。	1ms ～ 10s（最小単位 1ms）	−

モニタできる項目は以下のとおりです。

モニタ項目		説明	コントローラ接続時
タスク実行時間 *1	最小	タスク実行時間の最小値です。	あり
	平均	タスク実行時間の平均値です。	
	最大	タスク実行時間の最大値です。	
設定周期		設定したタスク周期です。*2	
周期超過		タスク実行時間がタスク周期を超過したとき（システム定義変数の「タスク周期超過フラグ」が TRUE のとき）、その程度を上のバーで示します。*2	
タスク実行回数		タスクの実行回数を示します。 システム定義変数の「タスク実行回数」の値です。	

*1. タスクの実行開始から実行完了までに、実際にかかった時間のことです。実行開始から実行完了までの間に実行された、別タスクやシステムサービスの時間も含めます。
*2. イベントタスクの場合は、表示されません。

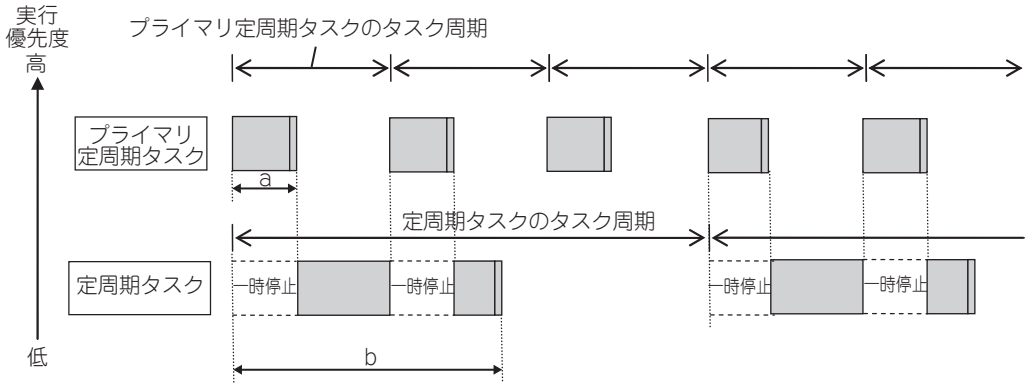
使用上の注意

必ず、実機コントローラ接続にて確認の上、設計の検討および本運転を行うようにしてください。

タスク実行時間とタスク実処理時間の意味

タスクの実行時間のモニタで表示される「タスク実行時間」、「タスク実処理時間」の意味は、つぎのとおりです。

表示される時間	意味
タスク実行時間	タスク周期の開始から、タスク処理が終了するまでの時間です。 ただし、定周期タスクが、それより実行優先度の高いタスクの実行中に一時停止している時間も含みます。



a　:プライマリ定周期タスクのタスク実行時間
b　:定周期タスクのタスク実行時間

5-9　タスク設計方法および入出力応答時間

ここでは、タスク設計の指針、タスク実行時間の見積もり方法、タスク設計の例、および入出力応答時間について説明しています。

NY シリーズコントローラでは、プライマリ定周期タスクや定周期タスクは、設定された周期にもとづいて動作します。
実際の実行時間が設定された周期を超えた場合、各種エラー処理を行います。

ここでは、プライマリ定周期タスク 1 つ（シングルタスク）で構成する場合を例として、見積もり方や評価方法を説明します。

使用上の注意

実際のコントローラは、ユーザプログラムに記載されたロジックの動作有無、通信コマンドの発生やデータリンクの動作有無、データトレースなど各種機能の使用有無などによって、タスクの実行時間が変動します。
最終的には、必ず実機を使用してあらゆる条件で性能確認を行い、タスク周期オーバーを起こさないことや適切な通信応答性が確保されていることを確認してください。

5-9-1　タスク実行時間の確認

ここで見積もったタスク実行時間の平均値や最大値が、設定されたタスク周期に十分に納まるように設計してください。

● 机上での計算の場合

「A-2 NY シリーズでのタスク実処理時間の目安の計算」（P.A-12）を参照して、机上にて、「タスク実行時間の平均値」の概算見積もりを行います。
なお、机上の場合は、最大値を見積もることはできません。

● 実機コントローラでの計算の場合

実機コントローラ接続時、Sysmac Studio の「タスク実行時間モニタ機能」では以下の値を確認することができます。

- タスク実行時間（最小値、平均値、最大値）
- 設定周期
- タスク実行回数
- 周期超過回数

Sysmac Studio で表示する最大値は、実機コントローラで動作した結果としての最大値です。
前述のとおり、タスク実行時間は、実機コントローラの内部状態により変動します。

参考

「タスク実行時間モニタ機能」で表示されるタスク実行時間の平均値は、10 回分のタスク実行時間の移動平均値です。

5-9-2　タスク設計の例

本項では、プライマリ定周期タスク 1 つ（シングルタスク）で構成する場合を例とした設計手順を説明します。
ただし、実際のアプリケーションや各種条件によって、設計手順や考慮しなければならない要素が変化しますので、1 つの参考としてご理解ください。

1 **装置仕様から必要なシステムとしての各入出力応答時間を求めます。**

2 **システムとしての各入出力応答時間から、プライマリ定周期タスクの「タスク周期」を決定します。**

3 **「タスク実行時間」が決定した「タスク周期」内に納まるかをまず確認します。**
机上で、タスク実行時間の予想平均値の見積もりを行います。

4 **実機コントローラにて、タスク実行時間が、タスク周期内に納まるかを検討します。**
Sysmac Studio を実機コントローラにオンライン接続し、「タスク実行時間モニタ機能」により、タスク実行時間を検証します。

● **プライマリ定周期タスク１つ（シングルタスク）だけで、設定したタスク周期に納まらない場合、次の観点で定周期タスクへのタスク分割を検討します。**

- タスク実行時間を短縮するために、プログラム実行指令（PrgStart）命令、およびプログラム停止指令（PrgStop）命令を使用し、プライマリ定周期タスクに割り付けたプログラムを必要なときにだけ実行します。
- これらを実施しても、プライマリ定周期タスクのタスク周期に納まらない場合、プライマリ定周期タスクの処理のうち、高速、高精度が必要でない処理部を定周期タスク（実行優先度 16）に分割します。
  ただし、プライマリ定周期タスクと定周期タスク（実行優先度 16）間で、軸変数などを参照する場合、プライマリ定周期タスクのタスク実行時間が長くなる場合があります。詳細については、🕮『NY シリーズ 産業用パネル型 PC ／産業用ボックス型 PC ユーザーズマニュアル モーション制御編（SBCE-379）』を参照してください。
- これらを実行しても、プライマリ定周期タスクのタスク周期に納まらない場合は、プライマリ定周期タスクのタスク周期を見直してください。

5-9-3　システムとしての入出力応答時間

システムとして入力してから出力するまでの時間は、以下のとおりです。
ケースごとに異なります。
なお、実際には、この時間にスレーブ／ユニットの外部との入力応答時間、外部との出力応答時間が加算されます。

基本 I/O ユニットをシーケンス制御する場合

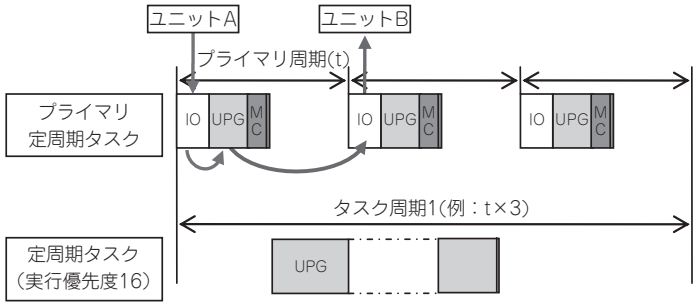
基本 I/O ユニットは、I/O リフレッシュを割り付けたタスクで、外部と I/O リフレッシュされます。
EtherCAT 通信を含む入出力応答時間は、以下のとおりです。

● プライマリ定周期タスクのユーザプログラムで制御する場合

以下の入出力応答時間で応答します。

最小入出力応答時間＝「プライマリ周期」

例）ユニット A、ユニット B をプライマリ定周期タスクで制御する場合



注. 上図は1回の入出力だけを記述しています。

ただし、ユニットからの入力タイミングによっては、以下の入出力応答時間となります。

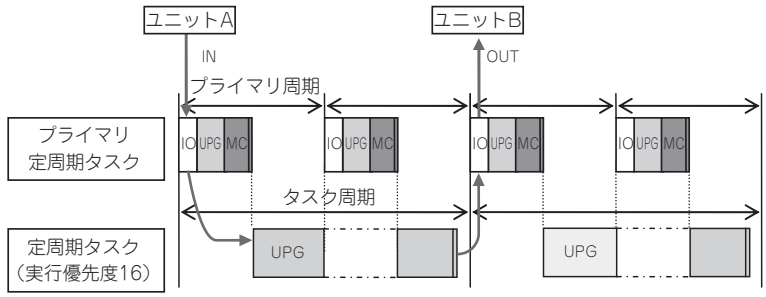
最大入出力応答時間＝「プライマリ周期」×2

● 定周期タスク（実行優先度 16）のユーザプログラムで制御する場合

以下の入出力応答時間で応答します。

最小入出力応答時間＝「定周期タスク（実行優先度 16）の周期」

例）ユニット A、ユニット B を定周期タスク（実行優先度 16）で制御する場合



注. 上図は1回の入出力だけを記述しています。

ただし、ユニットからの入力タイミングによっては、以下の入出力応答時間となります。

最大入出力応答時間＝「定周期タスク（実行優先度 16）の周期」×2

EtherCAT スレーブをシーケンス制御する場合

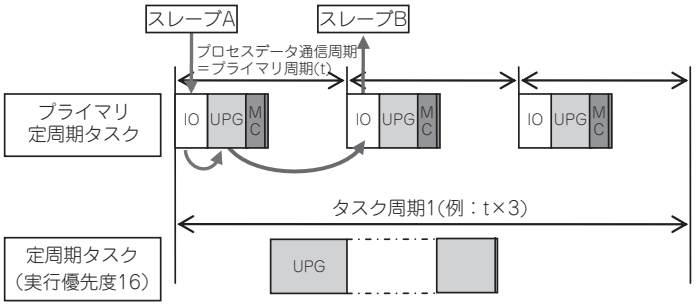
EtherCAT スレーブは、プライマリ定周期タスクの I/O リフレッシュにて、外部と EtherCAT 通信されます。
EtherCAT 通信を含む入出力応答時間は、以下のとおりです。

● プライマリ定周期タスクのユーザプログラムで制御する場合

以下の入出力応答時間で応答します。

最小入出力応答時間＝「プライマリ周期（＝プロセスデータ通信周期）」

例)EtherCAT INスレーブA、EtherCAT OUTスレーブBをプライマリ定周期タスクで制御する場合



注. 上図は1回の入出力だけを記述しています。

ただし、スレーブからの入力タイミングによっては、以下の入出力応答時間となります。

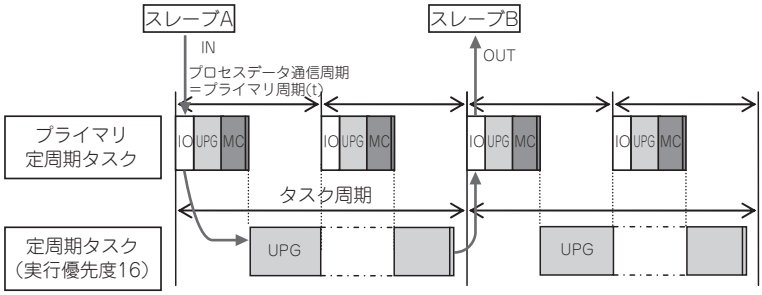
最大入出力応答時間＝「プライマリ周期（＝プロセスデータ通信周期）」×2

● 定周期タスク（実行優先度 16）のユーザプログラムで制御する場合

以下の入出力応答時間で応答します。

入出力応答時間＝「定周期タスク（実行優先度 16）の周期」

例）　EtherCAT IN スレーブ A、EtherCAT OUT スレーブ B を定周期タスク（実行優先度 16）で制御する場合



注. 上図は1回の入出力だけを記述しています。

ただし、スレーブからの入力タイミングによっては、以下の入出力応答時間となります。

最大入出力応答時間＝「定周期タスク（実行優先度 16）の周期」×2

モーション制御命令でモーション制御する場合

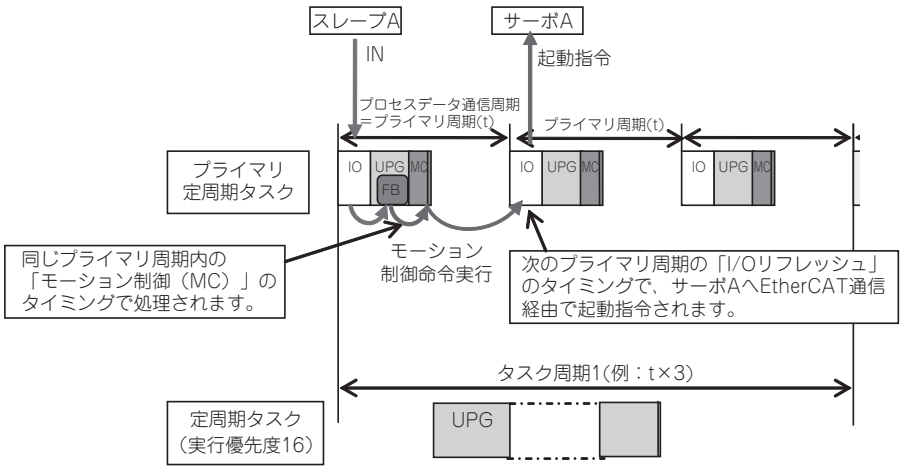
モーション制御命令は、[軸] に割り付けているサーボ／エンコーダ入力スレーブに対してアクセスします。
NY シリーズコントローラのモーション制御命令は、プライマリ定周期タスクまたは定周期タスク（実行優先度 16）に記述できます。
いずれの場合も、モーション制御命令は、プライマリ定周期タスク内の「モーション制御（MC）」にて実行されます。

EtherCAT 通信を含む入出力応答時間は、以下のとおりです。

● モーション制御命令をプライマリ定周期タスクに記述した場合

モーション制御命令の処理は、そのプライマリ定周期タスク内の直後の「モーション制御（MC）」にて実行されます。その処理結果は、その次のプライマリ定周期タスクの「I/O リフレッシュ」のタイミングで、[軸] に割り付けているサーボドライバへ、EtherCAT 通信経由で出力されます。
以下の入出力応答時間で応答します。

入出力応答時間＝「プライマリ周期（＝プロセスデータ通信周期）」



注. 上図は1回の入出力だけを記述しています。

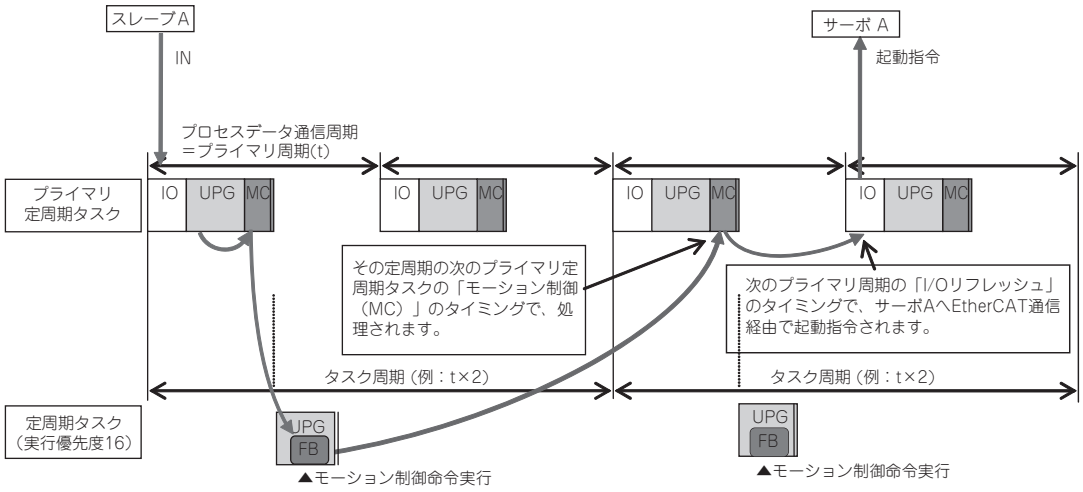
ただし、スレーブからの入力タイミングによっては、以下の入出力応答時間となります。

最大入出力応答時間＝「プライマリ周期（＝プロセスデータ通信周期）」×2

● モーション制御命令を定周期タスク（実行優先度 16）に記述した場合

モーション制御命令の処理は、その定周期タスク（実行優先度 16）の次のプライマリ定周期タスク内の「モーション制御（MC）」にて実行されます。その処理結果は、その次のプライマリ定周期タスクの「I/O リフレッシュ」のタイミングで、[軸] に割り付けているサーボドライバへ、EtherCAT 通信経由で出力されます。
そのため、モーション制御命令の実行タイミングにかかわらず、以下の入出力応答時間で応答します。

最小入出力応答時間＝「定周期タスク（実行優先度 16）の周期」＋ 「プライマリ周期（＝プロセスデータ通信周期）」



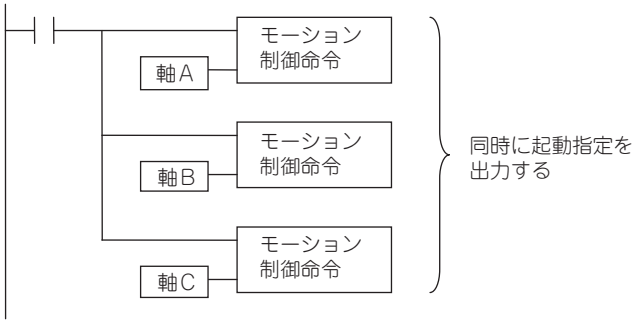
注. 上図は1回の入出力だけを記述しています。

ただし、スレーブからの入力タイミングによっては、以下の応答時間となります。

最大入出力応答時間＝「定周期タスク（実行優先度 16）の周期」＋ 「プライマリ周期（＝プロセスデータ通信周期）」×2

● 複数軸の同時起動

同じタスク内のユーザプログラムで、同じタスク周期に複数軸を起動した場合、同時に起動することができます。



参考

軸を制御するタスク以外のタスクで、軸変数の値を参照することができます。
使い方の詳細や注意事項については、🕮『NY シリーズ 産業用パネル型 PC ／産業用ボックス型 PC ユーザーズマニュアル モーション制御編（SBCE-379）』を参照してください。

6

プログラミング

本章では、プログラムの種類やプログラムを構成する変数や命令などプログラムの詳細について説明します。

6-1 プログラミング手順の概要 6-3

6-2 POU（プログラム構成単位） 6-5
6-2-1 POU とは 6-5
6-2-2 POU の 3 つのタイプの概要 6-6
6-2-3 プログラム、ファンクションブロック、ファンクションの相違点 6-7
6-2-4 プログラムの詳細 6-8
6-2-5 ファンクションブロック（FB）の詳細 6-9
6-2-6 ファンクション（FUN）の詳細 6-17
6-2-7 ファンクションブロック（FB）、ファンクション（FUN）共通動作 6-23
6-2-8 POU の制限事項 6-25

6-3 変数 6-27
6-3-1 変数とは 6-27
6-3-2 変数の種類 6-27
6-3-3 ユーザ定義変数の POU からみた種類 6-28
6-3-4 変数の属性 6-29
6-3-5 データ型 6-30
6-3-6 派生データ型 6-40
6-3-7 データ型の配列／範囲指定 6-50
6-3-8 変数の各属性の説明 6-58
6-3-9 状態変化に応じた変数の値の変化 6-64
6-3-10 ファンクションブロック（FB）のインスタンス 6-76
6-3-11 変数の値のモニタ 6-76
6-3-12 変数名などのプログラム関連の名称の制限 6-77

6-4 定数（リテラル） 6-79
6-4-1 定数とは 6-79
6-4-2 各データ型の記述形式 6-79

6-5 プログラム言語 6-84
6-5-1 プログラム言語とは 6-84
6-5-2 ラダー図言語（LD） 6-84
6-5-3 ST 言語 6-90

6-6 命令 6-128
6-6-1 命令とは 6-128
6-6-2 命令の基本知識 6-128
6-6-3 命令エラー 6-131

6-7　名前空間 6-137
6-7-1　名前空間とは 6-137
6-7-2　名前空間の仕様 6-138
6-7-3　名前空間の使用手順 6-141

6-8　ライブラリ 6-142
6-8-1　ライブラリの概要 6-142
6-8-2　ライブラリの仕様 6-143
6-8-3　ライブラリ部品の仕様 6-144
6-8-4　ライブラリの使用手順 6-145

6-9　ユーザプログラム作成上の留意事項 6-146
6-9-1　入力変数／出力変数／入出力変数が、配列指定の変数の場合 6-146
6-9-2　入力変数／出力変数／入出力変数が、構造体型変数の場合 6-146
6-9-3　マスタコントロール 6-147

6-1　プログラミング手順の概要

ここでは、プログラミング手順の概要を示します。
全体の手順の中で、以下の網掛けで示した位置に対応しています。

STEP 1. ソフト設計

- STEP1-1 I/O と処理の設計
- STEP1-2 タスク設計
- **STEP1-3 プログラムの設計**

↓

STEP 2. ソフトウェア設定とプログラミング

- STEP2-1 スレーブ／ユニット構成
- STEP2-2 コントローラ設定
- **STEP2-3 プログラミング**
- STEP2-4 オフラインデバッグ

↓

STEP 3. 装着とハードウェアの設定

↓

STEP 4. 配線

↓

STEP 5. 実機による動作確認と運転

詳細は、📖「1-4 NY シリーズの使用手順の全体」（P.1-10）を参照してください。

POU（処理の分割単位）の設計	参照
● **それぞれの処理を、どの POU で実行させるかを検討して、設計** 注：ファンクションには、ファンクションブロック型命令、ファンクションブロックを記述できません	🕮「6-2 POU（プログラム構成単位）」（P.6-5）
● **どの処理を、ラダー図、ST、インライン ST のどの言語で作成するかを設計** 注：「インライン ST」とは、ラダー図内に ST 言語を記述するラダー図言語の要素	🕮「6-5 プログラム言語」（P.6-84）



変数の設計	参照
● **どのようなユーザ定義変数を作成するかを設計**	🕮「6-3-1 変数とは」（P.6-27） 🕮「6-3-2 変数の種類」（P.6-27）
● **POU 共通で使用する変数（グローバル変数）と、各 POU だけで使用する変数（ローカル変数）を仕分け**	🕮「6-3-3 ユーザ定義変数の POU からみた種類」（P.6-28）
● **スレーブ／ユニットにアクセスするためのデバイス変数の変数名を、自動生成するか、ユーザ定義するかを設計**	🕮「3-3 I/O ポートとデバイス変数」（P.3-5）
● **変数の、以下の各属性を設計** 変数名、データ型、AT（割付先）指定、初期値、保持、コンスタント、ネットワーク公開など 変数の、データ型（配列指定、範囲指定、構造体型、列挙型）を設計	🕮「6-3-4 変数の属性」（P.6-29） 🕮「6-3-5 データ型」（P.6-30） 🕮「6-3-6 派生データ型」（P.6-40）
● **変数の設計時には、以下に注意して設計してください。** • 保持の設計： 電源投入時、動作モード変更時などに、変数をどの値とするかに応じて、保持属性を設計します。 • 構造体型の設計： 使用する命令の変数にて構造体データ型が使用されている場合、入力、出力または入出力パラメータ用に、その構造体データ型を使用するように設計します。 例）通信命令 • 配列指定の設計： 使用する命令の変数にて配列指定の変数が使用されている場合、入力、出力または入出力パラメータ用に配列指定の変数を設計します。 例）シフト命令、スタック / テーブル命令 • AT 指定の設計： • 一部の命令の入力パラメータ用の変数に AT 指定。 例）自由割付の I/O 割付 • ネットワーク公開の設計： EtherNet/IP タグデータリンクをする変数を設計します。	🕮「6-3-4 変数の属性」（P.6-29） 🕮「6-3-5 データ型」（P.6-30） 🕮「6-3-6 派生データ型」（P.6-40）

6-2　POU（プログラム構成単位）

NY シリーズコントローラで動作するユーザプログラムは、すべて POU（プログラム構成単位）の形式で作成し、それらを組み合わせて使用します。
ここでは、POU の構成、仕様などについて、詳しく説明します。
Sysmac Studio での POU の作成方法については、🕮『Sysmac Studio Version 1 オペレーションマニュアル（SBCA-470）』を参照してください。

6-2-1　POU とは

POU（Program Organization Unit）とは、IEC 61131-3 におけるユーザプログラムの実行モデルの単位です。ユーザプログラムを構成する基本的な要素として、ローカル変数テーブルとアルゴリズム（一連のコード / ロジック）を記述します。
複数の POU を組み合わせることにより、ユーザプログラム全体を構成します。

POU の構成要素には、以下の 3 種類があります。

- プログラム
  メインルーチンに相当する、アルゴリズムを記載する主となる要素です。
  アルゴリズムに、すべての命令、ファンクション、ファンクションブロックを記述することができます。
- ファンクションブロック（以降「FB」と略します）
  入力の値が同じでも、異なる値の出力をすることのできる要素です。プログラムまたは別の FB から呼び出されることによって起動します。
- ファンクション（以降「FUN」と略します）
  入力の値が同じであれば、常に均一の値の出力をすることのできる要素です。プログラム、FB、または FUN から呼び出されることによって起動します。

POU では上記の 3 つの要素を組み合わせて構成します。また、POU は複数作成できます。
作成したユーザプログラムはタスクに割り付けて起動させます。

6-2-2 POU の 3 つのタイプの概要

プログラムとは

● 起動方法／実行条件

- タスクが起動することにより、そのタスクに割り付けられているプログラムを起動します。
- 常時実行します。

● 記述

- POU には必ず 1 つ以上のプログラムを記述する必要があります。1 タスクあたりのプログラムの数は、最大 128 個です。

ファンクションブロック（FB）とは

● 起動方法／実行条件

- プログラム、FB からの呼び出しに応じて、起動します。
- 常時実行します。
- 条件が成立したときだけ実行させたい場合は、実行条件を意味する入力変数を定義してください。

● 記述

- アルゴリズムに、すべての命令、ユーザ定義 FUN、およびユーザ定義 FB を記述することができます。
- 内部変数の値を保持することができます。したがって、タイマやカウンタのように状態を持つことができます。
- ユーザ定義のものとシステム定義のものがあります。
  特に、ユーザ定義の FB は、「ユーザ定義 FB」と呼びます。システム定義の FB は、「FB 型命令」と呼びます。

FB の詳細は、📖「6-2-5 ファンクションブロック（FB）の詳細」（P.6-9）を参照してください。

ファンクション（FUN）とは

● 起動方法／実行条件

- プログラム、FB、または別の FUN からの呼び出しに応じて、起動します。
- 実行条件は入力変数（EN）で指定します。EN が TRUE のとき、一度実行します。

● 記述

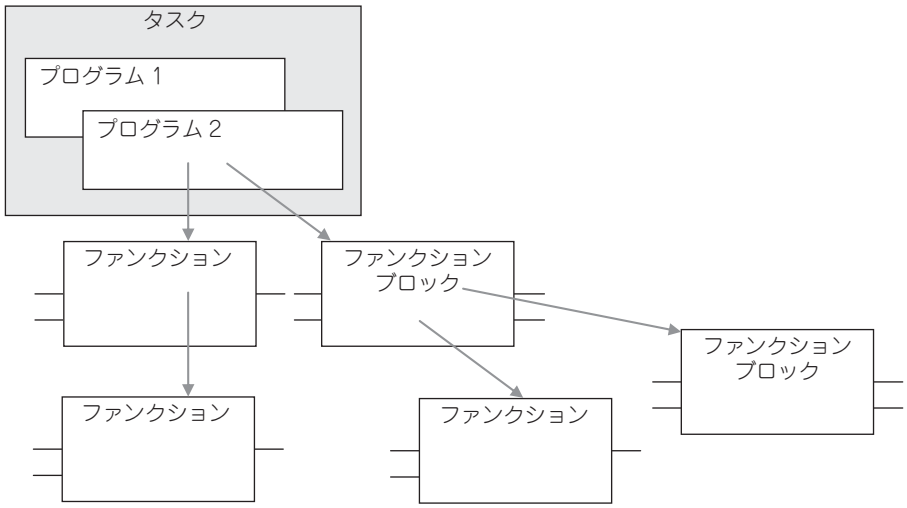
- アルゴリズムに、FB 型命令およびユーザ定義 FB を記述することはできません。
- 内部変数の値を保持することはできません。したがって、入力値が同じ場合、出力値は常に一定になります。
- ユーザ定義のものとシステム定義のものがあります。
  特に、ユーザ定義の FUN は、「ユーザ定義 FUN」と呼びます。システム定義の FUN は、「FUN 型命令」と呼びます。

FUN の詳細は、📖「6-2-6 ファンクション（FUN）の詳細」（P.6-17）を参照してください。

6-2-3　プログラム、ファンクションブロック、ファンクションの相違点

項目 ＼ POU 種類		プログラム	ファンクションブロック（FB）	ファンクション（FUN）
起動方法		割り付けられたタスクの起動により起動	プログラム、別の FB からの呼び出し	プログラム、FB、FUN からの呼び出し
アルゴリズム内	すべての命令	可	可	不可
	ユーザ定義 FUN	可	可	可
	ユーザ定義 FB	可	可	不可
実行条件		常時実行	常時実行 実行条件は入力変数によって指定	EN 入力で指定

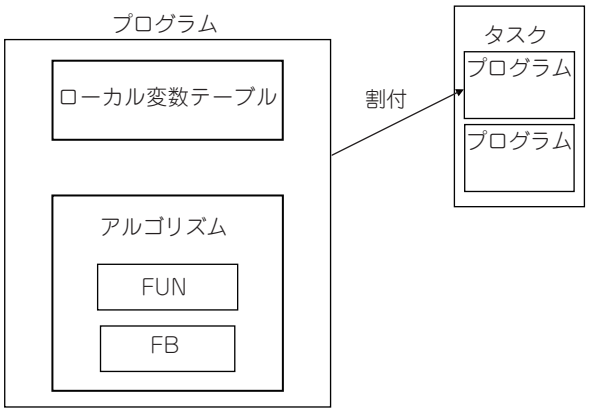
プログラム、FB、FUN の階層関係を、下図に示します。



6-2-4 プログラムの詳細

プログラムの構成

プログラム は、ローカル変数テーブルとアルゴリズム部とから成ります。
アルゴリズム部の中では、FUN か FB を記述することができます。



プログラムが、ほかの POU から呼び出されることはありません。

プログラムの起動条件

プログラムは、それが割り付けられているタスクが起動することによって、起動します。

● 実行順

タスクに割り付けられたプログラムは、タスクの中での実行順番を設定することができます。
Sysmac Studio の [タスク設定]-[プログラムの割付設定] で指定します。

● 関連システム定義変数

プログラムは、それぞれに以下のローカル変数のシステム定義変数を持っています。

変数名	名称	機能	データ型	R/W
P_First_Run Mode	運転開始時 1 周期 ON フラグ	コントローラの動作モードを、プログラムモードから運転モードに変更した際に、当該プログラムが実行中である場合、1 タスク周期だけ TRUE となります。 当該プログラムが実行中でなければ、FALSE のままです。 コントローラが運転を開始したときに初期処理を行う場合などに使用します。	BOOL	R
P_First_Run	プログラム起動時 1 周期 ON フラグ	当該プログラムが、実行を開始した際に、1 タスク周期だけ TRUE となります。*1 プログラムが実行を開始するごとに初期処理を行う場合などに使用します。	BOOL	R
P_PRGER	命令エラー発生フラグ	当該プログラムまたは当該プログラムから呼び出した FB や FUN で、命令エラーが発生したときに TRUE となり続けるフラグです。一度 TRUE となると、ユーザプログラムによって FALSE にされるまで TRUE となり続けます。	BOOL	RW
P_CY	キャリーフラグ	一部の命令にて更新するフラグです。	BOOL	R

*1. プログラムの実行／停止指令には、PrgStart 命令／ PrgStop 命令を使用します。PrgStart 命令は、P_First_Run を TRUE にすることなくプログラムを実行する設定が可能です。

6-2-5　ファンクションブロック（FB）の詳細

ファンクションブロック（FB）の作成手順

FB は、あらかじめ作成した「FB 定義」と、それを実際にプログラム上で使用するための「インスタンス」からなります。
FB は、以下の順序で作成していきます。

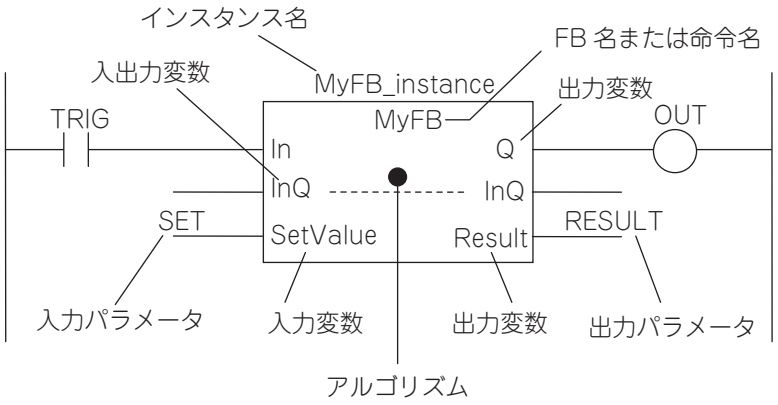
1　FB 定義を作成する
アルゴリズムを作成します。

2　FB 定義を、インスタンスとしてプログラムに配置する
FB 定義を、プログラムや他の FB から呼び出して実際に配置します。
1 つの FB 定義を、複数のプログラムや他の FB で呼び出して配置することができます。
FB 定義は、プログラムや他の FB に配置した時点で、インスタンスとして、個別に扱われるようになり、プログラムや他の FB 上で、実際に動作できるようになります。

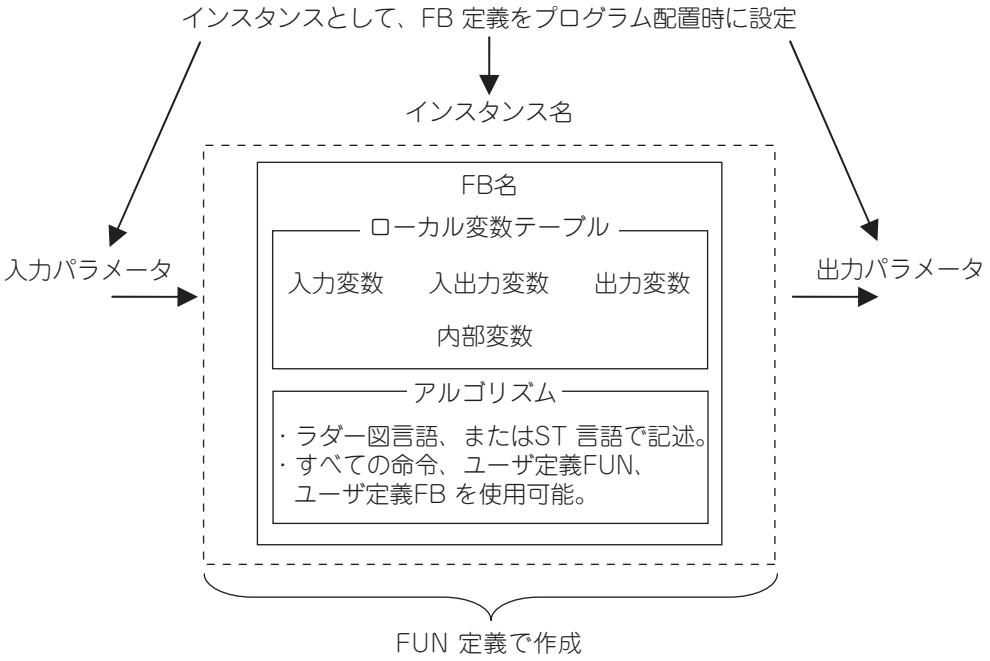
ファンクションブロック（FB）の構成

ラダー図言語では、FB は、以下のように長方形で表現されます。ST 言語での表現は、後述の「ST 言語でのファンクションブロック（FB）呼び出しの表現」（P.6-11）を参照してください。
FB は、以下から構成されます。

- ラダー図言語で表現した FB：



- FB の設定する項目
  FB 定義とインスタンスで配置時に、それぞれ以下のように設定します。



● FB 名または命令名

作成時に付けた FB 定義の名称または命令名です。

● インスタンス名

FB を管理するため、プログラムごとに配置した FB に対し、個別に付ける名前です。
プログラムや他の FB から、FB を呼び出す際に、インスタンス名を設定します。

● アルゴリズム

ラダー図言語または ST 言語で記述できます。
アルゴリズム部の中で、すべての命令、ユーザ定義 FUN、およびユーザ定義 FB を記述することができます。

● ローカル変数テーブル

入力変数、出力変数、入出力変数、内部変数、外部変数が定義されています。
📖「 ファンクションブロック（FB）内の変数の仕様」（P.6-12）を参照してください。

● パラメータ

入力変数への入力パラメータ
FB 実行開始時に、FB 内の入力変数に値を渡します。変数または定数を記述します。

出力変数からの出力パラメータ
FB 実行終了時に、FB 内の出力変数の値を受け取ります。変数を記述します。

入出力変数 ― 入出力変数間で共有する、入出力パラメータ
FB 内で値を変更します。入力と出力に同じ変数を記述します。

参考

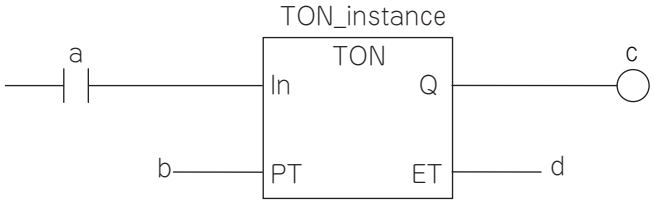
入力パラメータ、出力パラメータは省略することも可能です。動作の詳細は、📖「 パラメータ省略時の動作」（P.6-23）」を参照してください。

ST 言語でのファンクションブロック（FB）呼び出しの表現

ST 言語では、FB の呼び出しは次のように表現します。

インスタンス名（入力変数 1:= 入力パラメータ 1, ･･･入力変数 N:= 入力パラメータ N, 入出力変数 1:= 入出力パラメータ 1, ･･･入出力変数 N:= 入出力パラメータ N, 出力変数 1=> 出力パラメータ 1, ･･･出力変数 N=> 出力パラメータ N）;

記述するパラメータの順序を FB 定義に書かれたとおりにすることで入力変数名等を省略し、パラメータだけで記述することも可能です。
この場合、パラメータの数は FB 定義の入力変数等の数と一致している必要があります。



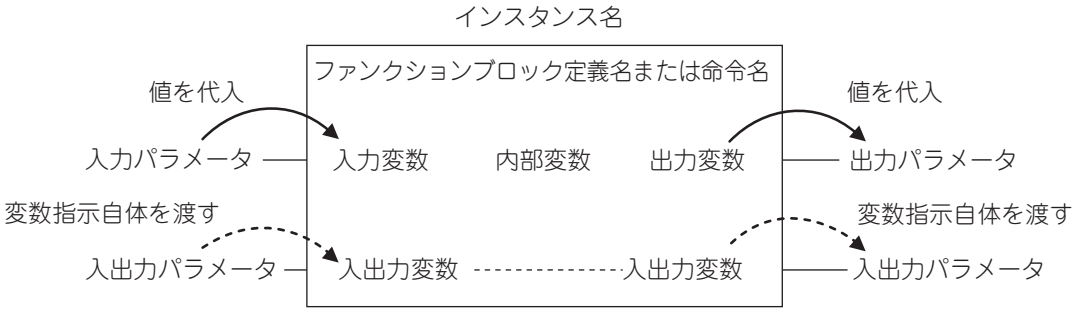
ST 言語で表現した FB：

インスタンス名

```
TON_instance(In:=a, PT:=b, Q=>c, ET=>d);
TON_instance(In:=a, PT:=b, Q=>c); (* 出力ETを省略した場合 *)
TON_instance(a,b,c,d); (* 入力変数、出力変数を省略した場合 *)
```

詳細は、後述の🕮「FB 呼出」（P.6-115）を参照してください。

ファンクションブロック（FB）内の変数の仕様



FB の変数の仕様は以下のとおりです。

変数	個数	仕様
入力変数	1 ～ 64	FB の入力引数として FB 内部で使用される（値を変更できない）変数です。
		• FB 実行開始時に、入力パラメータの値を入力変数にセットします。 • 入力パラメータには定数または変数を記述できます。 • 入力パラメータの省略： 🕮「 パラメータ省略時の動作」（P.6-23）を参照してください。 • 立ち上がり／立ち下がりエッジを指定できます。 • FB 外から、値を参照および変更することが可能です。「インスタンス名 . 入力変数名」で指定します。
出力変数[*1]	1 ～ 64	FB からの出力引数として FB 内部で使用される変数です。
		• FB 実行終了時に、出力変数の値を出力パラメータにセットします。 • 出力パラメータに、定数またはコンスタント属性ありの変数は指定できません。 • 出力パラメータ接続を省略することが可能です。省略した場合、出力変数の値は、どのパラメータにも代入されません。 • FB 外から、値を参照することが可能です。「インスタンス名 . 出力変数名」で読出可能です。ただし、書込不可です。
入出力変数	0 ～ 64	FB への入出力として FB 内部で使用される（値を変更できる）変数です。
		•「入出力パラメータ」を渡し、また受けます。 • 入出力パラメータに、定数またはコンスタント属性ありの変数は指定できません。 • FB 内で値を変更すると、その時点で入出力パラメータの値が変わります。 • 入出力パラメータを省略することはできません。
内部変数	無制限	FB 内での一時的な格納用変数です。
		• 非実行時も、実行後も、値が保持されます。 • 保持属性を持たせることができます。 • FB 外から、値を参照することはできません。
外部変数	無制限	グローバル変数を参照する変数です。
EN	0	FB では使用できません（ユーザ定義 FB、FB 型命令とも）。
ENO	0 または 1	一般的に正常終了時に TRUE、異常終了時または非起動時に FALSE となる BOOL 型の出力変数として用います。
		• ユーザ定義 FB では省略可能です。[*2] • その他の詳細は、後述の🕮「ENO」（P.6-20）を参照してください。

*1. ラダー図言語で FB を使用する場合、ENO を含めて BOOL 型の出力変数が 1 個以上必要です。
*2. 入力パラメータに異常が発生すると、パワーフロー出力が FALSE となります。ユーザ定義 FB で ENO を省略した場合、次回以降の FB 呼び出し時にも FALSE のままであり、TRUE にすることができません。入力パラメータに異常が発生する可能性がある場合は ENO を定義し、FB 呼び出し時に TRUE となるようにアルゴリズムを記述してください。入力パラメータの異常時の動作については、🕮「パラメータの異常時の動作」（P.6-24）を参照してください。

各変数の変数属性の設定の可否は、🕮「6-3-4 変数の属性」（P.6-29）を参照してください。

参考

グローバル変数の外部変数への定義は、グローバル変数と同名の変数を FB 内の外部変数で定義すると、自動的に設定されます。

● ENO

ENO が FALSE の場合、その他の出力変数の値は、前回の値を保持します。

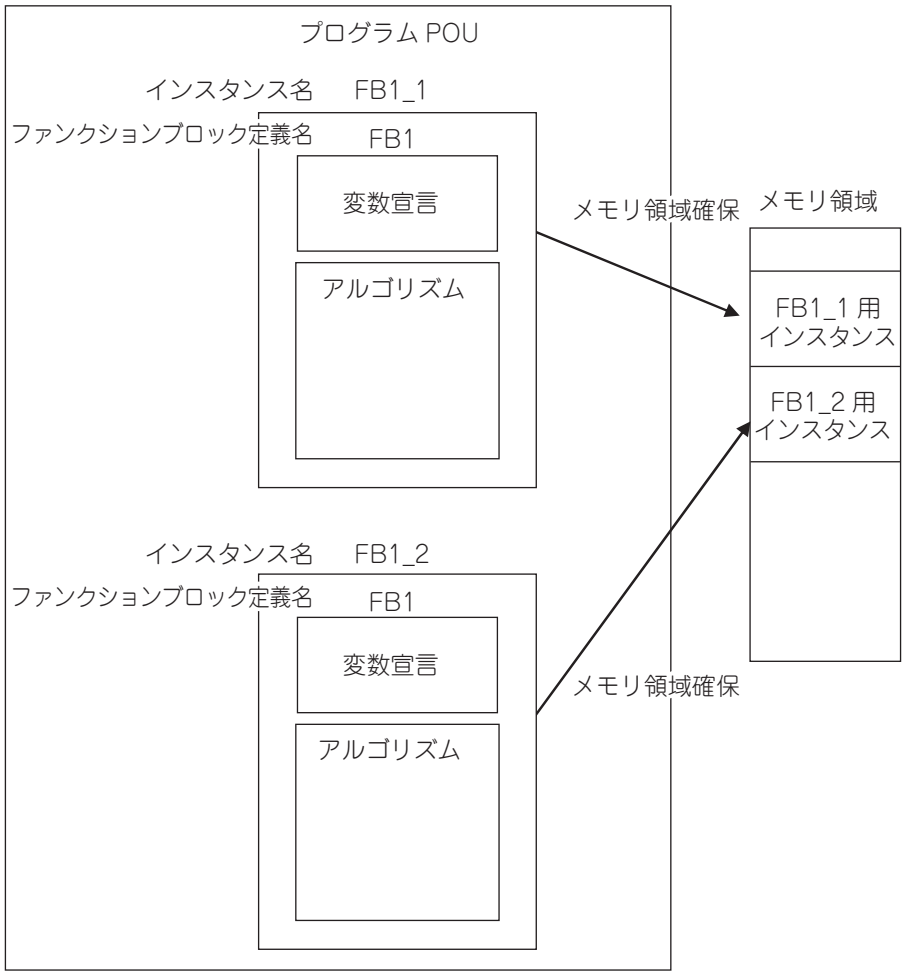
ファンクションブロック（FB）定義とインスタンス

FB は、あらかじめ作成した「FB 定義」と、それを実際にプログラム上で使用するための「インスタンス」からなります。
再利用の元になるものを「FB 定義」と呼びます。
「FB 定義」には、アルゴリズムとローカル変数テーブルを記載します。

● FB のインスタンス

FB 定義をプログラムまたは他の FB 上に配置すると、FB 定義は実際のプログラムや FB の一部として扱われるようになります。
FB 定義をプログラムまたは他の FB 上に配置したものは、「インスタンス」と呼びます。
各インスタンスには、インスタンス名という識別子を持ち、それぞれがメモリ領域を使用します。

1 つの FB 定義をベースに、複数の「インスタンス」を生成することにより、同一機能で異なる入出力データの処理をさせることができます。

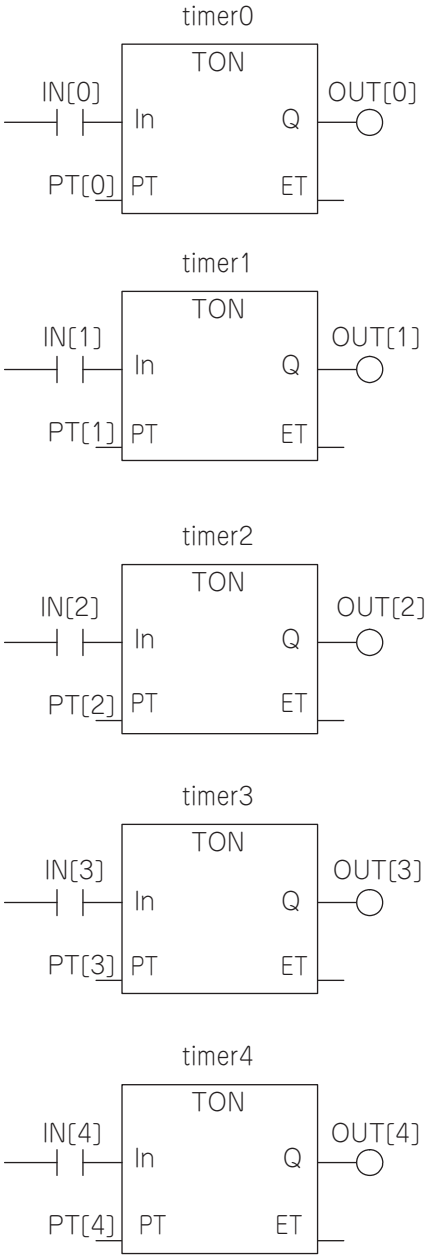


「インスタンス」は、異なるプログラムや、他の FB から読み出すことはできません。インスタンス名が同じでも、異なるプログラムや、他の FB に配置された場合は、異なるインスタンスとして動作します。

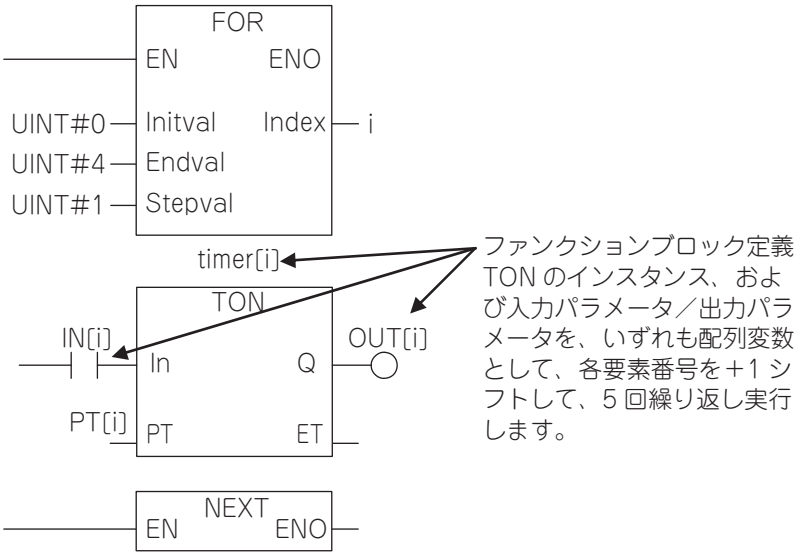
インスタンスの配列指定化

インスタンスに対して配列指定することができます。
要素番号を変数の値によって間接指定することで、ひとつのインスタンス名で、複数のインスタンスを実行することができます。
さらに、入力パラメータおよび出力パラメータも配列指定とし、共通の変数で要素番号を指定することで、ひとつのインスタンス名で、入力元、出力先を切り替えて、複数のインスタンスを実行することができます。
例）

インスタンスを配列指定しないとき



インスタンスを配列指定したとき



変数テーブル

変数名	データ型
IN	ARRAY [0..4] OF BOOL
OUT	ARRAY [0..4] OF BOOL
PT	ARRAY [0..4] OF TIME
timer	ARRAY [0..4] OF TON
i	UINT

ファンクションブロック（FB）の起動条件

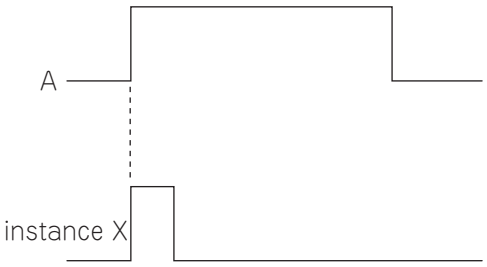
FB は、FUN と異なり、EN 入力をもたず、常時（毎タスク周期に）実行されます。

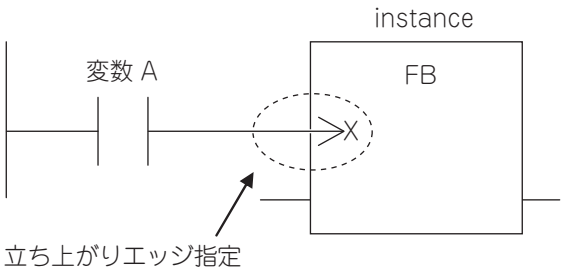
場合	FB 内アルゴリズム		ENO[*1]	ENO 以外の動作
通常時	実行されます。	正常終了時	TRUE	出力パラメータ： 内部アルゴリズムによって値を更新 入出力パラメータ： 内部アルゴリズムによって値を更新
		異常終了時	FALSE	出力パラメータ： 保持 入出力パラメータ： 内部アルゴリズムによって値を更新
マスタコントロール領域内	パワーフロー入力が FALSE の状態で実行する。		ユーザ任意	ENO の値によって上記のいずれかとなる

*1. FB の ENO の仕様については、🕮「ファンクションブロック（FB）内の変数の仕様」（P.6-12）を参照してください。

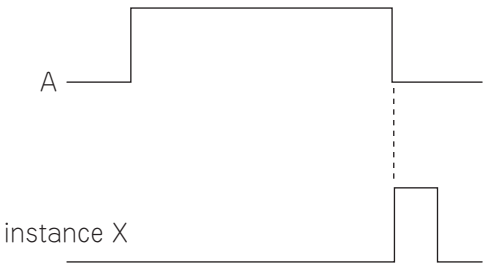
パワーフロー出力、パラメータ出力については、🕮「6-5-2 ラダー図言語（LD）」（P.6-84）を参照してください。

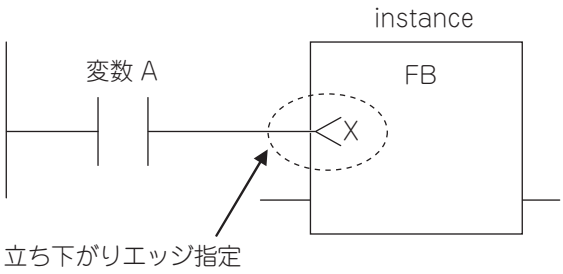
入力変数にエッジを指定することにより、その変数は入力パラメータの立ち上がり時に TRUE となります。





立ち下がり指定も可能です。





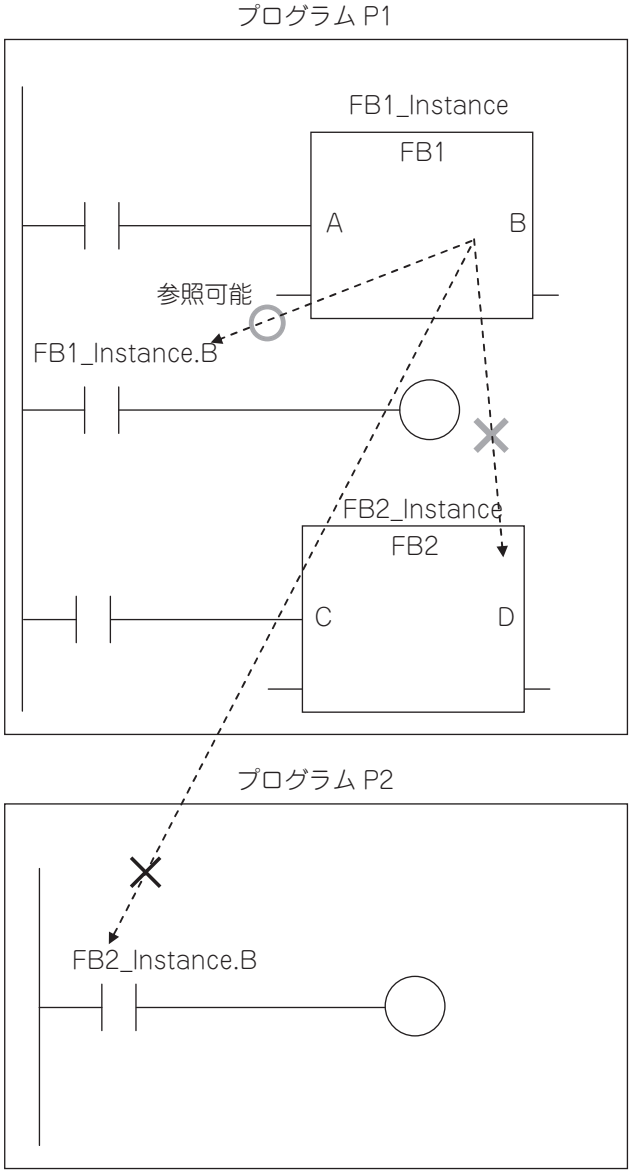
ファンクションブロック（FB）の変数の外部からの参照

FB の入力変数および出力変数は、FB の外部から参照することが可能です。変数の表記は、以下のようになります。

インスタンス名 . 変数名

（例） FB1 のインスタンス「FB1_Instance」内の出力変数「B」の場合
FB1_Instance.B

参照可能な範囲は、そのインスタンスがあるプログラム内だけです。
そのプログラム内の他の FB のインスタンスからは参照できません。また、他のプログラムからも参照できません。



FB1 の入力変数、出力変数は、
・プログラム P1 の中でしか参照できません。
・プログラム P2 からは参照できません。
・FB2 の中では参照できません。

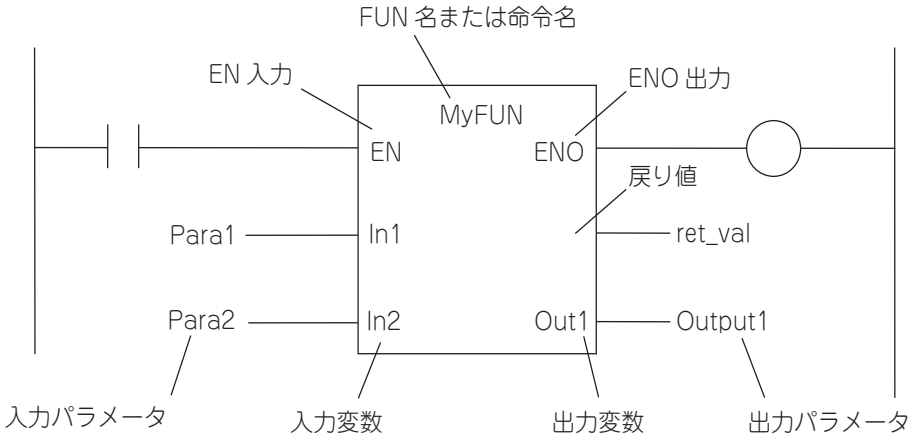
次の変数は、外部から参照できません。参照した場合は、ビルド時にエラーとなります。

- FB の入出力変数
- 入力パラメータが省略された場合に初期値が適用されない、FB 型命令の入力変数

6-2-6　ファンクション（FUN）の詳細

ファンクション（FUN）の構成

ラダー図言語では、FUN は、以下のように長方形で表現されます。
ST 言語での表現は、後述の 「ST 言語でのファンクション（FUN）の表現」（P.6-18）を参照してください。
FUN は、以下から構成されます。
ラダー図言語で表現した FUN：



● FUN 定義名または命令名

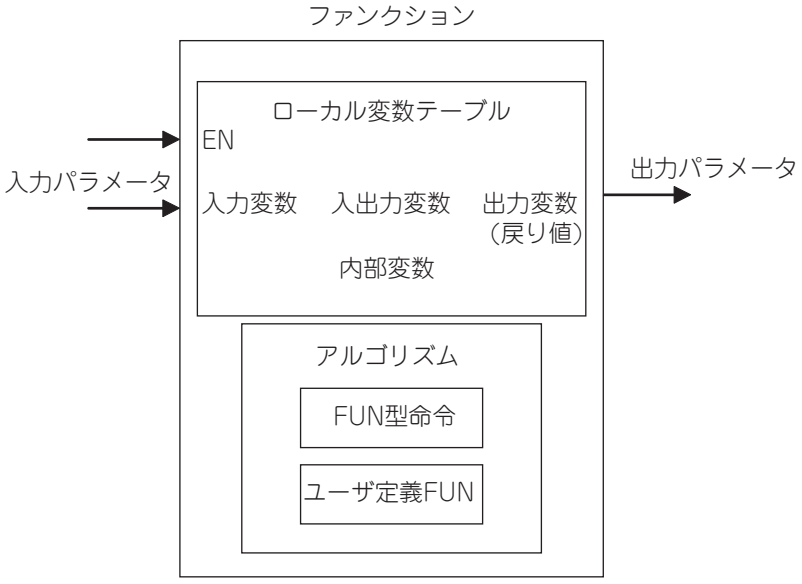
作成時に付けた FUN 定義の名称または命令名です。

● インスタンス名

ありません。

● アルゴリズム

ラダー図言語または ST 言語で記述できます。
アルゴリズム部の中で、FUN 型命令、ユーザ定義 FUN を記述することができます。
FB 型命令、ユーザ定義 FB を記述することはできません。
微分命令（R_TRIG、UP）を記述することはできません。
システム定義変数 P_First_RunMode、P_First_Run を使用することはできません。



● ローカル変数テーブル

入力変数、出力変数、入出力変数、内部変数、外部変数を登録します。
🕮「 ファンクション（FUN）内の変数の仕様」（P.6-19）を参照してください。

● パラメータ

入力変数への入力パラメータ
FUN 実行開始時に、FUN 内の入力変数に値を渡します。変数または定数を記述します。

出力変数からの出力パラメータ
FUN 実行終了時に、FUN 内の出力変数の値を受け取ります。変数を記述します。

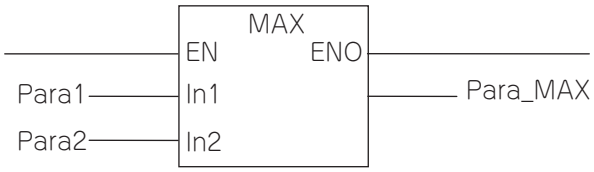
入出力変数 ― 入出力変数間で共用する入出力パラメータ
FUN 内で値を変更します。入力と出力に同じ変数を記述します。

ST 言語でのファンクション（FUN）の表現

ST 言語では、FUN は次のように表現します。

戻り値 :=FUN 名(入力変数 1:= 入力パラメータ 1, ･･･入力変数 N:= 入力パラメータ N, 入出力変数 1:= 入出力パラメータ 1, ･･･入出力変数 N:= 入出力パラメータ N, 出力変数 1=> 出力パラメータ 1, ･･･出力変数 N=> 出力パラメータ N）;
ただし、戻り値を省略することも可能。

記述するパラメータの順序を FUN 定義に書かれたとおりにすることで入力変数名等を省略し、パラメータだけで記述することも可能です。この場合、パラメータの数は FUN 定義の入力変数等の数と一致している必要があります。
ST 言語で表現した FUN の例：

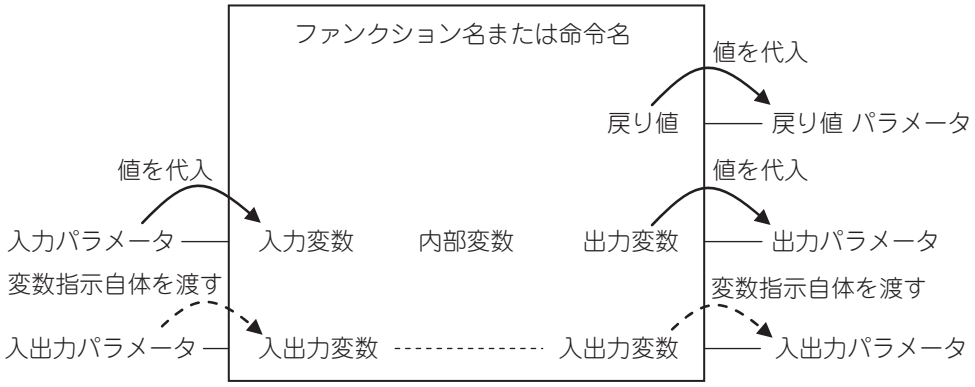


FUN名

```
Para_MAX := MAX(In1:=Para1, In2:=Para2);
Para_MAX := MAX(Para1, Para2);              (* 入力変数を省略した場合 *)
```

詳細は、後述の🕮「FUN 呼出」（P.6-118）を参照してください。

ファンクション（FUN）内の変数の仕様



FUN の変数の仕様は以下のとおりです。

変数	個数	仕様
入力変数	0 ～ 64	FUN の入力引数として FUN 内部で使用される（値を変更できない）変数です。
		• FUN 実行開始時に、入力パラメータの値を入力変数にセットします。 • 入力パラメータには定数または変数を記述できます。 • 入力パラメータの省略： 　「 パラメータ省略時の動作」（P.6-23）を参照してください。 • FB と異なり、立ち上がり／立ち下がりエッジを指定できません。 • FUN 外から、値を参照することはできません。 • オムロンが提供する命令には、入力変数の数が可変のものがありますが、ユーザが作成する FUN は、入力変数の数を可変にすることはできません。
出力変数	0 ～ 64	FUN からの出力引数として FUN 内部で使用される変数です。
		• FUN 実行終了時に、出力変数の値を出力パラメータにセットします。 • 出力パラメータに、定数またはコンスタント属性ありの変数は指定できません。 • ENO、戻り値を含み、最低 1 個は BOOL 型の出力変数が必要です。 • 出力パラメータ接続を省略することが可能です。 　省略した場合、出力変数の値は、どのパラメータにも代入されません。 • FUN 外から、値を参照することはできません。 • ユーザ定義の FUN の場合、FUN 内のアルゴリズムで必ず出力変数の値を設定してください。FUN 内のアルゴリズムで出力変数を設定しないと、出力変数の値が不定になります。
入出力変数	0 ～ 64	FUN への入出力として FUN 内部で使用される（値を変更できる）変数です。
		•「入出力パラメータ」そのもの（変数指示自体）を渡し、また受けます。 • 入出力パラメータに、定数またはコンスタント属性ありの変数は指定できません。 • FUN 内で値を変更すると、その時点で入出力パラメータの値が変わります。 • 入出力パラメータを省略することはできません。 • FUN 外から、値を参照することはできません。
内部変数	無制限	FUN 内での一時的な格納用変数です。
		• 実行終了後は、値が保持されません。 • FUN 外から、値を参照することはできません。
外部変数	無制限	グローバル変数を参照する変数です。
EN	1	FUN を起動するための BOOL 型の入力変数です。
		• EN の値が TRUE のときに、FUN が実行されます。 • 必ず 1 個存在します（ユーザ定義 FUN、FUN 型命令とも）。

変数	個数	仕様
ENO	0 または 1	一般的に正常終了時に TRUE、異常終了時または非起動時に FALSE となる BOOL 型の出力変数として用います。
		• ユーザ定義 FUN では省略可能です。 • その他の詳細は、後述の 🕮「ENO」（P.6-20）」を参照してください。
戻り値	1	FUN がアルゴリズムを終了し、呼び出し元に処理の結果として返す値です。
		• 必ず 1 個存在します。 • 基本データ型のすべて、列挙型が指定可能です。配列指定、構造体全体、共用体全体を指定することはできません。 • ユーザ定義の FUN の場合、FUN 内のアルゴリズムで必ず戻り値を設定してください。FUN 内のアルゴリズムで戻り値を設定しないと、戻り値が不定になります。 • その他の詳細は、後述の 🕮「戻り値」（P.6-21）を参照してください。

各変数の変数属性の設定の可否は、🕮「6-3-4 変数の属性」（P.6-29）を参照してください。

参考

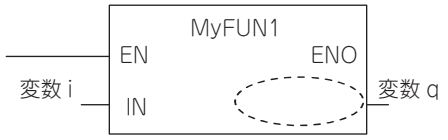
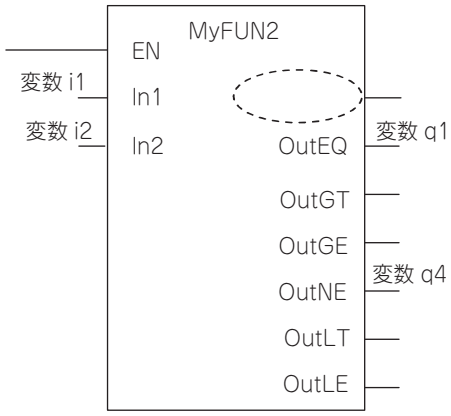
FUN 内変数テーブルにてグローバル変数を外部変数として登録することで、グローバル変数を参照することができます。
一般的にファンクションは、入力パラメータの値から出力値が一意となるように作ることが推奨されています。ファンクション内でグローバル変数を参照し、それによって出力値が影響を受けるアルゴリズムは好ましくありません。そのため、Sysmac Studio にてプログラムチェック時に、「ファンクション内でのグローバル変数の使用は推奨しておりません。」と警告が表示されます。確認の上、必要な措置をとってください。

● ENO

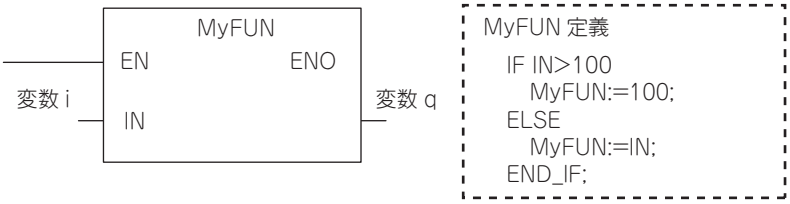
- ENO が FALSE の場合、その他の出力変数の値は、前回の値を保持します。

● 戻り値

- 戻り値はラダー図言語上では、空白で表示されます。

場合	ラダー図言語表記	ST 言語表記
戻り値を使用する場合		変数 q := MyFUN1（変数 i）;
戻り値を使用しない場合		MyFUN2（In1:= 変数 i1, In2:= 変数 i2, OutEQ=> 変数 q1, OutNE=> 変数 q4 ）;

- ラダー図言語、ST 言語とも、呼出側で戻り値を使用しないことも可能です。
- ファンクション内のアルゴリズムで、戻り値に値をセットする場合、ファンクション名と同名の変数に値をセットしてください。
  たとえば、「MyFUN」というファンクションの場合、戻り値は「MyFUN」となります。

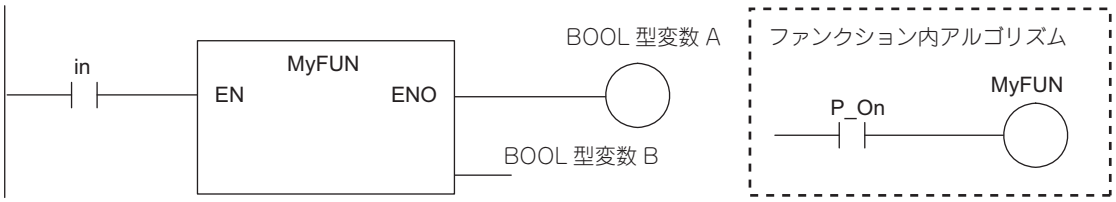


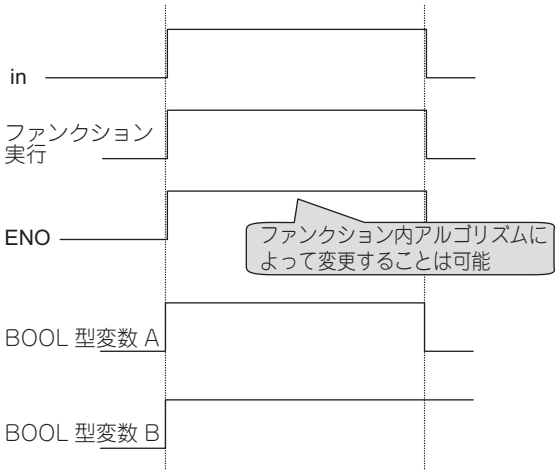
ファンクション（FUN）の起動条件

EN が TRUE になることで、FUN が起動します。EN が FALSE になると、FUN は処理を停止します。

入力変数	FUN 内アルゴリズム		ENO	ENO 以外の動作
EN=TRUE 時	実行されます	正常終了時	TRUE	出力パラメータ： 内部アルゴリズムによって値を更新 入出力パラメータ： 内部アルゴリズムによって値を更新
		異常終了時	FALSE	出力パラメータ： 保持 入出力パラメータ： 内部アルゴリズムによって値を更新
EN=FALSE 時	実行されません		FALSE	出力パラメータ、入出力パラメータ： 値を保持
マスタコントロール領域内	実行されません		FALSE	出力パラメータ、入出力パラメータ： 値を保持

例）





6-2-7　ファンクションブロック（FB)、ファンクション（FUN）共通動作

EN、ENO の使用／省略の可否

FB /FUN における、EN、ENO の使用／省略の可否は、以下のとおりです。

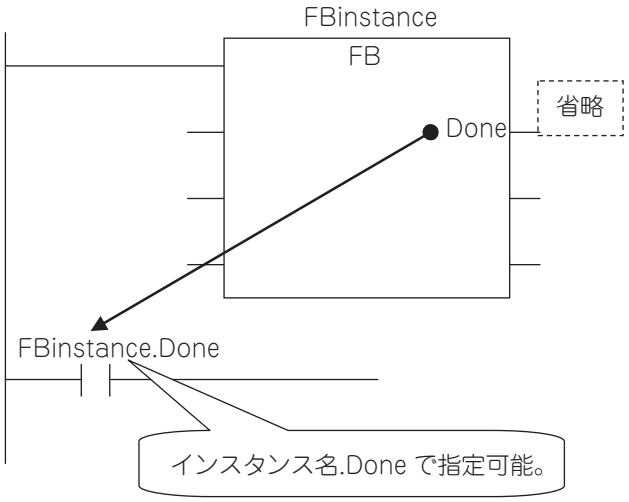
POU		EN	ENO
FB	ユーザ定義	使用不可。 Sysmac Studio にて変数テーブルに定義しようとすると、ビルド時にエラーとなります。	使用可能（省略可能)。 Sysmac Studio にてユーザが出力変数として定義します。
	命令	すべての FB 型命令は EN を使用しません。	ENO を使用する命令と、使用しない命令があります。
FUN	ユーザ定義	使用必須。 FUN 作成時に、Sysmac Studio がデフォルトで自動的に変数テーブルに定義します。	使用可能（省略可能)。 Sysmac Studio にてユーザが出力変数として定義します。
	命令	すべての FUN 型命令は EN を使用します。	ENO を使用する命令と、使用しない命令があります。

パラメータ省略時の動作

入力パラメータ・出力パラメータはともに省略することができます。

	省略した時の動作	
省略対象のパラメータ	FB	FUN
入力変数への入力パラメータ	• インスタンス初回起動時は、初期値で動作します。 • 以降は前回値で動作します（常に入力変数が省略されている場合は初期値で動作し続けます)。	• EN は TRUE で動作します。 • その他の入力パラメータは、初期値で動作します。
出力変数からの出力パラメータ	省略可能です。 その命令外で、「インスタンス名 . 出力変数名」でアクセス可能です（*)	省略可能ですが、省略時は動作結果を得る手段がありません。
入出力変数への、入出力変数からの入出力パラメータ	省略できません。	省略できません。

* FB の入力変数 / 出力変数を、「インスタンス名 . 変数名」で FB 外（ただしそのプログラムだけ）から直接参照することができます。FUN では、参照不可です。



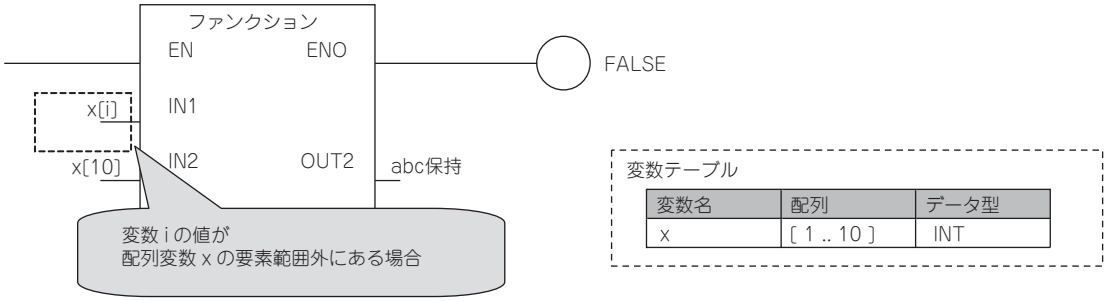
パラメータの異常時の動作

入力パラメータ、出力パラメータ、または入出力パラメータに異常が発生した場合、以下の動作となります。

● 入力に異常発生時

入力パラメータに異常を検知した場合、FB/FUN は実行せず、ENO を FALSE とします。同時に、パワーフロー出力は FALSE とし、それ以外は値を保持します。

例）



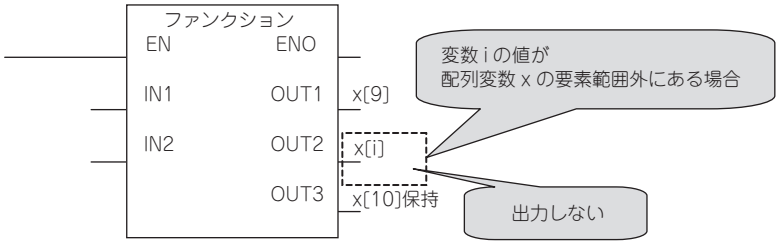
使用上の注意

入力パラメータに異常が発生すると、パワーフロー出力が FALSE となります。ユーザ定義 FB で ENO を省略した場合、次回以降の FB 呼び出し時にも FALSE のままであり、TRUE にすることができません。入力パラメータに異常が発生する可能性がある場合は ENO を定義し、FB 呼び出し時に TRUE となるようにアルゴリズムを記述してください。

● 出力パラメータに異常発生時

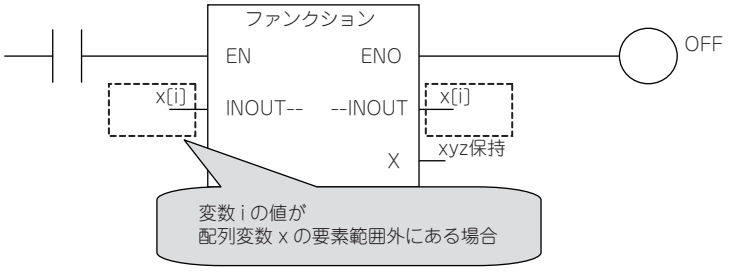
出力パラメータに異常を検知した場合、該当パラメータ以降に値を出力せず、値を保持します。

例）



● 入出力に異常発生時

入出力パラメータに異常を検知した場合、FB/FUN は実行せず、ENO を FALSE とします。同時に、パワーフロー出力は FALSE とし、それ以外は値を保持します。



再帰呼び出し

FB および FUN とも、以下の再帰呼出はできません。その場合、Sysmac Studio でビルド時にエラーとなります。

- 自分自身を呼び出す
- 呼び出された FB または FUN 側で、元の親を呼び出す

6-2-8　POU の制限事項

POU を作成するうえでの、各種制限事項を説明します。

名称

POU 名、FB のインスタンス名の制約は、📖「6-3-12 変数名などのプログラム関連の名称の制限」（P.6-77）を参照してください。

複数の引数渡し

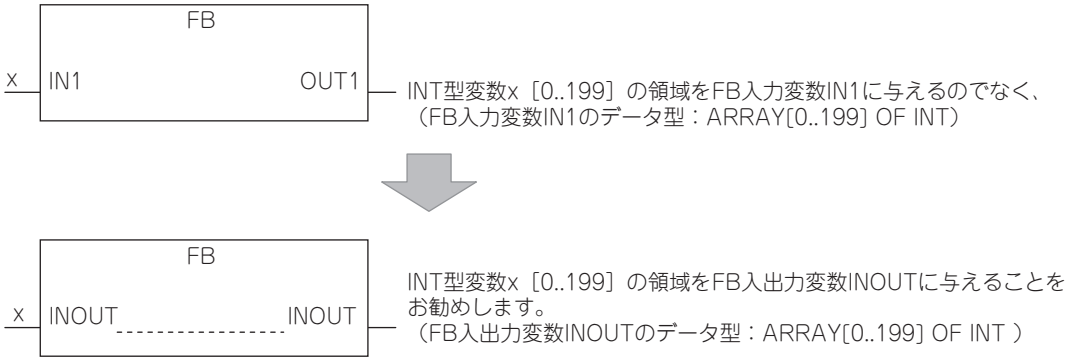
複数の引数を、FB や FUN に渡す場合、配列指定や構造体型でデータを渡す形式にすると、プログラムが簡潔になります。
この場合、入力変数で渡すよりも、入出力変数で渡す方が、処理時間が短くなります。ただし、入出力変数を使用すると、FB や FUN に引数として渡したデータの中身が書き替えられた場合は、元の値は残らないので、ご注意ください。

参考

配列指定の変数・構造体型変数のメンバ全体をパラメータとするとき
入力パラメータ、出力パラメータに、配列指定の変数または構造体型変数のメンバ全体を指定することも可能です。ただし、基本データ型の変数に比べ、データの受け渡しに時間を要します（サイズによる）。
そのため、FB の場合、配列指定の変数・構造体型変数のメンバ全体を受け渡しするときは、入出力変数に受け渡しをするように設計することをお勧めします。
例 1）配列指定の全要素を指定するとき

プログラムの変数テーブル

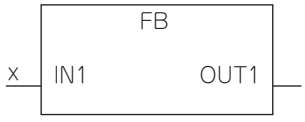
変数名	データ型
x	ARRAY[0..199] OF INT



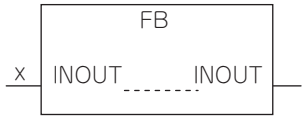
例 2）構造体型変数のメンバ全体を指定するとき

プログラムの変数テーブル

変数名	データ型
x	MyStructure



I MyStructure型変数xのメンバ全体の領域をFB入力変数IN1に与えるのでなく、
（FB入力変数IN1のデータ型：MyStructure）



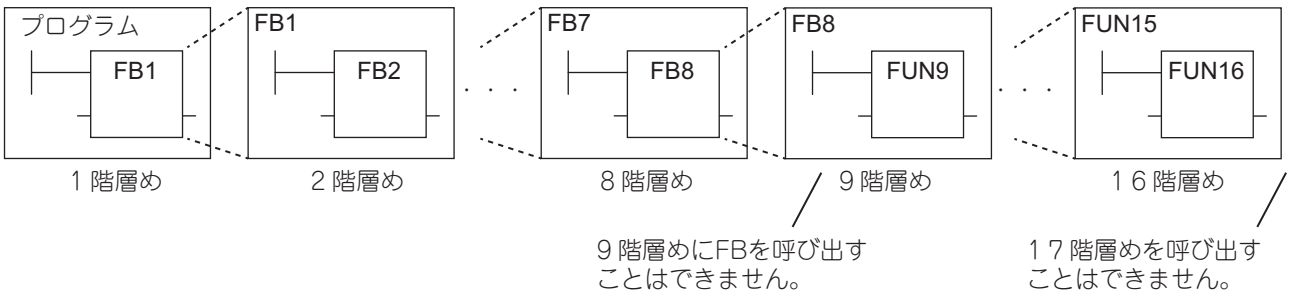
MyStructure型変数xのメンバ全体の領域をFB入出力変数INOUTに与えることをお勧めします。
（FB入出力変数INOUTのデータ型：MyStructure）

ネスティングの階層

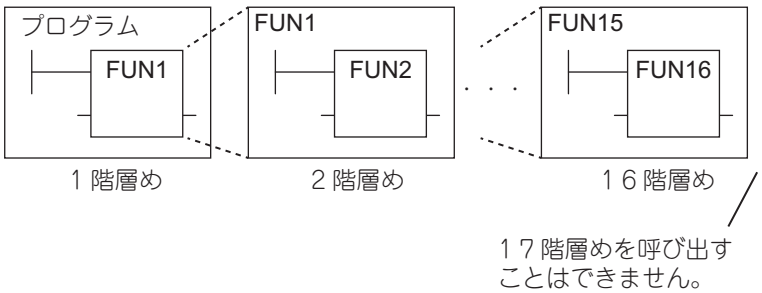
プログラムから呼び出された FB や FUN から、さらに別の FB や FUN を呼び出すことをネスティングと呼びます。ユーザ定義の、FB や FUN から呼び出せる POU と、ネスティングの階層の深さには、以下の制限があります。制限を超えると、ビルド時にエラーとなります。

POU	呼び出せる POU	階層の深さ
FB	FB、FUN	最大 8 階層
FUN	FUN	最大 16 階層

例 1）プログラムから 8 階層めまで FB を呼び出すことができます。さらに、そこから 16 階層めまで FUN を呼び出すことができます。



例 2）プログラムから 16 階層めまで FUN を呼び出すことができます。



6-3　変数

NY シリーズシステムでは、外部との入出力情報のやり取りやデータの演算などの処理を、すべて変数を介して行います。
ここでは、変数の仕様について、詳しく説明します。
Sysmac Studio での変数の設定方法については、🕮『Sysmac Studio Version 1 オペレーションマニュアル（SBCA-470）』を参照してください。

6-3-1　変数とは

変数とは、外部とやり取りした入出力データや、POU 内部処理時の一時データを格納する入れ物です。言いかえると、名前やデータ型などの属性を持ったデータの入れ物です。

変数とメモリアドレスとの割付を指定する必要はありません。NY シリーズコントローラが、変数のためのメモリ領域を自動的に割り付けます。

6-3-2　変数の種類

変数には、大別して以下の 3 種類があります。

● ユーザ定義変数

すべての属性がユーザによって定義される変数です。
本章の以下の説明では、この「ユーザ定義変数」について、説明します。

● 準ユーザ定義変数

特定のデバイス／データにアクセスするための変数です。
デバイス変数とカムデータ変数があります。
デバイス変数については、🕮「2-3-1 変数の種類」（P.2-10）および🕮「3-3-1 I/O ポート」（P.3-5）を参照してください。

● システム定義変数

あらかじめ NY システムとして、名称などのすべての属性が定義されており、かつ特定機能が割り付けられている変数です。機能モジュールごとに用意されています。詳細は🕮「A-3 システム定義変数一覧」（P.A-22）を参照してください。

変数の分類の詳細については、🕮「2-3-1 変数の種類」（P.2-10）を参照してください。

6-3-3　ユーザ定義変数の POU からみた種類

ユーザ定義変数には、POU での機能によって、以下の 6 種類があります。

○：定義可能

ユーザ定義変数の種類		POU タイプ		
		プログラム	FB	FUN
ローカル変数	内部変数	○	○	○
	入力変数	なし	○	○
	出力変数	なし	○	○
	入出力変数	なし	○	○
グローバル変数		○（*）	○（*）	○（*）
外部変数		○	○	○

* グローバル変数を外部変数として定義することで、外部変数を介して、グローバル変数を参照することができます。

ローカル変数

POU（プログラム、FB、FUN）からだけ読み書き可能な変数です。
ローカル変数は、POU がプログラムの場合、内部変数とイコールです。POU が FB または FUN の場合、内部変数、入力変数、出力変数、入出力変数、外部変数の総称です。

● 内部変数

POU の内部だけで使用する変数です。
POU のローカル変数テーブルで定義（宣言）します。
POU の外部から、内部変数の値を参照することはできません。
同じ名前の内部変数を異なる POU で宣言しても、それぞれの変数ごとに異なる割付先に割り付けられます。

● 入力変数

POU が呼び出される時に、呼出元から入力パラメータの値が代入される変数です。POU のローカル変数テーブルで定義（宣言）します。

● 出力変数

POU の処理が終了する前に、出力パラメータにその値が代入され、呼出元に返される変数です。
POU のローカル変数テーブルで定義（宣言）します。

● 入出力変数

POU 呼び出し時に、呼出元から入出力パラメータそのもの（変数指示自体）が代入される変数です。POU 内で書き替えられると、その時点で入出力パラメータの値が書き変わります。
POU のローカル変数テーブルで定義（宣言）します。

● 外部変数

POU からその外部を参照するための変数です。
登録されたグローバル変数を、POU から参照します。

グローバル変数

グローバル変数テーブルで定義（宣言）します。

スレーブ／ユニット構成から自動生成したデバイス変数、および全軸設定テーブルから自動生成した「軸 / 軸グループ変数」は、自動的にグローバル変数に登録されます。

6-3-4　変数の属性

変数には、以下の属性を設定・付加することができます。

変数の属性一覧と、変数種類別の属性有無

● 変数の属性一覧

属性	説明	指定	初期設定
変数名	変数を識別するための名称です。	−	−
データ型	変数に格納するデータを扱う形式を定義します。	−	BOOL
AT（割付先）指定	NX ユニットや EtherCAT スレーブの I/O ポートを変数として扱うために、その変数の割付先を指定します。	• 指定なし • 指定する	指定なし
保持	以下のいずれかのときに値を保持するかしないかを指定します。 • 電断後の電源投入時 • 運転モードへの切替時 • 全停止フォールトレベルのコントローラ異常発生時	• 保持：左記の時の値を保持する*1 • 非保持：初期値になる	非保持：初期値になる
初期値	初期値設定「なし」と「あり」を選択できます。 初期値設定「あり」の場合：保持指定なしで、以下のいずれかのときの変数の値を指定します。 • 電源投入時 • 動作モード変更時 • 全停止フォールトレベルのコントローラ異常発生時 初期値設定「なし」の場合：値は保持されません	初期値設定 • あり • なし	データ型によります。 （「 初期値」(P.6-60)」の項を参照してください。）
コンスタント	変数のダウンロード時に初期値を設定し、それ以降は値を書き換えできない属性です。	コンスタントにする／しない	−
ネットワーク公開	CIP 通信やデータリンク機能を使って、コントローラ外部から変数の読み書きができるようにします。	• 公開しない • 公開だけ • 入力 • 出力	公開しない
エッジ	FB の入力パラメータの、立ち上がり、立ち下がりを検出します。 BOOL 型の入力変数だけに設定できます。	• なし • 立ち上がり • 立ち下がり	なし

*1. 産業用 PC に UPS を接続し、正常にシャットダウンした場合に、不揮発性メモリに保持されます。

参考

タスク間排他制御機能
ひとつのタスクからしか、グローバル変数への書込ができないようにすることで、処理中にグローバル変数の値が変化することを防ぎます。変数の属性としてでなく、タスク設定で指定します。

● 変数種類別の属性の有無

変数種類		変数名	データ型	AT（割付先）指定	保持	初期値	コンスタント	ネットワーク公開	エッジ
グローバル変数		設定可能	設定可能	設定可能	保持可能	設定可能	指定可能	公開可能	不可
プログラム	内部変数	設定可能	設定可能	設定可能	保持可能	設定可能	指定可能	公開不可	不可
	外部変数	設定不可	設定不可	設定不可	保持不可	設定不可	指定可能	公開不可	不可
ファンクションブロック	内部変数	設定可能	設定可能	設定可能	保持可能	設定可能	指定可能	公開不可	不可
	入力変数	設定可能	設定可能	設定不可	保持可能	設定可能	指定可能	公開不可	可能
	出力変数	設定可能	設定可能	設定不可	保持可能	設定不可	指定不可	公開不可	不可
	入出力変数	設定可能	設定可能	設定不可	保持不可	設定不可	指定可能	公開不可	不可
	外部変数	設定不可	設定不可	設定不可	保持不可	設定不可	指定可能	公開不可	不可
ファンクション	内部変数	設定可能	設定可能	設定不可	保持不可	設定可能	指定可能	公開不可	不可
	入力変数	設定可能	設定可能	設定不可	保持不可	設定可能	指定可能	公開不可	不可
	出力変数	設定可能	設定可能	設定不可	保持不可	設定不可	指定不可	公開不可	不可
	入出力変数	設定可能	設定可能	設定不可	保持不可	設定不可	指定可能	公開不可	不可
	外部変数	設定不可	設定不可	設定不可	保持不可	設定不可	指定可能	公開不可	不可

6-3-5 データ型

データ型とは、変数が表現する値の形式と範囲を特定する属性です。

データ型の種類によって、その変数の宣言時に確保されるメモリのサイズが違います。メモリのサイズが大きいほど、表現できる値の範囲が大きくなります。

また、命令の入力／出力／入出力変数のデータ型は、命令ごとに決められています。命令の引数である入力／出力／入出力パラメータのデータ型は、対応する入力／出力／入出力変数のデータ型に合わせて設定してください。

基本データ型と派生データ型

データ型には、あらかじめ仕様が決まっている基本データ型と、ユーザが仕様を定義する派生データ型があります。

● 基本データ型の種類

基本データ型は、以下に分類できます。

分類	定義
ブール型	値が 0 または 1 のいずれかのデータ型です。
ビット列型	ビットの列で値を表現したデータ型です。
整数型	値が整数値のデータ型です。
実数型	値が実数値のデータ型です。
持続時間型	値が時間（日・時・分・秒・ミリ秒）のデータ型です。
時刻型	値が時刻（時・分・秒）のデータ型です。
日付型	値が日付（年・月・日）のデータ型です。
日付時刻型	値が日付時刻（年・月・日・時・分・秒・ミリ秒）のデータ型です。
文字列型	値が文字列のデータ型です。

基本データ型には、20 種類あります。それぞれの仕様を、以下に示します。

なお、下表の「データサイズ（容量値）」および「アライメント（境界値）」の意味は、
- データサイズ（容量値）：　　実際の値の容量です。
- アライメント（境界値）：　　メモリを確保するときの単位です。

分類	データ型	データサイズ（容量値）	アライメント（境界値）	値の範囲	記述方法
ブール型	BOOL	16 ビット	2 バイト	FALSE、TRUE	BOOL#0、BOOL#1 FALSE、TRUE
ビット列型	BYTE	8 ビット	1 バイト	BYTE#16#00 ～ FF	BYTE#2#01011010 BYTE#2#0101_1010 BYTE#16#5A （区切り文字「_」を使用することもできます。）
	WORD	16 ビット	2 バイト	WORD#16#0000 ～ FFFF	
	DWORD	32 ビット	4 バイト	DWORD#16#00000000 ～ FFFFFFFF	
	LWORD	64 ビット	8 バイト	LWORD#16#0000000000000000 ～ FFFFFFFFFFFFFFFF	
整数型*1	SINT	8 ビット	1 バイト	SINT#-128 ～ +127	100 INT#2#00000000_01100100 INT#8#144 INT#10#100 INT#16#64 -100
	INT	16 ビット	2 バイト	INT#-32768 ～ +32767	
	DINT	32 ビット	4 バイト	DINT#-2147483648 ～ +2147483647	
	LINT	64 ビット	8 バイト	LINT#-9223372036854775808 ～ +9223372036854775807	
	USINT	8 ビット	1 バイト	USINT#0 ～ +255	
	UINT	16 ビット	2 バイト	UINT#0 ～ +65535	
	UDINT	32 ビット	4 バイト	UDINT#0 ～ +4294967295	
	ULINT	64 ビット	8 バイト	ULINT#0 ～ +18446744073709551615	
実数型	REAL	32 ビット	4 バイト	REAL#-3.402823e+38 ～ -1.175495e-38 0 1.175495e-38 ～ 3.402823e+38 + ∞ /- ∞	REAL#3.14 LREAL#3.14 3.14 -3.14 1.0E+6 1.234e4
	LREAL	64 ビット	8 バイト	LREAL#-1.79769313486231e +308 ～ -2.22507385850721e-308 0 2.22507385850721e-308 ～ 1.79769313486231e+308 + ∞ /- ∞	
持続時間型 *1*2	TIME	64 ビット	8 バイト	T#-9223372036854.775808ms（T#-106751d_23h_47m_16s_854.775808ms）～ T#+9223372036854.775807ms（T#+106751d_23h_47m_16s_854.775807ms）	T#12d3h3s T#3s56ms TIME#6d_10m TIME#16d_5h_3m_4s T#12d3.5h T#10.12s T#61m5s （T#1h1m5s と同じ） TIME#25h_3m
日付型	DATE	64 ビット	8 バイト	D#1970-01-01 ～ D#2106-02-06 （1970 年 1 月 1 日～ 2106 年 2 月 6 日）	先頭に 'DATE#' または 'date#'、または 'D#' または 'd#' を付け、' 年 - 月 - 日 ' で記述します。 （例） d#1994-09-23
時刻型 *2	TIME_OF_DAY	64 ビット	8 バイト	TOD#00:00:00.000000000 ～ TOD#23:59:59.999999999 （0 時 0 分 0.000000000 秒 ～ 23 時 59 分 59.999999999 秒 ）	先頭に 'TIME_OF_DAY#' または 'time_of_day#'、または 'TOD#' または ' tod #' を付け、' 時 : 分 : 秒 ' で記述します。 （例） tod#12:16:28.12

分類	データ型	データサイズ（容量値）	アライメント（境界値）	値の範囲	記述方法
日付時刻型 *2	DATE_AND_TIME	64 ビット	8 バイト	DT#1970-01-01-00:00:00.000000000 ～ DT#2106-02-06-23:59:59.999999999 （1970 年 1 月 1 日 0 時 0 分 0.000000000 秒～ 2106 年 2 月 6 日 23 時 59 分 59.999999999 秒 ）	先頭に 'DT#' または 'dt#' を付け、' 年 - 月 - 日 - 時 : 分 : 秒 ' で記述します。 （例） dt#1994-09-23-12:16:28.12
文字列型	STRING	（半角文字数 +1）×8 ビット*3	1 バイト	文字コード：UTF-8。 0 ～ 1986 バイト（半角英数字で 0 ～ 1985 文字、日本語で 0 ～ 661 文字が目安です）。*4 デフォルトサイズは 256 バイトです。	「'」（半角のシングルクォート）で囲んで記述します。 （例） 'OMRON'　'PLC'

*1. 持続時間型と整数型の変換には、NanoSecToTime 命令や TimeToNanoSec 命令を使用してください。命令仕様の詳細は、🕮『NY シリーズ コマンドリファレンスマニュアル 基本編（SBCA-437）』を参照してください。

*2. 比較命令での変数の比較は ns 単位で行われます。比較の単位を変更するときは、TruncTime 命令／TruncDt 命令／TruncTod 命令を使用してください。命令仕様の詳細は、🕮『NY シリーズ コマンドリファレンスマニュアル 基本編（SBCA-437）』を参照してください。

*3. 文字列の終端には NULL（1 バイト）が付きます。そのため、扱う最大文字数 +1 文字分のメモリを確保してください。たとえば、半角英数字の最大文字数が 10 の場合は、11 文字分（11 バイト）の STRING 型変数を定義してください。STRING[11]

*4. タブや改行コードなどを記載する場合は、「$」（半角ドル記号）を接頭語としたエスケープ文字を記載します。エスケープ文字については🕮「文字列型」（P.6-82）を参照してください。

使用上の注意

全変数が必要とするメモリ容量は、各変数のデータサイズ（容量値）の合計とは一致しませんので、ご注意ください。
なぜなら、データがメモリに配置される先頭位置は、自動的に各データ型の「アライメント（境界値）」の倍数の位置に配置されるので、各データ型の間に空きが発生する場合があるからです。たとえば、データ型の種類が同じでも、データ型の順によって、全体の占有サイズが異なります。

各データ型の「アライメント（境界値）」は、構造体型変数などで他の機器とデータのやり取りをする場合に、データ位置を合わせなくてはいけないときに、意識する必要があります。
詳細は、🕮「A-6 変数のメモリ確保の方法」（P.A-83）を参照してください。

参考

- ビット列のデータ型（BYTE、WORD、DWORD、LWORD）は、大小比較をすることができません。大小比較をする場合は、WORD_TO_UINT 命令などのデータ型変換命令を使用して、変数のデータ型を整数に変換してください。

  例）
  BCD_data : WORD
  IF WORD_BCD_TO _UINT（BCD_data）> UINT#1234 THEN
- 整数のデータ型（SINT、INT、DINT、LINT、USINT、UINT、UDINT、ULINT）は、論理演算をすることができません。論理演算をする場合は、INT_TO_WORD 命令などのデータ型変換命令を使用して、変数のデータ型をビット列に変換してください。

  例）
  次のサンプルプログラムでは、INT 型変数 a の値が奇数の場合に、a に 1 が加算されます。

```
IF (INT_TO_WORD（a) AND WORD#16#0001) = WORD#16#0001 THEN
    a = a+1;
END_IF;
```

● 派生データ型の種類

派生データ型は、その仕様をユーザが定義するデータ型です。Sysmac Studio の「データ型」ビューに登録されます。派生データ型には、以下のものがあります。

種類	内容
構造体型	複数のデータ型を階層的にひとつにまとめたデータ型です。
共用体型	同じデータを、場合によって異なるデータ型として扱えるようにしたデータ型です。
列挙型	あらかじめ用意された名前リスト内のひとつを値とするデータ型です。

詳細は、後述の 🕮「6-3-6 派生データ型」（P.6-40）を参照してください。

● データ型に対する指定

各データ型に対して、以下の配列指定、範囲指定が可能です。

種類	内容
配列指定	同じデータ型の要素をまとめて、先頭からの番号（添え字）で指定します。 基本データ型および派生データ型に指定できます。
範囲指定	あらかじめ決められた範囲内の値をとることを明示した指定です。 基本データ型の整数型に指定ができます。

詳細は、後述の 🕮「6-3-7 データ型の配列／範囲指定」（P.6-50）を参照してください。

参考

データ型には、基本データ型と派生データ型以外に、POU インスタンス型があります。POU インスタンス型は、FB のインスタンスのデータ型です。FB のインスタンスを生成するには、インスタンス名を変数として、FB 定義名をデータ型として、変数テーブルに登録します。

データ型の使用制限

各 POU で使用できないデータ型は、以下のとおりです。

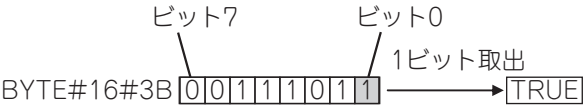
POU 種類	変数種類	使用できないデータ型		
		基本データ型	派生データ型	POU インスタンス型
プログラム	内部変数	なし		
	グローバル変数	なし		POU インスタンス型
FUN	入力変数、出力変数、入出力変数	なし	共用体型	POU インスタンス型
	内部変数	なし		POU インスタンス型
	戻り値	なし	構造体型、共用体型	POU インスタンス型
FB	入力変数、出力変数、入出力変数	なし	共用体型	POU インスタンス型
	内部変数	なし		

ビット列型、実数型、文字列型のデータフォーマット

以下に、ビット列型、実数型、文字列型のデータフォーマットを説明します。

● ビット列型のデータフォーマット

ビット列型変数のビット位置は、最下位けたを、「ビット 0」とします。
また、ビット値は 1 または 0 で表現しますが、1 ビットの値を BOOL 型変数として表現する場合は 1 が TRUE、0 が FALSE になります。



● 実数型（REAL 型、LREAL 型）

実数（REAL 型、LREAL 型）は、実数型のデータフォーマットを持つデータ型です。
その表現方法や演算処理について、説明します。

データサイズ

REAL 型のデータサイズは 32 ビット、LREAL 型のデータサイズは 64 ビットです。

データフォーマット

浮動小数点形式とは、実数を符号、指数、仮数によって表現する形式のことです。
ある実数を、下の式にあてはめて表現した場合の、s の値が符号、e の値が指数、f の値が仮数です。

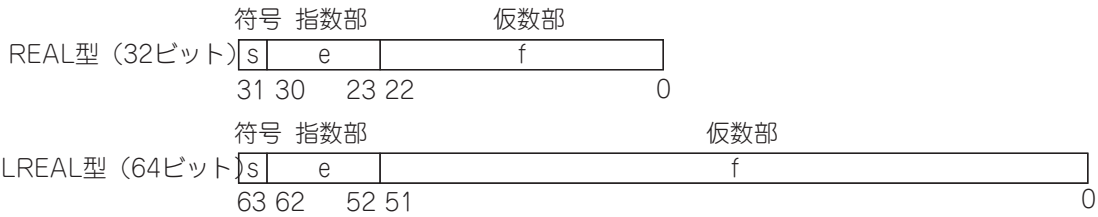
- REAL 型

  数値 = $(-1)^{s}2^{e-127}(1+f \times 2^{-23})$

- LREAL 型

  数値 = $(-1)^{s}2^{e-1023}(1+f \times 2^{-52})$

浮動小数点形式のデータフォーマットは、IEEE754 規格に準拠しています。それぞれを、以下に示します。



（例）REAL 型で、-86.625 を表現する場合

1 **負数なので、s=1 になります。**

2 **86.625 を 2 進数表現すると、1010110.101 になります。**

3 **上の値を正規化すると、1.010110101× 2^{6} になります。**

4 **上式より、e-127=6 となり、e=133（2 進数では 10000101）になります。**

5 **1.010110101の小数点以下は010110101です。これに仮数部23ビットに足りない分の0を付加したものがf になります。したがって、f=01011010100000000000000 です。**

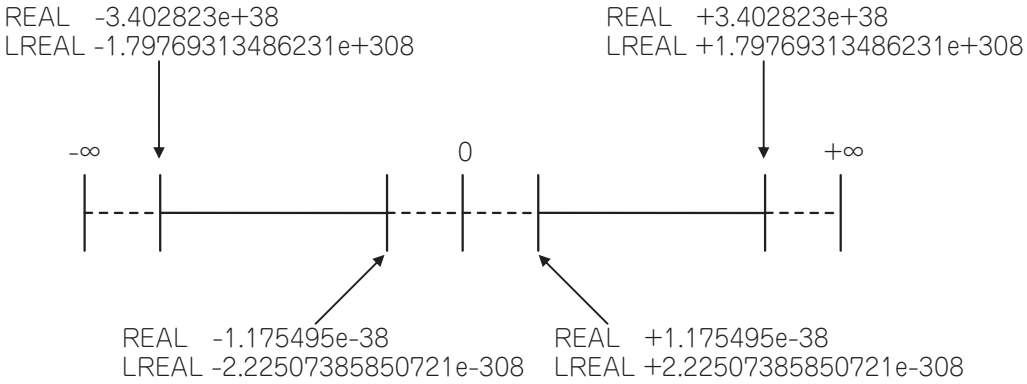
したがって、-86.625 は、下図のように表現できます。



有効範囲

REAL 型、LREAL 型の有効範囲は、下表のとおりです。
0 の近傍に、表現できない値の範囲が存在します。

データ型	- ∞	負数	0	正数	+ ∞
REAL	- ∞	-3.402823e+38 ～ -1.175495e-38	0	+1.175495e-38 ～ +3.402823e+38	+ ∞
LREAL	- ∞	-1.79769313486231e+308 ～ -2.22507385850721e-308	0	+2.22507385850721e-308 ～ +1.79769313486231e+308	+ ∞



特殊値

+ ∞、- ∞、+0、-0、非数を、特殊値と呼びます。
非数とは、浮動小数点数とは見なせないデータのことで、数値として扱われません。
+0 と -0 は、どちらも数学的には 0 と同じ意味ですが、演算処理内容に違いがあります。詳細は、後述します。
特殊値の符号 s、指数 e、仮数 f の値は、下表のとおりです。

データ型	特殊値	符号 s	指数 e	仮数 f
REAL	+ ∞	0	255	0
	- ∞	1	255	0
	+0	0	0	0
	-0	1	0	0
	非数	-	255	0 以外

データ型	特殊値	符号 s	指数 e	仮数 f
LREAL	+ ∞	0	2047	0
	- ∞	1	2047	0
	+0	0	0	0
	-0	1	0	0
	非数	-	2047	0 以外

非正規化数

浮動小数点形式は、0 の近傍（ 絶対値が非常に小さい値）を表現することができません。そのため、0 の近傍の有効範囲を拡大するために設けられたのが、非正規化数です。非正規化数は、通常のデータフォーマットで表現される値（ 正規化数と呼びます）よりも、絶対値の小さい値を表現できます。指数 e ＝ 0、かつ仮数 f ≠ 0 のものが、非正規化数と見なされ、値は以下のように表現されます。

- REAL 型

  数値＝（-1）$^{s}2^{-126}$（f × 2^{-23}）
- LREAL 型

  数値＝（-1）$^{s}2^{-1022}$（f × 2^{-52}）

（例）REAL 型で、0.75 × 2^{-127} を表現する場合

1 **正数なので、s=0 になります。**

2 **0.75 を 2 進数表現すると、0.11 になります。**

3 **$(0.11)_2 \times 2^{-127} = 2^{-126}$（$f \times 2^{-23}$）より、$f = (0.11)_2 \times 2^{22}$ になります。**

4 **上式より、f=01100000000000000000000 になります。**

したがって、0.75 × 2^{-127} は、下図のように表現できます。

	符号	指数部	仮数部
REAL型（32ビット）	0	00000000	01100000000000000000000
	31 30	23 22	0

非正規化数は、正規化数よりも有効けた数が少ないです。したがって、正規化数の演算結果が非正規化数になる場合、または演算の途中結果が非正規化数になる場合、演算結果の有効けた数が、正規化数の有効けた数未満になることがあります。

演算処理

浮動小数点形式は、近似値表現であるため、真値との間には誤差が存在する場合があります。また、値の有効範囲にも制限があります。そのため、演算の際には、丸め、オーバーフロー、アンダーフローの処理を行います。

使用上の注意

一般的に、プロセッサなどのハードウェアが異なると、実数型の演算結果が異なる場合があります。異なるコントローラ形式で、プログラムやライブラリを再利用する場合は、実数型の演算結果を十分に確認してください。

丸め

真値が、仮数の有効けた数を超えている場合、以下の規則にしたがって、値の丸めを行います。

- 浮動小数点形式で表現できる値の中で、最も真値に近い値を演算結果とします。
- 浮動小数点形式で表現できる値の中で、最も真値に近い値が 2 つある場合（ 真値が、2 つの近似値の中央値の場合）、仮数の最下位けたのビット値が 0 である方を、演算結果とします。

使用上の注意

2 つの値が等しいかどうかを検証する際の、真値との誤差の配慮
実数は浮動小数点形式で表現するため、真値との間に誤差が存在する場合があります。2 つの値が等しいかどうかを検証する場合、この誤差のために意図どおりの結果が得られない可能性があります。
たとえば、0.1+0.2 と 0.3 との比較 boolv := (0.1 + 0.2 = 0.3); を実行した場合、BOOL 型変数 boolv の値は TRUE にはならず、FALSE になります。
このような事態を避けるため、実数の 2 つの値が等しいかどうかを検証する際には、EQ、=、NE、<> を使用しないでください。その代わり、大小比較の命令を使用して、両者の差の絶対値が、十分に小さい許容範囲未満どうかを検証してください。
たとえば、REAL 型変数 real_a と real_b の和と、0.3 が等しいかどうかを検証する場合は、次のように記述してください。boolv の値が TRUE になれば、両者は等しいと判断します。

```
boolv := (ABS((real_a + real_b) - 0.3) < 0.000001);   // 0.000001 までの誤差を
                                                      // 許容範囲とする場合
```

オーバーフロー、アンダーフロー
真値の絶対値が、浮動小数点形式で表現できる値より大きくなることを、オーバーフローといいます。
逆に、浮動小数点形式で表現できる値より小さくなることを、アンダーフローといいます。
- オーバーフローの場合、真値の符号が正なら、演算結果を + ∞にします。真値の符号が負なら、演算結果を - ∞にします。
- アンダーフローの場合、真値の符号が正なら、演算結果を +0 にします。真値の符号が負なら、演算結果を -0 にします。

特殊値の演算

特殊値（+ ∞、- ∞、+0、-0、非数）の演算は、つぎの規則に従います。
- + ∞と - ∞の加算結果は、非数になります。
- 符号が同じ無限大どうしの減算結果は、非数になります。
- +0 と無限大、-0 と無限大の乗算結果は、非数になります。
- +0 どうしの除算結果、-0 どうしの除算結果、無限大どうしの除算結果は、非数になります。
- +0 と -0 の加算結果は、+0 になります。
- +0 どうしの減算結果、-0 どうしの減算結果は、+0 になります。
- 非数を含む四則演算の結果は、非数になります。
- 比較命令（Cmp 命令など）では、+0 と -0 は等しいと判断します。
- 比較命令では、比較対象に非数を含む場合、常に、等しくないと判断します。

● 文字列型のデータフォーマット

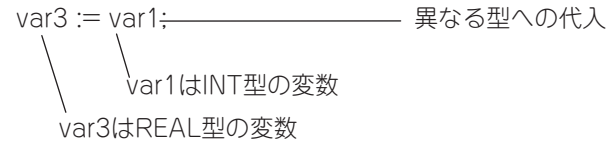
文字列型変数の終端には、終端を示す文字として、NULL 文字（ 文字コード BYTE#16#00）が付加されます。

データ型の変換

異なるデータ型の変数を使用した場合、データ型の変換が自動的に発生することがあります。または、ユーザが型変換命令によって変換することもできます。

データ型の変換とは

すべての変数は「データ型」を持つことから、プログラムではデータ型にしたがった操作を行うことが必要です。
たとえば、代入式において左辺と右辺は同じデータ型を持つべきです。しかし実際には、機能の実現のために異なるデータ型の変数への代入が必要となる場合があります。
例）



このような場合に var1 のデータを var3 のデータ型に適合させるための変換が必要になります。このような変換のことを「データ型の変換」、または略して「型変換」と呼びます。

● データ型の変換が発生する場合

データ型の変換は、以下の 2 つの場合に行います。

(1) データ型変換命令を使用することでユーザが明示的に変換

(2) 代入／命令による自動的な変換

- ST 言語の代入
- ラダー図言語における結線

参考

整数型と TIME 型の変換には、時間 / 時刻命令の NanoSecToTime 命令や TimeToNanoSec 命令を使用してください。詳細は、🕮『NY シリーズ コマンドリファレンスマニュアル 基本編 (SBCA-437)』を参照してください。

6-3-6 派生データ型

派生データ型は、基本データ型をもとに構成を定義したデータ型です。
派生データ型には、以下のものがあります。

- 構造体型
- 共用体型
- 列挙型

派生データ型を作成するとき、データ型名の文字数等の制約は、「6-3-12 変数名などのプログラム関連の名称の制限」（P.6-77）を参照してください。

参考

NY シリーズコントローラは、システム定義の派生データ型として、以下の 3 種類があらかじめ用意されています。

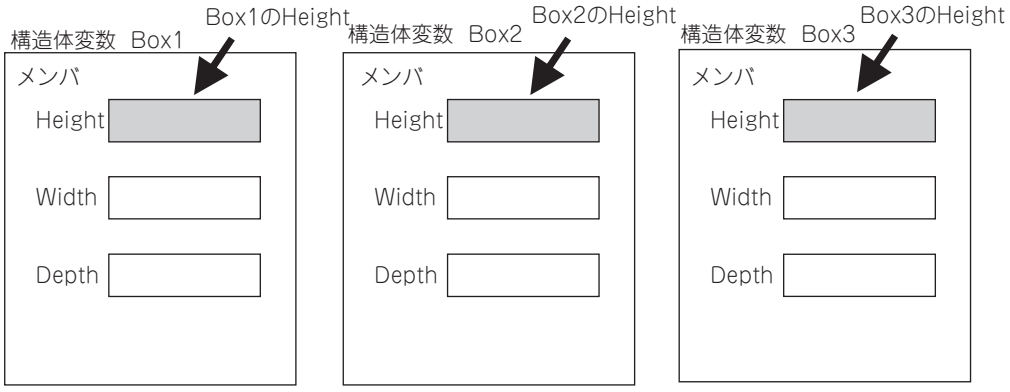
- システム定義変数の構造体型
- 命令の入力／出力／入出力変数が使用する構造体型などの派生データ型
- 高機能ユニットの拡張メモリ用の構造体型（ただし、ユニットエディタ上で登録することで使用可能）

構造体型

構造体型は、データ型が同じまたは異なる複数のデータをひとつにまとめて扱う派生データ型です。
データを構造体型にすることで、データの登録・変更などの管理を簡単にすることができます。

たとえば、『Box』という構造体型を定義する場合、メンバとして『Width』、『Height』、『Depth』の 3 つのデータをグループ化して定義します。
この構造体型のデータ型を使用して、『Box1』という変数を登録すると、プログラムでは、『Box1. Width』、『Box1. Height』のように「.（ピリオド）」を使用してデータを階層的に指定できるようになります。

