

マシンオートメーションコントローラ NX1P

活用ガイド はじめての NX1P プログラミング入門編

形 NX1P2-□□□□

形 SYSMAC-SE20□□

Practices
Guide



おことわり

- (1) 本ガイド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) 本ガイドの内容に関しては、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) 本ガイドの内容に関しては、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・著作権

- 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。
- Windows は、米国 Microsoft Corporation の米国およびその他の国における登録商標または商標です。
- 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。
- EtherNet/IP™は ODVA の商標です。
- Celeron、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。
- スクリーンショットはマイクロソフトの許可を得て使用しています。

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。

はじめに

このたびは、NXシリーズ NX1P2 CPU ユニットおよびSysmac Studio をお買い上げいただきまして、ありがとうございます。

活用ガイド はじめてのNX1P プログラミング入門編(以降「本ガイド」と省略します。)は、NX1P2 CPU ユニットの初めて使用する方向けに、NX1P のプログラミングがこれまでのコントローラと比べてどのような違いがあるのか、およびNX1P2 と Sysmac Studio を使ったプログラミングの方法について説明しています。本ガイドに沿って操作することで、NX1P2 CPU ユニットと Sysmac Studio の使い方の基礎を効率的に習得することができます。なお、本ガイドには安全に関する注意事項を含め、使用する際の詳細内容については記載していません。必ず、本ガイドで使用する各機器のマニュアルや取扱説明書を入手し、「安全上のご注意」「安全上の要点」「使用上の注意」などの安全に関する注意事項を含め、使用する際の詳細内容を確認したうえで使用してください。

モーション制御を行う場合の手順と操作については、『NJ/NX シリーズ スタートアップガイド モーション制御編 (SBCE-372)』を参照してください。

対象となる読者の方々

本ガイドは、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

対象となる製品

本ガイドは、以下の製品を対象にしています。

- マシンオートメーションコントローラ NX1P2 CPU ユニット
- オートメーションソフトウェア Sysmac Studio

アイコンについて

本ガイドで使用するアイコンには、次のような意味があります。



使用上の注意

製品が動作不能、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、使用するうえで参考となる内容について説明しています。

ご承諾事項

マシンオートメーションコントローラ NX1P2 CPU ユニット

「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まれます。
- ・「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・身体に危険が及ぶ用途)
- (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。

オートメーションソフトウェア

Sysmac Studio

●保証内容

(1) 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。)

(2) 保証範囲

- a) 本ソフトウェアの使用許諾を受けたお客様が、上記保証期間中にコンピュータ・プログラムの瑕疵(マニュアルとの重要な不一致)を発見し、当社に返却した場合は、当社は瑕疵(当該マニュアルとの重要な不一致)のないコンピュータ・プログラムを記録した媒体と交換いたします。もしくは、当社の選択により、当社ホームページより瑕疵(当該マニュアルとの重要な不一致)のないコンピュータ・プログラムをダウンロードしていただく方法により提供いたします。また、当社の責任によるコンピュータ・プログラムの記録媒体の不良を発見し、当社に返却した場合、当社は、無償で、良品の媒体に記録したコンピュータ・プログラムと交換いたします。
- b) 万一、当社がコンピュータ・プログラムの瑕疵(マニュアルとの重要な不一致)を除去できないと判断した場合は、お客様が本ソフトウェア購入代金として支払った金額をお返しいたします。

●責任の制限

- (1) 前条に定める交換または購入代金の返金は、本ソフトウェアの保証責任のすべてを定めるものであり、当社は本ソフトウェアの瑕疵により発生した、お客様の直接的、間接的あるいは波及効果による損害等いかなる損害に対しても一切の責任を負いません。
- (2) 当社は、本ソフトウェアを当社以外の第三者が変更、改造することにより生じた瑕疵につきましては、一切責任を負いません。
- (3) 当社は、本ソフトウェアに基づき、当社以外の第三者が開発したソフトウェアおよびそれにより生じた結果について一切の責任を負いません。

●本ソフトウェアの用途

本ソフトウェアを本マニュアルに記載の用途以外の用途で使用しないでください。

●仕様の変更

本ソフトウェアの仕様および付属品は改善またはその他の事由により、必要に応じて、変更される場合があります。

●適用範囲

以上の内容は、日本国内での取引および使用を前提としております。日本国外での取引および使用に関しては、当社営業担当者までご相談ください。

注意事項

- 実際のシステム構築に際しては、システムを構成する各機器や装置の仕様をご確認のうえ、定格や性能に対し余裕を持った使い方をしてください。万一故障があっても危険を最小にする安全回路などの安全対策を講じてください。
- システムを安全に使用していただくため、システムを構成する各機器や装置のマニュアルや取扱説明書などを入手し、「安全上のご注意」「安全上の要点」「使用上の注意」など安全に関する注意事項を含め、内容を確認のうえ使用してください。
- お客様の機械や装置が適合すべき規格や法規または規制は、お客様自身でご確認ください。

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。

関連マニュアル

関連するマニュアルは、下表のとおりです。併せてご覧ください。

マニュアル名称	Man.No.	形式	用途	内容
NX シリーズ 形NX1P2 CPU ユニット ユーザーズマニュアル ハード編	SBCA-448	形 NX1P2- □□□□	NX シリーズ 形 NX1P2 CPU ユニッ トの概要／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。	形NX1P2 CPU ユニットのシステム全体概 要、およびCPU ユニットに関して、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
NX シリーズ 形NX1P2 CPU ユニット ユーザーズマニュアル 内蔵I/O、オプションボ ード 機能編	SBCA-449	形 NX1P2- □□□□	NX シリーズ 形 NX1P2 CPU ユニッ トだけに搭載する機 能の詳細につい て、および、NJ/NX シリーズの機能の 概要を知りたいと き。	形NX1P2 CPU ユニットの機能のうち以下の 内容を説明します。 ・ 内蔵I/O ・ シリアル通信用オプションボード ・ アナログ入出力用オプションボード あわせて、NJ/NX シリーズCPU ユニットの 以下機能の概要を説明します。 ・ モーション制御機能 ・ EtherNet/IP 通信機能 ・ EtherCAT 通信機能
NJ/NX シリーズ CPU ユニット ユーザーズマニュアル ソフトウェア編	SBCA-467	形NX701- □□□□ 形NX102- □□□□ 形NX1P2- □□□□ 形NJ501- □□□□ 形NJ301- □□□□ 形NJ101- □□□□	NJ/NX シリーズ CPU ユニットのブ 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。 おもにソフトウェア に関する情報。	NJ シリーズ CPU ユニットに関して、以下 の内容を説明します。 ・ CPU ユニットの動作 ・ CPU ユニットの機能 ・ 初期設定 ・ IEC 61131-3 ベースの言語仕様とプログラミ ングユーザーズマニュアル ハードウェア編 (SBCA-448)と併せて使用してください。
NJ/NX シリーズ コマンドリファレンス マニュアル 基本編	SBCA-468	形NX701- □□□□ 形NX102- □□□□ 形NX1P2- □□□□ 形NJ501- □□□□ 形NJ301- □□□□ 形NJ101- □□□□	NJ/NX シリーズの 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。 プログラミング時に、ユーザーズマニュアル ハードウェア編(SBCA-466)／ソフトウェア 編(SBCA-467)と併せて使用してください。
NJ/NX シリーズ CPU ユニット ユーザーズマニュアル モーション制御編	SBCE-433	形NX701- □□□□ 形NX102- □□□□ 形NX1P2- □□□□ 形NJ501- □□□□ 形NJ301- □□□□ 形 NJ101- □□□□	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CPU ユニットの設 定や動作、プログラミングの考え方について 説明します。 プログラミング時に、ユーザーズマニュアル ハードウェア編(SBCA-466)／ソフトウェア 編(SBCA-467)と併せて使用してください。

NJ/NX 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-434	形NX701- □□□□ 形NX102- □□□□ 形NX1P2- □□□□ 形NJ501- □□□□ 形NJ301- □□□□ 形 NJ101- □□□□	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 ハードウェア編(SBCA-466)／ソフトウェア 編(SBCA-467)、ユーザーズマニュアル モ ーション制御(SBCE-433)と併せて使用して ください。
NJ/NX シリーズ CPU ユニット 内蔵EtherCAT®ポート ユーザーズマニュアル	SBCD-376	形NX701- □□□□ 形NX102- □□□□ 形NX1P2- □□□□ 形NJ501- □□□□ 形NJ301- □□□□ 形 NJ101- □□□□	NJ/NX シリーズ CPU ユニットの内 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。 ユーザーズマニュアル ハードウェア編 (SBCA-466)／ソフトウェア編(SBCA-467) と併せて使用してください。
NJ/NX シリーズ CPU ユニット 内蔵EtherNet/IP™ ポー ト ユーザーズマニュアル	SBCD-377	形NX701- □□□□ 形NX102- □□□□ 形NX1P2- □□□□ 形NJ501- □□□□ 形NJ301- □□□□ 形 NJ101- □□□□	NJ/NX シリーズ CPU ユニットの内 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。 ユーザーズマニュアル ハードウェア編 (SBCA-466)／ソフトウェア編(SBCA-467) と併せて使用してください。
NJ/NX シリーズ トラブルシューティング マニュアル	SBCA-469	形NX701- □□□□ 形NX102- □□□□ 形NX1P2- □□□□ 形NJ501- □□□□ 形NJ301- □□□□ 形 NJ101- □□□□	NJ/NX シリーズで 検出する異常の詳 細について知りたい とき。	NJ 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。 ユーザーズマニュアル ハードウェア編 (SBCA-466)／ソフトウェア編(SBCA-467) と併せて使用してください。
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。
NJ/NX シリーズ スタートアップガイド モーション制御編	SBCE-372	形 NX1P2-□□□□ 形 NX701-□□□□ 形 NJ501-□□□□ 形 NJ301-□□□□ 形 NJ101-□□□□ 形 SYSMAC-SE20□□ 形 R88M-1□ 形 R88D-1SN□-ECT	NJ/NX シリーズの モーション制御機能 を初めて使用する 場合の立ち上げ手 順とSysmac Studio の操作方法を知り たいとき。	軸のパラメータ設定、簡単な1 軸位置決め や2 軸直線補間について、ハードウェア組 み立てからデバッグまでの操作について説 明します。

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。

Man.No.	SBCA-109B
---------	------------------

↑
改訂記号

改訂記号	改訂年月	改訂理由
A	2017 年 6 月	初版
B	2018 年 12 月	誤記修正・ご承諾事項変更

目次

はじめに.....	3
対象となる読者の方々	3
対象となる製品	3
アイコンについて	3
ご承諾事項	4
マシンオートメーションコントローラ NX1P2 CPU ユニット.....	4
オートメーションソフトウェア Sysmac Studio	6
注意事項	7
ソフトウェアのライセンスと著作権	7
関連マニュアル	8
マニュアル改訂履歴	10
1 NX1Pのプログラミング	14
1-1 NX1P の概要	15
1-2 NX1P プログラミングの特長	16
1-2-1 プログラミング環境の変化と NX1P の対応.....	16
1-2-2 プログラムの追加変更への対応	17
1-2-3 モーションプログラムが簡単.....	18
1-2-4 ST 言語により演算処理が簡単	19
1-3 変数プログラミング	20
1-3-1 NX1P のプログラミング	20
1-3-2 データ型	23
1-3-3 データ型のメリット.....	24
1-3-4 グローバル規格 IEC61131-3	25
1-4 プログラミングツール	26
1-4-1 プログラミングツール Sysmac Studio	26
1-4-2 Sysmac Studio シミュレーション機能	27
2 事前に準備すること	28
2-1 システム構成と使用機種	29
2-1-1 概要	29
2-1-2 配線	30
2-2 Sysmac Studio をインストール	32