もし新しい別の箱のデータが必要なときは、『Box1』と同じ要領で、『Box2』を変数テーブルに登録するだけです。



使用する命令の変数にて構造体型が使用されている場合、入力、出力または入出力パラメータ用に、その構造体型を選択して変数を登録することが必要です。
例）通信命令

● 構造体型変数の表現

各メンバの指定

構造体型を構成するデータのことを、「メンバ」と呼びます。
構造体型のメンバは、構造体型全体を表す変数名に、半角ピリオド　.　とメンバ名を付けて書き表します。
また、構造体型のメンバが、さらに構造体型であってもかまいません。

例）　abc.x　　構造体型変数 abc のメンバ x
　　abc.Order.z　構造体型変数 abc のメンバ（構造体型）Order のメンバ z

全メンバ（構造体全体）の指定

構造体を構成する全メンバを指して、「構造体全体」と呼びます。
構造体全体は、構造体の変数名だけで書き表します。
上記の例では、「abc」と書き表します。

● 構造体型の作成方法

1 **データ型テーブルにて、構造体型のデータ型を作成します。**
データ型名、メンバ、そのデータ型を指定します。

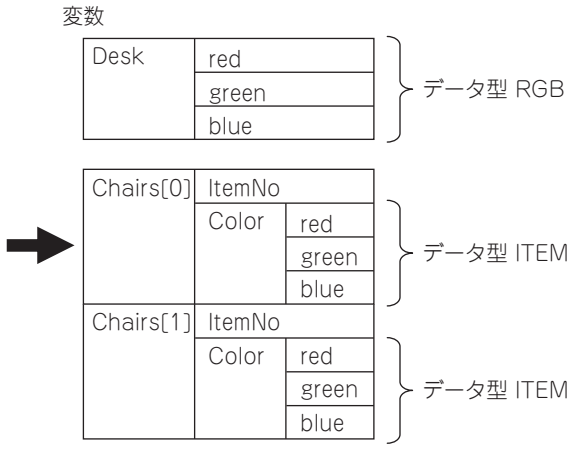
データ型テーブル

名称	メンバ	データ型
RGB	red	INT
	green	INT
	blue	INT
ITEM	ItemNo	INT
	Color	RGB

2 **変数テーブルにて、データ型として上記の構造体型データ型、メンバ名を指定して、変数を登録します。**

変数テーブル

変数名	データ型
Desk	RGB
Chairs	ARRAY[0..1]OF ITEM



● 構造体型の仕様

構造体型の仕様は以下のとおりです。

項目	仕様
構造体型名	アルファベットの大文字／小文字は区別しません。 禁則文字や文字数の制限は変数名と同じです。
メンバのデータ型	後述する「構造体のメンバのデータ型」の表を参照してください。
メンバの属性	メンバ名 コメント
メンバ数	1 ～ 2048
構造体型の階層の深さ	最大 8 階層 （ただし、メンバ名を含む変数名が 511 バイト以下）
1 構造体変数の最大サイズ	8MB

構造体型のメンバのデータ型

分類	データ型	使用
基本データ型	ブール型、ビット列型、整数型、実数型、持続時間型、日付型、時刻型、日付時刻型、文字列型	可能
	ブール型、ビット列型、整数型、実数型、持続時間型、日付型、時刻型、日付時刻型、文字列型の配列指定	可能
派生データ型	構造体型（*）、共用体型、列挙型 * 再帰または循環するものは不可です（プログラムチェック時にエラーとなります）。	可能
	構造体型、共用体型、列挙型の配列指定	可能
POU インスタンス型		不可

● 配列指定と構造体型

構造体型を要素とする配列指定を設定することも可能です。また、配列指定をメンバとする構造体型を設定することも可能です。

● 構造体のメンバのオフセット指定

メンバのオフセット指定により、構造体型ごとにメンバのメモリ配置を任意に設定できます。これにより、CJ シリーズ CPU ユニットや外部機器とタグデータリンクする場合に、両者の構造体型のメンバのメモリ配置を一致させることができます。

構造体型のメンバのオフセット種別は、「NJ」、「CJ」、「任意」のうちから選択できます。「NJ」を選択した場合は、自動的に NY シリーズに最適なメモリ配置になります。NY シリーズのメモリ配置については、🕮「A-6 変数のメモリ確保の方法」（P.A-83）を参照してください。CJ シリーズ CPU ユニットとタグデータリンクする例については、🕮「A-6-2 留意するべきケース例」（P.A-93）を参照してください。

オフセット種別の意味は、以下のとおりです。

オフセット種別	意味
NJ	自動的に、NY シリーズに最適なメモリ配置になり、最も高速に動作します。
CJ	自動的に、CJ シリーズと同じメモリ配置になります。 CJ シリーズ CPU ユニットのメモリ配置に合わせることができます。
任意	各メンバのメモリ配置を、任意に設定できます。 CJ シリーズ以外の外部機器とメモリ配置を合わせる場合に使用します。

オフセットの設定

オフセット種別を「任意」にした場合、構造体型のメンバごとにメモリ配置のオフセットが設定できます。オフセットには、「バイトオフセット」と「ビットオフセット」があります。オフセット種別を「NJ」、「CJ」にした場合は、自動的にメモリ配置が決定されるため、オフセットの設定は不要です。

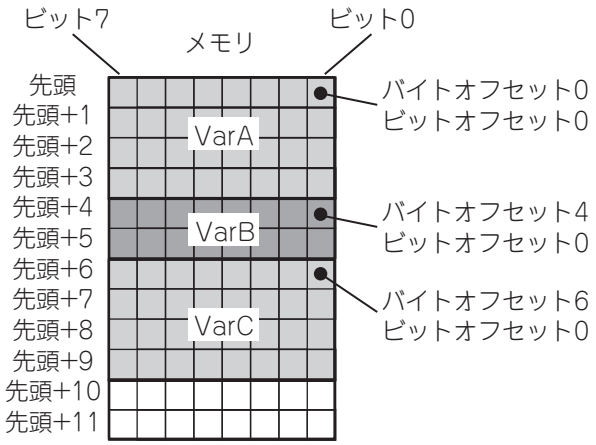
オフセットの意味は、以下のとおりです。

オフセット	意味	単位	値の範囲
バイトオフセット	各メンバの、構造体型の先頭からのオフセットを設定します。基本データ型と派生データ型のすべてのデータ型で使用できます。	バイト	0 ～ 1023
ビットオフセット	各メンバの、バイトオフセットで設定したバイト位置の先頭からのオフセットを設定します。	ビット	0 ～ 63

例）

構造体型エディタの設定が以下の場合のメモリ配置を示します。

名称	データ型	オフセット種別	バイトオフセット	ビットオフセット
StrA	STRUCT	任意		
VarA	DINT		0	0
VarB	INT		4	0
VarC	DINT		6	0



オフセットの設定可否

オフセット種別を「任意」にした場合でも、データ型によってオフセットが固定のものがあります。オフセットの設定可否を、以下に示します。

分類	データ型	バイトオフセット	ビットオフセット
ブール型	BOOL	設定可	設定可
ビット列型	BYTE,WORD,DWORD,LWORD	設定可	固定
整数型	SINT,INT,DINT,LINT,USINT,UINT,UDINT,ULINT	設定可	固定
実数型	REAL,LREAL	設定可	固定
持続時間型	TIME	設定可	固定
日付型	DATE	設定可	固定
時刻型	TIME_OF_DAY	設定可	固定
日付時刻型	DATE_AND_TIME	設定可	固定
文字列型	STRING	設定可	固定
配列指定		設定可	要素のデータ型が BOOL の場合だけ設定可
構造体型		設定可	固定
共用体型		設定可	固定
列挙型		設定可	固定
POU インスタンス型		固定	固定

メンバのオフセット指定の制約

メンバのオフセット指定をする場合は、以下の制約があります。メンバのオフセット指定をした構造体型を、メンバに含む構造体型も同じ制約があります。

- オフセット種別を「任意」にした場合、その構造体型のすべてのメンバに対してオフセットの設定が必要です。
- オフセット指定した構造体型のメンバは、初期値を任意の値に設定することができません。初期値は、データ型ごとにデフォルトで決められた値になります。詳細は、🕮「初期値指定が空白のとき」（P.6-61）を参照してください。
- 構造体型のメモリサイズは、各メンバのサイズ、データ型のアライメント（境界値）とメモリ配置によって決定されます。

メンバのオフセット指定に関係するエラー

メンバのオフセット指定に関係するエラーには、以下のものがあります。

エラー名	意味	オフセット種別	対処方法
オフセット範囲外エラー	オフセットに、範囲外の値が設定されています。	任意	オフセットの値を適切に変更してください。
オフセット未設定エラー	オフセットが設定されていないメンバがあります。	任意	すべてのメンバにオフセットを設定してください。
メモリ配置重複エラー	メンバのメモリ配置が、重複しています。	任意	オフセットの値を適切に変更してください。
初期値設定エラー	変数テーブル作成時に、オフセット指定されている構造体型のメンバに、初期値を設定しようとしています。	CJ、任意	初期値を設定しないでください。

● 構造体全体をパラメータとする命令

命令の中には、構造体変数をパラメータとして渡せるものがあります。その場合、入力パラメータとして、構造体全体だけを指定します。
例）MOVE 命令に、構造体型メンバを 1 個だけ渡す場合と、構造体変数全体を渡す場合

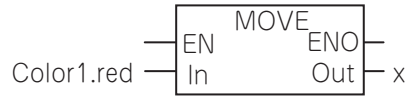
データ型テーブル

名称	メンバ	データ型
RGB	red	UINT
	green	UINT
	blue	UINT

変数テーブル

変数名	データ型
Color1	RGB

■構造体型メンバを1 個だけ指定



RGBの構造体型変数Color1のメンバredだけをxに転送する

■構造体型変数全体を指定



RGBの構造体型変数 Color1全体を xに転送する

特定の命令の、システム定義の構造体型入力変数への値の渡し方

命令によっては、入力変数にあらかじめ用意された構造体型の変数を使用している、特定の命令があります。
例）シリアル通信命令の入力変数「Port」（相手先ポート指定）は、データ型名「_sPORT」の構造体。
その特定の命令を使用する場合、以下の手順で構造体型のユーザ定義変数を作成して、入力パラメータに指定してください。

1 **Sysmac Studio には、命令用のシステム定義のデータ型が、あらかじめ登録されています。**
Sysmac Studio にて、そのシステム定義のデータ型を選択して、ユーザ定義の構造体型変数を変数テーブルへ登録します。

2 **その構造体型変数の各メンバの値を、ユーザプログラムにより、または初期値を利用して、設定します。**

3 **その構造体型変数を命令の入力パラメータに指定します。**

共用体型

共用体型は、同一のデータに対して、複数の異なるデータ型でアクセスできるようにした派生データ型です。
アクセス可能な複数の異なるデータ型として、たとえば、要素数 16 の BOOL 型の配列指定や 16 個の BOOL 型と、WORD 型を指定します。

● 共用体型の表現

各メンバの指定

共用体型を定義するとき、アクセス可能な各データ型に名前を付けます。この名前のことを、「メンバ」と呼びます。

共用体型のメンバは、共用体型全体を表す変数名に、半角ピリオド　.　とメンバ名を付けて書き表します。

例）
共用体型のデータ型としてたとえば「My Union」を以下のように定義

データ型定義：

名称	メンバ	データ型
My Union	data	WORD
	bit	ARRAY [0..15] OF BOOL

変数テーブル：

変数名	データ型
Output	My Union

としたとき

Output.bit[0]：　共用体型 Output を 16 ビットの BOOL 型の配列指定の変数と見なした場合の要素番号 0 つまり 0 ビットめの値
Output.data：　共用体型 Output を 1 個の WORD 型変数と見なした場合の値

全メンバ（共用体全体）の指定

共用体を構成する全メンバを指して、「共用体全体」と呼びます。
共用体全体は、共用体の変数名だけで書き表します。
上記の例では、「Output」と書き表します。

● 共用体型の作成方法

1 **共用体型テーブルにて、共用体型のデータ型を作成します。**
データ型名、および複数の異なるデータ型をそのメンバとして指定します。

2 **変数テーブルにて、データ型として上記の共用体型を指定して、変数を登録します。**

例）
共用体型のデータ型として、OUT16_ACCESS を定義します。
この共用体型のメンバは、要素数 16 の BOOL 型の配列指定、および WORD 型とします。
データ型 OUT16_ACCESS を使用した変数 Output を登録します。
変数 Output には、16 ビット中の任意の 1 ビットに BOOL 型の値を読み書きすることも、WORD 型の値を読み書きすることもできます。

データ型定義

名称	メンバ	データ型
OUT16ACCESS	BoolData	ARRAY[0..15] of BOOL
	ByteData	ARRAY[0..1] of BYTE
	WordData	WORD

変数名	データ型
Output	OUT16ACCESS

BoolData [15]	・・・	BoolData [8]	BoolData [7]	・・・	BoolData [0]
ByteData [1]			ByteData [0]		
WordData					

```
Output.WordData := WORD#16#1234;
```

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BoolData	0	0	0	1	0	0	1	0	0	0	1	1	0	1	0	0
ByteData	16#12								16#34							
WordData	16#1234															

```
Output.BoolData[11] :=TRUE;
```

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BoolData	*	*	*	*	1	*	*	*	*	*	*	*	*	*	*	*
ByteData	ByteData[1]の 4bit 目が TRUE								変化なし							
WordData	WordData の 11bit 目が TRUE															

● 共用体型の仕様

項目	仕様
メンバに指定できるデータ型	後述する「共用体型のメンバに指定できるデータ型」の表を参照してください。
メンバ数	最大 4
初期値設定	不可。必ずゼロとなります。

共用体型のメンバに指定できるデータ型

分類	データ型	使用
基本データ型	BOOL 型、ビット列型	可能
	BOOL 型、ビット列型の配列指定	可能
	その他の基本データ型	不可
派生データ型	構造体型、共用体型、列挙型の配列指定	不可
POU インスタンス型	−	不可

● 制限事項

- 共用体の初期値はゼロ固定となります。
- 共用体全体の転送はできません。
- POU のパラメータに共用体全体を指定できません。

列挙型（ENUM）

列挙型は、変数の値を「列挙子」と呼ぶ文字で表現した派生データ型です。
あらかじめその変数の取り得る値を、列挙子（文字）として設定しておきます。
列挙型を使うと、変数の値の意味が、人にとって理解しやすくなります。

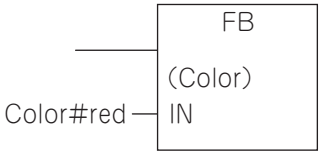
● 列挙型の表現

列挙型を定義するとき、変数が取り得る値に、列挙子と呼ぶ名前を付けます。

● 列挙型の作成方法

1 列挙型テーブルにて、列挙型のデータ型を作成します。
取り得る列挙子と対応する値を設定します。

2 変数テーブルにて、データ型として上記の列挙型を指定して、変数を登録します。
例）
列挙型のデータ型として、Color を作成します。取り得る列挙子は、「red」、「yellow」、「green」の 3 つとします。列挙子に対応する数値は、red=0、yellow=1、green=2 とします。
データ型 Color を使用した変数 DiscColor を登録します。変数 DiscColor の取り得る値は、red(0)、yellow(1)、green(2) のいずれかになります。



列挙型テーブル

データ型	
Color	ENUM
列挙子	値
red	0
yellow	1
green	2

変数テーブル

変数名	データ型
DiscColor	Color

● 列挙型の仕様

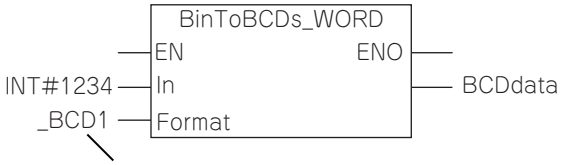
項目	仕様
列挙子	半角英数字だけとし、アルファベットの大文字／小文字を区別しません。禁則文字は変数名と同じです。 同じ列挙子を複数指定した場合はビルド時にエラーとなります。 ユーザプログラム内にある変数と同じ列挙子を指定した場合や、他の列挙型に存在する列挙子と同じ名前を指定した場合もビルド時にエラーとなります。
値	有効範囲は、-2147483648 ～ +2147483647 の整数値です。 連続値でなくてもかまいません。 同じ値を複数指定した場合はビルド時にエラーとなります。 注 . 列挙型の変数の大小比較はできません。一致／不一致だけが比較できます。
列挙子の数	1 ～ 2048 個

● 列挙子を FB、FUN のパラメータとする場合の記述

列挙子を FB、FUN のパラメータとして渡す場合、以下の 2 つの記述方法があります。

列挙子だけの記述

パラメータが列挙型であることが指定されている FB、FUN の場合、列挙子だけの記述が可能です。
例）BCDToBins_WORD 命令に、列挙子を渡す場合



BinToBCDs_WORD命令の入力変数Formatは、列挙型の指定があります。
したがって、列挙子だけの形式で記述できます。

列挙型 # 列挙子の記述

パラメータのデータ型に特に指定がない FB、FUN の場合、列挙子だけの記述は無効です。ビルド時にエラーになります。パラメータが列挙子であることを明示するため、列挙型 # 列挙子の形式で記述します。

例）MOVE 命令に、列挙子を渡す場合



MOVE命令の入力変数Inは、データ型の指定がありません。
列挙子を渡す場合は、列挙型#列挙子の形式で記述します。

参考

パラメータが列挙型であることが指定されている FB、FUN の場合に、パラメータを列挙型 # 列挙子の形式で記述してもかまいません。
したがって、上記の BinToBCDs_WORD 命令の場合、Format 変数に渡すパラメータを、_eBCD_FORMAT# _BCD1 と記述してもかまいません。

● 値のチェック

命令実行によって、列挙型変数に値が書き込まれる際、その値が列挙型の列挙子として定義されたものでなくても、エラーにはなりません。したがって、値が列挙型の列挙子として定義されたものかどうかをチェックする必要がある場合は、ユーザプログラムの中で、値をチェックする処理を行ってください。

6-3-7 データ型の配列／範囲指定

各データ型の変数には、以下の属性を指定することができます。

- 配列指定
  - 固定長配列指定
  - 可変長配列指定
- 範囲指定

配列指定（ARRAY[]OF）

配列指定は、同じ属性のデータをまとめて、ひとつのかたまりとして扱うデータ型に対する指定です。
基本データ型、派生データ型に対して指定可能です。
モーション制御の際の座標値のように、同じ概念のデータを複数扱う場合に便利です。

● 配列指定の表現

各要素の指定

配列指定を構成する各データのことを、「要素」と呼びます。
配列指定の要素は、配列指定全体を表す配列指定の変数名に、先頭から数えた添え字（要素番号）を付けて書き表します。
添え字は半角の角かっこ [] で囲みます。添え字は、定数でも変数でもかまいません。また、ST 言語の場合は、添え字を式で表現することも可能です。

例）

変数テーブル

変数名	データ型
Mem	ARRAY[0..99] OF INT

```
x:=10;
Mem[x]      配列指定の変数 Mem の要素番号 x（変数 x の値は 10 なので、要素番号 10）
```

変数テーブル

変数名	データ型
Data	ARRAY[0..99] OF INT

```
x:=10;
y:=20;
Data[x+y]  配列指定の変数 Data の要素番号（x+y）（変数 x の値は 10、
           変数 y の値は 20 なので、要素番号 30）
```

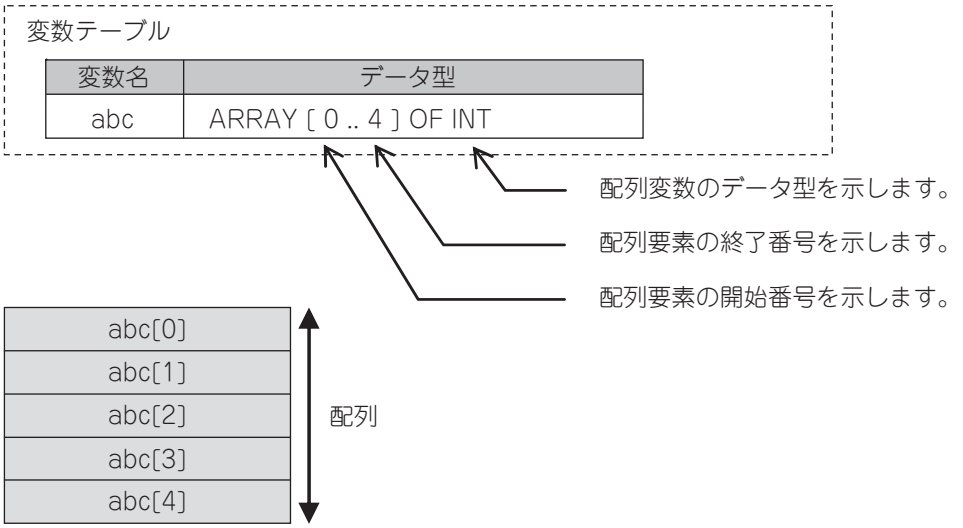
全要素（配列全体）の指定

一つの配列指定を構成する全要素を指して、「配列全体」と呼びます。
配列全体は、配列指定の変数名だけで書き表します。
上記の例では「Mem」、「Data」と書き表します。

● **配列指定の作成方法**

1 **変数テーブルのデータ型の欄にて”A“と入力して、候補となるデータ型名から「ARRAY[?..?] OF ?」を選択します。**

2 **[?..?]部の左の？に要素の開始番号、右の？に終了番号、OF ？の？部にデータ型を入力して、変数を登録します。**



参考

添え字に開始番号と終了番号を指定すると、固定長配列指定の変数となります。また、添え字に半角アスタリスク (*) を指定すると、可変長配列指定の変数となります。
可変長配列指定とする場合、固定長配列指定に対して制約事項があります。可変長配列指定の詳細については、「 可変長配列指定」（P.6-54）を参照してください。

バージョン関連情報

可変長配列指定は、コントローラのユニットバージョン Ver.1.18 以降と Sysmac Studio Ver.1.22 以降の組み合わせで使用可能です。

● 配列指定の仕様

<table>
<tr><th>項目</th><th>仕様</th></tr>
<tr><td>配列指定の変数の最大要素数</td><td>65,535</td></tr>
<tr><td>要素番号</td><td>0 ～ 65535
要素の開始番号が 0 以外でもかまいません。</td></tr>
<tr><td>添え字</td><td>定数：0 ～ 65535 の整数値
変数：
<table>
<tr><th>分類</th><th colspan="2">データ型</th><th>使用</th></tr>
<tr><td rowspan="3">基本データ型</td><td rowspan="2">整数型</td><td>SINT、INT、DINT、USINT、UINT 、UDINT</td><td>可能</td></tr>
<tr><td>LINT、ULINT</td><td>不可</td></tr>
<tr><td colspan="2">ブール型、ビット列型、実数型、持続時間型、日付型、時刻型、日付時刻型、文字列型</td><td>不可</td></tr>
<tr><td>派生データ型</td><td colspan="2">構造体型、共用体型、列挙型</td><td>不可</td></tr>
<tr><td>POU インスタンス型</td><td colspan="2"></td><td>不可</td></tr>
</table>
演算式：ST 言語の場合だけ演算式を指定可能　　例）y := x[a+b];</td></tr>
<tr><td>1 配列変数の最大サイズ</td><td>8MB</td></tr>
</table>

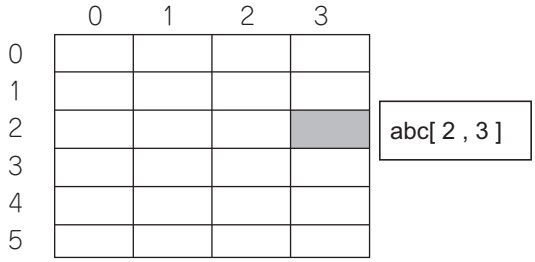
● 配列指定の次元

1 次元の配列指定は、要素が 1 列に並んだ 1 次元データと考えることができます。
同様に、2 次元や 3 次元の配列指定を設定することができます。
その場合、配列指定要素は、配列指定の変数名に次元と同じ数の添え字を付けて表現します。次元数の最大値は 3 です。

2 次元配列指定

変数テーブル

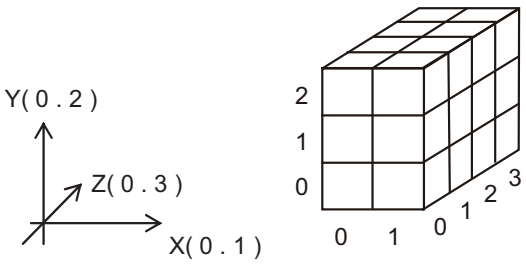
変数名	データ型
abc	ARRAY [0..5 , 0..3] OF INT



3 次元配列指定

変数テーブル

変数名	データ型
ITEM	ARRAY [0..1, 0..2, 0..3] OF INT



● 配列指定と構造体型

構造体型を要素とする配列指定を設定することも可能です。
また、配列指定をメンバとする構造体型を設定することも可能です。

構造体型を要素とする配列指定

データ型テーブル

データ型	メンバ	メンバ
Str	x y	INT DINT

変数テーブル

変数名	データ型
abc	ARRAY [1..4] OF Str



変数

abc[1].x
abc[1].y
:
abc[4].x
abc[4].y

配列指定をメンバとする構造体型

データ型テーブル

データ型	メンバ	データ型
Str	x	ARRAY [0..1] OF INT
	y	DINT

変数テーブル

変数名	データ型
abc	ARRAY [1..4] OF Str



変数

abc[1].x[0]
abc[1].x[1]
abc[1].y
:
abc[4].x[0]
abc[4].x[1]
abc[4].y

● 配列指定をパラメータとする命令

命令の中には、配列全体をパラメータとして渡せるものがあります。その場合、入力パラメータとして、配列指定の変数名だけを指定します。

例）MOVE 命令に、配列指定要素 1 個だけを渡す場合と、配列全体を渡す場合

■配列 1 要素を指定



INT 型変数 x[0]を y[0]に転送する

変数テーブル

変数名	データ型
x	ARRAY [0 .. 199]OF INT
y	ARRAY [0 .. 199]OF INT

■配列の全体を指定

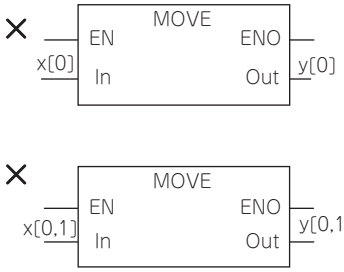


INT 型変数 x[0..199]を y[0..199]に転送する

・制限事項：
配列の全体を指定する場合は、要素番号の範囲も含めて同じデータ型である必要があります。

参考

多次元配列の部分的な配列全体は、パラメータとして指定できません。



変数テーブル

変数名	データ型
x	ARRAY [0..9 , 0..9 , 0..9]OF INT
y	ARRAY [0..9 , 0..9 , 0..9]OF INT

● 配列指定の保護機能

配列の要素数を超えてアクセスしようとした場合、以下のエラーが発生します。

添え字が定数の場合

Sysmac Studio で入力するとき、およびプログラムチェック時にエラー表示されます。

変数テーブル

変数名	データ型
x	ARRAY [1..10] OF INT

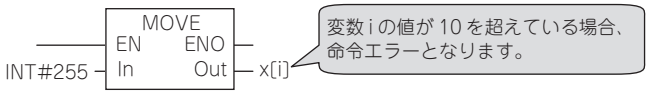


添え字が変数の場合

コントローラが、命令実行後に出力パラメータを出力変数に代入する際に、要素数超えのチェックを行います。指定された添え字変数が配列変数の要素数を超える場合は、命令エラーとして扱われます。この場合、命令の内部処理は正常終了しているので、ENO の値は TRUE になります。ご注意ください。

変数テーブル

変数名	データ型
x	ARRAY [1..10] OF INT



可変長配列指定

配列指定の変数を宣言する際、添え字に半角アスタリスク（*）を指定することで、可変長配列指定の変数となります。最大要素数、次元、および宣言可能なデータ型などは、固定長配列の仕様と同じです。ただし、変数宣言での制約事項、および演算での制約事項があります。

● 変数宣言での制約事項

可変長配列指定の変数を宣言できるのは、ファンクション（FUN）の入出力変数およびファンクションブロック（FB）の入出力変数だけです。その他の変数では宣言できません。

宣言する配列変数	可／不可
FUN の入出力変数	可
FB の入出力変数	可
FUN の入力変数／出力変数	不可
FB の入力変数／出力変数	不可
内部変数、外部変数、グローバル変数	不可
構造体型変数のメンバ／共用体型変数のメンバ	不可

2 次元や 3 次元で可変長配列指定の変数を宣言する場合、すべての次元に半角アスタリスク (*) を指定する必要があります。可変長の次元と固定長の次元が混在した変数は宣言できません。

宣言する配列変数	指定例	可／不可
すべての次元が可変長配列	• ARRAY[*] • ARRAY[*, *] • ARRAY[*, *, *]	可
可変長の次元と固定長の次元が混在	• ARRAY[*, 3..5] • ARRAY[*, *, 3..5]	不可
開始番号または終了番号への個別指定	• ARRAY[1..*] • ARRAY[*..*]	不可

● 演算での制約事項

ファンクション (FUN) およびファンクションブロック (FB) の入出力変数が可変長配列の場合、指定できる入出力パラメータは、データ型および次元数が同じ配列変数です。データ型と次元数が同じであれば、可変長配列でも固定長配列でもかまいません。データ型または次元数の異なる配列変数を指定することはできません。
ファンクション (FUN) およびファンクションブロック (FB) の入出力変数が固定長配列の場合、入出力パラメータに可変長配列変数を指定することはできません。

FUN/FB の入出力変数の例	指定する入出力パラメータの例	可／不可
データ型が INT、次元数が 1 の可変長配列	データ型が INT、次元数が 1 の可変長配列	可
データ型が INT、次元数が 1 の可変長配列	データ型が INT、次元数が 1 の固定長配列	可
データ型が INT、次元数が 1 の可変長配列	データ型が UINT、次元数が 1 の可変長配列	不可
データ型が INT、次元数が 1 の可変長配列	データ型が INT、次元数が 2 の可変長配列	不可
データ型が INT、次元数が 1 の固定長配列	可変長配列	不可

ST 言語で配列全体の代入式を記述する場合、左辺または右辺のいずれか片方にでも可変長配列変数が含まれると、ビルド時にエラーとなります。
同様に、ラダー図言語の MOVE（転送）命令で配列全体を転送する場合、出力パラメータまたは入力パラメータのいずれか片方にでも可変長配列変数が含まれると、ビルド時にエラーとなります。

左辺／出力パラメータ	右辺／入力パラメータ	ST 言語での記述例[*1]	可／不可
可変長配列変数	可変長配列変数	varlen_ary := varlen_ary;	不可
可変長配列変数	固定長配列変数	varlen_ary := fixed_ary;	不可
固定長配列変数	可変長配列変数	fixed_ary := varlen_ary;	不可
固定長配列変数	固定長配列変数	fixed_ary := fixed_ary;	可

*1.「varlen_ary」は可変長配列変数、「fixed_ary」は固定長配列変数を表します。

● 可変長配列変数の要素の開始番号／終了番号

ユーザプログラムを実行すると、要素の開始番号と終了番号が定まります。LOWER_BOUND 命令および UPPER_BOUND 命令を使用することで、可変長配列変数の次元ごとの開始番号および終了番号を取得できます。

LOWER_BOUND 命令および UPPER_BOUND 命令の詳細は、🕮『NY シリーズ コマンドリファレンスマニュアル 基本編（SBCA-437)』を参照してください。

範囲指定（(..)）

範囲指定は、以下の整数型の変数に対して、その値の範囲を限定することを明示する指定です。

分類	データ型
整数型	SINT、INT、DINT、LINT、USINT、UINT、UDINT、ULINT

以下のときに、入力した値が許容範囲中にあるかどうかを判断することができます。

- 変数の初期値を指定するとき
- CIP メッセージ通信で値を書き込んだとき

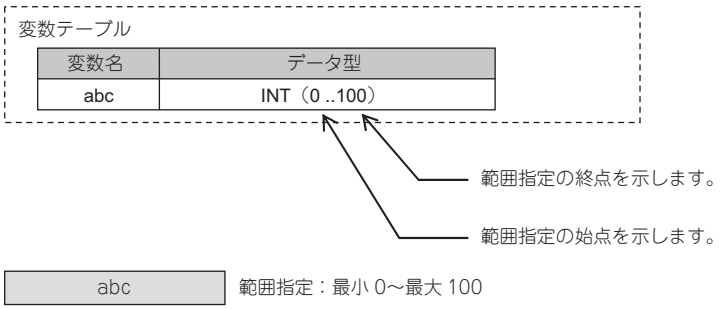
● 範囲指定の作成方法

変数テーブルのデータ型の欄にて、データ型名の後に、（始点 .. 終点）を入力します。

始点：　変数に格納可能な最小値

終点：　変数に格納可能な最大値

例）



● 範囲指定の仕様

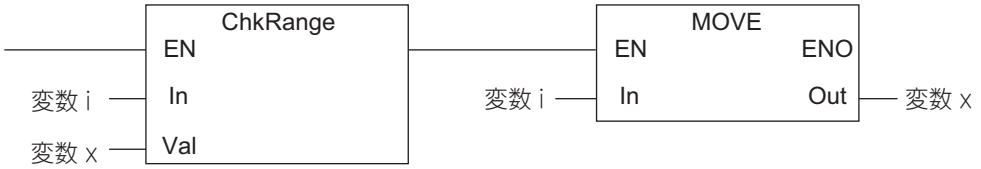
項目	仕様			
指定できるデータ型	整数型			
範囲外の値を書き込もうとしたときの動作	場合		動作	
	ユーザプログラム		エラーとならず、値は書き換わります。 命令実行によって変数の値が変化するとき、コントローラは範囲チェック行いません。	
	通信	Sysmac Studio からの書込、CIP メッセージ	整数型のとき	コマンドエラーとなります。
			整数型配列変数の要素のとき	
			整数型の構造体型のメンバのとき	
			整数型の構造体全体のとき	コマンドエラーとならず、値は書き換わります。
			整数型配列全体のとき	
		タグデータリンク（内蔵 EtherNet/IP、EtherNet/IP ユニットとも）	範囲を指定したメンバに対する単独の書込はエラーとなります。 範囲を指定したメンバを含めた構造体全体への書込はエラーとなりません。	
	スレーブ／ユニットからの IN リフレッシュ		エラーとならず、値は書き換わります。	
	強制値リフレッシュ機能による強制値		エラーとならず、値は書き換わります。	

使用上の注意

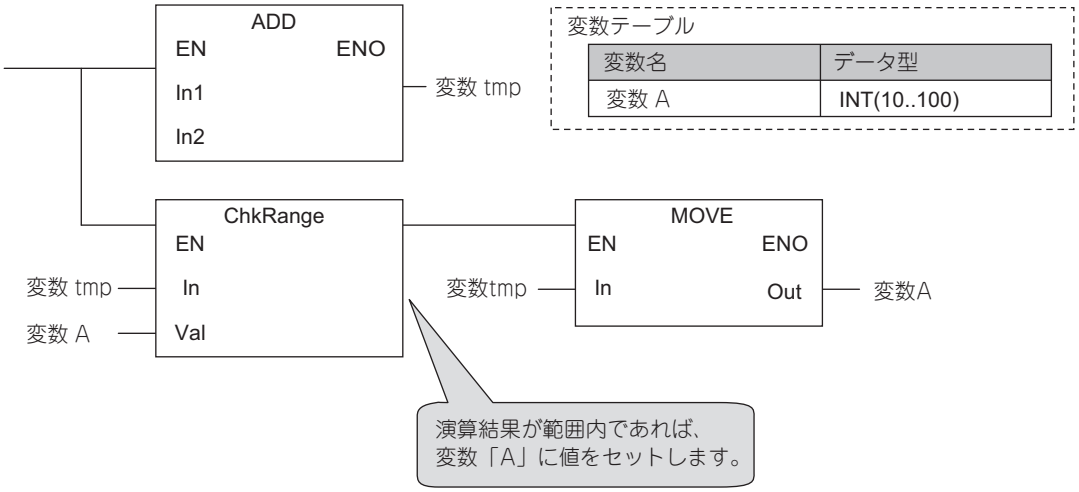
変数の範囲指定は、ユーザプログラムによる命令の実行によって変数の値が変化した場合まで、チェックするものではありません。ユーザプログラム実行によって変化した変数の値の範囲をチェックしたい場合、以下のように、範囲をチェックする命令を使用してください。

変数テーブル

変数名	データ型
変数 x	INT(10..100)



なお、四則演算結果などをセットする場合は事前チェックができないため、以下のように、四則演算（例：ADD）後の値の範囲をチェックしてください。



範囲指定を使用する場合、初期値にご注意ください。
Sysmac Studio の初期値の列が空白のとき、初期値は 0 となります。0 を含まない範囲を有効範囲とした場合でも同様に 0 となります。

6-3-8 変数の各属性の説明

データ型以外の、変数の各属性を以下に説明します。

変数名

変数名は、その変数を識別するための名称です。
ひとつの POU 内で、同じ変数名の変数を複数宣言することはできません。ただし、異なる POU で、同じ変数名のローカル変数を宣言することは可能であり、それらは異なる変数として扱われます。

変数名の制約は、「6-3-12 変数名などのプログラム関連の名称の制限」（P.6-77）を参照してください。

AT（割付先）指定

この属性は、デバイス変数を NX ユニットや EtherCAT スレーブの I/O ポートに特定するために使用します。変数を I/O ポートに対して割り付けることで、自動で内部的に AT（割付先）指定されます。

● AT（割付先）指定の表示

AT（割付先）指定は、変数テーブルおよび I/O マップの「割付先」の列に、以下で表示されます。

変数種類	「割付先」列での表示	例
EtherCAT スレーブデバイス変数	EtherCAT スレーブターミナル上の NX ユニットの場合： ECAT://node#[ノードアドレス .NX ユニット番号]/ [I/O ポート名]	ECAT://node#[10.15]/Input1
	上記以外の場合： ECAT://node#[ノードアドレス]/ [I/O ポート名]	ECAT://node#1/Input1
軸変数	MC://_MC_AX[]	MC://_MC_AX[1]
軸グループ変数	MC://_MC_GRP[]	MC://_MC_GRP[1]

● AT（割付先）指定できる変数の種類

変数単位で指定します。配列変数、構造体変数、共用体変数は全体に対して指定します。

● AT（割付先）指定した変数の属性

	指定	備考
名前	可	
データ型	可	
保持	可	AT（割付先）指定先のメモリの属性に従います。 NX ユニットおよび EtherCAT スレーブの I/O ポートはすべて非保持です。
初期値	可	メモリの値をそのまま利用する場合は、「初期値指定なし（None）」を設定します。
コンスタント	可	命令による値の書込はできません。
ネットワーク公開	可	
エッジ	不可	（ファンクションブロックの入力変数に対してだけ可能）

使用上の注意

同一アドレスに、複数の変数を割り付けることも可能ですが、プログラムの可読性およびデバッグの難しさから推奨できません。この場合、初期値を設定する変数は 1 つだけにしてください。それぞれの変数に異なる初期値設定をすると、初期値が不定になります。

保持

この属性は、以下のときに変数の値を保持するかどうかを指定するために使用します。

- 電断後の電源投入時
- 動作モード切替時
- 全停止フォールトレベルのコントローラ異常発生時

本属性を設定していない場合、変数の値は、上記のときに初期値になります。

生産数など、NY シリーズコントローラの電源が切れた後も保持しておきたいデータを格納する変数は、保持ありと指定しておくと便利です。

● 保持を有効にするために必要な条件

産業用 PC に UPS を接続し、正常にシャットダウンした場合に、不揮発性メモリに保持されます。
シャットダウンの詳細については、🕮「2-8 シャットダウン処理」（P.2-29）を参照してください。

● 保持変数に初期値を反映させたいとき

ユーザプログラム転送時に「保持変数の現在値を初期化する」をチェックします。

● 保持あり／なしの動作の違い

保持に関係があるケースと、その際に変数に設定される値との関係を、以下に示します。

ケース		変数の値	
		保持あり	保持なし
電断後の電源投入時		保持される	保持されない
動作モード切替時			
全停止フォールトレベルのコントローラ異常発生時			
ユーザプログラム転送時	「保持変数を初期化しない」設定		
	「保持変数を初期化する」設定	保持されない	

● 保持指定できる変数の種類

変数単位で指定します。配列変数、構造体変数、共用体変数は全体に対して指定します。

初期値

初期値とは、以下のときに設定される変数の値です。

- 電源投入時
- 運転モード⇔プログラムモードへの切替時
- ユーザプログラム転送時で「保持変数を初期化する」設定のとき
- 全停止フォールトレベルのコントローラ異常発生時

初期値を設定しておけば、変数を初期化するプログラムを省くことができます。たとえば、レシピなどのデータを初期値としてあらかじめ設定しておくこともできます。
なお、初期値を設定しなくてもかまいません。

● 初期値を設定できる変数の種類

変数の種類によって、初期値設定できるものとできないものがあります。以下に、その一覧を示します。

変数の種類	初期値設定
グローバル変数	可
内部変数	
入力変数	
出力変数	
FUN の戻り値	不可
入出力変数	
外部変数	

● 初期値あり／なしの選択

変数作成時に初期値あり／なしを選択できます。

初期値指定あり

初期値

（空白） または

初期値
3.14

初期値指定なし

初期値
None

種類ごとの、選択の可否は以下のとおりです。

種類		例	初期値あり／なしの選択
基本データ型変数		aaa	可能
配列指定の変数	配列全体	bbb	可能
	要素	bbb[2]	不可
構造体型変数	構造体全体	ddd	可能
	メンバ	ddd.xxx	不可
共用体型変数	共用体全体	eee	不可（初期値はゼロ固定）
	メンバ	eee.word	初期値は必ずゼロ
列挙型変数		ccc	可能
POU インスタンス型		instance	不可

参考

基本 I/O ユニットの一部では、1 つの I/O ポートに対してビット列型とブール型など複数のアクセス手段を持つものがあります。本 I/O ポートに対して初期値を利用する場合は、一方を「None（初期値指定なし）」としてください。

● 初期値が設定されるタイミング

初期値設定した変数に、実際に初期値が代入されるタイミングは、以下のとおりです。

- 電源投入時
- 動作モードがプログラムから運転、運転からプログラムに切り替わった時
- 「変数を初期化する」を指定して、ユーザプログラムを転送した時
- 全停止フォールトレベルのコントローラ異常発生時

● 初期値指定が空白のとき

初期値指定が空白のときは変数の初期値は以下になります。

データ型		初期値
ブール型／ビット列型	BOOL, BYTE, WORD, DWORD, LWORD	0
整数型	SINT, INT, DINT, LINT USINT, UINT, UDINT, ULINT	0
実数型	REAL, LREAL	0.0
持続時間型・日付型・時刻型・日付時刻型	TIME	T#0S
	DATE	D#1970-01-01
	TIME_OF_DAY	TOD#00:00:00
	DATE_AND_TIME	DT#1970-01-01-00:00:00
文字列型	STRING	''（空文字）

● 配列指定の変数の初期値

データ型	初期値の仕様
配列指定	• 要素ごとに初期値設定できます。 • 初期値を指定する場合は、すべての要素に対して値をセット、または空白とする必要があります。

● 派生データ型の初期値

データ型自体に初期値を指定するのではなく、変数ごとに初期値を指定します。

データ型	初期値の仕様
構造体型	• メンバごとに初期値設定できます。 • 初期値を指定する場合は、すべての要素に対して値をセット、または空白とする必要があります。
共用体型	• 指定できません。常にゼロとなります。
列挙型	• 指定できます。

● 初期値が適用されない変数

以下の変数の場合、電源投入時に初期値が適用されず、電断直前の値が保持されます。

- 保持属性の変数

使用上の注意

産業用 PC に UPS が接続されていない場合、あるいは正常にシャットダウンされなかった場合、上記の変数の値も初期値となります。

コンスタント

コンスタントとは、命令、ST 言語の演算子、または CIP メッセージ通信によって値を書き込むことを禁止する属性です。これにより、プログラムによる、変数への値の書込を防止できます。
この属性の変数の場合、初期値がセットされた以後は、命令による書込を禁止できます。
もし POU 内にコンスタント属性ありの変数に値を書き込む命令が存在すると、ビルドの時点でエラーになります。

● 動作

POU 内にコンスタント属性ありの変数に値を書き込む命令または演算子が存在するときの動作は、以下のとおりです。

書込元		値を書き込もうとしたときの動作
ユーザプログラム		プログラムチェック時にエラーとなります。Sysmac Studio がビルド時にチェックを行い、ビルドエラーとなります。
通信	Sysmac Studio からの書込	不可
	CIP メッセージ	コマンドエラーとなります。
	タグデータリンク	タグデータリンク開始時にエラーが発生しますが、タグデータリンクは継続します。 ただし、コンスタント属性を持つ変数の値は書き換わりません。
スレーブ／ユニットからの IN リフレッシュ		エラーとならず、値は書き換わります。
強制値リフレッシュ		

● コンスタント指定できる単位

変数単位で指定します。配列変数、構造体変数、共用体変数は全体に対して指定します。

参考

「コンスタント」属性は、ユーザプログラムからの書込も禁止します。

ネットワーク公開

ネットワーク公開属性とは、CIP メッセージ通信またはタグデータリンク機能によって、その変数を、外部（他のコントローラや上位パソコンなど）からも読み書きできるようにする（これを「ネットワーク公開」と呼びます）属性です。
本属性を設定していない場合、変数は、その変数を宣言したコントローラで読み書きができ、外部（他のコントローラや上位パソコンなど）からは読み書きできません。

ネットワーク公開された変数を、「ネットワーク変数」と呼びます。

● ネットワーク公開の仕様

ネットワーク公開には、「公開のみ」「入力」「出力」の指定があります。それぞれの仕様を、以下に示します。

ネットワーク公開		仕様
公開しない		外部からはアクセスできない。 サポートソフトからは公開有無に関わらずアクセスできる。
公開する	公開のみ	CIP 通信による外部からのアクセスが可能な変数。 タグデータリンクは不可。
	入力	CIP 通信またはタグデータリンクによる、外部からのアクセスが可能な変数。 タグデータリンク時は、データの入力用（他コントローラ→自コントローラ）の変数となる。
	出力	CIP 通信またはタグデータリンクによる、外部からのアクセスが可能な変数。 タグデータリンク時は、データの出力用（自コントローラ→他コントローラ）の変数となる。

● ネットワーク公開指定できる単位

変数単位で指定します。配列変数、構造体変数、共用体変数は全体に対して指定します。

エッジ

エッジ属性とは、BOOL 型の変数の値が FALSE → TRUE に立ち上がったとき、または TRUE → FALSE に立ち下がったときに、FB に TRUE の値を渡す属性です。
FB の BOOL 型の入力変数にだけ設定可能です。

● 用途

入力パラメータの、立ち上がり／立ち下がりだけを FB 側で受け取りたい場合に使用します。
たとえば、入力パラメータに変化があったときだけ、FB を起動するときなどに使用します。

● 動作

- 「立ち上がり」を指定すると、その入力変数に接続した入力パラメータの値が FALSE から TRUE に立ち上がったときだけ、入力変数の値が TRUE になります。
- 「立ち下がり」を指定すると、入力パラメータが TRUE から FALSE に立ち下がったときだけ、入力変数の値が TRUE になります。

指定	入力パラメータの値	変数の値
立ち上がり	FALSE → TRUE	TRUE
	それ以外	FALSE
立ち下がり	TRUE → FALSE	TRUE
	それ以外	FALSE
なし	−	入力パラメータの値にあわせて都度変化します

6-3-9　状態変化に応じた変数の値の変化

ここでは、コントローラの状態変化に応じた変数の値の変化について説明します。

バージョン関連情報

コントローラのユニットバージョンVer.1.14以降とSysmac Studio Ver.1.18以降の組み合わせでは、システム定義変数「_DeviceOutHoldCfg（デバイス出力保持設定）」の設定により、動作モード変更時およびダウンロード時の動作を設定できます。

電源投入、動作モード変更、ダウンロード時に、NY シリーズコントローラ内部の変数の値は、下表のように変化します。

変数の保持属性	変数種類		電源投入時	動作モード変更時	ダウンロード後	
					「保持変数を初期化する」設定時	「保持変数を初期化しない」設定時
非保持	ユーザ定義変数		• 初期値設定あり：指定した初期値となる。 • 初期値設定なし（None）：16#00*1			
	EtherCAT スレーブデバイス変数*2		• 初期値設定あり：指定した初期値となる。 • 初期値設定なし（None）：16#00*1			
保持	ユーザ定義変数	保持条件を満たす*3	変化なし （電断前の値を保持）	変化なし （動作モード変更の値を保持）	• 初期値設定あり：指定した初期値となる。 • 初期値設定なし（None）：16#00*1	ダウンロード前の値を保持
		保持条件を満たさない*3				• 初期値設定あり：指定した初期値となる。 • 初期値設定なし（None）：16#00*1

*1. データ型により、16#00 以外の値となる場合があります。詳細は、🕮「初期値指定が空白のとき」（P.6-61）を参照してください。
*2. システム定義変数「_DeviceOutHoldCfg（デバイス出力保持設定）」で、デバイス出力保持設定を有効（16#A5A5）に設定した場合、動作モード変更時およびダウンロード時でも、デバイス出力を保持します。
デバイス出力保持設定については、🕮「デバイス出力保持設定」（P.6-65）を参照してください。
*3. 保持条件については、🕮「保持条件」（P.6-66）を参照してください。

全停止フォールトレベルのコントローラ異常発生時、オンラインエディット時に、コントローラ内部の変数の値は、下表のように変化します。

変数の保持属性	変数種類	全停止フォールトレベルのコントローラ異常発生時	オンラインエディット時	
			オンラインエディット対象内の POU に、追加した変数	オンラインエディット対象内の POU にすでに存在していた変数
非保持	ユーザ定義変数／デバイス変数	• 初期値設定あり時：指定した初期値となる • 初期値設定なし時：16#00	• 初期値設定あり時：指定した初期値となる • 初期値設定なし時：16#00	変化なし
保持	ユーザ定義変数／デバイス変数	変化なし （異常発生前の値を保持）	• 初期値設定あり時：指定した初期値となる • 初期値設定なし時：16#00	
その他	強制値リフレッシュ状態	解除	変化なし	

デバイス出力保持設定

システム定義変数「_DeviceOutHoldCfg（デバイス出力保持設定）」の設定により、動作モード変更時およびダウンロード時の動作を設定できます。

デバイス出力保持設定を無効（16#A5A5 以外）にした場合、デバイス出力は、「6-3-9 状態変化に応じた変数の値の変化」（P.6-64）に示した動作となります。
デバイス出力保持設定を有効（16#A5A5）にした場合、動作モード変更時およびダウンロード時でも、デバイス出力を保持します。

状態	デバイス出力保持設定	
	無効（16#A5A5 以外）	有効（16#A5A5）
動作モード変更時	変数を初期化する	変数の値を保持する
ダウンロード時	I/O リフレッシュを停止し、ダウンロード後に変数を初期化する	I/O リフレッシュを実行し、ダウンロード後に変数の値を保持する

● 対象となるデバイス出力

デバイス出力保持設定の対象となるデバイス出力は以下です。

- EtherCAT スレーブデバイス変数

● ダウンロード時にデバイス出力を保持できる条件

ダウンロード時にデバイス出力を保持するための条件として、変更できる項目に制限があります。変更した場合でも、デバイス出力を保持できる項目を以下に示します。

項目	
EtherCAT	EtherCAT スレーブのバックアップパラメータ設定
I/O マップ	変数名
	変数コメント
	変数種別
イベント設定	
タスク設定	プログラム割付設定
プログラミング - POU	プログラム
	セクション
	外部／内部変数
	ファンクション
	ファンクションブロック
プログラミング - データ	データ型
	グローバル変数
EtherNet/IP コネクション設定	

上記以外の項目を変更し、ダウンロードした場合は、デバイス出力を保持しません。

また、Sysmac Studio の同期ウィンドウで、［以下を転送しない（すべて同期の対象外）。-CJ 高機能ユニットのパラメータ /EtherCAT スレーブのバックアップパラメータ］にチェックを入れてください。チェックを外すと、デバイス出力を保持しません。

デバイス出力を保持し、かつ EtherCAT スレーブのバックアップパラメータを変更したい場合は、以下のように行ってください。

1 **ユーザプログラムだけを変更し、ダウンロードします。**

2 **EtherCAT 構成編集画面のスレーブを選択します。**

3 **EtherCAT 構成の右側に表示されるスレーブのパラメータ設定にて、[バックアップパラメータ設定の編集] ボタンをクリックします。**

4 **[バックアップパラメータ設定の編集] ウィンドウの [転送 [パソコン→スレーブ]] をクリックします。**

[転送 [パソコン→スレーブ]] を実行すると、転送完了後に転送対象のスレーブがリスタートします。この場合、転送対象のスレーブのデバイス出力は保持されません。しかし、転送対象外のスレーブのデバイス出力は保持されます。

● 関連システム定義変数

システム定義変数による方法でバックアップを行う場合、関連するシステム定義変数には、以下のものがあります。システム定義変数の詳細は、🕮「A-4 システム定義変数の個別仕様」(P.A-46)を参照してください。

変数名	名称	機能	データ型	R/W
_DeviceOutHoldCfg	デバイス出力保持設定	動作モード変更時またはダウンロード時に、対象のデバイス出力を保持する場合、16#A5A5 とします。 16#A5A5 以外では、動作モード変更時またはダウンロード時に、対象のデバイス出力は初期化されます。	WORD	RW
_DeviceOutHoldStatus	デバイス出力保持状態	動作モード変更時またはダウンロード時に、対象のデバイス出力が保持された場合、TRUE となります。 デバイス出力保持設定が 16#A5A5 以外のとき、または全停止フォールトレベルのコントローラ異常発生時に、対象のデバイス出力が初期化された場合、FALSE となります。	BOOL	R

保持条件

保持条件とは、ダウンロード前と後で以下のすべての条件を満たすことを指します。

- 変数名が同じである。
- データ型名が同じである。
- 保持属性が、「保持あり」である。

保持属性が「保持あり」の変数について、POU 名変数名を変更したときの変数の値の詳細は、🕮「 POU 名の変更や POU の新規作成などによる保持変数の値」(P.6-66)を、データ型を変更したときの変数の値の詳細は、🕮「 保持変数のデータ型を変更したときの変数の値」(P.6-69)を参照してください。

POU 名の変更や POU の新規作成などによる保持変数の値

Sysmac Studio で作成したプロジェクトをコントローラにダウンロードすると、コントローラは、Sysmac Studio で表示される POU 名とは異なる POU 名で扱います。そのため、Sysmac Studio で同じ POU 名が表示される場合でも、コントローラは異なる POU として扱うことがあります。

Sysmac Studio で、POU 名の変更や POU の新規作成などの操作を行った後、ダウンロードを実行した場合について、コントローラが扱う POU 名と保持属性が「保持あり」のローカル変数の値について説明します。

● Sysmac Studio で操作する前の状態

コントローラには、Sysmac Studio で作成されたプロジェクト「Project_A」がダウンロードされています。「Project_A」には「POU_A」があり、「POU_A」にはローカル変数 VarA があります。Symac Studio で「POU_A」と表示される POU が、コントローラ側では「POU_XXX」として扱われているものとします。また、コントローラ側のローカル変数 VarA の現在値は「20」に変化しているものとします。

Sysmac Studio 側	
プロジェクト名	: Project_A
POU 名	: POU_A
変数名	: VarA
VarA の初期値	: 0

ダウンロード

コントローラ側	
プロジェクト名	: Project_A
POU 名	: POU_XXX
変数名	: VarA
VarA の現在値	: 20

● POU 名を変更した場合

Sysmac Studio で、「POU_A」の名称を「POU_B」に変更し、ダウンロードした場合でも、コントローラ側の POU 名およびローカル変数 VarA の現在値は変更されません。

Sysmac Studio 側	
プロジェクト名	: Project_A
POU 名	: POU_B
変数名	: VarA
VarA の初期値	: 0

ダウンロード

コントローラ側	
プロジェクト名	: Project_A
POU 名	: POU_XXX
変数名	: VarA
VarA の現在値	: 20

● POU を新規作成した場合

Sysmac Studio で、ローカル変数 VarA がある「POU_C」を新規作成し、ダウンロードした場合、コントローラ側は、新規の「POU_YYY」および新規のローカル変数 VarA として扱います。
したがって、新規のローカル変数 VarA の値は初期値の「0」となります。

Sysmac Studio 側	
プロジェクト名	: Project_A
POU 名	: POU_A
変数名	: VarA
VarA の初期値	: 0
POU 名	: POU_C
変数名	: VarA
VarA の初期値	: 0

ダウンロード

コントローラ側	
プロジェクト名	: Project_A
POU 名	: POU_XXX
変数名	: VarA
VarA の現在値	: 20
POU 名	: POU_YYY
変数名	: VarA
VarA の現在値	: 0

● POU をコピーして POU を作成した場合

Sysmac Studio で、「POU_A」をコピーして「POU_A_copy」を作成し、ダウンロードした場合、コントローラ側は、新規の「POU_ZZZ」および新規のローカル変数 VarA として扱います。
したがって、新規のローカル変数 VarA の値は初期値の「0」となります。

Sysmac Studio 側	
プロジェクト名	: Project_A
POU 名	: POU_A
変数名	: VarA
VarA の初期値	: 0
POU 名	: POU_A_copy
変数名	: VarA
VarA の初期値	: 0

ダウンロード

コントローラ側	
プロジェクト名	: Project_A
POU 名	: POU_XXX
変数名	: VarA
VarA の現在値	: 20
POU 名	: POU_ZZZ
変数名	: VarA
VarA の現在値	: 0

● エクスポートしたプロジェクトファイルをコピーした場合

Sysmac Studio で、「Project_A」をコピーして「Project_A_copy」を作成し、ダウンロードした場合でも、コントローラ側の POU 名およびローカル変数 VarA の現在値は変更されません。

Sysmac Studio 側	
プロジェクト名	: Project_A_copy
POU 名	: POU_A
変数名	: VarA
VarA の初期値	: 0

ダウンロード

コントローラ側	
プロジェクト名	: Project_A_copy
POU 名	: POU_XXX
変数名	: VarA
VarA の現在値	: 20

「Project_A」をコピーする時点で存在しなかったローカル変数 VarB を、「Project_A」に追加してダウンロードし、その後「Project_A_copy」に同じ定義のローカル変数 VarB を追加してダウンロードした場合、コントローラ側のローカル変数 VarB の現在値は変更されません。
「Project_A」からローカル変数 VarB をコピーしたり、変数テーブルをマージしたりした場合も同様にコントローラ側のローカル変数 VarB の現在値は変更されません。

Sysmac Studio 側	
プロジェクト名	: Project_A_copy
POU 名	: POU_A
変数名	: VarA
VarA の初期値	: 0
変数名	: VarB
VarB の初期値	: 0

ダウンロード

コントローラ側	
プロジェクト名	: Project_A_copy
POU 名	: POU_XXX
変数名	: VarA
VarA の現在値	: 20
変数名	: VarB
VarB の現在値	: 50

● 別に作成したプロジェクトに同じ名前の POU を作成

Sysmac Studio で、「Project_A」の POU 名と同じ「POU_A」を持つ「Project_B」を作成しダウンロードした場合、コントローラ側は、新規の「POU_YYY」および新規のローカル変数 VarA として扱います。したがって、新規のローカル変数 VarA の値は初期値の「0」となります。

Sysmac Studio 側	
プロジェクト名	: Project_B
POU 名	: POU_A
変数名	: VarA
VarA の初期値	: 0

ダウンロード

コントローラ側	
プロジェクト名	: Project_B
POU 名	: POU_YYY
変数名	: VarA
VarA の現在値	: 0

● 別のプロジェクトに元のプロジェクトの POU をコピーした場合

Sysmac Studio で「Project_B」を作成し、「Project_A」からコピーした「POU_A」をペーストしてダウンロードした場合、コントローラ側は、新規の「POU_YYY」および新規のローカル変数 VarA として扱います。したがって、新規のローカル変数 VarA の値は初期値の「0」となります。

Sysmac Studio 側	
プロジェクト名	: Project_B
POU 名	: POU_A
変数名	: VarA
VarA の初期値	: 0

ダウンロード



コントローラ側	
プロジェクト名	: Project_B
POU 名	: POU_YYY
変数名	: VarA
VarA の現在値	: 0

保持変数のデータ型を変更したときの変数の値

Sysmac Studio で、保持属性が「保持あり」の変数のデータ型を変更した後に、「保持あり」の変数の現在値を含めたリストアや転送機能、ダウンロードなどの操作を実行した場合について、コントローラ側の変数の値について説明します。

● 操作と変数の値

保持属性が「保持あり」の変数の動作には、2 つの動作パターンがあります。操作と動作パターンの関係を以下の表に示します。

操作	変数の値の動作パターン
• Sysmac Studio コントローラバックアップ機能によるリストア • SD メモリカードバックアップ機能によるリストア	パターン 1
• Sysmac Studio 変数／メモリのバックアップ機能による変数のリストア	パターン 2
• Sysmac Studio からダウンロード	パターン 2

以降に、パターン 1 とパターン 2 の動作について、データ型ごとに説明します。

● 基本データ型の変更

Sysmac Studio で変更する前の変数は以下の定義とします。

変数名	データ型	割付先
ABC	INT	−
DEF	INT	%D100

各変数の保持属性は「保持あり」とし、現在値が以下であるとします。

変数／メモリ	現在値
ABC	10
DEF	20

CJ ユニット用メモリの DM100 と DM101 の現在値が以下であるとします。

変数／メモリ	現在値
D100	20
D101	50

データ型の変更内容とコントローラ側の変数の動作の関係を以下の表に示します。

変更内容	パターン 1	パターン 2
変更なし	変数の値は保持されます。 ABC = 10 DEF = 20	
変数名を「ABC」から「GHI」に変更	変数の値は初期値となります。 GHI = 初期値	
変数「ABC」のデータ型を INT から DINT に変更	変数の値は初期値となります。 ABC = 初期値	
変数「ABC」の保持属性を「保持なし」に変更	変数の値は初期値となります。*1 ABC = 初期値	
変数「DEF」の割付先を「D100」から「D101」に変更*2	変数の値は変更した割付先の値となります。 DEF = 50	
変数「DEF」の割付先を削除	変数の値は初期値となります。 DEF = 初期値	

*1. 保持属性が「保持なし」の変数を「保持あり」に変更した場合も、変数の値は初期値となります。
*2. 割付先に CJ ユニット用メモリを指定した例です。NY シリーズコントローラでは、CJ ユニット用メモリを使用できないため、基本データ型を変更する場合の参考例としてください。

● 構造体型の変更

Sysmac Studio で変更する前の構造体型は以下の定義とします。

データ型名	メンバ	メンバのデータ型
STR_1	x	INT
	y	INT

変数 ABC のデータ型が「STR_1」で保持属性は「保持あり」とし、各メンバの現在値が以下であるとします。

メンバ	現在値
ABC.x	100
ABC.y	-200

データ型の変更内容とコントローラ側の変数の動作の関係を以下の表に示します。
保持される条件と初期値となる条件の両方に該当する場合、変数の値は初期値となります。

変更内容	パターン 1	パターン 2
変更なし	メンバの値は保持されます。 ABC.x = 100 ABC.y = -200	
変数名を「ABC」から「DEF」に変更	メンバの値は初期値となります。 DEF.x = 初期値 DEF.y = 初期値	
メンバ「x」を「x1」に変更	メンバの値は保持されます。 ABC.x1 = 100 ABC.y = -200	変更したメンバの値は初期値となります。 ABC.x1 = 初期値 ABC.y = -200
メンバ「x」のデータ型を INT から WORD に変更[*1]（データサイズは同じ）	メンバの値は保持されます。 ABC.x = WORD#16#0064 (= 100) ABC.y = -200	
メンバ「y」のデータ型を INT から UINT に変更[*1]（データサイズは同じで、符号ありから符号なしに変更）	メンバの値は保持されます。変更したメンバもそのまま代入されるため、絶対値と符号が変化します。 ABC.x = 100 ABC.y = 65336	
メンバ「y」のデータ型を INT から DINT に変更[*1]（データサイズが拡張）	メンバの値は初期値となります。 ABC.x = 初期値 ABC.y = 初期値	メンバの値は保持されます。変更したメンバは符号拡張され、絶対値と符号は変化しません。 ABC.x = 100 ABC.y = -200
メンバ「y」のデータ型を INT から UDINT に変更[*1]（データサイズが拡張し、符号ありから符号なしに変更）	メンバの値は初期値となります。 ABC.x = 初期値 ABC.y = 初期値	メンバの値は保持されます。変更したメンバは符号拡張され、絶対値と符号が変化します。 ABC.x = 100 ABC.y = 4294967096
メンバ「y」のデータ型を INT から SINT に変更[*1]（データサイズが縮小）	メンバの値は初期値となります。 ABC.x = 初期値 ABC.y = 初期値	データサイズを縮小したメンバの値は初期値となります。 ABC.x = 100 ABC.y = 初期値
メンバ「x」を「y」に、メンバ「y」を「x」に変更	メンバの値は不正に保持されます。 ABC.y = 100 ABC.x = -200	メンバの値は保持されます。 ABC.y = -200 ABC.x = 100

変更内容	パターン1	パターン2
メンバ「y」の次に INT のメンバ「z」を追加	メンバの値は初期値となります。*2 ABC.x = 初期値 ABC.y = 初期値 ABC.z = 初期値	既存のメンバの値は保持されます。追加したメンバの値は初期値となります。 ABC.x = 100 ABC.y = -200 ABC.z = 初期値
メンバ「x」とメンバ「y」の間に INT のメンバ「z」を追加	メンバの値は初期値となります。*2 ABC.x = 初期値 ABC.z = 初期値 ABC.y = 初期値	既存のメンバの値は保持されます。追加したメンバの値は初期値となります。 ABC.x = 100 ABC.z = 初期値 ABC.y = -200
メンバ「x」の配列の要素数を「1」から「3」に変更	メンバの値は初期値となります。*2 ABC.x[0] = 初期値 ABC.x[1] = 初期値 ABC.x[2] = 初期値 ABC.y = 初期値	既存のメンバの値は保持されます。配列の要素数を変更したメンバは、先頭から保持され、拡張されたメンバの値は初期値となります。 ABC.x[0] = 100 ABC.x[1] = 初期値 ABC.x[2] = 初期値 ABC.y = -200

*1. データ型を変更する場合は、暗黙のキャストに従うものとします。暗黙のキャストについては、🕮「暗黙のキャスト」（P.6-121）を参照してください。

*2. ダウンロードする前に変数の値をアップロードしておき、ダウンロード後に変数の値をダウンロードすることで、既存のメンバの値を保持することができます。

● 共用体型の変更

Sysmac Studio で変更する前の共用体型は以下の定義とします。

データ型名	メンバ	メンバのデータ型
UNI_1	x	WORD
	y	DWORD

変数 ABC のデータ型が「UNI_1」で保持属性は「保持あり」とし、各メンバの現在値が以下であるとします。

メンバ	現在値
ABC	WORD#16#0100

　データ型の変更内容とコントローラ側の変数の動作の関係を以下の表に示します。
保持される条件と初期値となる条件の両方に該当する場合、変数の値は初期値となります。

変更内容	パターン 1	パターン 2
変更なし	メンバの値は保持されます。 ABC = WORD#16#0100	
変数名を「ABC」から「DEF」に変更	メンバの値は初期値となります。 DEF = 初期値	
メンバ「x」のデータ型を WORD から BYTE に変更*1（共用体型としてのデータサイズは同じ）	メンバの値は保持されます。 ABC = WORD#16#0100	
メンバ「x」のデータ型を WORD から LWORD に変更*1（共用体型としてのデータサイズが拡張）	メンバの値は初期値となります。 ABC = 初期値	メンバの値は保持されます。 ABC = WORD#16#0100
メンバ「y」のデータ型を DWORD から BYTE に変更*1（共用体型としてのデータサイズが縮小）	メンバの値は初期値となります。 ABC = 初期値	
メンバ「x」を「y」に、メンバ「y」を「x」に変更	メンバの値は保持されます。 ABC = WORD#16#0100	
メンバ「y」の次に BYTE のメンバ「z」を追加（共用体型としてのデータサイズは同じ）	メンバの値は保持されます。 ABC = WORD#16#0100	
メンバ「y」の次に LWORD のメンバ「z」を追加（共用体型としてのデータサイズが拡張）	メンバの値は初期値となります。 ABC = 初期値	メンバの値は保持されます。 ABC = WORD#16#0100
メンバ「y」の次に BYTE のメンバ「z」を追加しメンバ「y」を削除（共用体型としてのデータサイズが縮小）	メンバの値は初期値となります。 ABC = 初期値	

*1. データ型を変更する場合は、暗黙のキャストに従うものとします。暗黙のキャストについては、📖「暗黙のキャスト」（P.6-121）を参照してください。

● 配列変数の変更

Sysmac Studio で変更する前の配列変数は以下の定義とします。

変数名	データ型
ABC	ARRAY[1..3] OF INT

変数 ABC の保持属性は「保持あり」とし、各要素の現在値が以下であるとします。

メンバ	現在値
ABC[1]	100
ABC[2]	200
ABC[3]	300

データ型の変更内容とコントローラ側の変数の動作の関係を以下の表に示します。
保持される条件と初期値となる条件の両方に該当する場合、変数の値は初期値となります。

変更内容	パターン 1	パターン 2
変更なし	要素の値は保持されます。 ABC[1] = 100 ABC[2] = 200 ABC[3] = 300	
変数名を「ABC」から「DEF」に変更	要素の値は初期値となります。 DEF[1] = 初期値 DEF[2] = 初期値 DEF[3] = 初期値	
要素のデータ型を INT から UINT に変更[*1] （データサイズは同じ）	要素の値は初期値となります。 ABC[1] = 初期値 ABC[2] = 初期値 ABC[3] = 初期値	
要素のデータ型を INT から DINT に変更[*1] （データサイズが拡張）	要素の値は初期値となります。 ABC[1] = 初期値 ABC[2] = 初期値 ABC[3] = 初期値	
要素のデータ型を INT から SINT に変更[*1] （データサイズが縮小）	要素の値は初期値となります。 ABC[1] = 初期値 ABC[2] = 初期値 ABC[3] = 初期値	
配列要素の終了番号を「3」から「4」に変更[*2] （要素数が「3」から「4」に拡張）	要素の値は初期値となります。 ABC[1] = 初期値 ABC[2] = 初期値 ABC[3] = 初期値 ABC[4] = 初期値	既存の要素番号の要素の値は保持されます。新規の要素番号の要素の値は初期値となります。 ABC[1] = 100 ABC[2] = 200 ABC[3] = 300 ABC[4] = 初期値
配列要素の開始番号を「1」から「0」に変更[*2] （要素数が「3」から「4」に拡張）	要素の値は初期値となります。 ABC[0] = 初期値 ABC[1] = 初期値 ABC[2] = 初期値 ABC[3] = 初期値	既存の要素番号の要素の値は保持されます。新規の要素番号の要素の値は初期値となります。 ABC[0] = 初期値 ABC[1] = 100 ABC[2] = 200 ABC[3] = 300
配列要素の終了番号を「3」から「2」に変更[*2] （要素数が「3」から「2」に縮小）	要素の値は初期値となります。 ABC[1] = 初期値 ABC[2] = 初期値	既存の要素番号の要素の値は保持されます。 ABC[1] = 100 ABC[2] = 200

変更内容	パターン 1	パターン 2
配列要素の開始番号を「1」から「2」に変更[*2] （要素数が「3」から「2」に縮小）	要素の値は初期値となります。 ABC[2] = 初期値 ABC[3] = 初期値	既存の要素番号の要素の値は保持されます。 ABC[2] = 200 ABC[3] = 300
配列要素の開始番号を「1」から「3」に、終了番号を「3」から「5」変更[*2] （要素数は同じ）	要素数が同じである場合、要素の値は先頭から保持されます。[*3] ABC[3] = 100 ABC[4] = 200 ABC[5] = 300	既存の要素番号の要素の値は保持されます。新規の要素番号の要素の値は初期値となります。 ABC[3] = 300 ABC[4] = 初期値 ABC[5] = 初期値

*1. データ型を変更する場合は、暗黙のキャストに従うものとします。暗黙のキャストについては、📖「暗黙のキャスト」（P.6-121）を参照してください。
*2. 変数のデータが基本データ型または派生データ型のいずれでも同じ結果となります。
*3. 要素数が拡張または縮小となる場合、すべての要素の値は初期値となります。

● 列挙型の変更

Sysmac Studio で変更する前の列挙型は以下の定義とします。

データ型名	列挙子	値
ENU_1	x	1
	y	2

変数 ABC のデータ型が「ENU_1」で保持属性は「保持あり」とし、変数の現在値が以下であるとします。

メンバ	現在値
ABC	1（= x）

データ型の変更内容とコントローラ側の変数の動作の関係を以下の表に示します。

変更内容	パターン 1	パターン 2
変更なし	変数の値は保持されます。 ABC = 1（= x）	
変数名を「ABC」から「DEF」に変更	変数の値は初期値となります。 ABC = 初期値	
列挙子「y」の次に値が「3」の列挙子「z」を追加	変数の値は保持されます。 ABC = 1（= x）	
列挙子「x」を削除	列挙子が存在しない場合も、変数の値は保持されます。 ABC = 1	
列挙子「X」の値を「1」から「5」に変更	列挙子の値を変更しても、変数の値は保持されます。 ABC = 1	

6-3-10　ファンクションブロック（FB）のインスタンス

インスタンス化された FB は、データ型のひとつとして、ローカル変数テーブルに登録･表示されます。



参考

FB のインスタンスは、インスタンスが記述されたプログラムのローカル変数扱いとなり、そのプログラムのローカル変数テーブルに登録・表示されます。グローバル変数とはならない点にご注意ください。

6-3-11　変数の値のモニタ

変数の値は Sysmac Studio のウォッチウィンドウからモニタできます。

1 **［表示］–［ウォッチウィンドウ］を選択します。**
ウォッチウィンドウが表示されます。

2 **コントローラにオンライン接続し、以下のいずれかによって、変数を登録します。**

(1)　ウォッチウィンドウの名称セルに変数名を入力する。

(2)　エディタや変数テーブルからウォッチウィンドウへドラッグ & ドロップして登録する。
その変数の現在値が表示されます。

6-3-12　変数名などのプログラム関連の名称の制限

プログラム関連の名称の制限を以下に一覧で示します。

文字制限

<table>
<tr><th>プログラム関連の名称</th><th>使用可能な文字</th><th>予約語</th><th>マルチバイト文字使用可否</th><th>大文字／小文字の区別</th><th>最大サイズ（NULL含まず）</th><th>文字コード</th></tr>
<tr><td>変数名
（POU インスタンス名含む）</td><td rowspan="14">● 使用できる文字
• 0 ～ 9 、A ～ Z 、a ～ z
• 半角カナ
• _（アンダーバー）
• マルチバイト文字（例 : 日本語）
予約語は、後述の「予約語」を参照してください。

● 組み合わせて使用できない文字
• 数字（0 ～ 9）で始まる文字列
• P_ で始まる文字列
• アンダーバーで始まる文字列
• アンダーバー（_）を 2 個以上連続した文字列
• アンダーバー（_）で終端する文字列
• 識別子を構成する文字列の接頭または、接尾に 1 文字以上の拡張空白類（全角空白など、Unicode まで拡張した空白系文字 ）を含む文字列</td><td rowspan="14">後述の「予約語」参照</td><td rowspan="14">可</td><td rowspan="11">しない</td><td rowspan="5">127 バイト</td><td rowspan="14">UTF-8
*1</td></tr>
<tr><td>POU 定義名</td></tr>
<tr><td>データ型名</td></tr>
<tr><td>構造体メンバ名
共用体メンバ名</td></tr>
<tr><td>列挙子</td></tr>
<tr><td>タスク名</td><td>63 バイト</td></tr>
<tr><td>名前空間名</td><td>93 バイト</td></tr>
<tr><td>変数名のフルパス</td><td>ネットワーク変数 : 255 バイト
他 : 511 バイト</td></tr>
<tr><td>デバイス名</td><td rowspan="5">127 バイト</td></tr>
<tr><td>セクション名</td><td rowspan="3">する</td></tr>
<tr><td>軸名</td></tr>
<tr><td>軸グループ名</td></tr>
<tr><td>カムテーブル名</td><td>しない</td></tr>
</table>

*1. UTF-8 では、半角英数字 1 文字は 1 バイト、マルチバイト文字は複数バイトを使用します。日本語文字は 3 バイトが目安です。

予約語

以下の名称は、プログラムチェック時にエラーとなります。

- 『NY シリーズ コマンドリファレンスマニュアル 基本編（SBCA-437）』の命令と同じ名称
- 『NY シリーズ コマンドリファレンスマニュアル モーション編（SBCE-380）』の命令と同じ名称
- システムが内部で使用する予約語

名称の重複の禁止

以下の名称の重複は禁止です。プログラムチェック時にエラーとなります。

- 同一のコントローラにおけるグローバル変数名の重複
- 同一の POU 内での変数名の重複
- 同一の POU 内でのセクション名の重複
- 同一の構造体または共用体内での、メンバ名の重複
- 同一の列挙型内での列挙子の重複
- ローカル変数名とグローバル変数名の重複
- POU 名とデータ型名の重複
- データ型名と変数名の重複
- 列挙子と、他の列挙型の列挙子と重複
- 列挙子と変数名の重複

6-4　定数（リテラル）

ここでは、定数について、詳しく説明します。

6-4-1　定数とは

変数の値は、その変数に代入するデータによって変わります。これに対して、値が変わることがないのが、定数です。
定数は、変数とは異なり、メモリに格納されている情報ではありません。定数は、宣言しなくても、POU のアルゴリズム部に記述することができます。
NY シリーズコントローラでは、定数にも、変数と同様、「データ型」という概念があります。

6-4-2　各データ型の記述形式

各データ型の定数の記述形式を、次に示します。定数を、決められた記述形式以外で記述すると、ビルド時にエラーとなります。

ブール型

ブール型は、データ型名として「BOOL」を使用します。また、値として、1 または 0 以外に「TRUE」と「FALSE」が使用可能です。記述形式と意味は、次のとおりです。

記述形式	意味
TRUE　または　FALSE	TRUE、BOOL#1、BOOL#TRUE、1 は等価です。 FALSE、BOOL#0、BOOL#FALSE、0 は等価です。
BOOL#1　または　BOOL#0	
BOOL#TRUE　または　BOOL#FALSE	
1　または　0	

ビット列型

ビット列型は、データ型名として「BYTE」、「WORD」、「DWORD」、「LWORD」のいずれかを使用します。また、基数として、「2」、「8」、「10」、「16」のいずれかを使用します。記述形式と記述例は、次のとおりです。

記述形式	記述例
{ データ型名 }#{ 基数 }#{ 数値 }	WORD#16#0064
{ 基数 }#{ 数値 }	16#0064
{ 数値 }*1	100

*1. 基数は 10、つまり 10 進数であると解釈されます。

整数型

整数型は、データ型名として「SINT」、「USINT」、「INT」、「UINT」、「DINT」、「UDINT」、「LINT」、「ULINT」のいずれかを使用します。また、基数として、「2」、「8」、「10」、「16」のいずれかを使用します。記述形式と記述例は、次のとおりです。

記述形式	記述例
{ データ型名 }#{ 基数 }#{ 数値 }	INT#10#-1
{ 基数 }#{ 数値 }	10#-1
{ データ型名 }#{ 数値 }*1	INT#-1
{ 数値 }*1	-1

*1. 基数は 10、つまり 10 進数であると解釈されます。

実数型

実数型は、データ型名として「REAL」と「LREAL」のいずれかを使用します。また、実数型で使用できる基数は「10」だけです。記述形式と記述例は、次のとおりです。

記述形式	記述例
{ データ型名 }#{ 基数 }#{ 数値 }	LREAL#10#-3.14
{ 基数 }#{ 数値 }	10#-3.14
{ データ型名 }#{ 数値 }*1	LREAL#-3.14
{ 数値 }*1	-3.14

*1. 基数は 10、つまり 10 進数であると解釈されます。

持続時間型

持続時間型は、データ型名として「TIME」と「T」のいずれかを使用します。記述形式と記述例は、次のとおりです。

記述形式	記述例
TIME#{ 日 }d{ 時 }h{ 分 }m{ 秒 } s{ ミリ秒 }ms	TIME#61m5s
T#{ 日 }d{ 時 }h{ 分 }m{ 秒 }s{ ミリ秒 }ms	T#61m5s

持続時間型の定数の記述には、次のルールがあります。

- { 日 }、{ 時 }、{ 分 }、{ 秒 }、{ ミリ秒 } のすべてを記述する必要はありません。そのうちのいずれか 1 つ以上を記述します。
- 「TIME#12d3.5h」のように、小数の記述が可能です。
- 時間の有効範囲を超えた記述が可能です。たとえば、「T#-61m5s」は「T#-1h1m5s」と同じ意味になります。
- 数値は 10 進数であると解釈されます。10 進数以外の数を記述した場合は、ビルド時にエラーとなります。
- 時間の並びを変えた記述が可能です。たとえば「T#1h2d」は「T#2d1h」と同じ意味になります。

日付型

日付型は、データ型名として「DATE」と「D」のいずれかを使用します。記述形式と記述例は、次のとおりです。

記述形式	記述例
DATE#{年}-{月}-{日}	DATE#2010-1-10
D#{年}-{月}-{日}	D#2010-1-10

日付型の定数の記述には、次のルールがあります。

- {年}、{月}、{日}のそれぞれの先頭に 1 つ以上の 0 を付けることが可能です。たとえば、「DATE#2010-01-10」は、「D#2010-1-10」と同じ意味になります。
- 日付の有効範囲を超えた記述をすると、ビルド時にエラーとなります。たとえば、「D#2010-01-35」はエラーとなります。
- 数値は 10 進数であると解釈されます。10 進数以外の数を記述した場合は、ビルド時にエラーとなります。

時刻型

時刻型は、データ型名として「TIME_OF_DAY」と「TOD」のいずれかを使用します。記述形式と記述例は、次のとおりです。

記述形式	記述例
TIME_OF_DAY#{時}:{分}:{秒}	TIME_OF_DAY#23:59:59.999999999
TOD#{時}:{分}:{秒}	TOD#23:59:59.999999999

時刻型の定数の記述には、次のルールがあります。

- {時}、{分}、{秒}のそれぞれの先頭に1つ以上の 0 を付けることができます。たとえば、「TOD#23:01:01」と「TOD#23:1:1」は同じ意味になります。
- 時分秒の有効範囲を超えた数値を記述すると、ビルド時にエラーとなります。たとえば、「TOD#24:00:00」はエラーとなります。
- 数値は 10 進数であると解釈されます。10 進数以外の数を記述した場合は、ビルド時にエラーとなります。

日付時刻型

日付時刻型は、データ型名として「DATE_AND_TIME」と「DT」のいずれかを使用します。記述形式と記述例は、次のとおりです。

記述形式	記述例
DATE_AND_TIME#{年}-{月}-{日}-{時}:{分}:{秒}	DATE_AND_TIME#2010-10-10-23:59:59.123
DT#{年}-{月}-{日}-{時}:{分}:{秒}	DT#2010-10-10-23:59:59.123

日付時刻型の定数の記述には、次のルールがあります。

- {年}、{月}、{日}、{時}、{分}、{秒}のそれぞれの先頭に1つ以上の0を付けることができます。たとえば、「DT#2010-01-10-23:01:01」と「DT#2010-1-10-23:1:1」は同じ意味になります。
- 年月日時分秒の有効範囲を超えた数値を記述すると、ビルド時にエラーとなります。たとえば、「DT#2010-01-35-00:00:00」や「DT#2010-01-30-24:00:00」はエラーとなります。
- 数値は 10 進数であると解釈されます。10 進数以外の数を記述した場合は、ビルド時にエラーとなります。

文字列型

文字列型は、文字列を 2 つの半角シングルクォートではさんで記述します。また、データ型名として「STRING」を使用することも可能です。記述形式と記述例は、次のとおりです。

記述形式	記述例
'{ 文字列 }'	' これは文字列 '
STRING#'{ 文字列 }'	STRING#' これは文字列 '

文字列の記述には、次のルールがあります。

- 文字数 0 の文字列も記述可能です。その場合の記述は、「''」となります。
- 以下のように、複数行にまたがる文字列を記述すると、ビルド時にエラーとなります。
  strVar := 'ABC
  DEF'
- タブや改行コードなどを記述する場合は、先頭に「$」を付けたエスケープ文字を使用します。エスケープ文字の名称と意味を次に示します。

エスケープ文字	名称	意味
$$	半角ドル文字	半角ドル記号文字（$：文字コード 0x24）
$'	半角のシングルクォート	半角のシングルクォート記号文字（'：文字コード 0x27）
$"	半角ダブルクォート	半角ダブルクォート記号文字（"：文字コード 0x22）
$L または $l	改行	カーソルを次の行に移動します。 LF 制御文字（Line Feed：文字コード 0x0A）
$N または $n	改行	カーソルを次の行に移動します。 NL 制御文字（New Line：文字コード 0x0A）
$P または $p	改ページ	カーソルを次のページに移動します。 FF 制御文字（Form Feed：文字コード 0x0C）
$R または $r	復帰	カーソルを行の先頭に移動します。 CR 制御文字（Carriage Return：文字コード 0x0D）
$T または $t	タブ文字	タブを表示します。 タブ文字（文字コード 0x09）
$（文字コード）	文字コード直接指定	文字コードを、2 桁の 16 進数で直接指定します。文字コードの範囲は、00 ～ FF です。 たとえば、‘$L’ は、‘$0A’ と同じ意味になります。 また、2 バイト以上の文字は、1 バイトごとに $ を付けます。 たとえば、文字コードが 0xE38182 であるひらがなの「あ」は、「$E3$81$82」と記述します。

- 文字コードを直接指定して文字を記述することも可能です。その場合も、先頭に「$」を付けます。文字コードは、2 桁の 16 進数で記述します。たとえば、改行「$L」の文字コードは 0x0A なので、「$0A」と記述できます。
- 文字コードを直接指定して、2 バイト以上の文字を記述する場合は、1 バイトごとに「$」を付けます。たとえば、文字コードが 0xE38182 であるひらがなの「あ」は、「$E3$81$82」と記述します。

列挙型

列挙型は、列挙型のデータ型名と列挙子を記述します。記述形式と記述例は、次のとおりです。

記述形式	記述例
{列挙型のデータ型名}＃{列挙子}	_eDAYOFWEEK#_WED



参考

パラメータのデータ型に列挙型が指定されている FB や FUN に列挙子を渡す場合は、列挙型のデータ型名を省略して、列挙子だけを記述することが可能です。
たとえば、BinToBCDs 命令の入力変数「Format」には、列挙型 _eBCD_FORMAT が指定されています。したがって、列挙型のデータ型名と列挙子を記述した「_eBCD_FORMAT#_BCD0」以外に、列挙型のデータ型名を省略した「_BCD0」も記述できます。

6-5　プログラム言語

ここでは、各プログラム言語について、詳しく説明します。
Sysmac Studio でのプログラム言語の入力方法については、『Sysmac Studio Version 1 オペレーションマニュアル（SBCA-470）』を参照してください。

6-5-1　プログラム言語とは

POU（プログラム、FB、FUN）内のアルゴリズムを表現するための言語を、プログラム言語と呼びます。
NY シリーズコントローラで使用できるプログラム言語は、ラダー図言語（LD）、ST 言語（Structured Text：構造化テキスト）の 2 種類です。

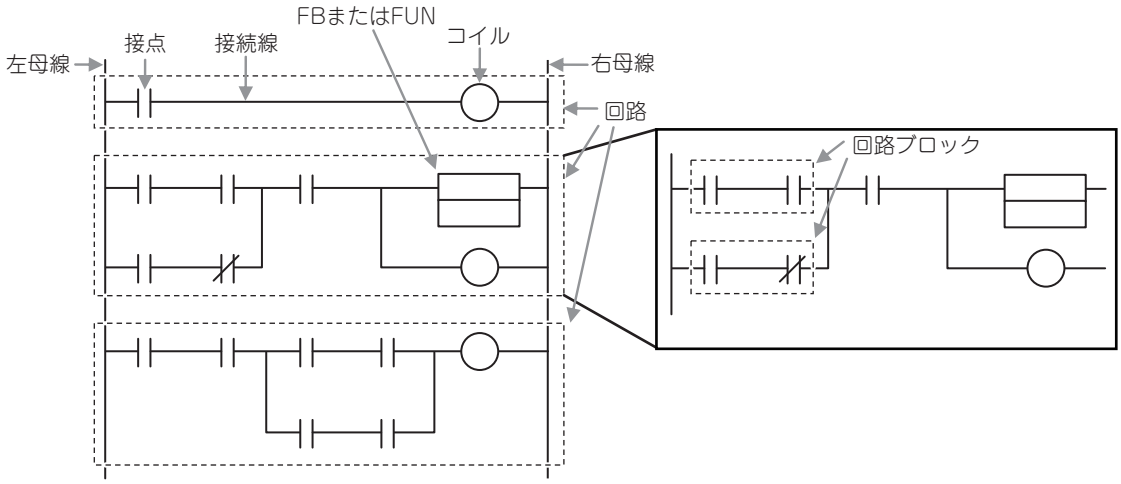
6-5-2　ラダー図言語（LD）

ラダー図言語は、電気回路のイメージで記述したグラフィカルなプログラム言語です。FB や FUN を含めた、処理の各部品がすべて図で表されており、それらを直線で接続することにより、一連のアルゴリズムを構築します。ラダー図言語で書かれたアルゴリズムを、ラダー図と呼びます。

ラダー図言語の構成要素

ラダー図言語は、左右の母線、接続線、ラダー図構文（接点、コイル）、FB、および FUN から成ります（*）。
* ジャンプ命令およびラベル命令だけは、ジャンプおよびラベルを示す記号で示します。

アルゴリズムは、それらを接続した複数の回路から構成されます。
回路とは、左母線から右母線までの構成要素のつながりのことです。さらに、回路は、（論理スタートを示す）LD/LDN 命令を先頭とした回路ブロックから構成されます。



● 母線

ラダー図の左端および右端の縦線を母線と呼びます。
それぞれの母線は、TRUE または FALSE のどちらかの状態をとります。これは、ラダー図を電気回路と見なした場合の、通電状態と非通電状態を表します。
ラダー図が書かれた POU が起動すると、左母線の値が TRUE となります。そして、左母線に接続している接点などの構成要素にも、TRUE の状態が伝達されます。それらの構成要素の動作にしたがって、TRUE の状態が順々に右方向に伝達されていきます。
この、TRUE の状態の流れを、「パワーフロー」と呼びます。母線は、パワーフローの源泉と言えます。

● 接続線

母線や各種構成要素を接続する水平方向の直線を、接続線と呼びます。
接続線は、TRUE または FALSE のどちらかの状態をとり、パワーフローを右方向へ伝える役割を果たします。

● 接点

接点は、接続線上に配置され、パワーフローを受けることにより、動作します。
接点には複数の種類があり、それぞれの仕様にしたがって、左から来たパワーフローを右に伝達したり、遮断したりします。
接点がパワーフローを右に伝達すると、接点の右側の接続線の状態が TRUE になり、遮断すると FALSE になります。
各接点の仕様の詳細は、🕮『NY シリーズ コマンドリファレンスマニュアル 基本編（SBCA-437)』を参照してください。

● コイル

コイルは、接続線上に配置され、パワーフローを受けることにより、動作します。
コイルの機能は、変数に TRUE または FALSE の値を書き込むことです。
コイルにも複数の種類があります。各コイルの仕様の詳細は、🕮『NY シリーズ コマンドリファレンスマニュアル 基本編（SBCA-437)』を参照してください。

● FB、FUN

FB、FUN は、接続線上に配置され、パワーフローを受けることにより、動作します。
各命令の仕様の詳細は、🕮『NY シリーズ コマンドリファレンスマニュアル 基本編（SBCA-437)』を参照してください。

ラダー図の実行順序

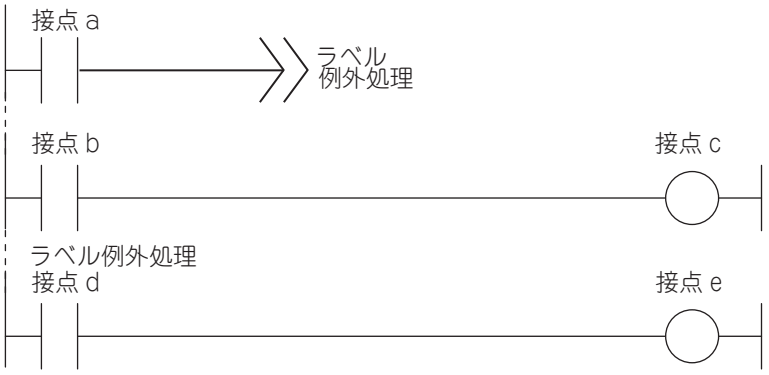
接点、コイル、FB、FUN は、パワーフローを受けることにより実行されます。
その実行順序は、ラダー図の上から下です。また、同じ高さの要素については、左から右の順序で実行されます。

ラダー図の終了

ラダー図は、上から順に実行され、一番下まで実行したら処理を終了します。
ただし、途中に END 命令または RETURN 命令があったら、その時点で処理を終了します。それらの命令より後に書かれた処理は、実行されません。

ラダー図の実行制御

ラダー図は、基本的には上から順に実行されますが、実行制御の命令を使用することにより、実行順序を変えることができます。
たとえば、以下の場合、接点 a が TRUE になると、「例外処理」のラベルが付けられた地点に実行が移ります。



ラダー図での FB、FUN の接続

● 接続形態

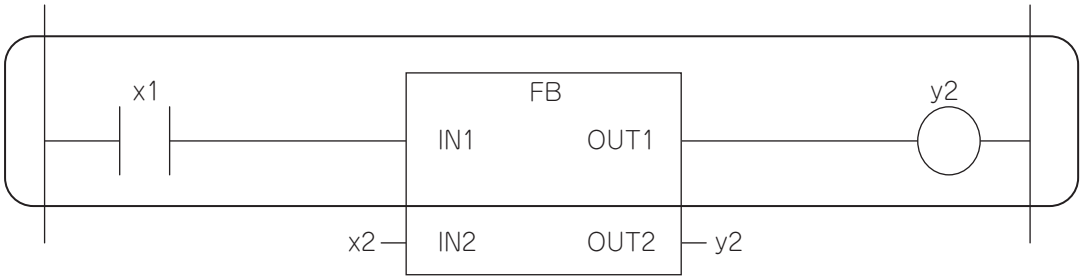
FB および FUN では、以下の 2 つの接続形態があります。

(1) パワーフロー入力、出力

ラダー図言語において、FB および FUN の入力変数を左母線、出力変数を右母線につなげる場合の、BOOL 型の入力、BOOL 型の出力を指します。

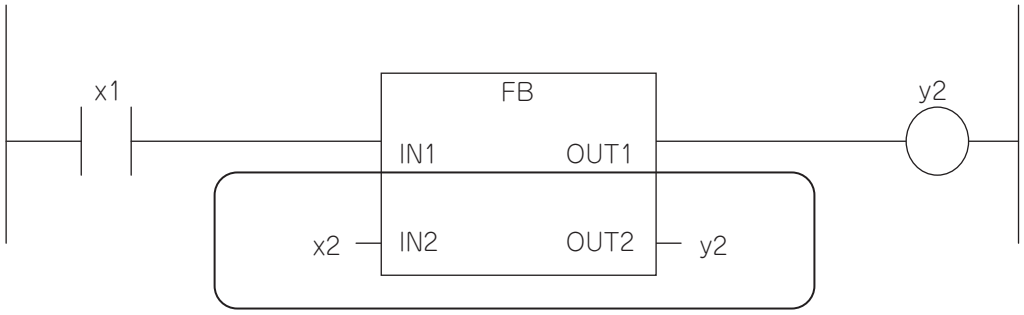
例）

左母線に接続されるパワーフローは、接点（CONTACT）が、右母線にはコイル（COIL）が接続されます。



(2) パラメータ入力、パラメータ出力

ラダー図言語において、FB および FUN の入力変数、出力変数を、左母線・右母線につなげない場合を指します。



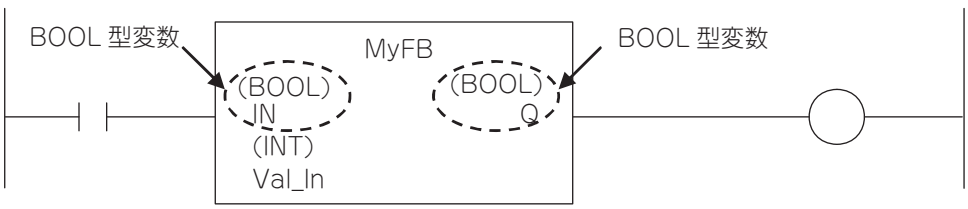
入力および出力のパラメータには、以下のように、変数または定数を指定可能です。

FB/FUN 側変数	入力パラメータ	出力パラメータ
入力変数	変数または定数を指定可能	−
出力変数	−	変数だけを指定可能
入出力変数	変数だけを指定可能	変数だけを指定可能

● BOOL 型変数の数

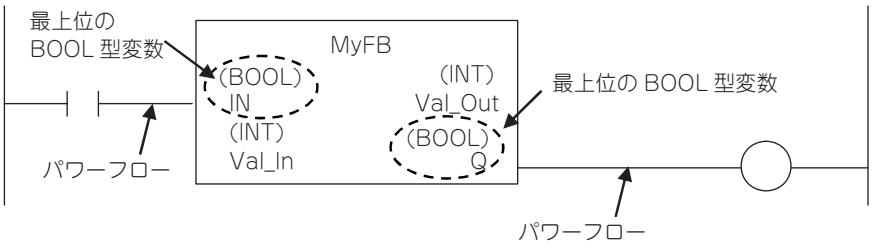
FB および FUN の入力および出力には、必ず BOOL 型変数（EN、ENO を含む）が 1 つ以上必要です。

例）

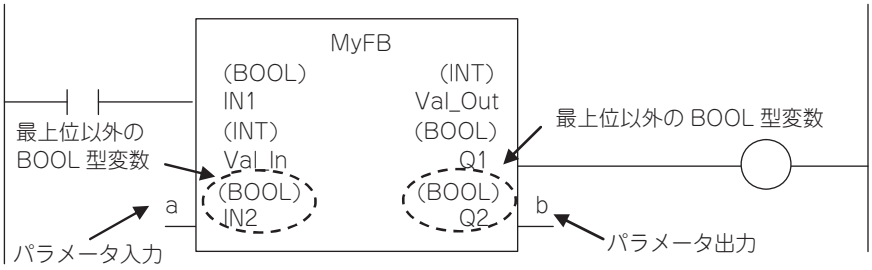


● BOOL 型変数の位置による接続形態

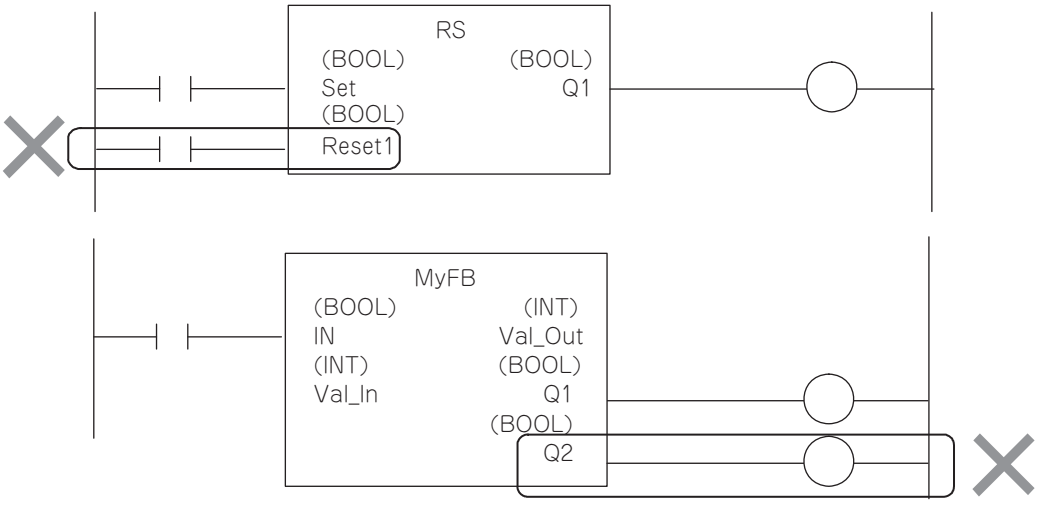
最上位にある BOOL 型変数は、左母線および右母線に接続されます。つまり、パワーフロー入力およびパワーフロー出力となります。



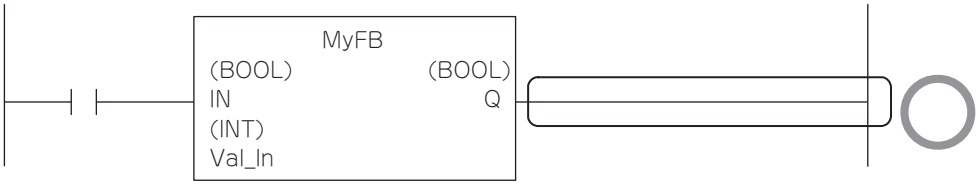
FB および FUN へのパワーフロー入力、パワーフロー出力は、それぞれ一つだけです。
最上位以外の BOOL 型変数は、パラメータ入力およびパラメータ出力となります。



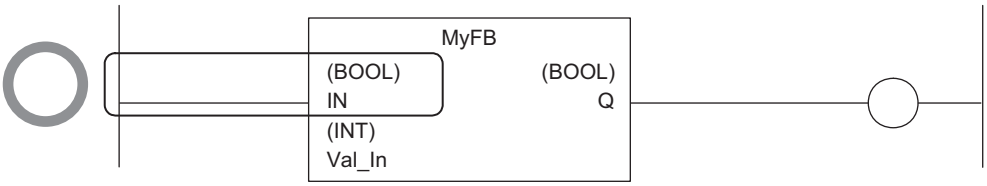
したがって、以下のように、複数の BOOL 型変数を、左母線または右母線に接続することはできません。



右母線に OUT 命令などがなく、直結してもかまいません。

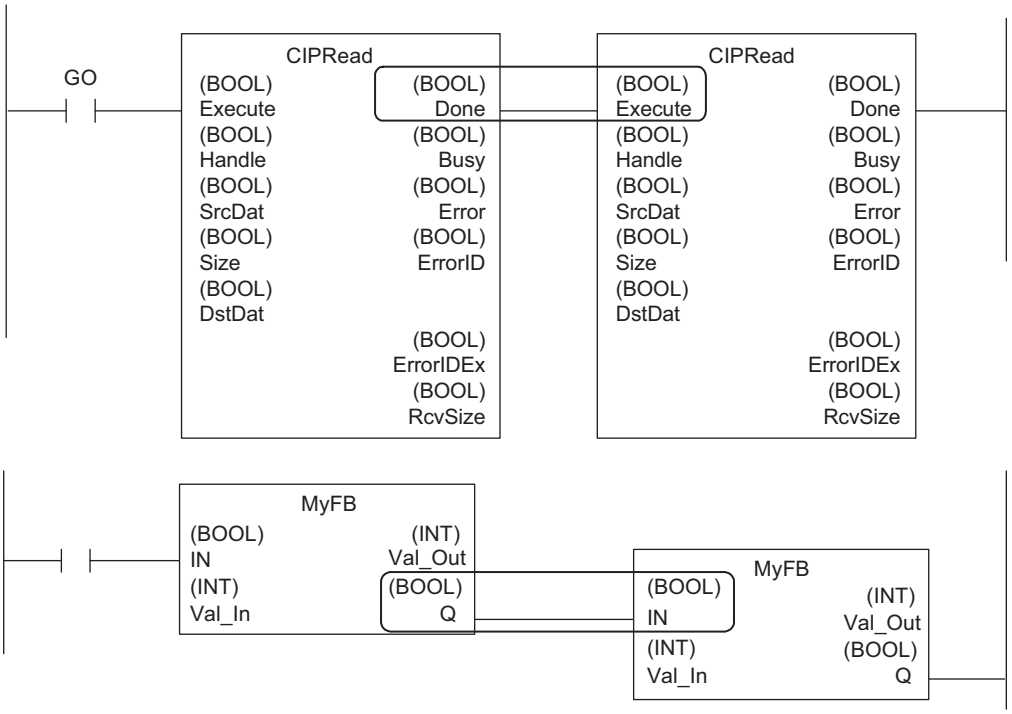


LD 命令などがなく左母線に直結してもかまいません。



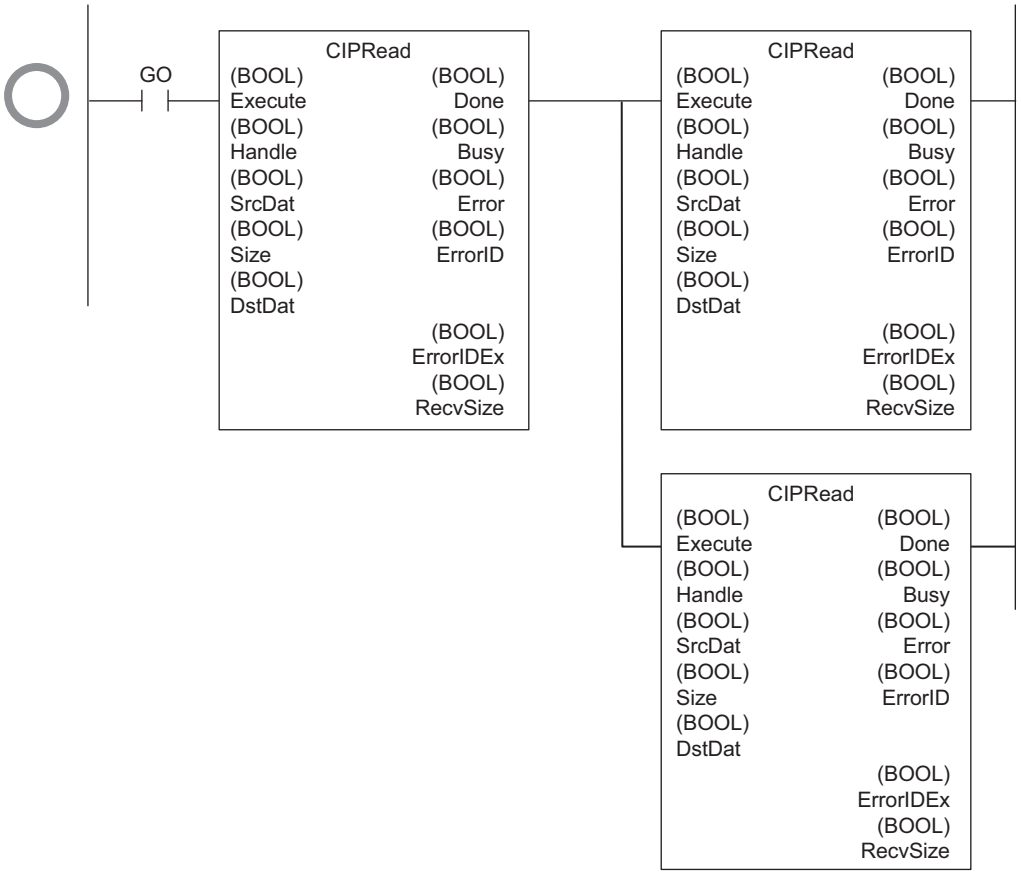
● カスケード接続

FB/FUN の出力を、他の FB/FUN の入力とするカスケード接続は、パワーフロー出力とパワー入力間だけで可能です。



• パワーフロー出力の分岐は可能です。

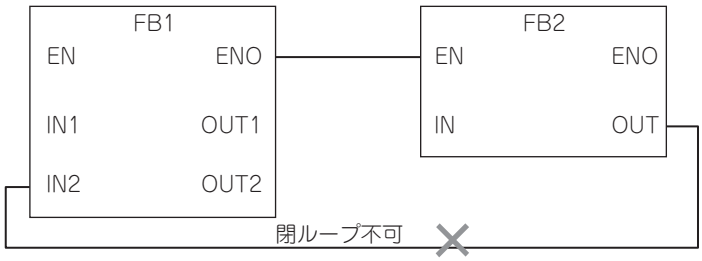
例）



制限

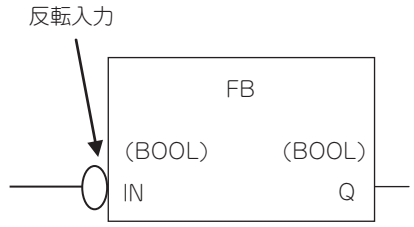
- 閉ループや接続線の交差はできません。

例）



● 入力の反転

BOOL 型の入力変数に対して、値を反転して入力させることができます。



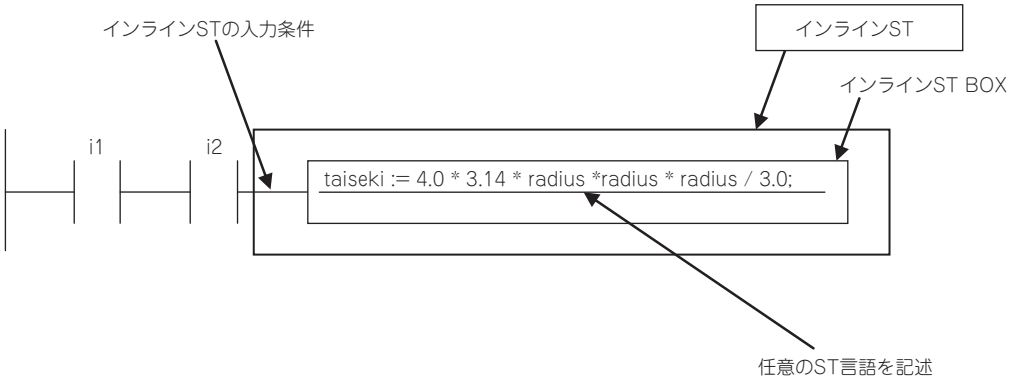
TRUE→FALSE

FALSE→TRUE

インライン ST

● 概要

インライン ST は、ラダー図の内の「インライン ST BOX」と呼ぶ箱（空白のテキスト入力エリア）の中に、ST 言語を記述するラダー図言語の要素です。
これにより、数値演算処理や文字列操作処理を、ラダー図言語内で簡単に記述できます。
「インライン ST BOX」への接続線が入力条件となり、それにしたがって、ST BOX 内の ST 言語が実行されます。下図のようになります。



なお、インライン ST は、ラダー図言語で記載された回路要素の一種として扱われます。そのため、FB や FUN とは異なり、入力変数、出力変数、入出力変数はありません。

● インライン ST 内の ST 言語の制限

インライン ST BOX 内には、任意の ST 言語プログラムを記述できます。

● インライン ST の実行条件

インライン ST の実行条件は以下のとおりです。

状態	動作
入力条件 TRUE 時	入力条件にしたがって動作する。 入力条件は、パワーフローまたは左母線直結が可能です。 立ち上がりまたは立ち下がり指定は、入力条件の接点で指定。
入力条件 FALSE 時	何もしない
マスタコントロールによるリセット時	何もしない

● インライン ST 内の変数のスコープ

インライン ST でアクセスできる変数のスコープは、インライン ST が記述されるラダー図の POU と同一です。

● インライン ST の制限仕様

項目	内容
1 回路あたりのインライン ST 数	1 つ

6-5-3　ST 言語

ST（Structured Text）言語とは、IEC 61131-3 で規定されている、工業制御（おもにプログラマブルコントローラ）用の高級言語です。
用意された制御文、演算子、関数により、ラダー図言語では記述しにくい数値演算処理を部分的に処理するのに適しています。
ST 言語には、以下の特長があります。

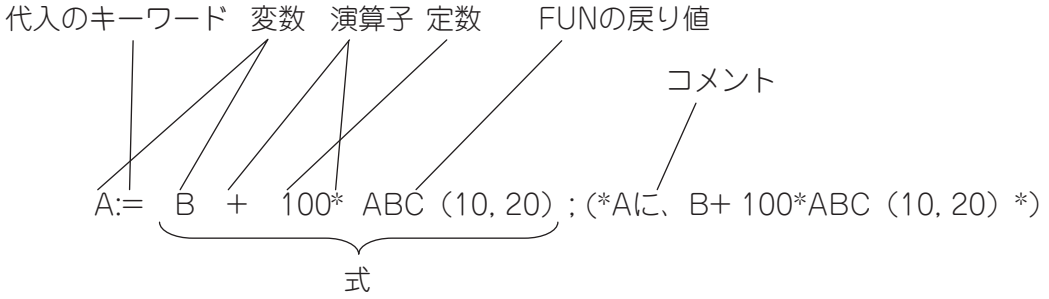
- ループ文や IF THEN ELSE などの制御文を、用意しています。
- C 言語などの高級言語と同様の記述や、漢字を含めた文字でのコメントにより、見やすいプログラムを作成できます。

```
1
2       // TableNo決定
3       FOR i:=0 TO ItemNum DO
4
5           IF (MinNo[i] <= ItemBox[i]) AND (ItemBox[i] <= MaxNo[i]) THEN   // 範囲内
6               TableNo[i] := ItemBox[i];
7               RangeOK[i] := TRUE;
8
9           ELSIF (ItemBox[i] > MaxNo[i]) THEN                              // 上限超え
10              TableNo[i] := MaxNo[i];
11              RangeOK[i] := FALSE;
12
13          ELSE                                                            // 下限超え
14              TableNo[i] := MinNo[i];
15              RangeOK[i] := FALSE;
16
17          END_IF;
18
19      END_FOR;
```

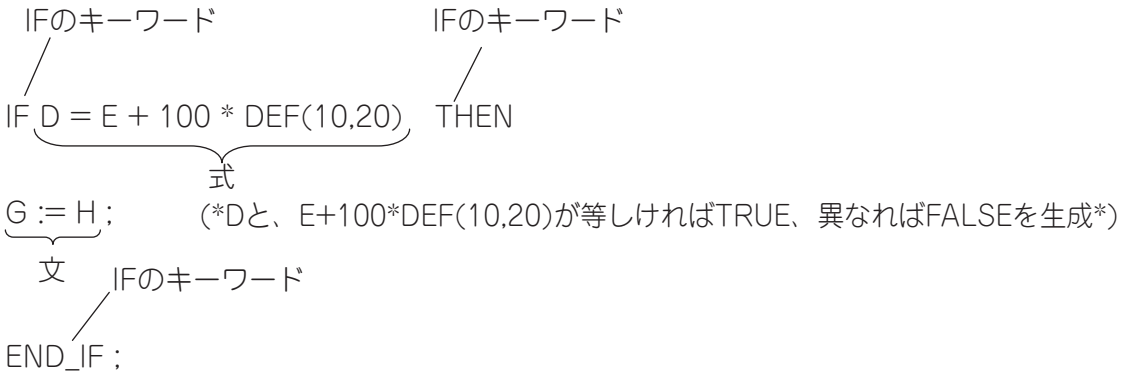
ST 言語の構成

ST 言語のプログラムは、1 つ以上の文から構成されます。1 つの文が、1 つの処理を表現します。
文が、上から下に 1 行ずつ順に実行されることにより、ST 言語で書かれたプログラム（ST プログラム）の処理が進んでいきます。
文は、キーワードと式で構成されます。キーワードとは、代入や実行制御などを示す、記号や文字列のことです。式とは、変数、定数、FUN の戻り値、またはそれらと演算子を組み合わせることによって、数値を生成する記述のことです。つまり、文は、それ単独で完結する処理を表し、式は生成した数値とキーワードを組み合わせることで文を構成します。

代入文の例：



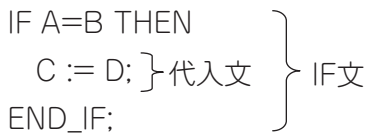
IF 文の例：



ST 言語の表現

● 文の区切り

- 文は、必ず、半角セミコロン（;）で終わります。単に改行しただけでは、文の終わりとはなりません。したがって、長い文を、複数行にわたって記述することが可能です。
- 1 つの文につき 1 つの半角セミコロン(;)を記述します。たとえば、以下では IF 文 1 つの中に代入文が 1 つ存在するので、それぞれの文の最後に半角セミコロン（;）を記述します。



● コメント

- プログラムの内容を、人が理解しやすくするために、プログラム中に任意のコメントを記述することができます。
- コメントとして記述した文は、実行されません。
- コメントの記述には、以下の 2 種類があります。

コメントの記述	例	備考
半角かっこと半角アスタリスク（* と *）で囲みます。	(*　複数行コメント IF ErrCode = 3 THEN 　Value := 1000; END_IF; ここまで　*)	複数行のコメントが記述できます。コメントの入れ子はできません。
半角スラッシュ 2 個 // で始まり、改行で終わります。	// コメント //　A :=　SIN（X）^2 ;	1 行ずつしかコメントアウトできません。

● スペース（空白）、改行、タブ

- スペース（空白）、改行、タブ、またはそれらの任意の組み合わせを、任意の数、任意の位置に記述してもかまいません。
  したがって、読みやすさを考慮して、演算子／キーワードと式の間を改行したり、文の前にスペースやタブを入力したりすることができます。
- 演算子／キーワードと変数名の間には、スペース（空白）、改行、タブを必ず入力してください。

例）■は、スペース（空白）、改行、タブなどを入力するべき場所です。

```
IF ■ A>0 ■ THEN ■ X:=10;
ELSE
    X:=0;
END_IF;
```

● 小文字／大文字、半角／全角

- 演算子／キーワードおよび変数名は、小文字、大文字を、区別しません（どちらでもかまいません）。
- 演算子／キーワードは、半角で入力してください。全角で入力すると、文法エラーとなります。

● 変数名の禁則文字

変数名の制約は、🕮「6-3-12 変数名などのプログラム関連の名称の制限」（P.6-77）を参照してください。

● 文字列

文字列の制約は、🕮「6-3-12 変数名などのプログラム関連の名称の制限」（P.6-77）を参照してください。

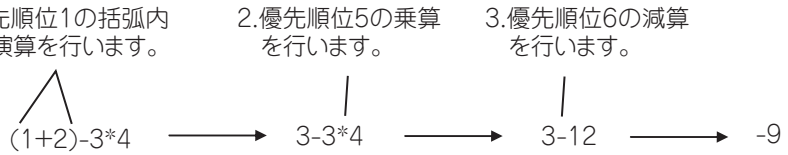
ST 言語のキーワードと演算子

● 文のキーワード

キーワード	意味	例
:=	代入	d := 10;
	FB/FUN の呼び出し	FBname（para1 := 10, para2 := 20）; 詳細は🕮「FB 呼出」（P.6-115）を参照
RETURN	戻り	
IF	選択	IF d < e THEN f := 1; ELSIF d = e THEN f :=2; ELSE f := 3; END_IF;
CASE	多方向分岐	CASE f OF 1: g :=11; 2: g :=12; ELSE g :=0; END_CASE;
FOR	繰り返し（1）	FOR i := 100 TO 1 BY -1 DO 　Val[i] := i; END_FOR;
WHILE	繰り返し（2）	WHILE Val < MaxVal DO 　Val := Val + 1; END_WHILE;
REPEAT	繰り返し（3）	REPEAT 　Val := Val + 1; UNTIL（ Val > 4 ） END_REPEAT;
EXIT	ループの中断	FOR i := 1 TO 100 DO 　FOR j := 1 TO 10 DO 　　IF Val[i, j]>100 THEN EXIT; 　　END_IF; 　END_FOR; END_FOR;
;	空文	Val[i] := i 　;（* 空文 *） WHILE（Var <>0）DO 　;（* 空文 *） END_WHILE;
(* テキスト *)	コメント	(* 複数行コメントアウト IF MyFun（ErrorCode）= 3 　THEN　ReturnValue := GetDetail（）; END_IF; ここまで *)
// テキスト	コメント	A := SIN（ X ）^ 2 + COS （ Y ）^2　+ 10; // A := SIN（ X ）^ 2 + COS （ Y ）^2　+ 5;

● 演算子

以下の表に、演算子の種類と優先順位を示します。
1 つの式に優先順位の異なる演算子が混在している場合、優先順位の高い演算を先に行います。1 つの式には、最大 64 個の演算子が記述できます。
例）X:=（1+2）-3*4; の場合、変数 X には -9 が代入されます。



演算	演算子	記述例と評価値	優先順位
括弧	()	(1+2) *（3+4） 値 21	1
FB、FUN の呼出		FUN1(FUN2(Var2A, Var2B), Var1B) 階層的に FB、FUN を呼び出した場合、下の階層が先に実行されます。上の例では、FUN2 が実行された後に FUN1 が実行されます。	2
符号	＋, －	+100 -100	3
否定	NOT	NOT TRUE 値 FALSE	
べき乗	**	-2**2 値 4 べき乗演算子よりマイナス符号の方が優先順位が上です。したがって上の例では、-2**2=(-2)**2 となるので、値は 4 になります。 2**3**2 値 64 べき乗演算子が複数重なった場合、左側から演算を行います。上の例では、2**3**2=(2**3)**2 となるので、値は 64 になります。	4
乗算	*	100*200 値 20000	5
除算	/	100/200 値 0.5	
剰余	MOD	10 MOD 7 値 3 -17 MOD 6 値 -5 -17 MOD （-6） 値 -5 17 MOD 6 値 5 17 MOD （-6） 値 5	
加算	＋	100+200 値 300	6
減算	－	100-200 値 -100	

演算	演算子	記述例と評価値	優先順位
比較	<, >, <=, >=	100<200 比較結果が正しければ値は TRUE, 正しくなければ値は FALSE になります。 上の例 100<200 は正しいので、値は TRUE になります	7
一致	=	100=200 2 つの値が一致すれば値は TRUE, 一致しなければ値は FALSE になります。 上の例 100=200 は一致しないので、値は FALSE になります	8
不一致	<>	100<>200 2 つの値が不一致なら値は TRUE, 不一致でなければ値は FALSE になります。 上の例 100<>200 は不一致なので、値は TRUE になります	
論理積	AND,&	1 ビットの論理積を、全ビット分実行します。 1 ビットの論理積の結果は、以下のようになります。 0 AND 0 = 0 0 AND 1 = 0 1 AND 0 = 0 1 AND 1 = 1 0101 AND 1100 値 0100	9
排他的論理和	XOR	1 ビットの排他的論理和を、全ビット分実行します。 1 ビットの排他的論理和の結果は、以下のようになります。 0 XOR 0 = 0 0 XOR 1 = 1 1 XOR 0 = 1 1 XOR 1 = 0 0101 XOR 1100 値 1001	10
論理和	OR	1 ビットの論理和を、全ビット分実行します。 1 ビットの論理和の結果は、以下のようになります。 0 OR 0 = 0 0 OR 1 = 1 1 OR 0 = 1 1 OR 1 = 1 0101 OR 1100 値 1101	11

使用上の注意

同じファンクションを階層的に記述すると、意図どおりの動作をしない場合があります。必ず、下の例のように文を分けて記述してください。

```
例 ) ×    out    := MyFunc( In1:=x1, In2:=MyFunc( In1:=x2, In2:=x3 ) );

     ○    tmp    := MyFunc( In1:=x2, In2:=x3 );
          out    := MyFunc( In1:=x1, In2:=tmp );
```

使用上の注意

演算子の優先順位は、規格やメーカの違いによって異なる場合があります。とくに、べき乗演算子の優先順位には、注意が必要です。意図どおりの順序で演算を行うため、括弧を用いて各演算の優先順位を明示することをお薦めします。
例） X:= -2**3**4; は、 X:= ((-2)**3)**4; と記述します。

参考

演算は、データ型に応じて行われます。
たとえば、整数型どうしの演算は、整数型で行われます。したがって、INT 型の変数 A = 3、B = 2 で A/B の演算を行うと、演算結果は 1.5 とはならずに、小数点以下が切り捨てられて 1 となります。よって、（A/B）*2 の演算結果は 3 とはならずに 2 となります。

● 各演算子のオペランドに使用可能なデータ型

1 つの演算子に使用されるオペランドがすべて同じデータ型である場合、以下のデータ型をオペランドに設定することができます。
ただし、演算子に異なるデータ型のオペランドが設定された場合、暗黙のキャストが発生します。
暗黙のキャストに関しては、🕮「暗黙のキャスト」（P.6-121）を参照してください。

データ型	代入演算子	引数設定演算子	数値演算子	剰余算演算子	べき算演算子	比較演算子	等価演算子	論理演算	正負記号
	:=	:= =>	+ - * /	MOD	**	< <= => >	= <>	NOT AND & OR XOR	+ -
BOOL 型	○	○	×	×	×	×	○	○	×
ビット列型	○	○	×	×	×	×	○	○	×
整数型	○	○	○	○	○*1	○	○	×	○
実数型	○	○	○	×	○	○	○	×	○
持続時間型	○	○	×	×	×	×	×	×	○
日付型	○	○	×	×	×	×	×	×	×
時刻型	○	○	×	×	×	×	×	×	×
日付時刻	○	○	×	×	×	×	×	×	×
文字列型	○	○	×	×	×	×*2	×*2	×	×
列挙型	○	○	×	×	×	×	○	×	×
構造体親	○	○	×	×	×	×	×	×	×
配列親	○	○	×	×	×	×	×	×	×

○：可能
×：ビルド時にエラーとなります

*1. Sysmac Studio のプロジェクトのデバイス選択画面にある［バージョン］に［Ver.1.15］以前を選択した場合、整数型の変数をオペランドに設定しても実数型として演算されます。演算結果に丸め誤差が含まれるときは、小数点以下が切り捨てられるため、意図した値にならないことがあります。小数点以下を丸める場合には、EXPT 命令と TO_**（整数変換グループ）命令を組み合わせて使用してください。
例）TO_INT(EXPT(X,Y))
*2. 文字列型変数の比較は、演算子ではなく、命令（EQascii など）で実行してください。

ST 言語の文

● 代入

概要：

左辺（変数）に、右辺（式の値）を代入します。

予約語：

:=

コロン（:）とイコール（=）の組み合わせ

文の構造：

```
＜変数＞:=＜式、変数、または定数＞;
```

用途：

変数に値を代入するときに、使用します。
たとえば、初期値の設定、計算結果の格納などに使用します。

説明：

＜式の値＞を、＜変数＞に、代入（格納）します。

例：

例 1）式 X+1 の結果を、変数 A に代入する

```
A:=X+1;
```

例 2）変数 B の値を、変数 A に代入する

```
A:=B;
```

例 3）定数 10 を、変数 A に代入する

```
A:=10;
```

留意事項：

- 代入元のデータ型は、代入先の変数のデータ型と一致しているか、暗黙のキャストで許可されたデータ型の組み合わせである必要があります。そうでなければ、ビルド時にエラーになります。
- 代入元が文字列の場合、代入先の STRING 型変数のサイズは、代入元の文字列のサイズより大きくしてください。そうでないと、エラーになります。
- 変数が STRING 型の場合、（左辺の変数サイズ）＞（右辺の変数に格納される文字列にサイズ）であれば代入可能です。
  例）
  以下の場合は代入可能です。
- 変数テーブル：

変数名	データ型	サイズ
Var1	STRING	10
Var2	STRING	20

- ユーザプログラム：
  Var2 :='ABC';
  Var1 := Var2;

共用体型変数全体に対する代入はできません。メンバ単位で代入してください。

● RETURN

概要：

ST を使用するプログラムの種類によって、以下のようになります。
< ST プログラム>
　途中で、ST プログラムを終了して、次のプログラムを実行します。
< FB のインスタンス内 FUN で使用する ST >
　途中で、FB、FUN を強制終了して、FB、FUN の呼出元の、次の命令を実行します。
<インライン ST >
　RETURN 文が存在するインライン ST を含む、POU を終了します。

予約語：

RETURN

文の構造：

```
RETURN;
```

用途：

実行中のプログラム、FB、または FUN を強制終了するときに使用します。

● IF（1 つの条件）

概要：

ある条件が成立したとき、指定した文を実行します。成立しないとき、別の文を実行します。
条件成立、不成立時は以下のように表現します。
　真：条件成立時
　偽：条件不成立時

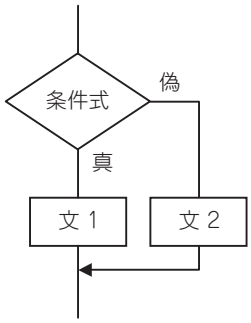
予約語：

IF、THEN、(ELSE)、END_IF
注：ELSE は省略可能。

文の構造：

```
IF ＜条件式＞ THEN
  ＜文 1>;
ELSE
  ＜文 2>;
END_IF;
```

処理フロー図：



用途：

ある 1 つの条件（条件式）が成立したか否かで場合分けするときに、使用します。

説明：

　＜条件式＞＝真のとき、＜文 1> を実行します。
　＜条件式＞＝偽のとき、＜文 2> を実行します。

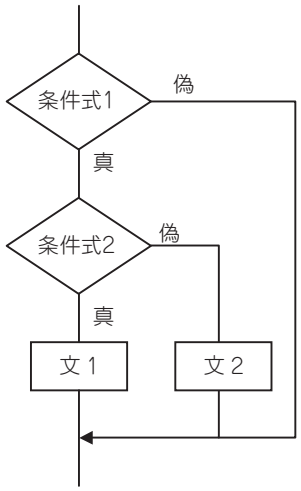
留意事項：

- IF は、必ず、END_IF と対で使用することが必要です。
- ＜条件式＞には、評価結果が真または偽の式（たとえば IF A>10）、BOOL 型変数（たとえば IF A）を記述することができます。
- ＜文 1>、＜文 2> に、複数行にわたってプログラムを記述できます。その場合、文と文との間を ; で区切ります。

例）＜文 1> に、さらに IF 文を入れた場合

```
IF ＜条件式 1>  THEN
   IF ＜条件式 2> THEN
              ＜文 1>;
   ELSE
              ＜文 2>;
              END_IF;
END_IF;
```

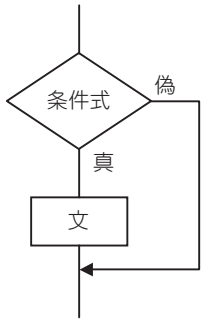
処理フロー図：



なお、ELSE は、上記のように、直前の THEN に対応します。

- ＜文 1>、＜文 2> の中は、ともに 1 つ以上の文を実行可能です。その場合、文と文の間を ; で区切ります。
- ELSE 文は省略可能です。省略した場合、＜条件式 > ＝偽のとき、何も実行しません。

処理フロー図：



例：

例 1）変数 A>0 が真のとき、変数 X に数値 10 を代入する。変数 A>0 が偽のとき、変数 X に数値 0 を代入する。

```
IF A>0 THEN
    X:=10;
ELSE
    X:=0;
END_IF;
```

例 2）変数 A>0 かつ変数 B>1 が真のとき、変数 X に数値 10 を代入し、変数 Y に数値 20 を代入する。変数 A>0 かつ変数 B>1 が偽のとき、変数 X に数値 0 を代入し、変数 Y に数値 0 を代入する。

```
IF A>0 AND B>1 THEN
    X:=10;Y:=20;
ELSE
    X:=0;Y:=0;
END_IF;
```

例 3）BOOL 型の変数 A = TRUE のとき、変数 X に数値 10 を代入する。変数 A = FALSE が偽のとき、変数 X に数値 0 を代入する。

```
IF A THEN X:=10;
ELSE X:=0;
END_IF;
```

● IF（複数の条件）

概要：

ある条件が成立したとき、指定した文を実行します。成立しないとき、さらに別の条件が成立したら、別の文を実行します。いずれも成立しないとき、さらに別の文を実行します。
条件成立、不成立時は以下のように表現します。

真：条件成立時
偽：条件不成立時

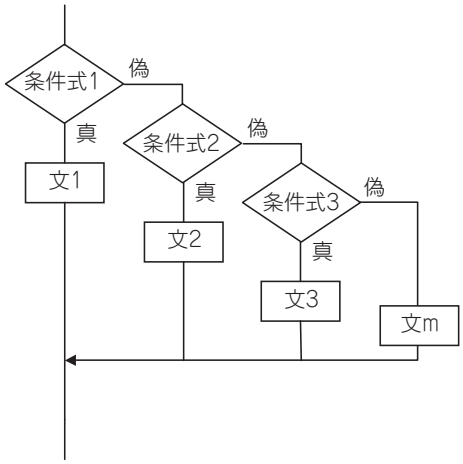
予約語：

IF、THEN、ELSIF、(ELSE)、END_IF
注：ELSE は省略可能。

文の構造：

```
IF ＜条件式 1＞ THEN ＜文 1＞;
   ELSIF ＜条件式 2＞ THEN ＜文 2＞;
   ELSIF ＜条件式 3＞ THEN ＜文 3＞;
         .
         .
         .
   ELSIF ＜条件式 n＞ THEN ＜文 n＞;
ELSE＜文 m＞;
END_IF;
```

処理フロー図：



用途：

ある複数の条件（条件式）が成立したか否かで場合分けするときに、使用します。

説明：

＜条件式 1＞＝真のとき、＜文 1＞を実行します。
＜条件式 1＞＝偽のとき、
＜条件式 2＞＝真なら、＜文 2＞を実行します。
＜条件式 2＞＝偽のとき、
＜条件式 3＞＝真なら、＜文 3＞を実行します。
.
.
.
＜条件式 n＞＝真なら、＜文 n＞を実行します。
いずれの条件にも合致しないとき、＜文 m＞を実行します。

留意事項：

- IF は、必ず、END_IF と対で使用することが必要です。
- ＜条件式□＞には、評価結果が真または偽の式を記述します。例）IF（A>10）
  なお、＜条件式□＞には、式でなく、BOOL 型の変数（FUN の BOOL 型戻り値を含む）だけを指定することもできます。その場合、変数の値が TRUE ＝評価結果真、FALSE ＝評価結果偽です。
- <文□>に、複数行にわたってプログラムを記述できます。その場合、文と文の間を;で区切ります。
- ELSE 文は省略可能です。省略した場合、いずれの条件にも合致しないとき、何も実行しません。

例：

変数 A>0 が真のとき、変数 X に数値 10 を代入する。
変数 A>0 が偽のとき、変数 B ＝ 1 が真なら、変数 X に数値 1 を代入する。
変数 A>0 が偽のとき、変数 B ＝ 2 が真なら、変数 X に数値 2 を代入する。
いずれの条件にも合致しないとき、変数 X に数値 0 を代入する。

```
IF A>0 THEN X:=10;
    ELSIF B=1 THEN X:=1;
    ELSIF B=2 THEN X:=2;
ELSE X:=0;
END_IF;
```

● CASE

概要：

整数式の値に一致する整数の選択値をもった文を実行します。

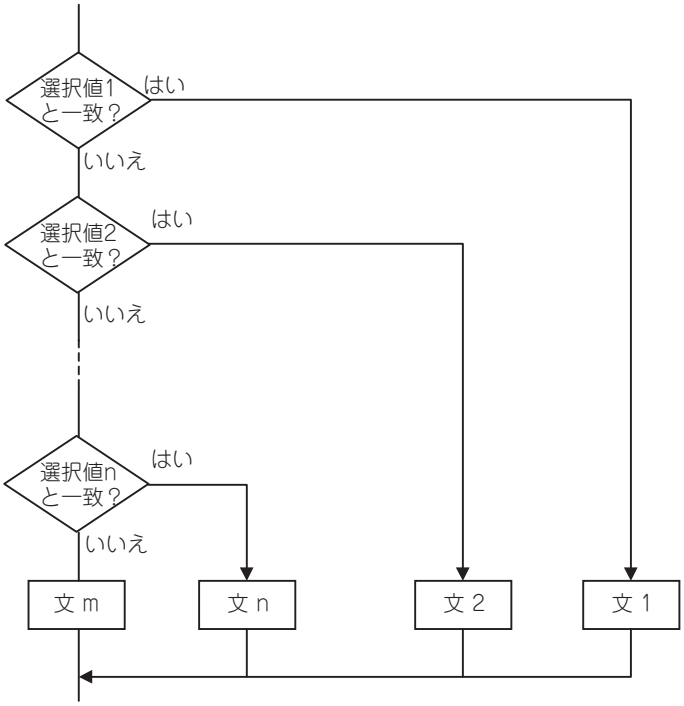
予約：

CASE

文の構造：

```
CASE < 整数式 > OF
    < 整数式の値 1>:< 文 1>;
    < 整数式の値 2>:< 文 2>;
            .
            .
            .
    < 整数式の値 n>:< 文 n>;
ELSE< 文 m>;
END_CASE;
```

処理フロー図：



用途：

ある整数の値によって場合分けするときに、使用します。

説明：

< 整数式 > が < 整数式の値 n> と一致したとき、< 文 n> を実行します。
< 整数式 > がいずれの < 整数式の値 n> とも一致しないとき、< 文 m> を実行します。

留意事項：

- CASE は、必ず、END_CASE と対で使用することが必要です。
- ＜整数式＞には、
  ・整数型や列挙型の変数　　（例：abc）
  ・整数式　　（例：abc+def）
  ・整数値を戻り値とする FUN　　（例：xyz（））
  を記述します。
- <文□>に、複数行にわたってプログラムを記述できます。その場合、文と文の間を;で区切ります。
- ＜整数式の値 n> で、複数の整数の OR 論理を指定するときは、数値を ,（コンマ）で区切ります。連続する整数を指定するときは、開始整数と終了整数の間を ..（ピリオドを 2 つ）で区切ります。
  例 1）特定の整数値に条件を指定することや、複数の整数値に同一の条件を指定することができます。

```
CASE A OF
    1: X:=1;   ---------------  変数Aが1のとき、変数Xに数値1を代入する
    2: X:=2;   ---------------  変数Aが2のとき、変数Xに数値2を代入する
    3: X:=3;   ---------------  変数Aが3のとき、変数Xに数値3を代入する
ELSE
    X:=0;      ---------------  いずれにも一致しないときは、変数Xに0を代入する
END_CASE;
```

```
CASE A OF
    1: X:=1;   ---------------  変数Aが1のとき、変数Xに数値1を代入する
    2,5: X:=2; ---------------  変数Aが2または5のとき、変数Xに数値2を代入する
    6..10: X:=3; -------------  変数Aが6～10のいずれかのとき、変数Xに数値3を代入する
    11,12,15..20: X:=4; ------  変数Aが11、12または15～20のとき、変数Xに数値4を代入する
ELSE
    X:=0;      ---------------  いずれにも一致しないときは、変数Xに0を代入する
END_CASE;
```

例 2）＜整数式＞には、整数型の変数・式・FUN の戻り値、および列挙型の変数・FUN の戻り値を記述できます。以下に、例を示します。

- 整数・列挙型変数の例

```
CASE ColorVar OF
    RED:
        X := 0;
    BLUE:
        X := 1;
    ELSE
        X := 2;
END_CASE;
```

- 整数式の例

```
CASE （a1 + a2）OF
    0:
        X := 0;
    1:
        X := 1;
    ELSE
        X := 2;
END_CASE;
```

- 整数・列挙型 FUN の戻り値の例

```
CASE FUN( ) OF
    0:  ---------------------  FUN( )の戻り値によって分岐させる
        X := 10;
    1:
        X := 11;
    ELSE
        X := 12;
END_CASE;
```

CASE に使用可能なデータ型

分類	データ型		＜整数式＞
基本データ型	整数型		○
	ブール型、ビット列型、実数型、持続時間型、日付型、時刻型、日付時刻型、文字列型		×
データ型の指定	配列指定	配列全体	×
		要素	○　整数型 , 列挙型だけ可能
派生データ型	構造体型	構造体全体	×
		メンバ	○　整数型 , 列挙型だけ可能
	共用体型	共用体全体	×
		メンバ	○　整数型 , 列挙型だけ可能
	列挙型		○

● FOR

概要：

変数（以下これを「反復変数」と呼ぶ）がある値からある値になるまで、ある文を繰り返して実行します。
条件成立、不成立時は以下のように表現します。
　真：条件成立時
　偽：条件不成立時

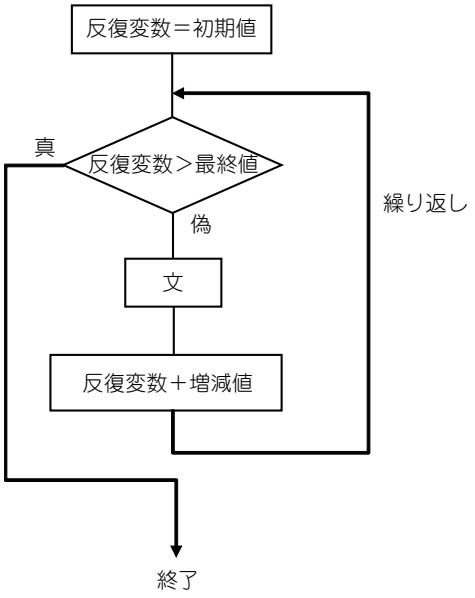
予約語：

FOR、TO、(BY)、DO、END_FOR
注：BY は省略可能。

文の構造：

```
FOR ＜反復変数＞:= ＜初期値＞ TO ＜最終値＞ BY ＜増減値＞ DO
    ＜文＞;
END_FOR;
```

処理フロー図：



用途：

繰り返し回数があらかじめわかっている場合に使用します。
特に、ある反復変数の値に応じて、配列指定の変数の各要素を切り替えて指定する場合に、有効です。

説明：

＜初期値＞、＜最終値＞、＜増減値＞を評価し決定します。
＜反復変数＞が＜初期値＞のとき、＜文＞を実行します。
実行後、＜反復変数＞に＜増減値＞で与えられる値を加算し、＜反復変数＞＜＜最終値＞の値であれば、＜文＞を実行します。
実行後、＜反復変数＞に＜増減値＞で与えられる値を加算し、＜反復変数＞＜＜最終値＞の値であれば、＜文＞を実行します。
これを繰り返します。
＜反復変数＞＞＜最終値＞となると、終了します。
なお、＜増減値＞が負数の場合は、大小比較記号の向きが逆になります。

留意事項：

- 反復変数が符号あり整数型の場合、＜増減値＞に、マイナスの値を指定することができます。
- FOR は、必ず END_FOR と対で使用します。
- 反復変数では最終値で処理を実行した後、（最終値 + 増減値）となり、反復処理を終了します。
  例）以下のような ST 文では FOR 文終了時に i の値は 101 となっています。

```
FOR i:=0 TO 100 DO
    X[i]:=0;
END_FOR;
// この時点で i は 101 となる
```

- FOR 文の内側で反復変数を直接変更するような処理は記述しないでください。予期せぬ動作となる場合があります。
  例）

```
FOR i:=0 TO 100 BY 1 DO
    X[i]:=0;
    i:=i+INT#5;
END_FOR;
```

- ＜文＞に、複数行にわたってプログラムを記述できます。その場合、文と文の間を ; で区切ります。
- BY＜増減値＞は省略可能です。省略した場合、BY は 1 として実行されます。
- ＜初期値＞、＜最終値＞、＜増減値＞には、整数型（SINT、INT、DINT、LINT、USINT、UINT、UDINT、または ULINT）の変数、または整数値を指定することが可能です。なお、FUN の整数型戻り値も指定可能です。
  例 1）反復変数 n = 0 から 5 つおきに 50 まで、繰り返し、配列指定の変数 SP[n] に 100 を代入します。

```
FOR n := 0 TO 50 BY 5 DO
    SP[n] := 100;
END_FOR;
```

例 2）配列指定の変数 DATA[n] の要素の DATA[1] から DATA[50] の合計値を算出し、変数 SUM に代入します。

```
IF a THEN
    FOR n := 0 TO 50 BY 1 DO
        DATA[n]:= 1 ;
    END_FOR;

    FOR n := 0 TO 50 BY 1 DO
        SUM:= SUM + DATA[n] ;
    END_FOR;

    a:=FALSE;
END_IF;
```

例 3）配列指定の変数 DATA[n] の要素の DATA[1] から DATA[50] の中の最大値と最小値を検索します。最大値を変数 MAX に代入し、最小値を変数 MIN に代入します。なお、DATA[n] の値は、0 ～ 1000 の範囲内とします。

```
MAX :=0;
MIN :=1000;
FOR n :=1 TO 50 BY 1 DO
    IF DATA[n] > MAX THEN
        MAX :=DATA[n];
    END_IF;
    IF DATA[n] < MIN THEN
        MIN :=DATA[n];
    END_IF;
END_FOR;
```

- <反復変数>に<初期値>から<増減値>を加算し、<最終値>を超えるまでにFOR文を含むプログラムの実行時間がタスク周期設定時間を超えた場合、タスク周期オーバーが発生します。
  - ロジック上 < 反復変数 > が < 最終値 > に到達しない場合

  例）

```
FOR i := 0 TO 100 BY 1 DO
    intArray[i] := i;
    i := INT#50;  ----------
END_FOR;
```

無限ループとなり、タスク周期オーバーが発生します

  例）

```
FOR i := 0 TO 100 BY 0 DO  --------
    ;
END_FOR;
```

無限ループとなり、タスク周期オーバーが発生します

- ＜反復変数＞が＜最終値＞を超えた時点でオーバーフロー / アンダーフローが発生する場合
  例）
  変数 i が USINT 型の場合、最終値 256 が表現できます。0 とみなし無限ループとなります。

```
FOR i := 0 TO 254 BY 2 DO
    INTArray[i] := i;
END_FOR;
```

参考

＜最終値＞や＜増減値＞には、演算式を記述することが可能です。
ただし、＜最終値＞や＜増減値＞が評価されるのは、FOR ループ内の演算が行われる前です。FOR ループ内の演算が開始された後に、＜最終値＞や＜増減値＞が変化することはありません。
下図の場合、＜最終値＞は 10、＜増減値＞は 3 となります。FOR ループ内の演算が開始され、変数 A や変数 C の値が変わっても、＜最終値＞は 10 のまま、＜増減値＞は 3 のままです。

```
A := INT#1;
B := INT#2;
C := INT#10;

FOR i := 0 TO C BY A+B DO
    INTArray[i] := i;
    A := B + i;
    C := C + i;
END_FOR;
```

FOR に使用可能なデータ型

分類	データ型		＜反復変数＞、＜初期値＞、＜最終値＞、＜増減値＞[*1]
基本データ型	ブール型、ビット列型、実数型、持続時間型、日付型、時刻型、日付時刻型、文字列型		×
	整数型		○
データ型の指定	配列指定	配列全体	×
		要素	○　整数型 , 列挙型だけ可能[*2]
派生データ型	構造体型	構造体全体	×
		メンバ	○　整数型 , 列挙型だけ可能[*2]
	共用体型	共用体全体	×
		メンバ	×
	列挙型		○[*2]

*1. ＜反復変数＞、＜最終値＞、＜増減値＞は 3 種ともデータ型を一致させてください。データ型が一致していないと、Sysmac Studio のプログラムビルド時にエラーとなります。
*2. ＜反復変数＞、＜最終値＞、＜増減値＞には、列挙型を使用できません。

● WHILE

概要：

ある条件式が真である間、指定した文を繰り返して実行します。

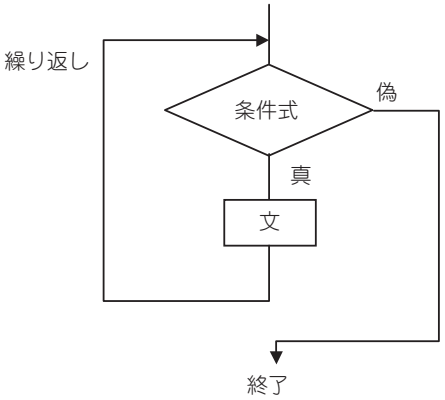
予約語：

WHILE、DO、END_WHILE

文の構造：

```
WHILE ＜条件式＞ DO
    ＜文＞;
END_WHILE;
```

処理フロー図：



用途：

繰り返し回数がわかっていない（条件によって何回になるかわからない）場合で、ある条件式が成立している間、ある処理を繰り返すときに使用します。
条件式が成立しているときだけ、ある処理を実行するときに使用します（前判定の繰り返し）。

説明：

＜文＞の実行前に、＜条件式＞が評価されます。
＜条件式＞が真の場合、＜文＞を実行します。その後、＜条件式＞が再度評価されます。これを繰り返し実行します。
＜条件式＞が偽の場合、＜文＞を実行せず、＜条件式＞の評価を終了します。

留意事項：

- WHILE は、必ず、END_WHILE と対で使用することが必要です。
- ＜文＞の実行前に、＜条件式＞が偽の場合、＜文＞を実行せず終了します。
- ＜文 1>、＜文 2> に、複数行にわたってプログラムを記述できます。その場合、分と分との間を ; で区切ります。
- ＜文＞の中で、1 つ以上の文を実行可能です。その場合、文と文の間を ; で区切ります。
- ＜条件式＞には、式でなく、BOOL 型の変数（FUN の BOOL 型戻り値を含む）を指定することもできます。

例 1）7 の倍数で 1000 を超える数を算出し、変数 A に代入します。

```
A := 0;
WHILE A <= 1000 DO
    A := A+INT#7;
END_WHILE;
```

例 2）変数 X ＜ 3000 のあいだ、X の値を 2 倍し、その値を配列指定の変数 DATA[1] に代入します。次にさらに X の値を 2 倍し、その値を配列指定の変数 DATA[2] に代入します。これを繰り返します。

```
n := 1;
X := 1;
WHILE X < 3000 DO
    X:= X*INT#10#2;
    DATA[n]:= X;
    n := n+INT#1;
END_WHILE;
```

- 正しい＜条件式＞を書かなかった場合、プログラム実行時間が延び、タスク周期オーバーを発生させる要因となります。

例）

```
boolVar := TRUE;
WHILE boolVar DO
intVar := intVar + INT#1;
END_WHILE;
```

● REPEAT

条件成立、不成立時は以下のように表現します。
　真：条件成立時
　偽：条件不成立時

概要：

ある文を実行し、ある条件式が真になるまで、その文を繰り返します。

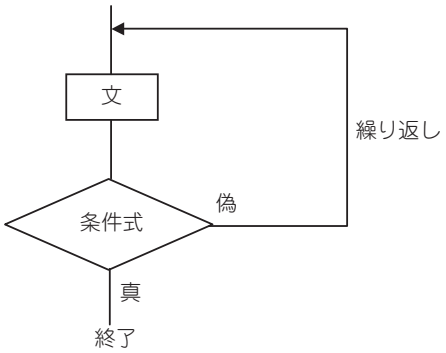
予約語：

REPEAT、UNTIL、END_ REPEAT

文の構造：

```
REPEAT
    ＜文＞;
UNTIL ＜条件式＞
END_REPEAT;
```

処理フロー図：



用途：

繰り返し回数がわかっていない（条件によって何回になるかわからない）場合で、ある処理後の条件式が成立している間、その処理を繰り返すときに使用します。
ある処理を実行した結果に応じて、繰り返すかどうかを判断するときに使用します（後判定の繰り返し）。

説明：

＜文＞が最初、無条件に、実行されます。その後、＜条件式＞が評価されます。
＜条件式＞が偽の場合、＜文＞を再度実行します。
＜条件式＞が真の場合、＜文＞を実行せず終了します。

留意事項：

- REPEAT は、必ず、END_REPEAT と対で使用することが必要です。
- ＜文＞の実行前に、＜条件式＞が真の場合でも、＜文＞が実行されます。
  つまり、＜文＞は、少なくとも 1 回は実行されます。
- ＜文＞に、文のキーワードに定める文を複数行にわたって記述できます。その場合、文と文の間を ; で区切ります。
- ＜条件式＞には、式でなく、BOOL 型の変数（FUN の BOOL 型戻り値を含む）を指定することもできます。

例）　1 から 10 までの数値を加算し、変数 TOTAL に代入します。

```
A := 1;
TOTAL := 0;
REPEAT
    TOTAL := TOTAL + A;
    A := A+INT#1;
UNTIL A>10
END_REPEAT;
```

- 正しい＜条件式＞を書かなかった場合、プログラム実行時間が延び、タスク周期オーバーを発生させる要因となります。

  例）

```
intVar := INT#1;
REPEAT
    intVar := intVar + INT#1;
UNTIL intVar = INT#0
END_REPEAT;
```

● EXIT

概要：

繰り返し文（FOR 文、WHILE 文、REPEAT 文）の中だけで使用し、繰り返し文を終了させます。
IF 文の中で使用すると、ある条件が成立したとき、繰り返し文を途中で抜けることができます。

予約語：

EXIT

文の構造（例：IF 文の中での使用）：

```
FOR（WHILE、REPEAT）文
          .
          .
          .
IF< 条件式 > THEN EXIT;
END_IF;
          .
          .
          .
END_FOR（WHILE、REPEAT）;
```

用途：

終了条件が成立する前に、繰り返し処理を強制的に終了するときに、使用します。

説明（例：IF 文の中での使用）：

＜条件式＞が真の場合、繰り返し文（FOR 文、WHILE 文、REPEAT 文）を強制終了し、EXIT 以降に記述されている文は実行されません。
（注）1 ＜条件式＞には、式でなく、BOOL 型の変数だけを指定することもできます。
2 ＜文＞の実行前に、＜条件式＞が真の場合でも、＜文＞が実行されます。

例：

例 1）変数 n = 1 から 1 つおきに 50 まで、繰り返し、配列指定の変数 DATA[n] に n を加算します。ただし、DATA[n] が 100 を超えたら、終了します。

```
IF A THEN
    DATA[3] :=98;
    FOR n := 1  TO 50 BY 1 DO
        DATA[n] := DATA[n] + n;
        IF DATA[n] > 100 THEN EXIT;
        END_IF;
    END_FOR;
    A :=FALSE;
END_IF;
```

● FB 呼出

概要：

FB を呼び出します。

予約語：なし

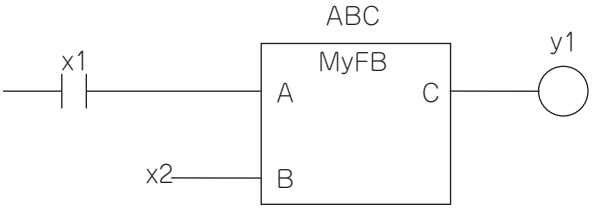
文の構造：

FB のインスタンス名の後に、( ) 内に、引数指定（指定した変数の値を、呼出先の FB の入力変数に渡す指定）と戻り値指定（呼出先の FB の出力変数の値を、指定した変数に受ける指定）を記述します。
以下の①または②のいずれかの記述方式が可能です。
ユーザプログラムの可読性から①を推奨します。

記述方式①：

呼び出す FB の変数名と呼出元のパラメータ名を、いずれも記述する場合
ABC（A:=x1, B:=x2, C=>y1）;
ABC　　：FB のインスタンス名
A、B　　：呼び出す FB の入力変数名または入出力変数名
x1、x2　：呼出元の入力パラメータまたは入出力パラメータ（定数を含む）
C　　：呼び出す FB の出力変数名
y1　　：呼出元の出力パラメータ

- ラダー図言語での表現

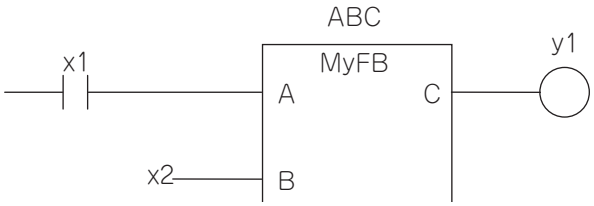


- 各引数や戻り値の記述順序は任意です。
- 入力変数名と入力パラメータ名は省略可能です。省略した場合、前回呼出時に入力変数に代入された値が、入力変数に代入されます。初めて呼び出された場合は、初期値が入力変数に代入されます。
- 出力変数と出力パラメータは省略可能です。省略した場合、出力変数の値は、どこにも代入されません。

記述方式②：

呼び出す FB の変数名を省略し、呼出元のパラメータ名だけを記述する場合
ABC（x1, x2, y1）;
ABC　　：FB のインスタンス名
A、B　　：省略します。（呼び出す FB の入力変数名または入出力変数名）
x1、x2　：呼出元の入力パラメータまたは入出力パラメータ（定数を含む）
C　　：省略します。（呼び出す FB の出力変数名または定数）。
y1　　：呼出元の出力パラメータ

- ラダー図言語での表現



- 記述するパラメータの順序は、FB の定義で決まっています。FB のローカル変数定義の上から下の順序です。

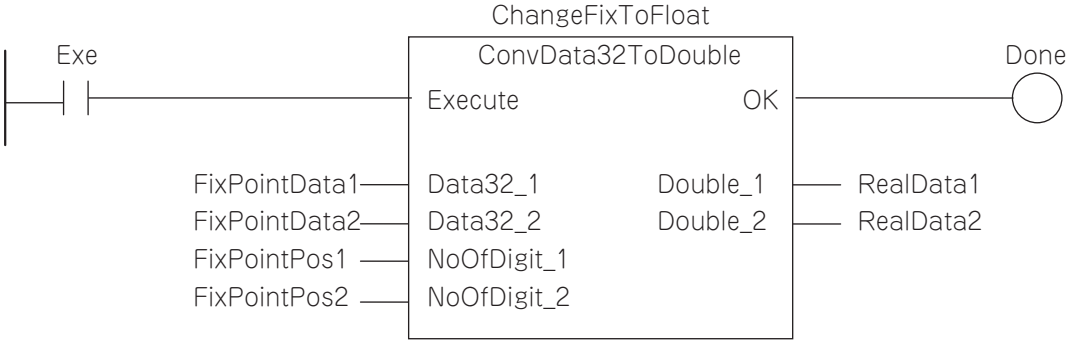
用途：
FB を呼び出します。

例
- プログラム

- 記述①
  ChangeFixToFloat（Execute:=Exe,Data32_1:=FixPointData1,
  Data32_2:=FixPointData2,
  NoOfDigit_1:=FixPointPos1,
  NoOfDigit_2:=FixPointPos2,OK=>Done,Double_1=>RealData1,
  Double_2=>RealData2）;

- 記述②
  ChangeFixToFloat（Exe, FixPointData1, FixPointData2, FixPointPos1, FixPointPos2,
  Done, RealData1, RealData2）;

- ラダー図言語での表現



- ファンクションブロックの定義
  ファンクションブロック名：ConvData32ToDouble
  ファンクションブロックの変数

入力／出力	変数名	データ型
入力変数	Execute	BOOL
	Data32_1	DINT
	Data32_2	DINT
	NoOfDigit_1	INT
	NoOfDigit_2	INT
出力変数	OK	BOOL
	Double_1	LREAL
	Double_2	LREAL

- プログラムの変数

変数名	データ型	コメント
ChangeFixToFloat	ConvData32ToDouble	固定小数点→浮動小数点変換
Exe	BOOL	起動トリガ
FixPointData1	DINT	小数点位置指定データ #1
FixPointPos1	INT	小数点以下桁数 #1
FixPointData2	DINT	小数点位置指定データ #2
FixPointPos2	INT	小数点以下桁数 #2
Done	BOOL	正常終了
RealData1	LREAL	浮動小数点データ #1
RealData2	LREAL	浮動小数点データ #2

パラメータの省略の可否

FB 呼び出し時に、パラメータに値を設定する必要がない場合は、記述するパラメータを省略することが可能です。
パラメータ省略の可否は以下のとおりです。

○： 記述可能。省略した場合はパラメータの代わりに初期値が代入されます。
×： ビルド時にエラーとなります。

POU の種類	呼び出す POU 側の変数	記述パターン		記述の可否
		記述するパラメータ	例	
FB	記述あり（記述方式①）	すべてのパラメータを記述	instance（x:=a,y:=b,z:=c）;	○
		パラメータを複数個記述	instance（x:=a,y:=b）;	
		パラメータを 1 個記述	instance（y:=b）;	
		パラメータ記述なし	instance（x:=）;	×
	記述なし（記述方式②）	すべてのパラメータを記述	instance（a,b,c）;	○
		すべてのパラメータなし	instance（）;	
		先頭パラメータだけを記述	instance（a）;	×
		パラメータを 1 個記述	instance（a, , ）;	
		パラメータを複数個記述	instance（a,b）;	

● FUN 呼出

概要：

FUN を呼び出します。

予約語：なし

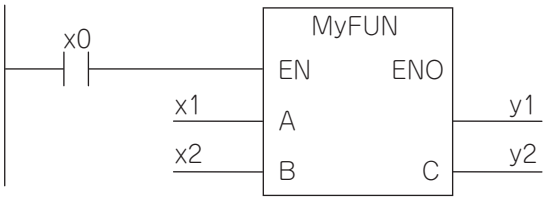
文の構造：

代入のキーワード（:=）の左辺に戻り値を代入する出力パラメータを記述します。右辺には、FUN 名の後の（ ）内に、引数指定（指定した変数の値を、呼出先の FUN の入力変数に渡す指定）を記述します。
以下の①または②のいずれかの記述方式が可能です。
ユーザプログラムの可読性から①を推奨します。

記述方式①：

```
IF （x0=TRUE）THEN
  y1 := MyFUN（A:=x1, B:=x2, C=>y2）;
END_IF;
```

- ラダー図言語での表現



MyFUN :FUN 名
x0 :FUN の呼び出し有無
A, B : 呼び出す FUN の入力変数名
x1, x2 : 呼び出す FUN の入力パラメータ
C : 呼び出す FUN の出力変数名
y1 : 呼び出す FUN の戻り値格納先
y2 : 呼び出す FUN の出力パラメータ

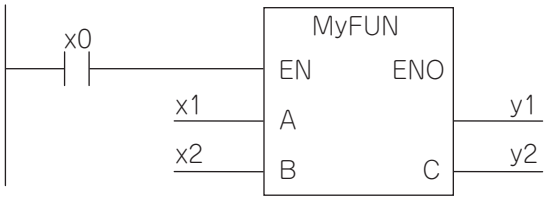
- 各引数の記述順序は任意です。
- 入力変数名と入力パラメータ名は省略可能です。省略した場合、入力変数には初期値が代入されます。
- EN も省略可能です。省略した場合は、EN に TRUE が代入されます。

記述方式②：

呼び出す FUN の変数名を省略し、呼出元のパラメータ名だけを記述する場合

```
IF （x0=TRUE）THEN
  y1 := MyFUN（x1, x2, y2）;
END_IF;
```

- ラダー図言語での表現



MyFUN　:FUN 名
x0　　:FUN の呼び出し有無
A, B　　: 呼び出す FUN の入力変数名
x1, x2　: 呼び出す FUN の入力パラメータ
C　　: 呼び出す FUN の出力変数名
y1　　: 呼び出す FUN の戻り値格納先
y2　　: 呼び出す FUN の出力パラメータ

- 記述するパラメータの順序は、FUN の定義で決まっています。FUN のローカル変数定義の上から下の順序です。

例

- プログラム

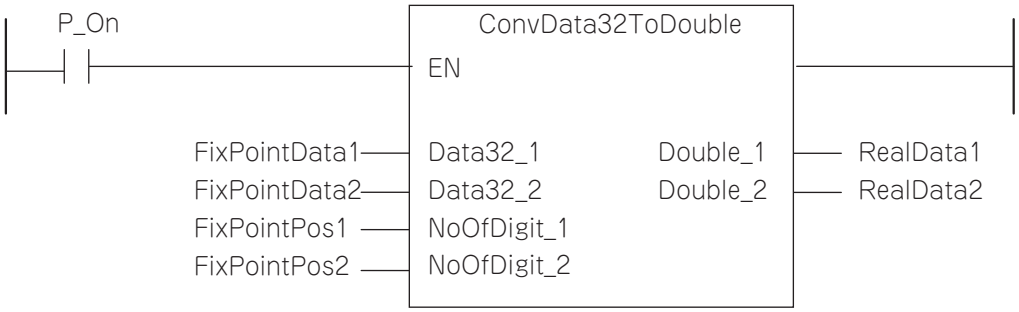
- 記述①

```
ConvData32ToDouble（Data32_1:=FixPointData1,Data32_2:=FixPointData2,
NoOfDigit_1:=FixPointPos1, NoOfDigit_2:=FixPointPos2,
Double_1=>RealData1, Double_2=>RealData2）;
```

- 記述②

```
ConvData32ToDouble（FixPointData1, FixPointData2, FixPointPos1, FixPointPos2,
RealData1, RealData2）;
```

- ラダー図言語での表現



• ファンクションの定義
ファンクション名：ConvData32ToDouble
ファンクションの変数

In/Out	変数名	データ型
入力変数	Execute	BOOL
	Data32_1	DINT
	Data32_2	DINT
	NoOfDigit_1	INT
	NoOfDigit_2	INT
出力変数	Double_1	LREAL
	Double_2	LREAL
戻り値	-	BOOL

• プログラムの変数

変数名	データ型	コメント
ChangeFixToFloat	ConvData32ToDouble	固定小数点→浮動小数点変換
Exe	BOOL	起動トリガ
FixPointData1	DINT	小数点位置指定データ #1
FixPointPos1	INT	小数点以下桁数 #1
FixPointData2	DINT	小数点位置指定データ #2
FixPointPos2	INT	小数点以下桁数 #2
Done	BOOL	正常終了
RealData1	LREAL	浮動小数点データ #1
RealData2	LREAL	浮動小数点データ #2

用途：
FUN を呼び出します。

パラメータの省略の可否
FUN 呼び出し時に、パラメータに値を設定する必要がない場合は、記述するパラメータを省略することが可能です。
パラメータ省略の可否は以下のとおりです。

○： 記述可能。省略した場合はパラメータの代わりに初期値が代入されます。
×： ビルド時にエラーとなります。

POU の種類	呼び出す POU 側の変数	記述パターン		記述の可否
		記述するパラメータ	例	
FUN	記述あり（記述方式①）	すべてのパラメータを記述	FUN（x:=a,y:=b,z:=c）;	○
		パラメータを複数個記述	FUN（x:=a,y:=b）;	
		パラメータを 1 個記述	FUN（y:=b）;	
		パラメータ記述なし	FUN（x:=）;	×
	記述なし（記述方式②）	すべてのパラメータを記述	FUN（a,b,c）	○
		パラメータ記述なし	FUN（）;	
		先頭パラメータだけを記述	FUN（a）;	×
		パラメータを 1 個記述	FUN（a, , ）;	
		パラメータを複数個記述	FUN（a,b）;	

ST 言語に関する注意

ST 言語でユーザプログラムを作成する際に、注意すべき事項を示します。

● 暗黙のキャスト

オペランドのデータ型が以下のように一致しない場合、暗黙のキャストルールによって、データ型が自動的に変換されて処理されます。
また、暗黙のキャストルールを満たさない場合は、ビルド時にエラーとなります。

(1) 代入文の右辺の式内で、オペランドのデータ型が一致していない場合

例）

```
A: = INT#10 + SINT#2;
```

(2) 代入文の右辺と左辺のオペランドのデータ型が一致していない場合

例）

```
   A: = B + C;
  /      \
REAL     INT
```

(3) 構文の式内で、オペランドのデータ型が一致していない場合

例）

```
      INT   LINT
        \   /
CASE A+B OF
   INT#1:
      def:=INT#10;
```

以下に、これら３つの場合のキャストルールを説明します。

代入文の右辺で演算する場合のキャストルール

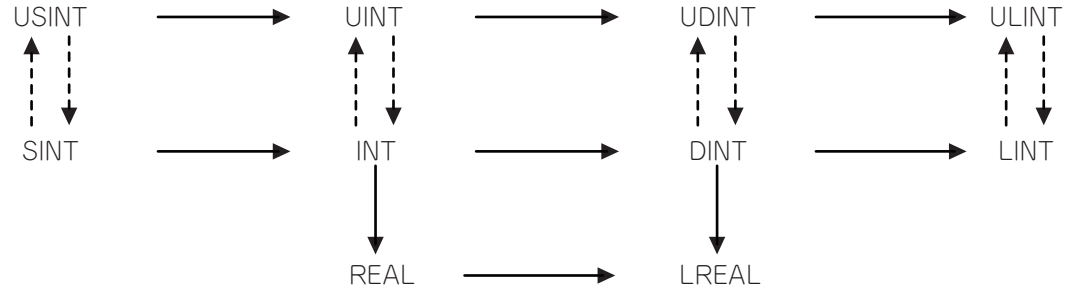
- 右辺のオペランドは、演算子のオペランドに使用可能なデータ型で許可されたデータ型であれば、いずれでも混在可能です。
- 右辺のオペランドのうち、最大ランクを持つデータ型が、右辺のデータ型となります。
（データ型のランクは、下記の [データ型のランク表] を参照してください）。

[データ型のランク表]：
ランクが大きいほど、そのデータ型が表現できる数値の範囲が大きいことを意味します。

ランク	データ型
1	SINT
2	USINT
3	INT
4	UINT
5	DINT
6	UDINT
7	LINT
8	ULINT
9	REAL
10	LREAL
11	BYTE
12	WORD
13	DWORD
14	LWORD

代入文の右辺の値を左辺に代入する場合のキャストルール

以下の図において、代入元の型から代入先の型に矢印が繋がっている場合は、キャストされます。繋がっていない場合はビルド時にエラーとなります。



→ 代入時、数値の符号と絶対値は変化しません。

⇢ 代入時、数値の符号や絶対値が変化する場合があります。

例）
```
intVar := -1;          (* intVar := 16#FFFF *)
uintVar := 1;
uintVar := intVar;     (* uintVar := 16#FFFF、
                          すなわち-1 を代入したが結果は 65535となる*)
```

矢印は、直接つながっていなくても代入可能です。
たとえば、SINT → USINT → UINT → UDINT → ULINT とつながっているので、ULINT:= SINT は記述可能です。

使用上の注意

UDINT → ULINT、DINT → LINT、DINT → LREAL のキャストの場合、以下のことに注意してください。
これらはすべて、データサイズ 32 ビットのデータ型から 64 ビットのデータ型へのキャストです。この場合、代入文の右辺の演算結果が、32 ビットで表現できる値の範囲を超えると、正しい数値が代入されないことがあります。

例）下のユーザプログラムでは、3 行目の加算結果が 32 ビットで表現できる値の範囲を超えます。その結果、オーバーフローが発生し、LintVar には 0 が代入されます。

```
UdintVar := UDINT#16#FFFF_FFFF;              // 32 ビットの上限値
DintVar  := DINT#1;                          // 1
LintVar  := (UdintVar + DintVar) / DINT#2;   // (32 ビットの上限値＋ 1) / 2
```

このような場合には、あらかじめ各数値を 64 ビットのデータ型に変換してから演算を行ってください。上の例は、以下のように変更します。

```
LintTmp1 := UDINT_TO_LINT(UDINT#16#FFFF_FFFF);  // UDINT → LINT 変換
LintTmp2 := DINT_TO_LINT(DINT#1);               // DINT → LINT 変換
LintVar  := (LintTmp1 + LintTmp2) / DINT#2;
```

構文の式内でのキャストルール

構文の式においても、代入文の右辺の演算式や、右辺から左辺へ代入時の暗黙のキャストルールが、同様に適用されます。

例）

```
CASE （A+B+C）OF
    Result1:
        ～;
    ResultN:
        ～;
END_CASE;
```

● FUN の実行順序

FUNを含む式では、FUNの実行順序は一意には決まりません。NYシリーズコントローラやSysmac Studio のバージョンや、記述のしかたによって、FUN の実行順序は異なります。したがって、以下のように FUN の実行順序によって結果が異なる可能性がある式は、後述の方法で記述してください。

- 同一のグローバル変数にアクセスする FUN を複数含む式
- FUN と、その FUN の実行によって値が変化する変数を含む式

同一のグローバル変数にアクセスする FUN を複数含む式

下式の場合、3 つの FUN の実行順序は、演算子の優先順位によって決まる演算の順序とは無関係です。したがって、3 つの FUN が FUN2、FUN3、FUN1 の順に実行されるとは限りません。

```
result := FUN1() + FUN2() * FUN3();
```

上式の 3 つの FUN が、同一のグローバル変数に、書込を含むアクセスをする FUN である場合、FUN の実行順序によって変数「result」の値が異なる可能性があります。
このような場合は、3 つの FUN の実行順序を一意にするため、式を分けて記述します。FUN2、FUN3、FUN1 の順で実行させたい場合は、以下のように記述してください。

```
tmp2  := FUN2();
tmp3  := FUN3();
result := FUN1() + tmp2 * tmp3;
```

FUN と、その FUN の実行によって値が変化する変数を含む式

下式には、FUN と、その FUN の実行によって値が変化する変数が含まれています。

```
result := varA + FUN4(out => varA);
```

上式の場合、右辺第 1 項の変数「varA」の評価が、FUN4 よりも先に実行されるとは限りません。したがって、「varA」の評価と FUN4 の実行順序によって、変数「result」の値が異なる可能性があります。
このような場合は、「var4」の評価と FUN4 の実行順序を一意にするため、式を分けて記述します。「varA」の評価を FUN4 より前に実行させたい場合は、以下のように記述してください。

```
tmp    := varA;
result := tmp + FUN4(out => varA);
```

「varA」の評価を FUN4 より後に実行させたい場合は、以下のように記述してください。

```
tmp    := FUN4(out => varA);
result := varA + tmp;
```

● データ型指定なしの定数を含む式の演算精度

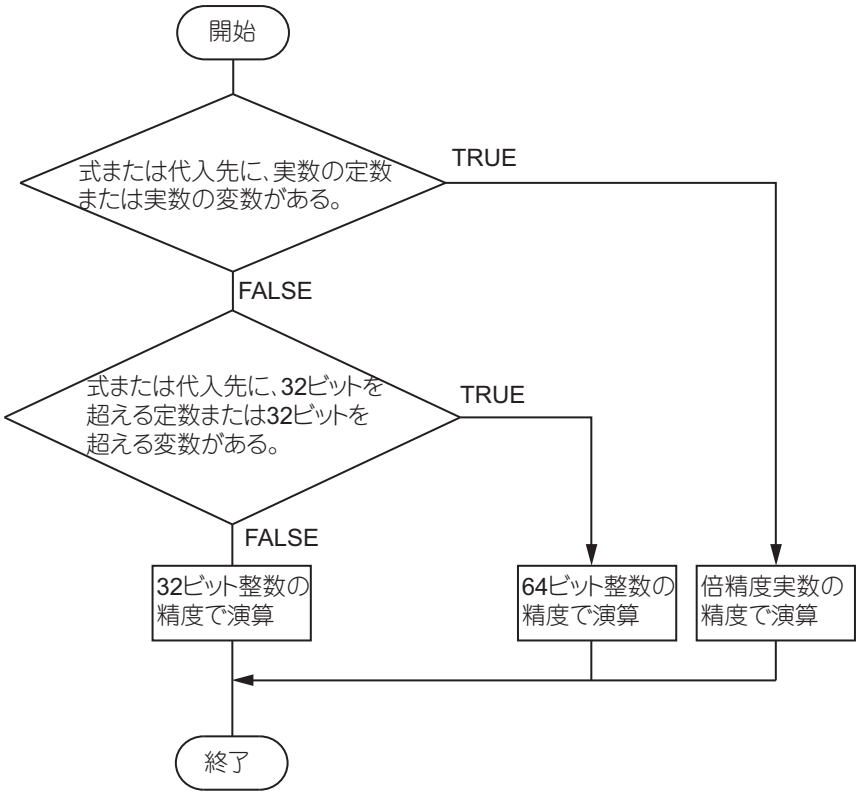
データ型指定なしの定数を含む式の演算精度は、その定数の表記、式および代入先に書かれている変数や定数のデータ型によって、自動的に決められます。

定数の表記

定数が「100」のように小数点なしで表記されると、整数として処理されます。「100.0」のように小数点付きで表記されると、実数として処理されます。

式の演算精度

式の演算精度は、式および代入先に書かれている変数や定数のデータ型によって、32 ビット整数、64 ビット整数、倍精度実数のいずれかになります。式の演算精度は、次のルールで決められます。



例）realv := 2 + 3 * 4;　// 変数 realv のデータ型は REAL
変数 realv が実数型なので、倍精度実数の精度で演算されます。演算結果は 14.0 になります。

realv := 2 + 3.0 *4;　// 変数 realv のデータ型は REAL
定数 3.0 が実数型の表記であり、変数 realv も実数型なので、倍精度実数の精度で演算されます。演算結果は 14.0 になります。

lintv := 2 + 3 * 4;　// 変数 lintv のデータ型は LINT
実数型の定数や変数がなく、変数 lintv が 32 ビットを超える定数なので、64 ビット整数の精度で演算されます。演算結果は 14 になります。

intv := 2 + 3 * 4;　// 変数 intv のデータ型は INT
実数型の定数や変数がなく、32 ビットを超える定数もないので、32 ビット整数の精度で演算されます。演算結果は 14 になります。

ただし、除算の演算精度は、除数と被除数だけで決まります。演算精度を決めるルールは、先のフローチャートと同じです。

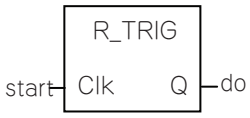
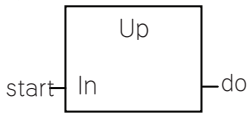
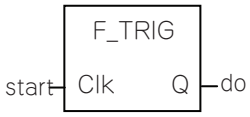
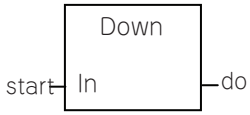
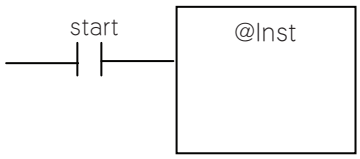
例）realv := 2 / 3 * 4;　// 変数 realv のデータ型は REAL
除算 2/3 は、被除数と除数ともに 32 ビットを超えない整数なので、32 ビット整数の精度で演算されます。除算結果は整数の 0 になります。
その後、変数 realv が実数型なので、0*4 は倍精度実数の精度で演算されます。演算結果は 0.0 になります。

使用上の注意

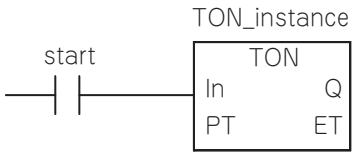
データ型指定なしの定数を含む式の演算精度は、その定数の表記や式内の変数および定数のデータ型などから自動的に決められます。そのために、ユーザが意図しない精度で演算される可能性があります。定数は、REAL#1.0 のように、データ型指定ありで表記することをお勧めします。

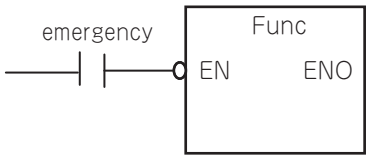
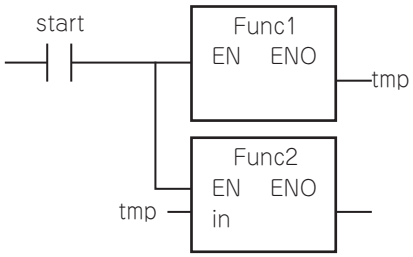
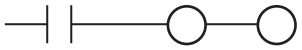
ST とラダー図の差異

ST とラダー図の間で差異がある例は以下のとおりです。

項目	ラダー図	ST（インライン ST 含む）
接点の微分	● **立ち上がり** ・方法1 start do ----┤↑├----------○ ・方法2 R_TRIG_instance  ・方法3  	● **立ち上がり** • **方法 1** R_TRIG_instance （Clk:=start, Q=>do）; ※ R_TRIG_instance は R_TRIG 命令のインスタンス
	● **立ち下がり** ・方法1 start do ----┤↓├----------○ ・方法2 F_TRIG_instance  ・方法3  	● **立ち下がり** • **方法 1** F_TRIG_instance （Clk:=start, Q=>do）; ※ F_TRIG_instance は F_TRIG 命令のインスタンス
命令の微分	● **立ち上がり微分**  	● **立ち上がり微分** 相当するものはないため、ロジックで実現する。たとえば、 • 方法 1 R_TRIG_instance （Clk:=start, Q=>do）; IF （do = TRUE）THEN Inst（）; END_IF; • 方法 2 IF （start = TRUE）THEN IF （pre_start = FALSE）THEN Inst（）; END_IF; END_IF; pre_start:=start;// 前回値の更新

項目	ラダー図	ST（インライン ST 含む）

複数タスク周期命令の記述	命令実行から完了まで複数サイクル要し、かつパワーフロー OFF のときにリセット動作する TON 命令において、1 つのインスタンス宣言で計時とリセットを兼ねる
	 
		以下のように、計時用とリセット用の 2 つのインスタンスを宣言する
		IF （start = TRUE）THEN
		TON_instance（In:=TRUE, 後省略）;
		// タイマ起動
		ELSE
		TON_instance（In:=FALSE, 後省略）;
		// タイマリセット
		END_IF;

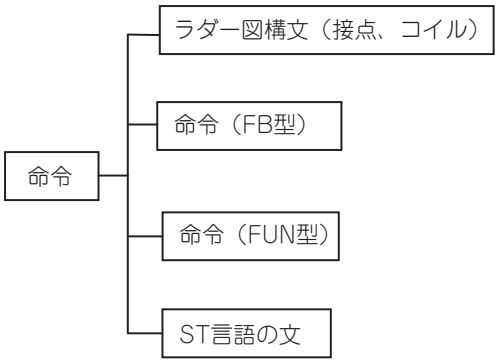
FB ／ファンクション引数反転指定	BOOL 型引数と FB ／ファンクションの交点に○を付加して反転を表す 	引数に NOT 演算子を付加する ※ NOT 演算子は引数に限らず、全ての BOOL 型変数に付加することができる IF （NOT emergency）THEN Func （）; END_IF;
多段接続		IF（start=TRUE）THEN Func2（ in := Func1 （））; END_IF;
ラダー図の後続	Out 命令の後 OUT 命令だけ後続可能 	インライン ST の後にラダー図を後続不可 
プログラムの分割単位	セクション機能あり	セクション機能なし

6-6　命令

ここでは、あらかじめ NY シリーズコントローラで定義している命令について、説明します。
命令の詳細については、🕮『NY シリーズ コマンドリファレンスマニュアル 基本編（SBCA-437）』および 🕮『NY シリーズ コマンドリファレンスマニュアル モーション編（SBCE-380）』を参照してください。

6-6-1　命令とは

命令とは、POU のアルゴリズムの中に記載する、オムロンが用意した最小単位の処理要素です。
命令には、以下の種類があります。



プログラム、ユーザ定義の FB、ユーザ定義の FUN は、この命令の組み合わせで構成されます。

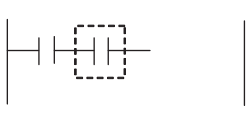
6-6-2　命令の基本知識

命令の基本的な仕様は、FB、FUN の仕様に従います。
ここでは、命令特有の仕様について説明します。

ラダー図構文（接点・コイル）

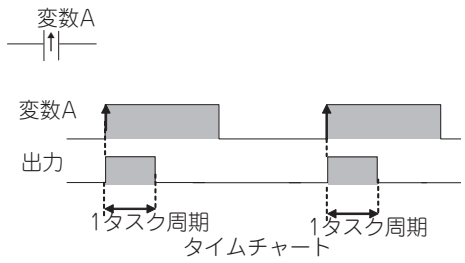
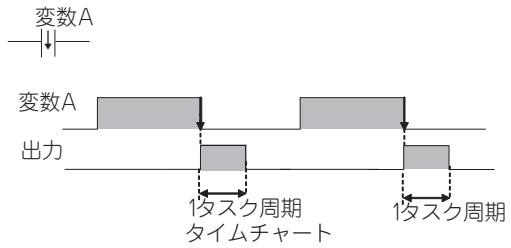
● 配置可能位置

ラダー図言語の接点・コイルの形態をした命令には、次に示すとおり、配置可能な位置があります。

分類		配置可能位置	図
入力系	論理スタート型（LD 型）	左母線への直接接続、または回路ブロックの開始点	
	次段接続型	論理スタート型と出力型の中間位置	
出力系		右母線への直接接続	

● 動作オプション

ラダー図言語の命令（接点）にも、上矢印（↑）、下矢印（↓）を付けることによって、立ち上がり微分、立ち下がり微分の動作をするものがあります。

立ち上がり（↑）	 タイムチャート	毎タスク周期、接点の読出／比較／ビットテストなどの処理を実行し、結果が FALSE → TRUE に変化していれば、パワーフローを出力する命令です。 出力するパワーフローは（1 タスク周期 TRUE のあと）次のタスク周期で FALSE となります。
立ち下がり（↓）	 タイムチャート	毎タスク周期、接点の読出などの処理を実行し、結果が TRUE → FALSE に変化していれば、パワーフローを出力する命令です。出力するパワーフローは（1 タスク周期 TRUE のあと）次のタスク周期で FALSE となります。

ファンクションブロック（FB）型の命令

● 実行条件

FB 型命令の実行条件は、命令によって異なります。
実行条件用の特定の入力変数が、命令ごとに定義されています。
例：　Execute：立ち上がりエッジ指定、立ち下がりエッジ指定
　　Enable：毎タスク周期実行

FB 型命令は、その命令の呼び出し元の POU が実行している間、実行し続けます。

● 動作オプション

指定できません。

ファンクション（FUN）型の命令

● 実行条件

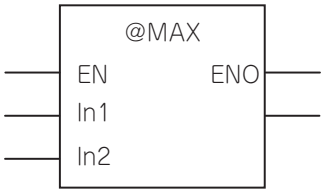
FUN 型命令の実行条件は、すべて、EN 入力です。EN が TRUE の間、毎タスク周期実行されます。

● 動作オプション

ラダー図言語の場合、次の動作オプションを付加することで、その実行条件を立ち上がり指定とすることが可能です。ST 言語では動作オプションはありません。

動作オプション		オプション記号	
微分オプション	立ち上がり	@	命令を入力立ち上がり微分型命令とします。 EN の立ち上がり時に、1 回だけ命令を実行します。

動作オプションを付加するには、命令名の前に、上記のオプション記号を付けます。
例）



ファンクションブロック（FB）型／ファンクション（FUN）型の命令共通

● コンディションフラグ

命令の処理の結果として値が代入されるシステム定義変数を、「コンディションフラグ」と呼びます。NY シリーズコントローラのコンディションフラグは、キャリーフラグ（P_CY）だけです。
キャリーフラグは、以下のいずれかの意味を持ちます。

- 命令の処理結果が、出力変数のデータ型で表現できる範囲を超えたかどうかの判定
- ビット列データのビットシフト命令などで、桁あふれが発生したかどうかの判定の詳細は、🕮『NY シリーズ コマンドリファレンスマニュアル 基本編（SBCA-437）』の各命令の説明を参照してください。

6-6-3　命令エラー

命令実行にともなって発生する異常を、命令エラーと呼びます。ここでは、命令エラーの発生タイミング、命令エラーで検出される異常、命令エラー発生時の動作などについて説明します。

命令エラーの発生タイミング

命令エラーの発生タイミングには、以下の 3 つがあります。命令エラーの発生タイミングによって、検出される異常や異常発生時の動作が異なります。

- 命令実行前に、入力パラメータまたは入出力パラメータの値をチェックしたときに発生
- 命令実行中に、命令の内部処理にともなって発生
- 命令実行後に、出力パラメータの値をチェックしたときに発生

命令エラーで検出される異常

命令エラーで検出される異常は、以下のとおりです。命令エラーの発生タイミングによって、検出される異常が異なります。

● 命令実行前、命令実行後に検出される異常

命令実行前、命令実行後に検出される異常は、以下のとおりです。

- 配列指定変数の、配列領域を超えた要素への読み書き
- STRING 型変数への、定義されているバイト数よりも長い文字列の代入
- STRING 型変数への、NULL 文字で終端されていない文字列の代入
- 整数型変数の、0 除算

● 命令実行中に検出される異常

命令実行中に検出される異常は、命令によって異なります。各命令で検出される異常の詳細は、🕮『NY シリーズ コマンドリファレンスマニュアル 基本編（SBCA-437)』を参照してください。

命令エラー発生時の動作

命令エラー発生の有無で動作が異なる要素に、出力変数「ENO」、出力変数「Error」、出力変数「ErrorID」、システム定義変数「P_PRGER」、イベントがあります。それぞれについて説明します。

● 出力変数「ENO」、出力変数「Error」、出力変数「ErrorID」

異常発生の有無を示す出力変数に、「ENO」（イネーブルアウト）、「Error」（エラー）、「ErroriD」（エラーコード）があります。命令によって、どの出力変数をもつかが異なります。各変数の意味と、命令エラー発生時の値は、以下のとおりです。命令エラー発生タイミングによって、振る舞いがことなります。

出力変数	データ型	意味	命令エラー発生時の値		
			命令実行前*1	命令実行中	命令実行後*2
「ENO」	BOOL	TRUE：正常終了 FALSE：異常終了、実行中、未実行	FALSE	FALSE	TRUE
「Error」	BOOL	TRUE：異常終了 FALSE：正常終了、実行中、未実行	FALSE	TRUE	FALSE
「ErrorID」	WORD	異常終了時はエラーコード、正常終了時は WORD#16#0	WORD#16#0	エラーコード	WORD#16#0

*1. 命令実行前に命令エラーが発生した場合は、命令が実行されません。したがって、各出力パラメータの値は、命令実行前のまま保持されます。

*2. 命令実行後に命令エラーが発生した場合は、命令自体は正常終了したと判断されます。したがって、各出力パラメータには、命令の出力変数の値が代入されます。ただし、異常が発生した出力パラメータの値は、命令実行前のまま保持されます。

● システム定義変数「P_PRGER」

システム定義変数「P_PRGER」は、命令エラー発生フラグです。命令エラーが発生すると、命令エラー発生タイミングに関係なく、値が TRUE になります。正常終了時には、値を保持します。

「P_PRGER」の詳細は、🕮「 命令エラー発生フラグ」（P.6-135）を参照してください。

● イベント

命令エラーが発生すると、異常内容に対するイベントが発生します。イベントの詳細は、🕮「 命令エラー発生に伴うイベント発生」（P.6-136）を参照してください。

ラダー図言語での出力パラメータ

ラダー図言語で記述された命令、ユーザ定義のファンクション、ユーザ定義のファンクションブロックが正常終了した場合と、命令エラーが発生した場合の、出力パラメータの値は、以下のようになります。

条件	出力パラメータの種類	出力パラメータの値
正常終了時	パワーフロー出力	内部アルゴリズムに従って値が更新されます。
	パラメータ出力 BOOL 型	
	パラメータ出力 BOOL 型以外	
命令エラー発生時	パワーフロー出力	FALSE になります。
	パラメータ出力 BOOL 型	それまでの値を保持します。
	パラメータ出力 BOOL 型以外	

ST 言語で記述された POU での構文エラー発生時の動作

● 代入文におけるエラー

ST 言語で記載された代入文にエラーが発生した場合は、その行を実行しません。

```
5  a = b / (c + d) + e * f + ABS(g);
6  x := 1;
```

たとえば、
5行めのb/(c+d)でゼロ割が発生した場合、
5行めの実行を中断し(aの値は変化しない)、
6行めを実行する。

ユーザ作成ファンクションの出力 ENO が OFF のときも同様です。

```
5  a = ユーザ作成ファンクション(b) + c;
6  x := 1;
```

ユーザ作成ファンクションの出力である
ENOがOFFのとき、
5行目の実行を中断し(aの値は変化しない)、
6行目を実行する。

● IF 構文におけるエラー

ST 言語の構文でエラーが発生した場合、構文に基づいてエラー処理を行います。
下記で（c+d）の値がゼロのとき、IF ～ END_IF 間は実行しません。

POU「○○」

```
 5  IF a = b / (c + d) THEN
 6    x := 1;
 7  ELSE
 8    x := 2;
 9  END_IF;
10  y := 10;
 :  :
    IF P_PRGER = TRUE THEN
      x := 初期値; (* 異常発生時処理 *)
      y := 初期値;
    END_IF;
```

ユーザはエラーが生じた場合の
安全処理を記載する。

● ST 言語における構文エラー

ST 言語における構文エラーには以下があります。

- 配列 [i] の要素数超え
- 入出力変数にパラメータがセットされていない
- 文字列型の代入：左辺の文字列型サイズ（Byte）＜右辺文字列長（Byte）の場合
- 除算の 0 割（浮動小数点演算を除く）

※ 浮動小数点数の値が非数であるときの演算は、演算結果が非数となるが、エラーとはみなしません。

● 構文エラー発生時の動作

P_PRGER フラグが TRUE となり、以下となります。

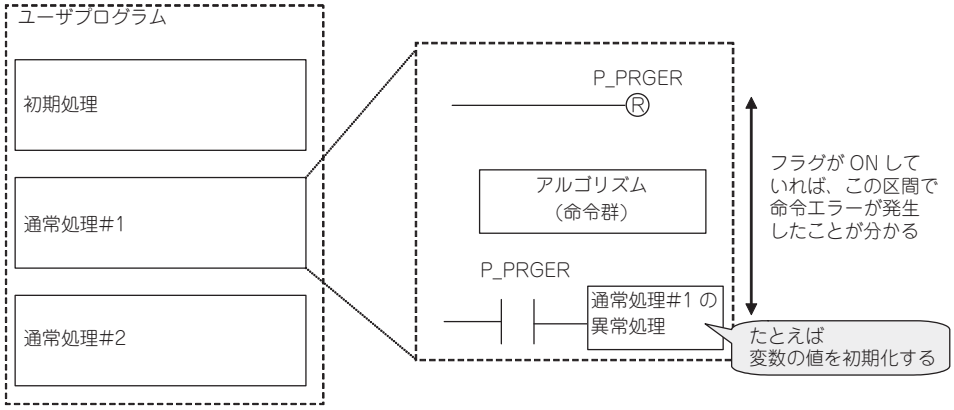
構文	エラー発生箇所	動作
代入文		その行を実行しない
制御文	**IF 条件式内**	IF ～ END_IF 間を実行しない
	CASE 条件式	CASE ～ END_CASE 間を実行しない
	FOR 条件式内	FOR ～ END_FOR 間を実行しない
	WHILE 条件式内	WHILE ～ END_WHILE 間を実行しない
	REPEAT 条件式内	REPEAT ～ END_REPEAT 間を実行しない

命令エラー発生フラグ

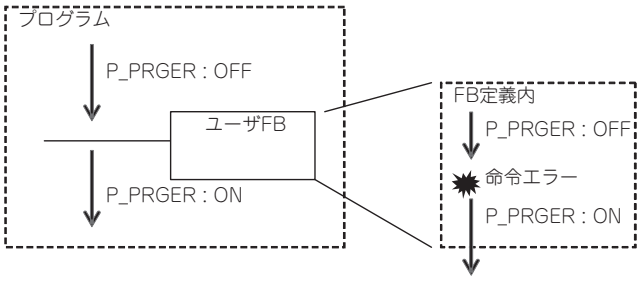
ラダー図アルゴリズム内で命令エラーまたは ST アルゴリズムで構文／関数でエラーがあった場合、システム定義変数「P_PRGER」（命令エラー発生フラグ）が TRUE となります。
P_PRGER フラグは、プログラムのローカル変数です。プログラム内で命令エラーが発生したときに TRUE となり、次タスク周期においても値を保持します。

変数名	名称	機能	データ型	値の範囲	初期値	R/W
P_PRGER	命令エラー発生フラグ	命令エラーが発生したときに TRUE となり続けるフラグです。一度 TRUE となると、ユーザプログラムによって FALSE にされるまで TRUE となり続けます。	BOOL	FALSE/TRUE	FALSE	RW

P_PRGER フラグは、ユーザによる上書きも可能です。これにより、以下のように、いったんユーザ操作で FALSE にすることで、特定の範囲で命令エラーが発生したかどうかを判断することもできます。一度 TRUE になると、動作モードを変更するかもしくは、ユーザプログラムなどで上書きするまで値を保持し続けます。
例）



プログラム内で使用しているユーザ定義 FB 内で命令エラーが発生した場合も、P_PRGER フラグは TRUE となります。
例）



命令エラー発生に伴うイベント発生

命令エラー発生に伴って、それに対応するイベントが発生します。

イベントの確認方法の詳細ついては、🕮「8-5 イベントログ機能」(P.8-45) を参照してください。また、イベントの内容については、🕮『NY シリーズ コマンドリファレンスマニュアル 基本編 (SBCA-437)』を参照してください。

使用上の注意

- 命令エラー発生時にイベントが発生するのは、Sysmac Studio のコントローラ設定で、[イベントログ設定]-[命令エラー出力] を [する] に設定している場合です。Sysmac Studio のコントローラ設定の詳細については、🕮「4-2-2 コントローラ設定」(P.4-4)、および、🕮『Sysmac Studio Version 1 オペレーションマニュアル (SBCA-470)』を参照してください。
- 命令エラー発生後にユーザプログラムを変更すると、イベントログに情報が正しく記録されない場合があります。
- 異常がある命令を繰り返し実行した場合、命令エラーは毎回発生します。したがって、イベントログの記録件数が、上限を超える場合があります。その際、古い情報から順に上書きされるので、ご注意ください。

参考

- モーション制御命令の実行時に異常が発生した場合、命令エラーに伴うイベントと、モーション制御命令固有のイベントの両方が発生します。モーション制御命令のイベントの詳細は、🕮『NY シリーズ コマンドリファレンスマニュアル モーション編 (SBCE-380)』を参照してください。
- Sysmac Studioのコントローラ設定で、[イベントログ設定]-[命令エラー出力]を[しない]に設定しても、モーション制御命令固有のイベントは発生します。

6-7　名前空間

ここでは、名前空間の仕様と使用方法について説明します。操作方法については、🕮『Sysmac Studio Version 1 オペレーションマニュアル（SBCA-470)』を参照してください。

6-7-1　名前空間とは

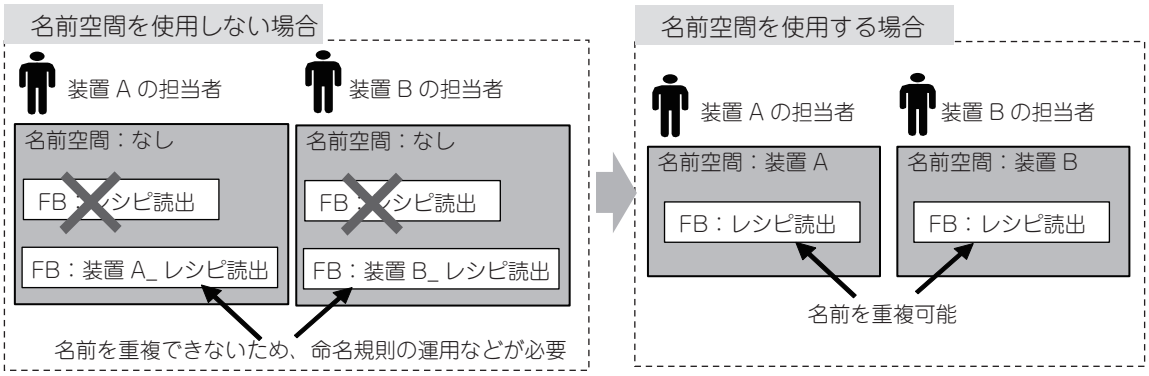
名前空間とは、FB 定義などをグループに区分けして階層構造で管理するしくみです。これは、ファイルをフォルダに区分けしてディレクトリの階層構造で管理することに相当します。名前空間を使用しない場合、FB 定義などに同じ名前を重複して使用できませんが、名前空間を使用して異なる名前空間を設定することで重複して使用できます。
名前空間は、使用してもしなくてもかまいません。

名前空間の特長

以下の特長があります。

● 名前の重複回避

異なる名前空間であれば、FB 定義名などに同じ名前を使用できます。
たとえば、複数装置のプログラミングをする場合に、装置ごとに別々の担当者がプログラミングをすると、FB 定義名などが重複する可能性があります。名前空間を使用しない場合は、命名規則の運用などにより名前の重複を回避します。名前空間を使用して装置ごとに名前空間を設定すれば、装置間での名前の重複を意識せずにプログラミングができます。



6-7-2　名前空間の仕様

ここでは、名前空間の対象と表現、および使用宣言について説明します。

名前空間の対象

名前空間で扱える対象は以下のとおりです。

対象	詳細
POU 定義	FB 定義名、FUN 定義名
データ型	構造体型名、共用体型名、列挙型名

名前空間の表現

名前空間は、「\（バックスラッシュ）」で区切って表現します。POU のアルゴリズム内で使用する場合は、名前空間の先頭に「\\（バックスラッシュ 2 つ）」を付加します。
「\（バックスラッシュ）」は、Windows のロケール設定やフォントなどにより「￥」と表示されることもあります。

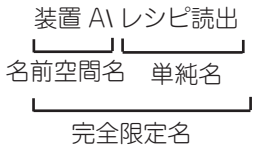
例）

使用場面	表現
アルゴリズム以外で使用する	装置 A\ レシピ読出
アルゴリズム内で使用する	\\ 装置 A\ レシピ読出

● 完全限定名と単純名

名前空間名を省略せずに、名前空間の対象を表現した名前を、完全限定名と呼びます。名前空間名を省略した名前を単純名と呼びます。
POU 定義のアルゴリズム内では同じ名前空間に属する POU 定義を単純名で記述できます。

例 ）



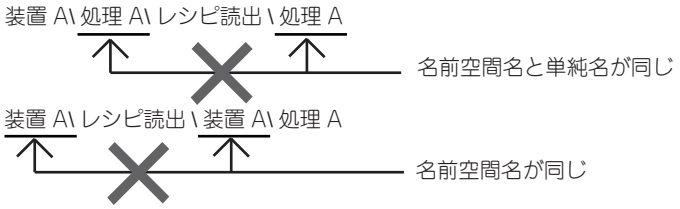
● 名前空間の表現の制限

- 使用できる文字は、変数名と同じです。詳細は、📖「6-3-12 変数名などのプログラム関連の名称の制限」（P.6-77）を参照してください。
- 名前空間の文字数の制限は以下のとおりです。

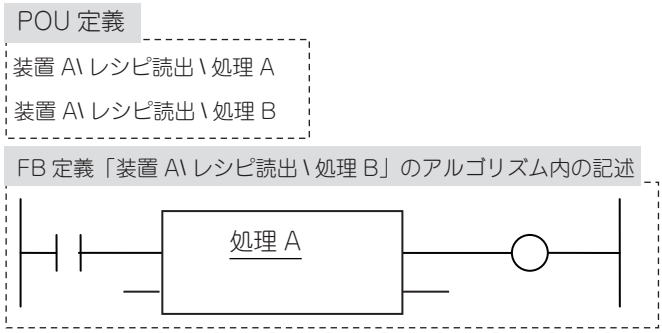
名前	最大サイズ	文字コード
名前空間名	93 バイト	UTF-8
単純名	127 バイト	

使用上の注意

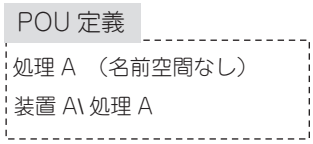
- 名前空間名と単純名が同じ場合は、ビルド時にエラーになります。



- POU定義のアルゴリズム内では同じ名前空間に属するPOU定義を単純名で記述できますが、その名前空間の上位の階層に同じ単純名の POU 定義またはデータ型などが存在すると、ビルド時にエラーになります。
  たとえば、下図のように POU 定義が存在する場合、名前空間「装置 A\ レシピ読出」上にある FB 定義「処理 B」のアルゴリズム内の記述で、同じ名前空間に属する「装置 A\ レシピ読出 \ 処理 A」を単純名で呼び出すことができます。



  ただし、以下のように、名前空間「装置 A\ レシピ読出」よりも上位の階層にも POU 定義「装置 A」が存在する場合は、複数の単純名「処理 A」が存在するため、ビルド時にエラーになります。
  この場合は、完全限定名で記述するか、単純名を変更してください。



- システムが内部で使用する予約語と合致した場合は、プログラムチェック時にエラーとなります。

名前空間の使用宣言

プログラミング時に、POU 定義のアルゴリズム内で使用する名前空間をあらかじめ宣言することを、使用宣言と呼びます。
POU 定義で使用宣言すると、使用宣言した名前空間に属する POU 定義などを単純名で使用できます。使用宣言した場合でも完全限定名を使用できます。
POU 定義のアルゴリズム内では、POU 定義と同じ名前空間に属する FB 定義や FUN 定義を使用する場合、使用宣言しなくても単純名を使用できます。
1 つの POU 定義に対して複数の使用宣言ができます。

● 記述例

FB 定義の作成時に、使用する対象の FB 定義の名前空間を使用宣言しない場合と使用宣言する場合の記述例を説明します。

例）
FB 定義「リフタ」のアルゴリズム内に、FB 定義「レシピ読出」および FB 定義「上限計算」を記述する例です。
それぞれの FB 定義は異なる名前空間に属しています。FB 定義「リフタ」には、名前空間「装置 C」だけを使用宣言しています。
名前空間「装置 C」に属さない FB 定義「レシピ読出」は、完全限定名で記述します。名前空間「装置 C」に属する FB 定義「上限計算」は、単純名で記述できます。

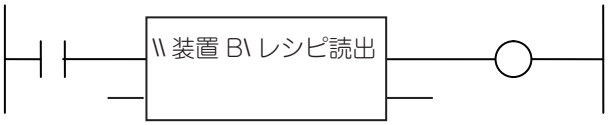
名前空間	対象の種類	単純名
装置A	FB定義	リフタ
装置B	FB定義	レシピ読出
装置C	FB定義	上限計算

FB 定義「リフタ」の [名前空間 - 使用宣言] の記述

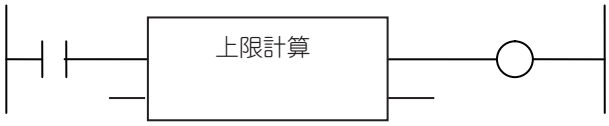
装置 C

FB 定義「リフタ」のアルゴリズム内の記述

・名前空間を使用宣言していない場合
FB 定義「レシピ読出」が属する名前空間「装置 B」を使用宣言していないので、完全限定名で記述します。



・名前空間を使用宣言した場合
FB 定義「上限計算」が属する名前空間「装置 C」を使用宣言しているので、単純名で記述できます。



● 使用宣言の制限

単純名は POU 定義のアルゴリズム内だけで使用できます。

使用上の注意

- 以下の場合は、プログラムチェックでエラーとなります。
  - 存在しない名前空間を使用宣言した場合
  - 1つのPOU定義に複数の名前空間を使用宣言し、それぞれの名前空間に同じ名前のPOU定義やデータ型などが存在する場合
- 以下で POU 定義やデータ型などに同じ名前をつけると、ビルド時にエラーになります。
  - POU 定義が属する名前空間とその上位の階層の名前空間
  - 使用宣言した名前空間
  - 名前空間なし

名前空間	対象の種類	単純名	
装置A\リフタ	FB定義	処理A	×POU 定義が属する名前空間
装置A	FB定義	処理A	×POU 定義が属する名前空間の上位の階層
装置B	FB定義	処理B	
		処理A	×使用宣言した名前空間
なし	FB定義	処理A	×名前空間なし

FB 定義「処理 A」の [名前空間 - 使用宣言] の記述

装置 B

参考

プログラム名に対して名前空間を設定することはできませんが、プログラムのアルゴリズム内で使用する部品の名前空間を使用宣言することはできます。

6-7-3　名前空間の使用手順

Sysmac Studio を使用して、名前空間の設定と使用宣言を行います。
データ型を作成したり、FB 定義や FUN 定義などの部品を作成したりするときに、手順 1 と手順 2 の設定を行います。
名前空間を設定した部品を使用するときに、手順 3 の使用宣言を行います。

1 **データ型エディタで、対象のデータ型に名前空間を設定します。**

2 **対象の FB 定義や FUN 定義のプロパティに、名前空間を設定します。**

3 **ラダーエディタまたはSTエディタで、対象のFB定義やFUN定義のプロパティに設定した名前空間を使用宣言します。**

4 **ユーザプログラムに、対象のデータ型や FB 定義などを記述します。**

操作の詳細については、🕮『Sysmac Studio Version 1 オペレーションマニュアル (SBCA-470)』を参照してください。

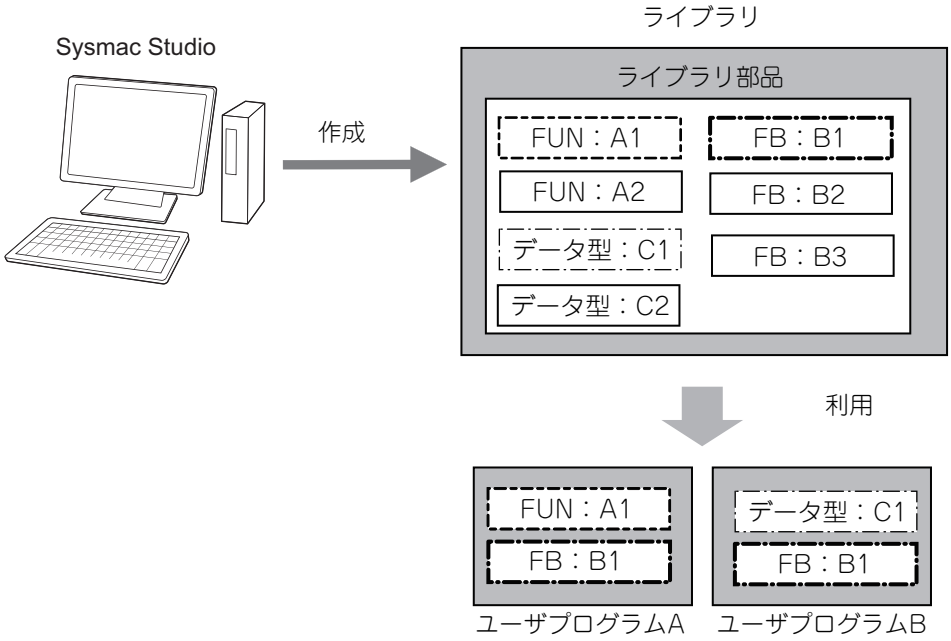
6-8　ライブラリ

ここでは、ライブラリの仕様について説明します。操作方法の詳細については、🕮『Sysmac Studio Version 1 オペレーションマニュアル（SBCA-470）』を参照してください。

6-8-1　ライブラリの概要

POU 定義やデータ型を、流用するための部品として利用可能な形にまとめたものをライブラリと呼びます。ライブラリの部品をライブラリ部品と呼びます。
NY シリーズコントローラでは、ライブラリの作成と利用ができます。

以下の図では、ライブラリの中のライブラリ部品のうち、ユーザプログラム A が FUN：A1 と FB：B1 を、ユーザプログラム B がデータ型：C1 と FB：B1 をそれぞれ利用しています。



6-8-2　ライブラリの仕様

ライブラリの設定、同期について説明します。

ライブラリの設定

ライブラリには、以下の項目を設定できます。

設定項目	説明
名前	ライブラリの名前です。
バージョン	ライブラリのバージョンです。
作成者	ライブラリ作成者の名前です。（任意）
作成日	ライブラリを新規作成した日付です。
更新日	ライブラリを更新した日付です。
コメント	ライブラリのコメントです。（任意）
会社名	ライブラリを作成した会社の名前です。（任意）
ID	ライブラリ照合用のユニークな ID です。自動生成され変更できません。
ソース表示不可	ソースの表示、または非表示を選択できます。*1
添付ファイル	複数の任意のファイルを添付できます。

*1. ライブラリ部品にデータプロテクトが設定されている場合は、ソースコードを表示するときにパスワードが必要になります。

ライブラリ部品は、別のライブラリを参照して作成することもできます。この場合は、参照先のライブラリのデータも含めるかどうかを選択します。

参考

参照先のライブラリを含めるとしたとき、参照先のライブラリのコピーを含めて参照元のライブラリが作成されます。そのため、ライブラリのファイルが 1 個ですみますが、参照元のライブラリが複数あるときの変更は個々に行う必要があります。
含めないとしたときは、参照先のライブラリは含まず参照元のライブラリが作成されます。そのため、ライブラリのファイルは参照元と参照先の複数個に分割されますが、参照元のライブラリが複数あるときの変更は参照先のライブラリだけですみます。

● ライブラリ部品の選択

ライブラリに含めるライブラリ部品を選択します。

ライブラリの同期

ライブラリ単位で、Sysmac Studio からのダウンロード、コントローラからのアップロード、コントローラとの照合ができます。

参考

- ソースを表示するライブラリが転送されているコントローラに、Sysmac Studio から「プログラムのソースを転送しない」設定でプロジェクトを転送した場合は、ライブラリのソースは転送されません。
- メモリオールクリアすると、コントローラ内のライブラリは消去されます。

6-8-3　ライブラリ部品の仕様

ライブラリ部品の対象と、ライブラリ部品の設定について説明します。

ライブラリ部品の対象

以下のデータをライブラリ部品として扱えます。

ライブラリ部品	詳細
POU 定義	FB、FUN
データ型*1	構造体型、共用体型、列挙型

*1. Sysmac Studio の部品選択画面で、データ型は常にライブラリに含まれます。

ライブラリ部品の設定

それぞれのライブラリ部品に以下の項目を設定できます。

設定項目	定義
名前	ライブラリ部品の名前です。
名前空間	ライブラリ部品の名前空間です。
バージョン*1	ライブラリ部品のバージョンです。
作成者*1	ライブラリ部品作成者の名前です。（任意）
作成日*1	ライブラリ部品を新規作成した日付です。
更新日*1	ライブラリ部品を更新した日付です。
コメント	ライブラリ部品のコメントです。（任意）

*1. Sysmac Studio の各 POU 定義のプロパティで、FB と FUN だけに対して設定できます。

6-8-4　ライブラリの使用手順

ライブラリの作成手順と利用手順は以下のとおりです。

ライブラリ作成手順

ライブラリはプロジェクトとして作成します。以下の手順でライブラリを作成し保存します。

1 **ライブラリプロジェクトを新規作成します。**
プロジェクトウィンドウの、プロジェクトの種類に「ライブラリプロジェクト」を選択してプロジェクトを作成します。

2 **ライブラリ部品を作成します。**
ライブラリプロジェクトに、POU 定義やデータ型などを作成し、動作確認します。

3 **ライブラリの設定をします。**
ライブラリプロジェクトに、ライブラリ部品の選択、ソースの表示、添付ファイルなどの設定をします。

4 **ライブラリファイルとして保存します。**
〔 ライブラリ作成 〕ウィンドウで、ライブラリファイルとして保存します。

参考

- すでに作成済みのプロジェクトでも、プロジェクトに登録されているデバイスがコントローラだけの場合は、プロジェクトのプロパティから種類を「ライブラリプロジェクト」へ変更することができます。
- ライブラリプロジェクトではライブラリ部品の対象以外のデータも作成できますが、ライブラリ部品として選択可能なのはライブラリ部品の対象だけです。
- ライブラリ部品として作成するファンクション、ファンクションブロック定義、データ型の名前には、複数のライブラリ間の名前の重複を防ぐため、名前空間を使用することをお勧めします。名前空間については、🕮「6-7 名前空間」（P.6-137）を参照してください。

ライブラリ利用手順

作成したライブラリをプロジェクトに読み込んで、ユーザプログラムで利用できます。以下の手順でライブラリを利用します。

1 **ライブラリを指定します。**
ライブラリを利用するプロジェクトの〔 ライブラリ参照 〕ウィンドウで、利用するライブラリファイルを指定します。

2 **ライブラリ部品を使用してプログラミングします。**
読み込んだライブラリのライブラリ部品をプロジェクト内で使用します。
使用方法は、通常のファンクション、ファンクションブロック定義、データ型と同じです。

6-9　ユーザプログラム作成上の留意事項

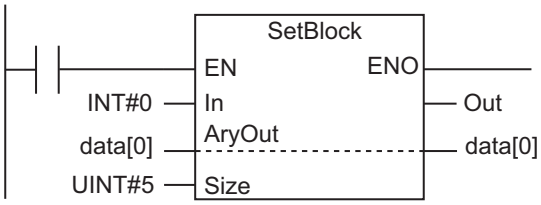
ここでは、ユーザプログラム作成上の留意事項について、説明します。

6-9-1　入力変数／出力変数／入出力変数が、配列指定の変数の場合

一部の命令には、配列変数を扱う命令が存在します。
例）

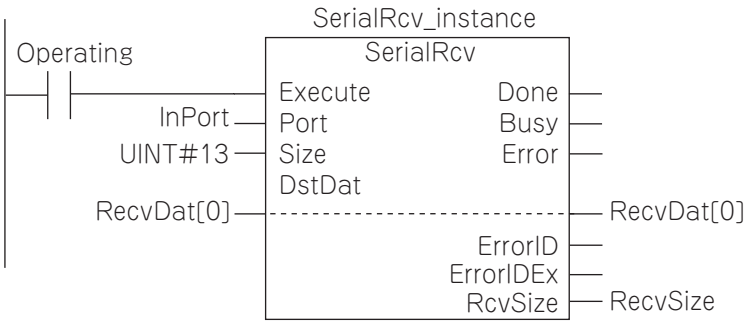
データ型テーブル

変数名	データ型
data	ARRAY[0..4] OF INT



6-9-2　入力変数／出力変数／入出力変数が、構造体型変数の場合

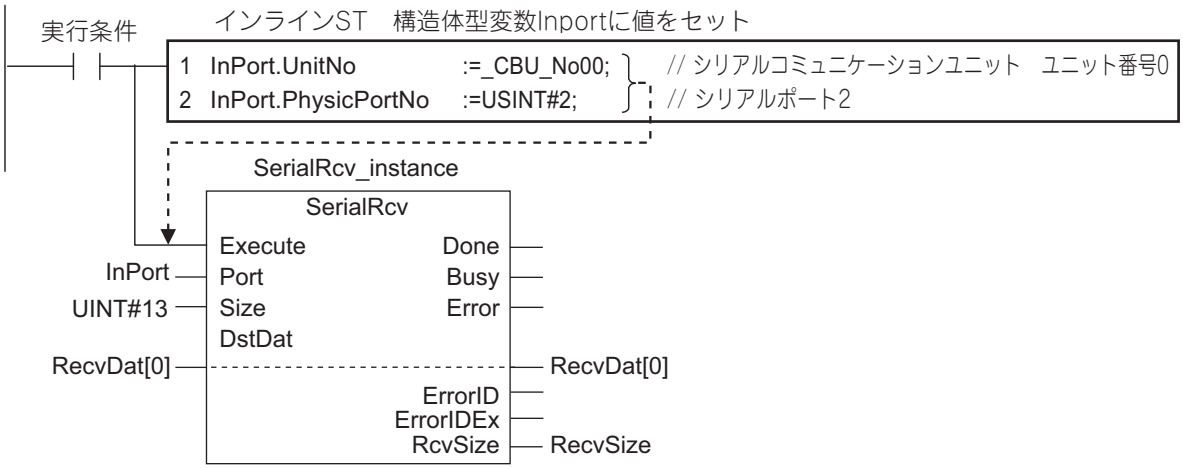
一部の命令には、入力変数、出力変数、入出力変数のデータ型が構造体型であるものがあります。
例）



この場合、事前に入力パラメータ、出力パラメータ、入出力パラメータ用に、構造体型変数を作成し、それに MOVE 命令などで値をセットする必要があります。
例）

変数テーブル

変数名	データ型
InPort	_sPORT



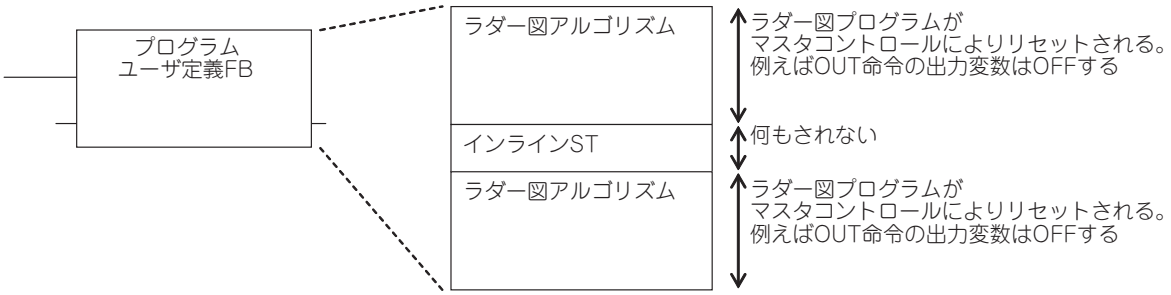
6-9-3　マスタコントロール

概要

マスタコントロールとは、MC（マスタコントロール開始）命令と MCR（マスタコントロール終了）命令の間に存在する処理を、OFF 出力させる機能です。
マスタコントロールは、比較的長い一連の処理の実行条件を制御するときに便利です。
詳細は、🕮『NY シリーズ コマンドリファレンスマニュアル 基本編（SBCA-437）』の MC/MCR 命令の項を参照してください。

マスタコントロールが使えるプログラム言語

マスタコントロールは、ラダー図言語で使用できます。
ST 言語では使用できません。ラダー図プログラム中のインライン ST でも使用できません。
例）
マスタコントロール領域内：



マスタコントロールリセット時の領域内の命令の動作

マスタコントロールリセット時の領域内の命令の動作については、🕮『NY シリーズ コマンドリファレンスマニュアル 基本編（SBCA-437）』「MC/MCR 命令」の項を参照してください。

7

動作確認と運転

本章では、オフラインデバッグを含めた動作確認の内容と、動作確認から運転までの手順を説明します。

7-1 動作確認と運転の手順の概要 7-2

7-2 オフラインデバッグ 7-3
7-2-1 シミュレーション機能の特長 7-3
7-2-2 シミュレーション機能の実行 7-3
7-2-3 シミュレーション機能の設定 7-6

7-3 実機での動作確認と運転 7-8
7-3-1 実機での動作確認から運転開始までの手順 7-8
7-3-2 プロジェクトの転送 7-9
7-3-3 I/O の配線確認 7-9
7-3-4 MC 試運転 7-9
7-3-5 ユーザプログラムの動作確認 7-10
7-3-6 本運転の開始 7-10

7-1　動作確認と運転の手順の概要

動作確認と運転は、全体の手順の中で下図の網掛けで示した位置に対応します。STEP2-4 オフラインデバッグでは、コントローラとオンライン接続せずにシミュレーション機能で動作確認します。STEP5. 実機による動作確認と運転では、コントローラとオンライン接続して実機の動作確認をし、動作確認が完了したら運転を開始します。

全体手順は、🕮「1-4 NY シリーズの使用手順の全体」（P.1-10）を参照してください。



7-2　オフラインデバッグ

ここでは、オフラインデバッグで使用するシミュレーション機能について説明します。シミュレーション機能とは、NY シリーズコントローラの動作をパソコン上でシミュレートすることにより、パソコンだけでユーザプログラムの動作確認をする機能です。シミュレーション機能は実機コントローラでは実現できないデバッグ機能を持つため、ユーザプログラム開発およびデバッグの効率を向上させることができます。

7-2-1　シミュレーション機能の特長

シミュレーション機能は、コントローラとオンライン接続して動作確認をするオンラインデバッグに比べて、以下のような場合に有効です。

- ブレークポイント、ステップ実行、一時停止などを使って、プログラムのロジックを確認する。
- シミュレーション対象を選択し、一部のプログラムだけの動作確認をする。
- シミュレーション実行速度変更により、実際よりゆっくりとした速度で動作確認をする。
- コントローラの外部からの入力などをデバッグプログラムで動作させる。

7-2-2　シミュレーション機能の実行

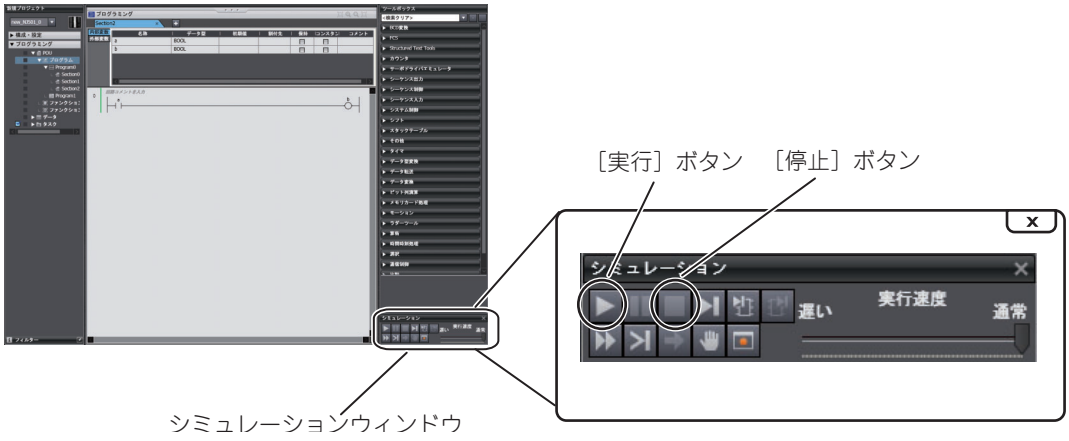
シミュレーション機能では以下を実行できます。

- シミュレータの起動と停止
- プログラムのロジックの確認
- オンライン接続時のデバッグ機能

シミュレータの起動と停止

シミュレーション機能は、Sysmac Studio のシミュレーションウィンドウからシミュレータを起動して実行します。シミュレーション機能による動作確認が完了したら、シミュレータを停止します。
シミュレーション実行と停止の操作の概要は以下のとおりです。

1　**Sysmac Studio で、メインメニューから［表示］-［シミュレーションウィンドウ］を選択します。**
ツール画面右下にシミュレーションウィンドウが表示されます。



2 **シミュレーションウィンドウの［実行］ボタンを押します。**
プログラムがシミュレータに転送され、シミュレーションが開始されます。シミュレーションが開始されると、エディタ等の Sysmac Studio の画面がコントローラとのオンライン接続時と同じ状態になります。

3 **動作確認が完了したら、シミュレーションウィンドウの［停止］ボタンを押してシミュレータを停止します。**

プログラムのロジックの確認

プログラムのロジックの正当性を確認するため、シミュレータの動作を止めたり、1 ステップずつ実行したりすることができます。シミュレーションウィンドウのボタンから以下の操作ができます。

操作名	操作内容
ブレークポイント	プログラム上の指定した位置で、プログラム実行が一時停止されます。
ステップ実行	ST 言語のプログラムでは 1 行、ラダー図言語のプログラムでは 1 命令を実行すると、シミュレータが一時停止されます。
連続ステップ実行	一定間隔で連続してステップ実行されます。
一時停止	シミュレーションの実行が一時停止されます。
ステップイン実行	ファンクションブロックまたはファンクションのソース内をステップ実行されます。
ステップアウト実行	ステップイン実行中のファンクションブロック、またはファンクションが終了するまで実行されます。
1 周期実行	現在実行しているタスクが 1 周期分実行されます。次の周期のプログラム先頭で一時停止されます。
条件ブレークポイント	ブレークポイントに停止条件を設定し、条件成立時だけプログラム実行が一時停止されます。

オンライン接続時のデバッグ機能

シミュレータでは、コントローラとのオンライン接続時に使用するデバッグ機能を一部だけ実行できます。コントローラでのオンラインデバッグと、シミュレータによるオフラインデバッグとのデバッグ機能の違いを下表に示します。各デバッグ機能でのコントローラとシミュレータの動作の違いは、🕮『Sysmac Studio Version 1 オペレーションマニュアル（SBCA-470)』を参照してください。

（○：使用できる、× 使用できない）

デバッグ機能	コントローラ	シミュレータ
モニタリング	○	○
ウォッチウィンドウでのモニタ	○	○
I/O マップでのモニタ	○	○
微分モニタ	○	○
接点のセット／リセット	○	○
接点の強制値リフレッシュ TRUE/FALSE/ 解除	○	○
データの現在値変更	○	○
メモリオールクリア	○	×
クロスリファレンスポップアップ	○	○
オンラインエディット	○	○
コントローラステータスモニタ	○	×
タスク実行状態モニタ	○	○
軸ステータスモニタ（MC モニタテーブル）	○	○
動作モードの変更	○	×
コントローラのリセット	○	×
データトレース	○	○
トリガの設定	○	○
サンプリング変数の設定	○	○
トレースの開始・停止	○	○
トレース結果の表示	○	○
トレース結果のエクスポート	○	○
3D 装置モデルの作成	○	○
デジタル・アナログチャートでの表示	○	○
3D 形式での軌跡表示	○	○
タスク実行時間のモニタ	○	×
プログラムのシミュレーションデバッグ	×	○
シミュレーションプログラムの設定	×	○
シミュレーションの実行速度の変更	×	○
ブレークポイントの設定	×	○
ステップ実行	×	○
トラブルシュート	○	○
異常情報モニタ	○	○
異常履歴表示	○	○
イベントテーブル設定	○	○
ユーザメモリの使用量モニタ	○	○
時計情報の設定	○	×
アクセス権の強制解放	○	×

7-2-3　シミュレーション機能の設定

シミュレーション機能では以下の設定ができます。

- シミュレーションプログラムの設定
- デバッグプログラムの設定
- シミュレーション実行速度の設定

シミュレーションプログラムの設定

シミュレーション機能では、タスク単位またはプログラム単位でシミュレーション対象を設定して、実行することができます。全体のプログラムのうち、シミュレーション対象の一部プログラムだけをシミュレータで動かすことができます。
シミュレーションプログラムを設定する操作の概要は以下のとおりです。

1　**シミュレーションウィンドウを表示させます。**
マルチビューエクスプローラの［プログラミング］–［タスク］の下のプログラムの左側に、シミュレーション対象設定用のチェックボックスが表示されます。



2　**タスク単位またはプログラム単位でシミュレーション対象の設定用チェックボックスをチェックします。**

デバッグプログラムの設定

デバッグプログラムとは、オフラインデバッグによる動作確認で使用するプログラムです。デバッグプログラムには、コントローラの外部からの仮想の入力処理やユーザ異常の強制発生などを記述します。デバッグプログラムは、シミュレータ上だけで実行できます。

デバッグプログラムを作成する操作の概要は以下のとおりです。

1　**マルチビューエクスプローラで［プログラミング］–［POU］–［プログラム］を右クリックし、メニューから［デバッグ用追加］–［ST］または［ラダー］を選択します。**
デバッグプログラムが新規作成されます。

2　**作成されたデバッグプログラムに、テスト用のプログラムコードを記述します。**

3　**デバッグプログラムをタスクに割り付けます。**

作成済みの通常のプログラムをデバッグプログラムに変更することもできます。

1　**マルチビューエクスプローラの［プログラミング］–［POU］–［プログラム］の各プログラムを右クリックして、［デバッグ用設定］–［有効］にします。**

シミュレーション速度の設定

シミュレーションウィンドウの「シミュレーション速度変更スライダ」で、0.1 ～ 1 倍の範囲でパソコン上での実行速度を変更できます。シミュレーション実行中、停止中のどちらでも変更できます。シミュレータ実行を実際よりゆっくりと表示したい場合に使用します。



7-3　実機での動作確認と運転

ここでは、実機での動作確認から運転開始までの手順の概要と、操作の概要について説明します。

7-3-1　実機での動作確認から運転開始までの手順

実機での動作確認から運転開始までの手順の概要は以下のとおりです。

STEP1　Sysmac Studio とのオンライン接続とプロジェクトのダウンロード	参照
1. コントローラの電源を投入します。 2. Sysmac Studio とオンライン接続します・ 3. Sysmac Studio からプロジェクト（ユーザプログラムやユニット構成などの各種設定）を転送します。	🕮『Sysmac Studio Version 1 オペレーションマニュアル（SBCA-470）』