2-2-1	Sysmac Studio をインストール	32
2-2-2	Sysmac Studio のインストール環境	32

3	NX1P のラダープログラミング	33
3-1	Sysmac Studio によるプログラミング	35
3-1-1	プログラミングの流れ	35
3-1-2	新規プロジェクトの作成	35
3-2	Sysmac Studio の画面構成	37
3-2-1	構成・設定画面	37
3-2-2	プログラミング画面	37
3-3	機器を配線した端子番号に変数名をつける	38
3-3-1	端子番号の変数名	38
3-3-2	I/O マップ設定	39
3-3-3	配線確認	41
3-4	ラダーの記述方法	42
3-4-1	ラダー回路の入力方法	42
3-4-2	ショートカットキー一覧	42
3-4-3	ラダー記述のルール	43
3-5	基本的なラダープログラム例	44
3-5-1	ラダーの入力練習	44
3-5-2	回路の入力練習をしてみましょう。	45
3-5-3	プログラムチェック	47
3-5-4	プログラムの保存	48
3-5-5	NX1P での動作確認	49
3-5-6	シミュレーションでの動作確認	50
3-5-7	プログラム作成時のエラーの例(オフライン)	52
3-5-8	運転時のエラーの例	52
3-6	タイマ命令を使用したラダープログラム例	53
3-6-1	自己保持回路	53
3-6-2	タイマ命令(TON)	54
3-6-3	演習:省エネルギー対策のエスカレータ	58
3-6-4	プログラムの動作を確認する	59
3-6-5	プログラムの動作を確認する(ウォッチウィンドウ)	60
3-7	時刻や時間を使ったラダープログラム例	62
3-7-1	NX1P での時刻・時間を使ったプログラミング	62
3-7-2	演習:エスカレータの連続移動時間	62
3-8	開発効率 UP のためのプログラミング基礎知識	66
3-8-1	POU(Program Organization Unit)	66
3-8-2	プログラムと実行優先順位(タスク)	66
3-8-3	ファンクション(FUN)とファンクションブロック(FB)	68
3-8-4	セクション	69
3-8-5	変数の種類	70

4	データを扱うプログラミング	73
4-1	データ処理に使用する変数	74

	4-1-1	配列	74
4-2		プログラミング実習	75
	4-2-1	アプリケーション例	75
	4-2-2	プログラミングの作成	75
	4-2-3	新規プロジェクトの作成	76
	4-2-4	アナログオプションボード設定	77
	4-2-5	アナログ入力と入力端子に変数を割り付ける	77
	4-2-6	プログラム例	78
	4-2-7	配列の設定方法	79
	4-2-8	プログラムの入力	80
	4-2-9	プログラムの動作確認	81
	4-2-10	配列データの参照の仕方	82
5		モーション FB プログラミング	84
5-1		モーション FB プログラミング	85
	5-1-1	モーション FB プログラミングとは	85
	5-1-2	プログラミングの流れ	85
5-2		サーボドライバの追加とパラメータ設定	86
	5-2-1	サーボドライバの登録	86
	5-2-2	軸の登録	87
	5-2-3	軸パラメータの設定	87
5-3		モーション FB プログラミング	89
	5-3-1	ラダープログラムの概要	89
	5-3-2	使用するモーション FB	89
	5-3-3	ラダープログラムの作成	90
5-4		データレース	93
	5-4-1	データレースによる動作確認	93
5-5		3D シミュレーション	95
	5-5-1	3D シミュレーションの起動方法	95
6		はじめての ST 言語プログラミング	97
6-1		ST プログラミングとは	98
	6-1-1	ST 言語の利点	98
	6-1-2	制御構文を含んだ ST プログラム	98
	6-1-3	ST 記述の基本文法と使用例	99
	6-1-4	演算子一覧	99
6-2		NX1P での ST プログラミング	100
	6-2-1	NX1P での ST プログラミングの記述方法	100
6-3		ST プログラミング実習	101
	6-3-1	数値計算プログラミング実習	101
	6-3-2	プログラム手順	102
	6-3-3	完成プログラムとプログラムチェック	104
	6-3-4	ST プログラムの動作確認	104

1

1 NX1Pのプログラミング

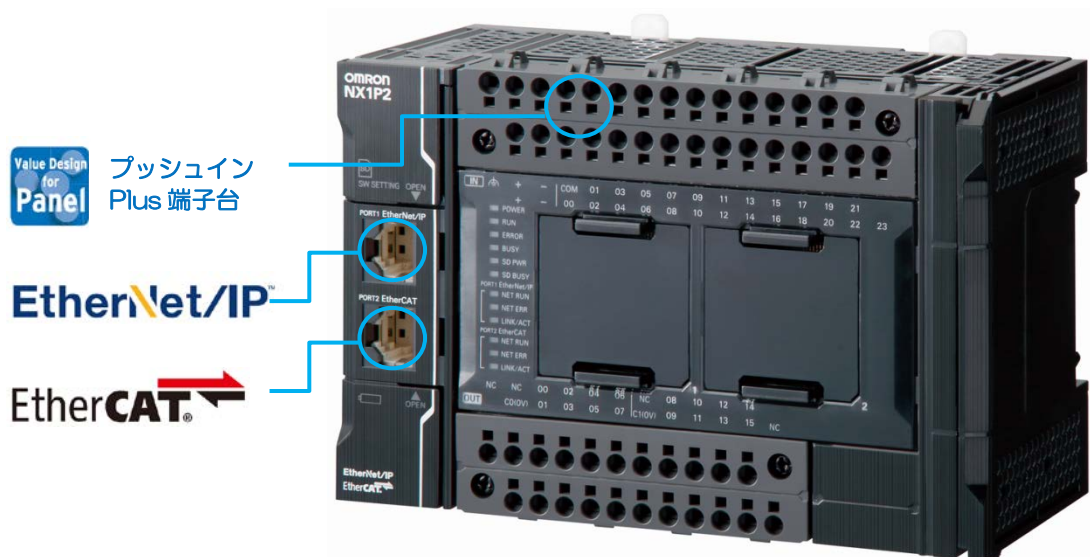
この章では、オートメーションコントローラ NX1P のプログラミングの基礎知識を説明します。

1-1	NX1P の概要	1-15
1-2	NX1P プログラミングの特長	1-16
1-2-1	プログラミング環境の変化と NX1P の対応	1-16
1-2-2	プログラムの追加変更への対応	1-17
1-2-3	モーションプログラムが簡単	1-18
1-2-4	ST 言語により演算処理が簡単	1-19
1-3	変数プログラミング	1-20
1-3-1	NX1P のプログラミング	1-20
1-3-2	データ型	1-23
1-3-3	データ型のメリット	1-24
1-3-4	グローバル規格 IEC61131-3	1-25
1-4	プログラミングツール	1-26
1-4-1	プログラミングツール Sysmac Studio	1-26
1-4-2	Sysmac Studio シミュレーション機能	1-27

1-1 NX1P の概要

この写真はNX1P2 CPU ユニットを示しています。電源配線/入出力機器配線用の端子台がプッシュイン Plus 端子台になっています。

また、通信用のインタフェースも EtherCAT と EtherNet/IP が搭載されています。



NX1P の特長

1. EtherCAT ポートを内蔵し、高性能モーション制御で、装置の高速・高精度化を実現
 - ・最大 4 軸のモーション制御
 - ・電子カム機能や、補間機能により装置の高速・高精度化を実現
 - ・単軸位置制御も含め最大 8 軸のサーボシステムを EtherCAT で簡単配線
2. IoT を実現するネットワークを実装
 - ・EtherNet/IP で上位 PC 通信や、NJ/NX シリーズ、CJ シリーズとのデータリンクが可能
3. プッシュイン端子台を採用
 - ・電線を挿し込むだけで配線できるプッシュイン Plus 端子台で、制御盤制作時、配線工数の削減が可能

また NX1P では開発生産性を高めるプログラミング環境を提供しています。

本ガイドでは NX1P のプログラミング方法の特長、NX1P と Sysmac Studio を使ったプログラミングの方法について説明しています。

1-2 NX1P プログラミングの特長

1-2-1 プログラミング環境の変化とNX1P の対応

生産現場では生産性向上や品質の向上が求められ、装置はますます高機能化、複雑化しています。そのため開発現場においては「エンジニアリングコストの圧縮」、「プログラム開發生産性の向上」、「社員教育コストの最小化」といった課題への対応に迫られています。

- ① 装置の高機能化により、複雑な位置決め制御や、それに伴う高度な演算処理が増えてきており、ラダーシーケンスでの実現では**エンジニアリングコスト増大**につながっている。
- ② さらに装置の高機能化に伴い、プログラム容量も増大し、**プログラム開發生産性向上**が課題となっている。
- ③ ラダープログラムは技術系出身者も入社後習得することが多く、**新入社員教育コストを最小化**したい。



生産現場の課題

NX1P ではこのような課題に対してそれぞれ、次の様な解決方法を提供することが可能です。



NX1Pでは

1

モーションFBプログラムで、位置決めプログラムが簡単に組める。
ST言語の採用で、演算処理や情報処理に適した言語の使い分けが可能。

2

変数プログラミングの採用で、簡単に稼働実績のあるプログラムの部品化や再利用ができる。

3

ST言語と変数プログラムの採用で、高級言語(BASICやC言語など)を習得している新入社員の即戦力化が可能。
ベテラン技術者の新人教育コストが不要。

NX1P のプログラミングの特長についてもう少し詳しく見ていきましょう。

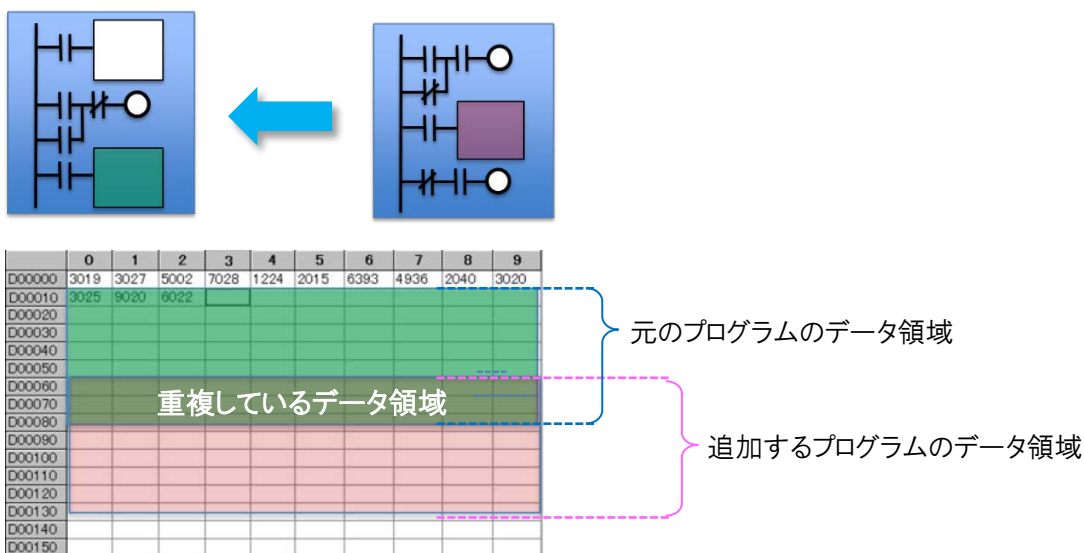
1-2-2 プログラムの追加変更への対応

従来の課題

従来の「アドレス」によるプログラミングでプログラムを追加する場合、元のプログラムと、追加するプログラムの中で、機器(I/O)のアドレスや、使用する作業領域(データメモリ)の割り当てやサイズが重複することが多く、そのたびにアドレスやデータ領域の確認/修正の作業や、修正後のデバッグなどが必要で開発生産性の向上が大きな課題でした。

元のプログラム

追加するプログラム(流用プログラム)

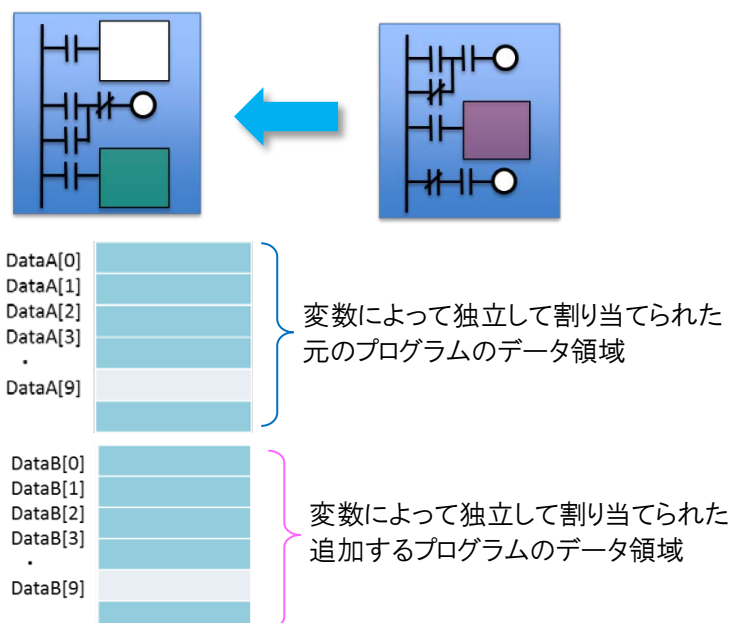


NX1P のプログラミングでは

プログラムを流用する場合に、プログラムによって使用するデータ領域が変数により独立して割り付けられるので、アドレスを気にせずに、データエリアを確保でき、追加/改造時も、アドレスの空きを探す必要がなく、プログラムの流用や修正時にデバッグ時間が削減できます。