STEP2　実機動作確認	参照
1. プログラムモードで、I/O マップまたはラダーエディタにて、実 I/O への強制値リフレッシュなどをして I/O の配線確認します。 2. モーション制御をする場合は、MC 試運転画面で、配線チェック、ジョグ送りでモータの方向確認、相対位置決めで移動量（電子ギアの設定）確認、原点復帰動作確認をします。 3. 運転モードにして、ユーザプログラムの動作確認をします。	🕮「7-3-3 I/O の配線確認」（P.7-9） 🕮『NY シリーズ 産業用パネル型 PC ／産業用ボックス型 PC ユーザーズマニュアル モーション制御編（SBCE-379）』 🕮「7-3-5 ユーザプログラムの動作確認」（P.7-10）



STEP3　コントローラの本運転	参照
1. 設計通りの動作であることを確認した上で、本運転を開始します。	−



参考

Sysmac Studio からプロジェクトをダウンロードするには、「同期機能」で、パソコンからコントローラへの転送を行います。「同期」の詳細は、🕮『Sysmac Studio　Version 1 オペレーションマニュアル（SBCA-470）』を参照してください。

7-3-2　プロジェクトの転送

Sysmac Studio から実機コントローラへプロジェクトを転送します。

1 **コントローラとオンライン接続の上、[コントローラ] メニューから [同期] を選択します。**
パソコン上のデータと実機コントローラのデータが自動照合されます。

2 **[転送 [パソコン→コントローラ]] ボタンをクリックします。**

7-3-3　I/O の配線確認

Sysmac Studio のウォッチウィンドウなどで強制値リフレッシュを実行して、I/O の配線を確認します。I/O の配線は、配線確認を行うスレーブまたはユニットの I/O に対して値を書き込み、結果をモニタすることで確認できます。

強制値リフレッシュの詳細は、🕮「8-4-1 強制値リフレッシュ」（P.8-27）を参照してください。

7-3-4　MC 試運転

MC 試運転とは、ユーザプログラムなしで、Sysmac Studio からおもに以下のことが行えます。

- 配線の確認：サーボドライバ接続コネクタの入出力信号のモニタ、
  サーボドライバの状態モニタ
- モータの動作および方向の確認：サーボ ON、ジョグ送り
- 電子ギアの設定確認：相対値位置決めを行い、移動量の確認／変更
- 原点復帰確認：原点復帰動作の確認

Sysmac Studio からコントローラにオンライン接続して MC 試運転画面にて行います。

詳細は、🕮『NY シリーズ 産業用パネル型 PC ／産業用ボックス型 PC ユーザーズマニュアル モーション制御編（SBCE-379）』を参照してください。

以下の操作を行います。

1 **必要な配線後、Sysmac Studio からコントローラにオンライン接続します。**

2 **「軸」の作成、軸の割付、以下の軸パラメータの設定をします。**
MC 試運転に必要な軸パラメータ設定：
表示単位選択、モータ 1 回転あたりのパルス数および移動量
最高速度、ジョグ最高速度、最大加速度・減速度、
ソフトウェアリミット機能選択、ソフトウェアリミット、カウントモード

3 **MC 試運転画面を開き、以下を行います。**
例）

- モニタ・配線確認
- ジョグ送りでモータの方向の確認
- 相対位置決めで移動量（電子ギアの設定）の確認
- 原点復帰動作の確認

7-3-5　ユーザプログラムの動作確認

実機を使ったユーザプログラムの動作確認は、コントローラの動作モードを運転モードにして行います。動作確認には、以下の項目があります。

- ユーザプログラムの動作を確認する
- オンラインエディットでユーザプログラムを修正する
- データトレースでユーザプログラム動作を確認する

ユーザプログラムの動作を確認する

ユーザプログラムの動作確認は以下の手順で行います。

1 **ユーザプログラムの実行状態をモニタする**

2 **接点やコイルの状態、変数の値を変更して動作確認する**

● ユーザプログラムの実行状態をモニタする

コントローラの接点やコイルの ON/OFF 状態、変数の現在値をモニタリングします。モニタリングは、Sysmac Studio のラダーエディタ、ウォッチウィンドウ、I/O マップで行うことができます。

● 接点やコイルの状態、変数の値を変更して動作確認する

ユーザプログラムの接点やコイルの ON/OFF 状態、変数の値を変更し、設計どおりの動作ができるかをモニタリングで確認します。接点やコイルの状態を変更するには「強制値リフレッシュ」、変数の値を変更するには「現在値変更」を使用します。

詳細は、📖「8-4-1 強制値リフレッシュ」（P.8-27）、📖「8-4-2 現在値変更」（P.8-31）を参照してください。

オンラインエディットでユーザプログラムを修正する

動作確認によって修正が必要と判断されたユーザプログラムを、「オンラインエディット」で修正します。「オンラインエディット」を使うとコントローラの運転を停止せずにユーザプログラムを変更することができます。

詳細は、📖「8-4-3 オンラインエディット」（P.8-33）を参照してください。

データトレースで動作を確認する

「データトレース」を使って、接点やコイルが ON/OFF するタイミングや、変数の値の変動などを確認します。

詳細は、📖「8-4-4 データトレース」（P.8-35）を参照してください。

7-3-6　本運転の開始

動作モードを 運転モードにして本運転を開始します。作成したユーザプログラムおよび各種データや設定値を十分に動作確認を行ったうえで、本運転に移行してください。

8

コントローラ内蔵機能

本章では、NY シリーズコントローラで使用できる機能について説明します。

8-1 データ管理・時計・運転関連機能 8-2
8-1-1 メモリオールクリア 8-2
8-1-2 時計機能 8-2

8-2 SD メモリカード機能 8-5
8-2-1 SD メモリカード機能の一覧 8-5
8-2-2 共有フォルダと仮想 SD メモリカードの設定 8-6
8-2-3 仮想 SD メモリカードの認識と認識解除 8-6
8-2-4 Windows の共有フォルダおよびファイル仕様 8-8
8-2-5 SD メモリカード命令 8-9
8-2-6 FTP クライアント通信命令 8-10
8-2-7 FTP サーバ機能 8-10
8-2-8 Sysmac Studio からのファイル操作 8-10
8-2-9 SD メモリカード関連のシステム定義変数一覧 8-11
8-2-10 仮想 SD メモリカード内のファイルアクセスの排他制御 8-12

8-3 セキュリティ機能 8-13
8-3-1 ユーザプログラム実行用 ID 認証機能 8-14
8-3-2 ユーザプログラム復元情報無し転送機能 8-17
8-3-3 プロジェクトファイル全体のプロテクト機能 8-18
8-3-4 データプロテクト機能 8-19
8-3-5 操作権限の認証機能 8-21
8-3-6 CPU ユニットへの書込プロテクト機能 8-23
8-3-7 CPU ユニット名称機能、およびシリアル ID 機能 8-24

8-4 デバッグ機能 8-27
8-4-1 強制値リフレッシュ 8-27
8-4-2 現在値変更 8-31
8-4-3 オンラインエディット 8-33
8-4-4 データトレース 8-35
8-4-5 微分モニタ 8-40

8-5 イベントログ機能 8-45
8-5-1 概要 8-45
8-5-2 イベントログ機能の詳細 8-46
8-5-3 コントローライベント（コントローラ異常／情報） 8-51
8-5-4 ユーザイベント（ユーザ異常／情報） 8-52

8-6 イベント重要度変更機能 8-58
8-6-1 イベント重要度変更の用途 8-58
8-6-2 イベント重要度変更が可能なイベント 8-58
8-6-3 イベント重要度変更の操作方法 8-59

8-1　データ管理・時計・運転関連機能

ここでは、データ管理機能、時計関連、運転関連機能について、説明します。

8-1-1　メモリオールクリア

Sysmac Studio から、NY シリーズコントローラ内のユーザプログラム、コントローラ構成・設定、変数、絶対値エンコーダ原点位置オフセットを工場出荷状態に初期化することができます。この操作を「メモリオールクリア」と呼びます。

使用上の注意

- メモリオールクリアは、プログラムモード時だけ実行可能です。
- セキュリティ機能の「CPU ユニットへの書込プロテクト」が設定されている場合、実行できません。
- メモリオールクリア中に NY シリーズコントローラの電源を OFF しないでください。

メモリオールクリア後は、工場出荷状態の NY シリーズコントローラでシステム構成を作成した直後と同様の動作となります。

● Sysmac Studio による操作

Sysmac Studio を NY シリーズコントローラにオンライン接続して、［コントローラ］メニューから［メモリオールクリア］を選択します。

操作の詳細については、🕮『Sysmac Studio Version 1 オペレーションマニュアル（SBCA-470）』を参照してください。

8-1-2　時計機能

ここでは、NY シリーズ産業用 PC の内蔵時計のうち、コントローラ側の時計について説明します。

概要

NY シリーズ産業用 PC では、Windows 側およびリアルタイム OS に組み込まれたコントローラ側のそれぞれが、時計とタイムゾーンを持っています。

時計については、Windows 側の時計の時刻がマスタークロックとなり、コントローラ側の時計の時刻は、Windows 側の時計に同期します。そのため、Windows 側の時計の時刻を変更することはできますが、コントローラ側の時計の時刻を変更することはできません。

Windows 側だけがシャットダウンするなど、Windows 側から時刻を取得できない場合は、コントローラ側の時計の時刻は、Windows 側の時計に同期しません。同期しない間でも、コントローラ側の時計は動作を継続しています。
Windows を再起動するなど、Windows 側から時刻を取得できるようになると、コントローラ側の時計の時刻は、再度 Windows 側の時計に同期します。

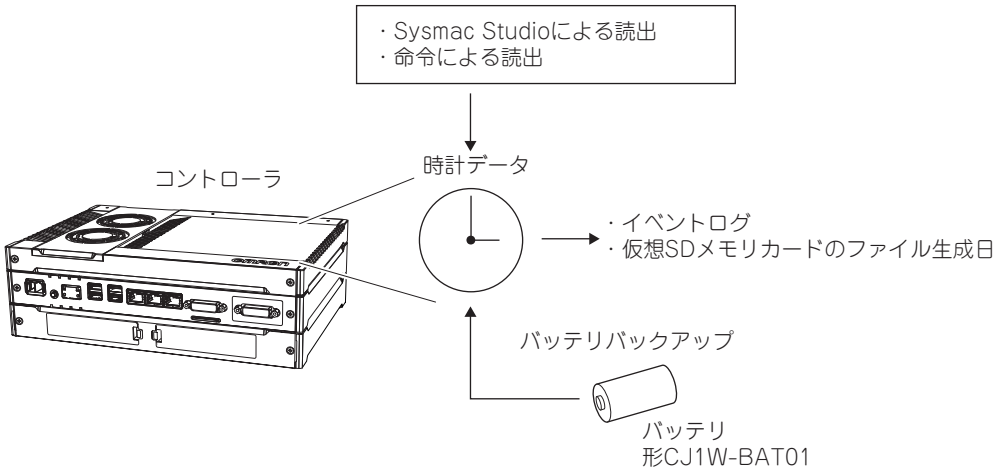
コントローラ側の時計の時刻データが、イベントログや仮想 SD メモリカードのファイル生成日時などに反映されます。

使用上の注意

コントローラ側の時計の時刻が Windows 側の時計に同期すると、コントローラ側の時計の時刻は補正され、時刻が進んだり、過去の時刻に戻る可能性があります。時刻データを用いたプログラムが誤動作しないことを確認してください。また、時刻データを伴う履歴確認の際は注意してください。

また、コントローラ側のタイムゾーンは、Windows 側のタイムゾーンに従います。
Windows 側のタイムゾーンとコントローラ側のタイムゾーンを、個別に設定することはできません。

コントローラ側の時計については、以下の機能が可能です。
- Sysmac Studio からの読出（書込は不可）
- 命令からの読出（書込は不可）
- システム定義変数からの読出（書込は不可）



● コントローラ側の時刻データの範囲

- 2000-01-01 ～　2099-12-31

● タイムゾーンとローカルタイムの設定

コントローラ側のタイムゾーンとローカルタイム（時刻データ）は、Windows 側の設定に従います。そのため、初めて使用するときに、Windows 側でタイムゾーンとローカルタイム（時刻データ）を登録してください。
EtherCAT スレーブなどが、コントローラから読み出す時刻データ、および設定する時刻データは、タイムゾーンに応じたローカルタイムとなります。

参考

バッテリが装着されていないとき、またはバッテリ切れのとき、タイムゾーンの設定自体は保持されますが、時刻データは保持されず不正な値になります。

時刻データの読出方法

時刻データが不正な場合は、そのまま不正値が読み出されます。

● 命令からの読出（書込は不可）

ユーザプログラム内では、GetTime 命令を用いて時刻データを読み出すことができます。

● システム定義変数からの読出（書込は不可）

以下のシステム定義変数を使用して、時刻データを読み出すことができます。
「_CurrentTime」（システム時刻）

● Sysmac Studio による読出（書込は不可）

Sysmac Studio を使用して、［コントローラ］メニューから ［コントローラ時計］にて、時刻データを読み出し、表示させることができます。

関連システム定義変数

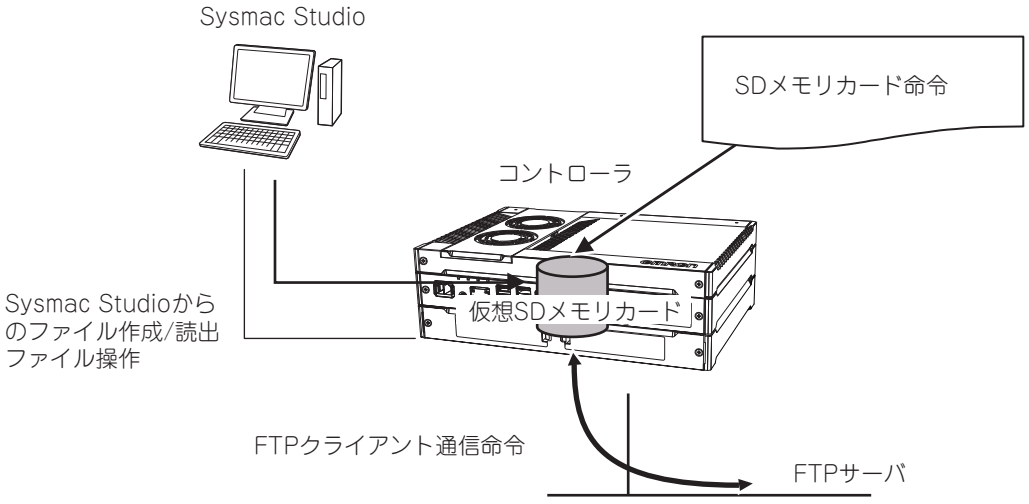
変数名	名称	機能	データ型	R/W
_CurrentTime	システム時刻	コントローラの内部時計データが格納されます。	DATE_AND_TIME	R

8-2　SD メモリカード機能

ここでは、SD メモリカード機能について、説明します。
NY シリーズコントローラでは、Windows との共有フォルダを仮想的な SD メモリカードとして扱います。本マニュアルでは「仮想」を省略し、単に「SD メモリカード」と記載する場合があります。

8-2-1　SD メモリカード機能の一覧

NY シリーズコントローラでサポートする SD メモリカード機能は以下のとおりです。



機能名	概要
SD メモリカード命令	ユーザプログラム上の命令から仮想 SD メモリカードにアクセスする機能です。
FTP クライアント通信命令	コントローラから、Ethernet 上のコンピュータやコントローラに対して、FTP によるファイル転送を行うための命令です。
FTP サーバ機能	イントラネット内の FTP クライアントから、FTP コマンドを使用して、EtherNet/IP 経由で仮想 SD メモリカード内の大容量ファイルを読み書きする機能です。
Sysmac Studio からのファイル操作機能*1	Sysmac Studio からコントローラの仮想 SD メモリカードに対して、ファイル操作を行う機能です。 ユーザは、仮想 SD メモリカードにコントローラ用ファイルだけでなく、汎用のドキュメントファイルなども保存可能です。
SD メモリカードバックアップ機能	コントローラの仮想 SD メモリカードを使用して、コントローラのユーザプログラムやデータなどを、バックアップやリストア、照合する機能です。 詳細は、「9-2 SD メモリカードバックアップ機能」（P.9-10）を参照してください。

*1. NY シリーズコントローラでは、SD メモリカードの初期化、フォルダやファイルの属性変更はできません。また、仮想 SD メモリカードは共有フォルダを使用しているため、Sysmac Studio の SD メモリカードウィンドウのステータスバーにボリュームラベルは表示されません。

8-2-2　共有フォルダと仮想 SD メモリカードの設定

NY シリーズコントローラで SD メモリカード機能を使用するための設定について説明します。

共有フォルダの設定

仮想SDメモリカードの設定を行う前に、Windows側で共有フォルダを設定しておく必要があります。NY シリーズ産業用 PC の Windows を初期設定した状態では、仮想 SD メモリカード用の共有フォルダとして、「D:\OMRON-NY\VirtualSDCard」が設定されています。

共有フォルダの設定の詳細については、🕮『NY シリーズ 産業用パネル型 PC ／産業用ボックス型 PC ユーザーズマニュアル セットアップ編（SBCA-441）』を参照してください。

仮想 SD メモリカードの設定

仮想 SD メモリカードの設定は、Sysmac Studio のコントローラ設定で行います。
仮想メモリカード設定の設定項目については、🕮「 仮想 SD メモリカード設定（P.4-5）」を参照してください。



参考

Symac Studio を使用せずに、Industrial PC Support Utility を使用して、仮想 SD メモリカード設定を行うことができます。
Industrial PC Support Utility による仮想 SD メモリカードの設定方法については、🕮『NY シリーズ 産業用パネル型 PC ／産業用ボックス型 PC ユーザーズマニュアル セットアップ編（SBCA-441）』を参照してください。

8-2-3　仮想 SD メモリカードの認識と認識解除

仮想 SD メモリカードの認識と認識解除について説明します。

仮想 SD メモリカードの認識

次の操作を実行した場合に仮想 SD メモリカードを認識します。
仮想 SD メモリカードを認識すると、システム定義変数「_Card1Ready（SD メモリカード使用可フラグ）が TRUE となります。

- NY シリーズコントローラの起動
- ダウンロードの実行
- リストアの実行
- Industrial PC Support Utility による仮想 SD メモリカードの設定変更
- Industrial PC Support Utility による仮想 SD メモリカードの認識確認

次の場合は、仮想 SD メモリカードを認識できません。

- Windows の停止
- Windows のファイル共有サービスの停止
- 仮想 SD メモリカード設定の不正

仮想 SD メモリカードの認識解除

次の場合に仮想 SD メモリカードの認識が解除されます。

要因
Windows の停止（再起動含む）
Windows の共有サービスの停止・無効化
Windows のコンピュータ名変更
Windows の内部ポートの IP アドレス変更
共有フォルダの共有解除
共有フォルダの削除
共有フォルダのアクセス権変更
コントローラ側の内部通信ポートの IP アドレスを、Windows 側の内部通信ポートと異なる IP アドレスに変更

仮想 SD メモリカードの認識が解除されると、システム定義変数「_Card1Ready（SD メモリカード使用可フラグ）は FALSE となり、「共有フォルダ認識解除完了」のイベントが発生します。
仮想 SD メモリカードにアクセス中に認識が解除された場合、実行中のファイルアクセスは失敗となります。

解除された要因により、解除後の動作が異なります。

●「共有フォルダの共有解除」、または「共有フォルダの削除」が要因の場合

仮想 SD メモリカードの認識が解除される要因が「共有フォルダの共有解除」、または「共有フォルダの削除」の場合、Windows 側で共有フォルダを再共有するだけでは、再度、認識状態となりません。
以下いずれかにより、再度、認識状態となります。

- 共有フォルダを再共有し、かつ、Industrial PC Support Utility から［認識確認］を実行する。ただし、共有が解除される以前に SD メモリカード命令でファイルオープンしている場合、対象のファイルをクローズしている必要があります。
- 共有フォルダを再共有し、かつ、共有解除から、Windows のセッション・最大アイドル時間（初期値 15 分）経過している。

●「共有フォルダの共有解除」、「共有フォルダの削除」以外が要因の場合

要因を解消し、復旧したときに自動的に認識状態となります。

仮想 SD メモリカードが認識されている場合に、ダウンロード、リストア、メモリオールクリア、あるいは Industrial PC Support Utility から設定を転送すると、以下のようになります。

- 仮想 SD メモリカードの認識が解除
- システム定義変数「_Card1Ready（SD メモリカード使用可フラグ）」が FALSE
- 「共有フォルダ認識解除完了」のイベントが発生

ただし、仮想 SD メモリカードにアクセス中の場合は、仮想 SD メモリカードの認識は解除されず、「共有フォルダ認識解除失敗」のイベントが発生します。

使用上の注意

仮想メモリカードにアクセス中に、産業用 PC の電源を OFF にしないでください。
電源 OFF すると、ファイルが壊れる場合があります。
また、次回起動時に、システム定義変数「_Card1PowerFail（SD メモリカードアクセス中電源断フラグ）」が TRUE となります。

8-2-4　Windows の共有フォルダおよびファイル仕様

仮想 SD メモリカードとして使用する共有フォルダとファイルの仕様について説明します。

共有フォルダの仕様

Windows 側で設定する共有フォルダの仕様について説明します。

● 設定可能なストレージ

産業用 PC に内蔵されているストレージデバイスに設定した共有フォルダを、仮想 SD メモリカードとして使用できます。
産業用 PC に内蔵されているストレージデバイス以外に設定した共有フォルダは、仮想 SD メモリカードの動作保証外です。

● 共有フォルダの容量の制約

仮想 SD メモリカードとして使用する共有フォルダは、16GB 以内 (*) で使用してください。
共有フォルダ内のファイルの容量が制約を超える場合、正しく動作しない可能性があります。

* コントローラのユニットバージョンが Ver.1.19 以前の場合、4GB 以内で使用してください。

● ファイルシステム

仮想 SD メモリカードとして使用できるファイルシステムは、NTFS だけです。

フォルダ／ファイルの仕様

● 文字制限

ユーザが名前を付けるもの	使用できる文字	予約語	マルチバイト文字使用可否	大文字／小文字の区別	最大サイズ（終端の NULL 文字含まず）
ディレクトリ名	0～9、A～Z、a～z $ % ' - _ @ ! ` ( ) ~ = # & + ^ [ ] { } , . ;	CON, PRN, AUX, CLOCK$, NUL, COM0, COM1, COM2, COM3, COM4, COM5, COM6, COM7, COM8, COM9, LPT0, LPT1, LPT2, LPT3, LPT4, LPT5, LPT6, LPT7, LPT8, LPT9	不可*1	しない	65 バイト
ファイル名					65 バイト

*1. マルチバイト文字（日本語等）は、パソコン上で使用できますが、コントローラでは使用できません。

使用上の注意

ディレクトリ名やファイル名に、使用できる文字以外の文字を含む場合、コントローラは正しく認識できません。命令、Sysmac Studio の SD メモリカードウィンドウ、および ＦＴＰ サーバー機能で、ファイルを特定できない場合があります。

● サブディレクトリ階層

最大 5 階層まで
例）共有フォルダの共有名を「VirtualSDCard」とした場合
VirtualSDCard\f1\f2\f3\f4\f5\abc.txt

● 格納するファイル数の制約

仮想 SD メモリカードに格納するファイル数は、下表の値以下で使用してください。ファイル数が下表の値を超える場合、正しく動作しない可能性があります。

ディレクトリ階層	格納するファイル数
ルートディレクトリ	65,535
サブディレクトリ	65,534

● 1 ファイルの最大サイズ

1 ファイルの最大サイズは、2,147,483,647 バイト (2GB-1 バイト) です。

8-2-5　SD メモリカード命令

以下の命令によって、仮想 SD メモリカードに対する各種操作を行うことができます。

命令名称	命令	機能
変数ファイル読出	**FileReadVar**	仮想 SD メモリカード内のファイル（バイナリ形式）を読み出し、指定した変数（配列変数や構造体変数も可）に格納する。
変数ファイル書込	**FileWriteVar**	指定した変数（配列変数や構造体変数も可）の値をバイナリ形式で仮想 SD メモリカード内のファイルに書き込む。 ファイル名に指定したディレクトリが存在しない場合は、そのディレクトリ自体も作成する。
ファイルオープン	**FileOpen**	指定したファイルをオープンする。
ファイルクローズ	**FileClose**	指定したファイルをクローズする。
ファイルシーク	**FileSeek**	指定したファイルのファイル位置指示子を設定する。
ファイル読出	**FileRead**	指定したファイルからデータを読み出す。
ファイル書込	**FileWrite**	指定したファイルにデータを書き込む。
文字列読出	**FileGets**	指定したファイルから 1 行分の文字列を読み出す。
文字列書込	**FilePuts**	指定したファイルに 1 行分の文字列を書き込む。
ファイル削除	**FileRemove**	仮想 SD メモリカード内の指定したファイルを削除する。
ファイル名変更	**FileRename**	指定したファイルのファイル名またはディレクトリ名を変更する。
ファイルコピー	**FileCopy**	指定したファイルを別のファイル名で複製する。
ディレクトリ作成	**DirCreate**	仮想 SD メモリカード内にディレクトリを作成する。
ディレクトリ削除	**DirRemove**	仮想 SD メモリカード内の指定したディレクトリを削除する。

8-2-6　FTP クライアント通信命令

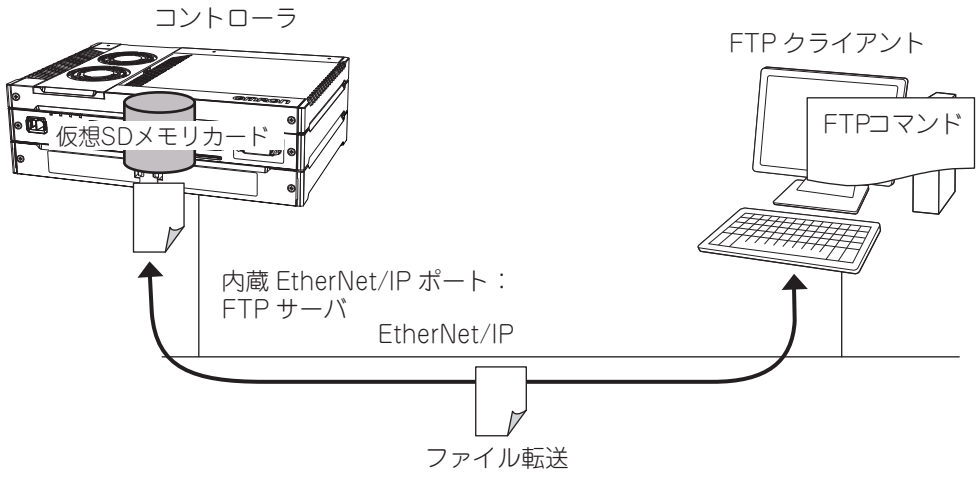
FTP クライアント通信命令とは、NY シリーズコントローラから、Ethernet 上のコンピュータやコントローラに対して、FTP によるファイル転送を行うための命令のことです。以下の命令では、命令実行の際に、仮想 SD メモリカードのファイルの読み書きが行われます。

命令名称	命令	機能
FTP サーバへのファイルアップロード	**FTPPutFile**	FTP クライアントの仮想 SD メモリカード内にあるファイルを、FTP サーバにアップロードする。
FTP サーバからのファイルダウンロード	**FTPGetFile**	FTP サーバ内のファイルを、FTP クライアントの仮想 SD メモリカードにダウンロードする。

8-2-7　FTP サーバ機能

FTP クライアントから、内蔵 EtherNet/IP ポートに対して FTP コマンドを発行することで、EtherNet/IP 経由で、仮想 SD メモリカード内のファイルを読み書きすることができます。

詳細は、🕮『NY シリーズ 産業用パネル型 PC ／産業用ボックス型 PC ユーザーズマニュアル 内蔵 EtherNet/IP ポート編 （SBCD-369)』を参照してください。



8-2-8　Sysmac Studio からのファイル操作

Sysmac Studio から NY シリーズコントローラの仮想 SD メモリカードに対して、ファイル操作を行うことができます。

仮想 SD メモリカードには、コントローラ用ファイルだけでなく、ドキュメントファイル等も置くことができます。

8-2-9　SD メモリカード関連のシステム定義変数一覧

SD メモリカード機能としての状態を通知するシステム定義変数は、以下のとおりです。

変数名	名称	機能	データ型
_Card1Ready	SD メモリカード使用可フラグ	NY シリーズでは、仮想 SD メモリカードの認識ができたとき、TRUE となります。 認識ができなかったとき、FALSE となります。 TRUE：使用可 FALSE：使用不可	BOOL
_Card1Protect	SD メモリカードライトプロテクトフラグ	NY シリーズでは、使用しません。 常に FALSE となります。	BOOL
_Card1Err	SD メモリカードエラーフラグ	NY シリーズでは、使用しません。 常に FALSE となります。	BOOL
_Card1Access	SD メモリカードアクセス中フラグ	NY シリーズでは、使用しません。 常に FALSE となります。	BOOL
_Card1Deteriorated	SD メモリカード寿命警告フラグ	NY シリーズでは、使用しません。 常に FALSE となります。	BOOL
_Card1PowerFail	SD メモリカードアクセス中電源断フラグ*1	仮想 SD メモリカードアクセス中にコントローラの電源断が発生したとき、TRUE となります。 TRUE：仮想 SD メモリカードアクセス中に電源断があった FALSE：正常	BOOL
_Card1Capacity	SD メモリカードストレージ容量	仮想 SD メモリカードの全容量を示します。 単位は MiB (1MiB=1,048,576Byte) です。 値は 60 秒ごとに更新されます。 仮想 SD メモリカードが無効の場合、“0”になります。	UDINT
_Card1Used	SD メモリカードストレージ使用量	仮想 SD メモリカードの使用量を示します。 単位は MiB (1MiB=1,048,576Byte) です。 値は 60 秒ごとに更新されます。 仮想 SD メモリカードが無効の場合、“0”になります。	UDINT

*1. SD メモリカードアクセス中電源断フラグ（「_Card1PowerFail」）使用時の注意点
SD メモリカードアクセス中電源断フラグが TRUE となった場合は、仮想 SD メモリカード内に正しいファイルが存在するか、また装置が正しく動作するか確認してください。
正しいファイルが存在しない場合、または装置が正しく動作しない場合は、仮想 SD メモリカードへ正しいファイルをダウンロードしてください。コントローラを電源再投入またはコントローラリセットして装置が正しく動作することを確認してください。
処置が完了したら、SD メモリカードアクセス中電源断フラグを FALSE にしてください。（本フラグは自動的に FALSE になりません）。

（注） SDメモリカードバックアップ機能に関連するシステム定義変数は、📖「9-2 SDメモリカードバックアップ機能」（P.9-10）を参照してください。

8-2-10　仮想 SD メモリカード内のファイルアクセスの排他制御

仮想 SD メモリカード内のファイルアクセス方法には、以下の 4 種類があります。

(1) FTP サーバ機能

(2) SD メモリカード命令

(3) FTP クライアント通信命令

(4) Sysmac Studio からのファイル操作

仮想 SD メモリカード内のファイルアクセスを、同一ファイルに対して同時に行うと、書込途中のファイルを読み出したり、読出途中のファイルに上書きしたりするなど、ユーザが意図しない処理が行われる可能性があります。したがって、これらの処理が同一ファイルに対して同時に行われないよう、排他制御を行う必要があります。以下に、排他制御が必要になる処理の組み合わせを示します。排他制御が必要な組み合わせの処理を行う場合は、一方の処理が終了したことを確認してから、もう一方の処理を実行してください。

<table>
<tr><th colspan="3" rowspan="3"></th><th colspan="6">先に実行中のアクセス</th></tr>
<tr><th colspan="2">命令*1</th><th colspan="2">Sysmac Studio からのファイル操作</th><th colspan="2">FTP サーバ機能</th></tr>
<tr><th>読出</th><th>書込</th><th>読出</th><th>書込</th><th>読出</th><th>書込</th></tr>
<tr><td rowspan="6">後から実行するアクセス</td><td rowspan="2">命令</td><td>読出</td><td colspan="2" rowspan="2">自動的に排他制御され、後から実行する命令がエラーとなります。</td><td>排他制御は不要です。</td><td>排他制御を行ってください。</td><td>排他制御は不要です。</td><td>排他制御を行ってください。</td></tr>
<tr><td>書込</td><td colspan="2">排他制御を行ってください。</td><td colspan="2">排他制御を行ってください。</td></tr>
<tr><td rowspan="2">Sysmac Studio からのファイル操作</td><td>読出</td><td>排他制御は不要です。</td><td>排他制御を行ってください。</td><td>排他制御は不要です。</td><td>排他制御を行ってください。</td><td>排他制御は不要です。</td><td>排他制御を行ってください。</td></tr>
<tr><td>書込</td><td colspan="2">排他制御を行ってください。</td><td colspan="2">排他制御を行ってください。</td><td colspan="2">排他制御を行ってください。</td></tr>
<tr><td rowspan="2">FTP サーバ機能</td><td>読出</td><td>排他制御は不要です。</td><td>排他制御を行ってください。</td><td>排他制御は不要です。</td><td>排他制御を行ってください。</td><td>排他制御は不要です。</td><td>排他制御を行ってください。</td></tr>
<tr><td>書込</td><td colspan="2">排他制御を行ってください。</td><td colspan="2">排他制御を行ってください。</td><td colspan="2">排他制御を行ってください。</td></tr>
</table>

*1. 命令には、SD メモリカード命令と FTP クライアント通信命令が含まれます。

NY シリーズコントローラでは、Windows 側から共有フォルダのファイルにアクセスできます。
この場合の排他制御については、Windows のファイル共有の仕様に従います。

8-3　セキュリティ機能

NY シリーズコントローラで使用できるセキュリティ機能について、説明します。

セキュリティ機能を使うと、資産の保護のためにコントローラ内のユーザプログラムや各種データを保護することができます。また、誤操作の防止のために Sysmac Studio での操作を制限したりすることもできます。

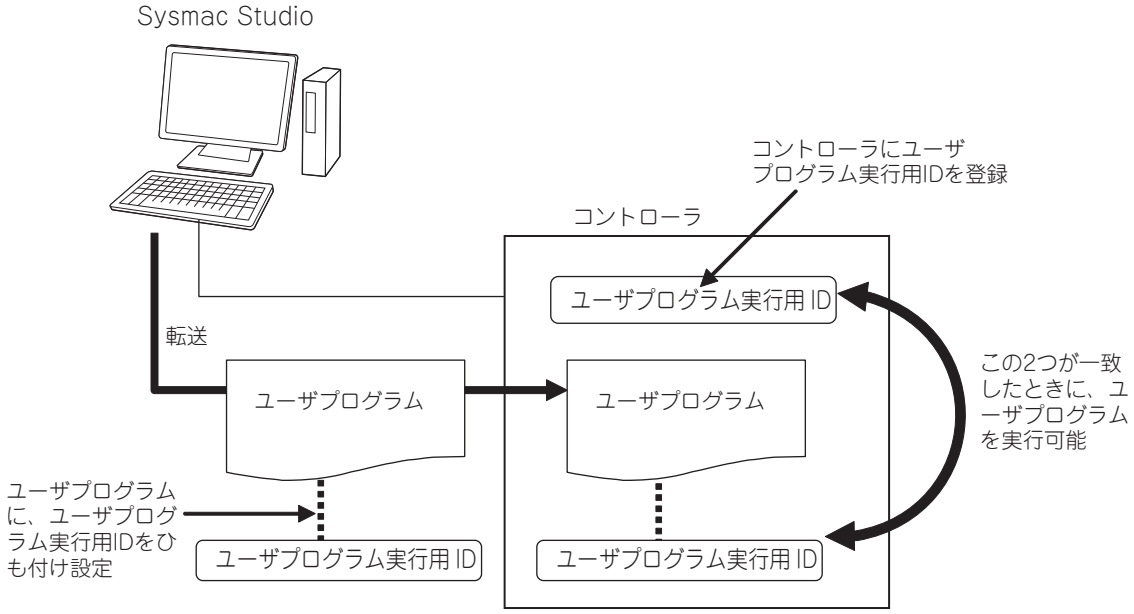
NY シリーズコントローラは以下のセキュリティ機能を持っています。

目的	セキュリティ機能	機能概要	参照
資産の盗用防止	ユーザプログラム実行用 ID 認証機能	ユーザプログラムを複製しても、別のコントローラでは動作させないようにします。	「8-3-1 ユーザプログラム実行用 ID 認証機能」（P.8-14）
	ユーザプログラム復元情報なし転送機能	コントローラにプログラムの復元情報（ソース）を転送しないことで、他のパソコンからアップロードしたときにプログラムを表示できないようにします。	「8-3-2 ユーザプログラム復元情報無し転送機能」（P.8-17）
	プロジェクトファイル全体のプロテクト機能	プロジェクトファイルにパスワードプロテクトをかけることにより、ユーザ資産情報を保護します。	「8-3-3 プロジェクトファイル全体のプロテクト機能」（P.8-18）
	データプロテクト機能	プロジェクトのうち、保護したいデータにプロテクトをかけることによりユーザ資産情報を保護します。	「8-3-4 データプロテクト機能」（P.8-19）
誤操作防止	操作権限の認証機能	Sysmac Studio からコントローラに対して操作権限を設定することで、操作できる機能を制限します。	「8-3-5 操作権限の認証機能」（P.8-21）
	CPU ユニットへの書込プロテクト機能	コントローラ内のデータを Sysmac Studio から書き換えできないようにします。	「8-3-6 CPU ユニットへの書込プロテクト機能」（P.8-23）
誤接続防止	CPU ユニット名称機能	パソコンとコントローラのコントローラの名称またはシリアル ID の一致確認をすることで、誤ってオンライン接続することを防止します。	「8-3-7 CPU ユニット名称機能、およびシリアル ID 機能」（P.8-24）

8-3-1　ユーザプログラム実行用 ID 認証機能

概要

あらかじめコントローラに特定の ID（「ユーザプログラム実行用 ID」と呼ぶ）を設定しておき、それと同じ ID をひも付け設定したユーザプログラムだけを実行することができます。



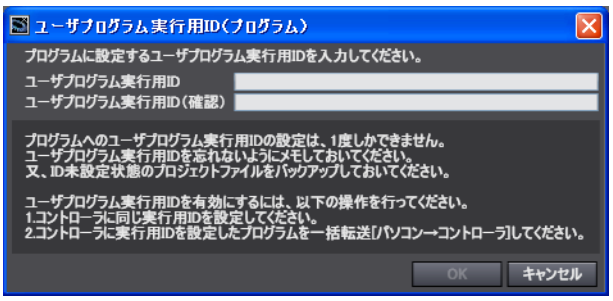
これにより、あるユーザプログラムを、異なるコントローラ（ハードウェア）では使用できないようにすることができます。

なお、本機能は、プロテクト機能とは異なり、プログラムの表示・編集は可能です。

操作方法

1　あらかじめ、まだユーザプログラム実行用 ID を紐付け設定していないプロジェクトファイルを必ずバックアップしておきます。

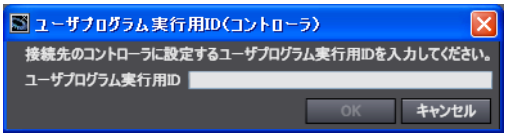
2　Sysmac Studio にて、オフラインで、ユーザプログラム側にユーザプログラム実行用 ID を紐付け設定します。



使用上の注意

一度ユーザプログラム実行用 ID を紐付け設定したユーザプログラムは、その ID を変更および削除することができません。
もし変更したい場合は、手順 1 であらかじめバックアップしたプロジェクトファイルを読み出して、再度、異なるユーザプログラム実行用 ID を紐付け設定してください。削除したい場合は、手順 1 であらかじめバックアップしたプロジェクトファイルを使用してください。

3 **Sysmac Studio をオンライン接続して、コントローラ側に、 2 で紐付け設定したユーザプログラム実行用 ID を登録します。**



イベントログに、ユーザプログラム実行用 ID のコントローラへの登録が記録されます。
このとき、ユーザプログラム実行用IDがすでに登録されている場合でも上書きで登録されます。

4 **同じユーザプログラム実行用 ID を紐付け設定したユーザプログラムを、コントローラに転送します。**
もし、ユーザプログラム側とコントローラ側のユーザプログラム実行用 ID が一致していない場合、または片方に存在しない場合は、運転モードに変更したときに ID 照合エラー（全停止フォールトレベルのコントローラ異常）が発生し運転できません。

使用上の注意

一度ユーザプログラム実行用 ID を紐付け設定したコントローラは、その ID を削除および読み出すことはできません。万一削除したい場合は、コントローラをメモリオールクリアしてください。

● **「ID 照合エラー」発生時の操作**

コントローラ側のユーザプログラム実行用 ID が正しくない、または登録されていないとき：

Sysmac Studio からコントローラにオンライン接続して、

1 **正しいユーザプログラム実行用 ID をコントローラに登録してください。**

2 **コントローラの電源を OFF → ON、または Sysmac Studio からリセット操作をしてください。**

ユーザプログラム側のユーザプログラム実行用 ID が正しくない、または紐付け設定されていないとき：

1 **バックアップしていたプロジェクトファイルを Sysmac Studio で読み出して、正しいユーザプログラム実行用 ID を紐付け設定してください。**

2 **Sysmac Studio をコントローラにオンライン接続して、ユーザプログラムを転送してください。**

3 **コントローラの電源を OFF → ON、または Sysmac Studio からコントローラリセット操作を行ってください。**

● **その他のとき**

ユーザプログラム側のユーザプログラム実行用 ID を削除したいとき：
バックアップしていたプロジェクトファイルを Sysmac Studio で読み出してください。

コントローラ側のユーザプログラム実行用 ID を削除したいとき：
Sysmac Studio をコントローラにオンライン接続してメモリオールクリアしてください。

ユーザプログラム側のユーザプログラム実行用 ID を確認したいとき：
セキュリティの観点から、Sysmac Studio などから、ユーザプログラムに紐付け設定したユーザプログラム実行用 ID を確認することはできないようになっています。バックアップしておいたプロジェクトファイルをもとに、紐付け設定をし直してください。

コントローラ側のユーザプログラム実行用 ID を確認したいとき：
セキュリティの観点から、Sysmac Studio などから、コントローラに登録したユーザプログラム実行用 ID を確認することはできません。メモリオールクリアの上、登録をし直してください。

仕様

● **ユーザプログラム実行用 ID の照合の仕様**

照合のタイミング
コントローラは、運転開始時に、コントローラに登録されたユーザプログラム実行用 ID と、ユーザプログラムに紐付け設定されているユーザプログラム実行用 ID を照合します。

照合の条件
照合の条件は以下のとおりです。
A、B は、ID を意味します。

コントローラ側に登録したユーザプログラム実行用 ID	ユーザプログラム側に紐付け設定したユーザプログラム実行用 ID	異常	運転
A	A	なし	可能
なし	なし		
なし	A	ID 照合エラー	不可
A	なし		
A	B		

照合の不一致時の動作
照合不一致のとき、「ID 照合エラー」（全停止フォールトレベルのコントローラ異常）となり、コントローラは運転できません。
異常を解除するためには、コントローラの電源を OFF → ON にするか、Sysmac Studio からコントローラリセットを行う必要があります。

● **ユーザプログラム実行用 ID で使用できる文字の仕様**

使用できる文字	大文字／小文字の区別	最大サイズ（NULL 含まず）
0～9、A～Z、a～z	する	8～32 文字

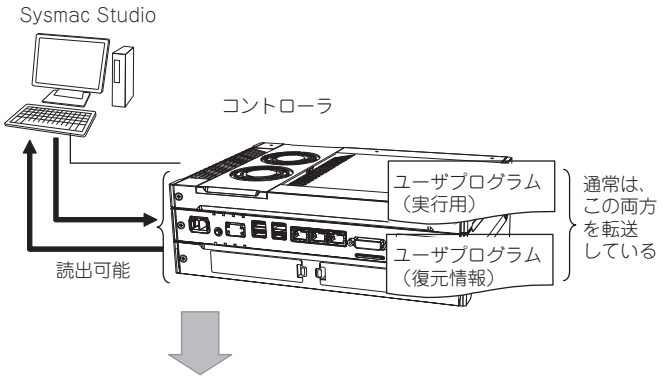
8-3-2　ユーザプログラム復元情報無し転送機能

コントローラにプログラムの復元情報（ソース）を転送しないことで、ほかのパソコンからアップロードしたときにプログラムを表示できないようにします。

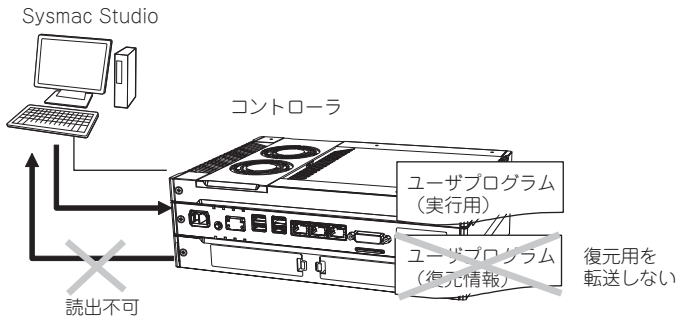
概要

通常は、Sysmac Studio からコントローラにユーザプログラムを転送するときに、その復元のための情報も転送しています。
本機能は、このときに、復元のための情報を転送しないことで、ユーザプログラムを読み出すことを不可能にする機能です。

●通常の転送時



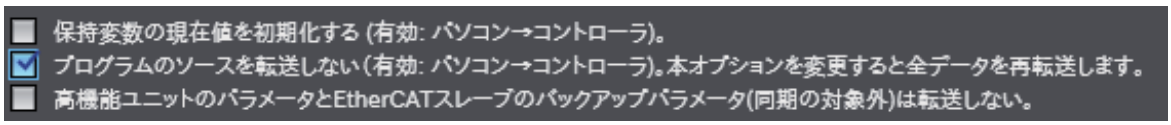
●ユーザプログラム復元情報なし転送時



現場でユーザプログラムを保守することがなく、かつユーザプログラムデータの盗用を防止したいときに使用します。

操作方法

ユーザプログラムをコントローラに転送するときに Sysmac Studio の同期ウィンドウで、［プログラムのソースを転送しない］設定をチェックして、［転送［パソコン｜コントローラ］］ボタンをクリックします。

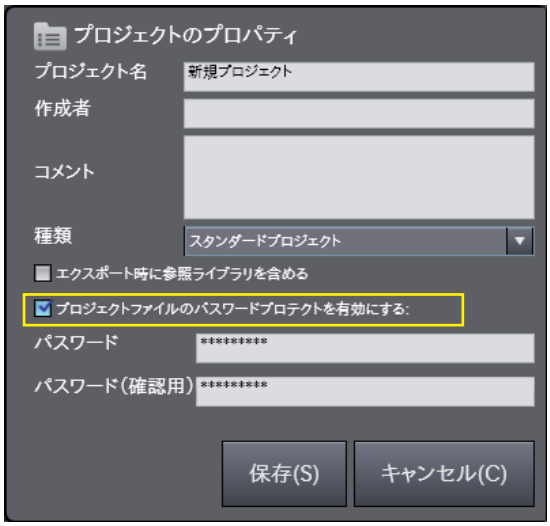


8-3-3　プロジェクトファイル全体のプロテクト機能

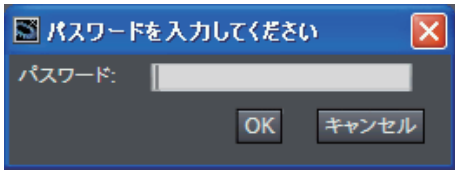
プロジェクトファイルにパスワードプロテクトをかけることにより、ユーザ資産情報を保護します。

操作方法

プロジェクトにパスワードを設定するときの操作を説明します。
プロジェクトファイルに名前を付けて保存するときに［プロジェクトファイルのパスワードプロテクトを有効にする］にチェックを入れることでパスワード設定ができるようになります。



パスワードを設定したプロジェクトを開くときの操作を説明します。
パスワード設定されたプロジェクトファイルを開くとき、またはインポートするときに［パスワード入力］ダイアログボックスが表示されます。



この機能の操作の詳細については、🕮『Sysmac Studio Version 1 オペレーションマニュアル (SBCA-470)』を参照してください。

8-3-4　データプロテクト機能

プロジェクトのデータのうち保護したいデータにプロテクトをかけることにより、ユーザ資産情報を保護します。

概要

プロジェクトのデータのうち保護したいデータの単位でプロテクトを設定し、データへのアクセスを制限します。
プロテクトを設定するときにアクセス制限の強度を 3 種類から選択して設定します。
アクセス制限をかけたデータにアクセスするにはプロテクトの一時解除が必要です。一時解除の有効期間は行った操作によって異なります。

プロテクト対象データ

プロテクト設定ができるデータは以下のとおりです。

- ラダー図プログラム（プログラム / ファンクション / ファンクションブロック共通）
- ST プログラム（プログラム / ファンクション / ファンクションブロック共通）
- カムプロファイル

アクセス制限の種類

アクセス制限の強度は以下の 3 種類から選択します。ただし、カムプロファイルは変更禁止しか選択できません。

- 複製、表示、変更禁止
- 表示、変更禁止
- 変更禁止

データへのアクセス操作と、アクセス制限の種類ごとの制限とプロテクトを一時解除したときの有効期間との関係は以下のとおりです。

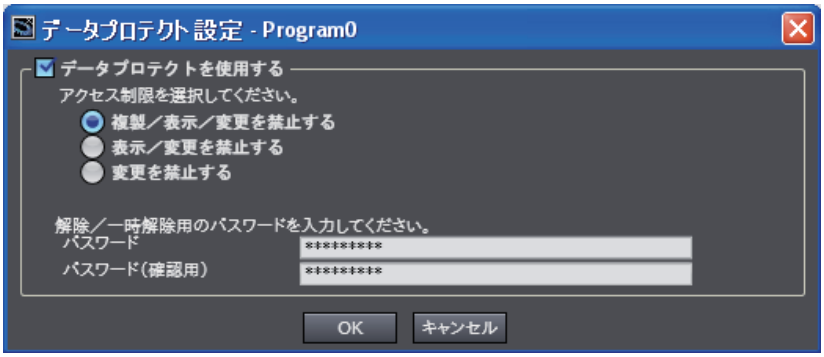
（○：操作が制限できる、－：操作が制限できない）

データアクセスの操作	アクセス制限の種類			一時解除有効期間
	複製、表示、変更禁止	表示、変更禁止	変更禁止	
対象物の表示	○	○	－	プロジェクトを開いている間
対象物の印刷	○	○	－	
対象物の変更	○	○	○	
対象物のコピー	○	－	－	操作ごとに一時解除が必要
照合結果（概要）の表示	－	－	－	この操作は制限されません
照合結果（詳細）の表示	○	○	－	プロジェクトを開いている間
イベントログやクロスリファレンスから内容へのジャンプ	○	○	－	
ライブラリへの登録	○	－	－	一時解除はできません

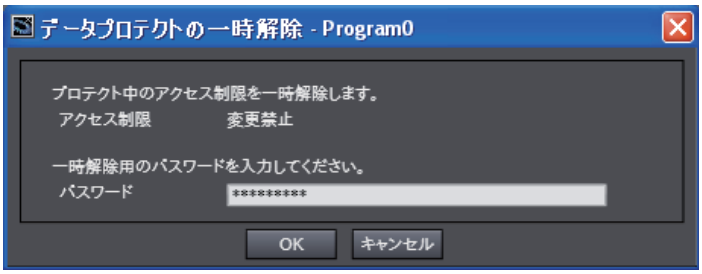
操作方法

この機能は Sysmac Studio をオフラインで操作します。
設定内容はプロジェクトに保存されます。Sysmac Studio の同期機能でプロジェクトを転送すると、転送元（パソコン／コントローラ）の対象データのデータプロテクト設定が転送先（コントローラ／パソコン）に転送されます。

プロテクト設定をするには、Sysmac Studio の［コントローラ］-［セキュリティ］-［データプロテクト設定／解除］で設定をします。



プロテクトの一時解除をするには、［コントローラ］-［セキュリティ］-［変更禁止の一時解除］を選択します。



プロテクトの一時解除を終了するには、［コントローラ］-［セキュリティ］-［変更禁止の一時解除の終了］を選択します。

この機能の操作の詳細については、🕮『Sysmac Studio Version 1 オペレーションマニュアル (SBCA-470)』を参照してください。

8-3-5　操作権限の認証機能

概要

操作ミスによっては装置や人に危害を加える可能性があるため、操作権限に応じて、オンラインでの操作機能を制限する機能です。
たとえば、以下です。

- I/O モニタ　　：書込、強制値リフレッシュなど
- コントローラ操作　：動作モード変更、オンラインエディット、軸試運転など

Sysmac Studio でコントローラに操作権限のパスワードを登録し、オンライン接続時に操作権限のパスワードを入力すると、操作権限カテゴリに応じたオンラインでの操作機能が使用可能となります。

管理者が操作権限ごとにパスワードを設定し、スキルに応じて操作権限名とパスワードを使用者に連絡することで運用します。

操作権限関連設定の操作の詳細は、🕮『Sysmac Studio Version 1 オペレーションマニュアル（SBCA-470）』を参照してください。

操作方法

操作権限の認証機能は、Sysmac Studio の［コントローラ］-［セキュリティ］-［操作権限の設定］を選択して実行します。
操作方法の詳細は、🕮『Sysmac Studio Version 1 オペレーションマニュアル（SBCA-470）』を参照してください。

仕様

● 操作権限の種類

Sysmac Studio の操作権限の種類は、権限の大きい順に以下の 5 種類が可能です。

日本語名	英語名	パスワード設定
管理者	Administrator	必要
設計者	Planning Engineer	任意*1［操作権限認証設定］ダイアログボックスのパスワード入力を省略時の操作権限で要否を設定します。
保全者	Maintenance	
操作者	Operator	
観察者	Observer	不要

*1.［操作権限認証設定］ダイアログボックスのパスワード入力を省略時の操作権限で要否を設定します。省略時の操作権限よりも上位の権限にはパスワードが必要です。同じまたは下位の権限にはパスワードが不要です。

● 対象となるオンライン操作機能の例

オンライン操作の種類ごとの、操作権限の可否の例は、以下のとおりです。

詳細は、🕮『Sysmac Studio Version 1 オペレーションマニュアル（SBCA-470）』を参照してください。

（○：操作可能、△：操作時に都度認証実施、×：操作不可能）

ステータスモニタ機能（例）	管理者	設計者	保全者	操作者	観察者
異常モニタ（トラブルシュート）	○	○	○	○	○

I/O モニタ機能（例）	管理者	設計者	保全者	操作者	観察者
I/O モニタ（読出）	○	○	○	○	×
I/O モニタ（書込）	○	○	○	△	×
セット／リセット	○	○	○	△	×
強制値リフレッシュ	○	○	○	×	×

コントローラ操作機能（例）	管理者	設計者	保全者	操作者	観察者
運転モード／プログラムモード	○	○	△	×	×
オンラインエディット	○	○	△	×	×
コントローラリセット	○	○	×	×	×
異常解除（トラブルシュート）	○	○	○	△	×
軸試運転、軸試運転再開	○	○	△	×	×
ユーザプログラム実行用 ID（コントローラ）	○	×	×	×	×
CPU ユニットの書込プロテクト	○	○	○	×	×

● パスワード仕様

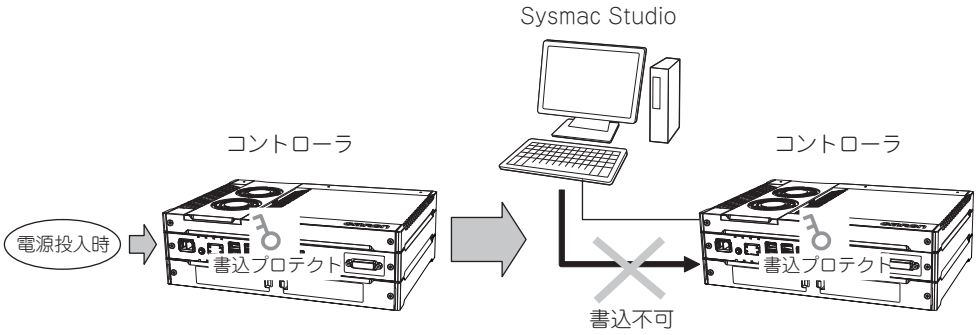
項目	内容
有効文字数	8 文字以上かつ 32 文字以下
使用可能文字	半角英数字（大文字、小文字は区別する）

8-3-6　CPU ユニットへの書込プロテクト機能

ユーザプログラム資産の保護または誤操作の防止のために、コントローラ内の各種データの書込をできないようにする機能です。以下の 2 つが可能です。

● コントローラの電源投入時の書込プロテクト設定

コントローラの電源投入時に、自動的に書込プロテクトを有効にする機能です。

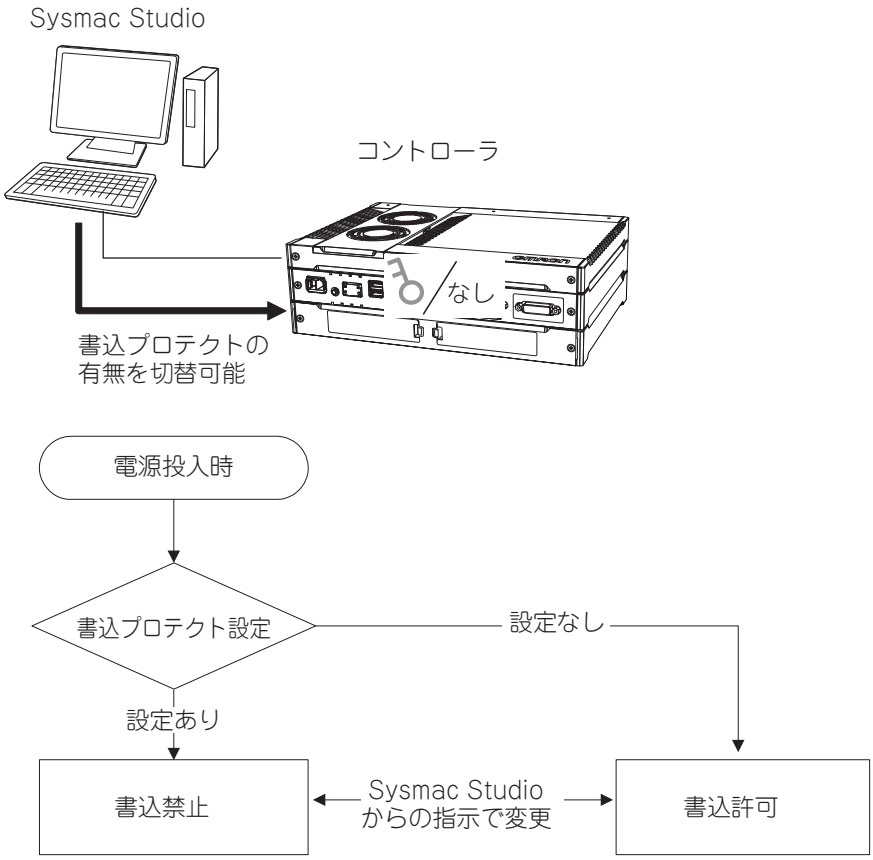


Sysmac Studio での操作の方法は、Sysmac Studio の [構成・設定]-[コントローラ設定]-[動作設定] にて、電源投入時に自動的に書込プロテクトを行うか否かを設定します。

設定 大項目	設定 中項目	設定 小項目	機能説明	設定値
動作設定ー 動作設定タブー 全般設定	セキュリティ設定	電源投入時の書込プロテクト	書込プロテクトを有効にする／しないを設定します。	プロテクトしない プロテクトする

● Sysmac Studio からの書込プロテクトの「セット」／「解除」の切替

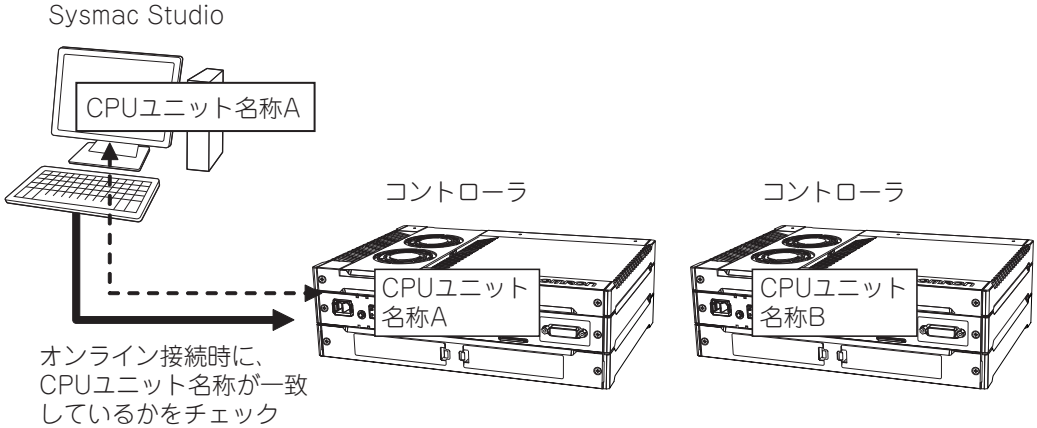
Sysmac Studio からオンライン接続して、[コントローラ] メニューから [セキュリティ] - [CPU ユニットへの書込プロテクト] にて、「セット」／「解除」を切り替えます。



8-3-7　CPU ユニット名称機能、およびシリアル ID 機能

概要

コントローラ内に CPU ユニット名称を登録してください。
Sysmac Studio からのオンライン接続時に、プロジェクト上の CPU ユニット名称と接続先コントローラの CPU ユニット名称が一致しているかどうかを確認できます。
これにより、Sysmac Studio から誤ったコントローラへの接続を防止することができます。特に、EtherNet/IP ネットワークを介した操作のときに有効です。



さらに、CPU ユニット名称以外に、コントローラの生産情報にもとづくシリアル ID を識別することもできます（オプション）。

設定方法

1 **Sysmac Studio でプロジェクト作成時に、CPU ユニット名称を設定します。**
CPU ユニット名称は以下に表示されます。



名前を変更する場合は、コントローラのアイコンを右クリックし、「名前の変更」にて編集します。

2 **コントローラと初めてオンライン接続する時に、Sysmac Studio はコントローラ内に CPU ユニット名称を記憶させるかどうかを促されます。**

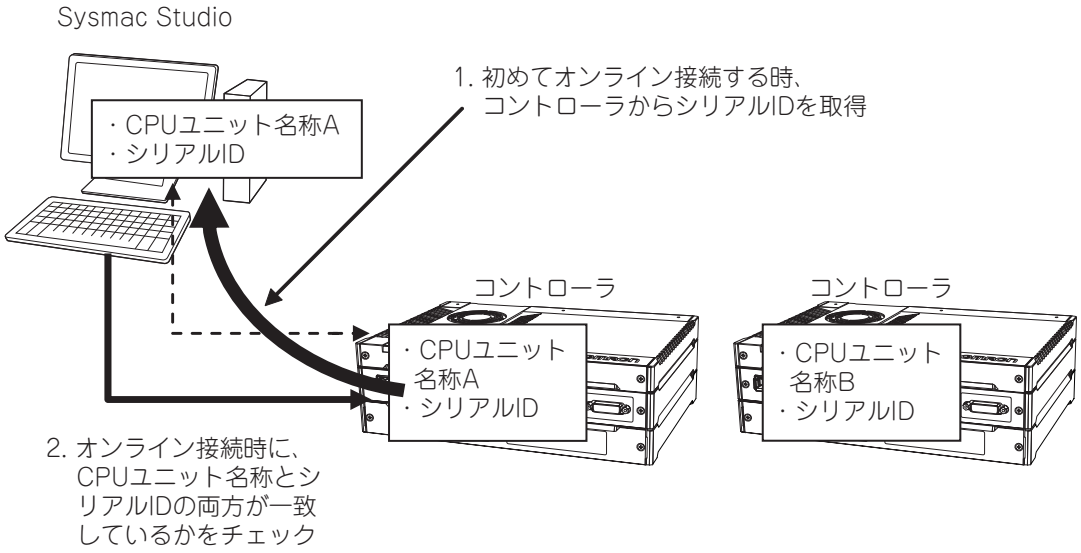
3 **その後、コントローラとのオンライン接続時に、Sysmac Studio は、プロジェクト上の CPU ユニット名称と、接続しようとするコントローラの CPU ユニット名称を照合します。**
不一致の場合は警告のダイアログが表示され、そのまま接続を継続するかどうかを促されます。

参考

NetworkConfigurator では、EtherNet/IP ポートに名称を付けることができます。

シリアル ID 機能

Sysmac Studio が初めてオンライン接続する時にコントローラの生産情報にもとづくシリアル ID を取得して、プロジェクトに記憶させることもできます。
その後、Sysmac Studio がオンライン接続する際、CPU ユニット名称とシリアル ID の両方の照合を実行します。これによって、より厳密なコントローラの照合が可能です。



8-4　デバッグ機能

ここでは、デバッグ機能について、説明します。

NY シリーズコントローラでは、以下のデバッグ機能があります。
- 強制値リフレッシュ
- 現在値変更
- オンラインエディット
- データトレース
- 微分モニタ

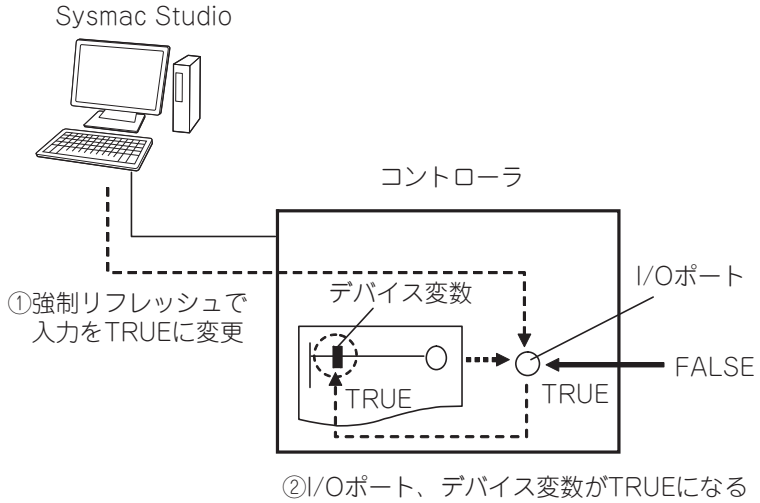
8-4-1　強制値リフレッシュ

概要

強制値リフレッシュとは、ユーザがデバッグ時などに外部からの入力および外部への出力を、Sysmac Studio で指定した強制値でリフレッシュすることができる機能です。
強制値リフレッシュは、指定したデバイス変数ではなく、デバイス変数に割り付けている I/O ポートに対して実行されます。
強制値リフレッシュで指定した状態は、Sysmac Studio から強制状態解除の指示をするまで、保持し続けます。(コントローラの状態変化に対する保持／解除については、後述の「強制値リフレッシュの保持／解除」を参照してください)。
また、異常停止時、メモリオールクリア時、モード変更時、電断時、ダウンロード時は保持されずクリアされます。

● 入力

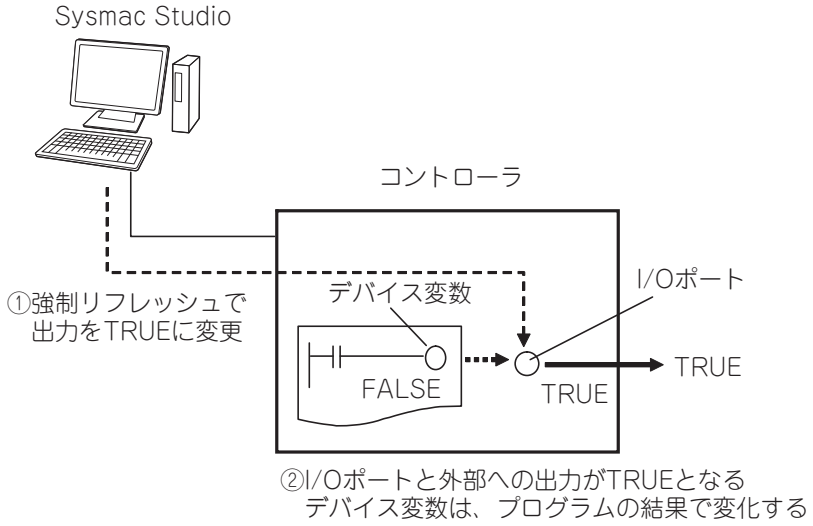
外部からの入力に関係なく、I/O ポートとデバイス変数が、強制値リフレッシュで指定した状態になります。



②I/Oポート、デバイス変数がTRUEになる

● 出力

I/O ポートと外部への出力が、強制値リフレッシュで指定した状態になります。
ユーザプログラム上にある、I/O ポートに割り付けられたデバイス変数の状態は、強制値リフレッシュで指定した状態ではなく、ユーザプログラムの結果で変化します。



対象エリア

強制値リフレッシュは、以下の I/O ポートに対して、実行できます。

- EtherCAT スレーブの I/O ポート

同時に強制値リフレッシュできる数

強制値リフレッシュできる個数は、以下のとおりです。

- EtherCAT スレーブ：合計 64 点

上記の個数は外部入出力接点の数です。たとえば、複数の変数が 1 つの外部入出力接点を割付先に指定している場合は 1 個と数えます。

用途

● 入力

- 入力信号を擬似的に与えて、ユーザプログラムのデバッグを行う
- 故障時にしかあり得ない状態を作り出す（排他的な 2 接点を同時に ON/OFF など）

● 出力

- 配線チェックのために出力を ON/OFF する
- ユーザプログラムの結果によらず、動作させたくない出力を強制的に OFF する

操作方法

以下の画面から操作することができます。
- プログラムの画面（ラダー図言語）
- I/O マップ
- ウォッチウィンドウ

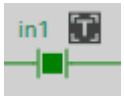
● ラダーエディタでの強制値リフレッシュの操作手順

1 **[コントローラ] の [モニタ] でモニタを ON します。**

2 **マルチビューエクスプローラの [プログラミング] で、対象のラダー図プログラムまたはラダー図ファンクションブロック / ファンクションをダブルクリックします。**
ラダーエディタ上に回路がモニタ状態で表示されます。

3 **強制値リフレッシュで On する接点、コイルを選択して、右クリックし、[強制値リフレッシュ] – [TRUE] を選択します。または、強制値リフレッシュで Off したい接点、コイルを選択して、右クリックし、[強制値リフレッシュ] – [FALSE] を選択します。**

4 **ラダーエディタ上の対象の接点、コイルが強制的に On または Off 状態になり、On の場合は導通状態、Off の場合は非導通状態になります。**
接点、コイルには、以下のような強制状態のマークが表示されます。



ラダー図

強制状態の「TRUE」「FALSE」表示は、強制値リフレッシュで指定した状態を表示しており、接点やコイルの現在値を表すものではありません。

強制状態のマーク	操作
T	強制値リフレッシュ（TRUE）を指定した場合
F	強制値リフレッシュ（FALSE）を指定した場合

参考

強制値リフレッシュした変数の AT（割付先）指定先と同じメモリアドレスを指定している変数があった場合、その変数にも強制状態のマークが表示されます。

動作モード、電源断との関連

● 強制値リフレッシュできる動作モード

プログラムモード、運転モードのいずれでも実行できます。
ただし、全停止フォールトレベルのコントローラ異常発生中の強制はできません。

● 動作モード変更時または電源断時の強制値リフレッシュ状態

運転モード⇔プログラムモード変更時または電源断時に、強制値リフレッシュ状態は解除されます。

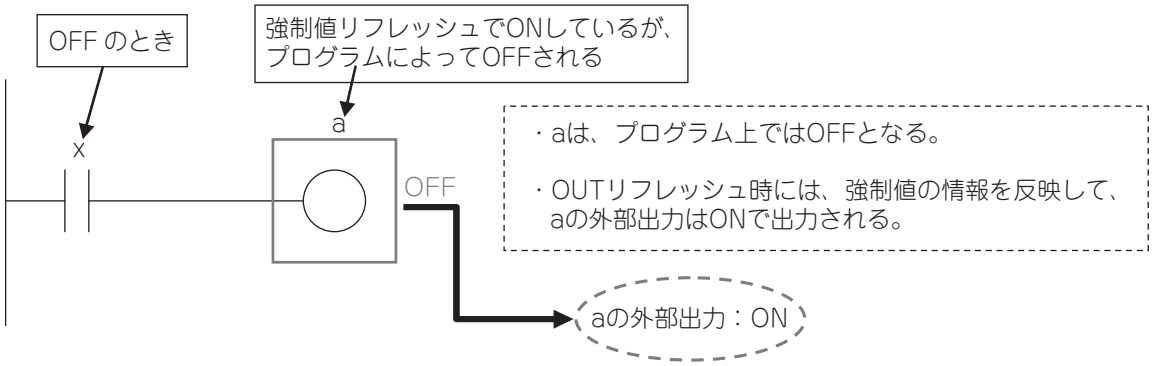
強制値リフレッシュの保持／解除

コントローラの状態変化に対して、強制値リフレッシュの保持／解除は以下のとおりです。

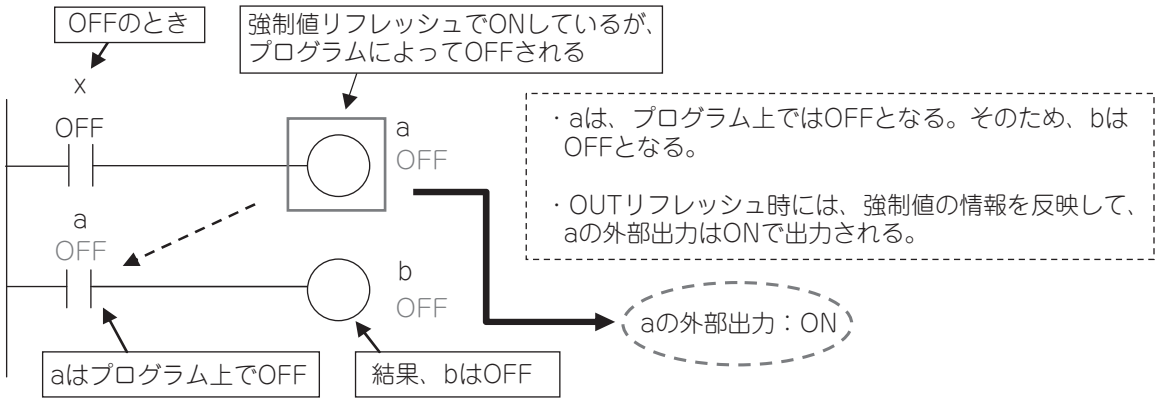
状態変化		強制値リフレッシュ状態
電源投入時		解除
動作モード変更時	運転モード→プログラムモード プログラムモード→運転モード	解除
ダウンロード後		解除
全停止フォールトレベルのコントローラ異常発生時		解除
オンラインエディット時		保持

強制値をプログラム上で使用する場合の留意事項

ユーザプログラム上で強制値リフレッシュを実行する場合は、強制値リフレッシュの対象にした変数はユーザプログラムによって上書きされます。そのため、強制値リフレッシュで指定した状態に保持されません。
ただし、外部へのリフレッシュは、ユーザプログラム上の変数の状態に関係なく、強制値リフレッシュで指定した値で出力され続けます。
強制値をプログラム上で使用するときは、プログラム上の変数と、外部出力の状態に違いがありますので注意してください。
例）a を強制値リフレッシュで ON の場合



さらに、その接点を受けたとき：



使用上の注意

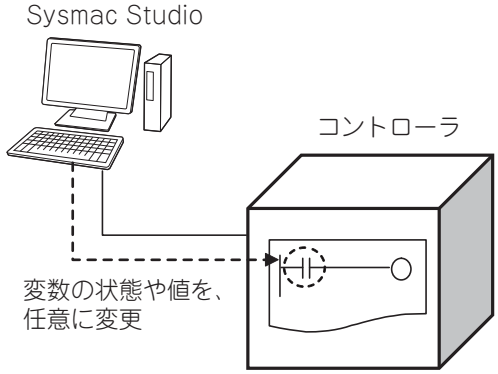
- 強制値リフレッシュの操作は、設備に影響がないかを確認したうえで行ってください。
- 強制値リフレッシュはユーザプログラム実行結果にかかわらず、設定された値で上書きして I/O リフレッシュする機能です。
  そのため、I/O リフレッシュの対象でない接点は本機能によっていったん設定された値になりますが、その後ユーザプログラムによって書き変わります。

強制状態の相違によって、制御対象が予期しない動作をするおそれがあります。

8-4-2　現在値変更

概要

現在値変更とは、プログラムや設定が正しく動作するかどうかを確認するために、Sysmac Studio から、プログラムや設定で使用している変数の値の変更や、接点やコイルを ON/OFF（TRUE/FALSE）することができる機能です。



使用上の注意

設備に影響がないことを確認してから、現在値変更してください。

用途

● 接点やコイルの ON/OFF（TRUE/FALSE）を任意に変更する

任意の BOOL 型変数の状態を ON/OFF します。この状態はユーザプログラムでの上書きで実行状態へ、モード変更および電源リスタートにより初期値の状態に戻ります。
ラダーエディタ、ウォッチウィンドウ、I/O マップで行うことができます

● 変数の値を任意に変更する

ユーザ定義変数やシステム変数、デバイス変数の現在値を任意の値に変更し、反映することです。
ウォッチウィンドウで行うことができます。

使用上の注意

設備に影響がないことを確認してから、現在値変更してください。

操作方法

現在値変更は、以下の画面から操作することができます。各画面での操作方法の詳細は、🕮『Sysmac Studio Version 1 オペレーションマニュアル（SBCA-470)』を参照してください。

- プログラムの画面（ラダー図言語、ST 言語）
- I/O マップ
- ウォッチウィンドウ

現在値変更により、外部への出力状態を変更する場合の留意事項

現在値変更により、EtherCAT 出力スレーブの I/O ポートに出力する状態を変更する場合には、以下について注意してください。

● 運転モード時に、I/O マップから現在値変更する場合

I/O マップから I/O ポートの値を変更しても、ユーザプログラムの実行結果で上書きされます。
よって、現在値変更で変更した値で、外部へは出力されません。
I/O ポートの値を任意の値に変更して、外部に出力したい場合は、強制値リフレッシュで行ってください。

● プログラムモード時に、ウォッチウィンドウから現在値変更する場合

ウォッチウィンドウから、デバイス変数（*）（外部変数、ローカル変数）の値を変更しても、外部へは出力されません。
外部に任意の値を出力したい場合は、以下のように行ってください。

- 強制リフレッシュする
- ウォッチウィンドウから、デバイス変数（*）（グローバル変数）に対して現在値変更する

* EtherCAT 出力スレーブの I/O ポートに付けたデバイス変数です。

8-4-3　オンラインエディット

ここでは、オンラインエディットの概要を説明します。詳細は、🕮『Sysmac Studio Version 1 オペレーションマニュアル（SBCA-470)』を参照してください。

概要

オンラインエディットとは、Sysmac Studio により、直接コントローラ内のユーザプログラムの一部を変更・追加する機能です。

以下を選択してオンラインエディットすることができます。
- POU（プログラム、FB、FUN）
  ラダー図言語で記述されたプログラムの場合はセクションを選択します。
- グローバル変数

用途

コントローラの運転を止めずにユーザプログラム変更をしたいときに使用します。

Sysmac Studio での操作

● オンラインエディットの操作

1 **オンラインエディット対象を選択します。**

2 **［プロジェクト］メニューから［オンラインエディット］–［開始］を選択します。**

3 **変更を行います。**

4 **［プロジェクト］メニューから［オンラインエディット］–［変更を転送］を選択します。**

5 **反映確認を行います。**

6 **オンラインエディット後のユーザプログラムで動作が開始されます。**

⚠ 注 意

入出力タイミングが乱れても影響のないことを確認してからオンラインエディットを実行してください。オンラインエディットを実行すると、タスク実行時間がタスク周期を上回り、外部との入出力が更新されず、入力信号を読み取れないことが発生したり、出力タイミングが乱れたりする場合があります。



使用上の注意

- オンラインエディットで編集した回路に含まれる微分命令の微分状態は、初期化されます。
- オンラインエディット内容を反映するときに、タスクの実行時間が延びます。タスク周期は、オンラインエディットによって「タスク周期オーバー」にならないように設定してください。
- オンラインエディット反映中（*）にコントローラが電源断した場合、電源断復帰時に全停止フォールトレベルのコントローラ異常（「ユーザプログラム／コントローラ構成・設定時転送エラー」、「ユーザプログラム / コントローラ構成・設定不正」、「不揮発性メモリ修復あり」のいずれか）が発生します。このとき、ユーザプログラムを再ダウンロードしてください。
- オンラインエディット反映中（*）に MC_SaveCamTable（カムテーブル保存）命令を実行しないでください。実行すると、オンラインエディットによる変更が正しく反映されない可能性があります。

＊　反映確認ウィンドウで［はい］を選択してから、「反映確認ウィンドウ」の後に表示される「内蔵不揮発性メモリバックアップ中ウィンドウ」が閉じるまでの間を指します。

8-4-4　データトレース

プログラムレスで指定した変数をサンプリングする機能です。Sysmac Studio で読み出して確認、およびファイル保存することができます。装置立ち上げ、運用／保守で使用できます。

以下にデータトレースの概要を説明します。操作方法は、🕮『Sysmac Studio Version 1 オペレーションマニュアル（SBCA-470)』を参照してください。

使用上の注意

以下の変数をデータトレースのサンプリング対象とする場合、正しいデータをサンプリングできないことがあります。

- モーション制御システム変数を除く、データサイズが 16 ビット以上の構造体のメンバ
- データサイズが 16 ビット以上の配列の要素

上記の変数で、データをサンプリングする場合は、以下のいずれかを行ってください。

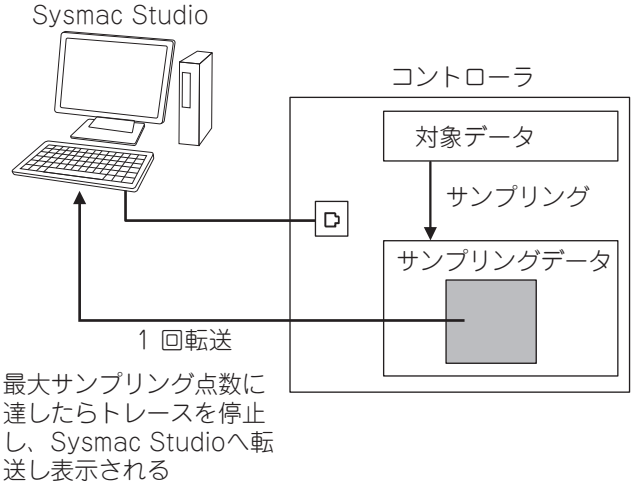
- 64 ビット以外の基本データ型の内部変数にコピーし、コピーした変数をデータトレース対象とする。
- 変数のタスク間排他制御設定機能を使用し、データトレースのサンプリング対象とするタスクを更新タスクとする。

トレース方法は、以下の 2 種類が可能です。

● トリガトレース

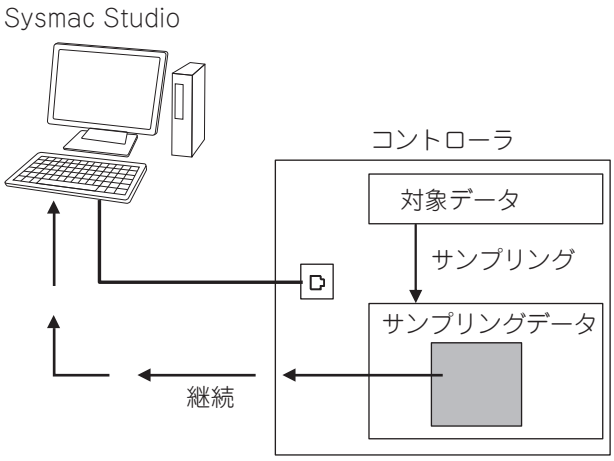
トリガ条件を設定することで、事象が発生した前後のデータを記録します。最大サンプリング点数に達すると自動的にサンプリングを終了します。Sysmac Studio と接続していないときも、トリガ条件成立時にデータトレースし、Sysmac Studio と接続したあとにアップロードすることができます。

- 変数の現在値の変更状況からプログラムの流れを確認できます。
- 意図しない変数の値の変更要因を調査するときに使用します。



● 連続トレース

トリガなしにサンプリングを開始し、10,000 点を超えてもサンプリングを続けます。サンプリングされたデータは順次パソコンに転送され、ファイルに保存されます。
表示バッファサイズになると自動的に CSV ファイルに保存します。
これにより、トレースを開始してから停止するまでの長期のトレース結果データを、複数の CSV ファイルとして格納することができます。



データトレースの仕様

データトレースの仕様は以下のとおりです。

項目		内容
データトレースの種類	**トリガトレース**	サンプリングを開始するトリガ条件を設定し、条件成立前後のデータを記録します。
	連続トレース	サンプリングされたデータは順次パソコンに転送され、ファイルに保存されます。
サンプリングタイミング	**指定タスクの周期**	タスクを指定し、そのタスクの周期をサンプリング周期とします。
	固定時間間隔	入力した時間をサンプリング周期とします。ただし、時間はプライマリ周期タスクの整数倍に丸められます。
	トレースサンプリング命令	ユーザプログラムで TraceSamp 命令が実行されたタイミングでサンプリングを実行します。
サンプリング対象設定*1	**最大対象数**	192 変数
	データ型	基本データ型（文字列を除く） 配列（要素を指定） 列挙型 構造体型／共用体型（メンバを指定）
レコード最大数		1 変数あたり 10,000 点
トリガ位置設定		全トレース時間（数）におけるトリガ位置を設定します。
トリガ設定	**条件のデータ型**	基本データ型（時刻、時間長、日付、文字列は除く） 配列（要素を指定）、構造体／共用体（メンバを指定）
	条件式*2	以下の条件式の成立時にトレースが開始されます。 BOOL：立ち上がり、立ち下がり BOOL 以外：＝、＞、≧、＜、≦、≠*3
	Sysmac Studio からの指示	トリガ ON ボタンがクリックされたときにトレースが開始されます。
	データトレーストリガ条件成立命令	TraceTrig 命令を実行したときにトレースが開始されします。
	判定タイミング	• サンプリングタイミング設定がトレースサンプリング命令以外のときは、指定したタスク周期にトリガ判定が 1 回実行されます。 • サンプリングタイミング設定がトレースサンプリング命令のときは、命令実行時にトリガ判定が実行されます。
	ディレー値	トリガ成立前／ 成立後のサンプリング数割合をスライダーで設定します。（例：20%/80%）
トレース開始	**Sysmac Studio からの指示**	Sysmac Studio の「実行ボタン」を押すと開始されます。
	運転時開始設定	運転開始（プログラムモード→運転モード）時に自動でトレースが開始されます。

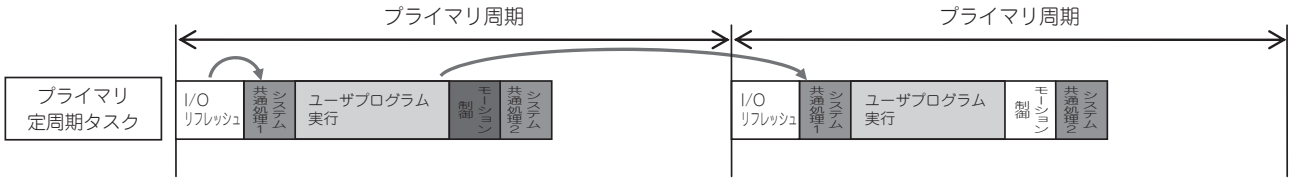
項目		内容
トレース停止	トリガトレース	• サンプリング上限値の 10,000 点に達すると停止されます。 • Sysmac Studio の「停止ボタン」を押すと停止されます。
	連続トレース	• 上限到達時の設定が「トレース停止」のときは、上限サイズもしくは上限期間に達すると停止されます。 • Sysmac Studio の「停止ボタン」を押すと停止されます。
連続トレース専用設定	データ保存期間上限	連続トレースデータの保存期間上限を設定します。
	データ保存サイズ上限	連続トレースで保存されるファイルの合計サイズの上限を設定します。
	1 ファイルあたりのデータ数	連続トレース時に、1 つのファイルに保存するデータ数を設定します。
	ファイル保存先	連続トレース時に、データを保存するファイルを作成する場所を指定します。
	ファイル名プリフィックス	自動的につけられるファイル名の先頭部の名前を指定します。
	上限到達時の設定	保存期間上限もしくは保存サイズ上限に達したときの動作を指定します。 • トレース停止 • 古いファイルから削除して続行
トレース結果表示	グラフ表示	X 軸：時間、Y 軸：変数の値となるグラフを表示します。BOOL 型とそれ以外の変数をひとつのグラフに表示することもできます。
	テーブル表示	各変数の最大値、最小値、平均値および指定された時間における値をテーブル形式で表示します。
	3D モーションモニタ表示	3D 空間に仮想機構モデルを設置し、モーション軸の指令位置および現在位置に合わせて機構の動きを仮想表示します。
トレースデータエクスポート	CSV ファイルエクスポート	トレース結果およびトレース番号以外の設定を CSV 形式ファイルに保存します。
同時に起動できるデータトレース*4		4 個
トレースデータインポート		CSV 形式で保存されたトレース結果を、現在表示されているグラフに重ね合わせて表示します。
保存		トレース結果をトレース設定とともにプロジェクト内に保存します。
印刷		グラフを印刷します。Sysmac Studio の印刷機能が使用されます。

*1. システム定義変数の EN、ENO、P_off、P_on、P_CY、P_First_RunMode、P_First_Run、P_PRGER、および、ファンクションブロックのインスタンスで使用される入出力変数、ファンクション内の変数はサンプリング対象外です。
*2. データトレース開始時点で、トリガの条件式が成立状態であっても、データトレースは開始しません。
*3. 複数の条件式の組み合わせはできません。また、比較定数の有効範囲は、条件式左辺の変数の型のリテラル表現の有効範囲にしたがいます。
*4. NY シリーズでは、0 ～ 3 のトレース番号が設定されます。この番号を使って、命令の実行やシステム定義変数へのアクセスを行います。

データトレースの動作

データトレース処理（サンプリングおよびトリガ判定）は、I/O リフレッシュ処理の後、ユーザプログラム実行前の、「システム共通処理 1」で行われます。

例）サンプリングタイミングに「プライマリ定周期タスク」を指定した場合は、下図の「システム共通処理 1」のタイミングで、データトレースが実行されます。



参考

タスク処理では、「I/O リフレッシュ、ユーザプログラム、モーション制御」は同一タスク周期ですが、データトレースの表示では、「ユーザプログラム、モーション制御、次のタスク周期の I/O リフレッシュ」が同一タスク周期として表示されます。なお、📖『NY シリーズ コマンドリファレンスマニュアル モーション編（SBCE-380）』に記載されているタイミングチャートは、タスク周期を基準に記載していますので、データトレースの表示とは異なります。

以下に、指定タスク周期でサンプリングした場合の、データトレースの動作および実行結果の表示例を示します。

例 1）

センサの入力信号に割り付けた変数「Sensor1」が TRUE になったとき、変数「SysRun」を TRUE にするプログラムを作成した場合



データトレースの動作および実行結果の表示は、以下のとおりとなります。

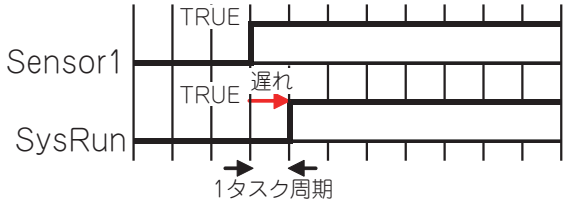
(1) システム共通処理 1 のデータトレース処理で「Sensor1」の TRUE を取得する

(2) ユーザプログラム中で、「SysRun」を TRUE にする

(3) 次のプライマリ周期のシステム共通処理1のデータトレース処理で「SysRun」のTRUEを取得する

したがって、データトレースでの表示は、「Sensor1」より 1 タスク周期遅れて「SysRun」が TRUE となります。

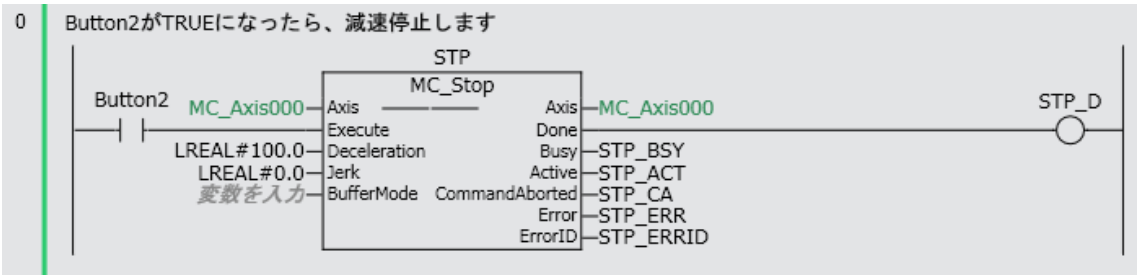
データトレースの表示



参考

ユーザプログラム中で変数の値を変更した場合、データトレースの表示では、値の変化と I/O リフレッシュによる出力処理が同じタスク周期で変化します。

例 2）
速度制御で動作中に、押しボタンの入力信号に割り付けた変数「Button2」が TRUE になったら軸 0（「MC_Axis000」）を減速停止させるプログラムを作成した場合



データトレースの動作および実行結果の表示は、以下のとおりとなります。

(1) システム共通処理 1 のデータトレース処理で「Button2」の TRUE を取得する

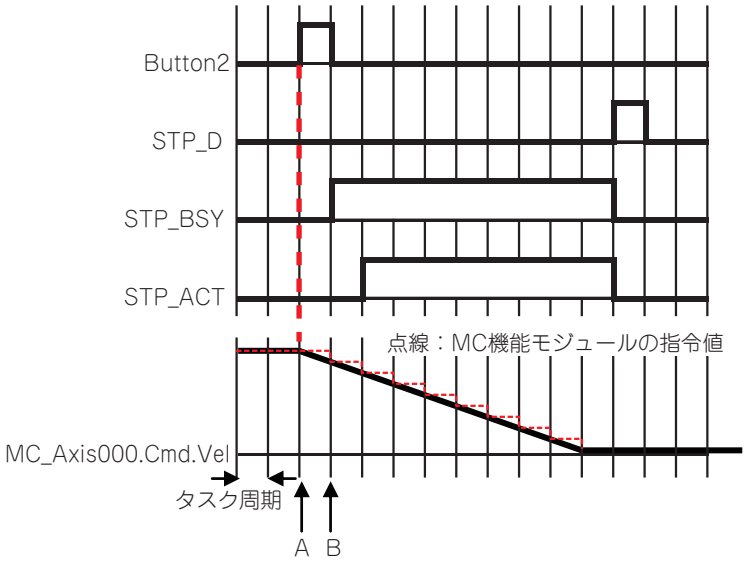
(2) ユーザプログラム中で「STP_BSY」を TRUE にし、モーション制御機能モジュールが減速処理を行う

(3) 次のプライマリ周期のシステム共通処理 1 のデータトレース処理で STP_BSY の TRUE と、モーション変数の取得を行う

(4) ユーザプログラム中で、「STP_ACT」を TRUE にする

(5) 次のプライマリ周期のシステム共通処理1のデータトレース処理でSTP_ACTのTRUEを取得する

モーション制御機能モジュールの指令値は、ユーザプログラム中の「STP_BSY」が TRUE になり、モーション制御機能モジュールで減速処理が開始された時点から変化を開始します（下図のBの時点）。指令値は、プライマリ定周期タスクに同期して階段状に変化します。
これに対して、データトレースは、前周期の値と次周期の値を直線で補間します。そのため、モーション変数の指令速度「MC_Axis000.Cmd.Vel」は、指令値が変化する 1 周期前、つまり「Button2」が TRUE となるタイミング（下図の A の時点）から、減速するように表示されます。減速から 1 周期遅れて「STP_BSY」が TRUE、さらに 1 周期遅れて「STP_ACT」が TRUE になるように表示されます。



参考

モーション制御命令を含むファンクションブロックでは、ファンクションブロック実行開始時に入力パラメータの値が入力変数にセットされ、ファンクションブロック実行終了時に出力変数の値が出力パラメータにセットされます（「ファンクションブロック（FB）内の変数の仕様」（P.6-12）参照）。データトレースの表示も、入力パラメータと入力変数、出力パラメータと出力変数が同じタスク周期で変化します。

関連システム定義変数

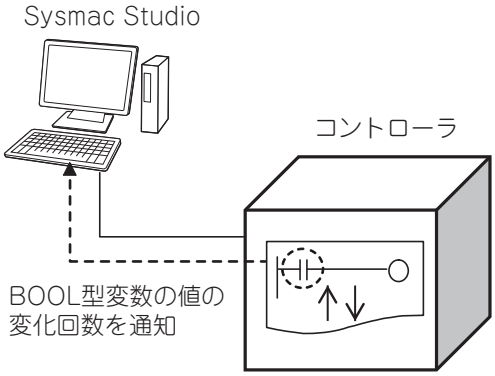
変数名	メンバ名	名称	機能	データ型	R/W
_PLC_TraceSta[0..3]*1				_sTRACE_STA	R
	.IsStart	トレース実行中 ON フラグ	トレースを開始したとき、TRUE となります。	BOOL	R
	.IsComplete	トレース完了時 ON フラグ	トレース実行が完了したとき、TRUE となります。 次にトレースを開始したとき、FALSE となります。	BOOL	R
	.IsTrigger	トレーストリガモニタフラグ	トレーストリガ条件が成立したとき、TRUE となります。 次にトレースを開始したとき、FALSE となります。	BOOL	R
	.ParamErr	トレースパラメータ異常フラグ	トレース開始時にトレース設定が異常のとき、TRUE となります。 正常のとき、FALSE となります。	BOOL	R

*1. データトレース 0 ～ 3 に対応します。

（注） 本システム定義変数は、ユーザプログラム上では使用できません。ユーザプログラムからデータトレースの状態を参照する場合は、GetTraceStatus 命令を使用してください。

8-4-5　微分モニタ

微分モニタとは、指定した BOOL 型変数の値が、指定した条件に一致した回数を通知する機能です。プライマリ定周期タスクのタスク周期 (プライマリ周期と呼びます) ごとに条件一致判定を実行し、それまでの条件一致回数を通知します。



用途

外部からの入力信号や、ユーザプログラム内の接点の ON/OFF の有無や回数が確認できます。設備立ち上げ時や稼働中の動作不良の原因調査などに役立ちます。

微分モニタの仕様

微分モニタの仕様を、以下に示します。

項目		仕様
微分モニタ条件	設定数	最大 8
	指定変数	• BOOL 型 • BOOL 型配列の 1 要素 • 構造体、共用体の BOOL 型の 1 メンバ
	条件式	• 立ち上がり (FALSE → TRUE) • 立ち下がり (TRUE → FALSE)
条件一致判定	タイミング	プライマリ周期に 1 回
	回数カウント	指定変数ごとに条件一致回数をカウント
開始、停止	開始条件	Sysmac Studio からの指示
	停止条件	• Sysmac Studio からの指示 • 全停止フォールトレベルのコントローラ異常の発生 • ユーザプログラムのダウンロード • メモリオールクリア • Sysmac Studio のオンライン接続解除
動作モード		• 運転モード • プログラムモード

微分モニタ条件

微分モニタの設定対象となる変数や変化種類のことを、微分モニタ条件と呼びます。それらの仕様を説明します。

● 設定数

最大 8 です。つまり、8 個の変数の値の条件一致回数を、並行して検出することができます。

● 指定変数

微分モニタの対象に指定できる変数のデータ型は、以下のとおりです。

- BOOL 型
- BOOL 型配列の 1 要素
- 構造体、共用体の BOOL 型の 1 メンバ

配列全体、構造体全体、共用体全体は、指定できません。

また、変数の種類としては、以下のものが指定可能です。

変数の種類		指定可否
システム定義変数		可*1
準ユーザ定義変数		可
ユーザ定義変数	グローバル変数	可
	プログラム内で使用する変数	可
	ファンクションブロック内で使用する変数	可*2
	ファンクション内で使用する変数	不可

*1. 以下の変数は除きます。
EN,ENO,P_Off,P_CY,P_First_RunMode,P_First_Run,P_PRGER
*2. 入出力変数は除きます。

● 条件式

検出する変数の変化条件を、条件式と呼びます。条件式は、以下の 2 種類があり、どちらかを選択します。条件式は、変数ごとに指定できます。

- 立ち上がり (FALSE → TRUE)
- 立ち下がり (TRUE → FALSE)

使用上の注意

たとえば、条件式を「立ち上がり」に設定したとします。微分モニタ機能を開始した時点で指定した変数の値が TRUE であっても、立ち上がりとは判定されません。変数の値がいったん FALSE になってから TRUE になった際に、立ち上がりと判定されます。

条件一致判定タイミングと回数カウント

指定した変数の条件一致判定は、プライマリ周期ごとに行います。前回判定時の変数の値と、今回判定時の変数の値を比較します。その結果、指定した条件式に一致する変数の値の変化があれば、その回数をカウントします。

- 条件一致回数カウントは、変数ごとに行います。
- カウント値は、微分モニタ開始時に 0 にリセットされます。
- 変数ごとにカウント値を 0 にリセットすることはできません。

使用上の注意

- プライマリ周期内に条件式に一致する値の変化が複数回あっても、1 回のプライマリ周期に 1 ずつしかカウントされません。
- 前回判定時と今回判定時の変数の値が同じであれば、その間に値の変化があったとしても条件式に一致したとは見なされません。

開始条件と停止条件

微分モニタを開始するには、Sysmac Studio から指示します。
微分モニタを停止するには、通常、Sysmac Studio から指示します。しかし、以下に示す場合も、微分モニタは停止します。

- 全停止フォールトレベルのコントローラ異常発生
- ユーザプログラムのダウンロード
- メモリオールクリア
- Sysmac Studio のオンライン接続解除

操作方法

微分モニタの操作方法は、以下のとおりです。詳細は、🕮『Sysmac Studio Version 1 オペレーションマニュアル（SBCA-470）』を参照してください。

1 **Sysmac Studio の [表示]-[微分モニタ] を選択します。**

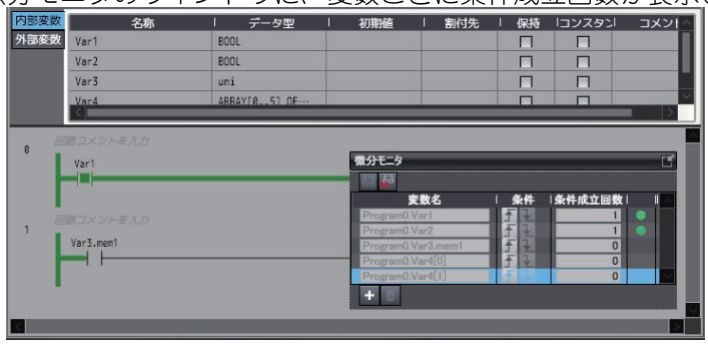
2 **微分モニタの対象に指定できる変数を選択して右クリックし、[微分モニタに追加] を選択します。**

3 **微分モニタのウィンドウで、変数ごとに微分モニタの条件式を設定します。**



4 **ユーザプログラムを実行します。**

微分モニタのウィンドウに、変数ごとに条件成立回数が表示されます。



微分モニタ機能の留意事項

微分モニタ機能を使用する際には、以下のことに留意してください。

● 微分モニタ実行中の Sysmac Studio との通信断

微分モニタ実行中に、通信ケーブルの断線、Sysmac Studio の異常終了などの原因により、Sysmac Studio との通信が途絶えたとします。その場合、コントローラは微分モニタ実行を継続します。あらためて微分モニタを実行する場合は、Sysmac Studio との通信を再開した後、いったん微分モニタを停止させてください。

● 微分モニタの複数実行不可

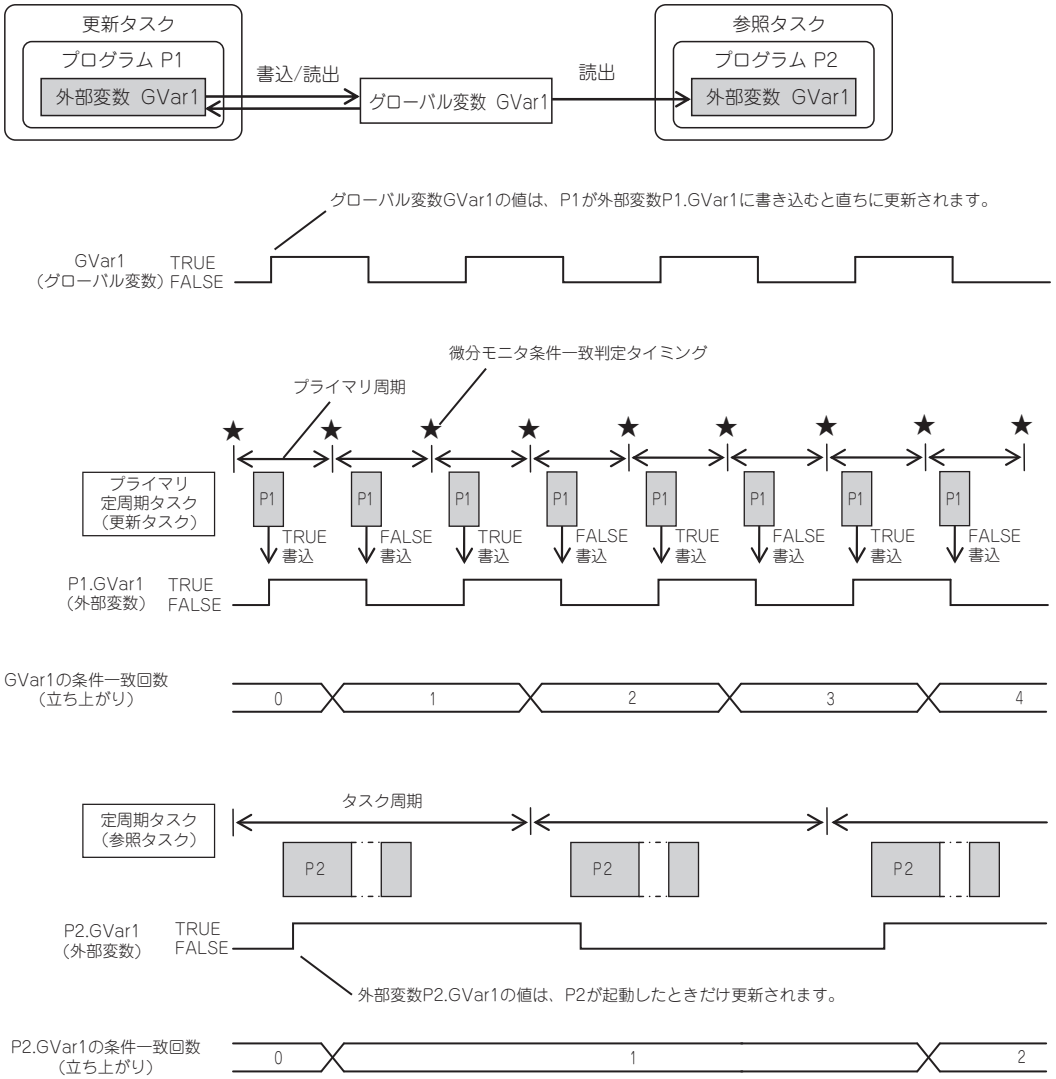
微分モニタは、複数の Sysmac Studio から同時に実行することはできません。

● グローバル変数、外部変数の指定

グローバル変数も、POU 内でそれを指定している外部変数も、微分モニタの指定変数にすることが可能です。ただし、グローバル変数と外部変数とでは、値の更新タイミングが異なることに注意してください。
グローバル変数は、値を書き込まれると直ちに更新されます。一方、外部変数は、その外部変数を宣言している POU が起動するタイミングでしか更新されません。
以下に、図で説明します。つぎの 2 つの変数をモニタするとします。

変数名	変数の種類	読み書きする POU
GVar1	グローバル変数	プライマリ定周期タスクに割り付けられたプログラム P1
P2.GVar1	プログラム P2 で、GVar1 を指定して宣言された外部変数	定周期タスクに割り付けられたプログラム P2

グローバル変数 GVar1 は、プライマリ定周期タスクに割り付けられたプログラム P1 で読み書きされるので、毎周期書込が発生するならばプライマリ周期で更新されることになります。一方、外部変数 P2.GVar1 は、定周期タスクに割り付けられたプログラム P2 が起動するタイミングでしか更新されません。つまり定周期タスクのタスク周期でしか更新されないことになります。定周期タスクのタスク周期は、プライマリ周期よりも長いため、P2.GVar1 は GVar1 よりも更新回数が少なくなります。

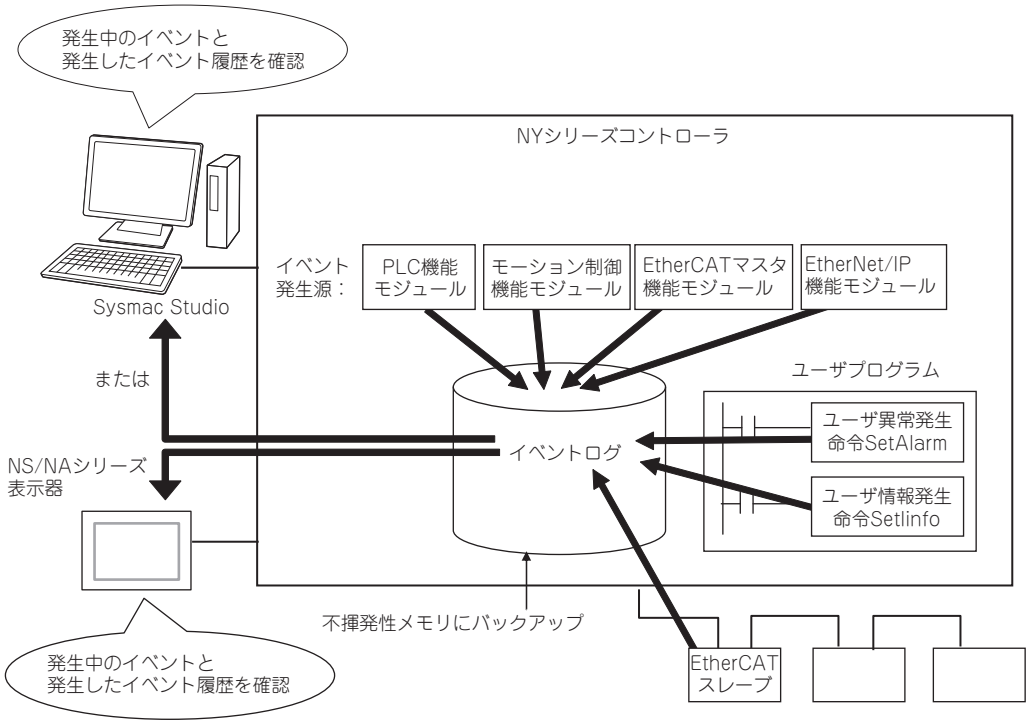


8-5　イベントログ機能

ここでは、イベントログ機能について、説明します。

8-5-1　概要

イベントログ機能とは、NY シリーズコントローラで発生した異常、状態変化、およびユーザ定義事象といった、イベント（*）を記録する機能です。



*「イベント」とは、コントローラで、非計画的に発生する、異常を含めた事象です。異常と、コントローラやユーザ定義によりユーザへ通知する必要のある異常ではない情報を合わせた総称です。
以下の 2 種 4 分類があります。

- コントローライベント
  コントローラ異常
  コントローラ情報
- ユーザイベント
  ユーザ異常
  ユーザ情報

NS/NA シリーズ表示器でイベントを確認する場合は、NS/NA シリーズ表示器をコントローラの内蔵 EtherNet/IP ポートで接続してください。

使用上の注意

表示器のトラブルシュータの対応範囲については、📖『NY シリーズ トラブルシューティングマニュアル（SBCA-438）』を参照してください。

特長

イベントログ機能には、以下の特長があります。

- 異常履歴に加え、電源投入／電源断や運転開始などの各種履歴も記録します。
- これら記録が時系列に確認できるため、トラブル発生時に異常要因の推測に役立てることができます。

記録できること

イベントには、以下の種類があります。

● システム定義のイベント（コントローライベント）

コントローラが自動で検知するイベントです。NY シリーズコントローラの各機能モジュール、EtherCAT スレーブのイベントです。以下のイベントがあります。

- コントローラ異常
- コントローラ情報

● ユーザが定義したイベント（ユーザイベント）

ユーザが設計するアプリケーション上のイベントです。命令実行により、以下のイベントが可能です。

- ユーザ異常
- ユーザ情報

Sysmac Studio または表示器によって、イベントログを読み出すことができます。

8-5-2　イベントログ機能の詳細

イベント発生源

コントローラのどこでイベントが発生しているかを識別する情報です。
コントローライベントとユーザイベントそれぞれの発生源は、以下のとおりです。

● コントローライベント発生源

コントローライベントでは、コントローラ内に存在する機能モジュールが発生源となります。
さらに機能モジュールによっては、より詳細なイベント発生源を持ちます。これを「イベント発生源詳細」と呼びます。
以下があります。

イベント発生源	イベント発生源詳細
PLC 機能モジュール	命令、Windows
モーション制御機能モジュール	共通、軸、軸グループ
EtherCAT マスタ機能モジュール	通信ポート、EtherCAT マスタ、EtherCAT カプラユニット、NX ユニット、EtherCAT スレーブ
EtherNet/IP 機能モジュール	通信ポート／通信ポート 1 ／内部通信ポート 1、CIP、FTP、NTP、SNMP

● ユーザイベント発生源

ユーザイベントでは、イベント発生源が PLC 機能モジュールとなります。

カテゴリ

イベントログの分類を表す情報です。
Sysmac Studio または表示器で参照するときの単位です。

イベントの種類	イベントログのカテゴリ	内容
コントローライベント	システムログ	コントローラ自身が検知したイベントの記録です。
	アクセスログ	ユーザアクションによりコントローラの動作へ影響を与えたイベントの記録です。
ユーザイベント	ユーザイベントログ	ユーザが定義したイベントの記録です。

記録件数

イベントログには、それぞれ以下のレコード件数分を記録できます。
コントローラは、以下の記録可能件数を超えてイベントが発生した場合、最も古い情報から上書きします。

イベントの種類	イベントログのカテゴリ	1 カテゴリあたりの記録可能件数
コントローライベント	システムログ	2,048
	アクセスログ	
ユーザイベント	ユーザイベントログ	

電断保持

NY シリーズコントローラは、電断が発生しても不揮発性メモリによってイベントログを保持します。

使用上の注意

イベントログは、不揮発性メモリによって保持されます。産業用 PC に UPS が接続されていない場合、あるいは正常にシャットダウンされなかった場合は、保持することはできませんのでご注意ください。
必要により、定期的にイベントログのエクスポートを実施してください。

イベントコード

イベントの種類に応じてあらかじめシステムがコントローライベントに割り付けているコード、またはユーザがユーザイベントに割り付けているコードです。コントローライベントのイベントコードは、16 進 8 桁で示されます。
発生中の異常のイベントコードは、異常ステータス取得命令により取得できます。
ユーザイベントには、10 進数で 1 ～ 60000 のイベントコードを付けることができます。