元のプログラム

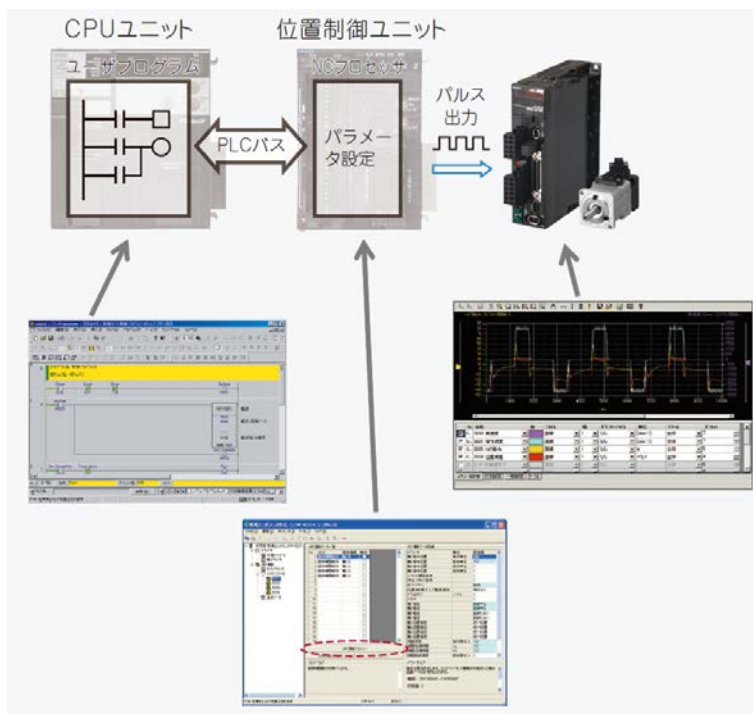
追加するプログラム(流用プログラム)



1-2-3 モーションプログラムが簡単

従来の課題

従来の PLC(CJ2 など)では、位置制御ユニットの設定、それを起動するラダープログラム、サーボの設定に 3 つのソフトウェアを使い、それぞれ設定モニタしながらのプログラムの作成が必要でした。

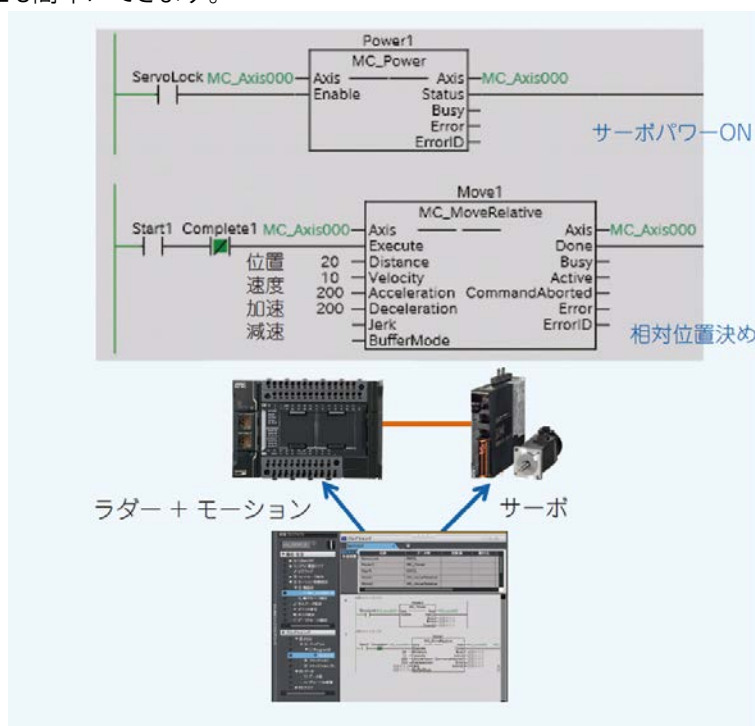


NX1P のプログラミングでは

一つのプログラム中に、モーションFBを使って、サーボONや、原点復帰、位置決めなどの動作を記述できます。

そのためプログラムの流れどおりにプログラムが組め、他の人から見やすいプログラムになります。

また、1 つのソフトウェアで「ラダー」+「モーション」+「サーボ設定」ができ、簡単に位置決め制御を実現し、モニタや修正も簡単にできます。



1-2-4 ST 言語により演算処理が簡単

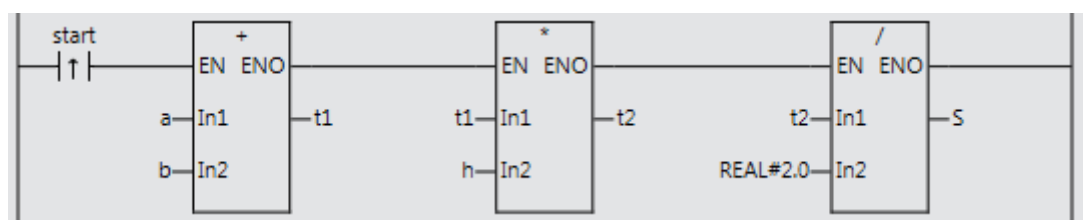
ST 言語とは

ST 言語は、PASCAL をベースに規定された構造化テキスト言語です。数値演算式の表現やネ스팅した条件分岐など、ラダー図言語が苦手としている用途で高い効果を発揮します。

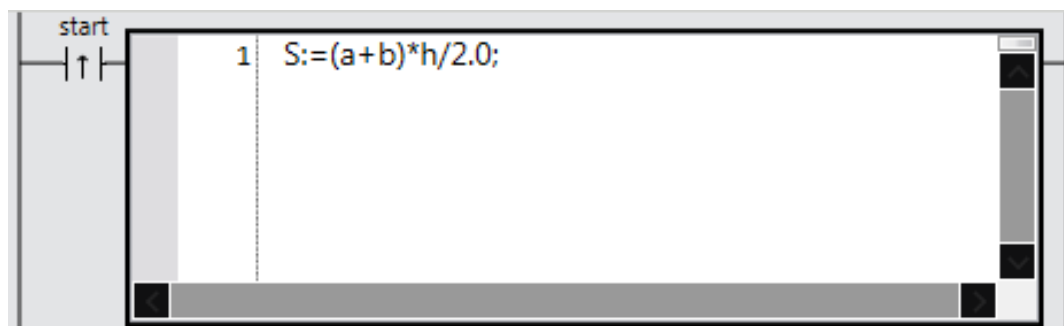
ST 言語の特長

シーケンスロジックはラダー言語、演算はST言語と、得意なプログラムで効率よく、見やすいプログラム作成が可能です。

例) 台形の面積 S の計算
(上底 + 下底) * 高さ / 2



計算式は、ST 言語を使えばさらに簡単に見やすくなります。



演算処理の記述に強い ST 言語をラダープログラムと混在させることができます。また、ST 言語だけでも、プログラムの作成が可能です。

こんな時に、ST 言語が便利です。

1. 四則演算や関数演算の記述
+, -, *, /、SIN、COS、TAN など
2. 条件分岐や繰り返しなどの制御文の記述
IF THEN、FOR NEXT など
3. 文字列処理の記述
文字列の連結、取り出し、検索、置き換えなど

次に、NX1P の変数プログラミングについて詳しく見ていきましょう。

1-3 変数プログラミング

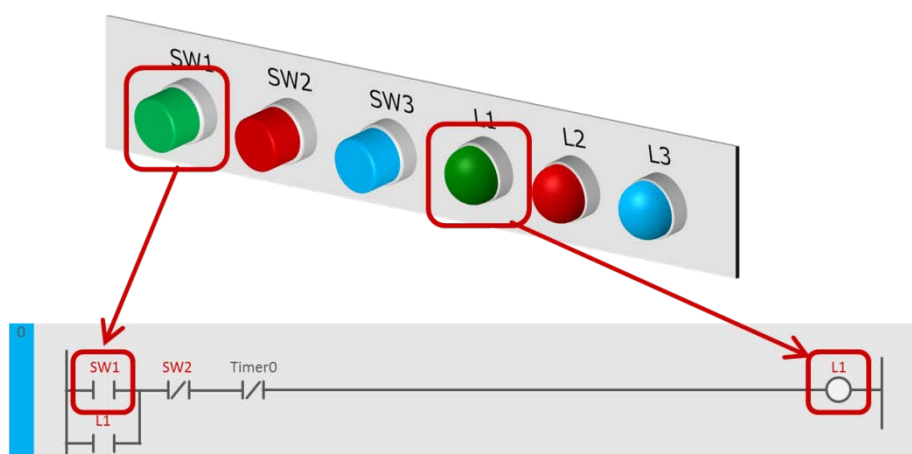
1-3-1 NX1P のプログラミング

NX1P は、従来の PLC (CJ2 など) で行われていた「アドレス」によるプログラミングではなく、ユーザが付けた「名前」(オムロンでは「変数」と呼んでいます)でプログラミングする方法を採用しています。

NX1P のプログラミングでは

変数でプログラミングすると、アドレス番号を覚える必要がないという利点だけでなく、さまざまな点で開発効率の向上につながります。

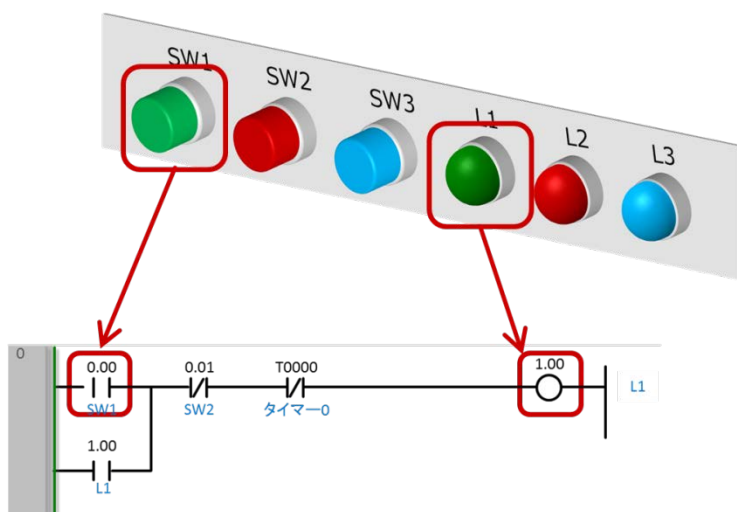
「変数でプログラミングする」とは、例えば次のように、操作盤やタッチパネルなどに表示されている「名前」でプログラミングすることです。



従来の PLC のプログラミングでは

従来の PLC (CJ2 など) では、下記のように「0.00」や「T0000」のような I/O 番号やタイマ番号(アドレス)でプログラミングしていました。

しかし、アドレス番号だけでは視認性や可読性が良くないため、多くの PLC メーカーでは I/O にコメントを併記できるようにしていました。オムロンでは「I/O コメント」という機能がそれにあたります。

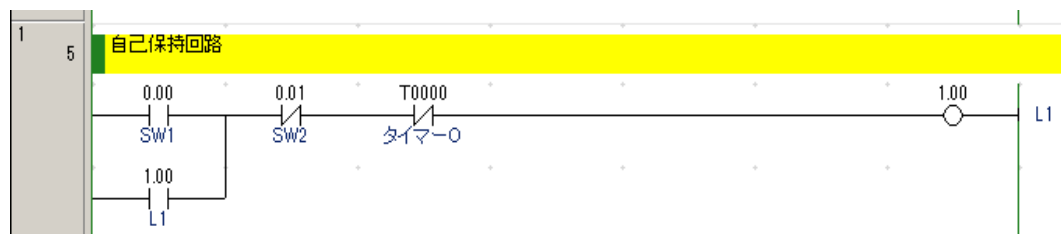


アドレスプログラミングと変数プログラミングの違い

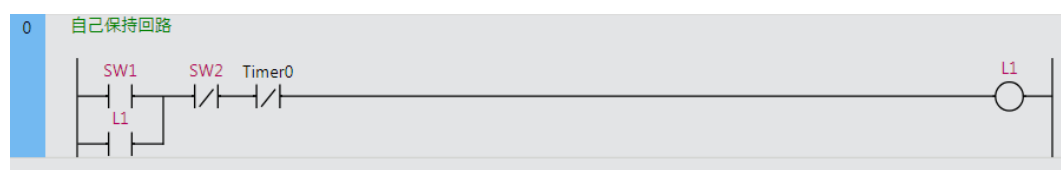
それぞれのプログラミング方法において、何が違うのでしょうか？

上段は CJ2 用プログラム(CX-P で作成)、下段は NX1P 用プログラム(Sysmac Studio で作成)です。

CJ2 のプログラミング例



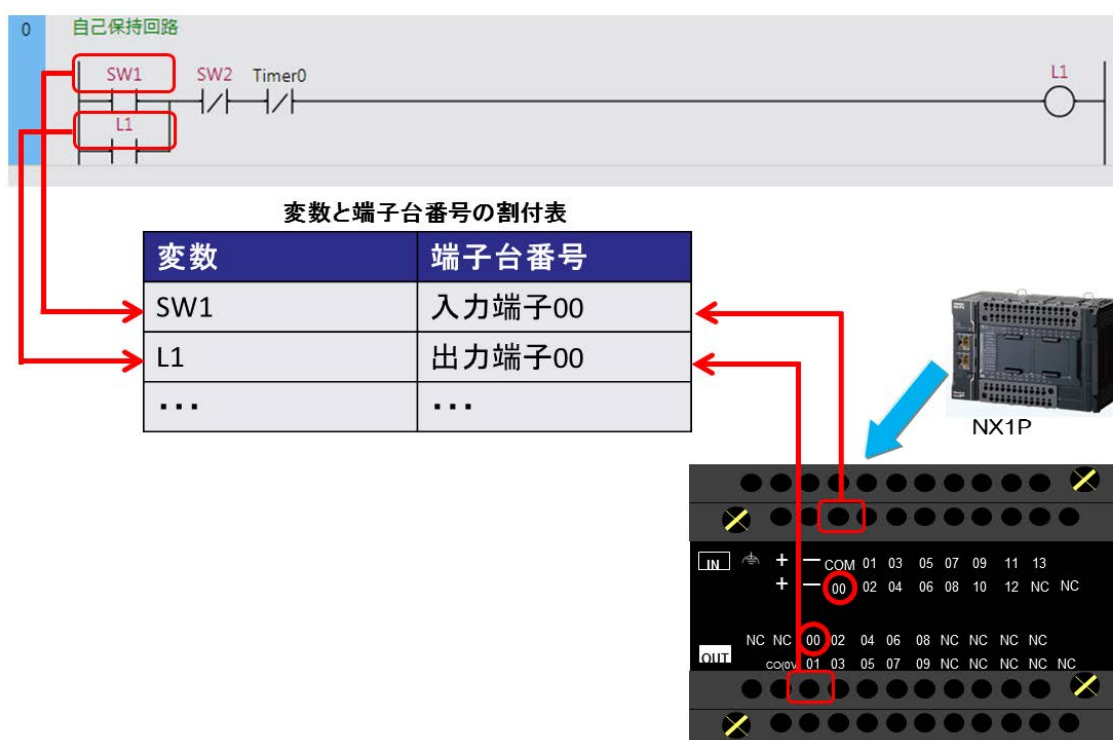
NX1P のプログラミング例



上のプログラムはどちらも同じ動きをします。

このとき“SW1”や“L1”などの変数は実際はどの端子番号の入出力機器に相当するのでしょうか？

NX1P のプログラミングでは“SW1”や“L1”などの変数に対して、実際に配線される I/O の端子台番号を I/O マップを使って割り付けます。

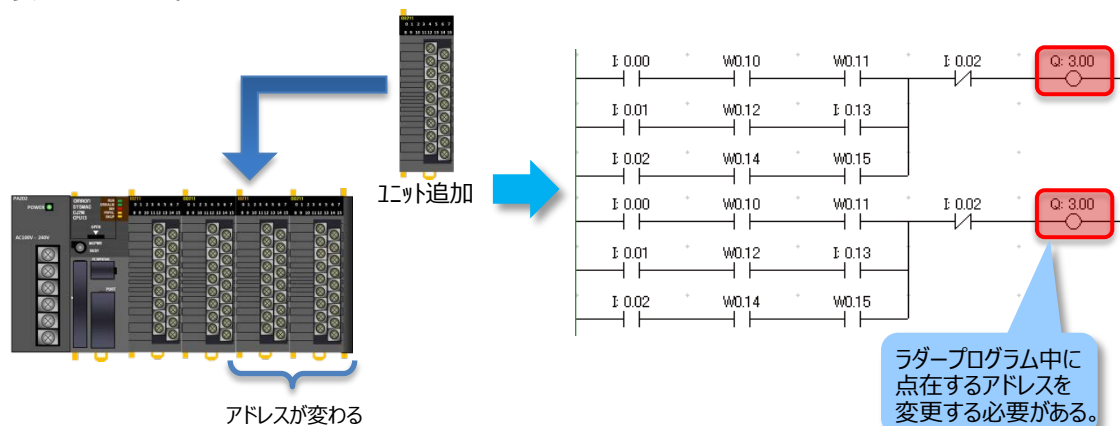


このように変数プログラミングは、プログラムで使用している変数と、実際の I/O(この場合は、NX1P の内蔵 I/O 端子番号)を紐づけています。

装置ごとに I/O 割付が変わっても、割付表の端子台番号を変更するだけです。プログラム自体を変更する必要がなくなることが変数プログラミングのメリットです。

従来の課題

ユニット構成が変わると、I/O の端子台配置が変更になるため、ラダープログラムのアドレスも変更する必要があります。



NX1P のプログラミングでは

ユニットを追加するなど、ユニットの構成が変わっても、変数で入出力エリアを割り付けている NX1P では、「I/O マップ」に、追加したユニットの入出力に変数を割り付けるだけで、プログラムは変更する必要はありません。



1-3-2 データ型

変数プログラミングにおいて、例えば、ランプに「L1」(Lamp の1 番目の意味)という名前を付けたとします。ランプなので、ON または OFF するデータということがすぐに理解できます。

しかし、あるデータに「Data1」という名前を付けた場合は、それが 10 進数データなのか、小数点データなのか、文字列データなのか分かりません。

そこで、変数がどんな種類のデータなのかを指定するために「データ型」が用意されています。

変数は、名前(変数名)とそれに対応するデータ型で表します。



データ型の一例

NX1P では以下のようなデータ型を変数に指定します。例えば、ON/OFF データであれば「BOOL」型、10 進数(整数)なら「INT」型、文字列なら「STRING」型を指定します。

また、10 進数でも扱えるデータ範囲によって、INT、DINT などを使い分けることができます。

この表の記載以外にも、ビット列を扱う WORD 型や、日付時刻を扱う、DATA AND TIME 型など、いろいろなデータ型に対応しています。

データ種類	主な使用用途	主なデータ型		データ範囲	データの例
ON/OFF データ	接点やコイルの ON/OFF	BOOL		0 ~ 1	・1 / 0 ・TRUE / FALSE
10 進数 (整数)	データの数値演算	符号付き	INT	-32768 ~ +32767	+30000、 -20000 など
			DINT	-2147483648 ~ +2147483647	+12345678 -200000000 など
		符号無	UINT	0 ~ 65535	60000 など
			UDINT	0 ~ 4294967295	200000000 など
浮動小数点	実数型	REAL		単精度複素数浮動小数点値	0.15625 など
		LREAL		倍精度複素数浮動小数点値	1.00000000000000002 など
文字列	表示器への文字列表示処理	STRING		文字列(UTF-8) (約 2000 バイト)	‘OMRON’ ‘不良率’ など

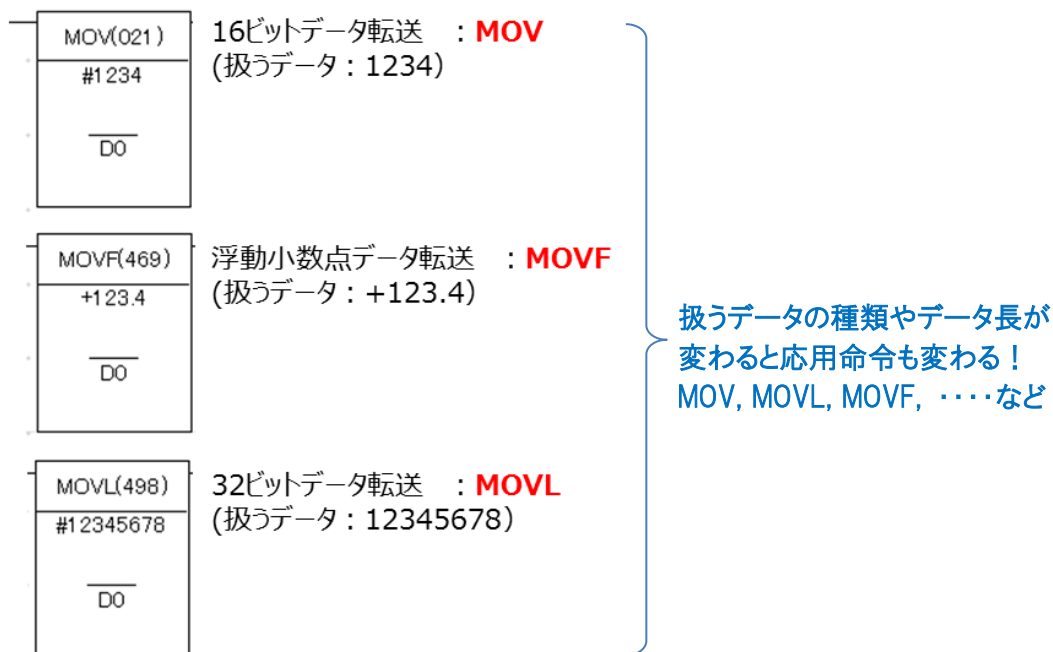
1-3-3 データ型のメリット

従来の PLC (CJ2 など) では、応用命令を使用する際、扱うデータの種類ごとに命令語を使い分ける必要がありました。

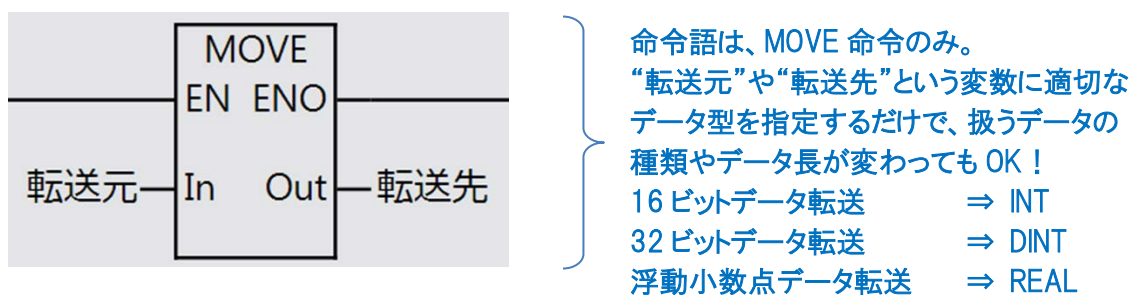
NX1P では、応用命令のオペランドも変数で指定し、その変数にはデータ型が設定されているので、これまでのように扱うデータの種類ごとに命令語を使い分ける必要がなくなりました。

また、データ長が16ビットから32ビットに変更になった場合も、変数のデータ型を変えるだけで済み、プログラムの変更やデータメモリ領域を確保する必要はありません。

● 従来の PLC の場合 (CJ2 など) の場合



● NX1P の場合



このように、「変数によるプログラミング方式」や「データ型」の考え方は、国際規格 IEC 61131-3に準拠したプログラミングスタイルで、日本国内でも、近年急速に普及してきています。

1-3-4 グローバル規格 IEC61131-3

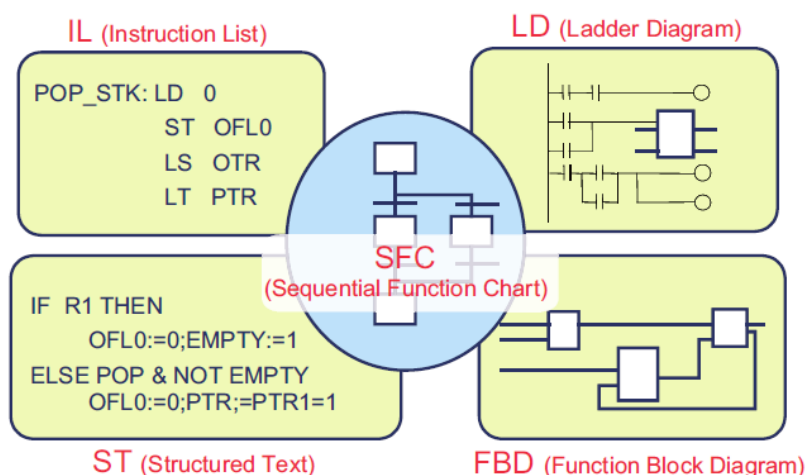
IEC61131-3 とは

国際規格 IEC 61131-3^{※1} とは、以下の点をコンセプトとして、1993 年に発行された国際規格です。

- ・メーカーや機種に依存しない
 - ・ソフトウェアの部品化・再利用化
 - ・用途や技術者のスキルに応じて使える、多種のプログラミング言語
- 日本では、JIS B 3503 として規格化されています。

※1: IEC(International Electrotechnical Commission / 国際電気標準会議)