イベント重要度

各イベントには、重要度を表す「イベント重要度」があります。
イベント重要度の分類は、イベントの種類によって異なります。コントローライベントとユーザイベントで独立して定義されます。

● コントローライベントの場合

コントローライベントは、制御に対する影響の大きさにより重要度を以下の5段階に分類しています。

No.	重要度		分類
1	高	コントローラ異常	全停止フォールトレベル
2	↕		部分停止フォールトレベル
3			軽度フォールトレベル
4			監視情報レベル
5	低	コントローラ情報	一般情報レベル

異常の重要度が高いほど、コントローラが提供する機能が損なわれており、復旧に対する緊急度も高くなります。
この分類は、異常発生時に Sysmac Studio や表示器に表示されます。

● ユーザイベントの場合

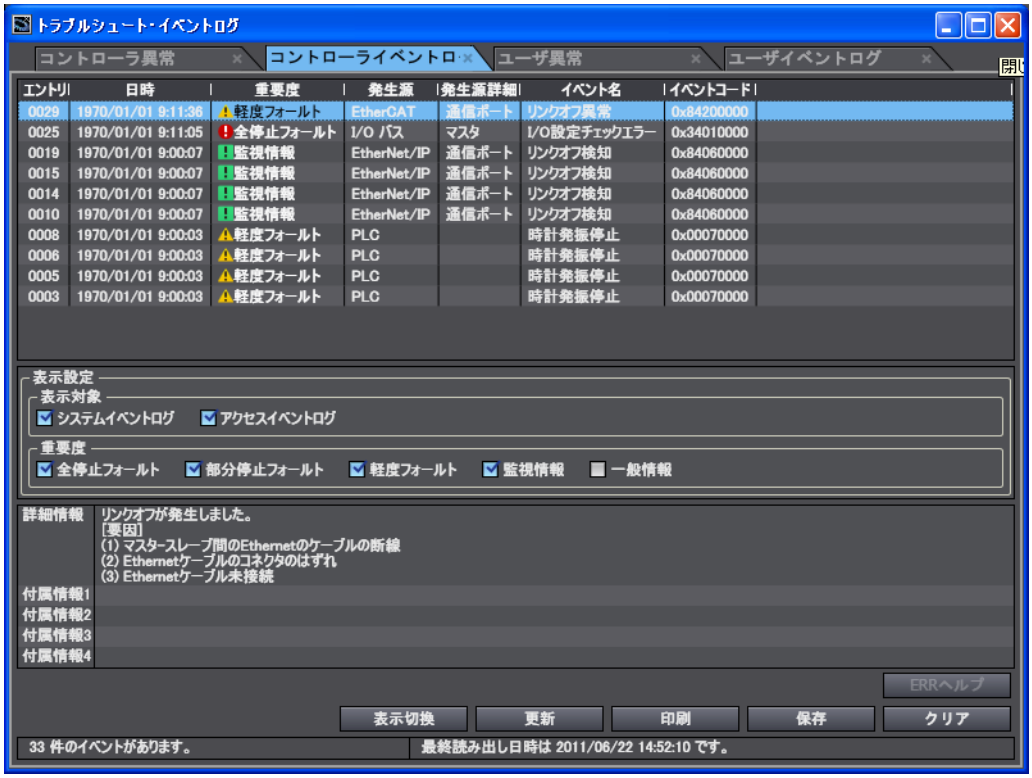
ユーザイベントを、NY シリーズシステムが提供する以下の重要度によって分類できます。
ユーザイベントをどの重要度で発生させるかを定義します。

No.	重要度	分類	意味
1	高	User fault Level1	アプリケーション上のユーザ定義の異常を表すイベント。ユーザは、SetAlarm（ユーザ異常発生）命令を実行してイベントを発生させます。
2	↕	User fault Level2	
3		User fault Level3	
4		User fault Level4	
5		User fault Level5	
6		User fault Level6	
7		User fault Level7	
8		User fault Level8	
9	低	User Information	アプリケーション上のユーザ定義の情報を表すイベント。ユーザは、SetInfo（ユーザ情報発生）命令を実行してイベントを発生させます。

イベントログの表示

Sysmac Studio または表示器は、「コントローラログ」「ユーザログ」のカテゴリ別にイベントログの一覧を表示します。「コントローラログ」には、「アクセスログ」と「システムログ」が表示されます。また、Sysmac Studio は、EtherCAT スレーブ内に記録された異常履歴を表示することもできます。

Sysmac Studio では以下のように一覧が表示されます。一覧からイベントを選択すると、その詳細情報が表示されます。



参考

Sysmac Studio または表示器のバージョンで、未対応のイベントがコントローラで発生した場合は、発生源を「不明」、イベント名を「不明なイベント」として表示されます。イベントコードと付属情報はそのまま表示されます。

イベントログの消去

● Sysmac Studio ／表示器からの消去

Sysmac Studio または表示器からイベントログのコントローライベントとユーザイベント別にイベントログを消去できます。

使用上の注意

- Sysmac Studio ／表示器からコントローラ内のイベントログを消去する場合、これまで取得したすべてのイベントの情報が消去されてもかまわないことを確認した上で、実行してください。特に、一般情報レベル／監視情報レベルのコントローライベントやユーザイベントの場合、一度も確認していない可能性がありますので、ご注意ください。
- 表示器からイベントログを消去する場合の制約事項については、🕮『NY シリーズ　トラブルシューティングマニュアル（SBCA-438)』を参照してください。

● メモリオールクリアでの消去

Sysmac Studio から NY シリーズコントローラに対してメモリオールクリアの操作を行った場合、イベントログを消去するかどうかを選択できます。

イベントログのエクスポート

Sysmac Studio または表示器を使って表示したイベントログの内容を、CSV ファイルとしてエクスポートすることができます。

8-5-3　コントローライベント（コントローラ異常／情報）

概要

NY シリーズのシステムで定義した異常／情報です。
NY シリーズシステムが異常要因、情報要因を検知すると、発生します。

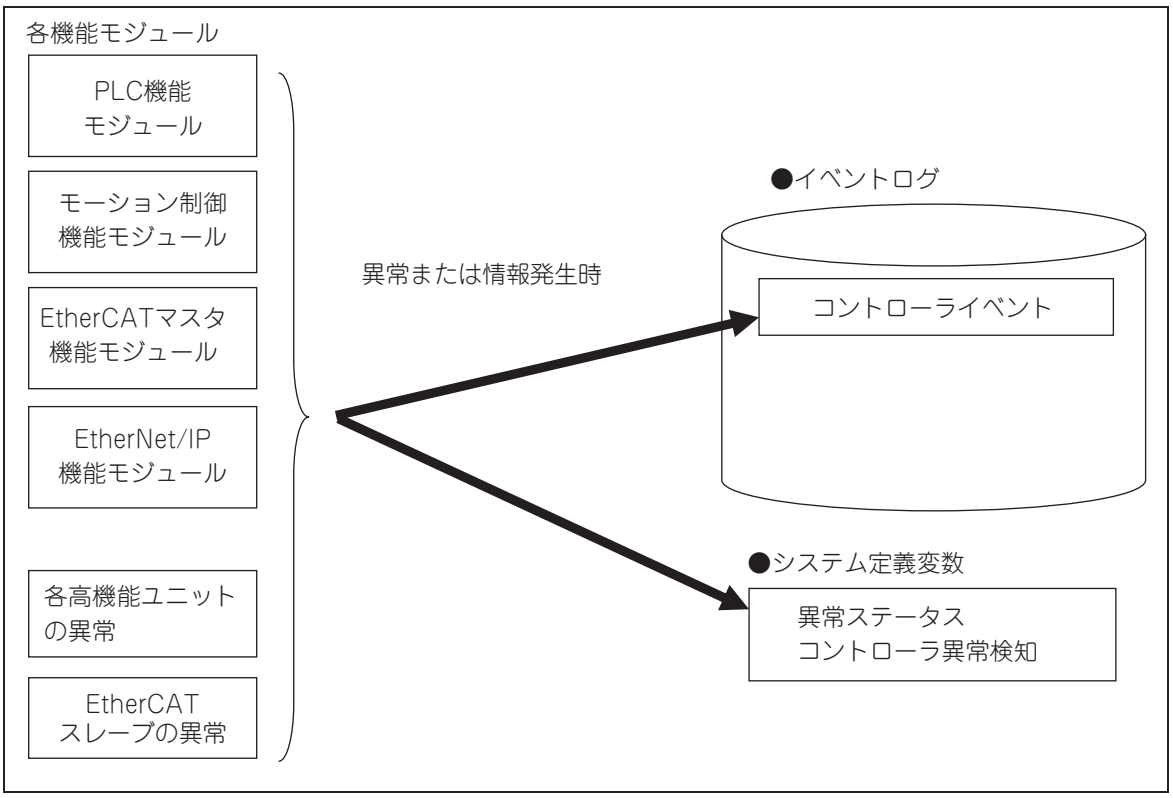
● コントローラ異常

システム定義の異常です。
「全停止フォールトレベル」、「部分停止フォールトレベル」、「軽度フォールトレベル」、「監視情報レベル」のコントローライベントの総称です。
NY シリーズコントローラの各機能モジュール、EtherCAT スレーブの各異常を検知します。このとき、コントローラ異常がイベントログへ記録されます。
発生したコントローラ異常の状況をユーザプログラム上で確認する場合は、異常ステータス取得命令によりシステム定義変数の「異常ステータス変数」を参照します。

コントローラ異常は、動作モードの変更により、解除されません。
コントローラ異常の詳細については、🕮『NY シリーズ トラブルシューティングマニュアル（SBCA-438）』を参照してください。

● コントローラ情報

異常ではないシステム定義の通知情報です。「一般情報レベル」のコントローライベントです。
たとえば、電源 ON/OFF、運転開始／停止、Sysmac Studio のオンライン接続、ユーザプログラムのダウンロードなどの、異常以外の事象です。



8-5-4　ユーザイベント（ユーザ異常／情報）

概要

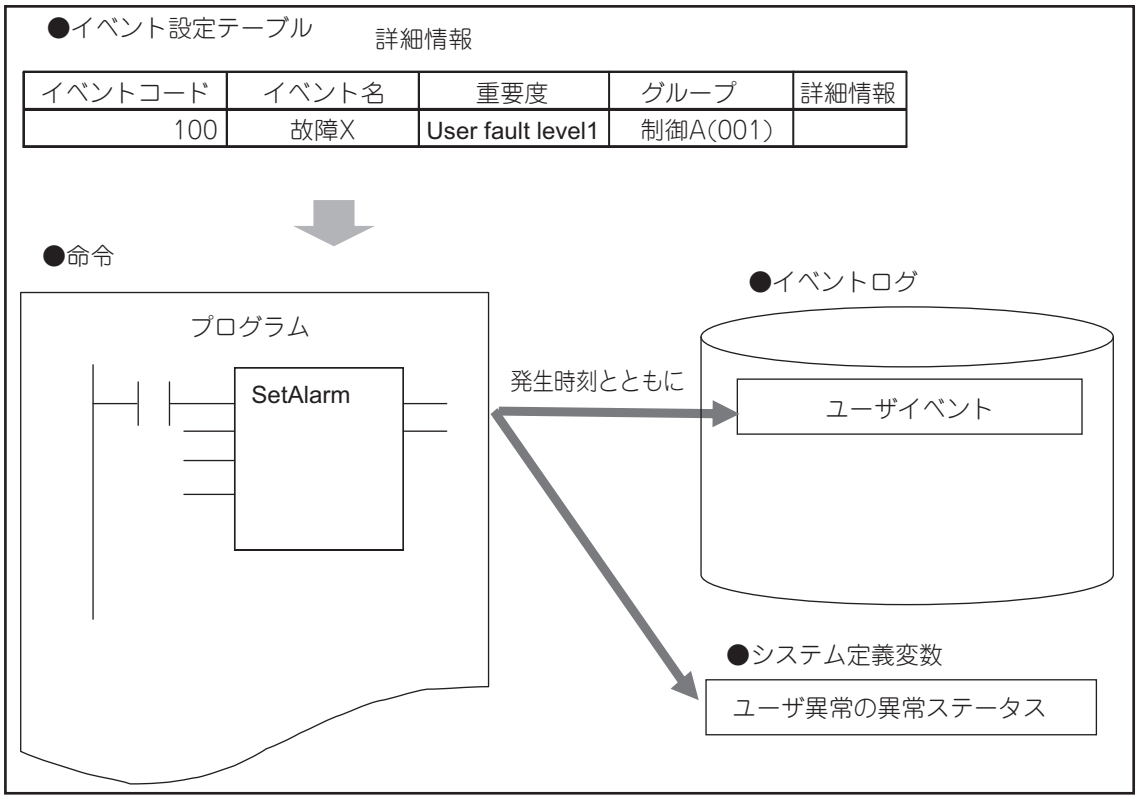
ユーザが、定義した異常／情報です。命令によって発生させることができます。

● ユーザ異常

ユーザ定義の異常です。
ユーザ異常発生（SetAlarm）命令によって、ユーザ異常を発生させます。このとき、ユーザ異常がイベントログへ記録されます。
対応するシステム定義変数が TRUE となります。
ユーザ異常は、動作モードの変更により解除されません。

● ユーザ情報

異常ではないユーザ定義の通知情報です。
ユーザ情報発生（SetInfo）命令によって、ユーザ情報を発生させます。
このとき、ユーザ情報がイベントログへ記録されます。



使用手順

以下の手順で使用します。

● ユーザ異常

1. イベント設定テーブルに、「ユーザ異常」を登録
↓
2. ユーザ異常発生命令 SetAlarm を実行
（イベント設定テーブルで定義したイベントコードを指定）
↓
3. ユーザ異常が発生
↓
4. 該当するシステム定義変数「_AlarmFlag」（ユーザ異常ステータス）が TRUE となる。その条件で、任意の処理を実行
↓
5. Sysmac Studio、命令、または表示器から、ユーザ異常を確認

● ユーザ情報

1. イベント設定テーブルに、「ユーザ情報」を登録
↓
2. ユーザ情報発生命令 SetInfo を実行
↓
3. イベントログへの登録を確認

イベント設定テーブルの設定

「ユーザ異常」または「ユーザ情報」を発生させるには、まず Sysmac Studio にて、「イベント設定テーブル」に「ユーザ異常」または「ユーザ情報」をあらかじめ登録しておきます。
ここで設定したユーザイベントは、Sysmac Studio または表示器で、同じ内容を表示できます。
イベント設定テーブルに登録できるイベントの総数は、最大 5120 件です。

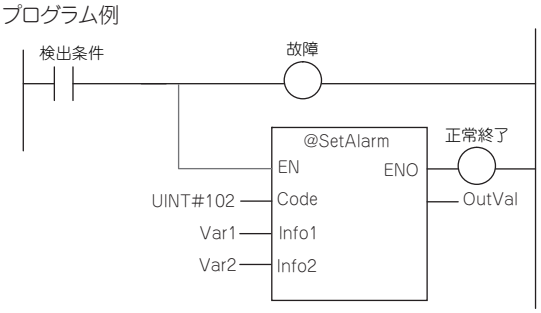
イベント設定テーブルの設定画面

イベント設定テーブル

イベントコード	イベント名	重要度	グループ	詳細情報
10001	故障X	User fault Level3	制御A(001)	
:	:	:	:	:

詳細情報
■内容
故障Xが発生しました。
■対処
安全を確認し、要因コードに従って対処してください。

プログラム例



イベント設定テーブルに以下を設定します。



● イベント設定テーブルの内容

項目	内容	値
イベントコード	イベントを識別する番号をイベント重要度に応じて指定することができます。	ユーザ異常：1 ～ 40000 ユーザ情報：40001 ～ 60000
イベント名	イベントの名称を表すタイトルを記載することができます。	最大 128 文字
重要度	イベントの重要度を指定します。 重要度を数値で表し、数値が小さいほど重要度が高いと位置づけられます。	ユーザ異常：User fault Level1 ～ 8 ユーザ情報：User Information
グループ	イベントの発生箇所や分類などを表すグループ名を指定することができます。各イベントをユーザ定義でグルーピングすることができます。	最大 32 文字 使用できる文字制限なし。 大文字／小文字の区別あり。 予約語なし。
詳細情報	イベントの内容を説明するメッセージを記載することができます。 ユーザが自由に任意の文字列を記入できます。 Sysmac Studio ／表示器でイベントを表示する際に使用します。	最大 1024 文字 使用できる文字制限なし。 大文字／小文字の区別あり。 予約語なし。
全停止フォールトレベルのコントローラ異常発生時、表示器に表示する異常詳細情報	後述の「参考：全停止フォールトレベルのコントローラ異常発生時に表示器でユーザメッセージを表示する機能：」を参照してください。	最大 128 文字 使用できる文字制限なし。 大文字 / 小文字の区別あり。 予約語なし。
コメント	複数テーブルごとに情報を添付できます。	

参考

同一イベントコードに対して、複数の地域または複数のユーザ向けに、最大 9 言語（9 通り）の設定が可能です。イベント設定テーブルは、Sysmac Studio にて、Microsoft Excel ファイルからのクリップボード経由でのインポートが可能です。

参考

全停止フォールトレベルのコントローラ異常発生時に表示器でユーザメッセージを表示する機能：
全停止フォールトレベルのコントローラ異常発生時、ユーザプログラムの実行は停止します。NY シリーズでは、全停止フォールトレベルのコントローラ異常発生時に表示器にユーザ向けメッセージを表示させる機能があります。そのメッセージは Sysmac Studio のイベント設定テーブルのユーザイベント一覧の下段に設定します。

● イベント重要度とイベントコード

イベント種別	重要度	イベント重要度の分類名 (*)	対応するイベントコードの範囲	説明
ユーザ異常	高い	User fault Level1	1 ～ 5000	ユーザは、8 段階から選択する。
		User fault Level2	5001 ～ 10000	
		User fault Level3	10001 ～ 15000	
		User fault Level4	15001 ～ 20000	
		User fault Level5	20001 ～ 25000	
		User fault Level6	25001 ～ 30000	
		User fault Level7	30001 ～ 35000	
	低い	User fault Level8	35001 ～ 40000	
ユーザ情報	もっとも低い	User Information	40001 ～ 60000	イベント種別が「ユーザ情報」の重要度

* ユーザ異常の重要度は、コントローラ異常の重要度と異なる分類です。

使用上の注意

イベント設定テーブルを更新し、コントローラに転送した場合、それ以前のユーザイベントのイベントログは古い内容で記録されています。イベント設定テーブルは新しくなっているため不整合が発生する場合があります。ご注意ください。

関連命令

ユーザ異常を発生、解除する命令と、発生したユーザ異常を確認する命令があります。

● ユーザ異常を発生、解除する

以下の命令で、ユーザ異常を発生／解除、ユーザ情報を発生させます。
同時に発生中の状態にできるユーザ異常の数は、各イベント重要度ごとに 32 件ずつ、合計で最大 256 件です。

命令名称	命令	機能
ユーザ異常発生命令	SetAlarm	ユーザ異常を発生させます。
ユーザ異常解除命令	ResetAlarm	ユーザ異常を解除します。
ユーザ情報発生命令	SetInfo	任意のユーザ情報を、イベントログへ記録します。

● ユーザ異常を確認する

ユーザ異常ステータス取得 （GetAlarm）命令で、ユーザ異常の発生有無と、発生中のユーザ異常のうち最重要のイベント重要度とそのイベントコードを取得できます。

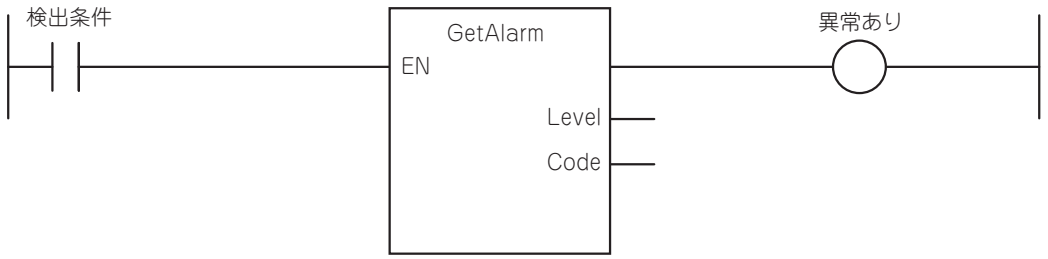
例）

イベント設定テーブル

イベントコード	イベント名	重要度	グループ	詳細情報
10001	故障X	User fault Level3	制御A(001)	
:	:	:	:	:

詳細情報
■内容 故障Xが発生しました。 ■対処 安全を確認し、要因コードに従って対処してください。

実際のプログラム



参考

ユーザ異常によって、発生したコントローラ異常に対して、対処方法などのメッセージを付加できます。
発生中のコントローラ異常について、異常ステータスやイベントコードを GetPLCError などのコントローラ異常のステータス取得命令で取得し、その情報をトリガにユーザ異常を発生させます。

例 1

バッテリ電圧低下発生時、イベントコード（16#000B0000）を取得し、以下のメッセージを表示します。

バッテリ切れ。 バッテリの交換前には 5 分以上通電し、電源 OFF 後 5 分以内に、新しいバッテリと交換してください。

例 2

部分停止フォールトレベルのコントローラ異常が発生時、イベント重要度（最重要のステータス：2）を取得し、以下のメッセージを表示します。

装置が故障しました。下記にお問い合わせください。 ＜修理の連絡先＞ 受付時間 AM8：00 ～ PM9：00 TEL ＊＊＊ - ＊＊＊＊ - ＊＊＊＊

ユーザ異常関連のシステム定義変数

変数名	名称	機能	データ型	R/W
_AlarmFlag	ユーザ異常ステータス	ユーザ異常発生中はイベント重要度に該当するビットが TRUE となります。 User fault Level1 ～ 8 が、ビット 00 ～ 07 に対応します。	WORD	R

イベントログへの記録

ユーザ異常またはユーザ情報発生時、ResetAlarm によって解除した時にイベントログが記録されます。その際、イベントログのユーザイベントログには、発生時刻、イベントコード、イベント重要度、付属情報 1、2 が記録されます。

ユーザ異常の解除

ユーザ異常は、NY シリーズコントローラの電源を OFF → ON にすることで解除されます。
Sysmac Studio、命令（ユーザ異常解除命令 ResetAlarm）、または表示器から、異常を解除することもできます。
詳細は、🕮『Sysmac Studio Version 1 オペレーションマニュアル（SBCA-470）』を参照してください。

8-6　イベント重要度変更機能

NY シリーズコントローラで発生する、異常、状態変化、ユーザ定義事象などを総称して、イベントと呼びます。ユーザは、Sysmac Studio の表示や、産業用 PC の LED の動作によって、どのようなイベントが発生したかを知ることができます。
イベントには、システムによってあらかじめ定義されているコントローライベントと、ユーザが任意に定義するユーザイベントがあります。そして、各コントローライベントは、5 段階のイベント重要度に分類されています。イベント重要度の詳細は、🕮「 イベント重要度」（P.8-48）を参照してください。
イベント重要度変更機能は、各コントローライベントに割り当てられているイベント重要度を、ユーザが任意に変更できる機能です。

8-6-1　イベント重要度変更の用途

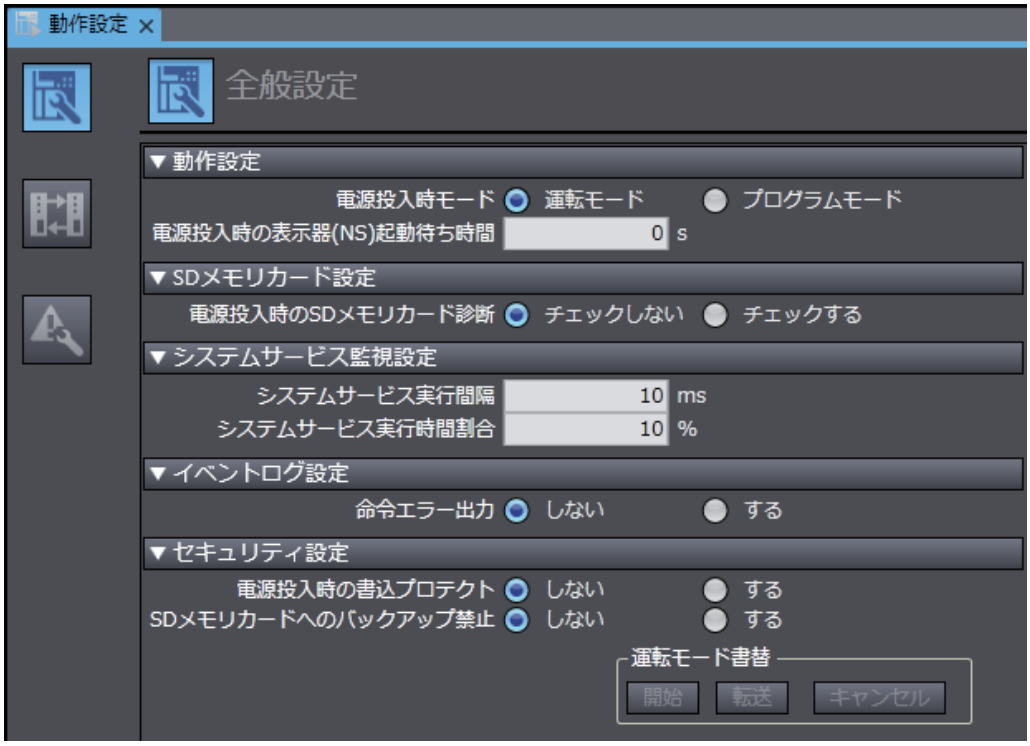
発生したコントローライベントの重要度に応じて、産業用 PC の LED などの動作が、あらかじめ規定されています。イベント重要度変更機能を使用することにより、任意のイベント発生に対して、このようなコントローラの動作を変更することが可能になります。
たとえば ERROR LED は、イベント重要度が軽度フォールトレベルなら点滅しますが、監視情報レベルなら消灯のままです。イベント重要度変更機能を使えば、任意のイベント発生に対して ERROR LED が消灯するように設定したり、点滅するように設定したりすることができます。
コントローラのイベント重要度ごとの動作については、🕮『NY シリーズ トラブルシューティングマニュアル（SBCA-438）』を参照してください。

8-6-2　イベント重要度変更が可能なイベント

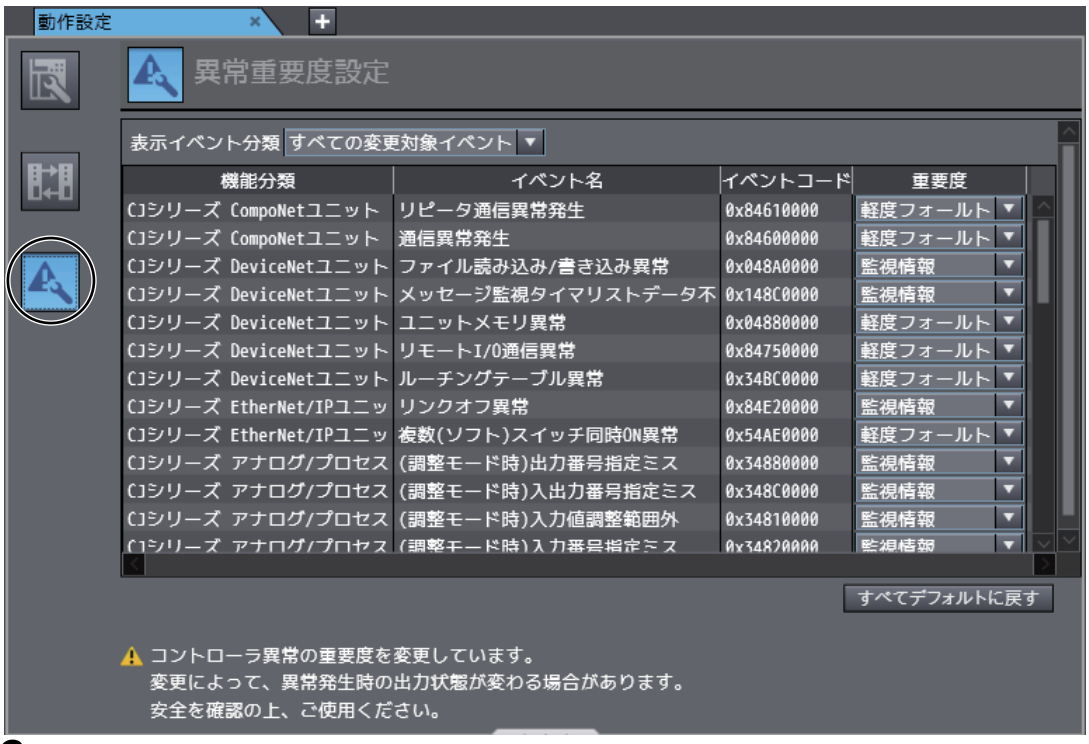
イベントの重要度が変更可能かどうかは、イベントごとに決められています。
コントローライベントの種類、重要度、重要度変更の可否の詳細は、🕮『NY シリーズ トラブルシューティングマニュアル（SBCA-438）』を参照してください。

8-6-3　イベント重要度変更の操作方法

1 **Sysmac Studio の、[構成・設定]-[コントローラ設定]-[動作設定] をダブルクリック、または右クリックして [編集] を選択します。**
エディットウィンドウに動作設定の [全般設定] ビューが表示されます。



2 **[異常重要度設定] ボタンをクリックします。**
重要度の変更が可能なイベントの一覧が表示されます。



3 **重要度を変更するイベントの [重要度] を、変更先の重要度に設定します。**

使用上の注意

Sysmac Studio でイベント重要度変更設定を行い、それをコントローラにダウンロードした際に、すでにそのイベントが発生していた場合、ダウンロード開始時にそのイベントはいったん解除されます。そして、ダウンロード中に、再度そのイベントが発生した場合、変更前の重要度に応じて動作します。ダウンロード完了後にそのイベントが発生した場合は、変更後の重要度に応じた動作に切り替わります。

9

バックアップ機能

本章では、NY シリーズコントローラの各種設定データの、バックアップ機能について説明します。バックアップ機能には、いくつかの種類があり、それぞれでバックアップ対象となるデータや保存先が異なります。まずバックアップ機能の全体を説明し、その後で、種類ごとの説明をします。

9-1 バックアップ機能の全体 9-3
9-1-1 バックアップ機能の用途 9-3
9-1-2 バックアップ機能の対象データ 9-4
9-1-3 バックアップ機能の種類 9-5
9-1-4 バックアップ機能の種類とデータグループの関係 9-7
9-1-5 バックアップ機能の対象範囲 9-8

9-2 SD メモリカードバックアップ機能 9-10
9-2-1 バックアップ（コントローラから仮想 SD メモリカード） 9-11
9-2-2 照合（コントローラと仮想 SD メモリカード間） 9-15

9-3 SD メモリカードへのバックアップ禁止機能 9-19

9-4 Sysmac Studio コントローラバックアップ機能 9-20
9-4-1 バックアップ（コントローラからパソコン） 9-21
9-4-2 リストア（パソコンからコントローラ） 9-22
9-4-3 照合（コントローラとパソコン間） 9-23

9-5 Sysmac Studio バックアップファイルのインポート / エクスポート機能 . . 9-24

9-6 Sysmac Studio 変数／メモリのバックアップ機能 9-25
9-6-1 Sysmac Studio 変数／メモリのバックアップ機能の対象データ 9-25
9-6-2 Sysmac Studio 変数／メモリのバックアップ機能の操作方法 9-25
9-6-3 NY シリーズコントローラの形式間の互換性 9-25

9-7 EtherCAT スレーブ接続時のバックアップ機能 9-26
9-7-1 EtherCAT スレーブのバックアップ対象データ 9-26
9-7-2 コントローラの各状態でのバックアップ機能実行可否 9-27
9-7-3 EtherCAT スレーブのリストア実行条件 9-28
9-7-4 バックアップ機能の対象 EtherCAT スレーブ 9-29

9-8 EtherCAT スレーブターミナル接続時のバックアップ機能 9-31
9-8-1 EtherCAT スレーブターミナルのバックアップ対象データ 9-31
9-8-2 EtherCAT スレーブターミナルの各状態でのバックアップ機能実行可否 . . 9-32
9-8-3 EtherCAT スレーブターミナルのリストア実行条件 9-32

9-9 バックアップ関連ファイル 9-33
9-9-1 バックアップ関連ファイルの種類 9-33
9-9-2 バックアップファイルの仕様 9-34
9-9-3 リストアコマンドファイルの仕様 9-35

9-9-4 コントローラ照合結果ファイルの仕様 9-38
9-9-5 EtherCAT 照合結果ファイルの仕様 9-39
9-9-6 EtherCAT スレーブターミナル照合結果ファイルの仕様 9-40

9-10 バックアップ機能のファイル間互換性 9-41
9-10-1 バックアップ機能間の互換性 9-41
9-10-2 NY シリーズコントローラの形式間の互換性 9-42
9-10-3 NY シリーズコントローラのユニットバージョンの互換性 9-43

9-11 バックアップ機能と同時に実行できない機能 9-44

9-1　バックアップ機能の全体

NY シリーズコントローラのバックアップ機能とは、以下の 3 つを指します。

機能	内容
バックアップ	コントローラ内の各種設定データをまとめて仮想 SD メモリカードやパソコンに保存する機能です。保存するファイルのことをバックアップファイルと呼びます。
リストア	仮想 SD メモリカードやパソコン内のバックアップファイルを、コントローラに転送する機能です。コントローラ内の各種設定データが、バックアップ実行時のものに置き換わります。
照合	仮想 SD メモリカードやパソコン内のバックアップファイルの内容と、現在のコントローラ内の各種設定データとが一致するかどうかを判定する機能です。

ここでは、バックアップ機能の全体として、以下のことを説明します。

項目	内容
バックアップ機能の用途	バックアップ機能の有効な使い方を説明します。
バックアップ機能の操作手順の例	バックアップ機能は、簡単な操作で実行できます。その例を示します。
バックアップ機能の対象データ	バックアップ機能を使うと、接続しているユニットやスレーブ内の、どのデータが保存できるのかを説明します。
バックアップ機能の種類	バックアップ機能には、データの保存先などの違いによって、いくつかの種類があります。どんな種類があるかと、それらの違いについて説明します。
バックアップ機能の種類とデータグループの関係	バックアップ機能の種類によって、対象となるデータグループが異なります。バックアップ機能の種類と、データグループとの関係を、説明します。
バックアップ機能の対象範囲	バックアップ機能を使うと、接続しているどのユニットやスレーブの設定データが保存できるのかを説明します。

安全上の要点

変数の現在値のバックアップは、保持変数が更新されない状態で実施することを推奨します。保持変数の値が更新される状態で以下の変数をバックアップすると、正しく保存できないことがあります。

- データサイズが 16 ビット以上の構造体のメンバ
- データサイズが 16 ビット以上の配列の要素

9-1-1　バックアップ機能の用途

バックアップ機能は、以下のような用途で使用します。

項目	用途
プログラム、設定変更	稼動中の装置のユーザプログラムや設定を変更するとき。
ハード交換	産業用 PC、ユニット、スレーブのハード交換をするとき。
装置異常時の原因調査	装置に異常が発生したときに原因を解析するためコントローラのデータを保存するとき。
装置のバックアップと原状回復	装置に異常が発生したとき、装置を正常な状態データで復元するとき。また、あらかじめ稼働中の装置のデータをバックアップするとき。
装置の製造	同じ装置を製造するため複製元の装置のデータを購入後の初期状態の装置に転送するとき。

9-1-2 バックアップ機能の対象データ

バックアップ機能の対象データは以下のとおりです。
なお、バックアップ機能の詳細説明では対象データを表すときに「データグループ」を使用します。

データグループ	データ項目
ユーザプログラム・設定	EtherCAT 構成（EtherCAT スレーブ構成、EtherCAT マスタ設定） ユニット構成・ユニット設定 I/O マップ コントローラ設定（動作設定、内蔵 EtherNet/IP ポート設定） モーション制御設定 カムデータ設定 イベント設定 タスク設定 データトレース設定 タグデータリンクテーブル コントローラ名称 操作権限認証 ユーザプログラム実行用 ID（ユーザプログラム内） POU データ（データ型、グローバル変数）
内蔵 EtherNet/IP ポートの IP アドレス*1	内蔵 EtherNet/IP ポート設定の TCP/IP 設定のうち、設定種別、IP アドレス、サブネットマスク、デフォルトゲートウェイ
変数の現在値	保持属性ありの変数の値*2
ユニット・スレーブ設定	EtherCAT スレーブのバックアップパラメータ*3
絶対値エンコーダ原点位置オフセット	絶対値エンコーダつきのサーボドライバの現在位置を復元するための設定値

*1. コントローラのユニットバージョン Ver.1.14 以降と Sysmac StudioVer.1.18 以降の組み合わせで、内蔵 EtherNet/IP ポート設定の IP アドレスは、1 つのデータグループとして使用可能です。
上記以外の組み合わせでは、IP アドレスはユーザプログラム・設定に含まれています。
*2. 保持属性ありのシステム定義変数のうち、バックアップ機能の対象外となるシステム定義変数があります。詳細は、🕮「A-4 システム定義変数の個別仕様」（P.A-46）で個別の仕様を確認してください。
*3. EtherCAT スレーブの中には、設定パラメータの一部または全部がバックアップ対象ではない機種があります。バックアップ機能の対象 EtherCAT スレーブの詳細は、🕮「9-7-4 バックアップ機能の対象 EtherCAT スレーブ」（P.9-29）を参照してください。



安全上の要点

絶対値エンコーダ原点位置オフセットに関する注意
製品は絶対値エンコーダの情報である「絶対値エンコーダ原点位置オフセット」を不揮発性メモリによって保持しています。以下のいずれかの条件を満たす場合は、絶対値エンコーダ原点位置オフセットをリストア対象からはずしてリストアを実施してください。その後、新たに絶対値エンコーダ原点設定を行ってください。絶対値エンコーダ原点設定をしなかった場合、制御対象が予期しない動作をするおそれがあります。

- サーボモータとドライバの個体が、バックアップを作成したときの個体と異なる。
- バックアップを作成した以降に、絶対値エンコーダのセットアップを実行した。
- 絶対値エンコーダの絶対値データが失われている。

9-1-3　バックアップ機能の種類

NY シリーズコントローラで使用できるバックアップ機能には、データの保存先が仮想 SD メモリカードのものと、パソコンのものがあります。また、操作方法として、システム定義変数によるもの、Sysmac Studio 画面によるもの、Industrial PC Support Utility によるものの 3 つがあります。

仮想 SD メモリカードにデータを保存する機能

仮想 SD メモリカードにデータをバックアップ、照合するのが、SD メモリカードバックアップ機能です。関連機能として、SD メモリカードバックアップ禁止機能があります。

機能名		内容	操作方法		参照
			システム定義変数	Sysmac Studio 画面	
SD メモリカードバックアップ機能	バックアップ	コントローラのデータを、仮想 SD メモリカードにバックアップファイルとして保存します。	○	○	「9-2-1 バックアップ（コントローラから仮想 SD メモリカード）」（P.9-11）
	照合	コントローラのデータと仮想 SD メモリカードのバックアップファイルのデータを照合します。	○	○	「9-2-2 照合（コントローラと仮想 SD メモリカード間）」（P.9-15）
SD メモリカードバックアップ禁止機能		仮想 SD メモリカードへのバックアップを禁止します。		○	「9-3 SD メモリカードへのバックアップ禁止機能」（P.9-19）

参考

- 仮想 SD メモリカードに保存したバックアップファイルをコントローラにリストアする場合は、Industrial PC Support Utility のコントローラバックアップ機能のリストアで、仮想 SD メモリカードに設定している共有フォルダを指定します。
- Sysmac Studio のコントローラバックアップ機能でもリストアできます。その場合、仮想 SD メモリカードに設定している共有フォルダに保存されたバックアップファイルを、外部ストレージなどを利用し、Sysmac Studio がインストールされたパソコンに保存しておく必要があります。

パソコンにデータを保存する機能

コントローラのデータをパソコンにバックアップ、リストア、照合するのが、Sysmac Studio コントローラバックアップ機能です。
コントローラを使用せず、Sysmac Studio のプロジェクトから各データをパソコンに保存、読出するのが、Sysmac Studio バックアップファイルのインポート／エクスポート機能です。
不揮発性メモリに保存される現在値だけをパソコンにバックアップ、リストアするのが、Sysmac Studio 変数／メモリのバックアップ機能です。

機能名		内容	操作方法		参照
			システム定義変数	Sysmac Studio 画面	
Sysmac Studio コントローラバックアップ機能	バックアップ	コントローラのデータを、パソコンにバックアップファイルとして保存します。		○	「9-4-1 バックアップ（コントローラからパソコン）」（P.9-21）
	リストア	パソコン内のバックアップファイルのデータを、コントローラに転送します。		○	「9-4-2 リストア（パソコンからコントローラ）」（P.9-22）
	照合	コントローラのデータとパソコンのバックアップファイルのデータを照合します。		○	「9-4-3 照合（コントローラとパソコン間）」（P.9-23）
Sysmac Studio バックアップファイルのインポート / エクスポート機能	エクスポート	コントローラを使用せず、Sysmac Studio のプロジェクトからバックアップファイルを作成します。		○	「9-5 Sysmac Studio バックアップファイルのインポート / エクスポート機能」（P.9-24）
	インポート	コントローラを使用せず、バックアップファイルを Sysmac Studio のプロジェクトに読み出します。		○	
Sysmac Studio 変数／メモリのバックアップ機能	バックアップ	不揮発性メモリに保持されるデータの現在値を、パソコンにファイル（XML 形式）としてバックアップ（読出）します。		○	「9-6 Sysmac Studio 変数／メモリのバックアップ機能」（P.9-25）
	リストア	不揮発性メモリに保持されるデータの現在値を、パソコンからコントローラに対してリストア（書込）します。		○	

Windows 側のフォルダにデータを保存する機能

コントローラのデータを Windows 側のフォルダにバックアップ、リストア、照合するのが、Industrial PC Support Utility コントローラバックアップ機能です。

Industrial PC Support Utility コントローラバックアップ機能の詳細ついては、『NY シリーズ 産業用パネル型 PC ／産業用ボックス型 PC ユーザーズマニュアル セットアップ編（SBCA-441）』を参照してください。

9-1-4　バックアップ機能の種類とデータグループの関係

バックアップ機能の種類によって、対象となるデータグループが異なります。バックアップ機能の種類と対象データグループとの関係を、以下に示します。

（○：対象　×：対象外）

バックアップ機能の種類		データグループ				
		ユーザプログラム・設定		変数の現在値	ユニット・スレーブ設定	絶対値エンコーダ原点位置オフセット
			内蔵 EtherNet/IP ポートの IP アドレス*1			
SD メモリカードバックアップ機能	バックアップ	○	○	○*2	○	○
	リストア	×	×	×	×	×
	照合*3	○*4	○	×	○	×
Sysmac Studio コントローラバックアップ機能	バックアップ	○	○	○*2	○*5	○
	リストア	○	○	○*2	○*5	○
	照合	○*4	○	×	○*5	×
Industrial PC Support Utility コントローラバックアップ機能	バックアップ	○	○	○*2	○*5	○
	リストア	○	○	○*2	○*5	○
	照合	○*4	○	×	○*5	×
Sysmac Studio バックアップファイルのインポート / エクスポート機能	バックアップファイルのエクスポート	○*6	○	×	×	×
	バックアップファイルインポート	○*6	○	×	○	×
Sysmac Studio 変数／メモリのバックアップ機能	バックアップリストア	×	×	○*2	×	○

*1. コントローラのユニットバージョン Ver.1.14 以降と Sysmac Studio Ver.1.18 以降の組み合わせで、内蔵 EtherNet/IP ポート設定の IP アドレスは、1 つのデータグループとして使用可能です。
上記以外の組み合わせでは、IP アドレスはユーザプログラム・設定に含まれています。
*2. バックアップの対象データが、保持属性の変数の現在値の場合にだけ、処理します。
*3. 対象データグループのうち、ユーザプログラム・設定以外のデータは、リストアコマンドファイルで「リストアする」と指定したものだけを対象とします。
*4. ユーザプログラム・設定データグループのうち、データトレース設定は照合しません。
*5. バックアップの対象が EtherCAT スレーブの場合は、EtherCAT スレーブのバックアップパラメータを対象とします。
*6. 以下のデータは、処理の対象外です。内蔵 EtherNet/IP 用タグデータリンク設定、操作権限認証、データトレース設定。

参考

Sysmac Studio 変数／メモリのバックアップ機能で扱うファイルは、その他のバックアップ機能で扱うファイルと異なるため互換性がありません。
Sysmac Studio 変数／メモリのバックアップ機能の詳細は 🕮 「9-6 Sysmac Studio 変数／メモリのバックアップ機能」（P.9-25）を参照してください。

9-1-5　バックアップ機能の対象範囲

バックアップ機能によって、対象となるスレーブやユニットが異なります。各バックアップ機能の対象スレーブ、ユニットを、以下に示します。

（○：対象　×：対象外）

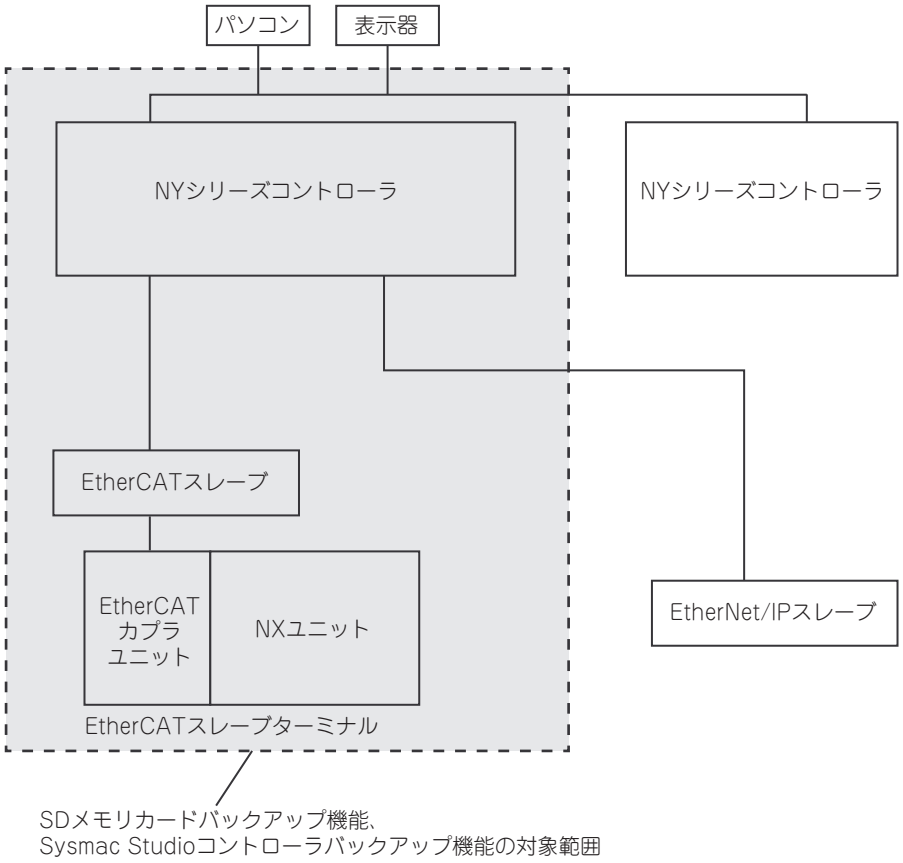
バックアップ機能の種類	スレーブ、ユニット			
	NY シリーズコントローラ	EtherCAT スレーブ*1	EtherNet/IP スレーブ	パソコン、表示器
SD メモリカードバックアップ機能	○	○*2	×	×
Sysmac Studio コントローラバックアップ機能	○	○*2	×	×
Sysmac Studio バックアップファイルのインポート / エクスポート機能	○	○*3	×	×
Sysmac Studio 変数／メモリのバックアップ機能	○	×	×	×

*1. EtherCAT スレーブターミナルを含みます。バックアップ機能の対象が EtherCAT スレーブターミナルの場合、EtherCAT カプラユニットも各 NX ユニットも、バックアップ機能の対象になります。

*2. セーフティコントロールユニットは、対象外です。セーフティコントロールユニットの設定データのインポート／エクスポート機能は、🕮『NX シリーズ セーフティコントロールユニット ユーザーズマニュアル（SGFM-710）』を参照してください。

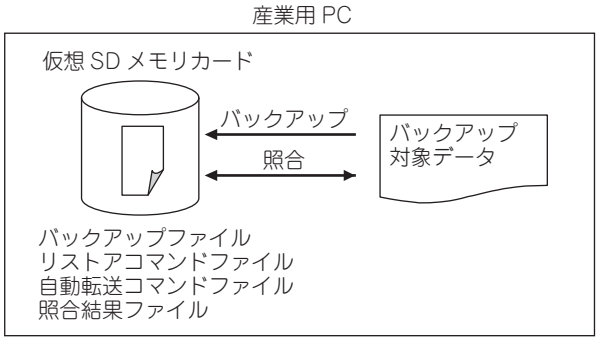
*3. インポートだけが可能です。エクスポート機能は、対象外です。

SD メモリカードバックアップ機能、および Sysmac Studio コントローラバックアップ機能の対象スレーブ、ユニットは、つぎの図の範囲です。



9-2　SD メモリカードバックアップ機能

仮想 SD メモリカードを用いて、コントローラのデータのバックアップ、照合をする機能です。



バックアップを実行すると、仮想 SD メモリカードに「バックアップファイル」「リストアコマンドファイル」「自動転送コマンドファイル」が指定ディレクトリに作成されます。また、照合を実行すると、「照合結果ファイル」が指定ディレクトリに作成されます。
これらを総称して「バックアップ関連ファイル」と呼びます。
バックアップ関連ファイルの役割は以下のとおりです。

ファイル	役割		
	内容	バックアップ	照合
バックアップファイル	バックアップ関連機能で扱う、コントローラのデータをファイル化したものです。	作成	参照
リストアコマンドファイル	リストアを実行するときに、リストア対象のデータグループを指定するファイルです。パソコンのテキストエディタでファイルを編集することでリストア対象を指定します。	作成	参照
自動転送コマンドファイル	NY シリーズコントローラでは、このファイルを使用しません。	作成	何もしない
照合結果ファイル	照合を実行した後に、照合結果を格納するファイルです。	何もしない	作成

この機能の操作方法と、対象ディレクトリ、実行可能な動作モードは以下のとおりです。

操作方法	ディレクトリ*1	実行可能な動作モード		
		バックアップ	リストア	照合
システム定義変数 *2*3	システム定義変数で指定したディレクトリ	運転モード、プログラムモード	実行不可	運転モード、プログラムモード
Sysmac Studio SD メモリカード画面	画面で指定したディレクトリ	運転モード、プログラムモード	実行不可	運転モード、プログラムモード

*1. 仮想 SD メモリカード内だけ指定可能です。
*2. 表示器から指示するときに使用します。ユーザプログラムからはこのシステム定義変数にアクセスできません。
*3. バックアップ中または照合中に、ほかの表示器からバックアップや照合の指示が出ないように、ユーザ側で管理してください。そうでないと、意図どおりの動作をしない場合があります。

9-2-1　バックアップ（コントローラから仮想 SD メモリカード）

コントローラのデータを、仮想 SD メモリカードに保存します。

処理内容

- バックアップの対象データは、すべてのデータグループです。
- バックアップを実行すると、仮想 SD メモリカードに「バックアップファイル」「リストアコマンドファイル」「自動転送コマンドファイル」が指定ディレクトリに作成されます。
- 指定ディレクトリに既存のバックアップ関連ファイルがあるときは、上書きされます。
- 仮想 SD メモリカードにバックアップ関連ファイルを書込中にエラー発生した場合は、既存のバックアップ関連ファイルが削除されます。そして、新しいバックアップ関連ファイルも作成されません。
- 新しいバックアップ関連ファイルの作成前にエラー発生したときは、既存のものが保持され、新しいファイルは作成されません。
- バックアップ完了後、仮想 SD メモリカードは認識状態を継続します。

操作方法

● システム定義変数 _Card1BkupCmd（SD メモリカードバックアップ指示）による方法

処理段階	操作方法
開始指示	システム定義変数 _Card1BkupCmd.DirName（ディレクトリ名）に保存先のディレクトリ名を格納します。 （例） ディレクトリ dirA の下のサブディレクトリ dirB を指定："dirA/dirB" システム定義変数 _Card1BkupCmd.ExecBkup（バックアップ起動フラグ）を FALSE → TRUE にすると実行が開始されます。
取消指示	バックアップ実行の指示を取り消します。 システム定義変数 _Card1BkupCmd.CancelBkup（バックアップ取消フラグ）を TRUE にすると、バックアップ処理が異常終了します。
実行中	システム定義変数 _Card1BkupSta.Active（実行中フラグ）が TRUE となります。 システム定義変数 _BackupBusy（バックアップ関連機能実行中フラグ）の値が TRUE になります。
実行結果	正常終了のとき システム定義変数 _Card1BkupSta.Done（完了フラグ）が TRUE となります。 異常終了のとき システム定義変数 _Card1BkupSta.Err（エラーフラグ）が TRUE となります。

（注） ユーザプログラムからは、このシステム定義変数にアクセスできません。

● Sysmac Studio の SD メモリカード画面による方法

処理段階	操作方法
開始指示	Sysmac Studio SD メモリカード画面の［SD メモリカードバックアップ］ボタンを押し、保存先のディレクトリ名を指定してバックアップ実行を指示します。
実行中	バックアップの進行状況を示す画面が表示されます。 システム定義変数 _BackupBusy（バックアップ関連機能実行中フラグ）の値が TRUE になります。
実行結果	完了メッセージが表示され、続けてバックアップデータの照合実施を確認されます。

操作の詳細は、🕮『Sysmac Studio Version 1 オペレーションマニュアル（SBCA-470）』を参照してください。

● 専用命令による方法

処理段階	操作方法
開始指示	ユーザプログラム内で、BackupToMemoryCard 命令を実行します。
実行中	BackupToMemoryCard 命令の出力変数 Busy の値が、TRUE になります。 システム定義変数 _BackupBusy（バックアップ関連機能実行中フラグ）の値が TRUE になります。
実行結果	正常終了のとき BackupToMemoryCard 命令の出力変数 Done の値が、TRUE になります。 異常終了のとき BackupToMemoryCard 命令の出力変数 Error の値が、TRUE になります。 BackupToMemoryCard 命令の出力変数 ErrorID に、エラーコードが格納されます。

関連システム定義変数

システム定義変数変数による方法でバックアップを行う場合、関連するシステム定義変数には、以下のものがあります。システム定義変数の詳細は、📖「A-4 システム定義変数の個別仕様」（P.A-46）を参照してください。

変数名 メンバ名	名称	機能	データ型	R/W
_Card1BkupCmd*1	SD メモリカードバックアップ指示		_sBKUP_CMD	RW
ExecBkup*1	バックアップ起動フラグ	コントローラのデータを仮想 SD メモリカードへバックアップするときに、TRUE にします。	BOOL	RW
CancelBkup*1	バックアップ取消フラグ	仮想 SD メモリカードへのバックアップを中止するときに、TRUE にします。	BOOL	RW
DirName*1	ディレクトリ名	バックアップ対象の仮想 SD メモリカードのディレクトリ名を指定します。	STRING (64)	RW
_Card1BkupSta*1	SD メモリカードバックアップステータス		_sBKUP_STA	R
Done*1	完了フラグ	バックアップ実行完了のとき、TRUE となります。	BOOL	R
Active*1	実行中フラグ	バックアップ実行中のとき、TRUE となります。	BOOL	R
Err*1	エラーフラグ	バックアップ処理が異常終了したとき、TRUE となります。	BOOL	R
_BackupBusy	バックアップ関連機能実行中フラグ	バックアップ、リストア、照合の実行中のとき、TRUE となります。	BOOL	R

*1. ユーザプログラムからは、このシステム定義変数にアクセスできません。

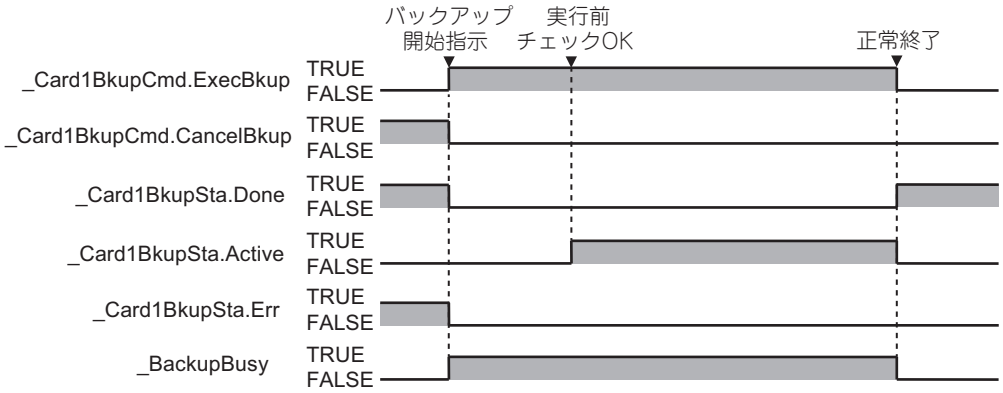
参考

- NY シリーズコントローラと NA シリーズ表示器を接続して使用する場合の変数マッピングについては、📖『プログラマブルターミナル NA シリーズ ユーザーズマニュアル ソフトウェア編 (SBSA-546)』を参照してください。
- NY シリーズコントローラと NS シリーズ表示器を接続して使用する場合に、CX-Designer の変数テーブルにこれらのシステム定義変数を登録する方法については、📖「A-7 CX-Designer の変数テーブル登録方法」（P.A-96）を参照してください。

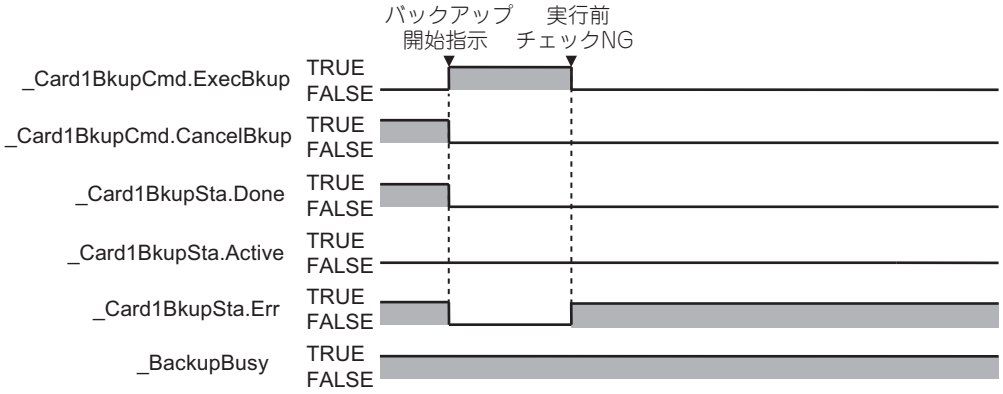
タイミングチャート

システム定義変数変数による方法でバックアップを行う場合の、システム定義変数の動作は以下のとおりです。
図中の「実行前チェック」とは、バックアップが開始される前に、仮想 SD メモリカードが存在するかどうかなどをチェックする処理のことです。_Card1BkupSta.Active（実行中フラグ）の値が TRUE になるのは、実行前チェックの結果が OK となり、実際にバックアップが開始されたときです。

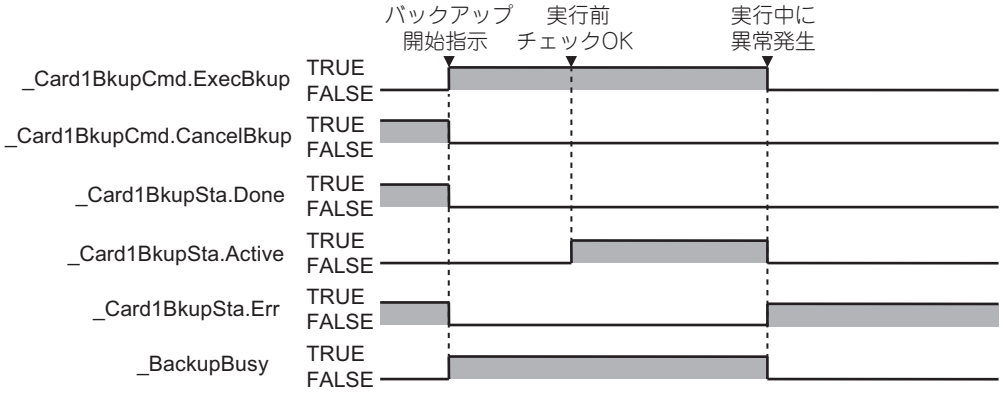
● 正常動作



● 別のバックアップ機能実行中により、バックアップを開始できなかった場合

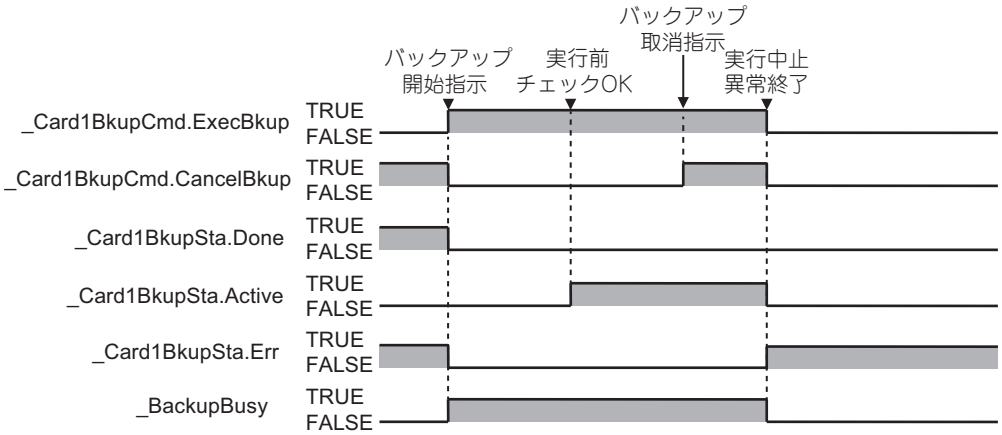


● 正常にバックアップ開始後、何らかの原因によりバックアップに失敗した場合



● バックアップ実行中に取消指示を出した場合

取消指示を出してからバックアップ実行中止までの時間は、バックアップ処理の進行状況によって変化します。



処理時間とバックアップファイルのサイズ

バックアップに要する時間は、動作モード、ユニット構成、ユーザプログラムの内容などによって異なります。また、バックアップファイルのサイズは、ユニット構成やユーザプログラムの内容などによって異なります。以下に、バックアップ時間とバックアップファイルのサイズの目安を示します。

コントローラ	動作モード	接続したEtherCATスレーブ	ユーザ定義のPOU数	ユーザプログラムのメモリサイズ [MB]	バックアップ時間 [s]	バックアップファイルのサイズ [MB]
形 NY5 □ 2- □□□□	プログラムモード	*1	72	3.82	約 155	20.27

*1. AC サーボドライバ 形 R88D-KNA-ECT:32 台、EtherCAT カプラユニット 形 NX-ECC201:5 台の構成。
EtherCAT カプラユニット上の NX ユニットは、以下の構成。
NX-PD1000:5 台、NX-AD4608:5 台、NX-DA3605:5 台、NX-ID3343:64 台、NX-OD3153:64 台

9-2-2　照合（コントローラと仮想 SD メモリカード間）

コントローラのデータと、仮想 SD メモリカードのバックアップファイルのデータを照合します。

処理内容

- コントローラのデータと仮想 SD メモリカードの指定ディレクトリに格納されたバックアップファイルのデータを照合します。
- 照合対象データは、リストアコマンドファイル（RestoreCommand.ini）で「リストアする」と指定したデータグループです。
- 変数の現在値、絶対値エンコーダ原点位置オフセットは、照合実行中に値が書き換わることがあるため、照合しません。
- 照合を実行すると、「照合結果ファイル（VerifyResult.log）」が指定ディレクトリに作成され、照合結果が格納されます。照合実行前に指定ディレクトリに照合結果ファイルがあるときは、上書きされます。
- 仮想 SD メモリカードの指定ディレクトリにリストアコマンドファイルが存在しない場合、指定ディレクトリに存在する照合可能なバックアップファイルのデータを、すべて照合します。
- バックアップファイルのユニット・スレーブ構成と、照合先の実際のユニット・スレーブ構成が異なっている場合、照合エラーが発生します。
- 照合完了後、仮想 SD メモリカードは認識状態を継続します。

操作方法

● システム定義変数 _Card1BkupCmd（SD メモリカードバックアップ指示）

処理段階	操作手順
開始指示	システム定義変数 _Card1BkupCmd.DirName（ディレクトリ名）に保存先のディレクトリ名を格納します。 （例） ディレクトリ dirA の下のサブディレクトリ dirB を指定："dirA/dirB" システム定義変数 _Card1BkupCmd.ExecVefy（照合起動フラグ）を FALSE → TRUE にすると実行が開始されます。
取消指示	照合実行の指示を取り消します。 システム定義変数 _Card1BkupCmd.CancelVefy（照合取消フラグ）を TRUE にすると、照合処理が異常終了します。
実行中	システム定義変数 _Card1VefySta.Active（実行中フラグ）が TRUE となります。 システム定義変数 _BackupBusy（バックアップ関連機能実行中フラグ）の値が TRUE になります。
実行結果	正常終了し照合一致のとき システム定義変数 _Card1BkupSta.Done（完了フラグ）と _Card1BkupSta.VefyRslt（照合結果フラグ）が TRUE となります。 正常終了し照合不一致のとき システム定義変数 _Card1BkupSta.Done（完了フラグ）と TRUE、 _Card1BkupSta.VefyRslt（照合結果フラグ）が FALSE、となります。 異常終了のとき システム定義変数 _Card1BkupSta.Err（エラーフラグ）が TRUE となります。

（注） ユーザプログラムからはこのシステム定義変数にアクセスできません。

● Sysmac Studio の SD メモリカード画面による方法

処理段階	処理手順
開始指示	Sysmac Studio SD メモリカード画面の［SD メモリカード照合］ボタンを押し、照合対象の保存先のディレクトリ名を指定して照合実行を指示します。
実行中	照合の進行状況を示す画面が表示されます。システム定義変数 _BackupBusy（バックアップ関連機能実行中フラグ）の値が TRUE になります。
実行結果	照合結果を示す画面が表示されます。

操作の詳細は、🕮『Sysmac Studio Version 1 オペレーションマニュアル（SBCA-470）』を参照してください。

関連システム定義変数

システム定義変数変数による方法で照合を行う場合、関連するシステム定義変数には、以下のものがあります。システム定義変数の詳細は、🕮「A-4 システム定義変数の個別仕様」（P.A-46）を参照してください。

変数名 メンバ名	名称	機能	データ型	R/W
_Card1BkupCmd*1	SD メモリカードバックアップ指示		_sBKUP_CMD	RW
ExecVefy*1	照合起動フラグ	コントローラと仮想 SD メモリカード内のバックアップファイルを照合するときに、TRUE にします。	BOOL	RW
CancelVefy*1	照合取消フラグ	コントローラと仮想 SD メモリカード内のバックアップファイルの照合を中止するときに、TRUE にします。	BOOL	RW
DirName*1	ディレクトリ名	バックアップ対象の仮想 SD メモリカードのディレクトリ名を指定します。	STRING（64）	RW
_Card1VefySta*1	SD メモリカード照合ステータス		_sVEFY_STA	R
Done*1	完了フラグ	照合実行完了のとき、TRUE となります。	BOOL	R
Active*1	実行中フラグ	照合実行中のとき、TRUE となります。	BOOL	R
VefyRslt*1	照合結果フラグ	照合一致のとき TRUE、照合不一致のとき FALSE となります。	BOOL	R
Err*1	エラーフラグ	照合処理が異常終了したとき、TRUE となります。	BOOL	R
_BackupBusy	バックアップ関連機能実行中フラグ	バックアップ、リストア、照合の実行中のとき、TRUE となります。	BOOL	R

*1. ユーザプログラムからは、このシステム定義変数にアクセスできません。

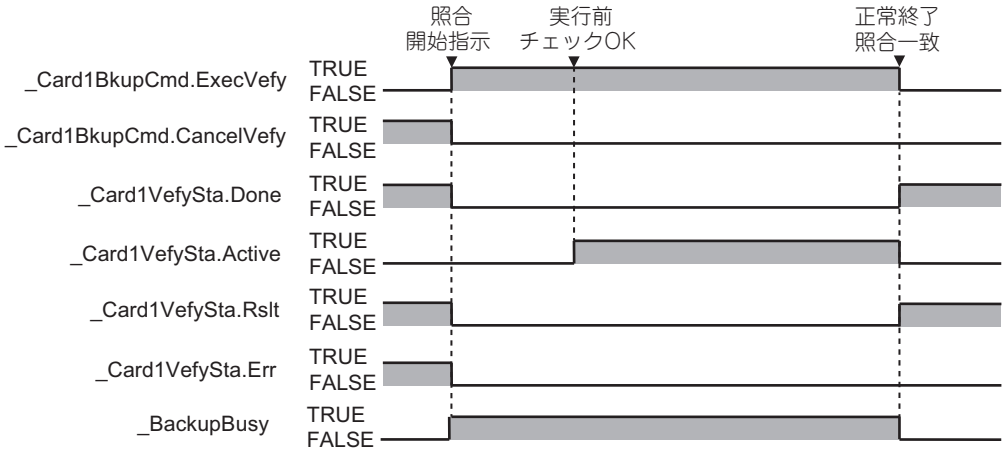
参考

- NYシリーズコントローラとNAシリーズ表示器を接続して使用する場合の変数マッピングについては、🕮『プログラマブルターミナル NA シリーズ ユーザーズマニュアル ソフトウェア編 (SBSA-546)』を参照してください。
- NYシリーズコントローラとNSシリーズ表示器を接続して使用する場合に、CX-Designerの変数テーブルにこれらのシステム定義変数を登録する方法については、🕮「A-7 CX-Designer の変数テーブル登録方法」（P.A-96）を参照してください。

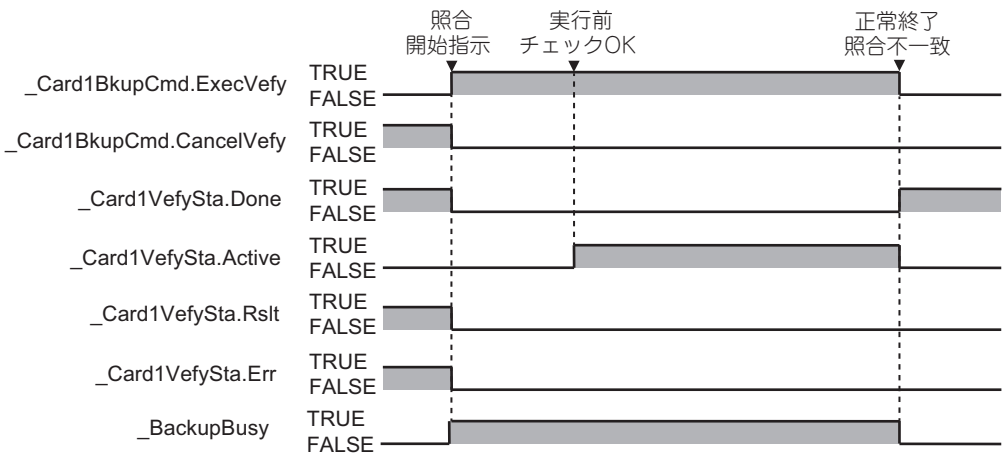
タイミングチャート

システム定義変数変数による方法で照合を行う場合の、システム定義変数の動作は以下のとおりです。図中の「実行前チェック」とは、照合が開始される前に、仮想 SD メモリカードが存在するかどうかなどをチェックする処理のことです。_Card1VefySta.Active（実行中フラグ）の値が TRUE になるのは、実行前チェックの結果が OK となり、実際に照合が開始されたときです。

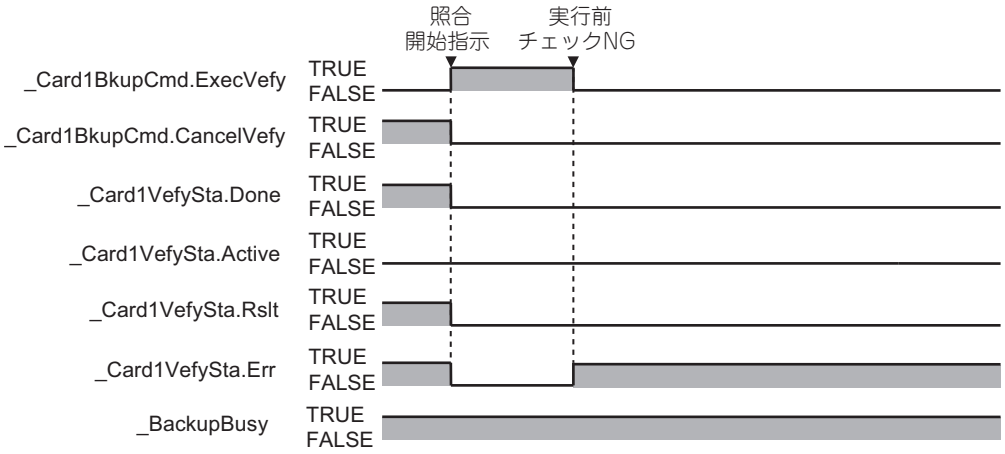
● 正常動作で照合一致の場合



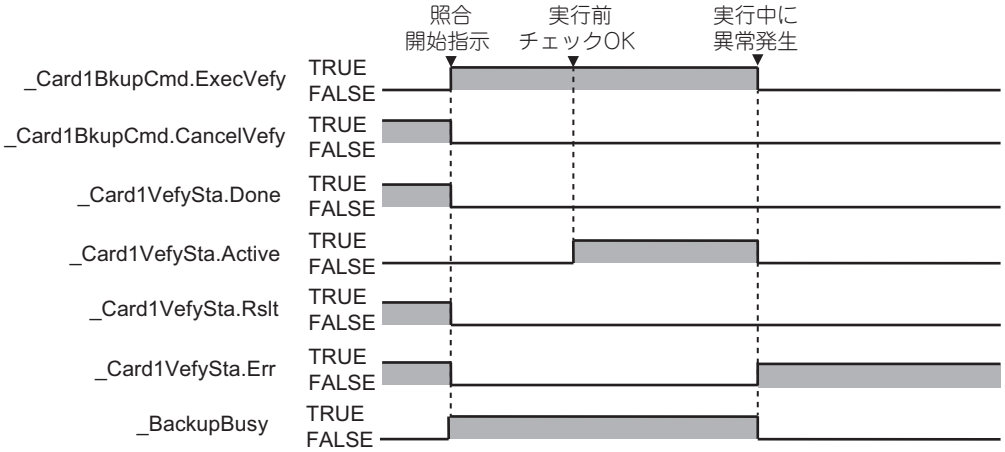
● 正常動作で照合不一致の場合



● 別のバックアップ機能実行中により、照合を開始できなかった場合

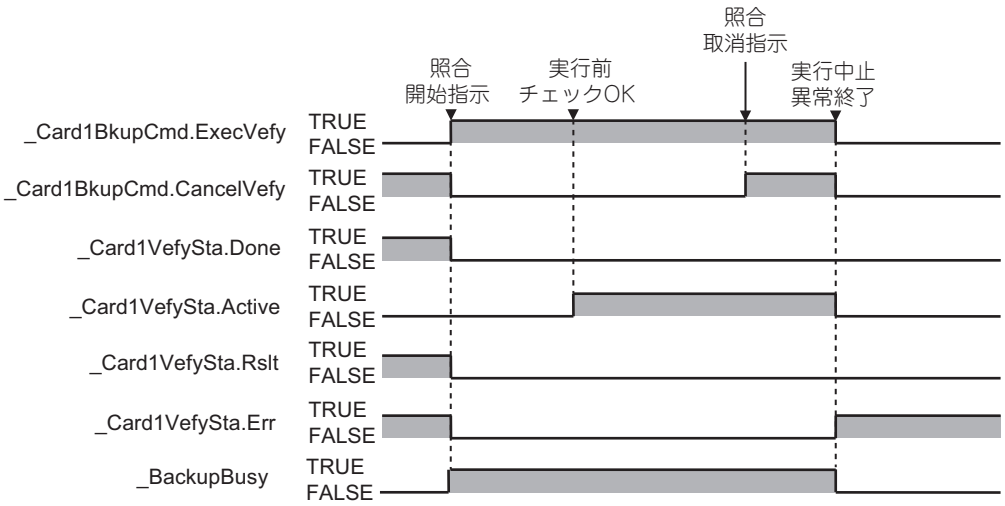


● 正常に照合開始後、何らかの原因により照合に失敗した場合



● 照合実行中に取消指示を出した場合

取消指示を出してから照合実行中止までの時間は、照合処理の進行状況によって変化します。



9-3　SD メモリカードへのバックアップ禁止機能

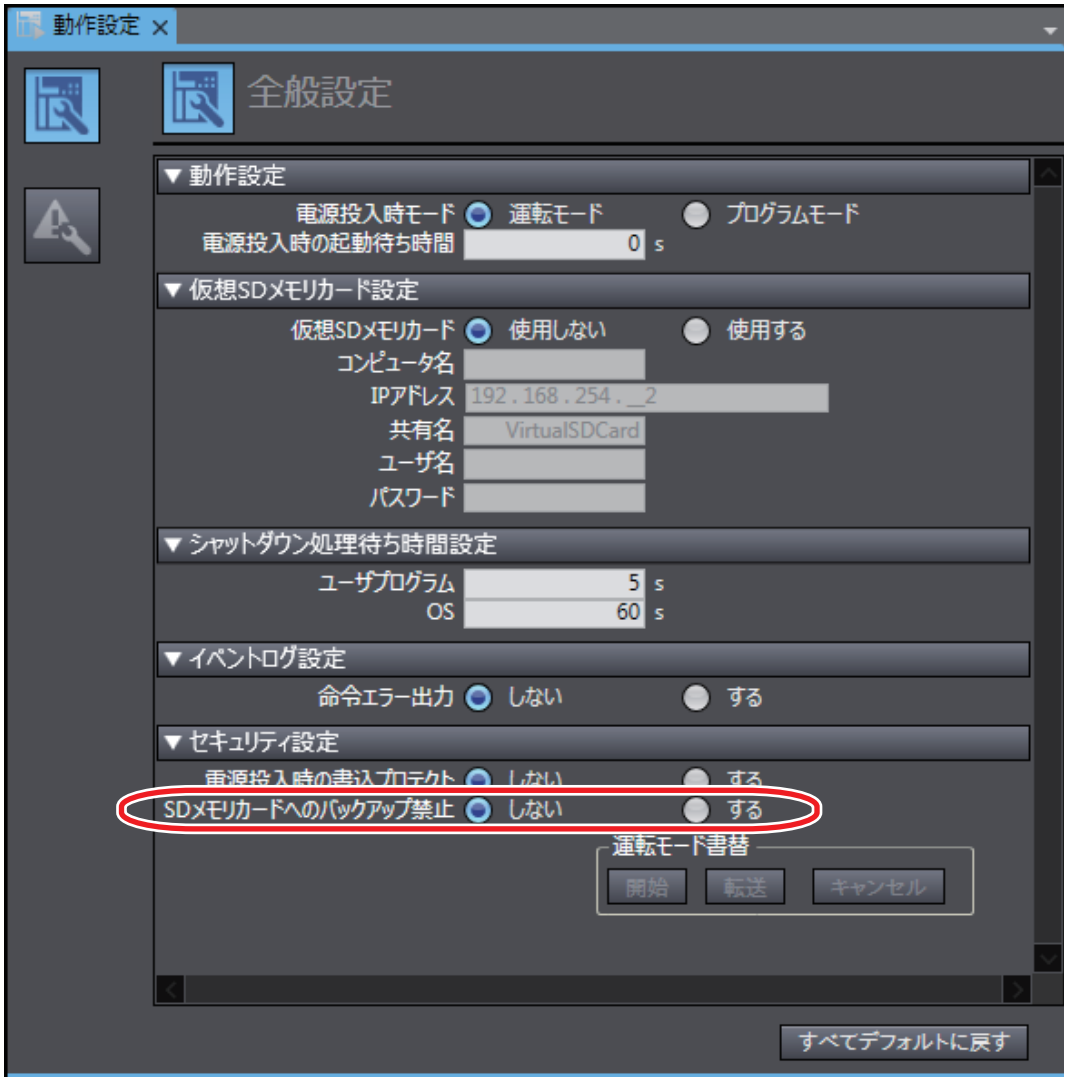
仮想 SD メモリカードへのバックアップを禁止する機能です。この機能を使用することにより、ユーザ資産情報を保護することができます。

SD メモリカードへのバックアップ禁止機能の対象は、以下の 2 つです。

- システム定義変数によるバックアップ
- Sysmac Studio SD メモリカード画面によるバックアップ

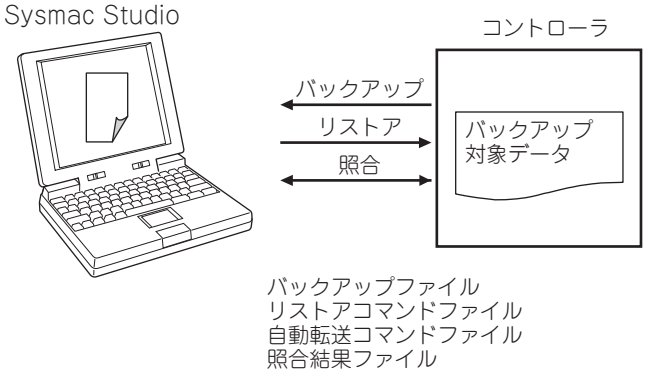
BackupToMemoryCard 命令によるバックアップは、バックアップ禁止機能の対象外です。つまり、SDメモリカードバックアップ禁止設定をしていても、BackupToMemoryCard命令によるバックアップは実行可能です。

SD メモリカードへのバックアップ禁止の操作方法は、以下のとおりです。
Sysmac Studio の [構成・設定]-[コントローラ設定]-[動作設定] の全般設定タブで、SD メモリカードのバックアップ禁止を [する] に設定します。



9-4　Sysmac Studio コントローラバックアップ機能

Sysmac Studio を用いて、パソコンからコントローラのデータのバックアップ、リストア、照合をする機能です。



バックアップを実行すると、「バックアップファイル」「リストアコマンドファイル」「自動転送コマンドファイル」が、パソコンの指定ディレクトリに作成されます。これら、バックアップ関連ファイルの役割は以下のとおりです。

ファイル	役割			
	内容	バックアップ	リストア	照合
バックアップファイル	バックアップ関連機能で扱う、コントローラのデータをファイル化したものです。	作成	参照	参照
リストアコマンドファイル	リストアを実行するときに、転送する対象のデータグループを指定するファイルです。パソコンのテキストエディタでファイルを編集することで転送対象を指定します。	作成	参照	参照
自動転送コマンドファイル	NY シリーズコントローラでは、このファイルを使用しません。	作成	何もしない	何もしない

この機能の実行可能な動作モードは以下のとおりです。

処理	実行可能な動作モード
バックアップ	運転モード、プログラムモード
リストア	プログラムモード
照合	運転モード、プログラムモード

参考

バックアップまたは照合を実行中に、コントローラの動作モードの変更が可能です。
ただし、コントローラのメモリが故障している場合などのように、正常にバックアップまたは照合を実行できないときは、エラーとなります。

9-4-1　バックアップ（コントローラからパソコン）

コントローラのデータを、パソコンの指定ディレクトリに保存します。

処理内容

- バックアップデータ内のユニット･スレーブ設定は､接続しているすべての EtherCAT スレーブだけを選択することができます。
- 各データグループのバックアップ条件は、以下の表のとおりです。

データグループ	バックアップ条件
ユーザプログラム・設定	「CPU ユニット」選択時
内蔵 EtherNet/IP ポートの IP アドレス*1	「CPU ユニット」選択時
変数の現在値	「CPU ユニット」選択時
イベントログ	「CPU ユニット」選択時
ユニット・スレーブ設定	「EtherCAT スレーブ」選択時
絶対値エンコーダ原点位置オフセット	「CPU ユニット」選択時

*1. コントローラのユニットバージョン Ver.1.14 以降と Sysmac Studio Ver.1.18 以降の組み合わせの場合に対象となります。

- バックアップを実行すると、「バックアップファイル」「リストアコマンドファイル」「自動転送コマンドファイル」が、パソコンの指定ディレクトリに作成されます。
- 指定ディレクトリに既存のバックアップ関連ファイルがあるときは、上書きされます。
- 指定ディレクトリにバックアップ関連ファイルを書込中にエラー発生した場合は、既存のバックアップ関連ファイルが削除されます。そして、新しいバックアップ関連ファイルも作成されません。
- 新しいバックアップ関連ファイルの作成前にエラー発生したときは、既存のものが保持され、新しいファイルは作成されません。
- 実行中は､システム定義変数 _BackupBusy（バックアップ関連機能実行中フラグ）の値が TRUE になります。

操作方法

1 **Sysmac Studio の [ツール]–[バックアップ]–[コントローラのバックアップ] を実行します。**

2 **バックアップファイル､リストアコマンドファイル､自動転送コマンドファイルを作成するディレクトリを指定します。**

3 **バックアップ確認画面で、[実行] ボタンを押します。**
バックアップが実行され、バックアップファイル、リストアコマンドファイル、自動転送コマンドファイルが作成されます。

9-4-2 リストア（パソコンからコントローラ）

パソコンの指定ディレクトリ内のバックアップファイルのデータを、コントローラにリストアします。プログラムモードでだけ実行可能です。

処理内容

- パソコンの指定ディレクトリ内のバックアップファイルのデータを、コントローラにリストアします。
- リストアの対象データグループは、Sysmac Studio で選択できます。リストアされるための条件は、以下の表のとおりです。

データグループ	リストア条件
ユーザプログラム・設定	「CPU ユニット」選択時
内蔵 EtherNet/IP ポートの IP アドレス*1	「内蔵 EtherNet/IP ポートの IP アドレス」選択時
変数の現在値	「保持属性の変数の現在値」選択時
ユニット・スレーブ設定	「EtherCAT スレーブ」選択時
絶対値エンコーダ原点位置オフセット	「絶対値エンコーダ原点位置オフセット」選択時

*1. コントローラのユニットバージョン Ver.1.14 以降と Sysmac Studio Ver.1.18 以降の組み合わせの場合に対象となります。

- リストア処理の開始前チェックでエラーが発生した場合は、コントローラ内のデータはリストア前のデータを保持します。
- リストア処理を実行中にコントローラの電源断が発生した場合は、「ユーザプログラム／コントローラ構成・設定転送エラー（全停止フォールトレベルのコントローラ異常）」が発生します。このときコントローラ内のデータは不定となりますので、以下の方法により異常を取り除いてください。
  - リストアを再度実施する。
  - Sysmac Studio でメモリオールクリア、プロジェクトのダウンロードを実施する。
- 保持属性ありの変数の現在値がリストア対象でない場合、その変数の現在値は、リストア前のものを保持します。ただし、変数自身を維持する条件を満たさない場合は、変数の値を初期化します。変数自身を維持する条件は、以下のとおりです。
  - リストア前の変数とリストアする変数の、変数名、データ型名、データ型のサイズがすべて同じである。
- リストア実行完了後は、すべての EtherCAT スレーブの電源を再投入してください。

操作方法

1 **Sysmac Studio の [ツール]–[バックアップ]–[コントローラのリストア] を実行します。**

2 **バックアップファイル、リストアコマンドファイルが保存されているディレクトリを指定します。**

3 **リストア確認画面で、[実行] ボタンを押します。**
リストアが実行されます。

9-4-3　照合（コントローラとパソコン間）

コントローラのデータと、パソコンの指定ディレクトリ内のバックアップファイルのデータを照合します。

処理内容

- コントローラのデータと、パソコンの指定ディレクトリ内のバックアップファイルのデータを照合します。照合の対象データグループは、Sysmac Studio で選択できます。照合されるための条件は、以下の表のとおりです。一括指定をすると、これらのデータがすべて照合されます。

データグループ	照合条件
ユーザプログラム・設定	「CPU ユニット」選択時
内蔵 EtherNet/IP ポートの IP アドレス*1	「内蔵 EtherNet/IP ポートの IP アドレス」選択時
ユニット・スレーブ設定	「EtherCAT スレーブ」選択時

*1. コントローラのユニットバージョン Ver.1.14 以降と Sysmac Studio Ver.1.18 以降の組み合わせの場合に対象となります。

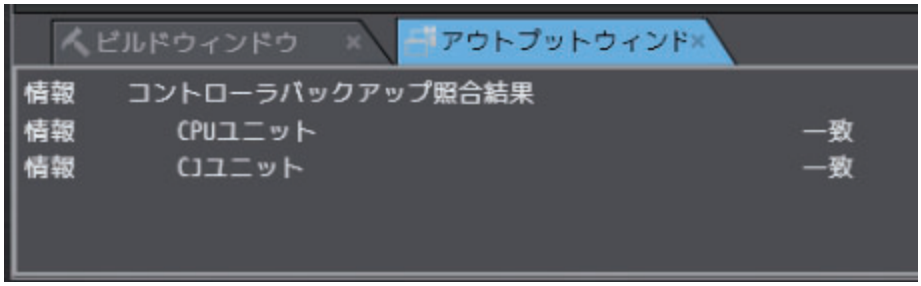
- 照合結果は、Sysmac Studio の画面に表示されます。
- 実行中は、システム定義変数 _BackupBusy（バックアップ関連機能実行中フラグ）の値が TRUE になります。

操作方法

1　**Sysmac Studio の [ツール]-[バックアップ]-[バックアップファイルの照合] を実行します。**

2　**バックアップファイルが保存されているディレクトリを指定します。**

3　**照合確認画面で、[実行] ボタンを押します。**
照合が実行され、バックアップファイルが保存されているディレクトリに、照合結果ファイルが作成されます。また、アウトプットウィンドウに、照合結果が表示されます。



9-5　Sysmac Studio バックアップファイルのインポート / エクスポート機能

コントローラを使用せず、Sysmac Studio のプロジェクトから、パソコンの指定ディレクトリへのバックアップファイルの作成および読出をする機能です。
対象データは、以下のとおりです。

機能		データグループ				
		ユーザプログラム・設定		変数の現在値	ユニット・スレーブ設定	絶対値エンコーダ原点位置オフセット
			内蔵 EtherNet/IP ポートの IP アドレス*1			
Sysmac Studio バックアップファイルのインポート / エクスポート機能	バックアップファイルのエクスポート	○*2	○	×	×	×
	バックアップファイルのインポート	○*3	○	×	○	×

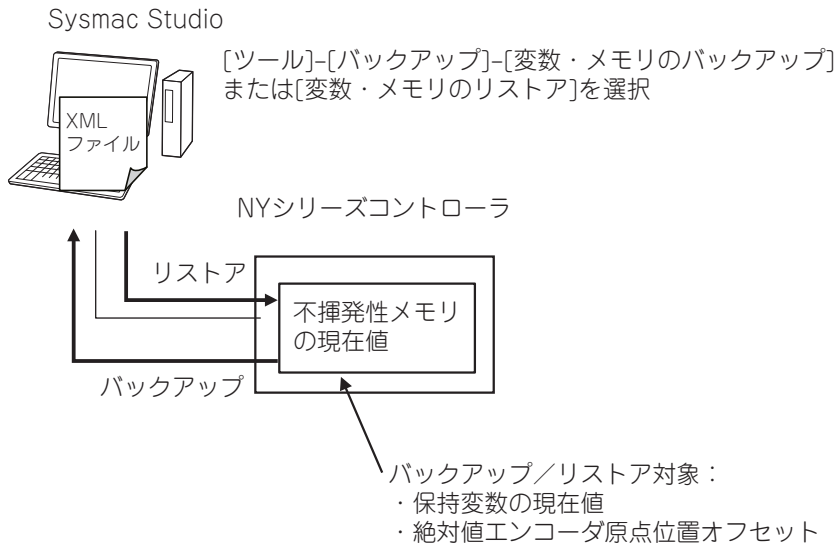
*1. コントローラのユニットバージョン Ver.1.14 以降と Sysmac Studio Ver.1.18 以降の組み合わせで、内蔵 EtherNet/IP ポート設定の IP アドレスは、1 つのデータグループとして使用可能です。
　上記以外の組み合わせでは、IP アドレスはユーザプログラム・設定に含まれています。
*2. 以下のデータは、処理の対象外です。
- コントローラ名称のうちの内蔵 EtherNet/IP ポート名
- コントローラ設定のうちの内蔵 EtherNet/IP 用タグデータリンク設定
- 操作権限認証
- データトレース設定

*3. 以下のデータは、処理の対象外です。
- コントローラ名称のうちの内蔵 EtherNet/IP ポート名
- 操作権限認証
- データトレース設定

この機能の詳細は、🕮『Sysmac Studio Version 1 オペレーションマニュアル（SBCA-470）』を参照してください。

9-6　Sysmac Studio 変数／メモリのバックアップ機能

NY シリーズコントローラ内の不揮発性メモリの現在値を、パソコンにファイル（XML 形式）としてバックアップ（読出）、またパソコンからコントローラに対してリストア（書込）することができます。ここでは、Sysmac Studio 変数／メモリのバックアップ機能の、対象データ、操作方法およびコントローラ形式間の互換性について説明します。



9-6-1　Sysmac Studio 変数／メモリのバックアップ機能の対象データ

Sysmac Studio 変数／メモリのバックアップ機能は、以下のデータを対象としています。

- 保持属性の変数の現在値
- 絶対値エンコーダ原点位置オフセット

9-6-2　Sysmac Studio 変数／メモリのバックアップ機能の操作方法

Sysmac Studio での操作方法は、以下のとおりです。
オンライン接続状態で、［ツール］メニューから［バックアップ］-［変数・メモリの バックアップ］または［変数・メモリのリストア］を選択します。

詳細は、🕮『Sysmac Studio Version 1 オペレーションマニュアル（SBCA-470）』を参照してください。

9-6-3　NY シリーズコントローラの形式間の互換性

Sysmac Studio 変数／メモリのバックアップ機能は、バックアップ元とリストア先の NY シリーズの形式が異なっていても、リストアが可能です。

9-7　EtherCAT スレーブ接続時のバックアップ機能

EtherCAT スレーブは、SD メモリカードバックアップ機能、Sysmac Studio コントローラバックアップ機能、および Sysmac Studio バックアップのインポート機能の対象です。
ここでは、EtherCAT スレーブを接続している場合の、バックアップ対象データ、コントローラの各状態でのバックアップ機能実行可否、リストア実行の条件、EtherCAT スレーブの機種ごとの注意事項について説明します。

参考

バックアップ機能の対象が　EtherCAT　スレーブターミナルの場合の詳細は、🕮「9-8 EtherCAT スレーブターミナル接続時のバックアップ機能」（P.9-31）を参照してください。

9-7-1　EtherCAT スレーブのバックアップ対象データ

EtherCAT スレーブのバックアップの対象となるデータは、以下のとおりです。

設定	バックアップ対象データ
EtherCAT マスタ設定	形式名、製品名称、スレーブ数、PDO 通信周期、縮退運転設定、全スレーブ加入待ち時間、PDO 通信連続タイムアウト検出回数、リビジョンチェック基準、シリアル番号チェック基準
EtherCAT スレーブ設定	デバイス名、形式名、製品名称、リビジョン、ノードアドレス、有効 / 無効設定、シリアル番号、PDO マッピング設定、DC 有効、リファレンスクロック機能、初期化パラメータ設定

9-7-2　コントローラの各状態でのバックアップ機能実行可否

コントローラの各状態での、EtherCAT スレーブのバックアップ、リストア、照合の実行可否の仕様は、以下のとおりです。

コントローラの状態		実行可否		
		バックアップ	リストア	照合
リンクオフ中		不可*1	不可*2	可*3
マスタ状態不正*4		不可 *1	不可 *2	可 *3
構成情報とのネットワーク構成不一致*5		不可 *1	不可 *2	可 *3
バックアップ時とのネットワーク構成不一致		可	不可 *2	可 *3
ネットワーク構成に無効設定のスレーブが存在	実構成にスレーブ存在	可	可	可
	実構成にスレーブ不在	可	可	可
ネットワーク構成に「離脱」指示により離脱状態になったスレーブが存在	実構成にスレーブ存在	不可 *1	可 離脱状態のスレーブも対象	可 離脱状態のスレーブも対象
	実構成にスレーブ不在	不可 *1	不可 *2	可 *3
スレーブ初期化異常		不可 *1	不可 *2	可 *3

*1. イベントログに、バックアップ実行失敗 (EtherCAT スレーブ) が記録されます。
*2. イベントログに、リストア実行失敗 (EtherCAT スレーブ) が記録されます。
*3. 照合結果が、不一致になります。
*4.「スレーブノードアドレス重複」、「ネットワーク構成情報異常」、「ネットワーク構成異常」、「スレーブ初期化異常」、「縮退運転設定を " 停止設定する " に設定している時のネットワーク構成照合異常」、「リンクオフ異常」のことです。
*5.「バックアップ時とネットワーク構成が完全に一致していない（分岐スレーブ上の接続ポート違いは不一致と見なす）」、「ネットワーク構成情報とネットワーク構成が一致していない（分岐スレーブ上の接続ポート違いは一致と見なす」のことです。

9-7-3　EtherCAT スレーブのリストア実行条件

EtherCAT スレーブのバックアップデータをリストアするためには、以下の条件を満たしていなければなりません。

- バックアップファイル内に EtherCAT スレーブのデータが存在している。
- ネットワーク構成情報と、リストア先の実ネットワーク構成が一致している。
- EtherCAT スレーブにあらかじめ設定されているリビジョン値が一致している。ただし、一致条件は、バックアップファイル内に格納されている「リビジョンチェック基準」に従います。また、「リビジョンチェック基準」が「チェックしない」に設定されていても、「設定リビジョン」＞「実機リビジョン」の場合は、リストア不可になります。また、リビジョン値は、ユーザが変更することはできません。
- バックアップファイル内に格納されている「シリアル番号照合の有無」が「照合する」に設定されている場合、シリアル番号が一致している。
- ハードウェアスイッチでノードアドレスが設定されている場合、ノードアドレスが一致している。

使用上の注意

- リストア実行完了後は、すべての EtherCAT スレーブの電源を再投入してください。
- リストア実行完了後は、全スレーブが離脱状態となります。対象スレーブの離脱状態を解除するには、対象スレーブへの再加入指示が必要です。
- バックアップファイルに格納されている「シリアル番号照合の有無」が「照合する」に設定されている場合、EtherCAT スレーブのハードウェア交換を行うと、リストアできなくなります。その場合は、Sysmac Studio でネットワーク構成を変更し、ダウンロード、およびスレーブパラメータの転送を行って原状回復してください。また、ノードアドレスをハードウェアスイッチで設定している場合は、バックアップ時と同じ設定としてください。

9-7-4　バックアップ機能の対象 EtherCAT スレーブ

バックアップ機能の対象となる EtherCAT スレーブを、以下に示します。注意事項に気をつけてお使いください。

EtherCAT スレーブの機種	注意事項
NX シリーズ EtherCAT カプラユニット 形 NX-ECC	EtherCAT スレーブターミナル上の、セーフティコントロールユニットは、バックアップ、リストア、照合の対象外です。セーフティコントロールユニットの設定データのインポート／エクスポート機能は、📖『NX シリーズ セーフティコントロールユニット ユーザーズマニュアル（SGFM-710）』を参照してください。
AC サーボドライバ 形 R88D-1SN □□□ -ECT 形 R88D-1SAN □□□ -ECT	*1
AC サーボドライバ（回転型） 形 R88D-KN □□□ -ECT	*1*2
AC サーボドライバ（リニア型） 形 R88D-KN □□□ -ECT-L	*1*2
インバータ 形 3G3AX-MX2-ECT 形 3G3AX-RX-ECT	インバータのパラメータはリストアの対象外です。後述📖「 インバータ形 3G3AX-MX2-ECT、および形 3G3AX-RX-ECT のパラメータの書き込み方法」（P.9-30）で、インバータに Sysmac Studio からのパラメータを書き込んでください。 パラメータを書き込まずに照合を実行すると、インバータについては、EtherCAT スレーブ照合結果ファイル内で、「Result=Not matched」となる可能性があります。
視覚センサ 形 FH-3 □□□□ 形 FH-1 □□□□	センサの設定データ（シーンデータ、システムデータなど）は、バックアップ、リストア、照合の対象外です。 センサの設定データを外部ファイルに転送したり、センサに転送したりするには、Sysmac Studio の構成・設定のエディット画面から、[ツール]-[センサデータ]-[ファイルへ保存] または [ファイルから読込] を実行してください。 詳細は、📖『画像処理システム FH/FZ5 シリーズ ユーザーズマニュアル（SDNB-712）』を参照してください。
視覚センサ 形 FQ-M □□□□ -ECT 形 FQ-M □□□□ -M-ECT	センサの設定データ（シーンデータ、システムデータなど）は、バックアップ、リストア、照合の対象外です。 センサの設定データを外部ファイルに転送したり、センサに転送したりするには、Sysmac Studio の構成・設定のエディット画面から、[ツール]-[センサデータ]-[ファイルへ保存] または [ファイルから読込] を実行してください。 詳細は、📖『視覚センサ FQ-M シリーズ 位置決め専用視覚センサ ユーザーズマニュアル（SDNB-706）』を参照してください。
視覚センサ 形 FZM1- □□□ -ECT	センサの設定データ（シーンデータ、システムデータなど）は、バックアップ、リストア、照合の対象外です。 センサの設定データを USB メモリに保存したり、コントローラに読み込んだりするには、専用ツールをお使いください。 詳細は、📖『視覚センサ FZ3 シリーズ ユーザーズマニュアル（SCHB-762）』を参照してください。
デジタル I/O ターミナル 形 GX- □ D16 □□ 形 GX- □ D32 □ 8 形 GX-OC1601	*1
アナログ I/O ターミナル 形 GX-AD0471 形 GX-DA0271	*1
エンコーダ入力ターミナル 形 GX-EC0211 形 GX-EC0241	ー

EtherCAT スレーブの機種	注意事項
EtherCAT 分岐スレーブ 形 GX-JC0 □	バックアップする必要があるデータを、内蔵していません。
ファイバ同軸変位センサ 形 ZW-CE1 □	すべての設定データが、バックアップ、リストア、照合の対象外です。 設定データを保存したり、コントローラに読み込んだりする方法は、🕮『変位センサ ZW シリーズ ファイバ同軸変位センサ ユーザーズマニュアル（SDNE-703）』を参照してください。
デジタルタイプセンサ 形 E3NW-ECT 形 E3X-ECT	センサのパラメータは、バックアップ、リストア、照合の対象外です。
他社製スレーブ	• ESI で正しく定義されたデータだけが、バックアップ、リストア、照合の対象です。ESI で定義されていないデータをバックアップ、リストア、照合する場合は、別途専用ツールを使用してください。 • バックアップ、リストア、照合に失敗した場合は、対処方法をスレーブのメーカにお問い合わせください。

*1. リストア実行完了後は、スレーブの電源を再投入してください。リストア後の照合は、スレーブの電源を再投入した後に実行してください。電源を再投入する前に照合を実行すると、照合不一致になります。
*2. 以下のいずれかの条件に該当する場合は、バックアップから照合までの間、またはリストアから照合までの間に、サーボオン状態にしないでください。バックアップから照合までの間、またはリストアから照合までの間にサーボオン状態にすると、照合前にパラメータが更新され、照合不一致になる可能性があります。
- リアルタイムオートチューニングモードが有効に設定されている（オブジェクト 3002h の値が、1 ～ 4 または 6）。
- 適応フィルタが有効に設定されている（オブジェクト 3200h の値が 1 または 2）。

インバータ形 3G3AX-MX2-ECT、および形 3G3AX-RX-ECT のパラメータの書き込み方法

インバータのパラメータはリストアの対象外です。
バックアップしたパラメータをインバータに書き込むには、Sysmac Studio を以下のように操作してください。詳細は、🕮『Sysmac Studio Version 1 オペレーションマニュアル（SBCA-470）』を参照してください。

1 **Sysmac Studio バックアップファイルのインポート機能を使って、バックアップファイル内のインバータのパラメータを読み出します。**
Sysmac Studio の構成・設定のインバータのパラメータタブに、インバータのパラメータが表示されます。

2 **読み出したパラメータのインバータの形式と、実際に接続しているインバータの形式が、一致していることを確認します。**

3 **構成・設定のインバータのパラメータタブの転送[パソコン→ドライブ]機能を使って、パラメータをインバータに書き込みます。**

使用上の注意

インバータ形 3G3AX-MX2-ECT の、インバータモード選択（パラメータ No.b171）機能を使用している場合は、パラメータを書き込む前に、接続しているインバータのモードを、バックアップ時のモードに切り替えてください。また、モード切替を有効にするためには、モード切替操作の後に、インバータの初期化が必要です。

9-8　EtherCAT スレーブターミナル接続時のバックアップ機能

EtherCAT スレーブターミナルは、SD メモリカードバックアップ機能、Sysmac Studio コントローラバックアップ機能、および Sysmac Studio バックアップのインポート機能の対象です。
ここでは、EtherCAT スレーブターミナルを接続している場合の、バックアップ対象データ、コントローラの各状態でのバックアップ機能実行可否、リストア実行の条件について説明します。

使用上の注意

EtherCAT スレーブターミナル上のセーフティコントロールユニットは、バックアップ、リストア、照合の対象外です。セーフティコントロールユニットの設定データのインポート／エクスポート機能は、🕮『NX シリーズ セーフティコントロールユニット ユーザーズマニュアル（SGFM-710）』を参照してください。

9-8-1　EtherCAT スレーブターミナルのバックアップ対象データ

EtherCAT スレーブターミナルのバックアップ対象データは、EtherCAT カプラユニットと NX ユニットで異なります。それぞれのバックアップ対象データは、以下のとおりです。

（○：対象　×：対象外）

ユニット	データ	バックアップ	リストア	照合
EtherCAT カプラユニット	構成情報*1	○	○	○
	ユニット動作設定	○	○	○
NX ユニット	構成情報*1	○	○	○
	ユニット動作設定	○	○	○
	ユニットアプリケーションデータ*2	○	○	○

*1. ユニット構成情報および I/O 割付情報の総称です。
*2. 各 NX ユニットに固有のデータのことです。ユニットアプリケーションデータをもたない NX ユニットもあります。

使用上の注意

バックアップ元と同じユニット構成の EtherCAT スレーブターミナルに、バックアップデータをリストアする場合、ハードウェアスイッチをバックアップ元と同じ設定にしてください。ハードウェアスイッチ設定がバックアップデータと異なる場合は、リストアできません。監視情報のイベントである「リストア開始失敗（EtherCAT スレーブ）」が発生します。

9-8-2 EtherCAT スレーブターミナルの各状態でのバックアップ機能実行可否

EtherCAT スレーブターミナルの各状態での、バックアップ、リストア、照合の実行可否の仕様は、以下のとおりです。

EtherCAT スレーブターミナルの状態	実行可否		
	バックアップ	リストア	照合
ユニット構成情報自動生成時	可*1	可*2	可
NX ユニット加入待ち	不可*3	不可*4	可*5
EtherCAT カプラユニットまたは NX ユニットで、ウォッチドッグタイマ異常発生中	不可*3	不可*4	可*5
バスコントローラ異常発生中	不可*3	不可*4	可*5
ユニット構成情報異常発生中	不可*3	可	可*5
ユニット構成照合異常発生中	不可*3	可	可*5
バックアップ時とのユニット構成情報不一致	ー	不可*4	可*5

*1.「ユニット構成情報が存在しない」という情報がバックアップされます。
*2. リストア後も、ユニット構成自動生成の状態になります。
*3. イベントログに、バックアップ実行失敗が記録されます。
*4. イベントログに、リストア実行失敗が記録されます。
*5. 照合結果が、不一致になります。

9-8-3 EtherCAT スレーブターミナルのリストア実行条件

EtherCAT スレーブターミナルのバックアップデータをリストアするためには、以下の条件を満たしていることが必要です。

- バックアップファイル内に、EtherCATカプラユニットおよびNXユニットのデータが存在している。
- バックアップ元のユニット構成情報と、リストア先の実際のユニット構成が一致している。
- バックアップ元の EtherCAT カプラユニットのシリアル番号と、リストア先の EtherCAT カプラユニットのシリアル番号が一致している。ただし、バックアップファイル内の〔通信カプラユニット〕|〔ユニット動作設定〕|〔シリアル番号チェック基準〕の設定が、「設定値＝実機」の場合に限る。
- バックアップ元の NX ユニットのシリアル番号と、リストア先の NX ユニットのシリアル番号が一致している。ただし、バックアップファイル内の〔通信カプラユニット〕|〔ユニット動作設定〕|〔シリアル番号チェック基準〕の設定が、「設定値＝実機」の場合に限る。
- バックアップ元の EtherCAT カプラユニットのハードウェアスイッチ情報と、リストア先の EtherCAT カプラユニットのハードウェアスイッチ設定が完全に一致している。
- バックアップ元の EtherCAT カプラユニットのユニットバージョン設定値が、リストア先の EtherCAT カプラユニットの実機ユニットバージョンと一致している、または古い。
- バックアップ元の NX ユニットのユニットバージョン設定値が、リストア先の NX ユニットの実機ユニットバージョンと一致している、または古い。

9-9　バックアップ関連ファイル

ここでは、バックアップ関連ファイルの仕様について説明します。
バックアップ関連ファイルは、「Sysmac Studio 変数／メモリのバックアップ機能」以外のバックアップ機能で使用するファイルです。

9-9-1　バックアップ関連ファイルの種類

バックアップ関連ファイルには、バックアップファイル、リストアコマンドファイル、自動転送コマンドファイル、照合結果ファイルの 4 種類があります。

● バックアップファイル

バックアップ関連機能で扱うコントローラのデータをファイル化したものです。バックアップを実行したときに作成されます。

● リストアコマンドファイル

仮想 SD メモリカードからリストアを実行するときに、転送する対象のデータグループを指定するファイルです。パソコンのテキストエディタでファイルを編集することで転送対象を指定します。バックアップを実行したときに作成されます。

● 自動転送コマンドファイル

NY シリーズコントローラでは、このファイルを使用しません。

● 照合結果ファイル

コントローラのデータと仮想 SD メモリカードのバックアップファイルのデータとの照合結果を格納するファイルです。
照合結果ファイルには、以下の 3 種類があります。いずれも、SD メモリカードバックアップ機能の、照合を実行したときに作成されます。

照合結果ファイル	内容
コントローラ照合結果ファイル	リストアコマンドファイルで指定された、すべてのバックアップデータの照合結果が記述されたファイルです。
EtherCAT スレーブ照合結果ファイル	EtherCAT スレーブごとの照合結果が記述されたファイルです。リストアコマンドファイルで、ユニット・スレーブ設定をリストア対象とし、かつ、バックアップファイル内に EtherCAT スレーブ設定が含まれている場合に作成されます。
EtherCAT スレーブターミナル照合結果ファイル	EtherCAT カプラユニット、および NX ユニットごとの照合結果が記述されたファイルです。以下の 3 つのすべての条件に該当する場合に作成されます。 • リストアコマンドファイルで、ユニット・スレーブ設定をリストア対象としている。 • バックアップファイル内に EtherCAT スレーブ設定が含まれている。 • EtherCAT スレーブターミナルを接続している。 EtherCAT スレーブターミナル照合結果ファイルが作成される場合は、必ず EtherCAT スレーブ照合結果ファイルも同時に作成されます。

9-9-2　バックアップファイルの仕様

バックアップファイルの、ファイル名、作成タイミング、作成ディレクトリは、以下のとおりです。

ファイル名

NY シリーズコントローラのバックアップファイルのファイル名は、次のとおりです。

ファイル	ファイル名
バックアップファイル	NYBackup.dat

ファイル作成タイミングと作成ディレクトリ

機能	操作方法	作成タイミング	作成ディレクトリ
SD メモリカードバックアップ機能	システム定義変数	バックアップ実行時	仮想 SD メモリカード内の、システム定義変数で指定したディレクトリ
	Sysmac Studio SD メモリカード画面	バックアップ実行時	仮想 SD メモリカード内の、画面でユーザが指定したディレクトリ
	専用命令	バックアップ実行時	仮想 SD メモリカード内の、BackupToMemoryCard 命令の入力変数でユーザが指定したディレクトリ
SysmacStudio コントローラバックアップ機能	Sysmac Studio コントローラバックアップ画面	バックアップ実行時	パソコン内の、画面でユーザが指定したディレクトリ
Industrial PC Support Utility コントローラバックアップ機能	Industrial PC Support Utility コントローラバックアップ画面	バックアップ実行時	産業用 PC 内の、画面でユーザが指定したディレクトリ
SysmacStudio バックアップファイルのインポート / エクスポート機能	Sysmac Studio バックアップファイルのエクスポート画面	エクスポート実行時	パソコン内の、画面でユーザが指定したディレクトリ

9-9-3　リストアコマンドファイルの仕様

リストアコマンドファイルの、ファイル名、作成タイミング、作成ディレクトリ、データグループの指定方法は、以下のとおりです。

ファイル名

ファイル	ファイル名
リストアコマンドファイル	RestoreCommand.ini

ファイル作成タイミングと作成ディレクトリ

機能	操作方法	作成タイミング	作成ディレクトリ
SD メモリカードバックアップ機能	システム定義変数	バックアップ実行時	バックアップファイルと同じディレクトリ
	Sysmac Studio SD メモリカード画面	バックアップ実行時	バックアップファイルと同じディレクトリ
SysmacStudio コントローラバックアップ機能	Sysmac Studio コントローラバックアップ画面	バックアップ実行時	バックアップファイルと同じディレクトリ
Industrial PC Support Utility コントローラバックアップ機能	Industrial PC Support Utility コントローラバックアップ画面	バックアップ実行時	バックアップファイルと同じディレクトリ
SysmacStudio バックアップファイルのインポート / エクスポート機能	Sysmac Studio バックアップファイルのエクスポート画面	エクスポート実行時	バックアップファイルと同じディレクトリ

リストアするデータグループの指定方法

リストアコマンドファイルは、データグループごとにリストア指定をするためのファイルです。
データグループごとの指定をしている部分をパソコンのテキストエディタで編集することで、指定を変更できます。
たとえば、下表の「ファイル内容」8 行めの「Variable=yes」を「Variable=no」に変更すると、「変数の現在値」がリストア対象から外されます。

ファイル内容（ファイル作成時のデフォルト）	説明
[Restore] ; --- User Program and Configuration. --- ; Always select "yes". UserProgram=yes	ユーザプログラム・設定 yes 固定：リストア必須
; --- IP Address of Built-in EtherNet/IP Port Settings. --- ; "yes":will be restored, "no":will not be restored IPAdr=yes	内蔵 EtherNet/IP ポートの IP アドレス yes/no：リストアする / リストアしない
; --- Present values of variables (Retained variables only). --- ; "yes":will be restored, "no":will not be restored Variable=yes	変数の現在値（保持属性のみ） yes/no：リストアする / リストアしない
:---Unit/Slave Parameters.--- ; "yes";will be restored."no";will not be restored UnitConfig=yes	ユニット・スレーブ設定 yes/no：リストアする / リストアしない
; --- Absolute encoder home offset. --- ; "yes":will be restored, "no":will not be restored AbsEncoder=yes	絶対値エンコーダ原点位置オフセット yes/no：リストアする / リストアしない