IEC 61131-3 で規定されている言語は以下の 5 種類です。



- ・IL(Instruction List): アセンブラ言語相当のニモニックを規格化したテキスト言語
- ・LD(Ladder Diagram): リレー回路を記号化したグラフィック言語
- ・ST(Structured Text): 高級言語の PASCAL に似た、演算や情報処理に適したテキスト言語
- ・FBD(Function Block Diagram): 機能をブロックに部品化したグラフィック言語
- ・SFC(Sequential Function Chart): 工程を状態遷移図のように記述できるグラフィック言語

NX1P では、この 5 言語のうち、LD と ST に対応しています。



IEC 61131-3 規格の普及状況

グローバル競争の中で、IEC 61131-3 規格の要求は益々高くなっています。

欧州、北米から導入が進み、アジア圏でも、工場進出してきた欧州企業の影響で普及し始めてきています。

NX1P では、この IEC 61131-3 に準拠したプログラミング環境を採用しているため、海外工場に設備を導入した場合の現地社員への教育も、スムーズに対応できます。

1-4 プログラミングツール

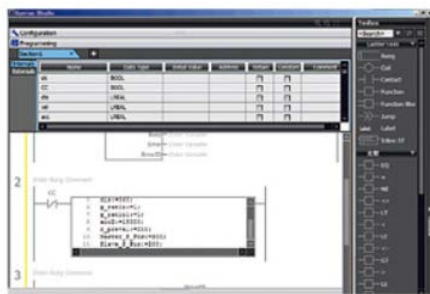
1-4-1 プログラミングツール Sysmac Studio

Sysmac Studio とは

Sysmac Studio は、NJ/NX シリーズ CPU ユニットをはじめとするマシンオートメーションコントローラのプログラミング、デバッグ、メンテナンスのための、統合開発環境を提供するソフトウェアです。

主な特長

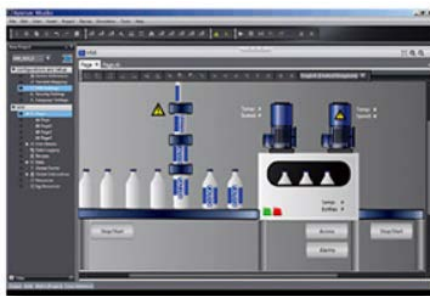
1. モーション、ロジック、セーフティ、画像、HMI の設定・プログラミングをひとつのソフトウェアに統合
2. 変数プログラミング、ラダー言語/ST 言語/FB をサポート
3. 複雑なモーション制御を簡単に実現できるモーション FB プログラミング(PLCopen 認定)や、カム動作を簡単に設定できるカムエディタを搭載
4. モーション統合シミュレーション、デバッグ機能の搭載により、PLC プログラムとそのモーション動作を2D や3D でグラフィカルに確認することができ、立ち上げ・調整期間短縮が可能



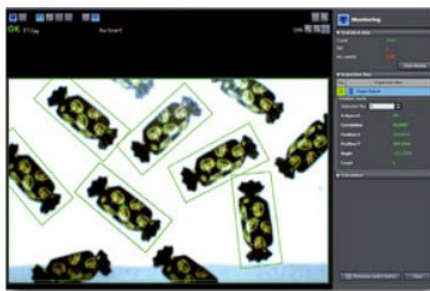
プログラミング



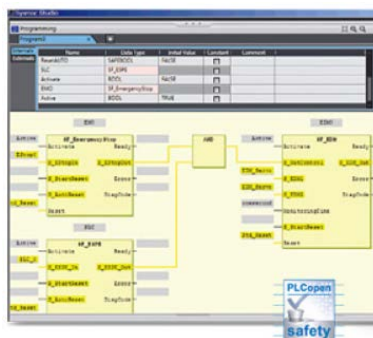
モーションコントロール



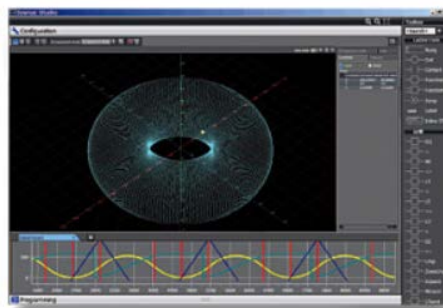
HMI



画像センサ



セーフティ

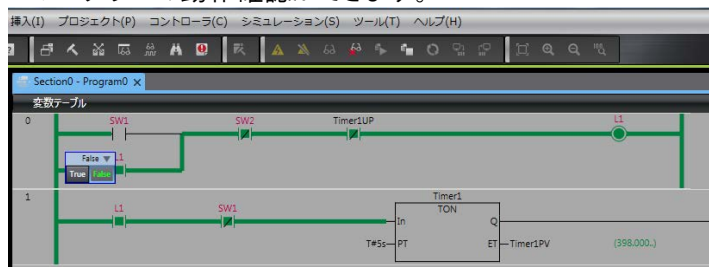


シミュレーション

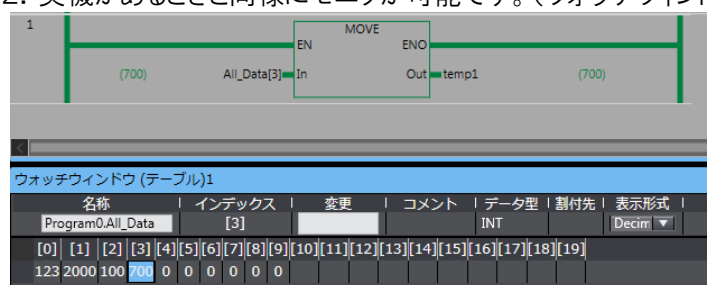
1-4-2 Sysmac Studio シミュレーション機能

Sysmac Studio では、充実したシミュレーション機能を搭載しています。
シミュレーション機能により、実機がなくても下記のような、ことが、ツール上で確認できます。

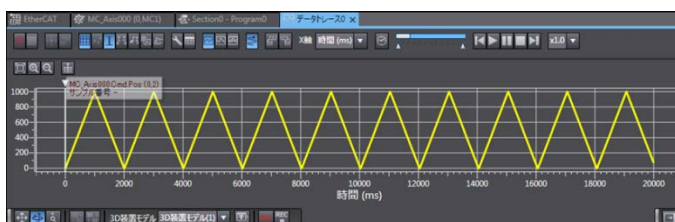
1. プログラムの動作確認ができます。



2. 実機があるときと同様にモニタが可能です。(ウォッチウィンドウでの変数データのモニタ)



3. 変化するデータのトレースや、モーションプログラムでの位置や速度などの変化もデータトレース機能で確認ができます。



4. モーション機能の3Dシミュレーションが可能で、実機がなくても、軌跡を確認できます。



2

2 事前に準備すること

ここでは、本ガイドで使用する機器とその配線、プログラミングを作成する Sysmac Studio のPCへのインストールを行います。

2-1	システム構成と使用機種	2-29
2-1-1	概要	2-29
2-1-2	配線	2-30
2-2	Sysmac Studio をインストール	2-32
2-2-1	Sysmac Studio をインストール	2-32
2-2-2	Sysmac Studio のインストール環境	2-32

2-1 システム構成と使用機種

2-1-1 概要

「第3章、NX1P のラダープログラミング」では、電源、押しボタン、ランプを NX1P と配線して、簡単なラダープログラミングを行います。

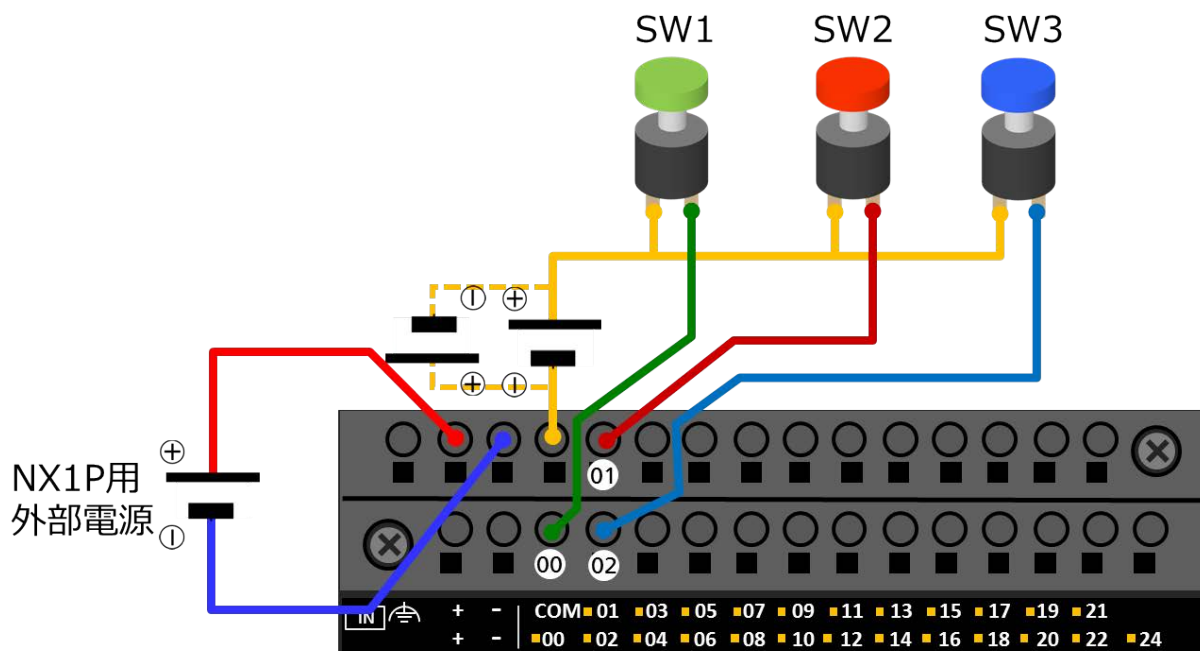
		
マシンオートメーションソフトウェア Sysmac Studio スタンダードエディション Ver.1.17 以降	マシンオートメーションコントローラ NX1P	Ethernet ケーブル (100BASE-TX/10BASE-T)
形SYSMAC-SE200D (メディアのみ) 形SYSMAC-SE201L (1ライセンス)	形 NX1P2-□□□□	-
		
押しボタン	ランプ	電源
形 A22NL など	形 M22N など	形 S8VK-S06024 など

実際に、NX1P や押しボタン、ランプの実機があるところでプログラミングの理解はしやすくなります。
実機が手元にない場合でも Sysmac Studio のシミュレーション機能を使って動作確認ができます。
また、4章、5章、6章では、シミュレーション機能を使って動作確認を行っています。

2-1-2 配線

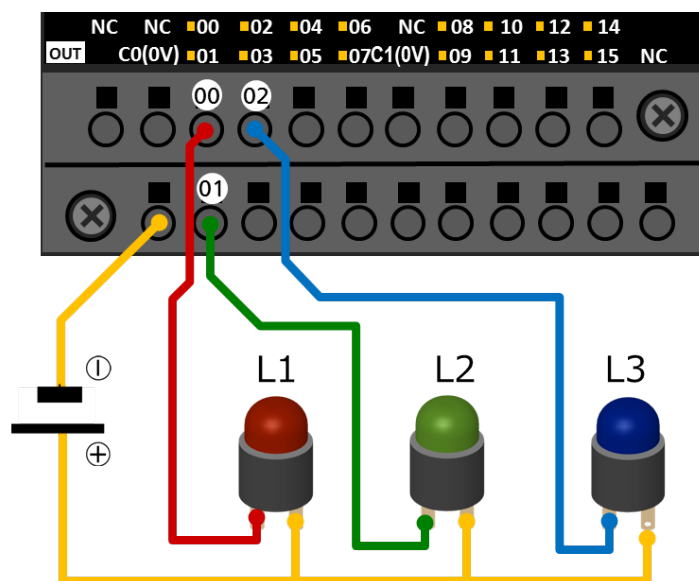
押しボタンスイッチの配線

ここでは、NX1P に押しボタンスイッチを配線して簡単なラダープログラミングを行います。
下記のように配線してください。



入力機器用電源には極性がありません。

ランプの配線



電源には極性があります。

プッシュイン Plus 端子台

プッシュイン端子台は、電線を挿し込むだけで配線できるプッシュイン方式を採用した端子台です。制御盤の制作において配線工数が削減でき、工数削減に貢献します。

プッシュイン Plus 端子台とは、電線(フェルール端子など)を挿し込むだけで接続できるプッシュイン方式を採用した端子台で、挿しやすく、かつ、抜けにくくしたオムロンが独自に開発した端子台です。

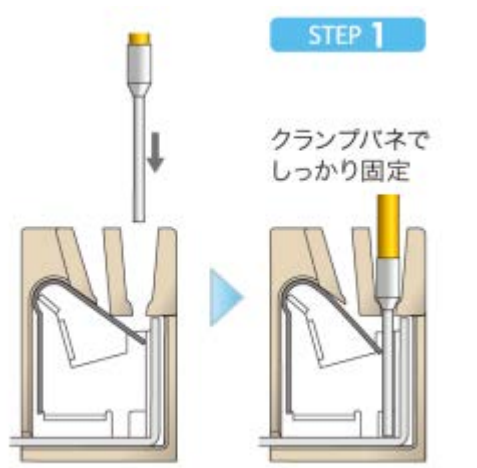


フェルール

プッシュイン Plus 端子台への電線(単線、フェルール)の接続の仕方と取り外し方は以下となります。(より線の場合についてはマニュアル等を参照ください)

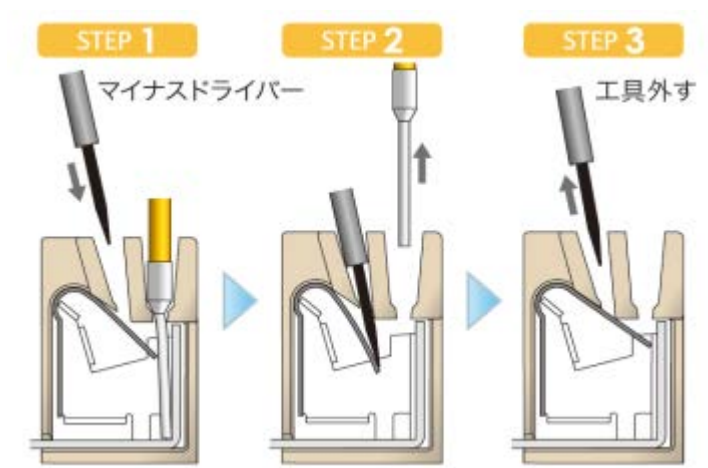
■ 電線背を接続するとき

電線を突き当たるまで挿入するだけです。より線の場合はフェルール端子を使うか、先に工具を使ってクランプバネを緩めてから電線を挿入し 工具を抜きます。



■ 電線を取り外すとき

工具(マイナスドライバー)をリリースホールに下図の様に傾けて押し込んでクランプバネを緩めてから電線を抜いてください。そのあと工具を抜きます。



2-2 Sysmac Studio をインストール

2-2-1 Sysmac Studio をインストール

NJ/NX シリーズでは、コントローラ構成やコントローラ設定の作成、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 を使用します。

以下の手順で使用するパソコンに Sysmac Studio をインストールします。

1. Sysmac Studio のインストールディスクを DVD-ROM ドライブにセットしてください。
セットアッププログラムが自動的に起動され、[セットアップ言語の選択]ダイアログボックスが表示されます。
2. インストールする言語を選択し、[OK]ボタンをクリックしてください。
Sysmac Studio のセットアップウィザードが表示されます。
3. セットアップウィザードに従ってインストールしてください。
4. インストールが完了したら、パソコンを再起動してください。

2-2-2 Sysmac Studio のインストール環境

Sysmac Studio をインストールするパソコンの動作環境は以下のとおりです。

OS	CPU		RAM	ディスプレイ
Windows 7 32 ビット版、64 ビット版 Windows 8 32 ビット版、64 ビット版 Windows 8.1 32 ビット版、64 ビット版 Windows 10 32 ビット版、64 ビット版	最低	Intel® Celeron® processor 540 (1.8GHz)以上のプロセッサを搭載した DOS/V(IBM AT 互換機)パーソナルコンピュータ	2GB	XGA 1024 x 768 1600 万色
	推奨	Intel® Core™ i5 M520 processor(2.4GHz)、または相当以上のプロセッサを搭載した DOS/V(IBM AT 互換機)パーソナルコンピュータ	4GB 以上	WXGA 1280 x 800 1600 万色



使用上の注意

CX-One の Ver.4 より前のバージョンをインストールしている場合は、Sysmac Studio のインストールは中止され、インストールができません。この場合は、CX-One をアンインストールしてから Sysmac Studio をインストールしてください。

3

3 NX1P のラダープログラミング

この章では、Sysmac Studio を使ったラダープログラミングの方法を学習します。

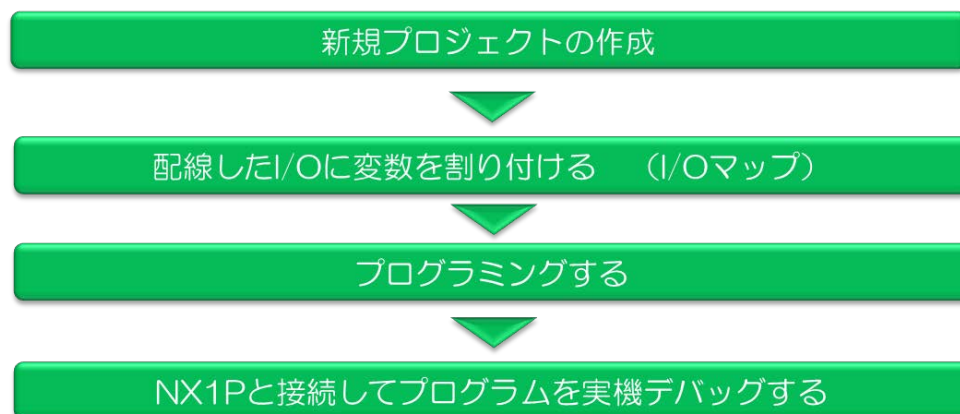
3-1	Sysmac Studio によるプログラミング	3-35
3-1-1	プログラミングの流れ	3-35
3-1-2	新規プロジェクトの作成	3-35
3-2	Sysmac Studio の画面構成	3-37
3-2-1	構成・設定画面	3-37
3-2-2	プログラミング画面	3-37
3-3	機器を配線した端子番号に変数名をつける	3-38
3-3-1	端子番号の変数名	3-38
3-3-2	I/O マップ設定	3-39
3-3-3	配線確認	3-41
3-4	ラダーの記述方法	3-42
3-4-1	ラダー回路の入力方法	3-42
3-4-2	ショートカットキー一覧	3-42
3-4-3	ラダー記述のルール	3-43
3-5	基本的なラダープログラム例	3-44
3-5-1	ラダーの入力練習	3-44
3-5-2	回路の入力練習をしてみましょう。	3-45
3-5-3	プログラムチェック	3-47
3-5-4	プログラムの保存	3-48
3-5-5	NX1P での動作確認	3-49
3-5-6	シミュレーションでの動作確認	3-50
3-5-7	プログラム作成時のエラーの例(オフライン)	3-52
3-5-8	運転時のエラーの例	3-52
3-6	タイマ命令を使用したラダープログラム例	3-53
3-6-1	自己保持回路	3-53
3-6-2	タイマ命令(TON)	3-54

3-6-3	演習：省エネルギー対策のエスカレータ	3-58
3-6-4	プログラムの動作を確認する	3-59
3-6-5	プログラムの動作を確認する(ウォッチウィンドウ)	3-60
3-7	時刻や時間を使ったラダープログラム例	3-62
3-7-1	NX1P での時刻・時間を使ったプログラミング	3-62
3-7-2	演習：エスカレータの連続移動時間	3-62
3-8	開発効率 UP のためのプログラミング基礎知識	3-66
3-8-1	POU(Program Organization Unit)	3-66
3-8-2	プログラムと実行優先順位(タスク)	3-66
3-8-3	ファンクション(FUN)とファンクションブロック(FB)	3-68
3-8-4	セクション	3-69
3-8-5	変数の種類	3-70