(注) 1 この表のファイル内容は、リストアコマンドファイル作成時のものです。記述されているすべてのデータグループがリストア対象となっています。
2 バックアップファイル作成時にバックアップファイル内にあり、かつリストア可能なデータグループの指定記述をしたファイルが、記述されます。
3 すべて半角英数字です。また、大文字と小文字は区別されません。
4 Ver.1.12 のコントローラでバックアップした場合、「内蔵 EtherNet/IP ポート設定の IP アドレス」のエントリは作成されません。
Ver.1.14 以降のコントローラで、「内蔵 EtherNet/IP ポート設定の IP アドレス」のエントリが作成されていないリストアコマンドファイルを使用すると、「IPAdr=yes」として動作します。
Ver.1.12 のリストアコマンドファイルと Ver.1.14 以降のリストアコマンドファイルの互換性については、「 リストアコマンドファイルの互換性」（P.9-37）を参照してください。

使用上の注意

リストアコマンドファイルを編集する場合は、選択可能なデータグループの「yes/no」の記述だけにしてください。それ以外の記述を編集してリストアすると、コントローラが意図しない動作をする場合があります。

リストアコマンドファイルの互換性

Ver.1.12 のリストアコマンドファイルと Ver.1.14 以降のリストアコマンドファイルの互換性を示します。

リストアコマンドファイルとバックアップファイルを作成したコントローラのユニットバージョン	リストア先のコントローラのユニットバージョン	
	Ver.1.12	Ver.1.14 以降
Ver.1.12	リストア可 「内蔵 EtherNet/IP ポート設定の IP アドレス」はリストアされる	リストア可 「内蔵 EtherNet/IP ポート設定の IP アドレス」はリストアされる
Ver.1.14 以降	リストア不可 Ver.1.12 のコントローラでは、「内蔵 EtherNet/IP ポート設定の IP アドレス」のエントリを解釈できないため、監視情報の「リストア開始失敗」が発生。このときの異常の詳細は、「0104Hex：リストアコマンドファイルの記述内容が不正」となる	リストア可 「内蔵 EtherNet/IP ポート設定の IP アドレス」は、IPAdr の yes/no に従う

9-9-4　コントローラ照合結果ファイルの仕様

コントローラ照合結果ファイルの、ファイル名、作成タイミング、作成ディレクトリ、照合結果の確認方法は、以下のとおりです。

ファイル名

ファイル	ファイル名
コントローラ照合結果ファイル	VerifyResult.log

ファイル作成タイミングと作成ディレクトリ

機能	操作方法	作成タイミング	作成ディレクトリ
SD メモリカードバックアップ機能	Sysmac Studio SD メモリカード画面	照合実行時	バックアップファイルと同じディレクトリ
	システム定義変数	照合実行時	バックアップファイルと同じディレクトリ

照合結果の確認方法

照合結果ファイルは、データグループごとにコントローラのデータと仮想 SD メモリカードのバックアップファイルのデータとの照合結果を格納してあります。
データグループごとの確認結果を格納している部分で照合結果を確認できます。
「Result=Matched」と書かれていれば照合一致、「Result=Not matched」と書かれていれば照合不一致です。
ファイル内容が以下の場合、ユーザプログラム・設定が照合一致、ユニット / スレーブ設定が照合不一致です。

ファイル内容	説明
[UserProgram] ; --- User Program and Configuration. --- Result=Matched [UnitConfig] ; --- Unit/Slave Parameters. --- Result=Not matched	ユーザプログラム・設定 　Matched：照合一致、Not matched：照合不一致 ユニット / スレーブ設定 　Matched：照合一致、Not matched：照合不一致

1　照合実施したデータグループの照合結果だけが記述されます。
2　「内蔵 EtherNet/IP ポート設定の IP アドレス」の照合結果は、Ver.1.14 以降のコントローラでも、「ユーザプログラム・設定」のエントリに含まれます。

9-9-5　EtherCAT 照合結果ファイルの仕様

EtherCAT 照合結果ファイルの、ファイル名、作成タイミング、作成ディレクトリ、照合結果の確認方法は、以下のとおりです。

ファイル名

ファイル	ファイル名
EtherCAT 照合結果ファイル	VerifyResult_ECAT.log

ファイル作成タイミングと作成ディレクトリ

機能	操作方法	作成タイミング	作成ディレクトリ
SD メモリカードバックアップ機能	Sysmac Studio SD メモリカード画面	照合実行時	バックアップファイルと同じディレクトリ
	システム定義変数	照合実行時	バックアップファイルと同じディレクトリ

照合結果の確認方法

照合結果ファイルは、データグループごとにコントローラのデータと仮想 SD メモリカードのバックアップファイルのデータとの照合結果を格納してあります。
EtherCAT スレーブごとの確認結果を格納している部分で照合結果を確認できます。
「Result=Matched」と書かれていれば照合一致、「Result=Not matched」と書かれていれば照合不一致です。
以下のファイル内容の場合、照合結果はつぎのようになります。

- 照合一致　　：EtherCAT スレーブ Master、EtherCAT スレーブターミナル E002
- 照合不一致　　：EtherCAT スレーブ E001

ファイル内容	説明
[Verification Results] ; --- EtherCAT Parameters. --- : --- See the VerifyResult_ECAT_NX.log about detail result if NX mark is included in square brackets. [Master] Result=Matched [E001] Result=Not matched Factor=Verification error [E002:NX] Result=Matched	スレーブは、ユーザが設定したデバイス名で示します。 EtherCAT スレーブターミナルの場合は、デバイス名の後ろに ":NX" が付きます。*1 照合結果は、以下の表記で示します。 Result=Matched　　照合一致 Result=Not matched　照合不一致

*1. EtherCAT スレーブターミナルが照合対象である場合、EtherCAT スレーブターミナル照合結果ファイルが作成されます。EtherCAT スレーブターミナルの照合結果の詳細は、EtherCAT スレーブターミナル照合結果ファイルに記述されます。

（注） 照合実施した EtherCAT スレーブの照合結果だけが記述されます。

9-9-6 EtherCAT スレーブターミナル照合結果ファイルの仕様

EtherCAT スレーブターミナル照合結果ファイルの、ファイル名、作成タイミング、作成ディレクトリ、照合結果の確認方法は、以下のとおりです。

ファイル名

ファイル	ファイル名
EtherCAT スレーブターミナル照合結果ファイル	VerifyResult_ECAT_NX.log

ファイル作成タイミングと作成ディレクトリ

機能	操作方法	作成タイミング	作成ディレクトリ
SD メモリカードバックアップ機能	Sysmac Studio SD メモリカード画面	照合実行時	バックアップファイルと同じディレクトリ
	システム定義変数	照合実行時	バックアップファイルと同じディレクトリ

照合結果の確認方法

照合結果ファイルは、データグループごとにコントローラのデータと仮想 SD メモリカードのバックアップファイルのデータとの照合結果を格納してあります。
EtherCAT カプラユニットおよび NX ユニットごとの確認結果を格納している部分で、照合結果を確認できます。
「Result=Matched」と書かれていれば照合一致、「Result=Not matched」と書かれていれば照合不一致です。
以下のファイル内容の場合、照合結果はつぎのようになります。

- 照合一致　　：EtherCAT カプラユニット E002、NX ユニット N1、NX ユニット N2
- 照合不一致　　：EtherCAT カプラユニット E005、NX ユニット N3

ファイル内容	説明
[Verification Results] ; --- NX Parameters. --- [E002:UnitNo.0 NX-ECC201] Result=Matched [N1:UnitNo.1 NX-AD2203] Result=Matched [N2:UnitNo.2 NX-DA2203] Result=Matched [N3:UnitNo.3 NX-TS3201] Result=Not matched Factor=Verification error [E005:UnitNo.0 NX-ECC201] Result=Not matched Factor=Verification error	ユニットは、以下の仕様で表記します。 { デバイス名 }:UnitNo.{ ユニット番号 }[空白]{ ユニット形式 } { デバイス名 } ユーザが設定したデバイス名。 { ユニット番号 } 10 進数の文字列。値の範囲は 0-125。 { ユニット形式 } ユニット形式を示す文字列。 形式の後部からの連続する空白は削除します。 照合結果は、以下の表記で示します。 Result=Matched　照合一致 Result=Not matched　照合不一致

9-10 バックアップ機能のファイル間互換性

バックアップを実行するときとリストアするときとで条件が異なると、ファイル間の互換性の考慮が必要となります。
互換性の考慮が必要となるのは以下の 3 点です。

- バックアップした機能とリストアする機能が異なるとき
- バックアップ元とリストア先の、コントローラの形式が異なるとき
- バックアップ元とリストア先の、コントローラやユニット、スレーブのユニットバージョンが異なるとき

なお、ここでは各バックアップ機能の「リストア」「自動転送」「読出」を、「リストア」と総称します。

9-10-1 バックアップ機能間の互換性

バックアップした機能とリストアする機能が異なるときの、ファイルの互換性は以下のとおりです。
（○：互換性あり、×：互換性なし）

バックアップした機能	リストアする機能				
	SD メモリカードバックアップ機能 (リストア) SD メモリカードからコントローラ	Sysmac Studio コントローラバックアップ機能 (リストア) パソコンからコントローラ	Industrial PC Support Utility コントローラバックアップ機能 (リストア) Windows からコントローラ	Sysmac Studio 変数／メモリのバックアップ機能（リストア) パソコンからコントローラ	Sysmac Studio バックアップファイルのインポート パソコンからプロジェクト
SD メモリカードバックアップ機能（バックアップ） コントローラから SD メモリカード	○	○	○	×	○*1
Sysmac Studio コントローラバックアップ機能（バックアップ） コントローラからパソコン	○	○	○	×	○*1
Industrial PC Support Utility コントローラバックアップ機能（バックアップ） コントローラから Windows	○	○	○	×	○*1
Sysmac Studio 変数／メモリのバックアップ機能（バックアップ） コントローラからパソコン	×	×	×	○	×
Sysmac Studio バックアップファイルのエクスポート プロジェクトからパソコン	○*1	○*1	○*1	×	○

*1. ただし、以下のデータは対象外です。

- コントローラ設定の内蔵 EtherNet/IP ポート名、内蔵 EtherNet/IP 用タグデータリンク設定
- 操作権限認証
- データトレース設定
- 変数の現在値
- 絶対値エンコーダ原点位置オフセット

参考

Sysmac Studio 変数／メモリのバックアップ機能で扱うファイルは、その他のバックアップ機能で扱うファイルと異なるため互換性がありません。
この機能の詳細は 🕮 「9-6 Sysmac Studio 変数／メモリのバックアップ機能」（P.9-25）を参照してください。

9-10-2 NY シリーズコントローラの形式間の互換性

バックアップ元とリストア先のコントローラの形式グループが異なる場合のファイルの互換性は以下のとおりです。

（○：互換性あり、×：互換性なし）

バックアップ元のコントローラ形式	リストア先のコントローラ形式	
	NY5□2-1500 NY5□2-1400 NY5□2-1300	NY5□2-5400
NY5□2-1500 NY5□2-1400 NY5□2-1300	○	×
NY5□2-5400	×	○

参考

NC 統合コントローラとは、互換性がありません。

コントローラの形式の互換性がある場合でも、それぞれのコントローラ形式間で制約がある場合があります。
コントローラの形式間で互換性がある場合とない場合とで、リストアする機能によって動作は以下のようになります。

（○：リストアを実行する、×：リストアを実行しない）

コントローラ形式間の互換性	リストアする機能				
	SD メモリカードバックアップ機能 (リストア) SD メモリカードからコントローラ	Sysmac Studio コントローラバックアップ機能 (リストア) パソコンからコントローラ*1	Industrial PC Support Utility コントローラバックアップ機能 (リストア) Windows からコントローラ	Sysmac Studio 変数／メモリのバックアップ機能（リストア) パソコンからコントローラ	Sysmac Studio バックアップファイルのインポート機能 パソコンからプロジェクト
互換性あり	○*2	○	○	○	○
互換性なし	×*3	×	×	×	×

*1. 本機能でバックアップしたファイルだけがリストア可能です。
*2. バックアップファイルの内容がリストア先の仕様範囲外のときは、正常に動作せず、全停止フォールトレベルのコントローラ異常、または部分停止フォールトレベルのコントローラ異常が発生します。たとえば、使用している制御軸数が仕様外であるときにこの動作となります。
*3. 監視情報の「リストア開始失敗」が発生します。

9-10-3 NY シリーズコントローラのユニットバージョンの互換性

バックアップ元のコントローラのユニットバージョンとリストア先のそれが異なるときの、バックアップファイルの互換性は以下のとおりです。
なお、バックアップ、リストア前後でユニットバージョンが同じ場合は、制約なくリストアを実行できます。

（○：リストアを実行する、×：リストアを実行しない）

コントローラのユニットバージョン	リストアする機能			
	SD メモリカードバックアップ機能 (リストア) SD メモリカードからコントローラ	Sysmac Studio コントローラバックアップ機能 (リストア) パソコンからコントローラ	Industrial PC Support Utility コントローラバックアップ機能 (リストア) Windows からコントローラ	Sysmac Studio 変数／メモリのバックアップ機能（リストア) パソコンからコントローラ
リストア先が新しい	○	○	○	○
リストア先が古い	×	×	×	○

9-11 バックアップ機能と同時に実行できない機能

以下に示す機能は、バックアップ機能と同時には実行できません。コントローラがこれらの機能を実行中の場合は、バックアップ機能を実行しないでください。また、バックアップ機能を実行中の場合は、これらの機能を実行しないでください。

- すでにバックアップ機能を実行中
- 同期転送（パソコン→コントローラ）
- オンラインエディット実行
- メモリオールクリア実行
- カムテーブル保存命令（MC_SaveCamTable）の実行
- CPU ユニット名称書込の実行
- スレーブターミナルのパラメータ設定の転送実行

10

通信設定

本章では、NY シリーズコントローラのオンライン接続、他機器との接続について説明します。

10-1 通信システムの概要 10-2
10-1-1 概要 10-3
10-2 Sysmac Studio の接続 10-4
10-2-1 オンライン接続が可能な構成 10-4
10-2-2 オンラインできない構成 10-5
10-3 他のコントローラやスレーブとの接続 10-6
10-3-1 コントローラとコントローラ間の接続構成 10-6
10-3-2 コントローラースレーブ間の接続構成 10-9
10-4 表示器との接続 10-10

10-1　通信システムの概要

ここでは、NY シリーズの通信システムの概要について、説明します。

全体の手順の中で、以下の網掛けで示した位置に対応しています。

STEP 1. ソフト設計

STEP1-1　I/O と処理の設計
STEP1-2 タスク設計
STEP1-3 プログラムの設計

↓

STEP 2. ソフトウェア設定とプログラミング

STEP2-1 スレーブ／ユニット構成
STEP2-2 コントローラ設定
STEP2-3 プログラミング
STEP2-4 オフラインデバッグ

↓

STEP 3. 装着とハードウェアの設定

↓

STEP 4. 配線

↓

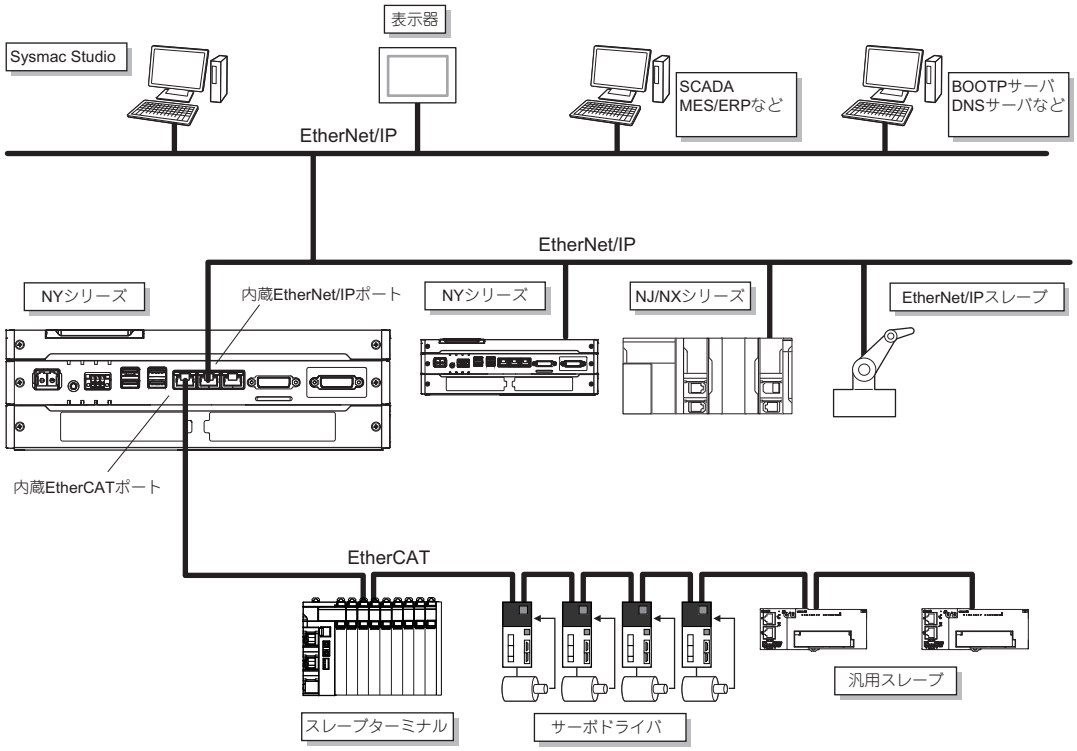
STEP 5. 実機による動作確認と運転

詳細は、📖「1-4 NY シリーズの使用手順の全体」(P.1-10) を参照してください。

10-1-1　概要

● NY シリーズコントローラシステム

NY シリーズのコントローラ機能では、以下の通信システムを構築できます。



接続対象		接続方法
Sysmac Studio の接続		内蔵 EtherNet/IP ポートを使用
コントローラ間の接続	NJ/NX/NY シリーズコントローラと接続	内蔵 EtherNet/IP ポートを使用
コントローラとスレーブ間の接続	サーボドライバ、汎用スレーブを接続	内蔵 EtherCAT ポートを使用
表示器の接続		内蔵 EtherNet/IP ポートを使用
各種システムとの接続	SCADA、MES/ERP を接続	内蔵 EtherNet/IP ポートを使用
各種サーバとの接続	FTP サーバ、BOOTP サーバ、DNS サーバ	内蔵 EtherNet/IP ポートを使用

参考

Windows 側で使用できる通信システムについては、🕮『NY シリーズ マシンオートメーションコントロールソフトウェア搭載 産業用パネル型 PC ／産業用ボックス型 PC ユーザーズマニュアル セットアップ編（SBCA-441）』を参照してください。

10-2 Sysmac Studio の接続

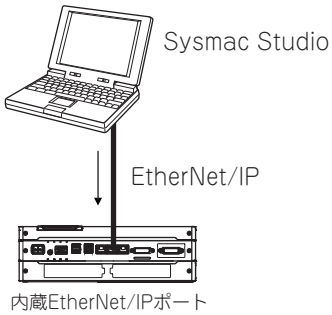
ここでは、Sysmac Studio の NY シリーズへの接続構成について、説明します。

10-2-1 オンライン接続が可能な構成

Sysmac Studio からは、NY シリーズの内蔵 EtherNet/IP ポートに接続して、オンライン接続ができます。

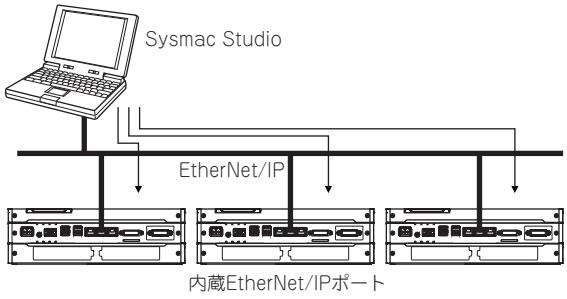
● EtherNet/IP での接続

1：1 の接続：



- Sysmac Studio から直接接続します。IP アドレスの指定、接続デバイスの指定は必要がありません。
- スイッチングハブの有無にかかわらず接続が可能です。
- クロスケーブル、ストレートケーブルのどちらでも可能です。

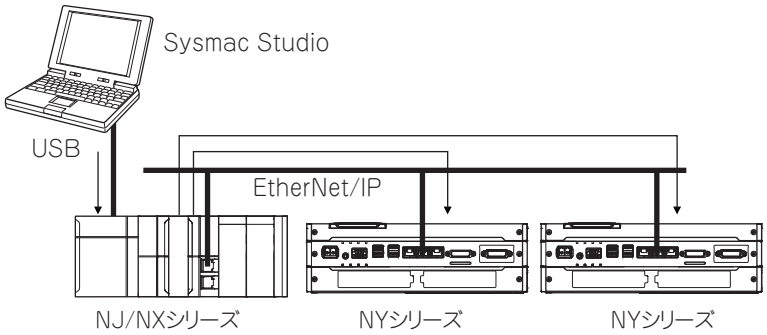
1：N の接続：



- Sysmac Studio から接続相手の IP アドレスを指定します。
- クロスケーブル、ストレートケーブルのどちらでも可能です。

● USB 経由 EtherNet/IP での接続

NJ/NX シリーズの USB ポートを経由し、NY シリーズ内蔵 EtherNet/IP ポートに接続できます。



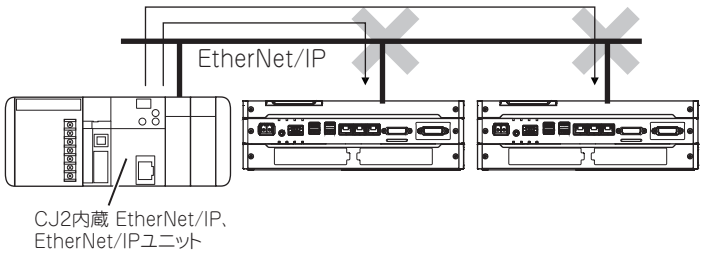
- Sysmac Studio から接続相手の IP アドレスを指定します。
- クロスケーブル、ストレートケーブルのどちらでも可能です。

（注） パソコンと NJ/NX シリーズ CPU ユニット間は、USB2.0 認証のケーブルを使用して、USB ハブを介さずに、接続してください。

10-2-2　オンラインできない構成

● CS/CJ シリーズの EtherNet/IP ユニット／ポートをルーティング

CS/CJ シリーズの EtherNet/IP ユニット／ポート（形 CS1W-EIP21、形 CJ1W-EIP21、CJ2 内蔵 EtherNet/IP ポート、CJ2M 内蔵 EtherNet/IP ポート）をルーティングして NY シリーズコントローラに接続することはできません。



10-3 他のコントローラやスレーブとの接続

ここでは、コントローラとコントローラ間、コントローラとスレーブ間の接続構成について、説明します。

10-3-1 コントローラとコントローラ間の接続構成

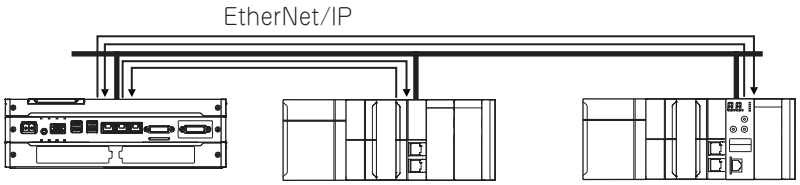
EtherNet/IP

内蔵 EtherNet/IP ポートを使用します。

内蔵 EtherNet/IP ポートについては、🕮 『NY シリーズ 産業用パネル型 PC ／産業用ボックス型 PC ユーザーズマニュアル 内蔵 EtherNet/IP ポート編 （SBCD-369)』を参照してください。

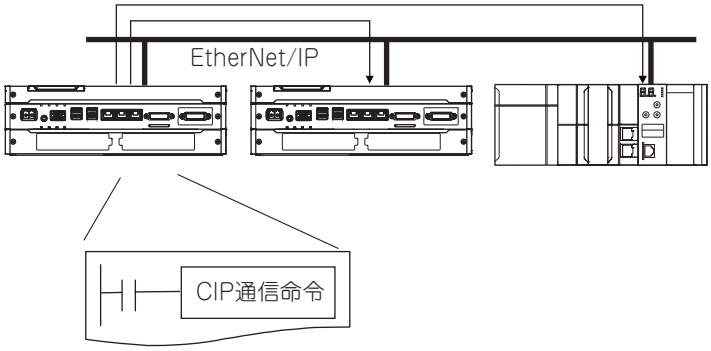
- **タグデータリンク**

  EtherNet/IP 上の NJ/NX/NY シリーズ コントローラとタグデータリンクをすることができます。



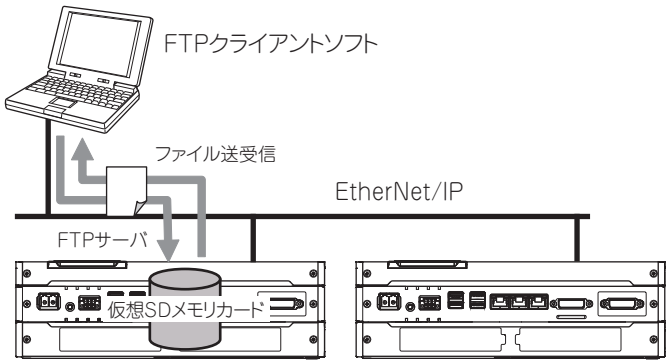
- **メッセージ通信**

  CIP メッセージをプログラム上から発行することができます。



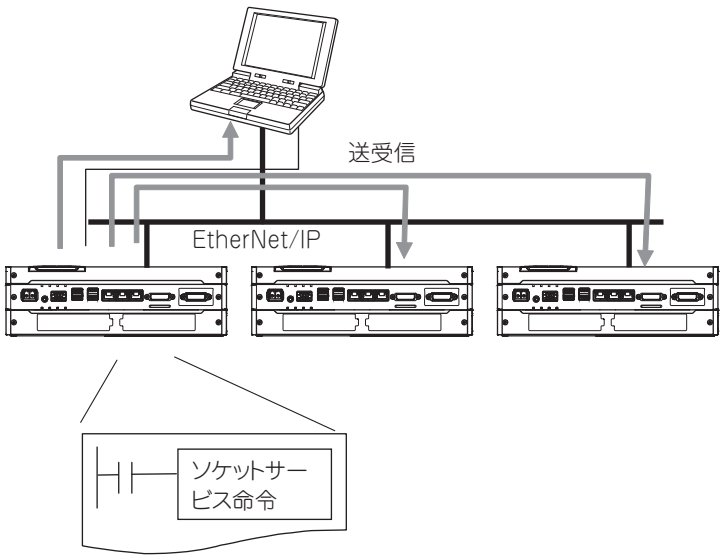
● ファイル送受信

FTP クライアントソフトから NY シリーズコントローラの仮想 SD メモリカード内のファイルの送受信をすることができます。



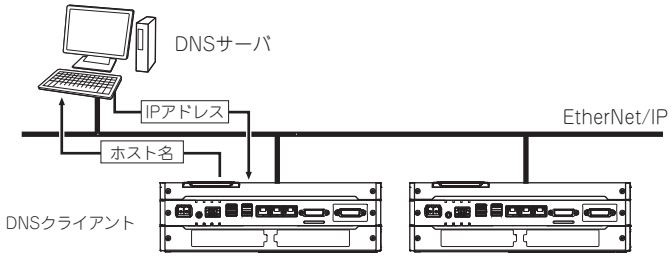
● ソケットサービス

TCP または UDP の機能をユーザプログラムから直接利用することで、ホストコンピューター PLC 間、またはコントローラ間で、任意のデータを相手ノードと送受信することができます。
この機能は内蔵 EtherNet/IP ポートでだけ使用できます。



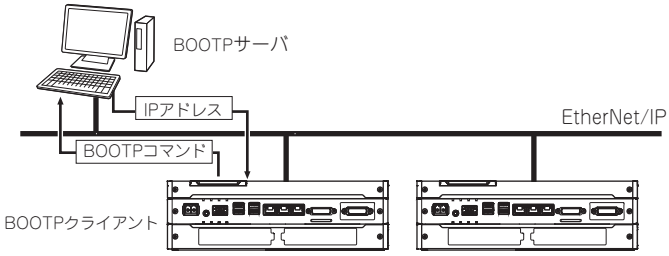
● ホスト名での指定

SNMP マネージャの IP アドレス、またはソケット命令や CIP 通信命令での送信先について、IP アドレスではなく、ホスト名で指定することができます（DNS クライアント機能または Hosts 設定）。
例：DNS サーバでホスト名を設定する



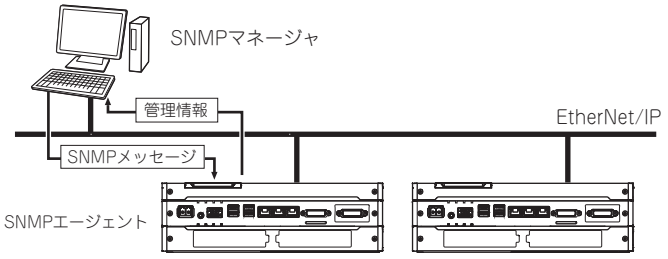
● 電源投入時 IP アドレスの取得

電源投入時、内蔵 EtherNet/IP ポートの IP アドレスを、BOOTP サーバから取得することが可能です。



● SNMP エージェント指定

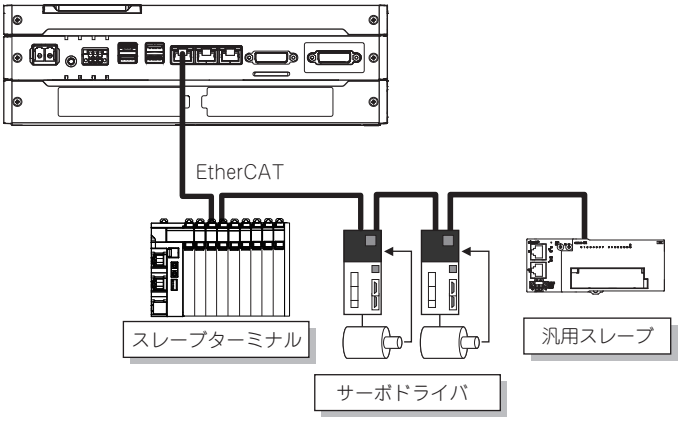
SNMP マネージャを使用したネットワーク管理ソフトウェアに、内蔵 EtherNet/IP ポートの内部状態の情報を提供します。



10-3-2 コントローラースレーブ間の接続構成

EtherCAT

サーボドライバ、汎用スレーブとの高速・高精度通信が可能です。
詳細は 🕮『NY シリーズ 産業用パネル型 PC ／産業用ボックス型 PC ユーザーズマニュアル 内蔵 EtherCAT ポート編（SBCD-368)』を参照してください。

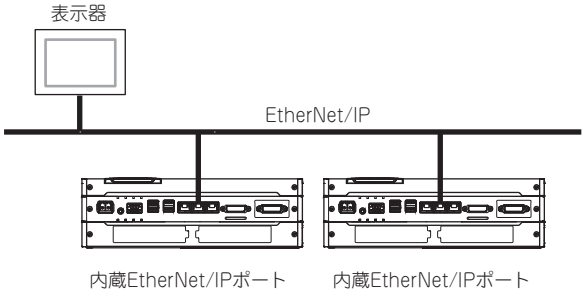


10-4　表示器との接続

ここでは、NS/NA シリーズなどの表示器と NY シリーズコントローラとの接続構成について説明します。

● EtherNet/IP

内蔵 EtherNet/IP ポートを使用します。



内蔵 EtherNet/IP ポートについては 🕮『NY シリーズ 産業用パネル型 PC ／産業用ボックス型 PC ユーザーズマニュアル 内蔵 EtherNet/IP ポート編 （SBCD-369)』を参照してください。

11

実際の使用手順例

本章では、NY シリーズコントローラを実際に使用する場合の使用手順例について説明します。

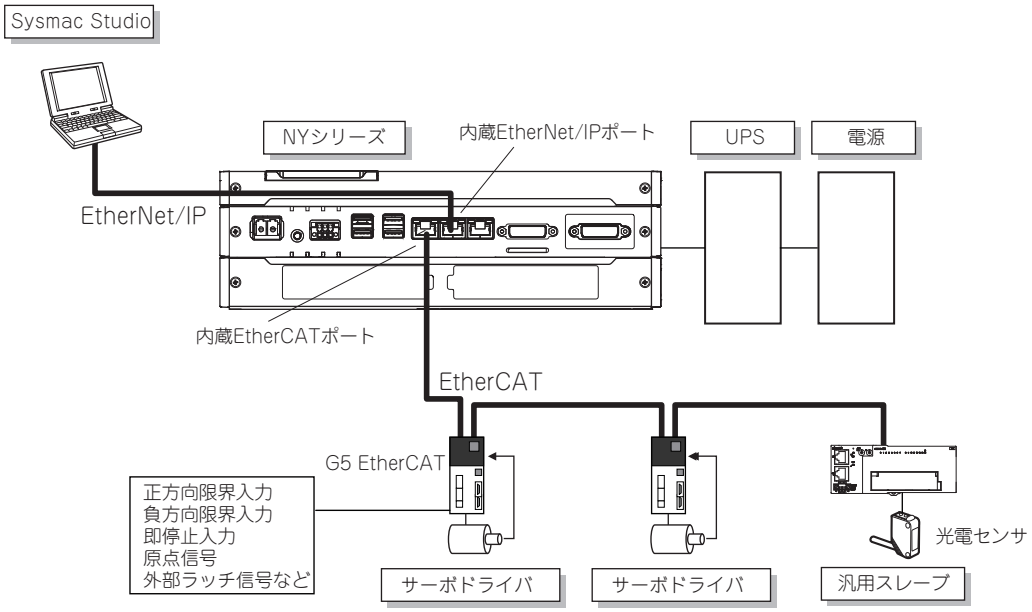
11-1 実際の使用例 11-2
11-1-1 システム構成 11-2
11-1-2 動作 11-2
11-2 操作手順例の概要 11-3
11-2-1 配線・設定 11-3
11-2-2 ソフト設計 11-3
11-2-3 Sysmac Studio でのソフトウェア設定 11-4
11-2-4 Sysmac Studio でのプログラミング 11-8
11-2-5 実機による動作確認と運転 11-9

11-1　実際の使用例

ここでは、NY シリーズの実際の使用例について説明します。

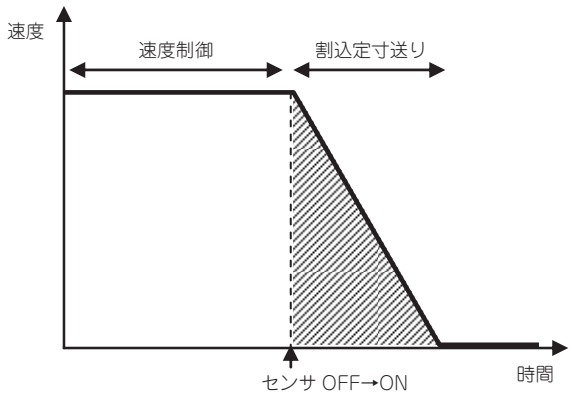
11-1-1　システム構成

ユニット名		台数	接続機器
電源		1 台	−
UPS（無停電電源装置）		1 台	−
NY シリーズ 産業用ボックス型 PC		1 台	−
EtherCAT スレーブ	サーボドライバ G5EtherCAT タイプ	2 台	−
	I/O ターミナル	1 台	光電センサ



11-1-2　動作

速度制御中に、センサの信号が OFF → ON で、割込定寸送りをします。



光電センサからの入力をもとに、縦方向の位置を変更します。

11-2 操作手順例の概要

ここでは、NY シリーズの実際の操作手順例について、説明します。

11-2-1 配線・設定

配線、およびハード的な設定をします。

11-2-2 ソフト設計

I/O の設計、タスク設計、POU 設計、および変数設計を行います。

I/O の設計

- 外部 I/O とユニットの構成関係を設計します。
- どの外部 I/O をどのくらいの間隔でリフレッシュするかを設計します。

タスクおよび POU の設計

以下を検討します。

- タスク構成をどうするか
- どのプログラムをどの周期のタスクに割り付けるか
- どのユニットをどのタスクに割り付けるか
- どの処理をプログラムにし、どの処理をファンクションブロック、ファンクションにするか

変数の設計

以下を検討します。

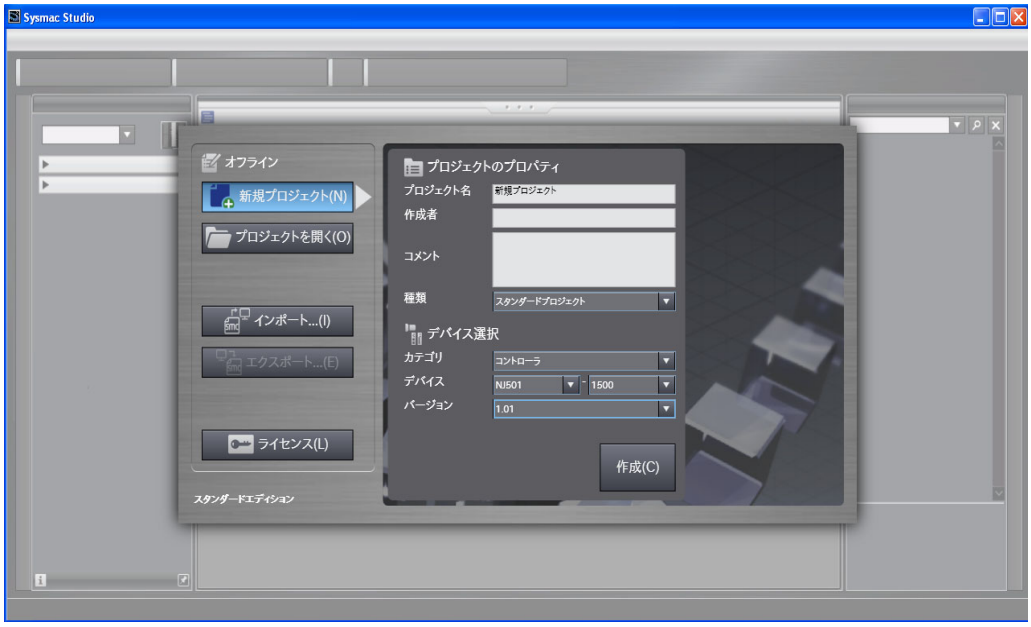
- POU 共通で使用する変数（グローバル変数）と、各 POU だけで使用する変数（ローカル変数）の仕分け
- スレーブにアクセスするためのデバイス変数の変数名を設計
- 変数の名称、保持などの各属性を設計
- 変数のデータ型を設計

11-2-3　Sysmac Studio でのソフトウェア設定

Sysmac Studio にて、スレーブ構成、グローバル変数／デバイス変数の登録、軸（軸変数）作成、およびコントローラ設定をします。

Sysmac Studio を起動します

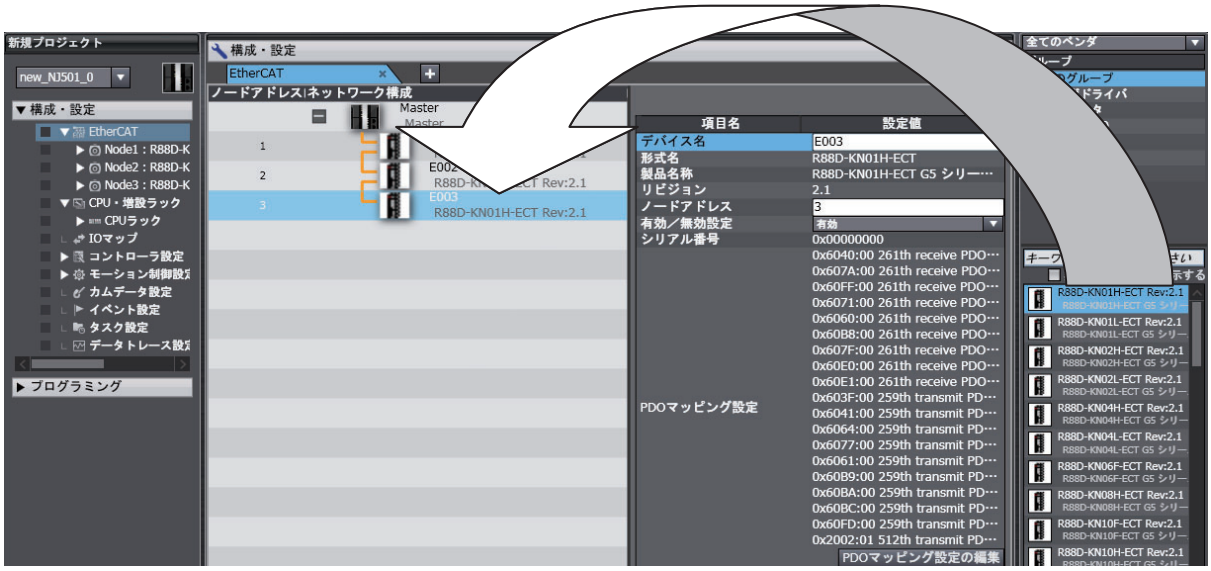
新規プロジェクトを作成します。



EtherCAT スレーブを構成します

1　［構成・設定］の下の［EtherCAT］をダブルクリックします。

2　スレーブをドラッグ & ドロップして、スレーブ構成を作成します。



3　マスタを選択して、マスタパラメータを設定します。

4　各スレーブを選択して、スレーブのパラメータを設定します。

参考

この時点でI/Oマップから強制値リフレッシュを行うことにより配線確認を行うことができます。

グローバル変数／デバイス変数を登録します

● グローバル変数の登録

1 **[プログラミング] の下の [データ] の下の [グローバル変数] をダブルクリックします。**

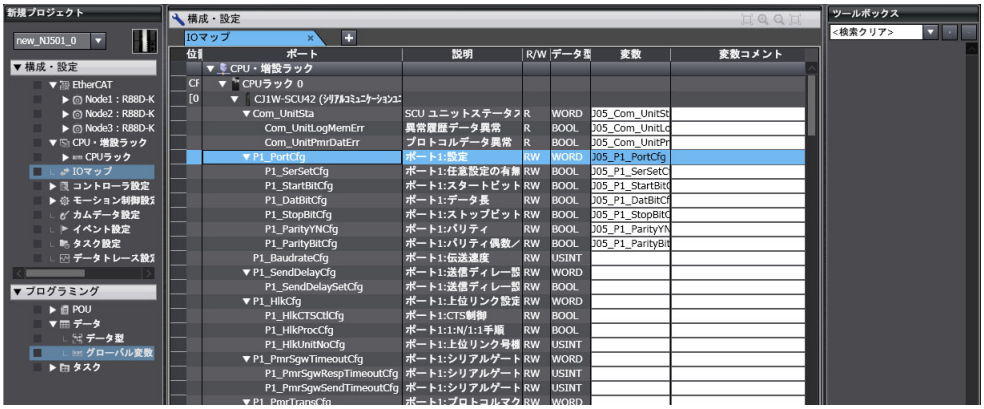
2 **グローバル変数テーブルに、グローバル変数を登録します。**

● デバイス変数の登録

1 **[構成・設定] の下の [IO マップ] をダブルクリックします。**

2 **I/O マップにて、（スレーブ構成によって自動的に作成された）I/O ポートに対して、変数を割り付けます。**

なお、Sysmac Studio でデバイス変数名を自動生成することも可能です。その場合、I/O ポートを選択して、右クリックして [デバイス変数生成] を選択します。

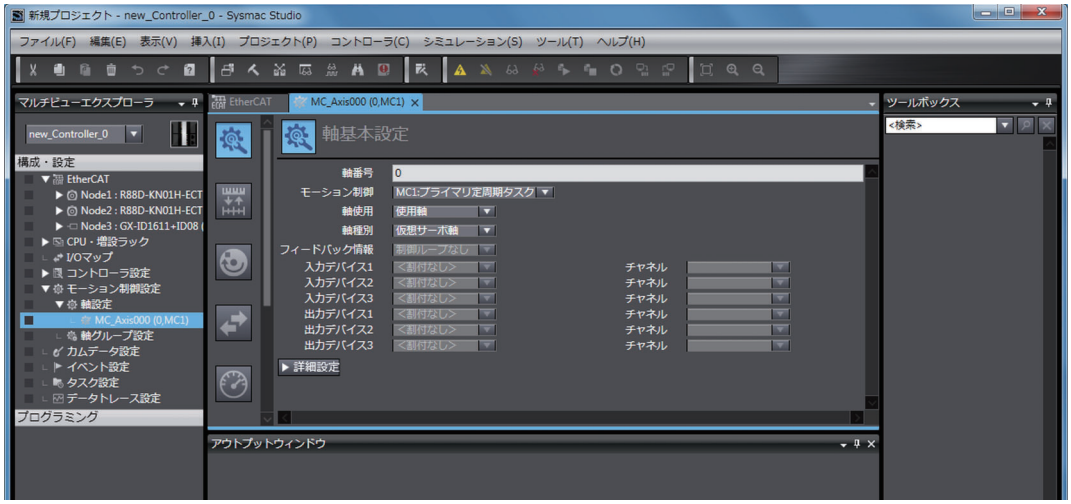


デフォルトでは、生成したデバイス変数は、グローバル変数テーブルに登録されます。
必要に応じて、[変数種別] を「グローバル変数」から各 POU のローカル変数（内部変数）に変更します。

「軸」（軸変数）を作成します

1 **［構成・設定］の下の［モーション制御設定］–［軸設定］を選択して右クリックして［追加］–［軸設定］を選択します。**

2 **作成した軸（軸変数）に対して、EtherCAT 構成上のサーボドライバを割り付けます。**



- ［軸使用］を［使用軸］に選択。
- ［軸種別］を［サーボ軸］に選択。
- ［入力デバイス］、［出力デバイス］に、スレーブ構成に登録した EtherCAT スレーブを選択。

その他、単位変換、動作設定などの軸パラメータを設定します。

コントローラ設定をします

- **PLC モジュール関連の初期設定：**

  コントローラ設定：電源投入時モードなど

11-2-4 Sysmac Studio でのプログラミング

Sysmac Studio にて、プログラム作成、タスク設定、ビルドをします。

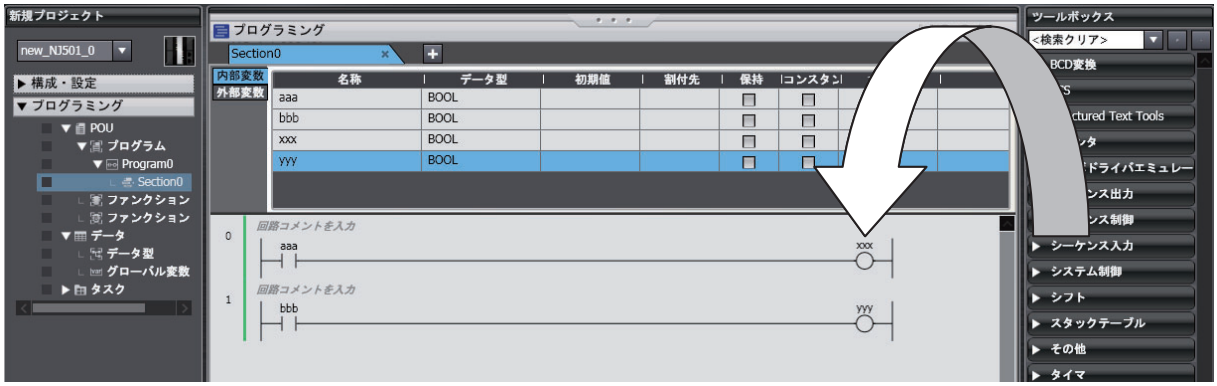
プログラミング

1 **[プログラミング]の下の[POU]の下の[プログラム]を選択し、右クリックして[追加]-[ラダー]または [ST] を選択します。**

2 **作成した「プログラム」の下の「Section□」をダブルクリックします。**

3 **各プログラムのローカル変数を登録します。**

4 **プログラムを作成します。**



以下の命令を記述したプログラムを作成します。

- 原点決め：MC_Home 命令
- 速度制御：MC_MoveVelocity 命令
- 割込定寸送り：MC_MoveFeed 命令
- 位置決め：MC_Move 命令

5 **必要に応じて、[プログラミング]の下の[POU]の下の[ファンクション]または[ファンクションブロック] を選択して、[追加]-[ラダー] または [ST] を選択します。**

作成した「Function□」または「FunctionBlock□」をダブルクリックします。
各ファンクションまたはファンクションブロックのローカル変数を登録します。
各アルゴリズムを作成します。
（注）　ラダーの場合、[R] キーを押して、次の回路を作成します。

タスク設定

[構成・設定] の下の [タスク設定] をダブルクリックします。

- [タスク設定] にて、プライマリ定周期タスクの「周期 / 実行条件」の、タスク周期をプルダウンリストから選択します。
- [I/O を制御するタスク設定] にて、各ユニット／スレーブを割り付けるタスク名を選択します。
- [プログラムの割付設定] にて、プログラムをプライマリ定周期タスクまたは定周期タスク（実行優先度 16）に割り付けます。

ビルド

[プロジェクト] メニューから [ビルド] を選択します。

11-2-5　実機による動作確認と運転

オンライン接続して、プロジェクトのダウンロード、配線確認、試運転の上、本運転をします。

オンライン接続

1. **NY シリーズコントローラの電源を投入します。**
2. **パソコンと NY シリーズコントローラを EtherNet/IP で接続します。**
3. **[コントローラ]メニューから[通信設定]を選択します。[接続方法]から接続形態に応じた接続方法を選択します。**
4. **[コントローラ]メニューから[オンライン]を選択します。**

プロジェクトのダウンロード（「同期」機能による）

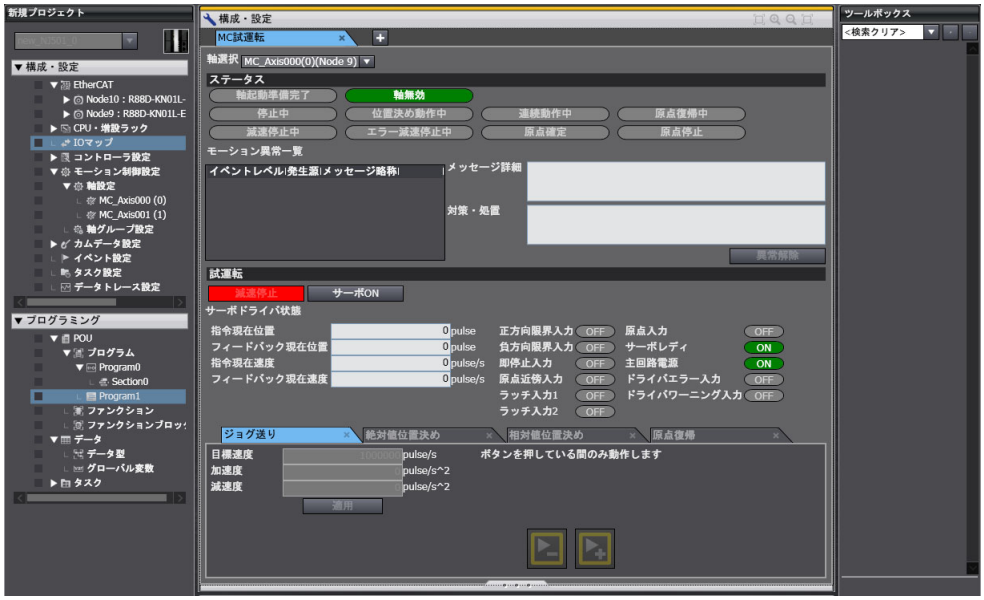
[コントローラ]メニューから[同期]を選択して、プロジェクトをコントローラにダウンロードします。
（注）「ダウンロード」および「アップロード」は、Sysmac Studio の「同期」機能を使用します。

配線確認

I/O マップまたはラダーエディタにて、実 I/O への強制値リフレッシュなどをして、配線確認をします。

MC 試運転

1. **MC 試運転画面を開きます。**
2. **NY シリーズコントローラをプログラムモードにします。**
3. **画面から入力信号をモニタし配線チェックを行います。**
4. **画面からジョグ送りをさせます。**



手動運転

NY シリーズコントローラを運転モードにします。

- サーボ ON/OFF：モーション制御命令の MC_Power を実行
- ジョグ送り：モーション制御命令の MC_MoveJog を実行

原点決め

原点決め：MC_Home 命令を実行

本運転

［コントローラ］メニューから［動作モード]-[運転モード］を選択します。
異常が発生している場合、原因を調査し、ユーザプログラムを修正します。

A

付録

本章では、NY シリーズコントローラの詳細仕様、タスク実処理時間、システム定義変数の個別仕様など、本文の補足的情報を記述します。

A-1 仕様 A-3
A-1-1 性能仕様 A-3
A-1-2 機能仕様 A-6

A-2 NY シリーズでのタスク実処理時間の目安の計算 A-12
A-2-1 タスク実処理時間の平均値の計算方法 A-13
A-2-2 タスク実処理時間の平均値の計算とタスク周期の設定の例 A-19

A-3 システム定義変数一覧 A-22
A-3-1 NY シリーズコントローラのシステム全体（カテゴリ名：なし） A-22
A-3-2 PLC 機能モジュール（カテゴリ名：_PLC） A-30
A-3-3 モーション制御機能モジュール（カテゴリ名：_MC） A-31
A-3-4 EtherCAT マスタ機能モジュール（カテゴリ名：_EC） A-32
A-3-5 EtherNet/IP 機能モジュール（カテゴリ名：_EIP） A-37
A-3-6 異常ステータスの各ビットの意味 A-45

A-4 システム定義変数の個別仕様 A-46
A-4-1 NY シリーズコントローラのシステム全体（カテゴリ名：なし） A-47
A-4-2 PLC 機能モジュール（カテゴリ名：_PLC） A-58
A-4-3 モーション制御機能モジュール（カテゴリ名：_MC） A-60
A-4-4 EtherCAT マスタ機能モジュール（カテゴリ名：_EC） A-62
A-4-5 EtherNet/IP 機能モジュール（カテゴリ名：_EIP） A-70

A-5 コントローラ内データの属性 A-80

A-6 変数のメモリ確保の方法 A-83
A-6-1 変数のメモリ確保の規則 A-83
A-6-2 留意するべきケース例 A-93

A-7 CX-Designer の変数テーブル登録方法 A-96

A-8 EtherCAT スレーブや軸の有効／無効切替 A-99
A-8-1 EtherCAT スレーブや軸を使用する場合のプロジェクト設定 A-99
A-8-2 EtherCAT スレーブや軸の有効／無効切替を行う命令 A-99
A-8-3 EtherCAT スレーブや軸の状態を示すシステム定義変数 A-100
A-8-4 プログラムの実行／停止指令 A-101
A-8-5 プログラムの実行／停止の確認 A-101
A-8-6 Sysmac Studio での設定 A-102
A-8-7 EtherCAT スレーブや軸の有効／無効切替のアプリケーション例 A-103

A-9 ユーザプログラムの大きさに関する制限 A-106
A-9-1 ユーザプログラムの各部品の制限 A-106

A-9-2　ユーザプログラムの各部品の数 A-108

A-10NY シリーズのバージョン関連情報 A-110
A-10-1 ユニットバージョンと Sysmac Studio のバージョンとの関係 A-110
A-10-2 ユニットバージョンの更新によるサポート機能の追加と変更 A-111

A-1 仕様

ここでは、NY シリーズ マシンオートメーションコントロールソフトウェア搭載 産業用パネル型 PC ／産業用ボックス型 PC の制御機能の性能仕様と機能仕様について、説明しています。

一般仕様については、🕮『NY シリーズ 産業用パネル型 PC ユーザーズマニュアル ハードウェア編（SBCA-435）』、🕮『NY シリーズ 産業用ボックス型 PC ユーザーズマニュアル ハードウェア編（SBCA-434）』を参照してください。

A-1-1 性能仕様

ここでは、NY シリーズ 産業用パネル型 PC ／産業用ボックス型 PC の制御機能の性能仕様について、説明しています。

項目				形 NY5□2-		
				15□□	□4□□	13□□
処理時間	命令実行時間	LD 命令		0.33ns ～		
		算術命令（倍精度実数型）		1.2ns ～		
プログラミング	プログラム容量*1	サイズ		40MB		
		数	POU 定義数	3,000		
			POU インスタンス数	24,000		
	変数容量	保持属性あり	サイズ	4MB		
			変数の数	40,000		
		保持属性なし	サイズ	64MB		
			変数の数	180,000		
	データ型	データ型の数		4,000		
モーション制御	制御軸数	制御軸最大数*2		64 軸	32 軸	16 軸
		使用実軸最大数*3		64 軸	32 軸	16 軸
		単軸制御最大数		64 軸	32 軸	16 軸
		直線補間制御最大数		1 軸グループあたり 4 軸		
		円弧補間制御軸数		1 軸グループあたり 2 軸		
	軸グループ最大数			32 グループ		
	モーション制御周期			EtherCAT 通信のプロセスデータ通信周期と同じ		
	カム	カムデータ点数	1 カムテーブルあたりの最大点数	65,535 点		
			全カムテーブルの最大点数	1,048,560 点		
		カムテーブル最大テーブル数		640 テーブル		
	位置単位			パルス、mm、μm、nm、degree、inch		
	オーバーライド			0.00、0.01 ～ 500.00%		

項目				形 NY5□2-		
				15□□	□4□□	13□□
内蔵 EtherNet/IP ポート	ポート数			1		
	物理層			10BASE-T/100BASE-TX/1000BASE-T		
	フレーム長			最大 1,514 バイト		
	媒体アクセス方式			CSMA/CD		
	変調方式			ベースバンド		
	トポロジ			スター型		
	伝送速度			1Gbps（1000BASE-T）		
	伝送媒体			ツイストペアケーブル（シールド付：STP）：カテゴリ 5、5e 以上		
	伝送距離（ハブとノード間の距離）最大値			100m		
	カスケード接続最大数			スイッチングハブの使用において制限なし		
	CIP サービス：タグデータリンク（サイクリック通信）	コネクション最大数		128		
		パケットインターバル *4		コネクションごとに設定可能 1 ～ 10000ms（1ms 単位）		
		ユニット許容通信帯域		20,000 pps*5（ハートビート含む）		
		タグセット最大数		128		
		タグ種別		ネットワーク変数		
		1 コネクション（=1 タグセット）あたりのタグ数		8（タグセットにコントローラステータスを含める場合は 7）		
		タグ最大数		256		
		1 ノードあたりの最大リンクデータサイズ（全タグの合計サイズ）		184,832 バイト		
		1 コネクションあたりの最大データサイズ		1,444 バイト		
		登録可能なタグセット最大数		128（1 コネクション= 1 タグセット）		
		1 タグセットの最大サイズ		1,444 バイト（タグセットにコントローラステータスを含める場合は 2 バイト分を使用）		
		マルチキャストパケットフィルタ機能*6		可		
	CIP メッセージサービス：Explicit メッセージ	Class3（コネクション数）		64（クライアント＋サーバ）		
		UCMM（非コネクション型）	同時通信可能なクライアント最大数	32		
			同時通信可能なサーバ最大数	32		
	TCP ソケット数			30		
内蔵 EtherCAT ポート	通信規格			IEC 61158 Type12		
	EtherCAT マスタ仕様			Class B 対応（Feature Pack Motion Control 対応）		
	物理層			100BASE-TX		
	変調方式			ベースバンド		
	伝送速度			100Mbps（100BASE-TX）		
	Duplex モード			Auto		
	トポロジ			ライン、ディジーチェーン、分岐		
	伝送媒体			カテゴリ 5 以上　ツイストペアケーブル（アルミテープと編組の二重遮へいシールドケーブル、ストレートを推奨）		
	ノード間距離最大値			100m		
	スレーブ最大数			128		
	設定可能ノードアドレス範囲			1 ～ 512		
	プロセスデータの最大サイズ			IN：5,736 バイト OUT：5,736 バイト ただし、4 フレーム（最大プロセスデータフレーム数）の範囲内であること		
	1 スレーブの最大サイズ			IN：1,434 バイト OUT：1,434 バイト		

項目			形 NY5□2-		
			15□□	□4□□	13□□
内蔵 EtherCAT ポート	通信周期		500/1,000/2,000/4,000/8,000 μs		
	同期ジッタ		1 μs 以下		
ユニット構成	装着できるユニット数	システム全体の NX ユニット最大数	4,096 (EtherCAT スレーブターミナル上)		
	増設ラック最大数		0		
内蔵時計			周囲温度　55 ℃　：　月差　-3.5 分～ +0.5 分 周囲温度　25 ℃　：　月差　-1.5 分～ +1.5 分 周囲温度　0 ℃　：　月差　-3 分～ +1 分		

***1. 実行オブジェクト、変数テーブル（変数名など）の容量です。**
***2. すべての軸種別の合計のことです。**
***3. 軸種別が [サーボ軸] または [エンコーダ軸] で、かつ軸使用が [使用軸] に設定された軸の合計のことです。**
***4. ノード数に依存せず、設定した周期で回線上をデータ更新します。**
***5. pps とは Packet Per Second を意味し、1 秒間に処理可能な送受信パケット数を示します。**
***6. EtherNet/IP ポートは、IGMP クライアントを実装しているため、IGMP Snooping 対応のスイッチングハブを使用することで不要なマルチキャストパケットのフィルタリングが行えます。**

A-1-2　機能仕様

ここでは、NY シリーズ 産業用パネル PC ／産業用ボックス型 PC の制御機能の機能仕様について、説明しています。

項目				形 NY5□2-□□□□
タスク機能	機能			実行条件と実行優先度を指定する「タスク」単位で、I/O リフレッシュおよびユーザプログラムの実行を行う機能
		定周期で実行するタスク	プライマリ定周期タスク最大数	1
			定周期タスク最大数	3
		条件成立時に実行するタスク	イベントタスク最大数	32
			実行条件	イベントタスク起動命令実行時、変数の条件式一致時
	設定	システムサービス監視設定		−
プログラミング機能	POU（Program Organization Unit）	プログラム		タスクに割り付ける単位の POU
		ファンクションブロック		状態を持つ部品を作成するときに使用する POU
		ファンクション		演算処理など、入力に対して出力が一意に決まる部品を作成するときに使用する POU
	プログラム言語	種類		ラダー図[*1]、ストラクチャードテキスト（ST）
	名前空間			名前で POU 定義のグループ分けを行う機能
	変数	変数の外部参照機能	ネットワーク変数	表示器や上位パソコン、他のコントローラなどからのアクセスを許可する機能
	データ型	基本データ型	ブール型	BOOL
			ビット列型	BYTE、WORD、DWORD、LWORD
			整数型	INT、SINT、DINT、LINT、UINT、USINT、UDINT、ULINT
			実数型	REAL、LREAL
			持続時間型	TIME
			日付型	DATE
			時刻型	TIME_OF_DAY
			日付時刻型	DATE_AND_TIME
			文字列型	STRING
		派生データ型		構造体型、共用体型、列挙型
		構造体型	機能	データ型の異なる複数のデータをひとつにまとめて扱う機能
			メンバ最大数	2,048
			ネスト最大数	8
			メンバのデータ型	基本データ型、構造体型、共用体型、列挙型、配列変数
			メンバのオフセット指定	構造体メンバを任意のメモリ位置に配置する機能
		共用体型	機能	同一のデータに対して、複数の異なるデータ型でアクセスできるようにする機能
			メンバ最大数	4
			メンバのデータ型	BOOL、BYTE、WORD、DWORD、LWORD
		列挙型	機能	変数の値を「列挙子」と呼ぶラベル（文字列）で表現した機能
	データ型の属性	配列指定	機能	同じデータ型の要素をまとめて、先頭からの番号（添え字）で指定する機能
			次元最大数	3
			要素最大数	65,535
			FB インスタンスの配列指定	可能
		範囲指定		あらかじめ決められた範囲内の値しかとることができないように明示する機能
	ライブラリ			(ユーザ) ライブラリ使用可能

項目				形 NY5□2-□□□□
モーション制御機能	制御モード			位置制御、速度制御、トルク制御
	軸種別			サーボ軸、仮想サーボ軸、エンコーダ軸、仮想エンコーダ軸
	管理可能な位置			指令位置、フィードバック位置
	単軸	単軸位置制御	絶対値位置決め	絶対座標の目標位置を指定して、位置決めを行う機能
			相対値位置決め	指令現在位置からの移動距離を指定して、位置決めを行う機能
			割込定寸位置決め	外部入力による割込入力が発生した位置からの移動距離を指定し、位置決めを行う機能
			サイクリック同期絶対位置制御	位置制御モードで、制御周期ごとに指令位置を出力する機能
		単軸速度制御	速度制御	位置制御モードで速度制御を行う機能
			サイクリック同期速度制御	速度制御モードで制御周期ごとに速度指令を出力する機能
		単軸トルク制御	トルク制御	モータのトルク制御を行う機能
		単軸同期制御	カム動作開始	指定したカムテーブルを用いてカム動作を開始する機能
			カム動作解除	入力パラメータで指定した軸のカム動作を終了する機能
			ギア動作開始	主軸と従軸間のギア比を設定しギア動作を行う機能
			位置指定ギア動作	主軸と従軸間のギア比と同期する位置を設定しギア動作を行う機能
			ギア動作解除	実行中のギア動作、位置指定ギア動作を中止する機能
			台形パターンカム	指定した主軸に同期して位置決めを行う機能
			主軸相対値位相補正	同期制御中の主軸の位相補正を行う機能
			加減算位置決め	2 軸の指令位置を加算した値、または減算した値を指令位置として出力する機能
		単軸手動操作	運転可	サーボドライバの状態をサーボ ON 状態に切り替えて、軸動作を可能にする機能
			ジョグ送り	指定した目標速度にしたがって、ジョグ送りを行う機能
		単軸制御補助	軸エラーリセット	軸の異常を解除する機能
			原点復帰	モータを動かして、限界信号、原点近傍信号、原点信号を用いて機械原点を決める機能
			パラメータ指定原点復帰	パラメータを指定しモータを動かして、限界信号、原点近傍信号、原点信号を用いて機械原点を決める機能
			高速原点復帰	絶対座標の「0」を目標位置とし、位置決めを行い、原点へ戻す機能
			強制停止	軸を減速停止する機能
			即停止	軸を即停止する機能
			オーバーライド値設定	軸の目標速度を変更する機能
			現在位置変更	軸の指令現在位置とフィードバック現在位置を任意の値に変更する機能
			外部ラッチ有効	トリガの発生により軸の位置を記録する機能
			外部ラッチ無効	実行中のラッチを無効にする機能
			ゾーン監視	軸の指令位置もしくはフィードバック現在位置が、指定された範囲（ゾーン）内に存在するかを判定する機能
			デジタルカムスイッチ有効	軸の位置に応じてデジタル出力を ON または OFF する機能
			軸間偏差監視	指定した 2 軸の指令位置またはフィードバック位置の差分が許容値を超えることがないかを監視する機能
			偏差カウンタリセット	指令現在位置とフィードバック現在位置の間の偏差をゼロにする機能
			トルク制限	サーボドライバのトルク制限機能の有効／無効の切り替えとトルク制限値の設定を行うことで、出力トルクを制限する機能
			従軸位置補正	同期制御中の従軸に対して位置補正する機能
			カムモニタ（*）	カム動作に関する情報（位相、変位など）をモニタする機能
			起動速度	軸動作が開始するときの初速度を設定する機能

項目				形 NY5□2-□□□□
モーション制御機能	軸グループ	多軸協調制御	絶対値直線補間	絶対位置を指定して直線補間を行う機能
			相対値直線補間	相対位置を指定して直線補間を行う機能
			2 軸円弧補間	2 軸の円弧補間を行う機能
			軸グループサイクリック同期絶対位置制御	位置制御モードで制御周期ごとに位置指令を出力する機能
		多軸協調制御補助	軸グループエラーリセット	軸グループおよび軸の異常を解除する機能
			軸グループ有効	軸グループの動作を有効にする機能
			軸グループ無効	軸グループの動作を無効にする機能
			軸グループ強制停止	補間動作中の全ての軸を減速停止する機能
			軸グループ即停止	補間動作中の全ての軸を即停止する機能
			軸グループオーバーライド値設定	補間動作中の合成目標速度を変更する機能
			軸グループ位置取得	軸グループの指令現在位置とフィードバック現在位置を取得する機能です。
			軸グループ構成軸書込	軸グループパラメータの［構成軸 ］を、一時的に書き換える機能
	共通	カム	カムテーブルプロパティ更新	入力パラメータで指定したカムテーブルの終点インデックスを更新する機能
			カムテーブル保存	入力パラメータで指定したカムテーブルを CPU ユニット内の不揮発性メモリに保存する機能
			カムテーブル生成	入力パラメータで指定したカムプロパティとカムノードから、カムテーブルを生成する機能
		パラメータ	MC 設定書込	軸パラメータおよび軸グループパラメータの一部を、一時的に書き換える機能
			軸パラメータの変更	軸パラメータを、ユーザプログラムから参照・変更する機能
	補助機能	カウントモード		リニアモード（有限長）またはロータリモード（無限長）を選択可能
		単位変換		各軸の表示単位を機械に合わせて設定可能
		加減速制御	自動加減速制御	軸や軸グループ動作時の加減速カーブをジャークで設定する機能。
			加減速度変更	加減速動作中でも、加減速度の変更する機能
		インポジションチェック		位置決め完了をチェックするためのインポジション幅とインポジションチェック時間を設定する機能
		停止方法選択		即停止入力信号や限界入力信号が有効になったときの停止方法を設定する機能
		モーション制御命令の再起動		実行中のモーション制御命令の入力変数を変更し再起動することで、動作中に目標値を変更する機能
		モーション制御命令の多重起動（バッファモード）		動作中に別のモーション制御命令を起動した場合の実行開始タイミングと動作間の速度の接続方法を指定する機能
		軸グループ動作の連続動作（トランジションモード）		軸グループ動作の多重起動による連続動作の方法を指定する機能
		監視機能	ソフトウェアリミット	軸の動作範囲を監視する機能
			位置偏差	軸の指令現在値とフィードバック現在値との間の位置偏差を監視する機能
			速度／加減速度／トルク／補間速度／補間加減速度	軸ごとおよび軸グループごとに警告値を設定し、監視する機能
		絶対値エンコーダ対応		オムロン製サーボドライバ 1S シリーズまたは G5 シリーズの絶対値エンコーダ付きモータを使用することで、電源投入時の原点復帰を不要とする機能
		入力信号の論理反転		即停止入力信号、正方向限界入力信号、負方向限界入力信号、原点近傍入力信号の論理を反転する機能
	外部 I/F 信号			サーボドライバ側の下記入力信号を使用可能。 原点信号、原点近傍信号、正方向限界信号、負方向限界信号、即停止信号、割込入力信号
ユニット（入出力）管理機能	EtherCAT スレーブ	スレーブ最大数		128

項目				形 NY5□2-□□□□
通信機能	EtherNet/IP ポート	通信プロトコル		TCP/IP、UDP/IP
		TCP/IP 機能	CIDR	IP アドレスのクラス (クラス A ～クラス C) を使わない IP アドレスの割り当てを行う機能
			IP Forwarding	IP パケットのインターフェース間転送機能
			Packet Filter*2	IP パケットを検査して送信元 IP アドレスや、TCP ポート番号などにより送受信させるかを判断する機能
			NAT	2 つの IP アドレスを変換して転送する機能
		CIP 通信サービス	タグデータリンク	EtherNet/IP ネットワーク上のデバイスとプログラムレスでサイクリックにデータ交換を行う機能
			メッセージ通信	EtherNet/IP ネットワーク上のデバイスと任意の CIP コマンドを送受信する機能
		TCP/IP アプリケーション	ソケットサービス	UDP または TCP プロトコルで、Ethernet 上の任意のノードと任意のデータの送受信を行う機能。ソケット通信用命令により実行する機能
			FTP クライアント	CPU ユニットから、Ethernet 上の他のコンピュータやコントローラに対して、FTP によるファイル転送を行う機能。FTP クライアント通信命令により実行する。
			FTP サーバ	Ethernet 上の他のコンピュータからコントローラの CPU ユニットの SD メモリカード内のファイルを読み書きする機能
			SNMP エージェント	SNMP マネージャを使用したネットワーク管理ソフトウェアに内蔵 EtherNet/IP ポートの内部状態の情報を提供する機能
	EtherCAT ポート	サポートサービス	プロセスデータ通信	CoE で定義されている、EtherCAT マスタとスレーブとの間で制御情報を一定周期のサイクリック通信としてデータ交換を行う通信方式
			SDO 通信	CoE で定義されている、EtherCAT マスタとスレーブとの間で制御情報を非定周期のイベント通信としてデータ交換を行う通信方式
		ネットワークスキャン		接続されているスレーブ機器の情報を読み出し、スレーブ構成を自動生成する機能
		DC （Distributed Clock）		すべての EtherCAT デバイス（マスタ含む）で同じ「EtherCAT System Time」を共有することで時刻同期を行う機能
		パケットモニタ		マスタが送信したフレームと、受信したフレームを保存するための機能。保存したデータは WireShark などで見ることが可能
		スレーブ有効／無効設定		スレーブを通信対象として、有効とするか無効とするかの設定する機能
		スレーブ離脱／再加入		スレーブの交換などの保守のため、EtherCAT ネットワークから該当のスレーブを一時的に離脱 / 再加入させる機能
		サポートアプリケーションプロトコル	CoE	スレーブに対して EtherCAT 上で CAN アプリケーションのメッセージ（SDO）を送る機能
	通信用命令			FTP クライアント命令、CIP 通信命令、ソケット通信用命令、SDO メッセージ命令、Modbus RTU プロトコル命令
システム管理機能	イベントログ	機能		イベントの発生を記録する機能
		最大件数	システムイベントログ	2,048
			アクセスイベントログ	1,024
			ユーザイベントログ	1,024

項目				形 NY5□2-□□□□
デバッグ機能	オンラインエディット			プログラム、ファンクションブロック、ファンクション、グローバル変数をオンラインで変更を行う機能。 ネットワーク経由で複数の作業者から個別の POU の変更が可能
	強制値リフレッシュ			特定の接点を強制値でリフレッシュする機能
		最大点数	EtherCAT スレーブデバイス変数	64
	MC 試運転			モータの動作や配線の確認を Sysmac Studio 上から行う機能
	同期			オンライン接続時に Sysmac Studio のプロジェクトファイルと CPU ユニットのデータを同一にさせる機能
	微分モニタ			接点の立ち上がり、もしくは立ち下がりをモニタする機能
		最大点数		8
	データトレース	種類	トリガトレース	トリガが成立してから、設定したサンプリング数に達するとトレースを自動的に停止する機能
			連続トレース	データトレースの実行を継続し、トレースデータを Sysmac Studio が常時収集する機能
		同時起動最大数		4
		レコード最大数		10,000
		サンプリング最大点数		192 変数
		サンプリングタイミング		指定タスク周期、指定時間、サンプリング命令実行時
		トリガトレース		トリガ条件を設定することで、事象が発生した前後のデータを記録する機能
			トリガ条件	• BOOL 型変数の立ち上がり／立ち下がり • BOOL 型変数以外の定数値比較 比較方法：＝、＞、≧、＜、≦、≠
			ディレー値	トリガ成立前／成立後のサンプリング数割合を設定する機能
	シミュレーション機能			Sysmac Studio 上で CPU ユニットの動作をシミュレートする機能
高信頼機能	自己診断	コントローラ異常	重要度	全停止フォールト、部分停止フォールト、軽度フォールト、監視情報、一般情報
			メッセージ言語最大数	9(Sysmac Studio) 2(NS シリーズ表示器)
		ユーザ異常	機能	任意の異常を設計してあらかじめ登録し、命令実行によって記録を残す機能
			重要度	8 段階
			メッセージ言語最大数	9
セキュリティ機能	顧客資産保護／誤操作防止	CPUユニット名称機能およびシリアル ID 機能		Sysmac Studio からのオンライン接続時に、プロジェクト上の CPU ユニット名称と接続先 CPU ユニットの CPU ユニット名称が一致しているかどうかを確認する機能
		プロテクト機能	ユーザプログラム復元情報なし転送機能	Sysmac Studio から、CPU ユニット内にあるデータを読み出すことができないようにする機能
			CPU ユニットへの書込プロテクト機能	Sysmac Studio ／ SD メモリカードから、CPU ユニット内にあるデータを書き込むことができないようにする機能
			プロジェクトファイル全体のプロテクト機能	Sysmac Studio にて、パスワードによって、.smc ファイルを開くことを禁止する機能
			データプロテクト機能	Sysmac Studio にて、パスワードによって、POU に対するプロテクトをする機能
		操作権限の認証機能		操作ミスによって、装置や人に危害を加える可能性がある場合に、操作権限に応じて、オンラインでの操作機能を制限する機能
			グループ数	5
		ユーザプログラム実行用 ID 認証機能		特定のハードウェア（CPU ユニット）に対しては、Sysmac Studio からユーザプログラム実行用 ID を入力しない限り、ユーザプログラムを実行できないようにする機能
SD メモリカード機能	保存場所			共有フォルダ[*3]
	アプリケーション	SD メモリカード操作命令		ユーザプログラム上の命令から、仮想 SD メモリカードにアクセスする機能
		Sysmac Studio からのファイル操作		ユーザは、仮想 SD メモリカードにコントローラ用ファイルだけでなく、汎用のドキュメントファイルなども保存や読出が可能
		FTP Client/Server 機能からのファイル操作		FTP クライアント機能や FTP サーバー機能による保存や読み出しが可能

項目				形 NY5□2-□□□□
バックアップ機能	SD メモリカードバックアップ機能	操作方法	システム定義変数による指示	システム定義変数の操作で、バックアップや照合を行う機能
			SysmacStudio メモリカード画面	Sysmac Studio の SD メモリカード操作画面からバックアップや照合を行う機能
			専用命令	専用命令からバックアップを行う機能
		プロテクト機能	SD メモリカードへのバックアップ禁止	仮想 SD メモリカードへバックアップするのを禁止する機能
	Sysmac Studio コントローラバックップ機能			Sysmac Studio を使用して、コントローラのバックアップ、リストア、照合を行う機能

*1. インライン ST（ラダー図の中に、ST 言語を記述するラダー図言語の要素）が可能です。
*2. 内部通信ポートだけです。
*3. Windows が動作するストレージデバイス上に存在するフォルダです。コントローラ側は、仮想 SD メモリカードとして扱います。

（注）表の（*）が付いている項目は、バージョンアップで追加された機能です。バージョンアップに関する情報は、📖「A-10 NY シリーズのバージョン関連情報」（P.A-110）を参照してください。

A-2 NY シリーズでのタスク実処理時間の目安の計算

ここでは、NY シリーズシステムでのタスク実処理時間の平均値の目安の計算方法について、説明しています。

タスク実処理時間とタスク実行時間は、実機コントローラで確認することが必要です。詳細は、📖「5-9 タスク設計方法および入出力応答時間」（P.5-56）を参照してください。

使用上の注意

実際のコントローラでは、ユーザプログラムに記載されたロジックの動作有無、通信コマンドの発生やデータリンクの動作有無、データトレースなど各種機能の使用有無、および Windows 側の負荷、産業用モニタの接続有無、PCIe カードの使用有無などによって、タスク実行時間が変動します。
最終的には、必ず実機を使用してあらゆる条件で性能確認を行い、タスク周期オーバーを起こさないことや適切な通信応答性が確保されていることを確認してください。

参考

定周期タスクは、それより実行優先度の高いタスクの実行中には一時停止します。「タスク実処理時間」には、一時停止している時間は含みません。一時停止している時間を含めて、タスクの開始から終了までに実際にかかった時間を示すのが「タスク実行時間」です。「タスク実処理時間」と「タスク実行時間」の違いの詳細は、📖「 タスク実行時間とタスク実処理時間の意味」（P.5-55）を参照してください。

A-2-1 タスク実処理時間の平均値の計算方法

タスク実処理時間の平均値は、I/O リフレッシュ処理時間、ユーザプログラム実行時間、モーション制御処理時間、共通処理時間の合計です。

タスク実処理時間の平均値＝ I/O リフレッシュ処理時間 ＋ ユーザプログラム実行時間
＋ モーション制御処理時間 ＋ 共通処理時間

各処理内容は以下のとおりです。

○：あり、－：なし

処理		処理内容	プライマリ定周期タスク	定周期タスク（実行優先度 16）	定周期タスク（実行優先度 17、18）
I/O リフレッシュ処理		EtherCAT スレーブとのリフレッシュを実行します。	○	－	－
ユーザプログラム実行		• タスクに割り付けられたプログラムを割付順に実行します。	○	○	○
モーション制御処理		• ユーザプログラムからのモーション制御指令を実行します。 • モーション出力処理	○	－	－
共通処理時間	システム共通処理 1	• 変数の更新処理（参照タスクの場合） • モーション入力処理 • データトレース処理	○	○	○
	システム共通処理 2	• 変数の更新処理（更新タスクの場合） • タスク実行との同時性を確保させるためのコントローラ外部からの変数アクセス処理	○	○	○
	システムオーバーヘッド時間	上記以外のシステム共通処理	○	○	○

以降に、各処理時間の計算の目安を示します。

I/O リフレッシュ処理時間

I/O リフレッシュ処理時間は、以下の式で求められます。

I/O リフレッシュ処理時間 =EtherCAT スレーブ処理時間

次に、EtherCAT スレーブ処理時間の求め方を示します。



参考

EtherCAT をスレーブを割り付けていないタスクでは、EtherCAT スレーブ処理時間は「0」となります。

● プライマリ定周期タスクでの EtherCAT スレーブ処理時間

プライマリ定周期タスクでの EtherCAT スレーブ処理時間は、以下の式で求められます。

EtherCAT スレーブ処理時間 [μs] = 0.0008×pDout + 0.0028×pDin + 0.082×pDinout + (1.24×Snum + 0.01×Clen)

pDout	:	プライマリ定周期タスクに割り付けたEtherCATスレーブの出力プロセスデータサイズ合計 [バイト]
pDin	:	プライマリ定周期タスクに割り付けたEtherCATスレーブの入力プロセスデータサイズ合計 [バイト]
pDinout	:	プライマリ定周期タスクに割り付けたEtherCATスレーブごとの出力プロセスデータサイズと入力プロセスデータサイズの大きい方の合計 [バイト]
Snum	:	内蔵 EtherCAT ポートに接続した EtherCAT スレーブのすべての台数 [台]
Clen	:	内蔵 EtherCAT ポートに接続したケーブルの総延長 [m]

ユーザプログラム実行時間

ユーザプログラム実行時間は、命令種類 × 命令数に依存します。

目安として、大きく以下の 3 つの分類での命令数を、計測または予想します。

- 標準的な命令
- 倍精度実数（LREAL）型の四則演算命令
- 倍精度実数（LREAL）型の三角関数命令

また、ラダー図言語と ST 言語では、使用する命令が異なります。それぞれの命令の構成については、「「ラダー図言語の標準的な命令」の命令種類の構成」（P.A-15）、および「「ST 言語の標準的な命令」の命令種類の構成」（P.A-17）を参照してください。

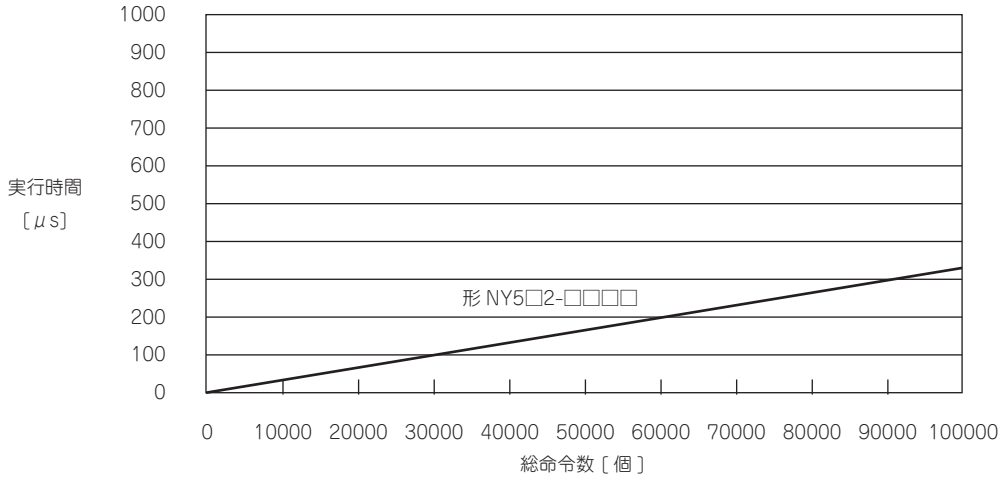
● 簡単な試算

各分類の命令数をもとに、下記グラフから、以下の各命令の実行時間を読み取り、合計します。

- 標準的な命令の実行時間
- 倍精度実数（LREAL）型の四則演算命令の実行時間
- 倍精度実数（LREAL）型の三角関数命令の実行時間

これにより、ユーザプログラムの実行時間を見積もることができます。

ラダー図言語の標準的な命令の実行時間

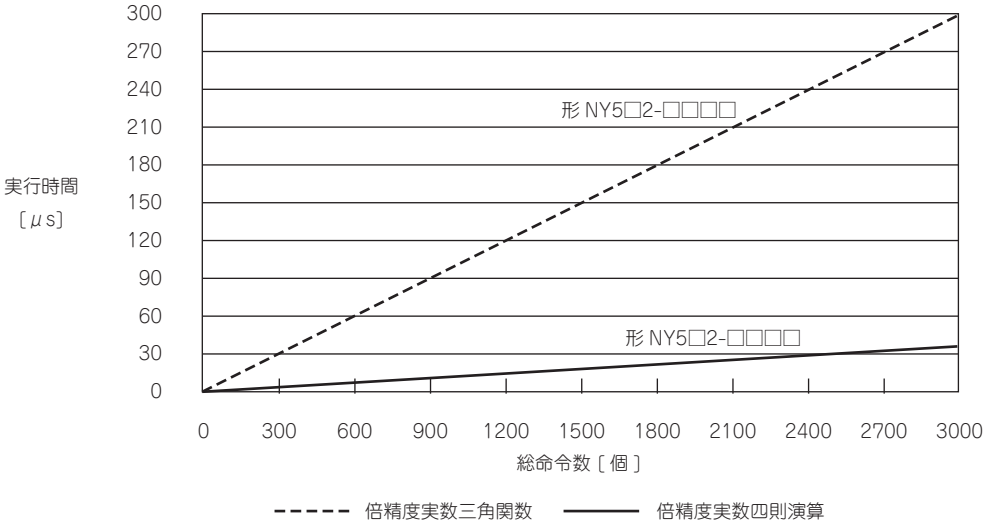


•「ラダー図言語の標準的な命令」の命令種類の構成

この構成での命令の実行率は 20％です。

命令の種類	命令名	命令個数の比率 [%]	命令の種類ごとの実行時間比率 [%]
ラダー図命令	LD, AND, OUT, SET, RESET	81.0	40.2
比較命令	EQ, LT	4.1	8.3
タイマ命令・カウンタ命令	Timer, TON/TOF, CTU/CTD	1.6	7.3
算術命令	+, -, *, /, ADD, SUB, MUL, DIV	2.4	6.5
BCD 変換命令・データ変換命令	INT_TO_DINT, WORD_BCD_TO_UINT	0.2	1.2
ビット列演算命令	AND, OR	6.2	13.0
データ転送命令	MOVE	4.6	23.5
合計		100.0	100.0

ラダー図言語の倍精度実数（LREAL）型の四則演算命令と三角関数命令の実行時間



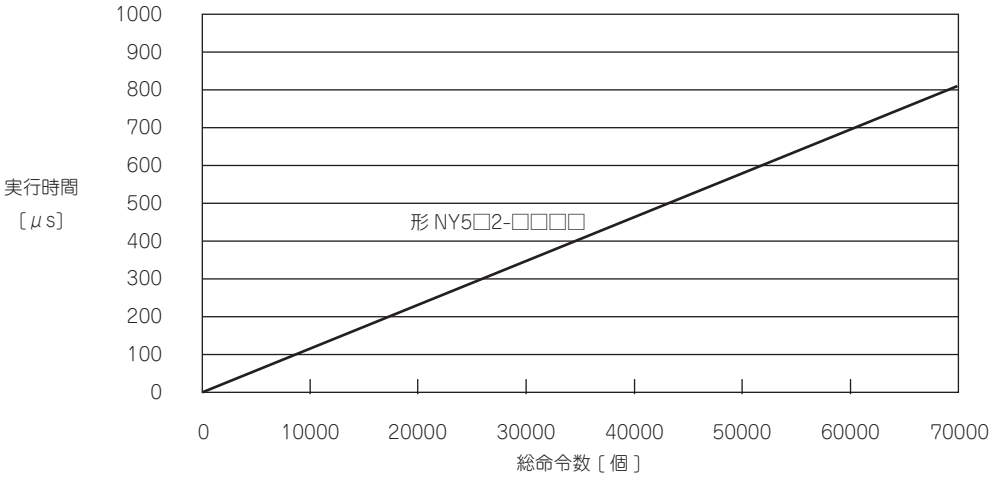
- 「倍精度実数の四則演算」の構成

命令名	命令個数の比率 [%]
倍精度実数の加算	20.0
倍精度実数の減算	20.0
倍精度実数の乗算	30.0
倍精度実数の除算	30.0
合計	100.0

- 「倍精度実数の三角関数」の構成

命令名	命令個数の比率 [%]
倍精度実数の Sin	16.7
倍精度実数の Cos	16.7
倍精度実数の Tan	16.7
倍精度実数の Sin^{-1}	16.7
倍精度実数の Cos^{-1}	16.7
倍精度実数の Tan^{-1}	16.7
合計	100.0

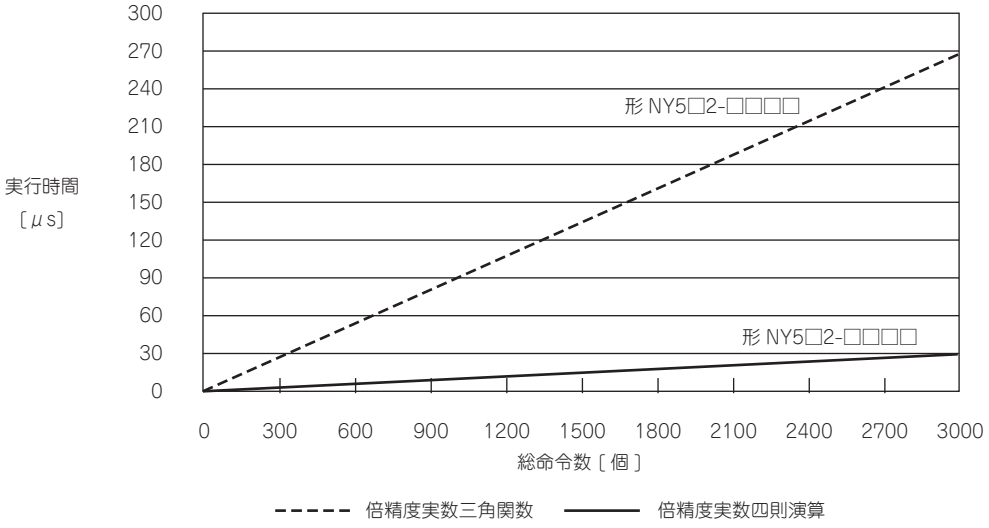
ST 言語の標準的な命令の実行時間



• 「ST 言語の標準的な命令」の命令種類の構成

命令の種類	命令名	命令個数の比率 [%]	命令の種類ごとの実行時間比率 [%]
ST 構文命令	IF ELSIF END_IF	75.4	41.6
比較命令	EQ, LT	5.2	8.7
タイマ命令・カウンタ命令	Timer, TON/TOF, CTU/CTD	2.1	18.8
算術命令	+, -, *, /	3.1	10.2
BCD 変換命令・データ変換命令	INT_TO_DINT, WORD_BCD_TO_UINT	0.2	1.6
ビット列演算命令	AND, OR	8.0	11.7
データ転送命令	:=	5.9	7.3
合計		100.0	100.0

ST 言語の倍精度実数（LREAL）型の四則演算命令と三角関数命令の実行時間



• 「倍精度実数の四則演算」の構成

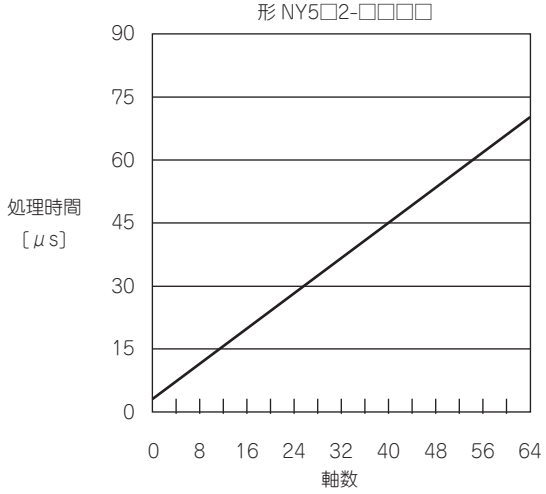
命令名	命令個数の比率 [%]
倍精度実数の加算	20.0
倍精度実数の減算	20.0
倍精度実数の乗算	30.0
倍精度実数の除算	30.0
合計	100.0

• 「倍精度実数の三角関数」の構成

命令名	命令個数の比率 [%]
倍精度実数の Sin	16.7
倍精度実数の Cos	16.7
倍精度実数の Tan	16.7
倍精度実数の Sin^{-1}	16.7
倍精度実数の Cos^{-1}	16.7
倍精度実数の Tan^{-1}	16.7
合計	100.0

モーション制御処理時間

モーション制御処理時間は、使用するサーボの軸数（仮想軸を含む）に依存します。
使用するサーボ軸数をもとに、下記グラフから、モーション制御処理時間を読み取ります。



共通処理時間

共通処理時間は、システムオーバーヘッド、システム共通処理 1 およびシステム共通処理 2 を合計して、以下の値となります。共通処理時間は、タスク種類によって異なります。

タスク種類	共通処理時間 [μs]（参考値） 形 NY5 □ 2- □□□□
プライマリ定周期タスク	258
定周期タスク（実行優先度 16、17、18）	20

A-2-2　タスク実処理時間の平均値の計算とタスク周期の設定の例

タスク実処理時間の平均値の計算例

形 NY5□2-□□□□ のユニットバージョン Ver.1.12 を使用する場合で、以下の条件のときのタスク実処理時間の平均値を求めます。
タスク種類は、プライマリ定周期タスクとします。

項目		条件	
使用するスレーブ／ユニット	EtherCAT スレーブ	入力スレーブ	形 GX-ID1611(Ver.1.1)：1 台
		出力スレーブ	形 GX-OD1611(Ver.1.1)：1 台
		サーボドライバ	形 R88D-1SN □□□ -ECT：4 台
		EtherCAT スレーブターミナル：1 台	
		EtherCAT カプラユニット	形 NX-ECC203：1 台
		DC 入力ユニット	形 NX-ID5342：1 台
		トランジスタ出力ユニット	形 NX-OD5121：1 台
		アナログ入力ユニット	形 NX-AD3608：1 台
		アナログ出力ユニット	形 NX-DA2605：1 台
		コミュニケーション インタフェースユニット	形 NX-CIF101：1 台 形 NX-CIF105：1 台
ユーザプログラム：	使用言語	ラダー図言語	
	標準的な命令構成	命令数：5,000 個	
	倍精度実数の四則演算	命令数：200 個	
	倍精度実数の三角関数	命令数：100 個	
モーション制御処理	軸数	4 軸	

（注） 内蔵 EtherCAT ポートに接続したケーブルの総延長は 10[m] とします。

● I/O リフレッシュ時間

EtherCAT スレーブ処理時間：

入力スレーブ形 GX-ID1611(Ver.1.1)、出力スレーブ形 GX-OD1611(Ver.1.1)、サーボドライバ形 R88D-1SN □□□ -ECT、および上記に示した構成での EtherCAT スレーブターミナルの、出力プロセスデータサイズ pDout、入力プロセスデータサイズ pDin、出力データサイズと入力データサイズの大きい方 pDinout の値は、以下のとおりです。

EtherCAT スレーブ	出力プロセスデータサイズ pDout [バイト]	入力プロセスデータサイズ pDin [バイト]	出力データサイズと入力データサイズの大きい方 pDinout
形 GX-ID1611(Ver.1.1)	0	3	3
形 GX-OD1611(Ver.1.1)	2	1	2
形 R88D-1SN □□□ -ECT	23	26	26
EtherCAT スレーブターミナル合計	62	88	88
形 NX-ECC203	0	18	－
形 NX-ID5342	0	2	
形 NX-OD5121	2	0	
形 NX-AD3608	0	8	
形 NX-DA2605	4	0	
形 NX-CIF101	28	30	
形 NX-CIF105	28	30	

pDout、pDin、および pDinout のそれぞれの合計は、次の値となります。

pDout=2 + 23×4 + 62=156[バイト]
pDin=3 + 1 + 26×4 + 88=196[バイト]
pDinout=3 + 2 + 26×4 + 88=197[バイト]

これらの値より、I/O リフレッシュ時間は、以下の式で求められます。
I/O リフレッシュ処理時間
=EtherCAT スレーブ処理時間
=0.0008×pDout + 0.0028×pDin + 0.082×pDinout + (1.24×Snum + 0.01×Clen)
=0.0008×156 + 0.0028×196 + 0.082×197 + (1.24×7 + 0.01×10)
=0.1248 + 0.5488 + 16.154 + (8.68 + 0.1)
≒ 26[μs]

● ユーザプログラム実行時間

各グラフから、以下の数値を読み取ります。

- 標準的な命令構成
  ラダー図言語の標準的な命令の実行時間グラフより、形 NY5□2-□□□□□ の命令数 5000 個のユーザプログラム実行時間は、16.5μs。
- 倍精度実数の四則演算
  ラダー図言語の倍精度実数（LREAL）型の四則演算命令と三角関数命令の実行時間グラフより、形 NY5□2-□□□□□ の命令数 200 個のユーザプログラム実行時間は、2.5μs。
- 倍精度実数の三角関数
  ラダー図言語の倍精度実数（LREAL）型の四則演算命令と三角関数命令の実行時間グラフより、形 NY5□2-□□□□□ の命令数 100 個のユーザプログラム実行時間は、10μs。

したがって、ユーザプログラム実行時間は、上記の合計で以下の値になります。
ユーザプログラム実行時間 =16.5 + 2.5 + 10
=29[μs]

● モーション制御処理時間

モーション制御処理の実行時間グラフより、形 NY5□2-□□□□□ の 4 軸のモーション制御処理の実行時間を、7μs と読み取ります。

● 共通処理時間

タスク種類はプライマリ定周期タスクなので、形 NY5□2-□□□□□ の共通処理時間は 258μs となります。

したがって、タスク実処理時間の平均値は、以下の式で求められます。

タスク実処理時間の平均値 =I/O リフレッシュ時間+ユーザプログラム実行時間
+ モーション制御処理 + 共通処理時間
=26 + 29 + 7 + 258
=320[μs]

タスク周期の設定例

上で求めたタスク実処理時間の平均値をもとに、タスク周期を設定します。タスク種類は、プライマリ定周期タスクです。
タスク周期の値は、計算で求めたタスク実処理時間の平均値よりも大きくします。その際、余裕を見込んで、タスク周期の値を、タスク実処理時間の平均値の 1.1 倍よりも大きくしてください。

タスク周期≧タスク実処理時間の平均値 ×1.1

上で求めたタスク実処理時間の平均値は 320μs ですから、その 1.1 倍より大きな値として、タスク周期を 500μs に設定します。

実際のコントローラでは、ユーザプログラムに記載されたロジックの動作有無、通信コマンドの発生やデータリンクの動作有無、データトレースなど各種機能の使用有無、および Windows 側の負荷、産業用モニタの接続有無、PCIe カードの使用有無などによって、タスク実行時間は変動します。また、定周期タスクの場合は、それより実行優先度の高いタスクの動作有無によってもタスク実行時間が変動します。
実機のコントローラを使用して、タスク実行時間モニタをもとに、タスク実行時間の検証を必ず行ってください。

A-3 システム定義変数一覧

システムで特定の機能が割り付けられた変数です。
あらかじめ、グローバル変数テーブルまたは各 POU のローカル変数テーブルに登録されています。
変数名は固定です。“_（アンダーバー）” で始まるものと、“P_” で始めるものから成ります。

読出専用のシステム定義変数と、読出／書込可能なシステム定義変数があります。

ユーザプログラム、外部からの通信または Sysmac Studio、NS/NA シリーズ表示器から、読み書きして使用します。

システム定義変数は、基本的に機能モジュールごとに分類しています。
変数名は、以下のカテゴリ名から始まります。

機能モジュール	カテゴリ名
NY シリーズコントローラのシステム全体	_（なし）
PLC 機能モジュール	_PLC
モーション制御機能モジュール	_MC
EtherCAT マスタ機能モジュール	_EC
EtherNet/IP 機能モジュール	_EIP、_EIP1、_EIPIn1

一覧表の見方は、以下のとおりです。

変数名	名称	機能	データ型	値の範囲	参照ページ
［システム定義変数名です。先頭にカテゴリ名が付きます］	［名称です］	［機能を説明します］	［変数のデータ型を示します］	［変数の値の範囲を示します］	［個別仕様表のページを示します］

変数名欄の () 内は、そのシステム定義変数が追加された NY シリーズコントローラのユニットバージョンを示します。

使用上の注意

コントローラのシリーズによって、対応していなかったり、配列数などの仕様が異なったりするシステム定義変数があります。詳細は、🕮「A-4 システム定義変数の個別仕様」（P.A-46）で個別の仕様を確認してください。

A-3-1 NY シリーズコントローラのシステム全体（カテゴリ名：なし）

● 機能分類：時計関連

変数名	名称	機能	データ型	値の範囲	参照ページ
_CurrentTime	システム時刻	コントローラの内部時計データが格納されます。	DATE_AND_TIME	DT#2000-01-01-00:00:00 ～ DT#2099-12-31-23:59:59	A-47

● 機能分類：タスク関連

変数名	名称	機能	データ型	値の範囲	参照ページ
_＜タスク名称＞_Active	タスク実行中フラグ	タスク実行中のとき TRUE となります。タスク実行中でないとき FALSE となります。 （注） 本システム定義変数は、ユーザプログラム上では使用できません。Sysmac Studio のデータトレース機能でタスクの状態を参照するときにだけ使用します。	BOOL	TRUE、FALSE	A-47
_＜タスク名称＞_LastExecTime	前回タスク実行時間	前回に実行したタスク実行時間（0.1 μs 単位）を示します。 （注） 本システム定義変数は、ユーザプログラム上では使用できません。Sysmac Studio のデータトレース機能でタスクの状態を参照するときにだけ使用します。	TIME	データ型に従う	A-47
_＜タスク名称＞_MaxExecTime	タスク実行時間最大値	タスク実行時間の最大値（0.1 μs 単位）を示します。 （注） 本システム定義変数は、ユーザプログラム上では使用できません。Sysmac Studio のデータトレース機能でタスクの状態を参照するときにだけ使用します。	TIME	データ型に従う	A-47
_＜タスク名称＞_MinExecTime	タスク実行時間最小値	タスク実行時間の最小値（0.1 μs 単位）を示します。 （注） 本システム定義変数は、ユーザプログラム上では使用できません。Sysmac Studio のデータトレース機能でタスクの状態を参照するときにだけ使用します。	TIME	データ型に従う	A-47
_＜タスク名称＞_ExecCount	タスク実行回数	タスクの実行回数を示します。4294967295 回を超えたら、0 に戻して、カウントを継続します。 （注） 本システム定義変数は、ユーザプログラム上では使用できません。Sysmac Studio のデータトレース機能でタスクの状態を参照するときにだけ使用します。	UDINT	データ型に従う	A-48
_＜タスク名称＞_Exceeded	タスク周期超過フラグ	タスク周期を超過したら TRUE となります。タスク周期以内にタスク実行が完了したら FALSE となります。 （注） 本システム定義変数は、ユーザプログラム上では使用できません。Sysmac Studio のデータトレース機能でタスクの状態を参照するときにだけ使用します。	BOOL	TRUE、FALSE	A-48
_＜タスク名称＞_ExceedCount	タスク周期超過回数	周期を超過した回数が格納されます。現在値がデータ型の最大値を超えたら、現在値を 0 に戻して、カウントを継続します。 4294967295 回を超えたら、0 に戻して、カウントを継続します。 （注） 本システム定義変数は、ユーザプログラム上では使用できません。Sysmac Studio のデータトレース機能でタスクの状態を参照するときにだけ使用します。	UDINT	データ型に従う	A-48

● 機能分類：異常関連

変数名	名称	機能	データ型	値の範囲	参照ページ
_ErrSta	コントローラ異常ステータス	コントローラ異常発生中は TRUE となります。 コントローラ異常未発生中は FALSE となります。 （注） リアルタイム性、および各機能モジュールの異常要因を表すステータスとの同時性の観点から、ユーザプログラム上では使用できません。通信を介して外部から状態を参照するときにだけ使用します。各ビットの意味は後述の「異常ステータスの各ビットの意味」を参照してください。	WORD	16#0000 ～ 16#C0F0	A-49
_AlarmFlag	ユーザ異常ステータス	ユーザ異常発生中はイベント重要度に該当するビットが TRUE となります。 User fault Level 1 ～ 8 が、ビット 00 ～ 07 に対応します。 ユーザ異常未発生中は 16 進 0000 となります。	WORD	16#0000 ～ 16#00FF	A-49

● 機能分類：SD メモリカード関連

変数名	名称	機能	データ型	値の範囲	参照ページ
_Card1Ready	SD メモリカード使用可フラグ	NY シリーズでは、仮想 SD メモリカードの認識ができたとき、TRUE となります。 認識ができなかったとき、FALSE となります。TRUE：使用可 FALSE：使用不可	BOOL	TRUE、FALSE	A-49
_Card1Protect	SD メモリカードライトプロテクトフラグ	NY シリーズでは、使用しません。 常に FALSE となります。	BOOL	TRUE、FALSE	A-49
_Card1Err	SD メモリカードエラーフラグ	NY シリーズでは、使用しません。 常に FALSE となります。	BOOL	TRUE、FALSE	A-50
_Card1Access	SD メモリカードアクセス中フラグ	NY シリーズでは、使用しません。 常に FALSE となります。	BOOL	TRUE、FALSE	A-50
_Card1Deteriorated	SD メモリカード寿命警告フラグ	NY シリーズでは、使用しません。 常に FALSE となります。	BOOL	TRUE、FALSE	A-50
_Card1PowerFail	SD メモリカードアクセス中電源断フラグ	仮想 SD メモリカードアクセス中にコントローラの電源断が発生したとき、TRUE となります。 TRUE： 仮想メモリカードアクセス中に電源断があった FALSE：正常	BOOL	TRUE、FALSE	A-50
_Card1Capacity（Ver.1.21）	SD メモリカードストレージ容量	仮想 SD メモリカードの全容量を示します。単位は MiB（1MiB=1,048,576Byte）です。値は 60 秒ごとに更新されます。 仮想 SD メモリカードが無効の場合、“0” になります。	UDINT	データ型に従う	A-50
_Card1Used（Ver.1.21）	SD メモリカードストレージ使用量	仮想 SD メモリカードの使用量を示します。単位は MiB（1MiB=1,048,576Byte）です。値は 60 秒ごとに更新されます。 仮想 SD メモリカードが無効の場合、“0” になります。	UDINT	データ型に従う	A-51

変数名 メンバ名	名称	機能	データ型	値の範囲	参照ページ
_Card1BkupCmd	SD メモリカード バックアップ指示		_sBKUP_ CMD		A-51
ExecBkup	バックアップ起動フラグ	コントローラのデータを仮想 SD メモリカードへバックアップするときに、TRUE にします。 （注） 本システム定義変数は、ユーザプログラム上では使用できません。CIP メッセージ通信によって、表示器や上位パソコンから指令するときに使用します。	BOOL	TRUE、 FALSE	A-51
CancelBkup	バックアップ取消フラグ	仮想 SD メモリカードへのバックアップを中止するときに、TRUE にします。 （注） 本システム定義変数は、ユーザプログラム上では使用できません。CIP メッセージ通信によって、表示器や上位パソコンから指令するときに使用します。	BOOL	TRUE、 FALSE	A-51
ExecVefy	照合起動フラグ	コントローラと仮想 SD メモリカード内のバックアップファイルを照合するときに、TRUE にします。 （注） 本システム定義変数は、ユーザプログラム上では使用できません。CIP メッセージ通信によって、表示器や上位パソコンから指令するときに使用します。	BOOL	TRUE、 FALSE	A-51
CancelVefy	照合取消フラグ	コントローラと仮想 SD メモリカード内のバックアップファイルの照合を中止するときに、TRUE にします。 （注） 本システム定義変数は、ユーザプログラム上では使用できません。CIP メッセージ通信によって、表示器や上位パソコンから指令するときに使用します。	BOOL	TRUE、 FALSE	A-51
DirName	ディレクトリ名	バックアップ対象の仮想 SD メモリカードのディレクトリ名を指定します。 （注） 本システム定義変数は、ユーザプログラム上では使用できません。CIP メッセージ通信によって、表示器や上位パソコンから指令するときに使用します。	STRING(64)	データ型に従う	A-52
_Card1BkupSta	SD メモリカード バックアップ ステータス		_sBKUP_ STA		A-52
Done	完了フラグ	バックアップ実行完了のとき、TRUE となります。 （注） 本システム定義変数は、ユーザプログラム上では使用できません。CIP メッセージ通信によって、表示器や上位パソコンから指令するときに使用します。	BOOL	TRUE、 FALSE	A-52
Active	実行中フラグ	バックアップ実行中のとき、TRUE となります。 （注） 本システム定義変数は、ユーザプログラム上では使用できません。CIP メッセージ通信によって、表示器や上位パソコンから指令するときに使用します。	BOOL	TRUE、 FALSE	A-52
Err	エラーフラグ	バックアップ処理が異常終了したとき、TRUE となります。 （注） 本システム定義変数は、ユーザプログラム上では使用できません。CIP メッセージ通信によって、表示器や上位パソコンから指令するときに使用します。	BOOL	TRUE、 FALSE	A-52

変数名 メンバ名	名称	機能	データ型	値の範囲	参照ページ
_Card1VefySta	SD メモリカード照合ステータス		_sVEFY_STA		A-52
Done	完了フラグ	照合実行完了のとき、TRUE となります。 （注） 本システム定義変数は、ユーザプログラム上では使用できません。CIP メッセージ通信によって、表示器や上位パソコンから指令するときに使用します。	BOOL	TRUE、FALSE	A-52
Active	実行中フラグ	照合実行中のとき、TRUE となります。 （注） 本システム定義変数は、ユーザプログラム上では使用できません。CIP メッセージ通信によって、表示器や上位パソコンから指令するときに使用します。	BOOL	TRUE、FALSE	A-52
VefyRslt	照合結果フラグ	照合一致のとき TRUE、照合不一致のとき FALSE となります。 （注） 本システム定義変数は、ユーザプログラム上では使用できません。CIP メッセージ通信によって、表示器や上位パソコンから指令するときに使用します。	BOOL	TRUE、FALSE	A-53
Err	エラーフラグ	照合処理が異常終了したとき、TRUE となります。 （注） 本システム定義変数は、ユーザプログラム上では使用できません。CIP メッセージ通信によって、表示器や上位パソコンから指令するときに使用します。	BOOL	TRUE、FALSE	A-52

● 機能分類：バックアップ関連

変数名	名称	機能	データ型	値の範囲	参照ページ
_BackupBusy	バックアップ関連機能実行中フラグ	バックアップ、リストア、照合の実行中のとき、TRUE となります。	BOOL	TRUE、FALSE	A-53

● 機能分類：電源管理関連

変数名	名称	機能	データ型	値の範囲	参照ページ
_PowerOnHour	通電時間	通電時間を示します。 コントローラの通電時間を、1 時間単位で格納します。 リセットする場合、0 を格納してください。 4294967295 となった場合、それ以上は更新されません。 電源 ON 時に初期化されません。	UDINT	0 ～ 4294967295	A-53
_PowerOnCount	電源断発生回数	電源断の発生回数を示します。 コントローラの最初の電源投入から、電源断発生ごとに積算（+1）されます。 リセットする場合は、0 を格納してください。 4294967295 となった場合、それ以上は更新されません。 電源 ON 時に、初期化されません。	UDINT	0 ～ 4294967295	A-54
_RetainFail	電断保持失敗フラグ	以下のとき、TRUE（電断保持失敗）となります。 • 電源投入時、バッテリバックアップメモリ／不揮発性メモリのチェックでエラーのとき 以下のとき、FALSE（電断保持失敗なし）となります。 • 電源投入時、バッテリバックアップメモリ／不揮発性メモリのメモリチェックで OK のとき • ユーザプログラムがダウンロードされたとき • メモリオールクリアを実行したとき （注） 絶対値エンコーダ原点位置オフセットデータが保持できなかったときは、このフラグでなく、軸変数の異常ステータスに反映されます。	BOOL	TRUE、FALSE	A-54
_SelfTest_UPSSignal	UPS 信号検知フラグ	UPS からの一時電断信号を検知した場合、TRUE となります。	BOOL	TRUE、FALSE	A-54
_RequestShutdown	シャットダウン要求フラグ	稼働中に電源ボタンが押下された場合、TRUE となります。	BOOL	TRUE、FALSE	A-54

● 機能分類：OS(Windows) 関連

変数名	名称	機能	データ型	値の範囲	参照ページ
_OSRunning	OS 稼働中フラグ	コントローラが OS(Windows) の状態を稼働中と観測した場合、TRUE となります。	BOOL	TRUE、FALSE	A-54
_OSHalted	OS 停止中フラグ	コントローラが OS(Windows) の状態を停止中と観測した場合、TRUE となります。	BOOL	TRUE、FALSE	A-55
_OSErrorState	OS 異常状態フラグ	コントローラが OS(Windows) の状態を異常発生中と判断した場合、TRUE となります。	BOOL	TRUE、FALSE	A-55

● 機能分類：プログラミング関連

変数名	名称	機能	データ型	値の範囲	参照ページ
P_On	常時 ON フラグ	常に TRUE を示すフラグです。	BOOL	TRUE	A-55
P_Off	常時 OFF フラグ	常に FALSE を示すフラグです。	BOOL	FALSE	A-55
P_CY	キャリーフラグ	一部の命令にて更新するフラグです。	BOOL	TRUE、FALSE	A-55
P_First_RunMode	運転開始時 1 周期 ON フラグ	コントローラの動作モードを、プログラムモードから運転モードに変更した際に、当該プログラムが実行中である場合、1 タスク周期だけ TRUE となります。 当該プログラムが実行中でなければ、FALSE のままです。 コントローラが運転を開始したときに初期処理を行う場合などに使用します。 （注） 本システム定義変数は、ファンクション内では使用できません。	BOOL	TRUE、FALSE	A-55
P_First_Run	プログラム起動時 1 周期 ON フラグ	当該プログラムが、実行を開始した際に、1 タスク周期だけ TRUE となります。 プログラムが実行を開始するごとに初期処理を行う場合などに使用します。 （注） 本システム定義変数は、ファンクション内では使用できません。	BOOL	TRUE、FALSE	A-56
P_PRGER	命令エラー発生フラグ	当該プログラムまたは当該プログラムから呼び出した FB や FUN で、命令エラーが発生したときに TRUE となり続けるフラグです。一度 TRUE となると、ユーザプログラムによって FALSE にされるまで TRUE となり続けます。	BOOL	TRUE、FALSE	A-56