3-1 Sysmac Studio によるプログラミング

3-1-1 プログラミングの流れ

この章では、押しボタンスイッチとランプを使って簡単なラダープログラムを作成します。
プログラミングからデバッグまでの基本的な立ち上げ手順は下記のとおりです。



3-1-2 新規プロジェクトの作成

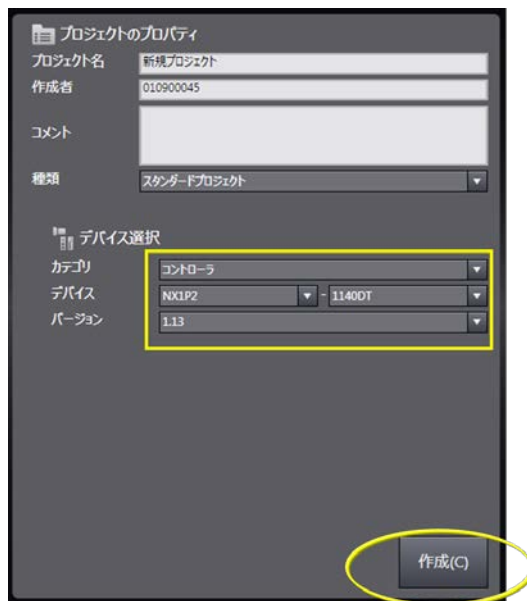
1. Sysmac Studio を起動します。



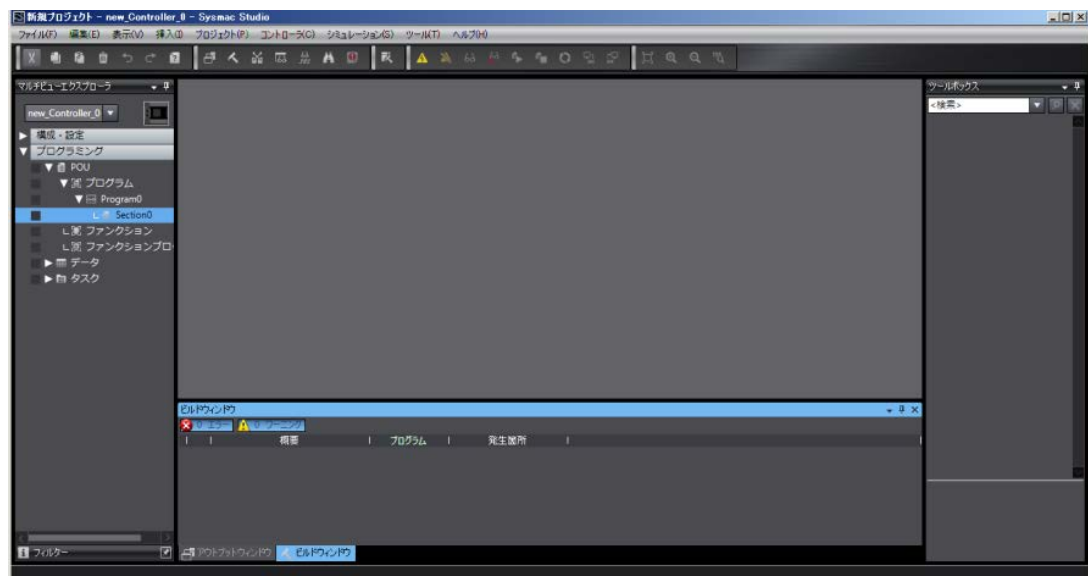
2. プロジェクト名を記載します。

デバイスは、「NX1P2」「9024DT/1140DT など」

バージョン「1.13(NX1P 本体に記載のバージョン)」を選択し、[作成]ボタンをクリックします。

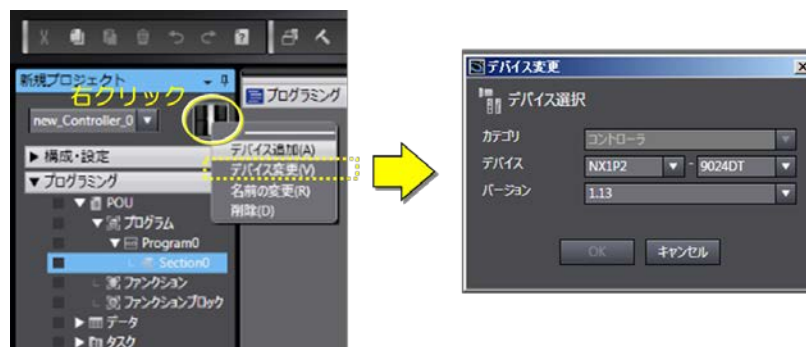


以下の操作画面が起動します。



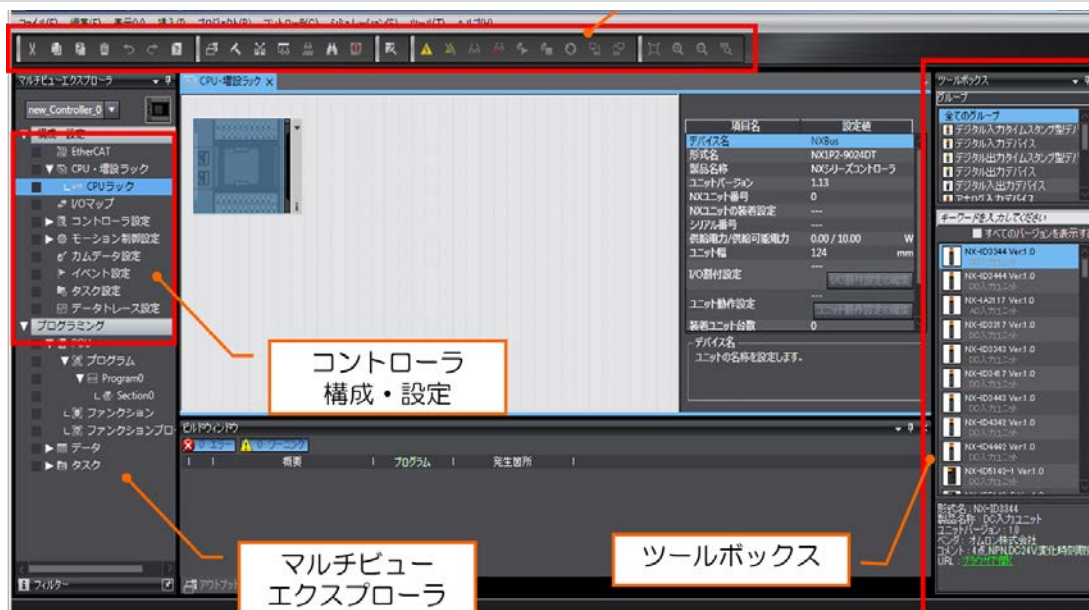
参考

プロジェクトファイルを作成した後も、NX1P の形式やバージョンの変更ができます。

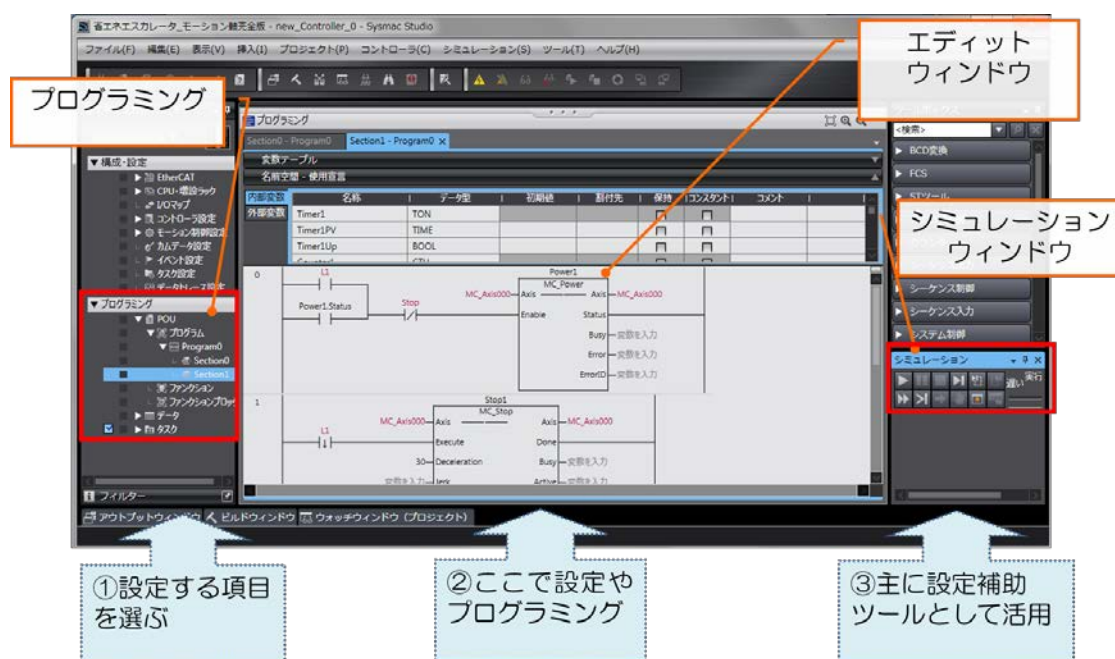


3-2 Sysmac Studio の画面構成

3-2-1 構成・設定画面



3-2-2 プログラミング画面



3-3 機器を配線した端子番号に変数名をつける

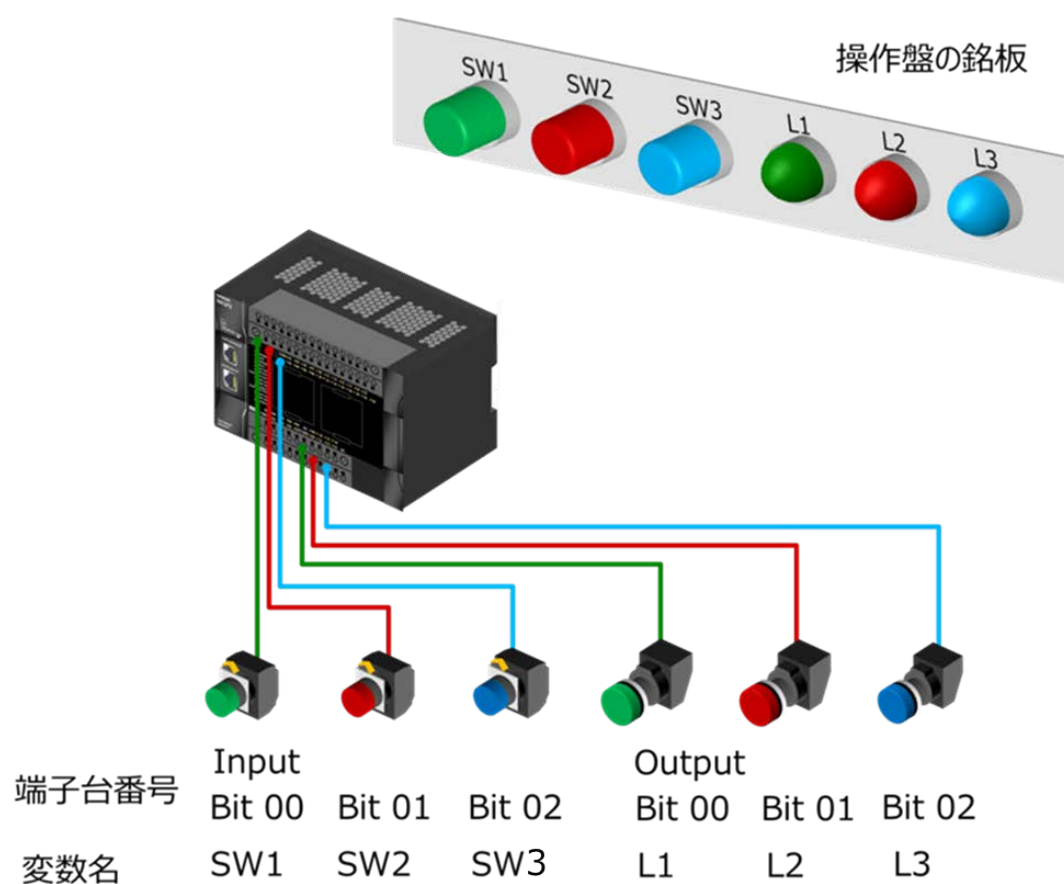
3-3-1 端子番号の変数名

NX1P の入力や出力の端子台にスイッチやランプを配線しましたが、現時点では単に物理的に配線されているだけでプログラム上では使うことができません。

それぞれの端子台に接続した機器 (I/O) をプログラムで記述できるようにするためには端子番号に変数名を付ける必要があります。

変数名は自由に付けることができますが、機器の種類や処理内容が連想できる名前にします。

例えば、NX1P の IN の端子台 00 に配線したスイッチには、操作盤の銘板の内容をそのまま使って、「SW1」という変数を付けるのもよいでしょう。



3-3-2 I/O マップ設定

端子番号に変数名を設定します。

I/O マップ設定とは、機器(I/O)が配線されている NX1P の端子台の

番号(ツール上では「I/O ポート名」と呼びます)とプログラムで使用する名前(変数名)の関係を、結びつけるしくみです。

1. マルチビューエクスプローラから[構成・設定]-[I/O マップ]をダブルクリックし、I/O マップ画面を表示させます。

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
	EtherCATネットワークコンフィグレーション						
	CPU-増設ラック						
Built-in	内蔵I/O設定						
	Input Bit 00	入力接点00	R	BOOL			
	Input Bit 01	入力接点01	R	BOOL			
	Input Bit 02	入力接点02	R	BOOL			
	Input Bit 03	入力接点03	R	BOOL			
	Input Bit 04	入力接点04	R	BOOL			
	Input Bit 05	入力接点05	R	BOOL			
	Input Bit 06	入力接点06	R	BOOL			
	Input Bit 07	入力接点07	R	BOOL			
	Input Bit 08	入力接点08	R	BOOL			
	Input Bit 09	入力接点09	R	BOOL			
	Input Bit 10	入力接点10	R	BOOL			
	Input Bit 11	入力接点11	R	BOOL			
	Input Bit 12	入力接点12	R	BOOL			
	Input Bit 13	入力接点13	R	BOOL			
	Output Bit 00	出力接点00	RW	BOOL			
	Output Bit 01	出力接点01	RW	BOOL			

デバイス選択で、NX1P を選択しているので、NX1P の I/O 端子台番号(「I/O ポート名」:Input bit 00 など)がすべて I/O マップ画面上に自動的に作成されています。

2. 変数を入力したい箇所でダブルクリックします。
配線した入力/出力の端子台番号ごとに変数名を入力します。

ここでは以下のように変数を設定していきます。

■入力端子番号につけた変数名

端子台番号	R/W	変数名	データ型
Input bit 00	R	SW1 *1	BOOL
Input bit 01	R	SW2	BOOL
Input bit 02	R	SW3	BOOL

■出力端子番号につけた変数名

端子台番号	R/W	変数名	データ型
Output bit 00	R	L1 *2	BOOL
Output bit 01	R	L2	BOOL
Output bit 02	R	L3	BOOL

*1 SW はスイッチの意味です。

*2 L はランプの意味です。

3. Input Bit 00 に「SW1」を設定します。

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
	EtherCATネットワークコンフィグレーション						
	CPU-増設ラック						
Built-in	内蔵I/O設定						
	Input Bit 00	入力接点00	R	BOOL	SW1		グローバル変数
	Input Bit 01	入力接点01	R	BOOL			
	Input Bit 02	入力接点02	R	BOOL			

4. 「InputBit 00～02」、および「OutBit 00～02」に SW1～SW3、L1～L3 という変数を設定します。

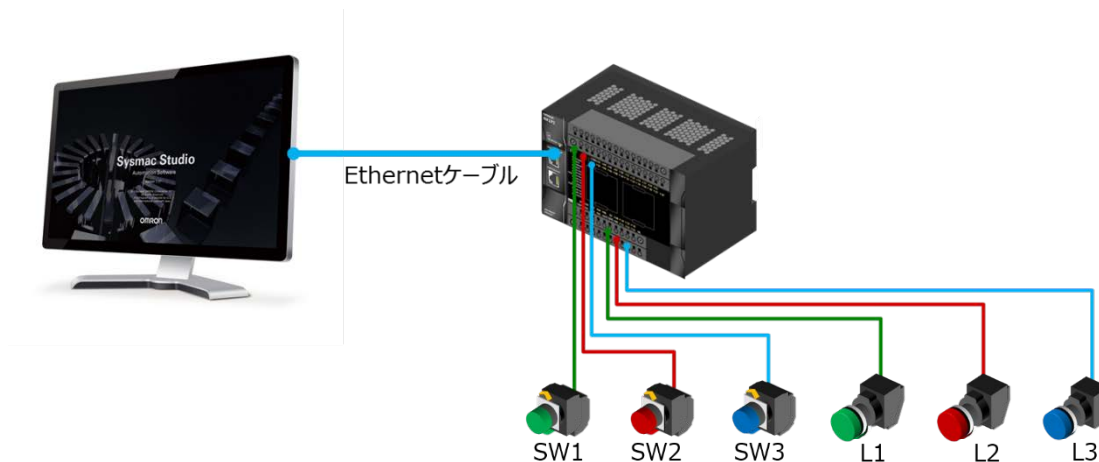
Built-in	▼ 内蔵/O設定						
	Input Bit 00	入力接点00	R	BOOL	SW1		グローバル変数
	Input Bit 01	入力接点01	R	BOOL	SW2		グローバル変数
	Input Bit 02	入力接点02	R	BOOL	SW3		グローバル変数
	Input Bit 03	入力接点03	R	BOOL			
	Input Bit 04	入力接点04	R	BOOL			
	Input Bit 05	入力接点05	R	BOOL			

	Output Bit 00	出力接点00	RW	BOOL	L1		グローバル変数
	Output Bit 01	出力接点01	RW	BOOL	L2		グローバル変数
	Output Bit 02	出力接点02	RW	BOOL	L3		グローバル変数
	Output Bit 03	出力接点03	RW	BOOL			
	Output Bit 04	出力接点04	RW	BOOL			
	Output Bit 05	出力接点05	RW	BOOL			