● 機能分類：バージョン管理関連

変数名	名称	機能	データ型	値の範囲	参照ページ
_UnitVersion	ユニットバージョン	コントローラのユニットバージョンが格納されます。 要素番号 0 に、ユニットバージョンの整数部が格納されます。 要素番号 1 に、ユニットバージョンの小数部が格納されます。 例 1) ユニットバージョン Ver.1.08 の場合、要素番号 0 に 1、要素番号 1 に 8 が格納されます。 例 2) ユニットバージョン Ver.1.10 の場合、要素番号 0 に 1、要素番号 1 に 10 が格納されます。	ARRAY[0..1] OF USINT	0 ～ 99	A-56
_HardwareRevision	ハードウェアリビジョン	コントローラのハードウェアリビジョンが格納されます。 ハードウェアリビジョンが「なし」の場合は「'-'」、それ以外の場合は、「'A' ～ 'Z'」が格納されます。	STRING[2]	'-'、'A' ～ 'Z'	A-56

● 機能分類：自己診断関連

変数名	名称	機能	データ型	値の範囲	参照ページ
_SelfTest_HighTemperature	CPU ユニット内部温度オーバーフラグ	コントローラ内部温度が高温のとき、TRUE となります。	BOOL	TRUE、FALSE	A-57
_SelfTest_LowBattery	バッテリ電圧低下フラグ	バッテリが外れているか、バッテリ電圧が低下しているとき、TRUE となります。	BOOL	TRUE、FALSE	A-57
_SelfTest_LowFanRevolution	FAN 回転数低下フラグ	FAN が外れているか、FAN の回転数が低下しているとき、TRUE となります。	BOOL	TRUE、FALSE	A-57

● 機能分類：PLC 内蔵関連

変数名	名称	機能	データ型	値の範囲	参照ページ
_DeviceOutHoldCfg (Ver.1.14)	デバイス出力保持設定	動作モード変更時またはダウンロード時に、対象のデバイス出力を保持する場合、16#A5A5 とします。 16#A5A5 以外では、動作モード変更時またはダウンロード時に、対象のデバイス出力は初期化されます。	WORD	16#0000 ～ 16#FFFF	A-57
_DeviceOutHoldStatus (Ver.1.14)	デバイス出力保持状態	動作モード変更時またはダウンロード時に、対象のデバイス出力が保持された場合、TRUE となります。 デバイス出力保持設定が 16#A5A5 以外のとき、または全停止フォールトレベルのコントローラ異常発生時に、対象のデバイス出力が初期化された場合、FALSE となります。	BOOL	TRUE、FALSE	A-57

A-3-2　PLC 機能モジュール（カテゴリ名：_PLC）

● 機能分類：デバッグ関連

変数名 メンバ名	名称	機能	データ型	値の範囲	参照ページ
_PLC_TraceSta[0..3]			_sTRACE_STA		A-58
.IsStart	トレース実行中 ON フラグ	トレースを開始したとき、TRUE となります。 （注）本システム定義変数は、ユーザプログラム上では使用できません。Sysmac Studio でデータトレースの状態をウォッチするときにだけ使用します。	BOOL	TRUE、FALSE	A-58
.IsComplete	トレース完了時 ON フラグ	トレース実行が完了したとき、TRUE となります。 （注）本システム定義変数は、ユーザプログラム上では使用できません。Sysmac Studio でデータトレースの状態をウォッチするときにだけ使用します。	BOOL	TRUE、FALSE	A-58
.IsTrigger	トレーストリガモニタフラグ	トレーストリガ条件が成立したとき、TRUE となります。 次にトレースを開始したとき、FALSE となります。 （注）本システム定義変数は、ユーザプログラム上では使用できません。Sysmac Studio でデータトレースの状態をウォッチするときにだけ使用します。	BOOL	TRUE、FALSE	A-58
.ParamErr	トレースパラメータ異常フラグ	トレース開始時にトレース設定が異常のとき、TRUE となります。 正常のとき、FALSE となります。 （注）本システム定義変数は、ユーザプログラム上では使用できません。Sysmac Studio でデータトレースの状態をウォッチするときにだけ使用します。	BOOL	TRUE、FALSE	A-58

● 機能分類：異常関連

変数名	名称	機能	データ型	値の範囲	参照ページ
_PLC_ErrSta	PLC 機能モジュール異常ステータス	PLC 機能モジュールにかかわるコントローラ異常発生中は TRUE となります。 PLC 機能モジュールにかかわるコントローラ異常未発生中は FALSE となります。 各ビットの意味は後述の「異常ステータスの各ビットの意味」を参照してください。	WORD	16#0000 ～ 16#00F0	A-59

A-3-3　モーション制御機能モジュール（カテゴリ名：_MC）

● 機能分類：モーション制御機能関連

変数名	名称	機能	データ型	値の範囲	参照ページ
_MC_ErrSta	モーション制御機能モジュールの異常ステータス	モーション制御機能モジュールで検知した異常の状態を表します。 ユーザプログラム上で使用（直接参照）できます。 各ビットの意味は後述の「異常ステータスの各ビットの意味」を参照してください。	WORD	16#0000 ～ 16#40F0	A-60
_MC_ComErrSta	共通異常ステータス	モーション制御において、共通の処理で検知した異常の状態を表します。 ユーザプログラム上で使用（直接参照）できます。各ビットの意味は後述の「異常ステータスの各ビットの意味」を参照してください。	WORD	16#0000 ～ 16#00F0	A-60
_MC_AX_ErrSta	軸異常ステータス	軸単位で検知した異常の状態を表します。最大 64 軸を表示します。 ユーザプログラム上で使用（直接参照）できます。各ビットの意味は後述の「異常ステータスの各ビットの意味」を参照してください。	ARRAY [0..63] OF WORD	16#0000 ～ 16#00F0	A-60
_MC_GRP_ErrSta	軸グループ異常ステータス	軸グループ単位で検知した異常の状態を表します。最大 32 軸グループを表示します。 ユーザプログラム上で使用（直接参照）できます。各ビットの意味は後述の「異常ステータスの各ビットの意味」を参照してください。	ARRAY [0..31] OF WORD	16#0000 ～ 16#00F0	A-60
_MC_COM	共通変数	モーション制御機能モジュール共通の状態モニタをします。 構造体のメンバの詳細は、🕮『NY シリーズ コマンドリファレンスマニュアル モーション編（SBCE-380）』を参照してください。	_sCOMMON_REF	－	A-61
_MC_GRP	軸グループ変数	NY シリーズコントローラの場合、モモーション制御命令での軸グループ指定、多軸協調制御の状態モニタ、多軸協調制御の設定モニタをします。 Sysmac Studio で「軸グループ」を作成すると、ユーザ定義の別名の軸グループ変数が生成されます。 通常は、別名の軸グループ変数を使用します。 構造体のメンバの詳細は、🕮『NY シリーズ コマンドリファレンスマニュアル モーション編（SBCE-380）』を参照してください。	ARRAY[0..31] OF _sGROUP_REF	－	A-61
_MC_AX	軸変数	NY シリーズコントローラの場合、モーション制御命令での軸指定、単軸制御の状態モニタ、単軸制御の設定モニタをします。 Sysmac Studio で「軸」を作成すると、ユーザ定義の別名の軸変数が生成されます。 通常は、別名の軸変数を使用します。 構造体のメンバの詳細は、🕮『NY シリーズ コマンドリファレンスマニュアル モーション編（SBCE-380）』を参照してください。	ARRAY[0..63] OF _sAXIS_REF	－	A-61

A-3-4　EtherCAT マスタ機能モジュール（カテゴリ名：_EC）

● 機能分類：EtherCAT 通信の異常関連

変数名	名称	機能	データ型	値の範囲	参照ページ
_EC_ErrSta	内蔵 EtherCAT 異常	EtherCAT マスタ機能モジュールの異常を集約したシステム定義変数です。 各ビットの意味は後述の「異常ステータスの各ビットの意味」を参照してください。	WORD	16#0000 ～ 16#40F0	A-62
_EC_PortErr	通信ポート異常	EtherCAT マスタ用通信ポートの異常を集約したシステム定義変数です。 各ビットの意味は後述の「異常ステータスの各ビットの意味」を参照してください。	WORD	16#0000 ～ 16#00F0	A-62
_EC_MstrErr	マスタ異常	EtherCAT マスタの異常と EtherCAT マスタが検知したスレーブの異常を集約したシステム定義変数です。 各ビットの意味は後述の「異常ステータスの各ビットの意味」を参照してください。	WORD	16#0000 ～ 16#00F0	A-62
_EC_SlavErr	スレーブ異常	EtherCAT スレーブ全体の異常ステータスを集約したシステム定義変数です。 各ビットの意味は後述の「異常ステータスの各ビットの意味」を参照してください。	WORD	16#0000 ～ 16#00F0	A-62
_EC_SlavErrTbl	スレーブ異常テーブル	EtherCAT スレーブ単位の異常ステータスを通知します。 本異常ステータスは実システム構成のスレーブ単位の異常状態を示します。本変数の配列は EtherCAT スレーブのノードアドレス（1 ～ 512）に対応し、異常が発生しているスレーブを示します。 各ビットの意味は後述の「異常ステータスの各ビットの意味」を参照してください。	Array [1..512] OF WORD	16#0000 ～ 16#00F0	A-63
_EC_MacAdrErr	MAC アドレス異常	イベント「MAC アドレス異常」(14400000Hex) が発生した場合、TRUE となります。	BOOL	TRUE、FALSE	A-63
_EC_LanHwErr	通信コントローラ異常	イベント「通信コントローラ異常」(047C0000Hex) が発生した場合、TRUE となります。	BOOL	TRUE、FALSE	A-63
_EC_LinkOffErr	リンクオフ異常	イベント「リンクオフ異常」(84200000Hex) が発生した場合、TRUE となります。	BOOL	TRUE、FALSE	A-63
_EC_NetCfgErr	ネットワーク構成情報異常	イベント「ネットワーク構成情報異常」(34400000Hex) が発生した場合、TRUE となります。	BOOL	TRUE、FALSE	A-63
_EC_NetCfgCmpErr	ネットワーク構成照合異常	以下のいずれかのイベントが発生した場合、TRUE となります。 •「ネットワーク構成照合異常」(84220000Hex) •「ネットワーク構成照合異常 (スレーブ不在)」(84380000Hex) •「ネットワーク構成照合異常 (スレーブ誤接続)」(84320003Hex) •「ネットワーク構成照合異常 (スレーブ不一致)」(84330004Hex) •「ネットワーク構成照合異常 (リング誤配線)」(843A0000Hex)	BOOL	TRUE、FALSE	A-64
_EC_NetTopologyErr	ネットワーク構成異常	以下のいずれかのイベントが発生した場合、TRUE となります。 •「ネットワーク構成異常」(84210000Hex) •「不正な配線検知」(843C0000Hex)	BOOL	TRUE、FALSE	A-64

変数名	名称	機能	データ型	値の範囲	参照ページ
_EC_PDCommErr	プロセスデータ通信異常	以下のいずれかのイベントが発生した場合、TRUE となります。 •「プロセスデータ通信異常」(842C0000Hex) •「スレーブの不正な切り離し検知」(84310002Hex) •「スレーブ PDI WDT 異常検知」(84340000Hex)	BOOL	TRUE、FALSE	A-64
_EC_PDTimeoutErr	プロセスデータ受信タイムアウト異常	イベント「プロセスデータ受信タイムアウト」(842B0000Hex) が発生した場合、TRUE となります。	BOOL	TRUE、FALSE	A-64
_EC_PDSendErr	プロセスデータ送信異常	イベント「プロセスデータ送信異常」(84290000Hex) が発生した場合、TRUE となります。	BOOL	TRUE、FALSE	A-64
_EC_SlavAdrDupErr	スレーブノードアドレス重複異常	イベント「スレーブノードアドレス重複」(24200000Hex) が発生した場合、TRUE となります。	BOOL	TRUE、FALSE	A-65
_EC_SlavInitErr	スレーブ初期化異常	以下のいずれかのイベントが発生した場合、TRUE となります。 •「スレーブ初期化異常」(84230000Hex) •「スレーブ状態遷移失敗」(84300001Hex)	BOOL	TRUE、FALSE	A-65
_EC_SlavAppErr	スレーブアプリケーション異常	以下のいずれかのイベントが発生した場合、TRUE となります。 •「スレーブアプリケーション異常」(84280000Hex) •「スレーブ AL ステータス異常検知」(84360000Hex)	BOOL	TRUE、FALSE	A-65
_EC_MsgErr	EtherCAT メッセージ異常	以下のいずれかのイベントが発生した場合、TRUE となります。 •「EtherCAT メッセージ異常」(842D0000Hex) •「不正メールボックス受信」(84350000Hex)	BOOL	TRUE、FALSE	A-65
_EC_SlavEmergErr	エマージェンシー・メッセージ検知	イベント「エマージェンシー・メッセージ検知」(64200000Hex) が発生した場合、TRUE となります。	BOOL	TRUE、FALSE	A-65
_EC_IndataInvalidErr (Ver.1.14)	プロセスデータ入力値無効異常	イベント「プロセスデータ入力値無効異常」(842F0000Hex) が発生した場合、TRUE となります。	BOOL	TRUE、FALSE	A-66
_EC_CommErrTbl	通信異常スレーブテーブル	スレーブノードアドレス順のテーブルで表します。マスタで検出した異常が発生しているスレーブの該当位置は TRUE となります。	Array [1..512] OF BOOL	TRUE、FALSE	A-66
_EC_CycleExceeded	EtherCAT 通信周期設定異常	イベント「EtherCAT 通信周期設定異常」(34410000Hex) が発生した場合、TRUE となります。	BOOL	TRUE、FALSE	A-66



参考

内蔵 EtherCAT 異常フラグの代表関係

変数名	名称	変数名	名称	変数名	名称	イベント重要度
_EC_ErrSta	内蔵 EtherCAT 異常	_EC_PortErr	通信ポート異常	_EC_MacAdrErr	MAC アドレス異常	部分停止フォールトレベル
				_EC_LanHwErr	通信コントローラ異常	
				_EC_LinkOffErr	リンクオフ異常	
		_EC_MstrErr	マスタ異常	_EC_NetCfgErr	ネットワーク構成情報異常	軽度フォールトレベル
				_EC_NetCfgCmpErr	ネットワーク構成照合異常	
				_EC_NetTopologyErr	ネットワーク構成異常	
				_EC_PDCommErr	プロセスデータ通信異常	
				_EC_PDTimeoutErr	プロセスデータ受信タイムアウト異常	
				_EC_PDSendErr	プロセスデータ送信異常	
				_EC_SlavAdrDupErr	スレーブノードアドレス重複異常	
				_EC_SlavInitErr	スレーブ初期化異常	
				_EC_SlavAppErr	スレーブアプリケーション異常	
				_EC_CommErrTbl	通信異常スレーブテーブル	
				_EC_CycleExceeded	EtherCAT 通信周期設定異常	
				_EC_MsgErr	EtherCAT メッセージ異常	監視情報
				_EC_SlavEmergErr	エマージェンシー・メッセージ検知	
		_EC_SlavErr	スレーブ異常	_EC_SlavErrTbl	スレーブ異常テーブル	スレーブ側で定義

（注） EtherCAT 通信の異常に関連するすべてのシステム定義変数は、異常要因を除去した上で、発生中のコントローラ異常を解除（Sysmac Studio のトラブルシュート／ ResetECError 命令）するまで、値は変化しません。

● 機能分類：EtherCAT 通信のステータス関連

変数名	名称	機能	データ型	値の範囲	参照ページ
_EC_RegSlavTbl	登録スレーブテーブル	ネットワーク構成情報に登録されているスレーブ一覧です。 スレーブノードアドレス順のテーブルで表します。 スレーブが登録されている場合、該当位置は TRUE となります。	Array [1..512] OF BOOL	TRUE、FALSE	A-66
_EC_EntrySlavTbl	ネットワーク加入スレーブテーブル	ネットワーク加入スレーブ一覧を通知します。 スレーブノードアドレス順のテーブルで表します。 スレーブがネットワーク加入した時、該当位置は TRUE となります。	Array [1..512] OF BOOL	TRUE、FALSE	A-66
_EC_MBXSlavTbl	メッセージ通信可能スレーブテーブル	メッセージ通信が可能なスレーブ一覧です。スレーブノードアドレス順のテーブルで表します。 スレーブがメッセージ通信可能状態（プレオペレーショナル、セーフオペレーショナル、オペレーショナル）に遷移した場合、該当位置は TRUE となります。 （注） EtherCATスレーブとメッセージ通信する場合、本データを参照して該当スレーブとメッセージ通信が可能かどうかを確認してください。	Array [1..512] OF BOOL	TRUE、FALSE	A-67
_EC_PDSlavTbl	プロセスデータ通信中スレーブテーブル	プロセスデータ通信中のスレーブ一覧です。スレーブノードアドレス順のテーブルで表します。 スレーブが IN、OUT ともにプロセスデータ有効状態（オペレーショナル）に遷移した場合、該当位置は TRUE となります。 （注） EtherCAT スレーブを制御する場合、本データを参照して該当スレーブのデータ有効性を確認してください。	Array [1..512] OF BOOL	TRUE、FALSE	A-67
_EC_DisconnSlavTbl	離脱指令中スレーブテーブル	スレーブノードアドレス順のテーブルで表します。 離脱指令中のスレーブの該当位置は TRUE となります。	Array [1..512] OF BOOL	TRUE、FALSE	A-67
_EC_DisableSlavTbl	無効設定スレーブテーブル	スレーブノードアドレス順のテーブルで表します。 無効に設定されているスレーブの該当位置は TRUE となります。	Array [1..512] OF BOOL	TRUE、FALSE	A-67
_EC_PDActive	プロセスデータ通信ステータス	全スレーブ (*) とのプロセスデータ通信中のとき TRUE となります。 * 無効に設定されているスレーブは含みません。	BOOL	TRUE、FALSE	A-67
_EC_PktMonStop	パケットモニタ停止	パケットモニタ停止中のとき TRUE となります。	BOOL	TRUE、FALSE	A-68
_EC_LinkStatus	リンクステータス	通信コントローラのリンクステータスが Link ON の場合、TRUE となります。	BOOL	TRUE、FALSE	A-68
_EC_PktSaving	パケットデータファイル保存中	パケットデータファイル保存処理中かどうかを示します。 TRUE： パケットデータファイル保存処理中 FALSE：パケットデータファイル保存未処理中	BOOL	TRUE、FALSE	A-68
_EC_InDataInvalid	入力データ無効	プライマリ定周期タスクで実行するプロセスデータ通信を正常に行えず、入力値が不正であるとき、TRUE となります。	BOOL	TRUE、FALSE	A-68
_EC_InData1Invalid	入力データ 1 無効	プライマリ定周期タスクで実行するプロセスデータ通信を正常に行えず、入力値が不正であるとき、TRUE となります。	BOOL	TRUE、FALSE	A-68

（注）EtherCAT 通信のステータスに関連するすべてのシステム定義変数は、現在のステータスを表します。

● 機能分類：EtherCAT 通信の診断統計情報関連

変数名	名称	機能	データ型	値の範囲	参照ページ
_EC_StatisticsLogEnable	診断統計情報ログ許可	診断統計情報ログ機能を開始するときに、FALSE から TRUE にします。 診断統計情報ログ機能を終了するときに、TRUE から FALSE にします。	BOOL	TRUE、FALSE	A-69
_EC_StatisticsLogCycleSec	診断統計情報ログ保存周期	診断統計情報ログ機能において統計診断情報の書き込みを行う周期を秒単位で指定します。 「0」を指定した場合、診断統計情報ログ機能を終了する時に 1 回だけ書き込みを行います。 （注） 診断統計情報ログ機能の実行中に、本システム定義変数の値を変更しても、実行中の書き込み周期は変更されません。	UINT	0、30 ～ 1800	A-69
_EC_StatisticsLogBusy	診断統計情報ログ実行状態フラグ	診断統計情報ログ機能を実行中に TRUE となります。	BOOL	TRUE、FALSE	A-69
_EC_StatisticsLogErr	診断統計情報ログ異常終了フラグ	診断統計情報ログ機能の開始に失敗したか、実行中にログを書き込めなかった場合に TRUE となります。 診断統計情報ログ機能の開始後、_EC_StatisticsLogBusy（診断統計情報ログ実行状態フラグ）が FALSE となったときに、本フラグの値が確定します。 異常終了には以下の要因が含まれます。 • 仮想SDメモリカードの容量が上限に達し、レコードをログファイルに追加できない。 • 仮想SDメモリカードが認識されていない。 • _EC_StatisticsLogCycleSec（診断統計情報ログ保存周期）に指定された値が不正なため、機能を開始できない。	BOOL	TRUE、FALSE	A-69

A-3-5　EtherNet/IP 機能モジュール（カテゴリ名：_EIP）

● 機能分類：EtherNet/IP 通信の異常関連

変数名	名称	機能	データ型	値の範囲	参照ページ
_EIP_ErrSta	内蔵 EtherNet/IP 異常	内蔵 EtherNet/IP ポートの異常ステータス変数です。 NY シリーズコントローラの場合、以下の異常フラグを集約します。 • _EIP1_PortErr（通信ポート 1 異常） • _EIPln1_PortErr（内部通信ポート 1 異常） • _EIP_CipErr（CIP 通信異常） • _EIP_TcpAppErr（TCP アプリ通信異常） （注） 各ビットの意味は後述の「異常ステータスの各ビットの意味」を参照してください。	WORD	16#0000 ～ 16#00F0	A-70
_EIP_PortErr	通信ポート異常	通信ポートの異常ステータス変数です。 NY シリーズコントローラの場合、以下の異常フラグを集約します。 • _EIP1_MacAdrErr（通信ポート 1 MAC アドレス異常） • _EIP1_LanHwErr（通信ポート 1 通信コントローラ異常） • _EIP1_EtnCfgErr（通信ポート 1Ethernet 基本設定異常） • _EIP1_IPAdrCfgErr（通信ポート 1IP アドレス設定異常） • _EIP1_IPAdrDupErr（通信ポート 1IP アドレス重複異常） • _EIP1_BootpErr（通信ポート 1BOOTP サーバ接続異常） • _EIP_DNSCfgErr(DNS 設定異常) • _EIP_DNSSrvErr（DNS サーバ接続異常） • _EIP_IPRTblErr（IP ルートテーブル異常） （注）「リンクオフ検知」または「内蔵 EtherNet/IP 処理異常」時にイベントログに登録して、本異常ビットを ON します。 各ビットの意味は後述の「異常ステータスの各ビットの意味」を参照してください。	WORD	16#0000 ～ 16#00F0	A-70
_EIP1_PortErr	通信ポート 1 異常	通信ポートの異常ステータス変数です。 以下の異常フラグを集約します。 • _EIP1_MacAdrErr（通信ポート 1 MAC アドレス異常） • _EIP1_LanHwErr（通信ポート 1 通信コントローラ異常） • _EIP1_EtnCfgErr（通信ポート 1Ethernet 基本設定異常） • _EIP1_IPAdrCfgErr（通信ポート 1IP アドレス設定異常） • _EIP1_IPAdrDupErr（通信ポート 1IP アドレス重複異常） • _EIP1_BootpErr（通信ポート 1BOOTP サーバ接続異常） • _EIP_DNSCfgErr(DNS 設定異常) • _EIP_DNSSrvErr（DNS サーバ接続異常） • _EIP_IPRTblErr（ＩＰルートテーブル異常） （注）「リンクオフ検知」または「内蔵 EtherNet/IP 処理異常」時にイベントログに登録して、本異常ビットを ON します。 各ビットの意味は後述の「異常ステータスの各ビットの意味」を参照してください。	WORD	16#0000 ～ 16#00F0	A-70

変数名	名称	機能	データ型	値の範囲	参照ページ
_EIPIn1_PortErr	内部通信ポート 1 異常	内部通信ポート 1 の異常ステータス変数です。 以下の異常フラグを集約します。 • _EIPIn1_IPAdrCfgErr（内部通信ポート 1IP アドレス設定異常） • _EIPIn1_IPAdrDupErr（内部通信ポート 1IP アドレス重複異常） • _EIP_DNSCfgErr(DNS 設定異常) • _EIP_DNSSrvErr（DNS サーバ接続異常） • _EIP_IPRTblErr（IP ルートテーブル異常） （注）「リンクオフ検知」または「内蔵 EtherNet/IP 処理異常」時にイベントログに登録して、本異常ビットを ON します。 各ビットの意味は後述の「異常ステータスの各ビットの意味」を参照してください。	WORD	16#0000 ～ 16#00F0	A-71
_EIP_CipErr	CIP 通信異常	CIP 通信の異常ステータス変数です。 以下の異常フラグを集約します。 • _EIP_IdentityErr（Identity 情報不正） • _EIP_TDLinkCfgErr（タグデータリンク設定異常） • _EIP_TDLinkOpnErr（タグデータリンクコネクション確立失敗） • _EIP_TDLinkErr（タグデータリンク通信異常） • _EIP_TagAdrErr（タグ解決異常） • _EIP_MultiSwONErr（複数スイッチ同時 ON 異常） （注）「タグ解決異常」時にイベントログに登録して、本異常を ON します。 各ビットの意味は後述の「異常ステータスの各ビットの意味」を参照してください。	WORD	16#0000 ～ 16#00F0	A-71
_EIP_TcpAppErr	TCP アプリ通信異常	TCP アプリ通信の異常ステータス変数です。 以下の異常フラグを集約します。 • _EIP_TcpAppCfgErr（TCP アプリケーション設定異常） • _EIP_NTPSrvErr（NTP サーバ接続異常） （注） 各ビットの意味は後述の「異常ステータスの各ビットの意味」を参照してください。	WORD	16#0000 ～ 16#00F0	A-71
_EIP_MacAdrErr	MAC アドレス異常	NY シリーズコントローラの場合、電源 ON 時に通信ポート 1 の MAC アドレスの読出異常が発生したことを示します。 TRUE：異常発生 FALSE：正常	BOOL	TRUE、FALSE	A-71
_EIP1_MacAdrErr	通信ポート 1MAC アドレス異常	電源 ON 時に通信ポート 1 の MAC アドレスの読出異常が発生したことを示します。 TRUE：異常発生 FALSE：正常	BOOL	TRUE、FALSE	A-72
_EIP_LanHwErr	通信コントローラ異常	NY シリーズコントローラの場合、通信ポート 1 で通信コントローラの故障が発生していることを示します。 TRUE： 故障発生 FALSE：正常	BOOL	TRUE、FALSE	A-72
_EIP1_LanHwErr	通信ポート 1 通信コントローラ異常	通信ポート 1 で通信コントローラの故障が発生していることを示します。 TRUE： 故障発生 FALSE：正常	BOOL	TRUE、FALSE	A-72

変数名	名称	機能	データ型	値の範囲	参照ページ
_EIP_EtnCfgErr	Ethernet 基本設定異常	NY シリーズコントローラの場合、通信ポート 1 の Ethernet 通信速度の設定（Speed ／ Duplex）が不正。または、読出に失敗したことを示します。 TRUE： 不正または読出失敗 FALSE：正常	BOOL	TRUE、FALSE	A-72
_EIP1_EtnCfgErr	通信ポート 1Ethernet 基本設定異常	通信ポート 1 の Ethernet 通信速度の設定（Speed ／ Duplex）が不正。または、読出に失敗したことを示します。 TRUE： 不正または読出失敗 FALSE：正常	BOOL	TRUE、FALSE	A-72
_EIP_IPAdrCfgErr	IP アドレス設定異常	NY シリーズコントローラの場合、通信ポート 1 の IP アドレス設定の異常を示します。 TRUE：• IP アドレスの設定値が不正 • 読出に失敗 • BOOTP サーバから取得した IP アドレスが不整合 FALSE：正常	BOOL	TRUE、FALSE	A-73
_EIP1_IPAdrCfgErr	通信ポート 1IP アドレス設定異常	通信ポート 1 の IP アドレス設定の異常を示します。 TRUE：• IP アドレスの設定値が不正 • 読出に失敗 • BOOTP サーバから取得した IP アドレスが不整合 FALSE：正常	BOOL	TRUE、FALSE	A-73
_EIPIn1_IPAdrCfgErr	内部通信ポート 1IP アドレス設定異常	内部通信ポート 1 の IP アドレス設定の異常を示します。 TRUE：• IP アドレスの設定値が不正 • 読出に失敗 • BOOTP サーバから取得した IP アドレスが不整合 FALSE：正常	BOOL	TRUE、FALSE	A-73
_EIP_IPAdrDupErr	IP アドレス重複異常	NY シリーズコントローラの場合、通信ポート 1 で IP アドレスの重複が発生していることを示します。 TRUE： 重複発生 FALSE：上記以外	BOOL	TRUE、FALSE	A-73
_EIP1_IPAdrDupErr	通信ポート 1IP アドレス重複異常	通信ポート 1 で IP アドレスの重複が発生していることを示します。 TRUE： 重複発生 FALSE：上記以外	BOOL	TRUE、FALSE	A-73
_EIPIn1_IPAdrDupErr	内部通信ポート 1IP アドレス重複異常	内部通信ポート 1 で IP アドレスの重複が発生していることを示します。 TRUE： 重複発生 FALSE：上記以外	BOOL	TRUE、FALSE	A-74
_EIP_DNSCfgErr	DNS 設定異常	DNS 設定値あるいは hosts 設定が不正。または、読出に失敗したことを示します。 TRUE： 不正または読出失敗 FALSE：正常	BOOL	TRUE、FALSE	A-74
_EIP_BootpErr	BOOTP サーバ接続異常	NY シリーズコントローラの場合、通信ポート 1 で BOOTP サーバの接続に失敗したことを示します。 TRUE： BOOTP サーバの接続に失敗（タイムアウト）しました。 FALSE：BOOTP 設定がされていないか、または、BOOTP 設定がされていて正常に BOOTP サーバから IP アドレスを取得しました。	BOOL	TRUE、FALSE	A-74
_EIP1_BootpErr	通信ポート 1BOOTP サーバ接続異常	通信ポート 1 で BOOTP サーバの接続に失敗したことを示します。 TRUE： BOOTP サーバの接続に失敗（タイムアウト）しました。 FALSE：BOOTP 設定がされていないか、または、BOOTP 設定がされていて正常に BOOTP サーバから IP アドレスを取得しました。	BOOL	TRUE、FALSE	A-74

変数名	名称	機能	データ型	値の範囲	参照ページ
_EIP_IPRTblErr	IP ルートテーブル異常	NY シリーズコントローラの場合、デフォルトゲートウェイの設定あるいは IP ルータテーブルの設定が不正。 または、読出に失敗したことを示します。 TRUE： 不正または読出失敗 FALSE：正常	BOOL	TRUE、FALSE	A-74
_EIP_IdentityErr	Identity 情報不正	NY シリーズコントローラの場合、CIP 通信 1 の Identity 情報（ユーザによる書換え不可）が不正。または、読出に失敗したことを示します。 TRUE： 不正または読出失敗 FALSE：正常	BOOL	TRUE、FALSE	A-75
_EIP_TDLinkCfgErr	タグデータリンク設定異常	NY シリーズコントローラの場合、CIP 通信 1 のタグデータリンクの設定値が不正。または、読出に失敗したことを示します。 TRUE： 不正または読出失敗 FALSE：正常	BOOL	TRUE、FALSE	A-75
_EIP_TDLinkOpnErr	タグデータリンクコネクション確立失敗	NY シリーズコントローラの場合、CIP 通信 1 のタグデータリンクコネクション確立が失敗したことを示します。 TRUE： タグデータリンクコネクション確立が、以下の原因により失敗しました。 • タグデータリンクパラメータに登録されている相手ノード情報と実際のノード情報が異なっている。 • 相手ノードからの応答がない。 FALSE：上記以外	BOOL	TRUE、FALSE	A-75
_EIP_TDLinkErr	タグデータリンク通信異常	NY シリーズコントローラの場合、CIP 通信 1 のタグデータリンクコネクションにタイムアウトが発生していることを示します。 TRUE： タイムアウト発生 FALSE：上記以外	BOOL	TRUE、FALSE	A-75
_EIP_TagAdrErr	タグ解決異常	NY シリーズコントローラの場合、CIP 通信 1 のタグ解決（タグ名からアドレスを特定すること）に失敗したことを示します。 TRUE： タグ解決（タグ名からアドレスを特定すること）に失敗しました。以下の要因があります。 • ネットワーク変数とタグ設定間で、サイズが異なる。 • タグデータリンクの設定入出力方向と、コントローラ内の変数の入出力方向が一致しない。 • コントローラ内に、タグ設定に該当するネットワーク変数が存在しない。 FALSE：上記以外	BOOL	TRUE、FALSE	A-75
_EIP_MultiSwONErr	複数スイッチ同時 ON 異常	NY シリーズコントローラの場合、CIP 通信 1 で複数のスイッチが同時に ON されたことを示します。 TRUE： 複数のデータリンク開始／データリンク停止スイッチが同時に TRUE とされました。 FALSE：上記以外	BOOL	TRUE、FALSE	A-76
_EIP_TcpAppCfgErr	TCP アプリケーション設定異常	TRUE： 少なくとも 1 つ以上、TCP アプリケーション機能（FTP、NTP、SNMP）の設定値に不正があります。または、読出に失敗しました。 FALSE：正常	BOOL	TRUE、FALSE	A-76
_EIP_NTPSrvErr	NTP サーバ接続異常	NY シリーズコントローラの場合、常に FALSE となります。	BOOL	TRUE、FALSE	A-76

変数名	名称	機能	データ型	値の範囲	参照ページ
_EIP_DNSSrvErr	DNS サーバ接続異常	TRUE： DNS クライアントがサーバの接続に失敗（タイムアウト）しました。 FALSE：DNS の設定がされていない。または、DNS の設定がされていて接続に成功しました。	BOOL	TRUE、FALSE	A-76

NY シリーズでの EtherNet/IP の異常関連システム定義変数の階層関係

EtherNet/IP の異常関連のシステム定義変数は、下表のように階層関係になっています。たとえば、第 2 階層の _EIP1_PortErr、_EIPIn1_PortErr、_EIP_CipErr、_EIP_TcpAppErr のいずれかの値が 0 でない場合に、その上位階層の _EIP_ErrSta の値が 0 でなくなります。つまり、上位の階層のシステム定義変数の値を参照することによって、それよりも下位の階層の異常が発生しているかどうかがわかります。

第 1 階層		第 2 階層		第 3 階層	
変数名	名称	変数名	名称	変数名	名称
_EIP_ErrSta	内蔵 EtherNet/IP 異常	_EIP1_PortErr	通信ポート 1 異常	_EIP1_MacAdrErr	通信ポート 1MAC アドレス異常
				_EIP1_LanHwErr	通信ポート 1 通信コントローラ異常
				_EIP1_EtnCfgErr	通信ポート 1Ethernet 基本設定異常
				_EIP1_IPAdrCfgErr	通信ポート 1IP アドレス設定異常
				_EIP1_IPAdrDupErr	通信ポート 1IP アドレス重複異常
				_EIP1_BootpErr	通信ポート 1BOOTP サーバ接続異常
				_EIP_DNSCfgErr	DNS 設定異常
				_EIP_DNSSrvErr	DNS サーバ接続異常
				_EIP_IPRTblErr	IP ルートテーブル異常
		_EIPIn1_PortErr	内部通信ポート 1 異常	_EIPIn1_IPAdrCfgErr	内部通信ポート 1IP アドレス設定異常
				_EIPIn1_IPAdrDupErr	内部通信ポート 1IP アドレス重複異常
				_EIP_DNSCfgErr	DNS 設定異常
				_EIP_DNSSrvErr	DNS サーバ接続異常
				_EIP_IPRTblErr	IP ルートテーブル異常
		_EIP_CipErr	CIP 通信異常	_EIP_IdentityErr	Identity 情報不正
				_EIP_TDLinkCfgErr	タグデータリンク設定異常
				_EIP_TDLinkOpnErr	タグデータリンクコネクション確立失敗
				_EIP_TDLinkErr	タグデータリンク通信異常
				_EIP_TagAdrErr	タグ解決異常
				_EIP_MultiSwONErr	複数スイッチ同時 ON
		_EIP_TcpAppErr	TCP アプリ通信異常	_EIP_TcpAppCfgErr	TCP アプリケーション設定異常
				_EIP_NTPSrvErr	NTP サーバ接続異常

（注） 変数名が「_EIP1」ではじまるシステム定義変数は、変数名が「_EIP」ではじまる同名のシステム定義変数と同じ値を参照します。たとえば、_EIP1_PortErr（通信ポート 1 異常）は、_EIP_PortErr（通信ポート異常）と同じ値を参照します。

● 機能分類：EtherNet/IP 通信のステータス関連

変数名	名称	機能	データ型	値の範囲	参照ページ
_EIP_EtnOnlineSta	オンライン	NY シリーズコントローラの場合、内蔵 EtherNet/IP ポートの通信ポート 1 の通信機能が利用可能（リンク ON かつ、IP アドレス決定。かつ、異常が発生していないこと）になっていることを示します。 TRUE： 利用可能 FALSE：初期処理中、再起動中、異常発生、リンクオフにより内蔵 EtherNet/IP ポートの通信機能が利用できません。	BOOL	TRUE、FALSE	A-76
_EIP1_EtnOnlineSta	通信ポート 1 オンライン	内蔵 EtherNet/IP ポートの通信ポート 1 の通信機能が利用可能（リンク ON かつ、IP アドレス決定。かつ、異常が発生していないこと）になっていることを示します。 TRUE： 利用可能 FALSE：初期処理中、再起動中、異常発生、リンクオフにより内蔵 EtherNet/IP ポートの通信機能が利用できません。	BOOL	TRUE、FALSE	A-77
_EIPIn1_EtnOnlineSta	内部通信ポート 1 オンライン	内蔵 EtherNet/IP ポートの内部通信ポート 1 の通信機能が利用可能（リンク ON かつ、IP アドレス決定。かつ、異常が発生していないこと）になっていることを示します。 TRUE： 利用可能 FALSE：初期処理中、再起動中、異常発生、リンクオフにより内蔵 EtherNet/IP ポートの通信機能が利用できません。	BOOL	TRUE、FALSE	A-77
_EIP_TDLinkRunSta	タグデータリンク通信中	NY シリーズコントローラの場合、CIP 通信 1 で少なくとも 1 つ以上のコネクションが正常動作していることを示します。 TRUE： 正常動作 FALSE：上記以外	BOOL	TRUE、FALSE	A-77
_EIP_TDLinkAllRunSta	全タグデータリンク通信中	NY シリーズコントローラの場合、CIP 通信 1 の全タグデータリンクが通信中であることを示します。 TRUE： オリジネータとしてすべてのコネクションでタグデータリンクが通信中であることを示します。 FALSE：少なくとも 1 つ以上のコネクションに異常が発生しています。	BOOL	TRUE、FALSE	A-77
_EIP_RegTargetSta[255]	登録ターゲットノード情報	NY シリーズコントローラの場合、内蔵 EtherNet/IP の CIP 通信 1 にコネクションが登録されているノードの一覧を表します。 本変数は内蔵 EtherNet/IP がオリジネータとなっている場合だけ有効となります。 配列 [x] が TRUE： ターゲットノード ID「x」のノードとのコネクションが登録されています。 配列 [x] が FALSE： ターゲットノード ID「x」のノードとのコネクションが登録されていません。	ARRAY [0..255] OF BOOL	TRUE、FALSE	A-77
_EIP_EstbTargetSta[255]	正常ターゲットノード情報	NY シリーズコントローラの場合、内蔵 EtherNet/IP の CIP 通信 1 がコネクションを正常に確立しているノードの一覧を表します。 配列 [x] が TRUE： ターゲットノード ID「x」のノードとのコネクションが正常に確立しています。 配列 [x] が FALSE： ターゲットノード ID「x」のノードとのコネクションに未確立または異常が発生しているものがあります。	ARRAY [0..255] OF BOOL	TRUE、FALSE	A-78

変数名	名称	機能	データ型	値の範囲	参照ページ
_EIP_TargetPLCModeSta［255］	ターゲット PLC 動作モード	NY シリーズコントローラの場合、内蔵 EtherNet/IP がオリジネータとなっている CIP 通信 1 のコネクションにおいて該当ターゲットノードのコントローラの運転情報を表します。 本エリアの情報は、正常ターゲットノード情報が TRUE のときだけ有効です。FALSE の場合は直前の値が保持されます。 配列 [x] が TRUE： ターゲットノード ID「x」のターゲットコントローラが運転している状態です。 配列 [x] が FALSE： 上記以外	ARRAY [0..255] OF BOOL	TRUE、FALSE	A-78
_EIP_TargetPLCErr［255］	ターゲット PLC 異常情報	NY シリーズコントローラの場合、内蔵 EtherNet/IP がオリジネータとなっている CIP 通信 1 のコネクションにおいて該当ターゲットノードのコントローラ本体の異常状態（運転停止、継続異常の論理和）を表します。本エリアの情報は、正常ターゲットノード情報が TRUE のときだけ有効です。FALSE の場合は直前の値を保持します。 配列 [x] が TRUE： ターゲットノード ID「x」のターゲットコントローラが運転継続、または、停止異常が発生しています。 配列 [x] が FALSE： 上記以外	ARRAY [0..255] OF BOOL	TRUE、FALSE	A-78
_EIP_TargetNodeErr［255］	ターゲットノード異常情報	NY シリーズコントローラの場合、CIP 通信 1 の登録ターゲットノード情報の該当するコネクションが確立していないか、ターゲットコントローラに異常が発生していることを示します。 本エリアは、登録ターゲットノード情報が TRUE の場合だけ有効です。 配列 [x] が TRUE： ターゲットノード ID「x」のターゲットノードとのコネクションが正常に確立していない （登録ターゲットノード情報が TRUE で正常ターゲットノード情報が FALSE）、 または、ターゲットノードとのコネクションが確立しているが、ターゲットコントローラに異常が発生しています。 配列 [x] が FALSE： ターゲットノード ID「x」のターゲットノードが登録されていない（登録ターゲットノード情報が FALSE）、または、ターゲットノードとのコネクションは確立していて、（登録ターゲットノード情報が TRUE で、正常ターゲットノード情報が TRUE） ターゲットコントローラに異常が発生しています（ターゲット PLC 異常情報が TRUE）。	ARRAY [0..255] OF BOOL	TRUE、FALSE	A-78
_EIP_NTPResult	NTP の動作情報	－	_sNTP_RESULT		A-79
.ExecTime	NTP の最終動作時刻	NY シリーズコントローラの場合、初期値から変化しません。	DATE_AND_TIME	データ型に従う	A-79
.ExecNormal	NTP の動作結果	NY シリーズコントローラの場合、初期値から変化しません。	BOOL	TRUE、FALSE	A-79



参考

ターゲットノードとの通信状態

NY シリーズコントローラとターゲットノードとの通信状態は、以下の 4 つのシステム定義変数の値の組み合わせで、次の表のように決まります。

- _EIP_RegTargetSta（登録ターゲットノード情報）
- _EIP_EstbTargetSta（正常ターゲットノード情報）
- _EIP_TargetPLCErr（ターゲット PLC 異常情報）
- _EIP_TargetNodeErr（ターゲットノード異常情報）

_EIP_Reg TargetSta の値	_EIP_Estb TargetSta の値	_EIP_Target PLCErr の値	_EIP_Target NodeErr の値	ターゲットノードとの通信状態
TRUE	TRUE	FALSE	FALSE	ターゲットノードとのコネクションが正常に確立し、かつターゲット PLC に異常が発生していません。
		TRUE	TRUE	ターゲットノードとのコネクションは確立していますが、ターゲット PLC に異常が発生しています。
	FALSE	―	TRUE	ターゲットノードとのコネクションが正常に確立していません。
FALSE	―	―	―	ターゲットノードが登録されていないため、情報は無効です。

● 機能分類：EtherNet/IP 通信のスイッチ関連

変数名	名称	機能	データ型	値の範囲	参照ページ
_EIP_TDLinkStartCmd	タグデータリンク通信開始スイッチ	NY シリーズコントローラの場合、FALSE から TRUE にすると、CIP 通信 1 のタグデータリンクを開始します。タグデータリンク動作が開始すると、自動的に FALSE になります。 （注） 自動的に FALSE になるまでは、このスイッチをユーザプログラムや Sysmac Studio などから強制的に操作しないでください。	BOOL	TRUE、FALSE	A-79
_EIP_TDLinkStopCmd	タグデータリンク通信停止スイッチ	NY シリーズコントローラの場合、FALSE から TRUE にすると、CIP 通信 1 のタグデータリンクが停止します。タグデータリンク動作が停止すると、自動的に FALSE になります。 （注） 自動的に FALSE になるまでは、このスイッチをユーザプログラムや Sysmac Studio などから強制的に操作しないでください。	BOOL	TRUE、FALSE	A-79

A-3-6　異常ステータスの各ビットの意味

以下の異常ステータスの各ビットの意味は、共通です。

- 「_ErrSta」（コントローラ異常ステータス）
- 「_PLC_ErrSta」（PLC 機能モジュール異常ステータス）
- 「_MC_ErrSta」（MC 異常ステータス）
- 「_MC_ComErrSta」（MC 共通異常ステータス）
- 「_MC_AX_ErrSta」（軸異常ステータス）
- 「_MC_GRP_ErrSta」（軸グループ異常ステータス）
- 「_EC_ErrSta」（内蔵 EtherCAT 異常）
- 「_EC_PortErr」（通信ポート異常）
- 「_EC_MstrErr」（マスタ異常）
- 「_EC_SlavErr」（スレーブ異常）
- 「_EC_SlavErrTbl」（スレーブ異常テーブル）
- 「_EIP_ErrSta」（内蔵 EtherNet/IP 異常）
- 「_EIP_PortErr」（通信ポート異常）、「_EIP1_PortErr」（通信ポート 1 異常）
- 「_EIP_CipErr」（CIP 通信異常）
- 「_EIP_TcpAppErr」（TCP アプリ通信異常）

以下のとおりです。ただし、「_ErrSta」（コントローラ異常ステータス）は、リアルタイム性、および各機能モジュールの異常要因を表すステータスとの同時性の観点から、ユーザプログラム上では使用できません。通信を介して外部から状態を参照するときにだけ使用します。

ビット:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WORD			-	-	-	-	-	-					-	-	-	-

ビット	内容
15	予約
14	スレーブ集約：イベント発生源（各機能モジュール）の下位階層（ユニット / スレーブ / 軸 / 軸グループ等）において、コントローラ異常が発生しているか否かを示します。 TRUE:　いずれかの下位階層でコントローラ異常あり FALSE:　いずれかの下位階層でコントローラ異常なし （_MC_ErrSta/_EC_ErrSta で有効。）
13 ～ 8	予約
7	全停止フォールト（Major fault）レベルのコントローラ異常が発生中か否かを示します。 TRUE:　全停止フォールトレベルのコントローラ異常が発生中 FALSE:　全停止フォールトレベルのコントローラ異常が発生中でない
6	部分停止フォールト（Partial fault）レベルのコントローラ異常が発生中か否かを示します。 TRUE:　部分停止フォールトレベルのコントローラ異常が発生中 FALSE:　部分停止フォールトレベルのコントローラ異常が発生中でない
5	軽度フォールト（Minor fault）レベルのコントローラ異常が発生中か否かを示します。 TRUE:　軽度フォールトレベルのコントローラ異常が発生中 FALSE:　軽度フォールトレベルのコントローラ異常が発生中でない
4	監視情報（Observation）レベルのコントローラ異常が発生中か否かを示します。 TRUE:　監視情報レベルのコントローラ異常が発生中 FALSE:　監視情報レベルのコントローラ異常が発生中でない
3 ～ 0	予約

A-4 システム定義変数の個別仕様

システム定義変数の個別仕様表の見方は、以下のとおりです。

変数名	［システム定義変数名です。先頭にカテゴリ名が付きます］			**メンバ（構造体型の場合）**	［構造体型の場合だけメンバ名を示します］
名称	［名称です］			**グローバル／ローカル**	「グローバル：グローバル変数、ローカル：ローカル変数」
機能	［機能を説明します］				
データ型	［変数のデータ型を示します］			**値の範囲**	［変数の値の範囲を示します］
R/W アクセス	［R：読出専用 RW：読書可能］	**保持**	［変数の保持属性を示します］	**ネットワーク公開**	［変数のネットワーク公開属性を示します］
ユーザプログラムでの使用可否	［ユーザプログラム上で直接指定して使用することが可能か否かを示します］	**関連命令**	［変数に関連する命令を示します。 「ユーザプログラムでの使用」が不可の場合は、アクセスするための専用命令を示します］		

A-4-1　NY シリーズコントローラのシステム全体（カテゴリ名：なし）

● 機能分類：時計関連

変数名	_CurrentTime				
名称	システム時刻			グローバル／ローカル	グローバル
機能	コントローラの内部時計データが格納されます。				
データ型	DATE_AND_TIME			値の範囲	DT#2000-01-01-00:00:00 ～ DT#2099-12-31-23:59:59
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	時計機能用命令		

● 機能分類：タスク関連

変数名	_＜タスク名称＞_Active				
名称	タスク実行中フラグ			グローバル／ローカル	グローバル
機能	タスク実行中のとき TRUE となります。 タスク実行中でないとき FALSE となります。 （注） 本システム定義変数は、ユーザプログラム上では使用できません。Sysmac Studio のデータトレース機能でタスクの状態を参照するときにだけ使用します。				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	非公開
ユーザプログラムでの使用可否	不可	関連命令	• ActEventTask ユーザプログラムからは以下の専用命令でだけアクセス可能です。 • Task_IsActive		

変数名	_＜タスク名称＞_LastExecTime				
名称	前回タスク実行時間			グローバル／ローカル	グローバル
機能	前回に実行したタスク実行時間（0.1 μs 単位）を示します。 （注） 本システム定義変数は、ユーザプログラム上では使用できません。Sysmac Studio のデータトレース機能でタスクの状態を参照するときにだけ使用します。				
データ型	TIME			値の範囲	データ型に従う
R/W アクセス	R	保持	非保持	ネットワーク公開	非公開
ユーザプログラムでの使用可否	不可	関連命令	ユーザプログラムからは以下の専用命令でだけアクセス可能です。 • GetMyTaskStatus		

変数名	_＜タスク名称＞_MaxExecTime				
名称	タスク実行時間最大値			グローバル／ローカル	グローバル
機能	タスク実行時間の最大値（0.1 μs 単位）を示します。 （注） 本システム定義変数は、ユーザプログラム上では使用できません。Sysmac Studio のデータトレース機能でタスクの状態を参照するときにだけ使用します。				
データ型	TIME			値の範囲	データ型に従う
R/W アクセス	R	保持	非保持	ネットワーク公開	非公開
ユーザプログラムでの使用可否	不可	関連命令	ユーザプログラムからは以下の専用命令でだけアクセス可能です。 • GetMyTaskStatus		

変数名	_＜タスク名称＞_MinExecTime				
名称	タスク実行時間最小値			グローバル／ローカル	グローバル
機能	タスク実行時間の最小値（0.1 μs 単位）を示します。 （注） 本システム定義変数は、ユーザプログラム上では使用できません。Sysmac Studio のデータトレース機能でタスクの状態を参照するときにだけ使用します。				
データ型	TIME			値の範囲	データ型に従う
R/W アクセス	R	保持	非保持	ネットワーク公開	非公開
ユーザプログラムでの使用可否	不可	関連命令	ユーザプログラムからは以下の専用命令でだけアクセス可能です。 • GetMyTaskStatus		

変数名	_＜タスク名称＞_ExecCount				
名称	タスク実行回数			グローバル／ローカル	グローバル
機能	タスクの実行回数を示します。 4294967295 回を超えたら、0 に戻して、カウントを継続します。 （注） 本システム定義変数は、ユーザプログラム上では使用できません。Sysmac Studio のデータトレース機能でタスクの状態を参照するときにだけ使用します。				
データ型	UDINT			値の範囲	データ型に従う
R/W アクセス	R	保持	非保持	ネットワーク公開	非公開
ユーザプログラムでの使用可否	不可	関連命令	ユーザプログラムからは以下の専用命令でだけアクセス可能です。 • GetMyTaskStatus		

変数名	_＜タスク名称＞_Exceeded				
名称	タスク周期超過フラグ			グローバル／ローカル	グローバル
機能	タスク周期を超過したら TRUE となります。 タスク周期以内にタスク実行が完了したら FALSE となります。 （注） 本システム定義変数は、ユーザプログラム上では使用できません。Sysmac Studio のデータトレース機能でタスクの状態を参照するときにだけ使用します。				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	非公開
ユーザプログラムでの使用可否	不可	関連命令	ユーザプログラムからは以下の専用命令でだけアクセス可能です。 • GetMyTaskStatus		

変数名	_＜タスク名称＞_ExceedCount				
名称	タスク周期超過回数			グローバル／ローカル	グローバル
機能	周期を超過した回数が格納されます。 現在値がデータ型の最大値を超えたら、現在値を 0 に戻して、カウントを継続します。 4294967295 回を超えたら、0 に戻して、カウントを継続します。 （注） 本システム定義変数は、ユーザプログラム上では使用できません。Sysmac Studio のデータトレース機能でタスクの状態を参照するときにだけ使用します。				
データ型	UDINT			値の範囲	データ型に従う
R/W アクセス	R	保持	非保持	ネットワーク公開	非公開
ユーザプログラムでの使用可否	不可	関連命令	ユーザプログラムからは以下の専用命令でだけアクセス可能です。 • GetMyTaskStatus		

● 機能分類：異常関連

変数名	_ErrSta				
名称	コントローラ異常ステータス			グローバル／ローカル	グローバル
機能	コントローラ異常発生中は TRUE となります。 コントローラ異常未発生中は FALSE となります。 （注） リアルタイム性、および各機能モジュールの異常要因を表すステータスとの同時性の観点から、ユーザプログラム上では使用できません。通信を介して外部から状態を参照するときにだけ使用します。各ビットの意味は、📖 「A-3-6 異常ステータスの各ビットの意味」（P.A-45）を参照してください。				
データ型	WORD			値の範囲	16#0000 ～ 16#C0F0
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	不可	関連命令	• ResetPLCError • ResetECError • ResetMCError • MC_Reset • MC_GroupReset ユーザプログラムからは以下の専用命令でだけアクセス可能です。 • GetPLCError • GetECError • GetMCError • GetEIPError		

変数名	_AlarmFlag				
名称	ユーザ異常ステータス			グローバル／ローカル	グローバル
機能	ユーザ異常発生中はイベント重要度に該当するビットが TRUE となります。 User fault Level 1 ～ 8 が、ビット 00 ～ 07 に対応します。 ユーザ異常未発生中は 16 進 0000 となります。				
データ型	WORD			値の範囲	16#0000 ～ 16#00FF
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	• SetAlarm • ResetAlarm • GetAlarm		

● 機能分類：SD メモリカード関連

変数名	_Card1Ready				
名称	SD メモリカード使用可フラグ			グローバル／ローカル	グローバル
機能	NY シリーズでは、仮想 SD メモリカードの認識ができたとき、TRUE となります。 認識ができなかったとき、FALSE となります。 TRUE：使用可 FALSE：使用不可				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	－		

変数名	_Card1Protect				
名称	SD メモリカードライトプロテクトフラグ			グローバル／ローカル	グローバル
機能	NY シリーズでは、使用しません。 常に FALSE となります。				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	－		

変数名	_Card1Err				
名称	SD メモリカードエラーフラグ			グローバル／ローカル	グローバル
機能	NY シリーズでは、使用しません。 常に FALSE となります。				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	－		

変数名	_Card1Access				
名称	SD メモリカードアクセス中フラグ			グローバル／ローカル	グローバル
機能	NY シリーズでは、使用しません。 常に FALSE となります。				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	－		

変数名	_Card1Deteriorated				
名称	SD メモリカード寿命警告フラグ			グローバル／ローカル	グローバル
機能	NY シリーズでは、使用しません。 常に FALSE となります。				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	－		

変数名	_Card1PowerFail				
名称	SD メモリカードアクセス中電源断フラグ			グローバル／ローカル	グローバル
機能	仮想 SD メモリカードアクセス中にコントローラの電源断が発生したとき、TRUE となります。 TRUE：仮想 SD メモリカードアクセス中に電源断があった FALSE：正常				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	RW	保持	保持*1	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	－		

*1. 保持属性ありのシステム定義変数ですが、バックアップ機能の対象外です。

変数名	_Card1Capacity*1				
名称	SD メモリカードストレージ容量			グローバル／ローカル	グローバル
機能	仮想 SD メモリカードの全容量を示します。 単位は MiB（1MiB=1,048,576Byte）です。 値は 60 秒ごとに更新されます。 仮想 SD メモリカードが無効の場合、“0” になります。				
データ型	UDINT			値の範囲	データ型に従う
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	－		

*1. CPU ユニットのユニットバージョン Ver.1.21 で追加されたシステム定義変数です。

変数名	_Card1Used*1				
名称	SD メモリカードストレージ使用量			グローバル／ローカル	グローバル
機能	仮想 SD メモリカードの使用量を示します。 単位は MiB (1MiB=1,048,576Byte) です。 値は 60 秒ごとに更新されます。 仮想 SD メモリカードが無効の場合、 “0” になります。				
データ型	UDINT			値の範囲	データ型に従う
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	−		

*1. CPU ユニットのユニットバージョン Ver.1.21 で追加されたシステム定義変数です。

変数名	_Card1BkupCmd			メンバ名	.ExecBkup
名称	バックアップ起動フラグ			グローバル／ローカル	グローバル
機能	コントローラのデータを仮想 SD メモリカードへバックアップするときに、TRUE にします。 （注） 本システム定義変数は、ユーザプログラム上では使用できません。CIP メッセージ通信によって、表示器や上位パソコンから指令するときに使用します。				
データ型	構造体：_sBKUP_CMD、メンバ：BOOL			値の範囲	TRUE、FALSE
R/W アクセス	RW	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	不可	関連命令	−		

変数名	_Card1BkupCmd			メンバ名	.CancelBkup
名称	バックアップ取消フラグ			グローバル／ローカル	グローバル
機能	仮想 SD メモリカードへのバックアップを中止するときに、TRUE にします。 （注） 本システム定義変数は、ユーザプログラム上では使用できません。CIP メッセージ通信によって、表示器や上位パソコンから指令するときに使用します。				
データ型	構造体：_sBKUP_CMD、メンバ：BOOL			値の範囲	TRUE、FALSE
R/W アクセス	RW	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	不可	関連命令	−		

変数名	_Card1BkupCmd			メンバ名	.ExecVefy
名称	照合起動フラグ			グローバル／ローカル	グローバル
機能	コントローラと仮想 SD メモリカード内のバックアップファイルを照合するときに、TRUE にします。 （注） 本システム定義変数は、ユーザプログラム上では使用できません。CIP メッセージ通信によって、表示器や上位パソコンから指令するときに使用します。				
データ型	構造体：_sBKUP_CMD、メンバ：BOOL			値の範囲	TRUE、FALSE
R/W アクセス	RW	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	不可	関連命令	−		

変数名	_Card1BkupCmd			メンバ名	.CancelVefy
名称	照合取消フラグ			グローバル／ローカル	グローバル
機能	コントローラと仮想 SD メモリカード内のバックアップファイルの照合を中止するときに、TRUE にします。 （注） 本システム定義変数は、ユーザプログラム上では使用できません。CIP メッセージ通信によって、表示器や上位パソコンから指令するときに使用します。				
データ型	構造体：_sBKUP_CMD、メンバ：BOOL			値の範囲	TRUE、FALSE
R/W アクセス	RW	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	不可	関連命令	−		

変数名	_Card1BkupCmd			メンバ名	.DirName
名称	ディレクトリ名			グローバル／ローカル	グローバル
機能	バックアップ対象の仮想 SD メモリカードのディレクトリ名を指定します。 （注） 本システム定義変数は、ユーザプログラム上では使用できません。CIP メッセージ通信によって、表示器や上位パソコンから指令するときに使用します。				
データ型	構造体：_sBKUP_CMD、メンバ：STRING(64)			値の範囲	データ型に従う
R/W アクセス	RW	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	不可	関連命令	－		

変数名	_Card1BkupSta			メンバ名	.Done
名称	完了フラグ			グローバル／ローカル	グローバル
機能	バックアップ実行完了のとき、TRUE となります。 （注） 本システム定義変数は、ユーザプログラム上では使用できません。CIP メッセージ通信によって、表示器や上位パソコンから読み出すときに使用します。				
データ型	構造体：_sBKUP_STA、メンバ：BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	不可	関連命令	－		

変数名	_Card1BkupSta			メンバ名	.Active
名称	実行中フラグ			グローバル／ローカル	グローバル
機能	バックアップ実行中のとき、TRUE となります。 （注） 本システム定義変数は、ユーザプログラム上では使用できません。CIP メッセージ通信によって、表示器や上位パソコンから読み出すときに使用します。				
データ型	構造体：_sBKUP_STA、メンバ：BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	不可	関連命令	－		

変数名	_Card1BkupSta			メンバ名	.Err
名称	エラーフラグ			グローバル／ローカル	グローバル
機能	バックアップ処理が異常終了したとき、TRUE となります。 （注） 本システム定義変数は、ユーザプログラム上では使用できません。CIP メッセージ通信によって、表示器や上位パソコンから読み出すときに使用します。				
データ型	構造体：_sBKUP_STA、メンバ：BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	不可	関連命令	－		

変数名	_Card1VefySta			メンバ名	.Done
名称	完了フラグ			グローバル／ローカル	グローバル
機能	照合実行完了のとき、TRUE となります。 （注） 本システム定義変数は、ユーザプログラム上では使用できません。CIP メッセージ通信によって、表示器や上位パソコンから読み出すときに使用します。				
データ型	構造体：_sVEFY_STA、メンバ：BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	不可	関連命令	－		

変数名	_Card1VefySta			メンバ名	.Active
名称	実行中フラグ			グローバル／ローカル	グローバル
機能	照合実行中のとき、TRUE となります。 （注） 本システム定義変数は、ユーザプログラム上では使用できません。CIP メッセージ通信によって、表示器や上位パソコンから読み出すときに使用します。				
データ型	構造体：_sVEFY_STA、メンバ：BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	不可	関連命令	−		

変数名	_Card1VefySta			メンバ名	.VefyRslt
名称	照合結果フラグ			グローバル／ローカル	グローバル
機能	照合一致のとき TRUE、照合不一致のとき FALSE となります。 （注） 本システム定義変数は、ユーザプログラム上では使用できません。CIP メッセージ通信によって、表示器や上位パソコンから読み出すときに使用します。				
データ型	構造体：_sVEFY_STA、メンバ：BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	不可	関連命令	−		

変数名	_Card1VefySta			メンバ名	.Err
名称	エラーフラグ			グローバル／ローカル	グローバル
機能	照合処理が異常終了したとき、TRUE となります。 （注） 本システム定義変数は、ユーザプログラム上では使用できません。CIP メッセージ通信によって、表示器や上位パソコンから読み出すときに使用します。				
データ型	構造体：_sVEFY_STA、メンバ：BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	不可	関連命令	−		

● 機能分類：バックアップ関連

変数名	_BackupBusy				
名称	バックアップ関連機能実行中フラグ			グローバル／ローカル	グローバル
機能	バックアップ、リストア、照合の実行中のとき、TRUE となります。				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	−		

● 機能分類：電源管理関連

変数名	_PowerOnHour				
名称	通電時間			グローバル／ローカル	グローバル
機能	通電時間を示します。 コントローラの通電時間を、1 時間単位で格納します。 リセットする場合、0 を格納してください。 4294967295 となった場合、それ以上は更新されません。 電源 ON 時に初期化されません。				
データ型	UDINT			値の範囲	0 ～ 4294967295
R/W アクセス	RW	保持	保持*1	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	−		

*1. 保持属性ありのシステム定義変数ですが、バックアップ機能の対象外です。

変数名	_PowerOnCount				
名称	電源断発生回数			グローバル／ローカル	グローバル
機能	電源断の発生回数を示します。 コントローラの最初の電源投入から、電源断発生ごとに積算（+1）されます。 リセットする場合は、0 を格納してください。 4294967295 となった場合、それ以上は更新されません。 電源 ON 時に、初期化されません。				
データ型	UDINT			値の範囲	0 ～ 4294967295
R/W アクセス	RW	保持	保持[*1]	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	－		

*1. 保持属性ありのシステム定義変数ですが、バックアップ機能の対象外です。

変数名	_RetainFail				
名称	電断保持失敗フラグ			グローバル／ローカル	グローバル
機能	以下のとき、TRUE（電断保持失敗）となります。 • 電源投入時、バッテリバックアップメモリ／不揮発性メモリのチェックでエラーのとき 以下のとき、FALSE（電断保持失敗なし）となります。 • 電源投入時、バッテリバックアップメモリ／不揮発性メモリのメモリチェックで OK のとき • ユーザプログラムがダウンロードされたとき • メモリオールクリアを実行したとき （注） 絶対値エンコーダ原点位置オフセットデータが保持できなかったときは、このフラグでなく、軸変数の異常ステータスに反映されます。				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	非公開
ユーザプログラムでの使用可否	可能	関連命令	－		

変数名	_SelfTest_UPSSignal				
名称	UPS 信号検知フラグ			グローバル／ローカル	グローバル
機能	UPS からの一時電断信号を検知した場合、TRUE となります。				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	• IPC_EnableShutdown		

変数名	_RequestShutdown				
名称	シャットダウン要求フラグ			グローバル／ローカル	グローバル
機能	稼働中に電源ボタンが押下された場合、TRUE となります。				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	• IPC_EnableShutdown		

● 機能分類：OS(Windows) 関連

変数名	_OSRunning				
名称	OS 稼働中フラグ			グローバル／ローカル	グローバル
機能	コントローラが OS(Windows) の状態を稼働中と観測した場合、TRUE となります。				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	• IPC_GetOSStatus • IPC_ShutdownOS • IPC_RebootOS		

変数名	_OSHalted				
名称	OS 停止中フラグ			グローバル／ローカル	グローバル
機能	コントローラが OS(Windows) の状態を停止中と観測した場合、TRUE となります。				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	• IPC_GetOSStatus • IPC_ShutdownOS • IPC_RebootOS		

変数名	_OSErrorState				
名称	OS 異常状態フラグ			グローバル／ローカル	グローバル
機能	コントローラが OS(Windows) の状態を異常発生中と判断した場合、TRUE となります。				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	• IPC_GetOSStatus • IPC_ShutdownOS • IPC_RebootOS		

● 機能分類：プログラミング関連

変数名	P_On				
名称	常時 ON フラグ			グローバル／ローカル	グローバル
機能	常に TRUE を示すフラグです。				
データ型	BOOL			値の範囲	TRUE
R/W アクセス	R	保持	非保持	ネットワーク公開	非公開
ユーザプログラムでの使用可否	可能	関連命令	−		

変数名	P_Off				
名称	常時 OFF フラグ			グローバル／ローカル	グローバル
機能	常に FALSE を示すフラグです。				
データ型	BOOL			値の範囲	FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	非公開
ユーザプログラムでの使用可否	可能	関連命令	−		

変数名	P_CY				
名称	キャリーフラグ			グローバル／ローカル	ローカル
機能	一部の命令にて更新するフラグです。				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	非公開
ユーザプログラムでの使用可否	可能	関連命令	−		

変数名	P_First_RunMode				
名称	運転開始時 1 周期 ON フラグ			グローバル／ローカル	ローカル
機能	コントローラの動作モードを、プログラムモードから運転モードに変更した際に、当該プログラムが実行中である場合、1 タスク周期だけ TRUE となります。 当該プログラムが実行中でなければ、FALSE のままです。 コントローラが運転を開始したときに初期処理を行う場合などに使用します。 （注） 本システム定義変数は、ファンクション内では使用できません。				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	非公開
ユーザプログラムでの使用可否	可能	関連命令	−		

変数名	P_First_Run				
名称	プログラム起動時 1 周期 ON フラグ			グローバル／ローカル	ローカル
機能	当該プログラムが、実行を開始した際に、1 タスク周期だけ TRUE となります。 プログラムが実行を開始するごとに初期処理を行う場合などに使用します。 （注） 本システム定義変数は、ファンクション内では使用できません。				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	非公開
ユーザプログラムでの使用可否	可能	関連命令	－		

変数名	P_PRGER				
名称	命令エラー発生フラグ			グローバル／ローカル	ローカル
機能	当該プログラムまたは当該プログラムから呼び出した FB や FUN で、命令エラーが発生したときに TRUE となり続けるフラグです。一度 TRUE となると、ユーザプログラムによって FALSE にされるまで TRUE となり続けます。				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	RW	保持	非保持	ネットワーク公開	非公開
ユーザプログラムでの使用可否	可能	関連命令	－		

● 機能分類：バージョン管理関連

変数名	_UnitVersion				
名称	ユニットバージョン			グローバル／ローカル	グローバル
機能	コントローラのユニットバージョンが格納されます。 要素番号 0 に、ユニットバージョンの整数部が格納されます。 要素番号 1 に、ユニットバージョンの小数部が格納されます。 例 1) ユニットバージョン Ver.1.08 の場合、要素番号 0 に 1、要素番号 1 に 8 が格納されます。 例 2) ユニットバージョン Ver.1.10 の場合、要素番号 0 に 1、要素番号 1 に 10 が格納されます。				
データ型	ARRAY[0..1] OF USINT			値の範囲	0 ～ 99
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	－		

変数名	_HardwareRevision				
名称	ハードウェアリビジョン			グローバル／ローカル	グローバル
機能	コントローラのハードウェアリビジョンが格納されます。 ハードウェアリビジョンが「なし」の場合は「'-'」、それ以外の場合は、「'A' ～ 'Z'」が格納されます。				
データ型	STRING[2]			値の範囲	'-'、'A' ～ 'Z'
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	－		

● 機能分類：自己診断関連

変数名	_SelfTest_HighTemperature				
名称	CPU ユニット内部温度オーバーフラグ			グローバル／ローカル	グローバル
機能	コントローラ内部温度が高温のとき、TRUE となります。				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	－		

変数名	_SelfTest_LowBattery				
名称	バッテリ電圧低下フラグ			グローバル／ローカル	グローバル
機能	バッテリが外れているか、バッテリ電圧が低下しているとき、TRUE となります。				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	－		

変数名	_SelfTest_LowFanRevolution				
名称	FAN 回転数低下フラグ			グローバル／ローカル	グローバル
機能	FAN が外れているか、FAN の回転数が低下しているとき、TRUE となります。				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	－		

● 機能分類：PLC 内蔵関連

変数名	_DeviceOutHoldCfg*1				
名称	デバイス出力保持設定			グローバル／ローカル	グローバル
機能	動作モード変更時またはダウンロード時に、対象のデバイス出力を保持する場合、16#A5A5 とします。 16#A5A5 以外では、動作モード変更時またはダウンロード時に、対象のデバイス出力は初期化されます。				
データ型	WORD			値の範囲	16#0000 ～ 16#FFFF
R/W アクセス	RW	保持	保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	－		

*1. コントローラのユニットバージョン Ver.1.14 で追加されたシステム定義変数です。

変数名	_DeviceOutHoldStatus*1				
名称	デバイス出力保持状態			グローバル／ローカル	グローバル
機能	動作モード変更時またはダウンロード時に、対象のデバイス出力が保持された場合、TRUE となります。 デバイス出力保持設定が 16#A5A5 以外のとき、または全停止フォールトレベルのコントローラ異常発生時に、対象のデバイス出力が初期化された場合、FALSE となります。				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	－		

*1. コントローラのユニットバージョン Ver.1.14 で追加されたシステム定義変数です。

A-4-2　PLC 機能モジュール（カテゴリ名：_PLC）

● 機能分類：デバッグ関連

変数名	_PLC_TraceSta[0..3]			メンバ名	.IsStart
名称	トレース実行中 ON フラグ			グローバル／ローカル	グローバル
機能	トレースを開始したとき、TRUE となります。 （注） 本システム定義変数は、ユーザプログラム上では使用できません。Sysmac Studio でデータトレースの状態をウォッチするときにだけ使用します。				
データ型	構造体：_sTRACE_STA、メンバ：BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	非公開
ユーザプログラムでの使用可否	不可	関連命令	• TraceTrig • TraceSamp ユーザプログラムからは以下の専用命令でだけアクセス可能です。 • GetTraceStatus		

変数名	_PLC_TraceSta[0..3]			メンバ名	.IsComplete
名称	トレース完了時 ON フラグ			グローバル／ローカル	グローバル
機能	トレース実行が完了したとき、TRUE となります。 （注） 本システム定義変数は、ユーザプログラム上では使用できません。Sysmac Studio でデータトレースの状態をウォッチするときにだけ使用します。				
データ型	構造体：_sTRACE_STA、メンバ：BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	非公開
ユーザプログラムでの使用可否	不可	関連命令	• TraceTrig • TraceSamp ユーザプログラムからは以下の専用命令でだけアクセス可能です。 • GetTraceStatus		

変数名	_PLC_TraceSta[0..3]			メンバ名	.IsTrigger
名称	トレーストリガモニタフラグ			グローバル／ローカル	グローバル
機能	トレーストリガ条件が成立したとき、TRUE となります。 次にトレースを開始したとき、FALSE となります。 （注） 本システム定義変数は、ユーザプログラム上では使用できません。Sysmac Studio でデータトレースの状態をウォッチするときにだけ使用します。				
データ型	構造体：_sTRACE_STA、メンバ：BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	保持	ネットワーク公開	非公開
ユーザプログラムでの使用可否	不可	関連命令	• TraceTrig • TraceSamp ユーザプログラムからは以下の専用命令でだけアクセス可能です。 • GetTraceStatus		

変数名	_PLC_TraceSta[0..3]			メンバ名	.ParamErr
名称	トレースパラメータ異常フラグ			グローバル／ローカル	グローバル
機能	トレース開始時にトレース設定が異常のとき、TRUE となります。 正常のとき、FALSE となります。 （注） 本システム定義変数は、ユーザプログラム上では使用できません。Sysmac Studio でデータトレースの状態をウォッチするときにだけ使用します。				
データ型	構造体：_sTRACE_STA、メンバ：BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	非公開
ユーザプログラムでの使用可否	不可	関連命令	ユーザプログラムからは以下の専用命令でだけアクセス可能です。 • GetTraceStatus		

● 機能分類：異常関連

変数名	_PLC_ErrSta				
名称	PLC 機能モジュール異常ステータス			グローバル／ローカル	グローバル
機能	PLC 機能モジュールにかかわるコントローラ異常発生中は TRUE となります。 PLC 機能モジュールにかかわるコントローラ異常未発生中は FALSE となります。 各ビットの意味は、「A-3-6 異常ステータスの各ビットの意味」（P.A-45）を参照してください。				
データ型	WORD			値の範囲	16#0000 ～ 16#00F0
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	• GetPLCError 以下の専用命令で解除可能 • ResetPLCError		

A-4-3　モーション制御機能モジュール（カテゴリ名：_MC）

● 機能分類：モーション制御機能関連

変数名	_MC_ErrSta				
名称	モーション制御機能モジュールの異常ステータス			グローバル／ローカル	グローバル
機能	モーション制御機能モジュールで検知した異常の状態を表します。 ユーザプログラム上で使用（直接参照）できます。各ビットの意味は、「A-3-6 異常ステータスの各ビットの意味」（P.A-45）を参照してください。				
データ型	WORD			値の範囲	16#0000 ～ 16#40F0
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	• GetMCError • ResetMCError • MC_Reset • MC_GroupReset		

変数名	_MC_ComErrSta				
名称	共通異常ステータス			グローバル／ローカル	グローバル
機能	モーション制御において、共通の処理で検知した異常の状態を表します。 ユーザプログラム上で使用（直接参照）できます。各ビットの意味は、「A-3-6 異常ステータスの各ビットの意味」（P.A-45）を参照してください。				
データ型	WORD			値の範囲	16#0000 ～ 16#00F0
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	• GetMCError • ResetMCError		

変数名	_MC_AX_ErrSta				
名称	軸異常ステータス			グローバル／ローカル	グローバル
機能	軸単位で検知した異常の状態を表します。最大 64 軸を表示します。 ユーザプログラム上で使用（直接参照）できます。各ビットの意味は、「A-3-6 異常ステータスの各ビットの意味」（P.A-45）を参照してください。				
データ型	ARRAY [0..63] OF WORD			値の範囲	16#0000 ～ 16#00F0
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	• GetMCError • ResetMCError • MC_Reset		

変数名	_MC_GRP_ErrSta				
名称	軸グループ異常ステータス			グローバル／ローカル	グローバル
機能	軸グループ単位で検知した異常の状態を表します。最大 32 軸グループを表示します。 ユーザプログラム上で使用（直接参照）できます。各ビットの意味は、「A-3-6 異常ステータスの各ビットの意味」（P.A-45）を参照してください。				
データ型	ARRAY [0..31] OF WORD			値の範囲	16#0000 ～ 16#00F0
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	• GetMCError • ResetMCError • MC_GroupReset		

変数名	_MC_COM				
名称	共通変数			グローバル／ローカル	グローバル
機能	モーション制御機能モジュール共通の状態モニタをします。 構造体のメンバの詳細は、📖『NY シリーズ コマンドリファレンスマニュアル モーション編（SBCE-380）』を参照してください。				
データ型	_sCOMMON_REF			値の範囲	－
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	－		

変数名	_MC_GRP				
名称	軸グループ変数			グローバル／ローカル	グローバル
機能	NY シリーズコントローラの場合、モーション制御命令での軸グループ指定、多軸協調制御の状態モニタ、多軸協調制御の設定モニタをします。 Sysmac Studio で「軸グループ」を作成すると、ユーザ定義の別名の軸グループ変数が生成されます。 通常は、別名の軸グループ変数を使用します。 構造体のメンバの詳細は、📖『NY シリーズ コマンドリファレンスマニュアル モーション編（SBCE-380）』を参照してください。				
データ型	ARRAY[0..31] OF _sGROUP_REF			値の範囲	－
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	－		

変数名	_MC_AX				
名称	軸変数			グローバル／ローカル	グローバル
機能	NY シリーズコントローラの場合、モーション制御命令での軸指定、単軸制御の状態モニタ、単軸制御の設定モニタをします。 Sysmac Studio で「軸」を作成すると、ユーザ定義の別名の軸変数が生成されます。 通常は、別名の軸変数を使用します。 構造体のメンバの詳細は、📖『NY シリーズ コマンドリファレンスマニュアル モーション編（SBCE-380）』を参照してください。				
データ型	ARRAY[0..63] OF _sAXIS_REF			値の範囲	－
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	－		

A-4-4　EtherCAT マスタ機能モジュール（カテゴリ名：_EC）

● 機能分類：EtherCAT 通信の異常関連

変数名	_EC_ErrSta				
名称	内蔵 EtherCAT 異常			グローバル／ローカル	グローバル
機能	EtherCAT マスタ機能モジュールの異常を集約したシステム定義変数です。 各ビットの意味は、📖「A-3-6 異常ステータスの各ビットの意味」（P.A-45）を参照してください。				
データ型	WORD			値の範囲	16#0000 ～ 16#40F0
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	EtherCAT 異常ステータス取得命令 • GetECError EtherCAT 異常解除命令 • ResetECError		

変数名	_EC_PortErr				
名称	通信ポート異常			グローバル／ローカル	グローバル
機能	EtherCAT マスタ用通信ポートの異常を集約したシステム定義変数です。 各ビットの意味は、📖「A-3-6 異常ステータスの各ビットの意味」（P.A-45）を参照してください。				
データ型	WORD			値の範囲	16#0000 ～ 16#00F0
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	EtherCAT 異常ステータス取得命令 • GetECError EtherCAT 異常解除命令 • ResetECError		

変数名	_EC_MstrErr				
名称	マスタ異常			グローバル／ローカル	グローバル
機能	EtherCAT マスタの異常と EtherCAT マスタが検知したスレーブの異常を集約したシステム定義変数です。 各ビットの意味は、📖「A-3-6 異常ステータスの各ビットの意味」（P.A-45）を参照してください。				
データ型	WORD			値の範囲	16#0000 ～ 16#00F0
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	EtherCAT 異常ステータス取得命令 • GetECError EtherCAT 異常解除命令 • ResetECError		

変数名	_EC_SlavErr				
名称	スレーブ異常			グローバル／ローカル	グローバル
機能	EtherCAT スレーブ全体の異常ステータスを集約したシステム定義変数です。 各ビットの意味は、📖「A-3-6 異常ステータスの各ビットの意味」（P.A-45）を参照してください。				
データ型	WORD			値の範囲	16#0000 ～ 16#00F0
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	EtherCAT 異常ステータス取得命令 • GetECError EtherCAT 異常解除命令 • ResetECError		

変数名	_EC_SlavErrTbl				
名称	スレーブ異常テーブル			グローバル／ローカル	グローバル
機能	EtherCAT スレーブ単位の異常ステータスを通知します。 本異常ステータスは実システム構成のスレーブ単位の異常状態を示します。本変数の配列は EtherCAT スレーブのノードアドレス（1 ～ 512）に対応し、異常が発生しているスレーブを示します。 各ビットの意味は、📖 「A-3-6 異常ステータスの各ビットの意味」（P.A-45）を参照してください。				
データ型	Array [1..512] OF WORD			値の範囲	16#0000 ～ 16#00F0
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	EtherCAT 異常ステータス取得命令 • GetECError EtherCAT 異常解除命令 • ResetECError		

変数名	_EC_MacAdrErr				
名称	MAC アドレス異常			グローバル／ローカル	グローバル
機能	イベント「MAC アドレス異常」(14400000Hex) が発生した場合、TRUE となります。				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	－		

変数名	_EC_LanHwErr				
名称	通信コントローラ異常			グローバル／ローカル	グローバル
機能	イベント「通信コントローラ異常」(047C0000Hex) が発生した場合、TRUE となります。				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	－		

変数名	_EC_LinkOffErr				
名称	リンクオフ異常			グローバル／ローカル	グローバル
機能	イベント「リンクオフ異常」(84200000Hex) が発生した場合、TRUE となります。				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	EtherCAT 異常解除命令 • ResetECError		

変数名	_EC_NetCfgErr				
名称	ネットワーク構成情報異常			グローバル／ローカル	グローバル
機能	イベント「ネットワーク構成情報異常」(34400000Hex) が発生した場合、TRUE となります。				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	－		

変数名	_EC_NetCfgCmpErr				
名称	ネットワーク構成照合異常			グローバル／ローカル	グローバル
機能	以下のいずれかのイベントが発生した場合、TRUE となります。 •「ネットワーク構成照合異常」(84220000Hex) •「ネットワーク構成照合異常 (スレーブ不在)」(84380000Hex) •「ネットワーク構成照合異常 (スレーブ誤接続)」(84320003Hex) •「ネットワーク構成照合異常 (スレーブ不一致)」(84330004Hex) •「ネットワーク構成照合異常 (リング誤配線)」(843A0000Hex)				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	EtherCAT 異常解除命令 • ResetECError		

変数名	_EC_NetTopologyErr				
名称	ネットワーク構成異常			グローバル／ローカル	グローバル
機能	以下のいずれかのイベントが発生した場合、TRUE となります。 •「ネットワーク構成異常」(84210000Hex) •「不正な配線検知」(843C0000Hex)				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	EtherCAT 異常解除命令 • ResetECError		

変数名	_EC_PDCommErr				
名称	プロセスデータ通信異常			グローバル／ローカル	グローバル
機能	以下のいずれかのイベントが発生した場合、TRUE となります。 •「プロセスデータ通信異常」(842C0000Hex) •「スレーブの不正な切り離し検知」(84310002Hex) •「スレーブ PDI WDT 異常検知」(84340000Hex)				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	EtherCAT 異常解除命令 • ResetECError		

変数名	_EC_PDTimeoutErr				
名称	プロセスデータ受信タイムアウト異常			グローバル／ローカル	グローバル
機能	イベント「プロセスデータ受信タイムアウト」(842B0000Hex) が発生した場合、TRUE となります。				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	EtherCAT 異常解除命令 • ResetECError		

変数名	_EC_PDSendErr				
名称	プロセスデータ送信異常			グローバル／ローカル	グローバル
機能	イベント「プロセスデータ送信異常」(84290000Hex) が発生した場合、TRUE となります。				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	EtherCAT 異常解除命令 • ResetECError		

変数名	_EC_SlavAdrDupErr				
名称	スレーブノードアドレス重複異常			グローバル／ローカル	グローバル
機能	イベント「スレーブノードアドレス重複」(24200000Hex) が発生した場合、TRUE となります。				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	EtherCAT 異常解除命令 • ResetECError		

変数名	_EC_SlavInitErr				
名称	スレーブ初期化異常			グローバル／ローカル	グローバル
機能	以下のいずれかのイベントが発生した場合、TRUE となります。 •「スレーブ初期化異常」(84230000Hex) •「スレーブ状態遷移失敗」(84300001Hex)				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	EtherCAT 異常解除命令 • ResetECError		

変数名	_EC_SlavAppErr				
名称	スレーブアプリケーション異常			グローバル／ローカル	グローバル
機能	以下のいずれかのイベントが発生した場合、TRUE となります。 •「スレーブアプリケーション異常」(84280000Hex) •「スレーブ AL ステータス異常検知」(84360000Hex)				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	EtherCAT 異常解除命令 • ResetECError		

変数名	_EC_MsgErr				
名称	EtherCAT メッセージ異常			グローバル／ローカル	グローバル
機能	以下のいずれかのイベントが発生した場合、TRUE となります。 •「EtherCAT メッセージ異常」(842D0000Hex) •「不正メールボックス受信」(84350000Hex)				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	CoE メッセージ（CoE SDO 読出） • EC_CoESDORead CoE メッセージ（CoE SDO 書込） • EC_CoESDOWrite		

変数名	_EC_SlavEmergErr				
名称	エマージェンシー・メッセージ検知			グローバル／ローカル	グローバル
機能	イベント「エマージェンシー・メッセージ検知」(64200000Hex) が発生した場合、TRUE となります。				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	EtherCAT 異常解除命令 • ResetECError		

変数名	_EC_IndataInvalidErr*1				
名称	プロセスデータ入力値無効異常			グローバル／ローカル	グローバル
機能	イベント「プロセスデータ入力値無効異常」(842F0000Hex) が発生した場合、TRUE となります。				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	EtherCAT 異常解除命令 • ResetECError		

*1. コントローラのユニットバージョン Ver.1.14 で追加されたシステム定義変数です。

変数名	_EC_CommErrTbl				
名称	通信異常スレーブテーブル			グローバル／ローカル	グローバル
機能	スレーブノードアドレス順のテーブルで表します。マスタで検出した異常が発生しているスレーブの該当位置は TRUE となります。				
データ型	Array [1..512] OF BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	EtherCAT 異常解除命令 • ResetECError		

(注) EtherCAT 通信の異常に関連するすべてのシステム定義変数は、異常要因を除去した上で、発生中のコントローラ異常を解除（Sysmac Studio のトラブルシュート／ ResetECError 命令）するまで、値は変化しません。

変数名	_EC_CycleExceeded				
名称	EtherCAT 通信周期設定異常			グローバル／ローカル	グローバル
機能	イベント「EtherCAT 通信周期設定異常」(34410000Hex) が発生した場合、TRUE となります。				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	−		

● 機能分類：EtherCAT 通信のステータス関連

変数名	_EC_RegSlavTbl				
名称	登録スレーブテーブル			グローバル／ローカル	グローバル
機能	ネットワーク構成情報に登録されているスレーブ一覧です。 スレーブノードアドレス順のテーブルで表します。 スレーブが登録されている場合、該当位置は TRUE となります。				
データ型	Array [1..512] OF BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	−		

変数名	_EC_EntrySlavTbl				
名称	ネットワーク加入スレーブテーブル			グローバル／ローカル	グローバル
機能	ネットワーク加入スレーブ一覧を通知します。スレーブノードアドレス順のテーブルで表します。 スレーブがネットワーク加入した時、該当位置は TRUE となります。				
データ型	Array [1..512] OF BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	−		

変数名	_EC_MBXSlavTbl				
名称	メッセージ通信可能スレーブテーブル			グローバル／ローカル	グローバル
機能	メッセージ通信が可能なスレーブ一覧です。スレーブノードアドレス順のテーブルで表します。 スレーブがメッセージ通信可能状態（プレオペレーショナル、セーフオペレーショナル、オペレーショナル）に遷移した場合、該当位置は TRUE となります。 （注） EtherCAT スレーブとメッセージ通信する場合、本データを参照して該当スレーブとメッセージ通信が可能かどうかを確認してください。				
データ型	Array [1..512] OF BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	EtherCAT スレーブ離脱 • EC_DisconnectSlave EtherCAT スレーブ再加入 • EC_ConnectSlave		

変数名	_EC_PDSlavTbl				
名称	プロセスデータ通信中スレーブテーブル			グローバル／ローカル	グローバル
機能	プロセスデータ通信中のスレーブ一覧です。スレーブノードアドレス順のテーブルで表します。 スレーブが IN、OUT ともにプロセスデータ有効状態（オペレーショナル）に遷移した場合、該当位置は TRUE となります。 （注） EtherCAT スレーブを制御する場合、本データを参照して該当スレーブのデータ有効性を確認してください。				
データ型	Array [1..512] OF BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	EtherCAT スレーブ離脱 • EC_DisconnectSlave EtherCAT スレーブ再加入 • EC_ConnectSlave		

変数名	_EC_DisconnSlavTbl				
名称	離脱指令中スレーブテーブル			グローバル／ローカル	グローバル
機能	スレーブノードアドレス順のテーブルで表します。 離脱指令中のスレーブの該当位置は TRUE となります。				
データ型	Array [1..512] OF BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	EtherCAT スレーブ離脱 • EC_DisconnectSlave EtherCAT スレーブ再加入 • EC_ConnectSlave		

変数名	_EC_DisableSlavTbl				
名称	無効設定スレーブテーブル			グローバル／ローカル	グローバル
機能	スレーブノードアドレス順のテーブルで表します。 無効に設定されているスレーブの該当位置は TRUE となります。				
データ型	Array [1..512] OF BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	−		

変数名	_EC_PDActive				
名称	プロセスデータ通信ステータス			グローバル／ローカル	グローバル
機能	全スレーブ*1 とのプロセスデータ通信中のとき TRUE となります。				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	EtherCAT スレーブ離脱 • EC_DisconnectSlave EtherCAT スレーブ再加入 • EC_ConnectSlave		

*1. 無効に設定されているスレーブは含みません。

変数名	_EC_PktMonStop				
名称	パケットモニタ停止			グローバル／ローカル	グローバル
機能	パケットモニタ停止中のとき TRUE となります。				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	パケットモニタの停止 • EC_StopMon パケットモニタの起動 • EC_StartMon		

変数名	_EC_LinkStatus				
名称	リンクステータス			グローバル／ローカル	グローバル
機能	通信コントローラのリンクステータスが Link ON の場合、TRUE となります。				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	－		

変数名	_EC_PktSaving				
名称	パケットデータファイル保存中			グローバル／ローカル	グローバル
機能	パケットデータファイル保存処理中かどうかを示します。 TRUE：パケットデータファイル保存処理中 FALSE：パケットデータファイル保存未処理中				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	パケットデータのファイル保存 • EC_SaveMon		

変数名	_EC_InDataInvalid				
名称	入力データ無効			グローバル／ローカル	グローバル
機能	プライマリ定周期タスクで実行するプロセスデータ通信を正常に行えず、入力値が不正であるとき、TRUE となります。				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	－		

(注) 1 EtherCAT 通信のステータスに関連するすべてのシステム定義変数は、現在のステータスを表します。
2 EC_DisconnectSlave（EtherCAT スレーブ離脱）命令、EC_ConnectSlave（EtherCAT スレーブ再加入）命令、または EC_ChangeEnableSetting（EtherCAT スレーブ有効／無効切替）命令を実行すると、一時的に TRUE になります。

変数名	_EC_InData1Invalid				
名称	入力データ 1 無効			グローバル／ローカル	グローバル
機能	プライマリ定周期タスクで実行するプロセスデータ通信を正常に行えず、入力値が不正であるとき、TRUE となります。				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	－		

(注) 1 EtherCAT 通信のステータスに関連するすべてのシステム定義変数は、現在のステータスを表します。
2 EC_DisconnectSlave（EtherCAT スレーブ離脱）命令、EC_ConnectSlave（EtherCAT スレーブ再加入）命令、または EC_ChangeEnableSetting（EtherCAT スレーブ有効／無効切替）命令を実行すると、一時的に TRUE になります。

● 機能分類：EtherCAT 通信の診断統計情報関連

変数名	_EC_StatisticsLogEnable				
名称	診断統計情報ログ許可			グローバル／ローカル	グローバル
機能	診断統計情報ログ機能を開始するときに、FALSE から TRUE にします。 診断統計情報ログ機能を終了するときに、TRUE から FALSE にします。				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	RW	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	－		

変数名	_EC_StatisticsLogCycleSec				
名称	診断統計情報ログ保存周期			グローバル／ローカル	グローバル
機能	診断統計情報ログ機能において統計診断情報の書き込みを行う周期を秒単位で指定します。 「0」を指定した場合、診断統計情報ログ機能を終了する時に 1 回だけ書き込みを行います。 （注） 診断統計情報ログ機能の実行中に、本システム定義変数の値を変更しても、実行中の書き込み周期は変更されません。				
データ型	UINT			値の範囲	0、30 ～ 1800
R/W アクセス	RW	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	－		

変数名	_EC_StatisticsLogBusy				
名称	診断統計情報ログ実行状態フラグ			グローバル／ローカル	グローバル
機能	診断統計情報ログ機能を実行中に TRUE となります。				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	－		

変数名	_EC_StatisticsLogErr				
名称	診断統計情報ログ異常終了フラグ			グローバル／ローカル	グローバル
機能	診断統計情報ログ機能の開始に失敗したか、実行中にログを書き込めなかった場合に TRUE となります。 診断統計情報ログ機能の開始後、_EC_StatisticsLogBusy（診断統計情報ログ実行状態フラグ）が FALSE となったときに、本フラグの値が確定します。 異常終了には以下の要因が含まれます。 • 仮想 SD メモリカードの容量が上限に達し、レコードをログファイルに追加できない。 • 仮想 SD メモリカードが認識されていない。 • _EC_StatisticsLogCycleSec（診断統計情報ログ保存周期）に指定された値が不正なため、機能を開始できない。				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	－		

A-4-5　EtherNet/IP 機能モジュール（カテゴリ名：_EIP）

● 機能分類：EtherNet/IP 通信の異常関連

変数名	_EIP_ErrSta				
名称	内蔵 EtherNet/IP 異常			グローバル／ローカル	グローバル
機能	内蔵 EtherNet/IP ポートの異常ステータス変数です。 NY シリーズコントローラの場合、以下の異常フラグを集約します。 • _EIP1_PortErr（通信ポート 1 異常） • _EIPIn1_PortErr（内部通信ポート 1 異常） • _EIP_CipErr（CIP 通信異常） • _EIP_TcpAppErr（TCP アプリ通信異常） （注） 各ビットの意味は、📖 「A-3-6 異常ステータスの各ビットの意味」（P.A-45）を参照してください。				
データ型	WORD			値の範囲	16#0000 ～ 16#00F0
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	ユーザプログラムからは以下の専用命令でアクセス可能です。 • GetEIPError		

変数名	_EIP_PortErr				
名称	通信ポート異常			グローバル／ローカル	グローバル
機能	通信ポートの異常ステータス変数です。 NY シリーズコントローラの場合、以下の異常フラグを集約します。 • _EIP1_MacAdrErr（通信ポート 1 MAC アドレス異常） • _EIP1_LanHwErr（通信ポート 1 通信コントローラ異常） • _EIP1_EtnCfgErr（通信ポート 1Ethernet 基本設定異常） • _EIP1_IPAdrCfgErr（通信ポート 1IP アドレス設定異常） • _EIP1_IPAdrDupErr（通信ポート 1IP アドレス重複異常） • _EIP1_BootpErr（通信ポート 1BOOTP サーバ接続異常） • _EIP_DNSCfgErr(DNS 設定異常) • _EIP_DNSSrvErr（DNS サーバ接続異常） • _EIP_IPRTblErr（IP ルートテーブル異常） （注）「リンクオフ検知」または「内蔵 EtherNet/IP 異常」時にイベントログに登録して、本異常ビットを ON します。各ビットの意味は、📖 「A-3-6 異常ステータスの各ビットの意味」（P.A-45）を参照してください。				
データ型	WORD			値の範囲	16#0000 ～ 16#00F0
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	ユーザプログラムからは以下の専用命令でアクセス可能です。 • GetEIPError		

変数名	_EIP1_PortErr				
名称	通信ポート 1 異常			グローバル／ローカル	グローバル
機能	通信ポート 1 の異常ステータス変数です。 以下の異常フラグを集約します。 • _EIP1_MacAdrErr（通信ポート 1 MAC アドレス異常） • _EIP1_LanHwErr（通信ポート 1 通信コントローラ異常） • _EIP1_EtnCfgErr（通信ポート 1Ethernet 基本設定異常） • _EIP1_IPAdrCfgErr（通信ポート 1IP アドレス設定異常） • _EIP1_IPAdrDupErr（通信ポート 1IP アドレス重複異常） • _EIP1_BootpErr（通信ポート 1BOOTP サーバ接続異常） • _EIP_DNSCfgErr(DNS 設定異常) • _EIP_DNSSrvErr（DNS サーバ接続異常） • _EIP_IPRTblErr（IP ルートテーブル異常） （注）「リンクオフ検知」または「内蔵 EtherNet/IP 異常」時にイベントログに登録して、本異常ビットを ON します。各ビットの意味は、📖 「A-3-6 異常ステータスの各ビットの意味」（P.A-45）を参照してください。				
データ型	WORD			値の範囲	16#0000 ～ 16#00F0
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	ユーザプログラムからは以下の専用命令でアクセス可能です。 • GetEIPError		

変数名	_EIPIn1_PortErr				
名称	内部通信ポート 1 異常			グローバル／ローカル	グローバル
機能	内部通信ポート 1 の異常ステータス変数です。 以下の異常フラグを集約します。 • _EIPIn1_IPAdrCfgErr（内部通信ポート 1IP アドレス設定異常） • _EIPIn1_IPAdrDupErr（内部通信ポート 1IP アドレス重複異常） • _EIP_DNSCfgErr(DNS 設定異常) • _EIP_DNSSrvErr（DNS サーバ接続異常） • _EIP_IPRTblErr（IP ルートテーブル異常） （注）「リンクオフ検知」または「内蔵 EtherNet/IP 異常」時にイベントログに登録して、本異常ビットを ON します。各ビットの意味は、🕮 「A-3-6 異常ステータスの各ビットの意味」(P.A-45) を参照してください。				
データ型	WORD			値の範囲	16#0000 ～ 16#00F0
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	ユーザプログラムからは以下の専用命令でアクセス可能です。 • GetEIPError		

変数名	_EIP_CipErr				
名称	CIP 通信異常			グローバル／ローカル	グローバル
機能	CIP 通信の異常ステータス変数です。 以下の異常フラグを集約します。 • _EIP_IdentityErr（Identity 情報不正） • _EIP_TDLinkCfgErr（タグデータリンク設定異常） • _EIP_TDLinkOpnErr（タグデータリンクコネクション確立失敗） • _EIP_TDLinkErr（タグデータリンク通信異常） • _EIP_TagAdrErr（タグ解決異常） • _EIP_MultiSwONErr（複数スイッチ同時 ON 異常） （注）「タグ解決異常」時にイベントログに登録して、本異常を ON します。各ビットの意味は、🕮 「A-3-6 異常ステータスの各ビットの意味」(P.A-45) を参照してください。				
データ型	WORD			値の範囲	16#0000 ～ 16#00F0
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	ユーザプログラムからは以下の専用命令でアクセス可能です。 • GetEIPError		

変数名	_EIP_TcpAppErr				
名称	TCP アプリ通信異常			グローバル／ローカル	グローバル
機能	TCP アプリ通信の異常ステータス変数です。 以下の異常フラグを集約します。 • _EIP_TcpAppCfgErr（TCP アプリケーション設定異常） • _EIP_NTPSrvErr（NTP サーバ接続異常） （注） 各ビットの意味は、🕮 「A-3-6 異常ステータスの各ビットの意味」(P.A-45) を参照してください。				
データ型	WORD			値の範囲	16#0000 ～ 16#00F0
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	ユーザプログラムからは以下の専用命令でアクセス可能です。 • GetEIPError		

変数名	_EIP_MacAdrErr				
名称	MAC アドレス異常			グローバル／ローカル	グローバル
機能	NY シリーズコントローラの場合、電源 ON 時に通信ポート 1 の MAC アドレスの読出異常が発生したことを示します。 TRUE：異常発生 FALSE：正常				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	－		

変数名	_EIP1_MacAdrErr				
名称	通信ポート 1MAC アドレス異常			グローバル／ローカル	グローバル
機能	電源 ON 時に通信ポート 1 の MAC アドレスの読出異常が発生したことを示します。 TRUE：異常発生 FALSE：正常				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	－		

変数名	_EIP_LanHwErr				
名称	通信コントローラ異常			グローバル／ローカル	グローバル
機能	NY シリーズコントローラの場合、通信ポート 1 で通信コントローラの故障が発生していることを示します。 TRUE：故障発生 FALSE：正常				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	－		

変数名	_EIP1_LanHwErr				
名称	通信ポート 1 通信コントローラ異常			グローバル／ローカル	グローバル
機能	通信ポート 1 で通信コントローラの故障が発生していることを示します。 TRUE：故障発生 FALSE：正常				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	－		

変数名	_EIP_EtnCfgErr				
名称	Ethernet 基本設定異常			グローバル／ローカル	グローバル
機能	NY シリーズコントローラの場合、通信ポート 1 の Ethernet 通信速度の設定（Speed ／ Duplex）が不正。または、読出に失敗したことを示します。 TRUE：不正または読出失敗 FALSE：正常				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	－		

変数名	_EIP1_EtnCfgErr				
名称	通信ポート 1Ethernet 基本設定異常			グローバル／ローカル	グローバル
機能	通信ポート 1 の Ethernet 通信速度の設定（Speed ／ Duplex）が不正。または、読出に失敗したことを示します。 TRUE：不正または読出失敗 FALSE：正常				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	－		

変数名	_EIP_IPAdrCfgErr				
名称	IP アドレス設定異常			グローバル／ローカル	グローバル
機能	NY シリーズコントローラの場合、通信ポート 1 の IP アドレス設定の異常を示します。 TRUE：• IP アドレスの設定値が不正 • 読出に失敗 • BOOTP サーバから取得した IP アドレスが不整合 FALSE：正常				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	－		

変数名	_EIP1_IPAdrCfgErr				
名称	通信ポート 1IP アドレス設定異常			グローバル／ローカル	グローバル
機能	通信ポート 1 の IP アドレス設定の異常を示します。 TRUE：• IP アドレスの設定値が不正 • 読出に失敗 • BOOTP サーバから取得した IP アドレスが不整合 FALSE：正常				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	－		

変数名	_EIPIn1_IPAdrCfgErr				
名称	内部通信ポート 1IP アドレス設定異常			グローバル／ローカル	グローバル
機能	内部通信ポート 1 の IP アドレス設定の異常を示します。 TRUE：• IP アドレスの設定値が不正 • 読出に失敗 • BOOTP サーバから取得した IP アドレスが不整合 FALSE：正常				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	－		

変数名	_EIP_IPAdrDupErr				
名称	IP アドレス重複異常			グローバル／ローカル	グローバル
機能	NY シリーズコントローラの場合、通信ポート 1 で IP アドレスの重複が発生していることを示します。 TRUE：重複発生 FALSE：上記以外				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	－		

変数名	_EIP1_IPAdrDupErr				
名称	通信ポート 1IP アドレス重複異常			グローバル／ローカル	グローバル
機能	通信ポート 1 で IP アドレスの重複が発生していることを示します。 TRUE：重複発生 FALSE：上記以外				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	－		

変数名	_EIPIn1_IPAdrDupErr				
名称	内部通信ポート 1IP アドレス重複異常			グローバル／ローカル	グローバル
機能	内部通信ポート 1 で IP アドレスの重複が発生していることを示します。 TRUE：重複発生 FALSE：上記以外				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	－		

変数名	_EIP_DNSCfgErr				
名称	DNS 設定異常			グローバル／ローカル	グローバル
機能	DNS 設定値あるいは hosts 設定が不正。または、読出に失敗したことを示します。 TRUE：不正または読出失敗 FALSE：正常				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	－		

変数名	_EIP_BootpErr				
名称	BOOTP サーバ接続異常			グローバル／ローカル	グローバル
機能	NY シリーズコントローラの場合、通信ポート 1 で BOOTP サーバの接続に失敗したことを示します。 TRUE：BOOTP サーバの接続に失敗（タイムアウト）しました。 FALSE：BOOTP 設定がされていないか、または、BOOTP 設定がされていて正常に BOOTP サーバから IP アドレスを取得しました。				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	－		

変数名	_EIP1_BootpErr				
名称	通信ポート 1BOOTP サーバ接続異常			グローバル／ローカル	グローバル
機能	通信ポート 1 で BOOTP サーバの接続に失敗したことを示します。 TRUE：BOOTP サーバの接続に失敗（タイムアウト）しました。 FALSE：BOOTP 設定がされていないか、または、BOOTP 設定がされていて正常に BOOTP サーバから IP アドレスを取得しました。				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	－		

変数名	_EIP_IPRTblErr				
名称	IP ルートテーブル異常			グローバル／ローカル	グローバル
機能	NY シリーズコントローラの場合、デフォルトゲートウェイの設定あるいは IP ルータテーブルの設定が不正。または、読出に失敗したことを示します。 TRUE：不正または読出失敗 FALSE：正常				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	－		

変数名	_EIP_IdentityErr				
名称	Identity 情報不正			グローバル／ローカル	グローバル
機能	NY シリーズコントローラの場合、CIP 通信 1 の Identity 情報（ユーザによる書換え不可）が不正。または、読出に失敗したことを示します。 TRUE：不正または読出失敗 FALSE：正常				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	－		

変数名	_EIP_TDLinkCfgErr				
名称	タグデータリンク設定異常			グローバル／ローカル	グローバル
機能	NY シリーズコントローラの場合、CIP 通信 1 のタグデータリンクの設定値が不正。または、読出に失敗したことを示します。 TRUE：不正または読出失敗 FALSE：正常				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	－		

変数名	_EIP_TDLinkOpnErr				
名称	タグデータリンクコネクション確立失敗			グローバル／ローカル	グローバル
機能	NY シリーズコントローラの場合、CIP 通信 1 のタグデータリンクコネクション確立が失敗したことを示します。 TRUE：タグデータリンクコネクション確立が、以下の原因により失敗しました。 • タグデータリンクパラメータに登録されている相手ノード情報と実際のノード情報が異なっている。 • 相手ノードからの応答がない。 FALSE：上記以外				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	－		

変数名	_EIP_TDLinkErr				
名称	タグデータリンク通信異常			グローバル／ローカル	グローバル
機能	NY シリーズコントローラの場合、CIP 通信 1 のタグデータリンクコネクションにタイムアウトが発生していることを示します。 TRUE：タイムアウト発生 FALSE：上記以外				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	－		

変数名	_EIP_TagAdrErr				
名称	タグ解決異常			グローバル／ローカル	グローバル
機能	NY シリーズコントローラの場合、CIP 通信 1 のタグ解決（タグ名からアドレスを特定すること）に失敗したことを示します。 TRUE：タグ解決（タグ名からアドレスを特定すること）に失敗しました。以下の要因があります。 • ネットワーク変数とタグ設定間で、サイズが異なる。 • タグデータリンクの設定入出力方向と、コントローラ内の変数の入出力方向が一致しない。 • コントローラ内に、タグ設定に該当するネットワーク変数が存在しない。 FALSE：上記以外				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	－		

変数名	_EIP_MultiSwONErr				
名称	複数スイッチ同時 ON 異常			グローバル／ローカル	グローバル
機能	NY シリーズコントローラの場合、CIP 通信 1 で複数のスイッチが同時に ON されたことを示します。 TRUE：複数のデータリンク開始／データリンク停止スイッチが同時に TRUE とされました。 FALSE：上記以外				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	－		

変数名	_EIP_TcpAppCfgErr				
名称	TCP アプリケーション設定異常			グローバル／ローカル	グローバル
機能	TRUE：少なくとも 1 つ以上、TCP アプリケーション機能（FTP、NTP、SNMP）の設定値に不正があります。 または、読出に失敗しました。 FALSE：正常				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	－		

変数名	_EIP_NTPSrvErr				
名称	NTP サーバ接続異常			グローバル／ローカル	グローバル
機能	NY シリーズコントローラの場合、常に FALSE となります。				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	－		

変数名	_EIP_DNSSrvErr				
名称	DNS サーバ接続異常			グローバル／ローカル	グローバル
機能	TRUE：DNS クライアントがサーバの接続に失敗（タイムアウト）しました。 FALSE：DNS の設定がされていない。または、DNS の設定がされていて接続に成功しました。				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	－		

● 機能分類：EtherNet/IP 通信のステータス関連

変数名	_EIP_EtnOnlineSta				
名称	オンライン			グローバル／ローカル	グローバル
機能	NY シリーズコントローラの場合、内蔵 EtherNet/IP ポートの通信ポート 1 の通信機能が利用可能（リンク ON かつ、IP アドレス決定。かつ、異常が発生していないこと）になっていることを示します。 TRUE：利用可能 FALSE：初期処理中、再起動中、異常発生、リンクオフにより内蔵 EtherNet/IP ポートの通信機能が利用できません。				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	－		

変数名	_EIP1_EtnOnlineSta				
名称	通信ポート 1 オンライン			グローバル／ローカル	グローバル
機能	内蔵 EtherNet/IP ポートの通信ポート 1 の通信機能が利用可能（リンク ON かつ、IP アドレス決定。かつ、異常が発生していないこと）になっていることを示します。 TRUE：利用可能 FALSE：初期処理中、再起動中、異常発生、リンクオフにより内蔵 EtherNet/IP ポートの通信機能が利用できません。				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	－		

変数名	_EIPIn1_EtnOnlineSta				
名称	内部通信ポート 1 オンライン			グローバル／ローカル	グローバル
機能	内蔵 EtherNet/IP ポートの内部通信ポート 1 の通信機能が利用可能（リンク ON かつ、IP アドレス決定。かつ、異常が発生していないこと）になっていることを示します。 TRUE：利用可能 FALSE：初期処理中、再起動中、異常発生、リンクオフにより内蔵 EtherNet/IP ポートの内部通信ポート 1 の通信機能が利用できません。				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	－		

変数名	_EIP_TDLinkRunSta				
名称	タグデータリンク通信中			グローバル／ローカル	グローバル
機能	NY シリーズコントローラの場合、CIP 通信 1 で少なくとも 1 つ以上のコネクションが正常動作していることを示します。 TRUE：正常動作 FALSE：上記以外				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	－		

変数名	_EIP_TDLinkAllRunSta				
名称	全タグデータリンク通信中			グローバル／ローカル	グローバル
機能	NY シリーズコントローラの場合、CIP 通信 1 の全タグデータリンクが通信中であることを示します。 TRUE：オリジネータとしてすべてのコネクションでタグデータリンクが通信中であることを示します。 FALSE：少なくとも 1 つ以上のコネクションに異常が発生しています。				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	－		

変数名	_EIP_RegTargetSta［255］				
名称	登録ターゲットノード情報			グローバル／ローカル	グローバル
機能	NY シリーズコントローラの場合、内蔵 EtherNet/IP の CIP 通信 1 にコネクションが登録されているノードの一覧を表します。 本変数は内蔵 EtherNet/IP がオリジネータとなっている場合だけ有効となります。 配列 [x] が TRUE：ターゲットノード ID「x」のノードとのコネクションが登録されています。 配列 [x] が FALSE：ターゲットノード ID「x」のノードとのコネクションが登録されていません。				
データ型	ARRAY [0..255] OF BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	－		

変数名	_EIP_EstbTargetSta［255］				
名称	正常ターゲットノード情報			グローバル／ローカル	グローバル
機能	NY シリーズコントローラの場合、内蔵 EtherNet/IP の CIP 通信 1 がコネクションを正常に確立しているノードの一覧を表します。 配列 [x] が TRUE：ターゲットノード ID「x」のノードとのコネクションが正常に確立しています。 配列 [x] が FALSE：ターゲットノード ID「x」のノードとのコネクションに未確立または異常が発生しているものがあります。				
データ型	ARRAY [0..255] OF BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	－		

変数名	_EIP_TargetPLCModeSta［255］				
名称	ターゲット PLC 動作モード			グローバル／ローカル	グローバル
機能	NY シリーズコントローラの場合、内蔵 EtherNet/IP がオリジネータとなっている CIP 通信 1 のコネクションにおいて該当ターゲットノードのコントローラの運転情報を表します。 本エリアの情報は、正常ターゲットノード情報が TRUE のときだけ有効です。FALSE の場合は直前の値が保持されます。 配列 [x] が TRUE：ターゲットノード ID「x」のターゲットコントローラが運転している状態です。 配列 [x] が FALSE：上記以外				
データ型	ARRAY [0..255] OF BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	－		

変数名	_EIP_TargetPLCErr［255］				
名称	ターゲット PLC 異常情報			グローバル／ローカル	グローバル
機能	NY シリーズコントローラの場合、内蔵 EtherNet/IP がオリジネータとなっている CIP 通信 1 のコネクションにおいて該当ターゲットノードのコントローラ本体の異常状態（運転停止、継続異常の論理和）を表します。本エリアの情報は、正常ターゲットノード情報が TRUE のときだけ有効です。FALSE の場合は直前の値を保持します。 配列 [x] が TRUE：ターゲットノード ID「x」のターゲットコントローラが運転継続、または、停止異常が発生しています。 配列 [x] が FALSE：上記以外				
データ型	ARRAY [0..255] OF BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	－		

変数名	_EIP_TargetNodeErr［255］				
名称	ターゲットノード異常情報			グローバル／ローカル	グローバル
機能	NY シリーズコントローラの場合、CIP 通信 1 の登録ターゲットノード情報の該当するコネクションが確立していないか、ターゲットコントローラに異常が発生していることを示します。 本エリアは、登録ターゲットノード情報が TRUE の場合だけ有効です。 配列 [x] が TRUE：ターゲットノード ID「x」のターゲットノードとのコネクションが正常に確立していない（登録ターゲットノード情報が TRUE で正常ターゲットノード情報が FALSE）、または、ターゲットノードとのコネクションが確立しているが、ターゲットコントローラに異常が発生しています。 配列 [x] が FALSE：ターゲットノード ID「x」のターゲットノードが登録されていない（登録ターゲットノード情報が FALSE）、または、ターゲットノードとのコネクションは確立していて、（登録ターゲットノード情報が TRUE で、正常ターゲットノード情報が TRUE）ターゲットコントローラに異常が発生しています（ターゲット PLC 異常情報が TRUE）。				
データ型	ARRAY [0..255] OF BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	－		

変数名	_EIP_NTPResult			メンバ名	.ExecTime
名称	NTP の最終動作時刻			グローバル／ローカル	グローバル
機能	NY シリーズコントローラの場合、初期値から変化しません。				
データ型	構造体：_sNTP_RESULT メンバ：DATE_AND_TIME			値の範囲	データ型に従う
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	不可	関連命令	GetNTPStatus 命令で読出可能		

変数名	_EIP_NTPResult			メンバ名	.ExecNormal
名称	NTP の動作結果			グローバル／ローカル	グローバル
機能	NY シリーズコントローラの場合、初期値から変化しません。				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	不可	関連命令	GetNTPStatus 命令で読出可能		

● 機能分類：EtherNet/IP 通信のスイッチ関連

変数名	_EIP_TDLinkStartCmd				
名称	タグデータリンク通信開始スイッチ			グローバル／ローカル	グローバル
機能	NY シリーズコントローラの場合、FALSE から TRUE にすると、CIP 通信 1 のタグデータリンクを開始します。 タグデータリンク動作が開始すると、自動的に FALSE になります。 （注） 自動的に FALSE になるまでは、このスイッチをユーザプログラムや Sysmac Studio などから強制的に操作しないでください。				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R/W	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	－		

変数名	_EIP_TDLinkStopCmd				
名称	タグデータリンク通信停止スイッチ			グローバル／ローカル	グローバル
機能	NY シリーズコントローラの場合、FALSE から TRUE にすると、CIP 通信 1 のタグデータリンクが停止します。 タグデータリンク動作が停止すると、自動的に FALSE になります。 （注） 自動的に FALSE になるまでは、このスイッチをユーザプログラムや Sysmac Studio などから強制的に操作しないでください。				
データ型	BOOL			値の範囲	TRUE、FALSE
R/W アクセス	R/W	保持	非保持	ネットワーク公開	公開
ユーザプログラムでの使用可否	可能	関連命令	－		

A-5 コントローラ内データの属性

コントローラ内データの、電断時、電源投入時、動作モード変更時、全停止フォールトレベルのコントローラ異常発生時の保持／非保持などの属性は、以下のとおりです。

コントローラ内データ項目			電断時の保持有無と方法 ○: 電断保持 ×: 電断非保持	電源投入時	状態変更		上書き禁止プロテクト中の書込	Sysmac Studioの転送機能	書込可能な動作モード	運転モードでの書き換え
					プログラムモード←→運転モード変更時	全停止フォールトレベルのコントローラ異常発生時		同期機能の対象		
ユーザプログラム	POUs、ユーザプログラム実行用ID（ユーザプログラム内）		○ （不揮発メモリによる）	電断前と同じ	保持	保持	不可	○	プログラムモード／運転モード（オンラインエディット）	○ オンラインエディット
タスク設定	タスク設定		○ （不揮発メモリによる）	電断前と同じ	保持	保持	不可	○	プログラムモード	×
変数	変数テーブル（変数の値を除く）	デバイス変数	○ （不揮発メモリによる）	電断前と同じ	保持	保持	不可	○	プログラムモード	×
		ユーザ定義変数	○ （不揮発メモリによる）	電断前と同じ	保持	保持	不可	○	プログラムモード／運転モード（オンラインエディット）	○ オンラインエディット
データ型		ユーザ定義データ型	○ （不揮発メモリによる）	電断前と同じ	保持	保持	不可	○	プログラムモード／運転モード（オンラインエディット）	○ オンラインエディット
コントローラ名		CPUユニット名	○ （不揮発メモリによる）	電断前と同じ	保持	保持	不可	○	プログラムモード／運転モード	○
		内蔵EtherNet/IPポート名	○ （不揮発メモリによる）	電断前と同じ	保持	保持	可	×	プログラムモード／運転モード	○

コントローラ内データ項目			電断時の保持有無と方法 ○: 電断保持 ×: 電断非保持	電源投入時	状態変更		上書き禁止プロテクト中の書込	Sysmac Studioの転送機能	書込可能な動作モード	運転モードでの書き換え
					プログラムモード←→運転モード変更時	全停止フォールトレベルのコントローラ異常発生時		同期機能の対象		
コントローラ設定	動作設定	動作設定 仮想 SD メモリカード設定 シャットダウン処理待ち時間設定 イベントログ設定 異常設定	○ （不揮発メモリによる）	電断前と同じ	保持	保持	不可	○	CPU ユニット名称：運転モード／プログラムモード その他：プログラムモード	×
	セキュリティ設定	電源投入時のプロテクト設定	○ （不揮発メモリによる）	電断前と同じ	保持	保持	不可	○	書込プロテクト、その他：プログラムモード	○
	内蔵 EtherNet/IP ポート設定	TCP/IP 内蔵 EtherNet/IP ポート LINK 設定 サービス設定 SNMP 設定 SNMP トラップ設定 NTP 設定 FTP 設定 IP ルータテーブル	○ （不揮発メモリによる）	電断前と同じ	保持	保持	不可	○	プログラムモード	×
		内蔵 EtherNet/IP ポート用タグデータリンク設定	○ （不揮発メモリによる）	電断前と同じ	保持	保持	可	×	プログラムモード／運転モード	×
モーション設定	軸割付 軸パラメータ設定 軸グループパラメータ設定 MC 共通パラメータ設定		○ （不揮発メモリによる）	電断前と同じ	保持	保持	不可	○	プログラムモード	×
カムデータ			○ （不揮発メモリによる）	電断前と同じ	保持	保持	不可	○	プログラムモード	×
イベント設定テーブル	イベント設定テーブル	ユーザ異常メッセージ	○ （不揮発メモリによる）	電断前と同じ	保持	保持	不可	○	運転モード／プログラムモード	×
EtherCAT 構成	EtherCAT ネットワーク構成	ネットワーク構成情報	○ （不揮発メモリによる）	電断前と同じ	保持	保持	不可	○	プログラムモード	×

コントローラ内データ項目			電断時の保持有無と方法 ○: 電断保持 ×: 電断非保持	電源投入時	状態変更		上書き禁止プロテクト中の書込	Sysmac Studioの転送機能	書込可能な動作モード	運転モードでの書き換え
					プログラムモード←→運転モード変更時	全停止フォールトレベルのコントローラ異常発生時		同期機能の対象		
EtherCAT設定	EtherCAT設定	マスタ	○ （不揮発メモリによる）	電断前と同じ	保持	保持	不可	○	プログラムモード	×
		スレーブ自身の設定	○ スレーブ依存	−	保持	保持	可	○	運転モード／プログラムモード	○
操作権限認証			○ （不揮発メモリによる）	電断前と同じ	保持	保持	不可	×	プログラムモード	×
ユーザプログラム実行用ID（コントローラ内）			○ （不揮発メモリによる）	電断前と同じ	保持	保持	不可	×	プログラムモード	×
変数の値	非保持変数の値	ユーザ定義変数 / デバイス変数	×	初期値	初期値	初期値	可	×	運転モード／プログラムモード	○
	保持変数の値	ユーザ定義変数 / デバイス変数	○ （不揮発メモリによる）	電断前と同じ	保持	保持	可	×	運転モード／プログラムモード	○
イベントログ	履歴	システムログ ユーザログ	○ （不揮発メモリによる）	電断前と同じ	保持	保持	可	×		○
内蔵時計	システム時計の現在時刻		○ （バッテリによる）	バッテリがある場合：保持（継続） バッテリがない場合：不定（停止している場合もあり）	保持（継続）	保持（継続）	可	×	運転モード／プログラムモード	○
絶対値エンコーダ原点位置オフセット			○ （不揮発メモリによる）	電断前と同じ	保持（継続）	保持（継続）	可	×		×

A-6 変数のメモリ確保の方法

変数のメモリ確保の方法は、構造体型や共用体型変数のメンバのメモリ上の位置を、他の機器と合わせる場合に意識する必要があります。主には、構造体型や共用体型変数を使って、他の機器と以下のような通信を行う場合に調整が必要です。

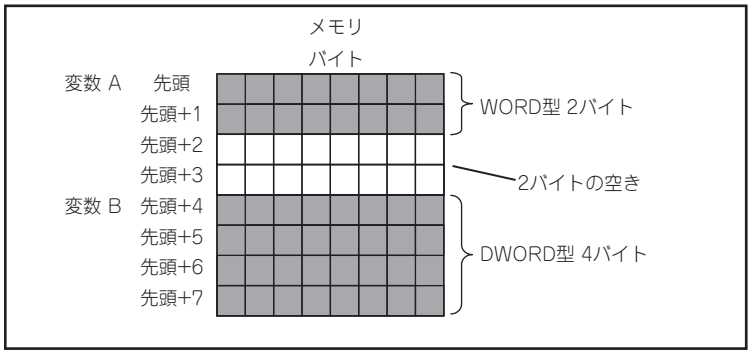
- NY シリーズコントローラと他コントローラ間で、EtherNet/IP のタグデータリンクや、CIP メッセージによる変数アクセスを行うとき
- ID タグなどのコントローラ以外の機器と、構造体型や共用体型変数でデータ交換を行うとき

A-6-1 変数のメモリ確保の規則

変数のために確保されるメモリの量とメモリ上の位置は、その変数のデータ型によって変わります。また、配列の要素、構造体のメンバ、共用体のメンバのために確保されるメモリの量とメモリ上の配置は、それらのデータ型に加えて、配列、構造体、共用体の宣言内容によって変わります。

データ型のアライメントと、確保されるメモリの量

データ型は、それぞれデータサイズが決められています。データサイズとは、そのデータ型の値を格納するために最低限必要なメモリの量を意味します。
一方、変数に対応するメモリは、最も効率よくアクセスできるようにコントローラにより自動的に配置されます。このため、変数のために確保されるメモリの総量と、それらの変数のデータサイズの合計とは必ずしも一致しません。たとえば、WORD 型と DWORD 型の変数を宣言した場合、データサイズの合計は 6 バイトですが、メモリは下図のように 8 バイト確保されます。



変数テーブル

名称	データ型
A	WORD
B	DWORD

このような、メモリ上の変数の配置が決められるための情報が、アライメントです。アライメントはデータ型ごとに決まっており、その変数のために確保されるメモリの量とメモリ上の位置は、次のとおりです。

項目	仕様
確保されるメモリの量	アライメントの整数倍。ただし、データサイズ以上。
メモリ上の位置	メモリ先頭から、アライメントの整数倍の位置。

基本データ型と列挙型の、アライメントと確保されるメモリの量は、以下のとおりです。

データ型	アライメント [バイト]	確保されるメモリの量 [バイト]
BOOL	2	2
BYTE,USINT,SINT	1	1
WORD,UINT,INT	2	2
DWORD,UDINT,DINT	4	4
LWORD,ULINT,LINT	8	8
REAL	4	4
LREAL	8	8
TIME,DATE,TIME_OF_DAT,DATE_ANT_TIME	8	8
STRING[N+1][*1]	1	N+1
列挙型	4	4

*1. N は、取り扱う最大文字数を表します。たとえば、半角英数字の最大文字数が 10 の場合、文字列終端の NULL を加えた、11 文字分のメモリを確保する必要があります。

配列、構造体型、共用体型は、それらの各要素や各メンバも、それぞれが最も効率よくアクセスできるようにメモリ上に配置されます。配列、構造体型、共用体型のアライメントと、確保されるメモリの量は、その宣言によって以下のように決まります。

データ型	アライメント	確保されるメモリの量
配列	要素のデータ型のアライメントと同じ	（要素のデータ型のために確保されるメモリの量）× 要素数[*1]
構造体型	すべてのメンバのアライメントのうち、最大の値	各メンバを、そのメンバのデータ型のアライメントの整数倍の位置に、順番に配置した際に確保されるメモリの総量以上、かつアライメントの整数倍
共用体型	すべてのメンバのアライメントのうち、最大の値	すべてのメンバの、確保されるメモリの量のうち、最大の値

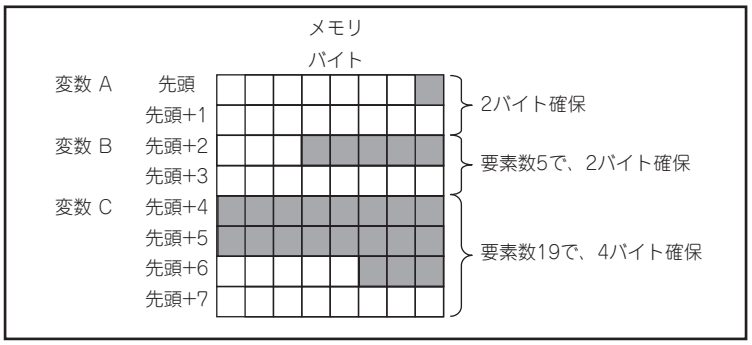
*1. BOOL 型の配列の場合は、例外です。BOOL 型の配列のために確保されるメモリの量については、後の使用上の注意を参照してください。

使用上の注意

BOOL 型の配列のために確保されるメモリの量

BOOL 型の、単独の変数、構造体のメンバ、共用体のメンバのために確保されるメモリの量は、2 バイトです。

しかし、BOOL 型の配列は、1 要素に 2 バイトずつのメモリが確保されるわけではありません。各要素に対応するメモリが 1 ビットずつ並んで確保されます。そして、配列全体としては、空きも含めて 2 バイトの整数倍のメモリが確保されます。



変数テーブル

名称	データ型
A	BOOL
B	ARRAY [1..5] OF BOOL
C	ARRAY [0..18] OF BOOL

したがって、BOOL 型の配列のために確保されるメモリの量は、以下の式で求められます。要素数が 1 ～ 16 の場合は 2 バイト、17 ～ 32 の場合は 4 バイトのメモリが確保されます。

$$\text{メモリの量}=2\left[\frac{\text{要素数-1}}{16}\right]+2$$

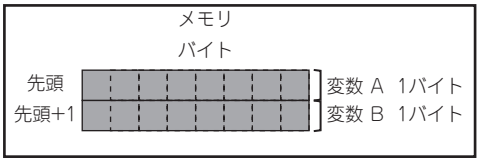
括弧内の計算結果の小数点以下は切り捨て

以下、各データ型の変数のメモリ確保の規則を、具体例を用いて説明します。

基本データ型

● BYTE 型などのアライメントが 1 バイトの変数

1 バイト単位のアライメントで、1 バイトのメモリが確保されます。
例）BYTE 型の変数が、2 個連続している場合

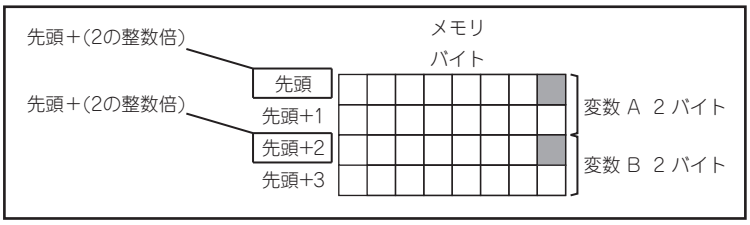


変数テーブル

名称	データ型
A	BYTE
B	BYTE

● BOOL 型、WORD 型などの、アライメントが 2 バイトの変数

2 バイト単位のアライメントで、2 バイトのメモリが確保されます。
例）BOOL 型の変数が、2 個連続している場合

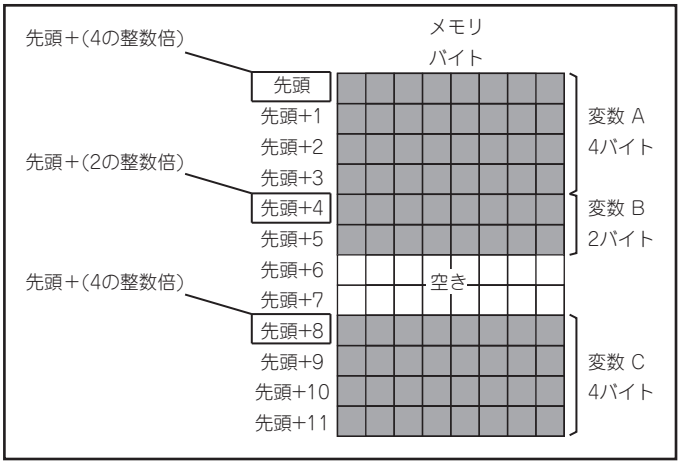


変数テーブル

名称	データ型
A	BOOL
B	BOOL

● DWORD 型などの、アライメントが 4 バイトの変数

4 バイト単位のアライメントで、4 バイトのメモリが確保されます。
データがメモリに配置される先頭位置が、4 バイトの整数倍になるので、WORD 型などのアライメントが 2 バイトの変数が挿入されると、2 バイト分空きが発生します。
例）DWORD 型、WORD 型、DWORD 型の順で、変数が連続している場合

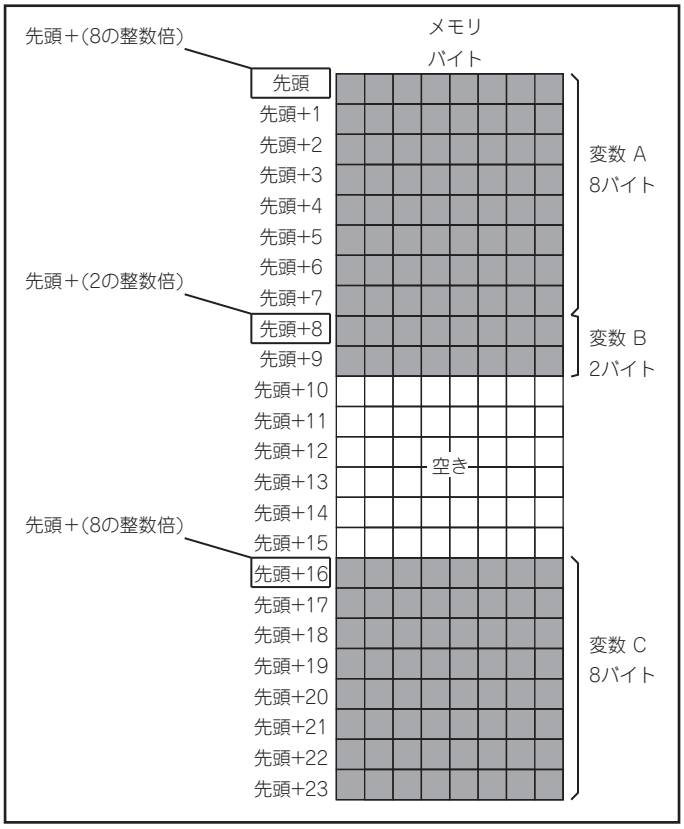


変数テーブル

名称	データ型
A	DWORD
B	WORD
C	DWORD

● LWORD 型などの、アライメントが 8 バイトの変数

8 バイト単位のアライメントで、8 バイトのメモリが確保されます。
データがメモリに配置される先頭位置が、8 バイトの整数倍になるので、WORD 型などのアライメントが 2 バイトの変数が挿入されると 6 バイト分、DWORD 型などのアライメントが 4 バイトの変数が挿入されていると 4 バイト分の、空きが発生します。
例）LWORD 型、WORD 型、LWORD 型の順で、変数が連続している場合



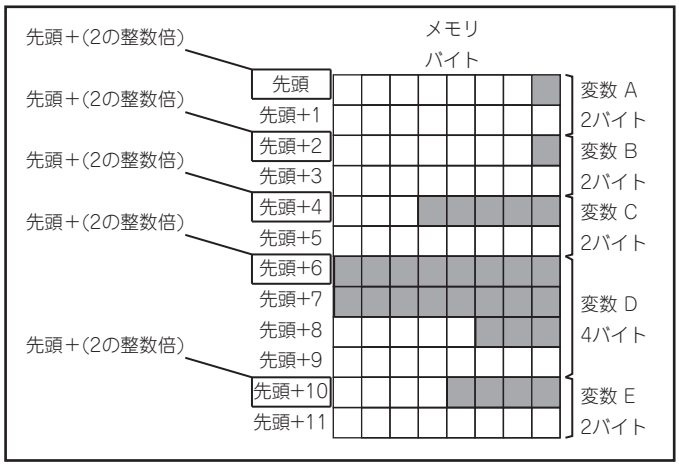
変数テーブル

名称	データ型
A	LWORD
B	WORD
C	LWORD

配列

配列変数は、配列要素のデータ型のデータサイズを単位として、要素数分だけ連続してメモリが確保されます。配列のアライメントは、要素のデータ型のアライメントと同じです。

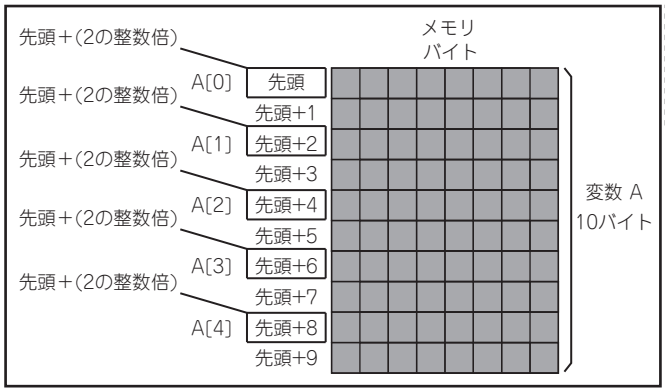
例）BOOL 型変数が 2 個、要素数 5 の BOOL 型配列、要素数 19 の BOOL 型配列、要素数 4 の BOOL 型配列の順で、変数が連続している場合



変数テーブル

名称	データ型
A	BOOL
B	BOOL
C	ARRAY [1..5] OF BOOL
D	ARRAY [0..18] OF BOOL
E	ARRAY [5..8] OF BOOL

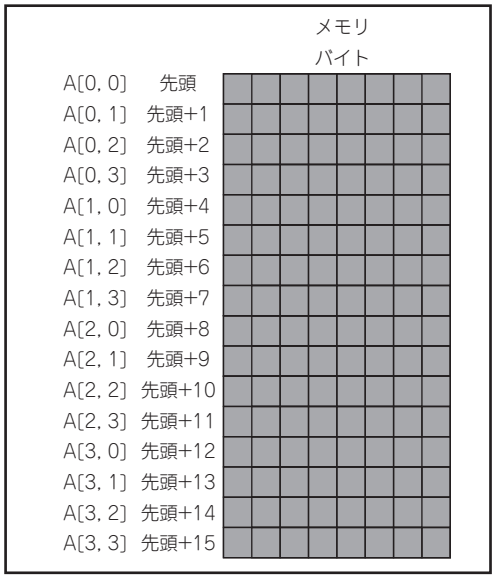
例）要素数 5 の INT 型配列の場合



変数テーブル

名称	データ型
変数 A	ARRAY[0..4] OF INT

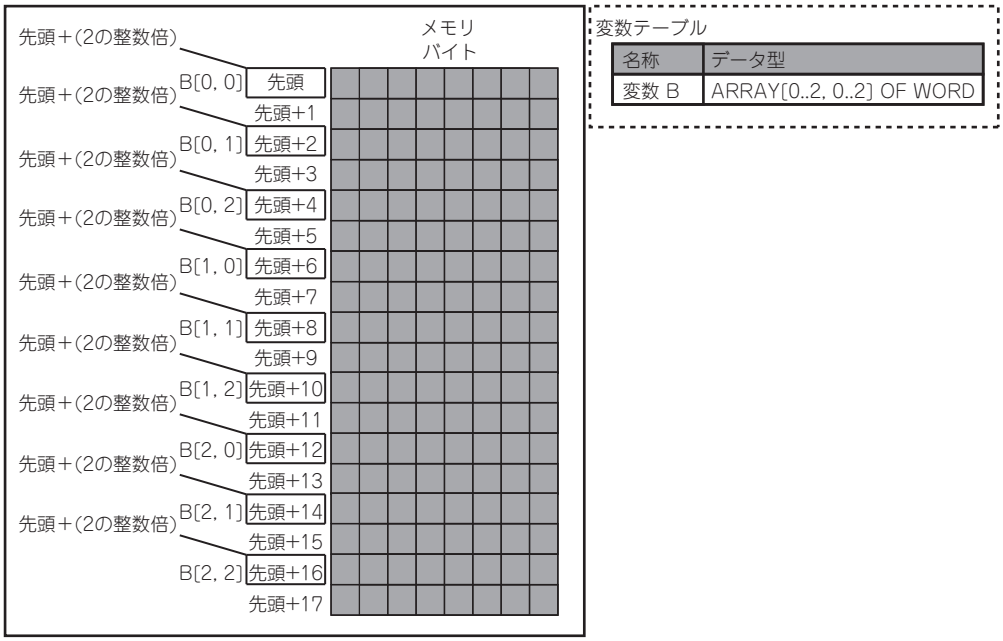
例）2 次元配列で、各次元の要素数が 4 の BYTE 型配列の場合



変数テーブル

名称	データ型
変数 A	ARRAY[0..3, 0..3] OF BYTE

例）2 次元配列で、各次元の要素数が 3 の WORD 型配列の場合

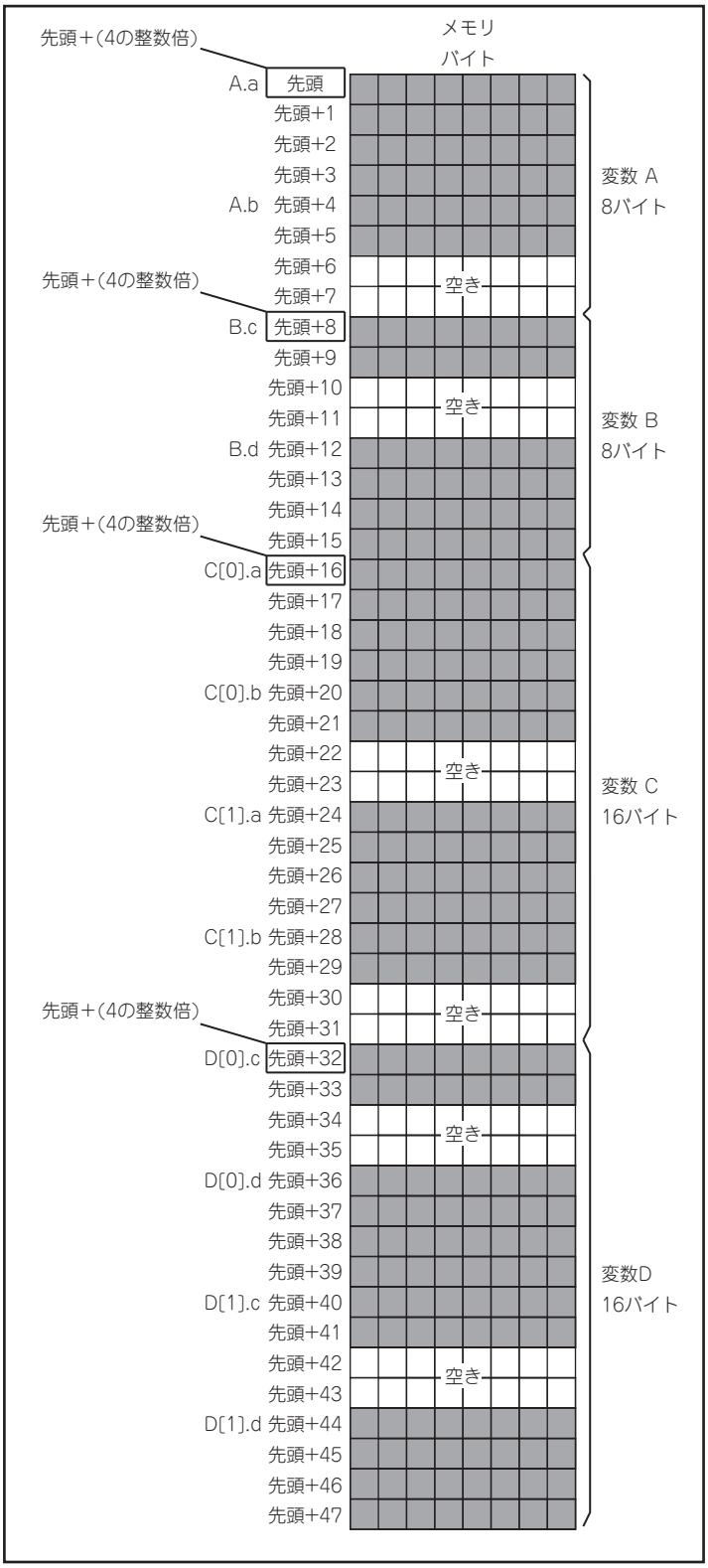


構造体型

構造体変数は、各メンバが宣言された順番にメモリに配置されます。また、各メンバが、そのメンバのデータ型のアライメントの整数倍の位置に配置されます。したがって、メンバ間やメンバの終端にメモリの空きが挿入される場合があります。構造体型のアライメントは、すべてのメンバのアライメントのうちの最大の値です。確保されるメモリの量は、各メンバを、そのメンバのデータ型のアライメントの整数倍の位置に順番に配置した際に確保されるメモリの総量以上で、かつアライメントの整数倍になります。

例）下図の 4 つの変数宣言の場合、それぞれのアライメントと、確保されるメモリの量は、以下の値になります。

変数	アライメント [バイト]	確保されるメモリの量 [バイト]
A	4	8
B	4	8
C	4	16
D	4	16



データ型定義

名称	データ型
構造体 STR_A	STRUCT
a	DINT
b	INT

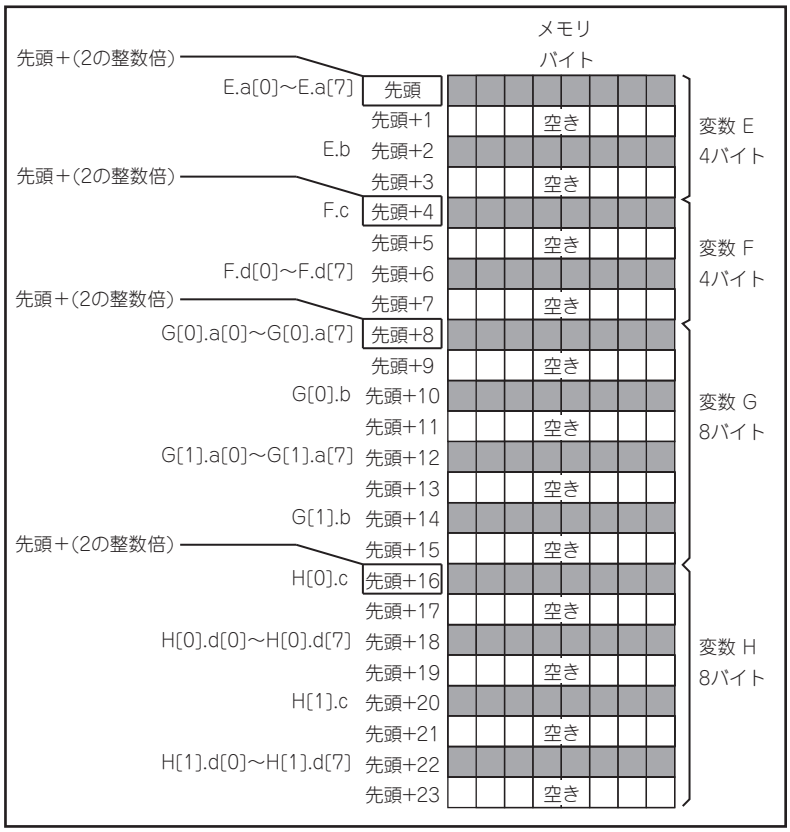
名称	データ型
構造体 STR_B	STRUCT
c	INT
d	DINT

変数テーブル

名称	データ型
変数 A	構造体 STR_A
変数 B	構造体 STR_B
変数 C	ARRAY[0..1] OF STR_A
変数 D	ARRAY[0..1] OF STR_B

例）下図の 4 つの変数宣言の場合、それぞれのアライメントと、確保されるメモリの量は、以下の値になります。

変数	アライメント [バイト]	確保されるメモリの量 [バイト]
E	2	4
F	2	4
G	2	8
H	2	8



データ型定義

名称	データ型
構造体 STR_C	STRUCT
a	ARRAY[0..7] OF BOOL
b	BYTE

名称	データ型
構造体 STR_D	STRUCT
c	BYTE
d	ARRAY[0..7] OF BOOL

変数テーブル

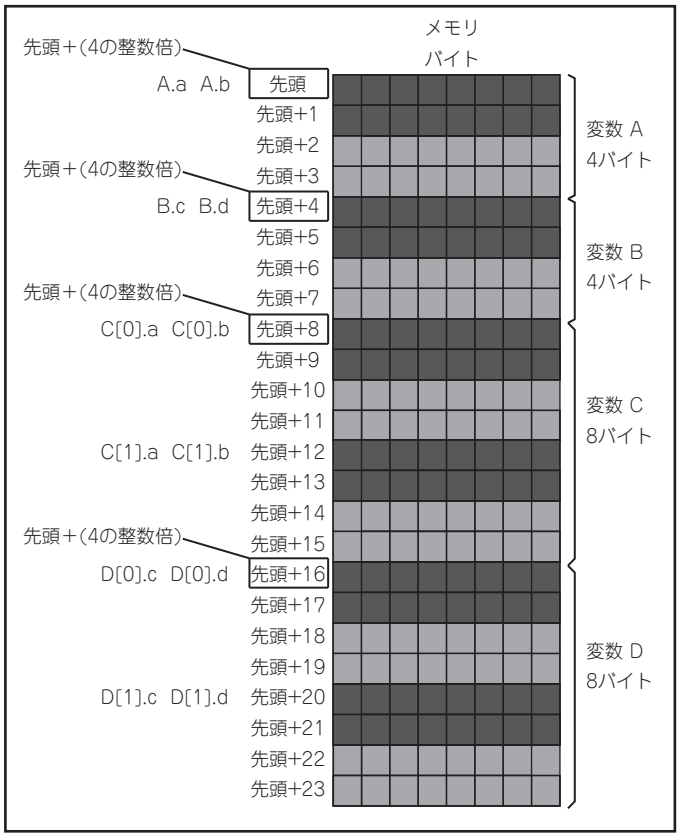
名称	データ型
変数 E	構造体 STR_C
変数 F	構造体 STR_D
変数 G	ARRAY[0..1] OF STR_C
変数 H	ARRAY[0..1] OF STR_D

共用体型

共用体型変数は、各メンバがメモリ上の同じ位置に重なって配置されます。
共用体型のアライメントは、すべてのメンバのアライメントのうちの最大の値です。確保されるメモリの量は、すべてのメンバの、確保されるメモリの量のうちの最大の値です。

例）下図の 4 つの変数宣言の場合、それぞれのアライメントと、確保されるメモリの量は、以下の値になります。

変数	アライメント [バイト]	確保されるメモリの量 [バイト]
A	4	4
B	4	4
C	4	8
D	4	8



データ型定義

名称	データ型
共用体 UNI_A	UNION
a	DWORD
b	WORD

名称	データ型
共用体 UNI_B	UNION
c	WORD
d	DWORD

変数テーブル

名称	データ型
変数 A	共用体 UNI_A
変数 B	共用体 UNI_B
変数 C	ARRAY[0..1] OF UNI_A
変数 D	ARRAY[0..1] OF UNI_B

A-6-2　留意するべきケース例

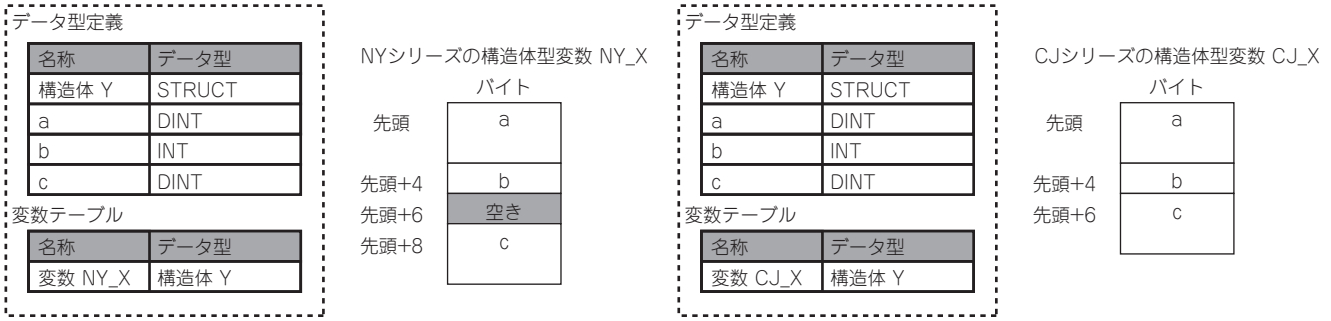
NY シリーズコントローラの構造体型変数で、他の機器との通信などでデータ交換するときには、構造体型変数のメンバのメモリ配置を、通信する相手機器と一致させる必要があります。
その場合は、NY シリーズコントローラ側、または通信相手の機器側か、いずれかで本項目で説明する対策を行ってください。

参考

NY シリーズコントローラどうしでデータ交換する場合は、特にメンバのメモリ配置を意識する必要はありません。

通信する相手機器とメモリ配置を一致させる方法

通信する相手機器とメモリ配置を一致させるには、2 つの方法があります。
たとえば、NY シリーズコントローラと CJ シリーズの CPU では、構造体型変数のメモリ構成が、以下のように異なります。
本項では、そのメモリ配置を一致させる方法を例に説明します。

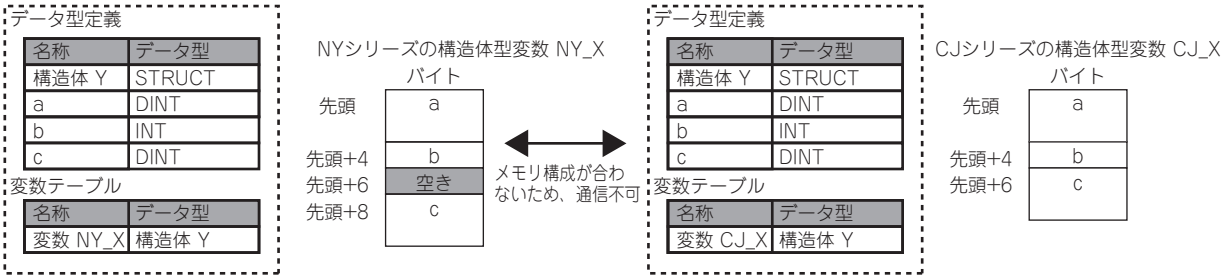


● 対策1:NYシリーズコントローラの構造体型変数のメンバのメモリ配置を変更する

NY シリーズコントローラでは、メンバのオフセット指定によって、構造体型変数ごとにメンバのメモリ配置を設定できます。NY シリーズコントローラ側で、構造体型変数のメンバのメモリ配置を、通信する相手機器に合わせて変更し、調整します。
構造体型変数のメンバのオフセット指定は、構造体型の登録時に行います。
通信する相手機器が CJ シリーズ CPU ユニットの場合、オフセット種別を「CJ」と選択することで自動的に CJ のメモリ配置にすることができます。
また、オフセット種別を「任意」と選択することでオフセットを自由に決めることができます。

本設定で構造体型変数のメモリ配置を調整した場合は、ネットワーク上にある他の NY シリーズコントローラにも同様の設定変更を行ってください。
構造体型変数のメモリ配置の操作方法については、🕮『Sysmac Studio Version 1 オペレーションマニュアル（SBCA-470)』を参照してください。

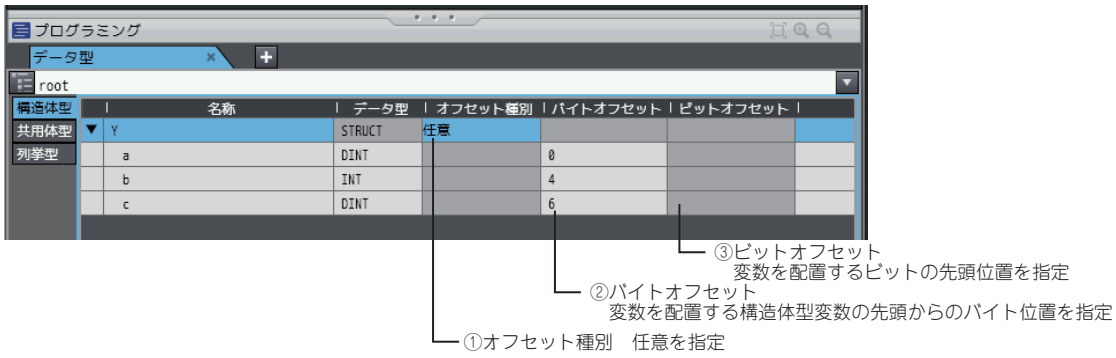
例）NY シリーズコントローラの構造体型変数のメンバのメモリ配置を、CJ シリーズ CPU ユニットの構造体型変数のメンバにあわせて変更する

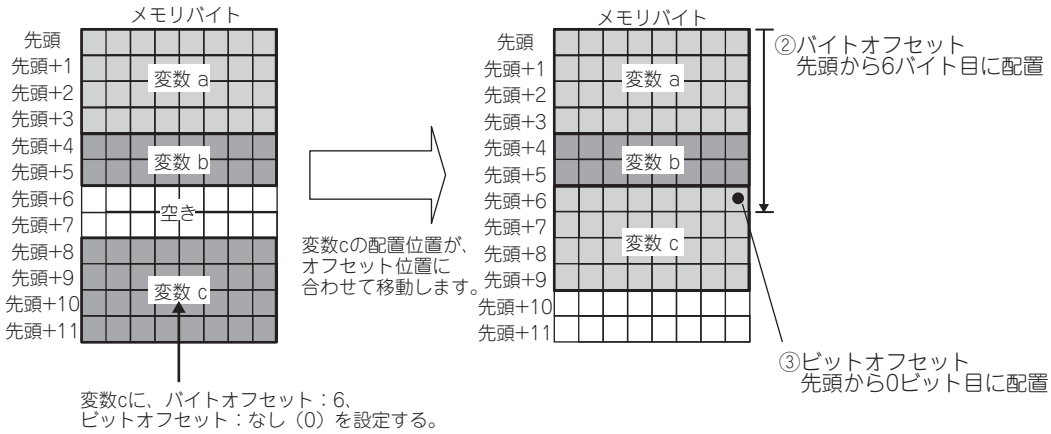


NYシリーズコントローラと、CJシリーズCPUユニットとのメモリ構成を合わすため、
Sysmac Studioオフセット指定します。

構造体型変数 NY_Xのデータ型Yのメンバcに、以下のオフセット値を指定します。



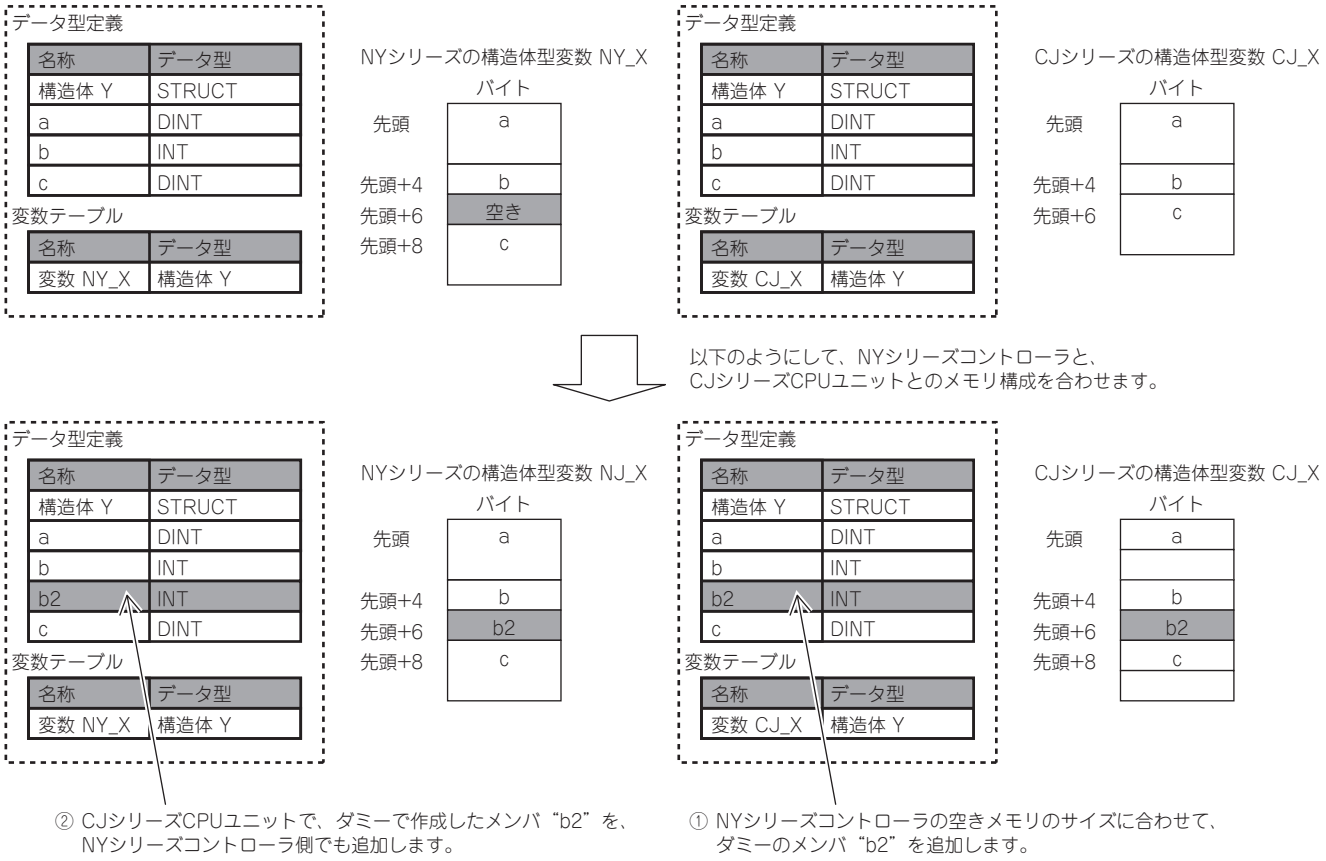




● 対策 2：通信する相手機器の構造体型変数のメモリ配置を変更する

NY シリーズコントローラの構造体型変数のメモリ配置に合うように、通信する相手機器の構造体型変数のメモリ配置を、配置調整用のメンバなどを挿入して調整します。
また、メモリ配置だけでなく、データ型も一致させる必要があるので、配置調整用のメンバは、通信する相手機器、NY シリーズコントローラともに作成する必要があります。

例）CJ シリーズ CPU ユニットの構造体型変数のメモリ配置を、NY シリーズコントローラの構造体型変数のメモリ配置にあわせて変更する



A-7 CX-Designer の変数テーブル登録方法

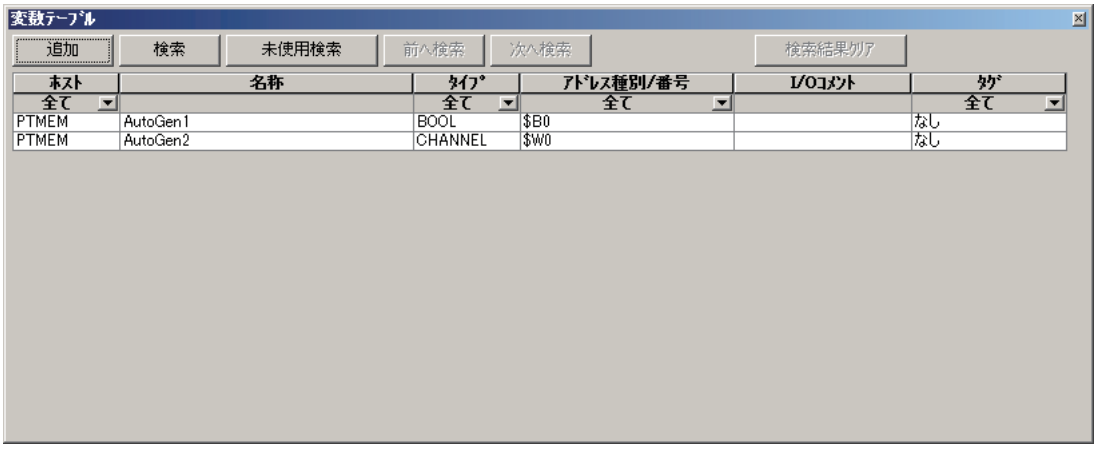
NY シリーズコントローラと NS シリーズ表示器を接続して使用する場合、CX-Designer では、各機能部品などに設定するアドレスとして変数を使用することができます。変数は、[変数テーブル] で管理します。ここでは、Microsoft Excel から変数表をコピーすることで、複数の変数を一括して変数テーブルに登録する方法を説明します。CX-Designer の詳細は、🕮『CX-Designer ユーザーズマニュアル（SBSA-532)』を参照してください。

1 Microsoft Excel で、以下の書式で変数表を作成します。

表の列の数や配置は、以下の書式どおりにしてください。[アドレス種別／番号] や [I/O コメント] のように、空白セルばかりの列も省略しないでください。

ホスト	名称	タイプ	アドレス種別 / 番号	I/O コメント	タグ
HOST3	_Card1BkupCmd.ExecBkup	BOOL			TRUE
HOST3	_Card1BkupCmd.CancelBkup	BOOL			TRUE
HOST3	_Card1BkupCmd.ExecVefy	BOOL			TRUE
HOST3	_Card1BkupCmd.CancelVefy	BOOL			TRUE
HOST3	_Card1BkupCmd.DirName	STRING(64)			TRUE
HOST3	_Card1BkupSta.Done	BOOL			TRUE
HOST3	_Card1BkupSta.Active	BOOL			TRUE
HOST3	_Card1BkupSta.Err	BOOL			TRUE
HOST3	_Card1VefySta.Done	BOOL			TRUE
HOST3	_Card1VefySta.Active	BOOL			TRUE
HOST3	_Card1VefySta.VefyRslt	BOOL			TRUE
HOST3	_Card1VefySta.Err	BOOL			TRUE
HOST3	_BackupBusy	BOOL			TRUE

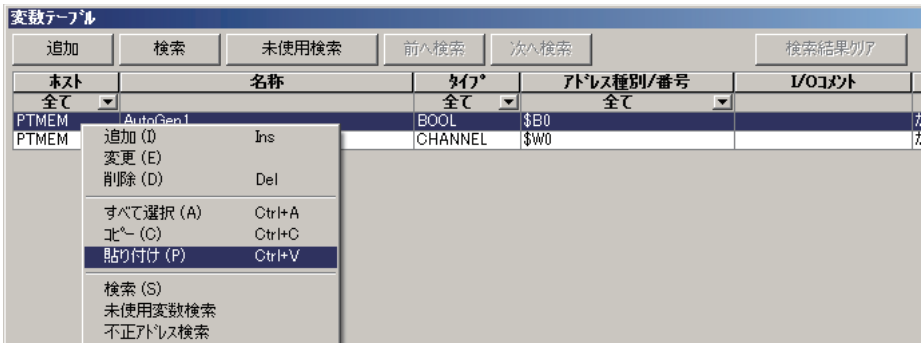
2 CX-Designer を起動し、変数テーブル画面を開きます。



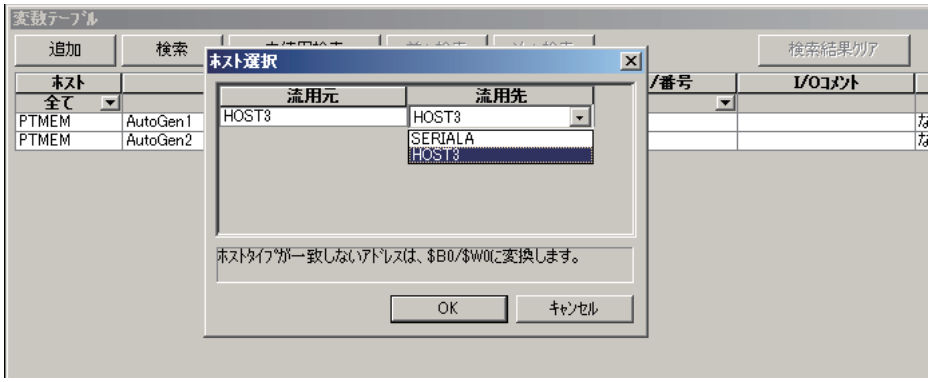
3 **Microsoft Excel の変数表の、網かけ部分をコピーします。**
必ずすべての列をコピーしてください。

ホスト	名称	タイプ	アドレス種別 / 番号	I/O コメント	タグ
HOST3	_Card1BkupCmd.ExecBkup	BOOL			TRUE
HOST3	_Card1BkupCmd.CancelBkup	BOOL			TRUE
HOST3	_Card1BkupCmd.ExecVefy	BOOL			TRUE
HOST3	_Card1BkupCmd.CancelVefy	BOOL			TRUE
HOST3	_Card1BkupCmd.DirName	STRING(64)			TRUE
HOST3	_Card1BkupSta.Done	BOOL			TRUE
HOST3	_Card1BkupSta.Active	BOOL			TRUE
HOST3	_Card1BkupSta.Err	BOOL			TRUE
HOST3	_Card1VefySta.Done	BOOL			TRUE
HOST3	_Card1VefySta.Active	BOOL			TRUE
HOST3	_Card1VefySta.VefyRslt	BOOL			TRUE
HOST3	_Card1VefySta.Err	BOOL			TRUE
HOST3	_BackupBusy	BOOL			TRUE

4 **CX-Designer の変数テーブル画面で右クリックし、[貼り付け] を実行します。**



5 **CX-Designer のホスト選択画面で、NY シリーズのホストを選択し [OK] ボタンを押します。**



これで、CX-Designer の変数テーブル画面に、変数が登録されます。



ホスト	名称	タイプ	アドレス種別/番号	I/Oコメント	タグ
全て		全て	全て		全て
PTMEM	AutoGen1	BOOL	$B0		なし
PTMEM	AutoGen2	CHANNEL	$W0		なし
HOST3	_Card1BkupCmd.ExecBkup	BOOL			ネットワーク変数
HOST3	_Card1BkupCmd.CancelBkup	BOOL			ネットワーク変数
HOST3	_Card1BkupCmd.ExecVefy	BOOL			ネットワーク変数
HOST3	_Card1BkupCmd.CancelVefy	BOOL			ネットワーク変数
HOST3	_Card1BkupCmd.DirName	STRING(64)			ネットワーク変数
HOST3	_Card1BkupSta.Done	BOOL			ネットワーク変数
HOST3	_Card1BkupSta.Active	BOOL			ネットワーク変数
HOST3	_Card1BkupSta.Err	BOOL			ネットワーク変数
HOST3	_Card1VefySta.Done	BOOL			ネットワーク変数
HOST3	_Card1VefySta.Active	BOOL			ネットワーク変数
HOST3	_Card1VefySta.VefyRslt	BOOL			ネットワーク変数
HOST3	_Card1VefySta.Err	BOOL			ネットワーク変数
HOST3	_BackupBusy	BOOL			ネットワーク変数

A-8 EtherCAT スレーブや軸の有効／無効切替

EtherCAT スレーブや軸の有効／無効切替を、命令を使って行うことが可能です。この機能により、以下のようなアプリケーションに対応できます。

- EtherCAT スレーブ構成や軸構成が異なる複数の装置を、Sysmac Studio のひとつのプロジェクトで管理できます。
- 一方の生産ラインを稼働させたまま、他方の生産ラインのEtherCATスレーブ構成や軸構成を変更することができます。

ここでは、本機能に使用する命令、システム定義変数、およびアプリケーション例について説明します。

A-8-1 EtherCAT スレーブや軸を使用する場合のプロジェクト設定

電源投入時やプロジェクトをダウンロードするときに、EtherCAT ネットワークに取り付けられていない可能性がある EtherCAT スレーブは、あらかじめ無効設定しておいてください。同様に、軸使用も未使用軸に設定しておいてください。EtherCAT ネットワークに取り付けられていない EtherCAT スレーブが有効設定されていたり、軸使用が使用軸に設定されていたりすると、運転実行時に異常が発生する可能性があります。

参考

- EtherCAT スレーブの有効／無効切替を、Sysmac Studio の [構成・設定]-[EtherCAT]-[ネットワーク構成]-[有効／無効設定] の設定で行うことも可能です。ただし、その方法だと、後述の「アプリケーション 1　EtherCAT スレーブ構成や軸構成が異なる装置の一元管理」(P.A-103) の場合、装置によって Sysmac Studio の設定をその都度手動で変更するか、プロジェクトファイルを変更しなければなりません。
- 無効設定したEtherCATスレーブは、EtherCATネットワークから取り外したり取り付けたりすることが可能になります。

A-8-2 EtherCAT スレーブや軸の有効／無効切替を行う命令

EtherCAT スレーブや軸の有効／無効切替は、ユーザプログラム内の命令を使用して実行します。EtherCAT スレーブの有効／無効切替と軸の有効／無効切替とは、異なる命令を使用します。それぞれの命令名は、以下のとおりです。

切替対象	命令名
EtherCAT スレーブ	EC_ChangeEnableSetting(EtherCAT スレーブ有効／無効切替) 命令
軸	MC_ChangeAxisUse(軸使用変更）命令

EC_ChangeEnableSetting 命令

EtherCAT スレーブの有効／無効切替には、EC_ChangeEnableSetting(EtherCAT スレーブ有効無効切替)命令を使用します。EC_ChangeEnableSetting 命令は、指定したノードアドレスの EtherCAT スレーブを、有効にしたり無効にしたりできる命令です。本命令を実行後、コントローラの電源を再投入すると、命令実行前の設定に戻ります。

EC_ChangeEnableSetting 命令の仕様の詳細は、🕮『NY シリーズ コマンドリファレンスマニュアル 基本編（SBCA-437)』を参照してください。

MC_ChangeAxisUse 命令

軸の有効／無効切替には、MC_ChangeAxisUse（軸使用変更）命令を使用します。MC_ChangeAxisUse 命令は、指定した軸パラメータの [軸使用] を、使用軸にしたり未使用軸にしたりできる命令です。本命令を実行後、コントローラの電源を再投入すると、命令実行前の設定に戻ります。

MC_ChangeAxisUse 命令の仕様の詳細は、🕮『NY シリーズ コマンドリファレンスマニュアル モーション編（SBCE-380）』を参照してください。

A-8-3 EtherCAT スレーブや軸の状態を示すシステム定義変数

EtherCAT スレーブや軸の現在の状態は、システム定義変数の値を参照して知ることができます。それぞれのシステム定義変数は、以下のとおりです。

状態を参照する対象	システム定義変数名
EtherCAT スレーブ	_EC_DisableSlavTbl[]（無効設定スレーブテーブル）
軸	_MC_AX[].Cfg.AxEnable（軸使用）

_EC_DisableSlavTbl[]（無効設定スレーブテーブル）

_EC_DisableSlavTbl[]（無効設定スレーブテーブル）は、EtherCAT スレーブの現在の状態が、無効かどうかを示すシステム定義変数です。配列の添え字には、ノードアドレスを指定します。_EC_DisableSlavTbl[]（無効設定スレーブテーブル）の値の意味は、以下のとおりです。

値	意味
TRUE	指定したノードアドレスの EtherCAT スレーブが無効
FALSE	指定したノードアドレスの EtherCAT スレーブが有効

_MC_AX[].Cfg.AxEnable（軸使用）

_MC_AX[].Cfg.AxEnable（軸使用）は、軸が未作成軸かどうか、および軸を使用するかどうかを示すシステム定義変数です。配列の添え字には、軸番号を指定します。_MC_AX[].Cfg.AxEnable（軸使用）の値の意味は、以下のとおりです。

値	意味
0:_mcNoneAxis	指定した軸が未作成軸
1:_mcUnusedAxis	指定した軸が未使用軸
2:_mcUsedAxis	指定した軸が使用軸

A-8-4　プログラムの実行／停止指令

有効／無効切替を行う EtherCAT スレーブや軸には、特定のプログラムが付随している場合があります。その場合、EtherCAT スレーブや軸の有効／無効切替にともなって、付随するプログラムを実行させたり停止させたりする必要があります。特定のプログラムの実行／停止指令は、ユーザプログラム内で、以下の命令を使って行うことができます。

機能	命令名
プログラムの実行指令	PrgStart 命令
プログラムの停止指令	PrgStop 命令

PrgStart 命令、PrgStop 命令の仕様の詳細は、🕮『NY シリーズ コマンドリファレンスマニュアル 基本編（SBCA-437）』を参照してください。

使用上の注意

プログラムを停止させる場合は、そのプログラムが付随する EtherCAT スレーブや軸の無効化を、プログラム停止よりも前に実行してください。

A-8-5　プログラムの実行／停止の確認

有効／無効切替を行う EtherCAT スレーブや軸に付随しているプログラムに対して、実行が指令されているか停止が指令されているかを、PrgStatus 命令を使って確認することができます。PrgStatus 命令の仕様の詳細は、🕮『NY シリーズ コマンドリファレンスマニュアル 基本編（SBCA-437）』を参照してください。

A-8-6 Sysmac Studio での設定

EtherCAT スレーブの有効／無効、軸の有効／無効、およびプログラムの運転開始時の実行／停止の各設定は、Sysmac Studio でも変更可能です。アプリケーションによっては、EtherCAT スレーブ、軸、および運転開始時のプログラムの状態を、Sysmac Studio で設定しておく場合があります。

Sysmac Studio での EtherCAT スレーブの有効／無効設定

Sysmac Studio で EtherCAT スレーブを有効にするには、以下の手順で設定を行います。

1 **[構成・設定]-[EtherCAT] を右クリックし、メニューから [編集] を選択します。**
EtherCAT 構成編集画面が表示されます。

2 **ツールボックスで、接続する EtherCAT スレーブを右クリックし、メニューから [挿入] を選択します。**
EtherCAT 構成編集画面で、EtherCAT マスタの下に、選択した EtherCAT スレーブが表示されます。また、EtherCAT 構成編集画面の右側に、EtherCAT スレーブのパラメータ設定画面が表示されます。

3 **EtherCAT スレーブのパラメータ設定画面で、[有効／無効設定] の設定値を、「有効」にします。**

Sysmac Studio での軸の有効／無効設定

Sysmac Studio で軸を有効にするには、以下の手順で設定を行います。

1 **[構成・設定]-[モーション制御設定]-[軸設定]を右クリックし、メニューから[追加]-[軸設定]を選択します。**
[軸設定] の下に、「MC_Axis000(0)」などの軸変数が追加されます。

2 **「MC_Axis000(0)」などの軸変数名を右クリックし、メニューから [編集] を選択します。**
軸パラメータ設定の「軸基本設定」画面が表示されます。

3 **[軸使用] を「使用軸」に設定します。**

Sysmac Studio での運転開始時のプログラム起動／停止設定

Sysmac Studio で、運転開始時にプログラムを実行させるためには、以下の手順で設定を行います。

1 **[構成・設定]-[タスク設定] を右クリックし、メニューから [編集] を選択します。**
タスク設定画面が、表示されます。

2 **[プログラムの割付設定] ボタンをクリックします。**
タスク設定の、プログラムの割付設定画面が表示されます。

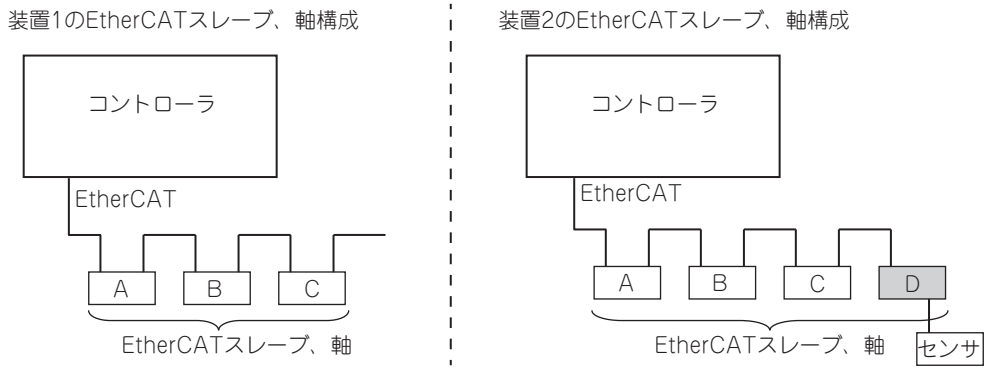
3 **プログラムの割付設定画面で、当該プログラムの [初期状態] を、「起動」にします。**

A-8-7　EtherCAT スレーブや軸の有効／無効切替のアプリケーション例

EtherCAT スレーブや軸の有効／無効切替の使用方法を、具体的なアプリケーションを例にとって説明します。

アプリケーション1　EtherCAT スレーブ構成や軸構成が異なる装置の一元管理

NY シリーズコントローラの EtherCAT スレーブや軸構成が、装置 1 と装置 2 で下図のように異なっているとします。これを、Sysmac Studio のひとつのプロジェクトで一元管理します。



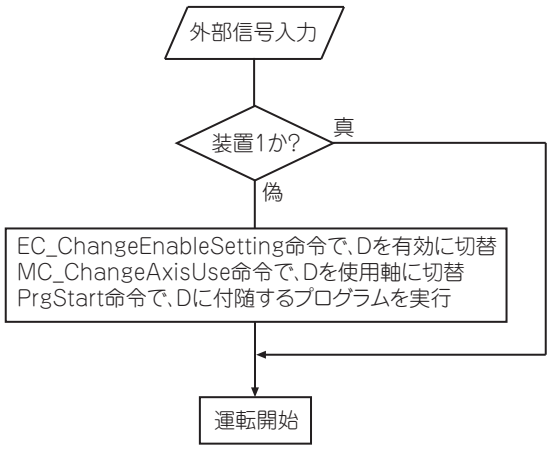
Sysmac Studio のプロジェクトでは、図中の A,B,C,D の 4 つの EtherCAT スレーブや軸のすべてに対して EtherCAT スレーブ構成を作成しておきます。そして、各 EtherCAT スレーブの有効／無効、軸使用の使用軸／未使用軸、および付随するプログラムの運転開始時の起動／停止の状態は、Sysmac Studio で以下のように装置 1 に合わせて設定しておきます。

EtherCAT スレーブ	EtherCAT ネットワークへの取付	有効／無効	軸使用	付随するプログラム
A,B,C	あり	有効	使用軸	運転開始時、起動
D	なし	無効	未使用軸	運転開始時、停止

装置 2 に対応する場合は、命令によって、EtherCAT スレーブの有効／無効、軸使用の使用軸／未使用軸、および付随するプログラムの実行／停止の設定を、以下のように切り替えます。

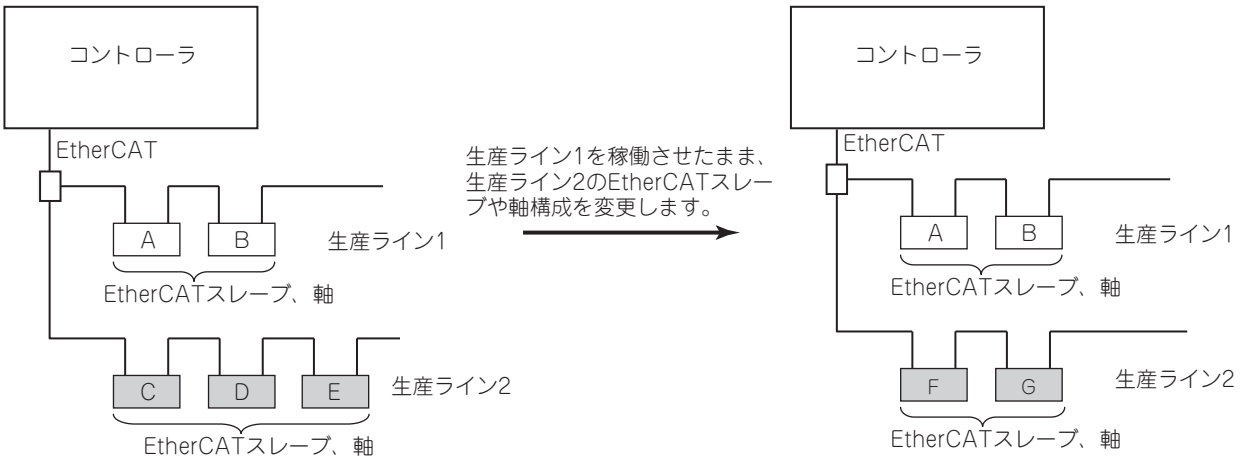
EtherCAT スレーブ	EtherCAT ネットワークへの取付	有効／無効	軸使用	付随するプログラム
A,B,C	あり	有効	使用軸	実行
D	あり	有効	使用軸	実行

ユーザプログラムのアルゴリズムは、下図のようにします。コントローラ外部から、運転する装置が装置 1 か装置 2 かを示す信号が入力されるとします。



アプリケーション 2　EtherCAT スレーブ構成や軸構成の、稼働中の変更

下図の、生産ライン 1 を稼働させたまま、生産ライン 2 の EtherCAT スレーブや軸構成を変更します。



Sysmac Studio のプロジェクトでは、図中の A ～ G の 7 つの EtherCAT スレーブや軸のすべてに対して EtherCAT スレーブ構成を作成しておきます。
構成変更前の、A ～ G の EtherCAT スレーブの有効／無効、軸使用の使用軸／未使用軸、および付随するプログラムの運転開始時の起動／停止の状態は、Syamac Studio を使用して以下のように設定しておきます。

EtherCAT スレーブ	EtherCAT ネットワークへの取付	有効／無効	軸使用	付随するプログラム
A,B	あり	有効	使用軸	運転開始時、起動
C,E,D	あり	有効	使用軸	運転開始時、起動
F,G	なし	無効	未使用軸	運転開始時、停止

使用する EtherCAT スレーブや軸を、C,D,E から F,G に変更する手順は、以下のとおりです。

1 **生産ライン 2 を停止させます。**

2 **MC_ChangeAxisUse 命令を使用して、C,D,E の [軸使用] を未使用軸にします。**

3 **EC_ChangeEnableSetting 命令を使用して、C,D,E の EtherCAT スレーブ設定を無効にします。**

4 **PrgStop 命令を使用して、C,D,E に付随するプログラムを停止させます。**

5 **EtherCAT スレーブ C,D,E を生産ライン 2 から外します。**

6 **EtherCAT スレーブ F,G を生産ライン 2 に取り付けます。**

7 **EC_ChangeEnableSetting 命令を使用して、F,G の EtherCAT スレーブ設定を有効にします。**

8 **MC_ChangeAxisUse 命令を使用して、F,G の [軸使用] を使用軸にします。**

9 **PrgStart 命令を使用して、F,G に付随するプログラムを実行します。**

10 **生産ライン 2 を再稼働させます。**

上記の結果、A ～ G の EtherCAT スレーブの有効／無効、軸使用の使用軸／未使用軸、および付随するプログラムの実行／停止の設定は、以下のように変わります。

EtherCAT スレーブ	EtherCAT ネットワークへの取付	有効／無効	軸使用	付随するプログラム
A,B	あり	有効	使用軸	実行
C,E,D	なし	無効	未使用軸	停止
F,G	あり	有効	使用軸	実行

使用上の注意

プログラムを停止させる場合は、そのプログラムが付随する EtherCAT スレーブや軸の無効化を、プログラム停止よりも前に実行してください。

A-9 ユーザプログラムの大きさに関する制限

ユーザプログラムには、コントローラ内のメモリ容量などの制約により、大きさに関する制限があります。制限を超えると、運転時に異常となります。ここでは、1 台のコントローラ上で作成できるユーザプログラムの大きさに関する制限を、項目ごとに説明します。
ユーザプログラムや変数の大きさの概略値は、Sysmac Studio のメモリ表示機能を使用して確認することができます。
ユーザプログラムを作成する際には、制限に注意してください。ただし、書かれている制限値はあくまでも目安として参考にするためのものです。将来のユーザプログラム拡張の可能性も含めて、制限値よりも余裕をもったユーザプログラムを作成することを推奨します。

使用上の注意

オンラインエディットを使用した場合、ユーザプログラムの大きさが制限値以下であっても異常となることがあります。これは、オンラインエディットによってユーザプログラムを変更しても、コントローラ内のメモリに確保されたままになるデータがあることによります。異常が発生した場合は、コントローラをプログラムモードにして、ユーザプログラムをコントローラに再転送すると、異常解除できることがあります。

A-9-1 ユーザプログラムの各部品の制限

ユーザプログラムの各部品の制限を説明します。制限がある部品は、以下のとおりです。

- POU
- 変数
- データ型定義
- 定数（リテラル）

POU の制限

POU の制限には、POU の定義の制限と POU のインスタンスの制限があります。

● POU の定義の制限

POU の定義には、以下の制限があります。

制限項目	形 NY5□2-□□□□
プログラム数の最大値	500
（FB 定義数＋ FUN 定義数＋ラダー図プログラムのセクション数）の最大値	3,000
（FB 定義と FUN 定義の、入力／出力／入出力変数の合計個数）の最大値	64

● POU のインスタンスの制限

POU のインスタンスには、以下の制限があります。

制限項目	形 NY5□2-□□□□
POU のインスタンス数の最大値	24,000

POU のインスタンスの数え方の詳細は、🕮「POU インスタンスの数」（P.A-108）を参照してください。

変数の制限

変数の制限には、変数の使用量の制限と変数定義の制限があります。

● 変数の使用量の制限

変数の使用量には、以下の制限があります。

制限項目	形 NY5□2-□□□□
保持属性なしの変数の合計サイズ [MB] の最大値*1	64
保持属性ありの変数の合計サイズ [MB] の最大値 *1	4
保持属性なしの変数の数*2 の最大値	180,000
保持属性ありの変数の数*3 の最大値	40,000
ネットワーク変数の数の最大値	40,000

*1. 変数　1　個あたりのデータサイズは、データ型によって異なります。各データ型のサイズは、🕮「6-3-5 データ型」（P.6-30）を参照してください。
*2. 数え方の詳細は、🕮「保持属性なしの変数の数」（P.A-108）を参照してください。
*3. 数え方の詳細は、🕮「保持属性ありの変数の数」（P.A-109）を参照してください。

● 変数定義の制限

変数定義には、以下の制限があります。

制限項目	形 NY5□2-□□□□
配列 1 個あたりの要素数の最大値	65,535
配列の次元数の最大値	3
配列 1 個あたりのサイズ [MB] の最大値	8
配列の添え字（要素番号）の値の最大値	65,535
文字列型変数のサイズ [バイト]*1 の最大値	1,986

*1. 終端 NULL 文字を含めます。したがって、サイズが 1986 バイトの場合の文字数は、半角英数字で 1985 文字です。

データ型定義の制限

データ型定義には、以下の制限があります。

制限項目	形 NY5□2-□□□□
データ型定義の数*1 の最大値	4,000
構造体型定義 1 個あたりの階層数の最大値	8
構造体型定義 1 個あたりのメンバ数の最大値	2,048
構造体型変数 1 個あたりのサイズ [MB] の最大値	8
共用体型定義 1 個あたりのメンバ数の最大値	4
列挙型定義 1 個あたりの列挙子数の最大値	2,048

*1. 数え方の詳細は、🕮「データ型定義の数」（P.A-109）を参照してください。

定数（リテラル）の制限

定数（リテラル）には、以下の制限があります。

制限項目	形 NY5□2-□□□□
文字列型定数（リテラル）のサイズ [バイト] の最大値	1,985

A-9-2　ユーザプログラムの各部品の数

POU インスタンス、保持属性なしの変数、保持属性ありの変数、およびデータ型定義の数え方を示します。ただし、ここに書かれている内容はあくまでも目安です。NY シリーズコントローラのユニットバージョンの違いなどによって、部品の数え方が少し異なる場合があります。したがって、ユーザプログラムの大きさが適切かどうかは、必ず Sysmac Studio で確認してください。

POU インスタンスの数

POU インスタンスは、以下のように数えます。

● POU インスタンスの計数対象

POU インスタンスの計数対象は、以下のとおりです。

- プログラム
- FB インスタンス（ユーザ作成と命令の両方を含む）
- FUN（ユーザ作成と命令の両方を含む）

● POU インスタンスの数え方の注意

POU インスタンスを数える場合に、以下の点に注意してください。

- 同一の FB 定義に対する FB インスタンスを n 個使用する場合は、n 個と数えます。
- 同一の FUN を同じタスク内で使用する場合、その FUN がいくつあっても 1 個と数えます。
- 同一の FUN を異なるタスクで使用する場合、タスクごとに、それぞれ 1 個と数えます。

保持属性なしの変数の数

保持属性なしの変数は、以下のように数えます。

● 保持属性なしの変数の計数対象

保持属性なしの変数の計数対象は、以下のとおりです。

- 保持属性なしのグローバル変数
- プログラムおよび FB インスタンス（ユーザ作成と命令の両方を含む）の、保持属性なしのローカル変数

● 保持属性なしの変数の数え方の注意

保持属性なしの変数を数える場合に、以下の点に注意してください。

- 配列は、要素数に関係なく 1 個と数えます。
- FB インスタンスは、1 個と数えます。ただし、FB インスタンスには、ユーザ作成と命令の両方を含みます。
- FB インスタンスを要素とする配列は、要素数に関係なく 1 個と数えます。ただし、その FB の中で使用している保持属性なしの変数の数は、配列の要素数分数えます。

保持属性ありの変数の数

保持属性ありの変数は、以下のように数えます。

● 保持属性ありの変数の計数対象

保持属性ありの変数の計数対象は、以下のとおりです。

- 保持属性ありのグローバル変数
- プログラムおよびFBインスタンス（ユーザ作成と命令の両方を含む）の、保持属性ありのローカル変数

● 保持属性ありの変数の数え方の注意

保持属性ありの変数を数える場合に、以下の点に注意してください。

- 配列は、要素数に関係なく1個と数えます。
- FBインスタンスを要素とする配列は、数に含めません。ただし、そのFBの中で使用している保持属性ありの変数の数は、配列の要素数分数えます。

データ型定義の数

データ型定義は、以下のように数えます。

● データ型定義の計数対象

データ型定義の計数対象は、以下のとおりです。

- ユーザ作成の構造体型定義
- ユーザ作成の共用体型定義
- ユーザ作成の列挙型定義

A-10 NY シリーズのバージョン関連情報

NY シリーズコントローラと Sysmac Studio とのバージョンの対応、およびユニットバージョンごとのサポート機能について説明します。

A-10-1　ユニットバージョンと Sysmac Studio のバージョンとの関係

NY シリーズコントローラのユニットバージョンと Sysmac Studio のバージョンには、対応関係があります。基本的に、対応するバージョンどうしの組み合わせでお使いください。

ユニットバージョンと Sysmac Studio の対応バージョン

NY シリーズコントローラのユニットバージョンと Sysmac Studio の対応バージョンの表を以下に示します。

NY シリーズコントローラのユニットバージョン	Sysmac Studio の対応バージョン
Ver.1.26	Ver.1.46
Ver.1.24	Ver.1.45
Ver.1.21	Ver.1.29
Ver.1.19	Ver.1.24
Ver.1.18	Ver.1.22
Ver.1.16	Ver.1.20
Ver.1.14	Ver.1.18
Ver.1.12	Ver.1.17

ユニットバージョンとSysmac Studioのバージョンが合っていない場合の仕様

NY シリーズコントローラのユニットバージョンと Sysmac Studio のバージョンが合っていない場合の仕様は、以下のとおりです。

● Sysmac Studio のバージョンが Ver.1.16 以前の場合

Ver.1.16 以前の Sysmac Studio では、NY シリーズコントローラを使用できません。

● Sysmac Studio のバージョンの方が古い場合

Sysmac Studio のバージョンの方が古い場合、使用できる機能は、Sysmac Studio のバージョンに対応するコントローラのユニットバージョンが搭載している機能になります。

例）コントローラのユニットバージョン：Ver.1.14
　　Sysmac Studio のバージョン：Ver.1.17
　　Syamac Studio Ver.1.17 に対応するコントローラのユニットバージョンは、Ver.1.12 です。したがって、使用できる機能は、コントローラ Ver.1.12 が搭載している機能になります。コントローラ Ver.1.14 以降で追加された機能は、使用できません。

● コントローラのユニットバージョンの方が古い場合

NY シリーズコントローラのユニットバージョンの方が古い場合は、Sysmac Studio のプロジェクトのデバイス選択画面で、接続しているコントローラのユニットバージョン以前を選択してください。使用できる機能は、接続しているコントローラのユニットバージョンが搭載している機能になります。

例）コントローラのユニットバージョン：Ver.1.12
　　Sysmac Studio のバージョン：Ver.1.18

Syamac Studio Ver.1.18 に対応するコントローラのユニットバージョンは、Ver.1.14 です。しかし、接続しているコントローラのユニットバージョンが Ver.1.12 なので、プロジェクトのデバイス選択画面の「バージョン」に、「Ver.1.12」を選択してください。
デバイス選択画面の「バージョン」に「Ver.1.12」を選択した場合、使用できる機能は、コントローラ Ver.1.12 が搭載している機能になります。コントローラ Ver.1.14 以降で追加された機能は、使用できません。



参考

プロジェクトのデバイスに設定するユニットバージョン

- Sysmac Studio では、ユニットバージョンの選択をデバイス選択画面で設定します。
- 設定できるユニットバージョンは、使用するコントローラのユニットバージョン以前です。たとえば、コントローラのユニットバージョンが Ver.1.14 の場合は、「1.12」または「1.14」を選択して設定してください。
- Sysmac Studioでは、プロジェクトのデバイスに設定したユニットバージョンを、コントローラのユニットバージョンとして扱います。たとえば、プロジェクトのデバイスに設定するユニットバージョンが Ver.1.12 の場合は、Sysmac Studio ではユニットバージョン Ver.1.12 までの機能を使用できます。
- プロジェクトのデバイスに設定したユニットバージョンが、転送先コントローラのユニットバージョン以前の場合に、Sysmac Studio からプロジェクトを転送できます。
- デバイス選択画面の詳細は、『Sysmac Studio Version 1 オペレーションマニュアル (SBCA-470)』を参照してください。

A-10-2 ユニットバージョンの更新によるサポート機能の追加と変更

NY シリーズコントローラのユニットバージョンの更新により、追加または変更されたサポート機能を紹介します。

● 機能仕様の追加と変更

追加または変更された機能仕様ごとに、コントローラのユニットバージョンと Sysmac Studio のバージョンを示します。

機能				追加／変更	ユニットバージョン	Sysmac Studio のバージョン	参照先
モーション制御機能	単軸	単軸制御補助	カムモニタ	追加	Ver.1.21	Ver.1.29	『NY シリーズ 産業用パネル型 PC ／産業用ボックス型 PC ユーザーズマニュアル モーション制御編 (SBCE-379)』

(注) コントローラのユニットバージョンの更新による、各機能モジュールの機能の追加、変更については、各機能モジュールのマニュアルを参照してください。

● 基本命令、およびモーション制御命令の追加と変更

コントローラのユニットバージョンの更新により、使用できる基本命令およびモーション制御命令が追加または変更されることがあります。

詳細は、🕮『NY シリーズ コマンドリファレンスマニュアル 基本編（SBCA-437）』、および 🕮『NY シリーズ コマンドリファレンス マニュアル モーション編（SBCE-380）』を参照してください。

● コントローライベントの追加と変更

コントローラのユニットバージョンの更新により、発生するイベントが追加または変更されることがあります。また、異常発生時の対処方法が変更されることもあります。

詳細は、🕮『NY シリーズ トラブルシューティングマニュアル（SBCA-438）』を参照してください。

● システム定義変数の追加と変更

コントローラのユニットバージョンの更新により、使用できるシステム定義変数が追加または変更されることがあります。

詳細は、🕮「A-3 システム定義変数一覧」（P.A-22）を参照してください。

I

索引

索引

A

_ <タスク名称> _Active（タスク実行中フラグ） 5-6, A-23, A-47
_AlarmFlag（ユーザ異常ステータス） 8-53, 8-57, A-24, A-49
AT（割付先）指定 6-58

B

_BackupBusy（バックアップ関連機能実行中フラグ） A-26, A-53
BOOL 6-31, 6-44
BOOTP サーバ接続異常 A-39, A-74
BYTE 6-31, 6-44

C

_Card1Access（SD メモリカードアクセス中フラグ） 8-11, A-24, A-50
_Card1BkupCmd（SD メモリカードバックアップ指示） A-25, A-51
_Card1BkupSta（SD メモリカードバックアップ ステータス） A-25, A-52
_Card1Capacity（SD メモリカードストレージ容量） 8-11, A-24, A-50
_Card1Deteriorated（SD メモリカード寿命警告フラグ） 8-11, A-24, A-50
_Card1Err（SD メモリカードエラーフラグ） 8-11, A-24, A-50
_Card1PowerFail（SD メモリカードアクセス中電源断 フラグ） 8-11, A-24, A-50
_Card1Protect(SD メモリカードライトプロテクトフラグ) 8-11, A-24, A-49
_Card1Ready（SD メモリカード使用可フラグ） 8-11, A-24, A-49
_Card1Used（SD メモリカードストレージ使用量） 8-11, A-24, A-51
_Card1VefySta（SD メモリカード照合ステータス） A-26, A-52
CIP 通信異常 A-38, A-71
CPU ユニット内部温度オーバーフラグ A-29, A-57
CPU ユニットへの書込プロテクト機能 8-23
CPU ユニット名称 8-24
CPU ユニット名称機能 8-24
_CurrentTime（システム時刻） 8-4, A-22, A-47
CX-Designer の変数テーブル登録方法 A-96

D

DATE 6-31, 6-44
DATE_ AND_ TIME 6-32, 6-44
_DeviceOutHoldCfg （デバイス出力保持設定） A-29, A-57
_DeviceOutHoldStatus（デバイス出力保持状態） A-29, A-57
DINT 6-31
DNS サーバ接続異常 A-41, A-76
DNS 設定異常 A-39, A-74
DWORD 6-31

E

_EC1_InDataInvalid（入力データ 1 無効） A-35
_EC_CommErrTbl（通信異常スレーブテーブル） A-33, A-66
_EC_CycleExceeded（EtherCAT 通信周期設定異常） A-33, A-66
_EC_DisableSlavTbl（無効設定スレーブテーブル） A-35, A-67
_EC_DisconnSlavTbl（離脱指令中スレーブテーブル） A-35, A-67
_EC_EntrySlavTbl（ネットワーク加入スレーブテーブル） A-35, A-66
_EC_ErrSta（内蔵 EtherCAT 異常） A-32, A-62
_EC_InData1Invalid（入力データ 1 無効） A-68
_EC_InDataInvalid（入力データ無効） A-35, A-68
_EC_IndataInvalidErr（プロセスデータ入力値無効異常） A-33, A-66
_EC_LanHwErr（通信コントローラ異常） A-32, A-63
_EC_LinkOffErr（リンクオフ異常） A-32, A-63
_EC_LinkStatus（リンクステータス） A-35, A-68
_EC_MacAdrErr（MAC アドレス異常） A-32, A-63
_EC_MBXSlavTbl（メッセージ通信可能スレーブテーブル） A-35, A-67
_EC_MsgErr（EtherCAT メッセージ異常） ... A-33, A-65
_EC_MstrErr（マスタ異常） A-32, A-62
_EC_NetCfgCmpErr（ネットワーク構成照合異常） A-32, A-64
_EC_NetCfgErr（ネットワーク構成情報異常） A-32, A-63
_EC_NetTopologyErr（ネットワーク構成異常） A-32, A-64
_EC_PDActive（プロセスデータ通信ステータス） A-35, A-67
_EC_PDCommErr（プロセスデータ通信異常） A-33, A-64
_EC_PDSendErr（プロセスデータ送信異常） A-33, A-64
_EC_PDSlavTbl(プロセスデータ通信中スレーブテーブル) A-35, A-67
_EC_PDTimeoutErr（プロセスデータ受信タイムアウト異常） A-33, A-64
_EC_PktMonStop（パケットモニタ停止） A-35, A-68
_EC_PktSaving（パケットデータファイル保存中） A-35, A-68
_EC_PortErr（通信ポート異常） A-32, A-62
_EC_RegSlavTbl（登録スレーブテーブル） A-35, A-66
_EC_SlavAdrDupErr（スレーブノードアドレス重複異常） A-33, A-65

_EC_SlavAppErr（スレーブアプリケーション異常）
.....A-33, A-65
_EC_SlavEmergErr（エマージェンシー・メッセージ検知）
.....A-33, A-65
_EC_SlavErr（スレーブ異常）.....A-32, A-62
_EC_SlavErrTbl（スレーブ異常テーブル）.....A-32, A-63
_EC_SlavInitErr（スレーブ初期化異常）.....A-33, A-65
_EC_StatisticsLogBusy（診断統計情報ログ実行状態
フラグ）.....A-36, A-69
_EC_StatisticsLogErr（診断統計情報ログ異常終了フラグ）
.....A-36
_EC_StatisticsLogCycleSec（診断統計情報ログ保存周期）
.....A-36, A-69
_EC_StatisticsLogEnable（診断統計情報ログ許可）A-36
_EC_StatisticsLogErr（診断統計情報ログ異常終了フラグ）
.....A-69
_EIP1_BootpErr（通信ポート 1BOOTP サーバ接続異常）
.....A-39, A-74
_EIP1_EtnCfgErr（通信ポート 1Ethernet 基本設定異常）
.....A-39, A-72
_EIP1_EtnOnlineSta（通信ポート 1 オンライン）
.....A-42, A-77
_EIP1_EtnOnlineSta（内部通信ポート 1 オンライン）
.....A-77
_EIP1_IPAdrCfgErr（通信ポート 1IP アドレス設定異常）
.....A-39, A-73
_EIP1_IPAdrDupErr（通信ポート 1IP アドレス重複異常）
.....A-39, A-73
_EIP1_IPAdrDupErr（内部通信ポート 1IP アドレス重複
異常）.....A-39
_EIP1_LanHwErr（通信ポート 1 通信コントローラ異常）
.....A-38, A-72
_EIP1_MacAdrErr（通信ポート 1MAC アドレス異常）
.....A-38, A-72
_EIP1_PortErr（通信ポート 1 異常）.....A-37, A-70
_EIP_BootpErr（BOOTP サーバ接続異常）...A-39, A-74
_EIP_CipErr（CIP 通信異常）.....A-38, A-71
_EIP_DNSCfgErr（DNS 設定異常）.....A-39, A-74
_EIP_DNSSrvErr（DNS サーバ接続異常）.....A-41, A-76
_EIP_ErrSta（内蔵 EtherNet/IP 異常）.....A-37, A-70
_EIP_EstbTargetSta（正常ターゲットノード情報）
.....A-42, A-78
_EIP_EtnCfgErr（Ethernet 基本設定異常）....A-39, A-72
_EIP_EtnOnlineSta（オンライン）.....A-42, A-76
_EIP_IdentityErr（Identity 情報不正）.....A-40, A-75
_EIPln1_EtnOnlineSta（内部通信ポート 1 オンライン）
.....A-42
_EIPln1_IPAdrCfgErr（内部通信ポート 1IP アドレス設定
異常）.....A-39, A-73
_EIPln1_IPAdrDupErr（内部通信ポート 1IP アドレス重複
異常）.....A-74
_EIPln1_PortErr（内部通信ポート 1 異常）....A-38, A-71
_EIP_IPAdrCfgErr（IP アドレス設定異常）....A-39, A-73
_EIP_IPAdrDupErr（IP アドレス重複異常）...A-39, A-73
_EIP_IPRTblErr（IP ルートテーブル異常）.....A-40, A-74
_EIP_LanHwErr（通信コントローラ異常）.....A-38, A-72
_EIP_MacAdrErr（MAC アドレス異常）.....A-38, A-71
_EIP_MultiSwONErr（複数スイッチ同時 ON 異常）
.....A-40, A-76
_EIP_NTPResult（NTP の動作情報）.....A-43
_EIP_NTPResult.ExecNormal（NTP の動作結果）
.....A-43, A-79
_EIP_NTPResult.ExecTime（NTP の最終動作時刻）
.....A-43, A-79
_EIP_NTPSrvErr（NTP サーバ接続異常）.....A-40, A-76
_EIP_PortErr（通信ポート異常）.....A-37, A-70
_EIP_RegTargetSta（登録ターゲットノード情報）
.....A-42, A-77
_EIP_TagAdrErr（タグ解決異常）.....A-40, A-75
_EIP_TargetNodeErr（ターゲットノード異常情報）
.....A-43, A-78
_EIP_TargetPLCErr（ターゲット PLC 異常情報）
.....A-43, A-78
_EIP_TargetPLCModeSta（ターゲット PLC 動作モード）
.....A-43, A-78
_EIP_TcpAppCfgErr（TCP/IP 設定異常）.....A-40
_EIP_TcpAppCfgErr（TCP アプリケーション設定異常）
.....A-76
_EIP_TcpAppErr（TCP アプリ通信異常）.....A-38, A-71
_EIP_TDLinkAllRunSta（全タグデータリンク通信中）
.....A-42, A-77
_EIP_TDLinkCfgErr（タグデータリンク設定異常）
.....A-40, A-75
_EIP_TDLinkErr（タグデータリンク通信異常）
.....A-40, A-75
_EIP_TDLinkOpnErr（タグデータリンクコネクション
確立失敗）.....A-40, A-75
_EIP_TDLinkRunSta（タグデータリンク通信中）
.....A-42, A-77
_EIP_TDLinkStartCmd（タグデータリンク通信開始
スイッチ）.....A-44, A-79
_EIP_TDLinkStopCmd（タグデータリンク通信停止
スイッチ）.....A-44, A-79
EN 6-19
ENO 6-20
_ErrSta（コントローラ異常ステータス）5-50, A-24, A-49
EtherCAT スレーブとのアクセス 2-13
EtherCAT スレーブや軸の有効／無効切替 A-99
EtherCAT 通信周期設定異常 A-33, A-66
EtherCAT ネットワーク構成 1-5
EtherCAT マスタ機能モジュールの初期設定 4-14
EtherCAT メッセージ異常 A-33, A-65
EtherNet/IP 機能モジュールの初期設定 4-15
Ethernet 基本設定異常 A-39, A-72
_ <タスク名称> _ExceedCount（タスク周期超過回数）
.....5-6, 5-50, A-23, A-48
_ <タスク名称> _Exceeded（タスク周期超過フラグ）
.....5-6, 5-50, 5-54, A-23, A-48
_ <タスク名称> _ExecCount（タスク実行回数）
.....5-6, 5-54, A-23, A-48

F

FAN 回転数低下フラグ A-29, A-57
FB 名または命令名 6-10
FTP クライアント通信命令 8-10
FTP サーバ機能 8-10
FUN 定義名または命令名 6-17

H

_HardwareRevision（ハード A-56

I

I/O ポート 3-5
I/O ポート名 3-6
I/O リフレッシュ周期オーバー 5-51
I/O を制御するタスク設定 4-8
Identity 情報不正 A-40, A-75
INT 6-31
IP アドレス設定異常 A-39, A-73
IP アドレス重複異常 A-39, A-73
IP ルートテーブル異常 A-40, A-74

L

_ ＜タスク名称＞ _LastExecTime（前回タスク実行時間）
..... 5-6, A-23, A-47
LD 6-84
LINT 6-31
LREAL 6-31
LWORD 6-31

M

MAC アドレス異常 A-32, A-38, A-63, A-71
_ ＜タスク名称＞ _MaxExecTime（タスク実行時間最大値）
..... 5-6, A-23, A-47
_MC_AX（軸変数） A-31, A-61
_MC_AX_ErrSta（軸異常ステータス） A-31, A-60
_MC_COM（共通変数） A-31, A-61
_MC_ComErrSta（共通異常ステータス） A-31, A-60
_MC_ErrSta（モーション制御機能モジュールの異常
ステータス） A-31, A-60
_MC_GRP（軸グループ変数） A-31, A-61
_MC_GRP_ErrSta（軸グループ異常ステータス）
..... A-31, A-60
MC 試運転 7-9
_ ＜タスク名称＞ _MinExecTime（タスク実行時間最小値）
..... 5-6, A-23, A-47

N

NTP サーバ接続異常 A-40, A-76
NTP の最終動作時刻 A-43, A-79
NTP の動作結果 A-43, A-79
NTP の動作情報 A-43

O

_OSErrorState（OS 異常状態フラグ） A-27, A-55
_OSHalted（OS 停止中フラグ） A-27, A-55
_OSRunning（OS 稼働中フラグ） A-27, A-54
OS 異常状態フラグ A-27, A-55
OS 稼働中フラグ A-27, A-54
OS 停止中フラグ A-27, A-55

P

P_CY（キャリーフラグ） 6-8, 6-130, A-28, A-55
P_First_Run（プログラム起動時 1 周期 ON フラグ）
..... A-28, A-56
P_First_Run（プログラム起動時周期 ON フラグ） 6-8
P_First_RunMode（運転開始時 1 周期 ON フラグ）
..... 6-8, A-28, A-55
_PLC_ErrSta（PLC 機能モジュール異常ステータス）
..... A-30, A-59
_PLC_TraceSta.IsComplete（トレース完了時 ON フラグ）
..... 8-40, A-30, A-58
_PLC_TraceSta.IsStart（トレース実行中 ON フラグ）
..... 8-40, A-30, A-58
_PLC_TraceSta.IsTrigger（トレーストリガモニタフラグ）
..... 8-40, A-30, A-58
_PLC_TraceSta.ParamErr（トレースパラメータ異常
フラグ） 8-40, A-30, A-58
PLC 機能モジュール 4-4
PLC 機能モジュール異常ステータス A-30, A-59
P_Off（常時 OFF フラグ） A-28, A-55
P_On（常時 ON フラグ） A-28, A-55
POU
　ファンクション（FUN） 6-6
　ファンクションブロック（FB） 6-6
　プログラム 6-6
POU（プログラム構成単位 ） 6-5
POU の制限事項 6-25
POU 名の変更や POU の新規作成などによる保持変数の
値 6-66
_PowerOnCount（電源断発生回数） A-27, A-54
_PowerOnHour（通電時間） A-27, A-53
P_PRGER（命令エラー発生フラグ）
..... 6-8, 6-132, 6-135, A-28, A-56

R

REAL 6-31, 6-44
_RequestShutdown（シャットダウン要求フラグ）
..... A-27, A-54
_RetainFail（電断保持失敗フラグ） A-27, A-54

S

SD メモリカード
　機能一覧 8-5
SD メモリカードアクセス中電源断フラグ
..... 8-11, A-24, A-50
SD メモリカードアクセス中フラグ 8-11, A-24, A-50
SD メモリカードエラーフラグ 8-11, A-24, A-50
SD メモリカード寿命警告フラグ 8-11, A-24, A-50
SD メモリカード使用可フラグ 8-11, A-24, A-49
SD メモリカード照合ステータス A-26, A-52
SD メモリカードストレージ使用量 8-11, A-24, A-51
SD メモリカードストレージ容量 8-11, A-24, A-50
SD メモリカードバックアップ機能 9-10
SD メモリカードバックアップ指示 A-25, A-51
SD メモリカードバックアップステータス A-25, A-52
SD メモリカードへのバックアップ禁止機能 9-19

SD メモリカード
　命令 8-9
SD メモリカードライトプロテクトフラグ
....8-11, A-24, A-49
_SelfTest_HighTemperature（CPU ユニット内部温度オーバーフラグ） A-29, A-57
_SelfTest_LowBattery（バッテリ電圧低下フラグ）
....A-29, A-57
_SelfTest_LowFanRevolution（FAN 回転数低下フラグ）
....A-29, A-57
_SelfTest_UPSSigna（UPS 信号検知フラグ） A-27
_SelfTest_UPSSignal（UPS 信号検知フラグ） A-54
SINT 6-31, 6-44
STRING 6-32, 6-44
ST 言語 6-90
　CASE 6-103
　EXIT 6-114
　FB 呼出 6-115
　FUN 呼出 6-118
　FOR 6-106
　IF（一つの条件） 6-99
　IF（複数の条件） 6-101
　REPEAT 6-112
　RETURN 6-98
　WHILE 6-110
　演算子 6-94
　各演算子のオペランドに使用可能なデータ型 6-96
　代入 6-97
　文のキーワード 6-93
Sysmac Studio 変数／メモリのバックアップ機能 ..9-25
Sysmac Studio コントローラバックアップ機能 9-20
Sysmac Studio バックアップファイルのインポート / エクスポート機能 9-24

T

TCP/IP 設定異常 A-40
TCP アプリケーション設定異常 A-76
TCP アプリ通信異常 A-38, A-71
TIME 6-31, 6-44
TIME_ OF_DAY 6-31, 6-44

U

UDINT 6-31
UINT 6-31
ULINT 6-31
_UnitVersion（ユニットバージョン） A-28, A-56
UPS 信号検知フラグ A-27, A-54
USINT 6-31

W

WORD 6-31

あ

アルゴリズム 6-10, 6-17
暗黙のキャスト 6-121

い

異常ステータス A-45
イベントコード 8-47
イベント重要度 8-48
イベント重要度変更機能 8-58
イベント設定テーブル 8-54
イベントタスク 5-14
　ActEventTask 命令 5-15
　設定項目 5-39
　変数の条件式一致 5-17
イベント発生源 8-46
イベントログ機能 8-45
イベントログ設定 4-5
インスタンス 6-13
インスタンス名 6-10
インライン ST 6-89

う

運転開始時 1 周期 ON フラグ 6-8, A-28, A-55
運転モード 2-25
運転モード書替 4-6

え

エッジ 6-63
エマージェンシー・メッセージ検知 A-33, A-65

お

オフラインデバッグ 7-3
オンライン A-42, A-76
オンラインエディット 8-33
オンライン接続が可能な構成 10-4
オンラインできない構成 10-5

か

外部変数 6-19
各状態における動作 2-24
仮想 SD メモリカード
　ファイルアクセスの排他制御 8-12
仮想 SD メモリカード設定 4-5
カテゴリ 8-47
可変長配列指定 6-54
カムデータ変数 2-11
完全限定名 6-138

き

機能モジュール
　EtherCAT マスタ機能モジュール 2-4
　EtherNet/IP 機能モジュール 2-4
　PLC 機能モジュール 2-4
　モーション制御機能モジュール 2-4
基本構成 1-5
基本データ型 6-30

キャリーフラグ 6-8, 6-130, A-28, A-55
強制値リフレッシュ 8-27
強制値をプログラム上で使用する場合の留意事項 8-30
共通異常ステータス A-31, A-60
共通変数 A-31, A-61
共用体型 6-45

く

グローバル変数 6-28

け

現在値変更 8-31
現在値変更により、外部への出力状態を変更する場合の
留意事項 8-32

こ

コイル 6-85
更新タスク 5-40
構造体のメンバのオフセット指定 6-42
構造体型 6-40
構文エラー 6-134
コンスタント 6-62
コンディションフラグ 6-130
コントローラ異常ステータス 5-50, A-24, A-49
コントローライベント 8-46, 8-51
コントローラ設定 4-4
コントローラ内のデータ 2-21
コントローラの状態 2-22

さ

サポートソフト 1-5, 1-6
参照タスク 5-40

し

シーケンス制御とモーション制御 のしくみ 2-16
シーケンス制御とモーション制御の同期 2-20
シーケンス制御のしくみ 2-18
軸 3-11
軸異常ステータス A-31, A-60
軸および軸変数の作成と使用方法 3-13
軸グループ異常ステータス A-31, A-60
軸グループ変数 3-11, A-31, A-61
軸変数 3-11, A-31, A-61
時刻型 6-31, 6-44
システムサービス 5-25
システム時刻 8-4, A-22, A-47
システム定義変数 2-12, 6-27
システム定義変数一覧 A-22
システム定義変数の個別仕様 A-46
持続時間型 6-31, 6-44
実数型 6-31, 6-44
シミュレーション機能 7-3
シミュレーション速度 7-7
シミュレーションプログラム 7-6
シミュレータの起動と停止 7-3
シャットダウン処理待ち時間設定 4-5
シャットダウン要求フラグ A-27, A-54
周期／実行条件 4-7
出力変数 6-19
準ユーザ定義変数 2-11, 6-27
仕様
　機能仕様 A-6
　性能仕様 A-3
照合 9-3
常時 OFF フラグ A-28, A-55
常時 ON フラグ A-28, A-55
仕様の概要 1-7
初期設定
　EtherCAT マスタ機能モジュールの初期設定 4-14
　EtherNet/IP 機能モジュールの初期設定 4-15
　モーション制御機能モジュールの初期設定 4-12
初期値 6-60
シリアル ID 機能 8-26
診断統計情報ログ異常終了フラグ A-36
診断統計情報ログ実行状態フラグ A-36
診断統計情報ログ異常終了フラグ A-69
診断統計情報ログ許可 A-36
診断統計情報ログ実行状態フラグ A-69
診断統計情報ログ保存周期 A-36, A-69

す

スレーブアプリケーション異常 A-33, A-65
スレーブ異常 A-32, A-62
スレーブ異常テーブル A-32, A-63
スレーブ初期化異常 A-33, A-65
スレーブノードアドレス重複異常 A-33, A-65

せ

正常ターゲットノード情報 A-42, A-78
整数型 6-31, 6-44
セキュリティ 設定 4-6
接続線 6-85
接点 6-85
前回タスク実行時間 5-6, A-23, A-47
全タグデータリンク通信中 A-42, A-77

そ

操作権限 8-21
操作権限の認証機能 8-21
ソフトウェア構成 2-4
ソフトウェアの動作 2-5

た

ターゲット PLC 異常情報 A-43, A-78
ターゲット PLC 動作モード A-43, A-78
ターゲットノード異常情報 A-43, A-78
タグ解決異常 A-40, A-75
タグデータリンクコネクション確立失敗 A-75, A-40
タグデータリンクサービス 5-25

タグデータリンク設定異常 A-40, A-75
タグデータリンク通信異常 A-40, A-75
タグデータリンク通信開始スイッチ A-44, A-79
タグデータリンク通信中 A-42, A-77
タグデータリンク通信停止スイッチ A-44, A-79
タスク
　I/O リフレッシュへの割付 5-31
　NY シリーズの仕様 5-8
　実行優先度 5-9
　プログラムの割付 5-37
タスク間での変数の値の同時性を確保する 5-40
タスク間排他制御設定 5-47
タスク間排他制御命令 5-43
タスク関連のシステム定義変数 5-6
タスク関連の命令 5-6
タスク実行回数 5-6, 5-54, A-23, A-48
タスク実行時間 5-55
タスク実行時間最小値 5-6, A-23, A-47
タスク実行時間最大値 5-6, A-23, A-47
タスク実行時間の確認 5-56
タスク実行状態モニタ 4-10
タスク実行タイムアウト 5-51
タスク実行タイムアウト時間 4-8
タスク実行中フラグ 5-6, A-23, A-47
タスク周期オーバー 5-50
タスク周期オーバ検知 4-8
タスク周期超過回数 5-6, 5-50, A-23, A-48
タスク周期超過フラグ 5-6, 5-50, 5-54, A-23, A-48
タスク種類 5-13
タスク設計の例 5-57
タスク設定 4-7
タスクタイプ 4-7
タスクに割り付けることができる POU 5-37
タスクの実行時間のモニタ 5-53
タスクの実行状態のモニタ 5-52
タスク名 4-7, 4-8
単純名 6-138

つ

通信異常スレーブテーブル A-33, A-66
通信コントローラ異常 A-32, A-38, A-63, A-72
通信ポート 1BOOTP サーバ接続異常 A-39, A-74
通信ポート 1Ethernet 基本設定異常 A-39, A-72
通信ポート 1IP アドレス設定異常 A-39, A-73
通信ポート 1IP アドレス重複異常 A-39, A-73
通信ポート 1MAC アドレス異常 A-38, A-72
通信ポート 1 異常 A-37, A-70
通信ポート 1 オンライン A-42, A-77
通信ポート 1 通信コントローラ異常 A-38, A-72
通信ポート異常 A-32, A-37, A-62, A-70
通電時間 A-27, A-53

て

データ型 6-30
データ型の変換 6-39
データトレース 8-35
データトレースの仕様 8-36
データトレースの動作 8-38
データの保持／非保持属性一覧 A-80
データプロテクト機能 8-19
定周期タスク（実行優先度 16、17、18）の設定項目 5-39
定周期タスク（実行優先度 16、17、18） 5-14
定数 6-79
　時刻型 6-81
　持続時間型 6-80
　実数型 6-80
　整数 6-80
　日付型 6-81
　日付時刻型 6-81
　ビット列型 6-79
　ブール型 6-79
　文字列型 6-82
　列挙型 6-83
デバイス出力保持設定 A-29, A-57
デバイス出力保持状態 A-29, A-57
デバイス出力保持設定 6-65
デバイス変数 2-11, 3-6
デバッグプログラム 7-6
電源断発生回数 A-27, A-54
電断保持失敗フラグ A-27, A-54

と

動作オプション 6-129, 6-130
動作設定 4-5
動作設定タブ 4-4
動作モード 2-25
動作モード切替時の変数の値の保持 2-25
動作モード切替 2-26
動作モードの確認 2-26
登録スレーブテーブル A-35, A-66
登録ターゲットノード情報 A-42, A-77
時計機能 8-2
トリガトレース 8-35
トレース完了時 ON フラグ 8-40, A-30, A-58
トレース実行中 ON フラグ 8-40, A-30, A-58
トレーストリガモニタフラグ 8-40, A-30, A-58
トレースパラメータ異常フラグ 8-40, A-30, A-58

な

内蔵 EtherCAT 異常 A-32, A-62
内蔵 EtherNet/IP 異常 A-37, A-70
内部通信ポート 1IP アドレス設定異常 A-39, A-73
内部通信ポート 1IP アドレス重複異常 A-39, A-74
内部通信ポート 1 異常 A-38, A-71
内部通信ポート 1 オンライン A-42, A-77
内部変数 6-19
名前空間 6-137
名前空間の使用宣言 6-139

に

入出力応答時間 5-58

入出力変数 6-19
入力データ 1 無効 A-35, A-68
入力データ無効 A-35, A-68
入力変数 6-19

ね

ネスティングの階層 6-26
ネットワーク加入スレーブテーブル A-35, A-66
ネットワーク公開 6-63
ネットワーク構成 1-6
ネットワーク構成異常 A-32, A-64
ネットワーク構成照合異常 A-32, A-64
ネットワーク構成情報異常 A-32, A-63

は

ハードウェアリビジョン A-56
配列指定（ARRAY OF）.....6-50
パケットデータファイル保存中 A-35, A-68
パケットモニタ停止 A-35, A-68
派生データ型 6-40
バックアップ 9-3
バックアップ関連機能実行中フラグ A-26, A-53
バッテリ電圧低下フラグ A-29, A-57
パラメータ 6-10, 6-18
範囲指定（(..)）.....6-56

ひ

日付型 6-31, 6-44
日付時刻型 6-32, 6-44
ビット列型 6-31, 6-44
ビット列型、実数型、文字列型のデータフォーマット 6-35
微分モニタ 8-40

ふ

ブール型 6-31, 6-44
ファンクション（FUN）
　EN、ENO の使用／省略の可否 6-23
　ST 言語での表現 6-18
　起動条件 6-22
　構成 6-17
　詳細 6-17
　内部変数の仕様 6-19
　パラメータ省略時の動作 6-23
　パラメータの異常時の動作 6-24
ファンクションブロック
　ST 言語での呼び出しの表現 6-11
　インスタンス 6-13
　起動条件 6-15
　内部変数の仕様 6-12
　変数の外部からの参照 6-16
ファンクションブロック（FB）
　EN、ENO の使用／省略の可否 6-23
　パラメータ省略時の動作 6-23
　パラメータの異常時の動作 6-24
ファンクションブロック（FB）構成 6-9
ファンクションブロック（FB）詳細 6-9
複数スイッチ同時 ON 異常 A-40, A-76
プライマリ定周期タスク 5-13
　設定項目 5-38
プログラム 6-8
プログラム起動時 1 周期 ON フラグ 6-8, A-28, A-56
プログラム言語 6-84
プログラム実行順番 4-9
プログラムの運転開始時初期状態 5-37
プログラムの実行順 5-37
プログラムの初期状態 4-9
プログラムの割付設定 4-9
プログラムモード 2-25
プロジェクトファイル全体のプロテクト機能 8-18
プロセスデータ受信タイムアウト異常 A-33, A-64
プロセスデータ送信異常 A-33, A-64
プロセスデータ通信異常 A-33, A-64
プロセスデータ通信ステータス A-35, A-67
プロセスデータ通信中スレーブテーブル A-35, A-67
プロセスデータ入力値無効異常 A-33, A-66

へ

変数
　概要 6-27
　種類 6-27
変数アクセス時間 4-8
変数アクセス時間設定 5-48
変数と I/O のアクセス 2-13
変数の種類
　システム定義変数 2-10
　準ユーザ定義変数 2-10
　ユーザ定義変数 2-10
変数の属性 6-29
変数のタスク間排他制御設定 4-10
変数のメモリ確保の方法 A-83
変数名 6-58
変数名などのプログラム関連の名称の制限 6-77

ほ

保持 6-59
保持条件 6-66
保持変数のデータ型を変更したときの変数の値 6-69
母線 6-84

ま

マスタ異常 A-32, A-62
マスタコントロール 6-147

む

無効設定スレーブテーブル A-35, A-67

め

命令 6-128
命令エラー 6-131

命令エラー発生フラグ ...6-8, 6-132, 6-135, A-28, A-56
メッセージ通信可能スレーブテーブル A-35, A-67
メモリオールクリア 8-2

も

モーション制御機能モジュールの異常ステータス
.....A-31, A-60
モーション制御機能モジュールの初期設定 4-12
モーション制御システム変数 2-12
モーション制御周期オーバー 5-51
モーション制御のしくみ 2-19
文字列型 6-32, 6-44
戻り値 6-20

ゆ

ユーザ異常ステータス 8-53, 8-57, A-24, A-49
ユーザイベント 8-46, 8-52
ユーザ定義変数 6-27
　種類 6-28
ユーザプログラム作成上の留意事項 6-146
ユーザプログラム実行用 ID 8-14
ユーザプログラム復元情報無し転送機能 8-17
ユニットバージョン A-28, A-56

ら

ライブラリ 6-142
ラダー図言語 6-84
ラダー図での FB、FUN の接続 6-86

り

リストア 9-3
離脱指令中スレーブテーブル A-35, A-67
リテラル 6-79
リンクオフ異常 A-32, A-63
リンクステータス A-35, A-68

れ

列挙型（ENUM）.....6-48
連続トレース 8-35

ろ

ローカル変数 6-28
　外部変数 6-28
　出力変数 6-28
　内部変数 6-28
　入出力変数 6-28
　入力変数 6-28
ローカル変数テーブル 6-10, 6-18

オムロン株式会社 インダストリアルオートメーションビジネスカンパニー

製品に関するお問い合わせ先

お客様相談室

フリー通話 **0120-919-066**（クイック　オムロン）

携帯電話・IP電話などではご利用いただけませんので、右記の電話番号へおかけください。 **055-982-5015**（通話料がかかります）

受付時間：9:00～19:00（12/31～1/3を除く）

オムロンFAクイックチャット
www.fa.omron.co.jp/contact/tech/chat/
技術相談員にチャットでお問い合わせいただけます。（I-Webメンバーズ限定）
受付時間：平日9:00～12:00 / 13:00～17:00（土日祝日・年末年始・当社休業日を除く）
※受付時間、営業日は変更の可能性がございます。最新情報はリンク先をご確認ください。

その他のお問い合わせ：納期・価格・サンプル・仕様書は貴社のお取引先、または貴社担当オムロン販売員にご相談ください。オムロン制御機器販売店やオムロン販売拠点は、Webページでご案内しています。

オムロン制御機器の最新情報をご覧いただけます。緊急時のご購入にもご利用ください。 **www.fa.omron.co.jp**

●本誌に記載のない条件や環境での使用、および原子力制御・鉄道・航空・車両・燃焼装置・医療機器・娯楽機械・安全機器、その他人命や財産に大きな影響が予測されるなど、特に安全性が要求される用途に使用される際には、当社の意図した特別な商品用途の場合や特別の合意がある場合を除き、当社は当社商品に対して一切保証をいたしません。
●本製品の内、外国為替及び外国貿易法に定める輸出許可、承認対象貨物（又は技術）に該当するものを輸出（又は非居住者に提供）する場合は同法に基づく輸出許可、承認（又は役務取引許可）が必要です。

オムロン商品のご用命は

Man. No. SBCA-436J　2023年12月現在

©OMRON Corporation 2016-2023 All Rights Reserved.
お断りなく仕様などを変更することがありますのでご了承ください