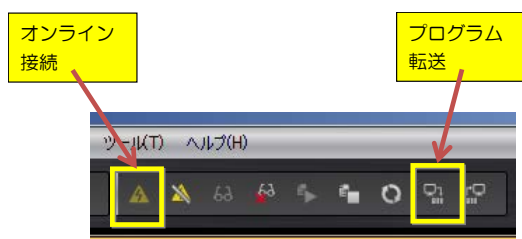
これで、端子番号に変数名が紐づきました。

3-3-3 配線確認

1. NX1P と PC (Sysmac Studio) を Ethernet ケーブルで接続します。



2. オンラインにして、I/O マップの設定内容を NX1P に転送します。



3. I/O チェック中はプログラムが実行されないようにプログラムモードに変更します。



4. 押しボタンスイッチを押して、I/O マップ画面上で入力接点の値が変化することを確認します。

5. 該当行で右クリックし、サブメニューから[セット]または[リセット]を選択してランプが ON または OFF することを確認します。

ポート	説明	R/W	データ型	値	変数
EtherCATネットワークコン					
CPU・増設ラック					
内部I/O設定					
Input Bit 00	入力接点00	R	BOOL	TRUE	SW1
Input Bit 01	入力接点01	R	BOOL	FALSE	SW2
Input Bit 02	入力接点02	R	BOOL	TRUE	SW3
Input Bit 03	入力接点03	R	BOOL	FALSE	
Input Bit 04	入力接点04	R	BOOL	FALSE	
Input Bit 05	入力接点05	R	BOOL	FALSE	
Input Bit 06	入力接点06	R	BOOL	FALSE	
Input Bit 07	入力接点07	R	BOOL	FALSE	
Input Bit 08	入力接点08	R	BOOL	FALSE	
Input Bit 09	入力接点09	R	BOOL	FALSE	

3-4 ラダーの記述方法

3-4-1 ラダー回路の入力方法

接点、コイルの挿入方法

- ショートカット入力の場合 : 回路に挿入したいところで下表のショートカットキーを押します。
例) a 接点:[C]キー、コイル:[O]キー
- ツールバー入力の場合 : ツールボックスからパーツを選んで挿入個所へドラッグ&ドロップします。
- 右クリック入力の場合 : 挿入個所で、右クリックメニューから[接点を挿入]、もしくは[コイルを挿入]を選択します。

並列回路の挿入方法

- ショートカット入力の場合 : W キーを押します。(並列に a 接点が挿入されます)
- ドラッグ&ドロップ入力の場合 : ドラッグして線を引きポップアップメニューからパーツを選択します。

回路行の追加

- ショートカットキー入力の場合 : 接続線上でカーソルをあて、[R]キーを押します。
- 右クリック入力の場合 : 回路の左端ヘッダで右クリックし、メニューから[回路を挿入]を選択します。

3-4-2 ショートカットキー一覧

覚えておくとラダー作成に便利なショートカットキーの一覧です

機能	ショートカットキー	参考(CX-P との入力比較)
a 接点を入力	[C]	同じ ※CX-P では[C]または[L]
b 接点を入力	[I]	同じ
a 接点 OR を入力	[W]	同じ ※追加する OR 回路のカーソル位置に注意
b 接点 OR を入力	[X]	同じ ※追加する OR 回路のカーソル位置に注意
出力コイルを入力	[O]	同じ
出力コイル(否定)入力	[Q]	同じ
ファンクションブロック呼出	[F]	同じ ※FUN/FB の区別なく[F]
ファンクション呼出	[I]	違う ※命令語の先頭文字を入力
ラダー回路の行追加(下)	[R]	同じ
ラダー回路の行追加(上)	[Shift]+[R]	

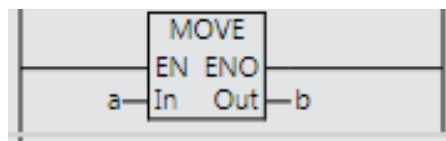
※[ヘルプ]-[ショートカットキーリファレンス]にてオンラインヘルプを表示することもできます。

3-4-3 ラダー記述のルール

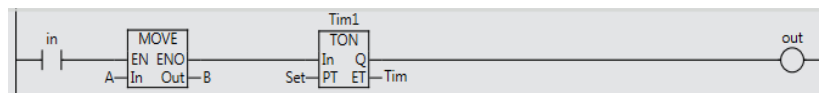
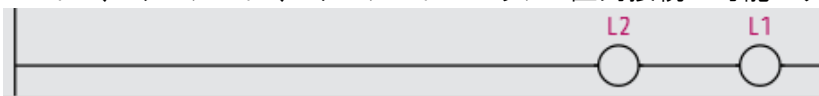
NX1P で記述できること

NX1P では、従来の PLC(CJ2 など)では記述できなかった回路(以下の回路例)も記述できるようになります。

- コイルは、接点がなくとも入力できます。(常時 ON リレーがなくともエラーになりません)
同様にファンクション、ファンクションブロックも可能です。



- コイル、ファンクション、ファンクションブロックの直列接続が可能です。



禁止事項

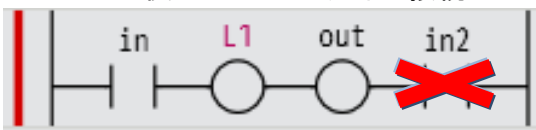
- 空回路は設定できません。



- 1つの回路接点だけの入力はできません。



- コイルの後にはコイル以外は接続できません



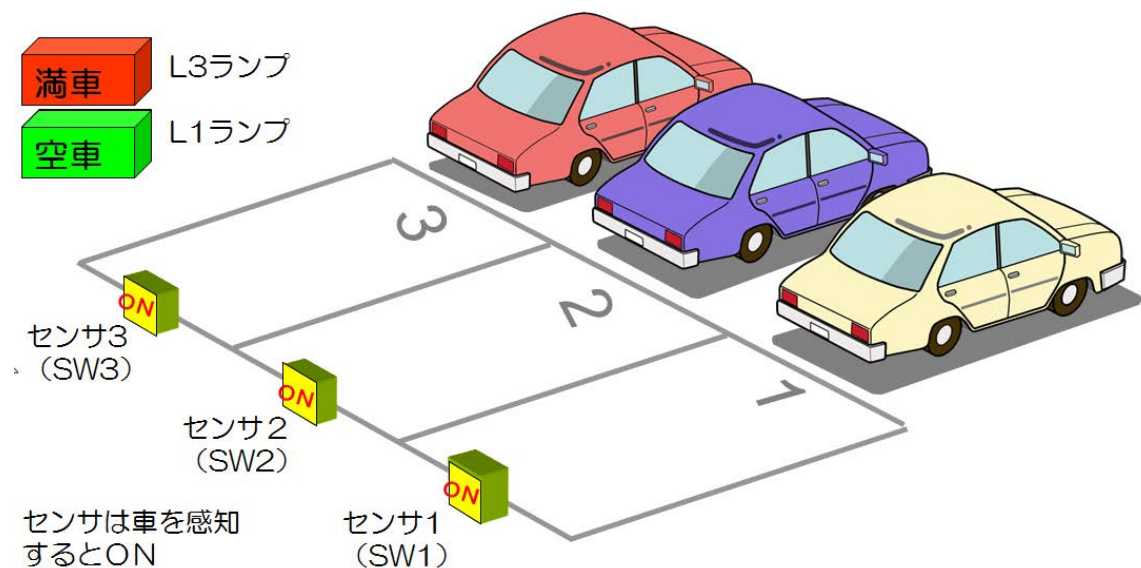
3-5 基本的なラダープログラム例

3-5-1 ラダーの入力練習

【演習】コインパーキング

3 台入庫できる駐車場で、満車であれば満車ランプ(L3 ランプ(赤))を点灯し、1 台でも空車があるときは空車ランプ(L1 ランプ(緑))を点灯させるプログラムを作成しましょう。

なお、車の有無のセンサは、SW1、SW2、SW3 を使用します。

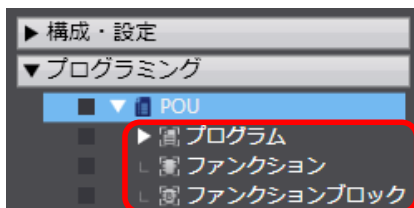


● 完成プログラム例

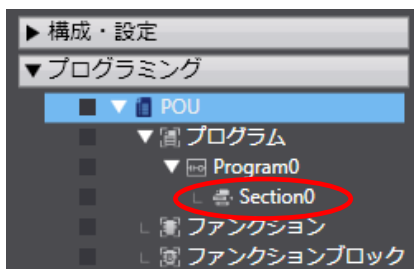


3-5-2 回路の入力練習をしてみましょう。

1. マルチビューエクスプローラから[プログラミング]-[POU]をクリックします。
POU の下に「プログラム」、「ファンクション」、「ファンクションブロック」の項目が表示されます。

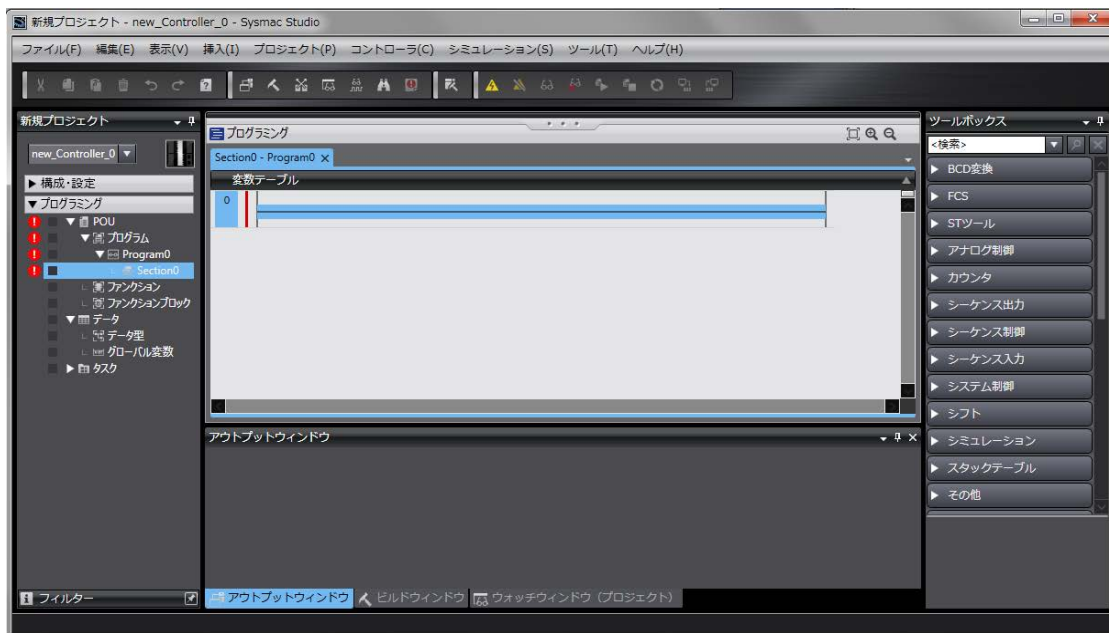


2. すでに登録されているプログラムの下の[Program0]-[Section0]をダブルクリックします。



ラダーエディタが表示されます。

3. ラダーエディタ上でラダープログラムを記述していきます。



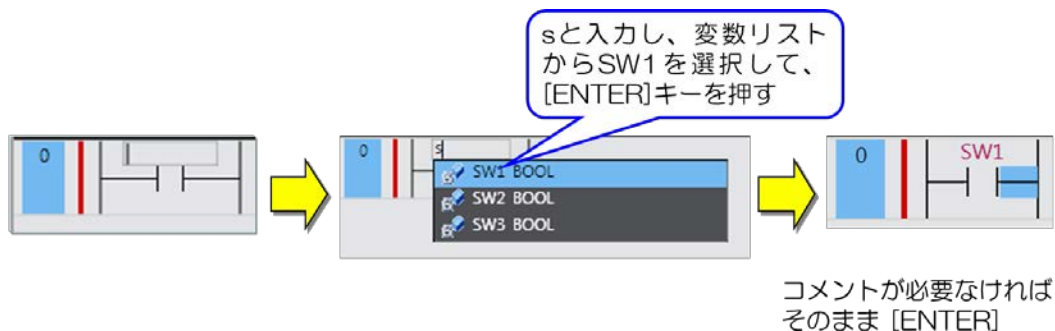
参考

この時マルチビューエクスプローラの Section0 上に赤！マーク(❗)が表示されますが、これはラダープログラムが新規に作成され、プログラムが不完全であることを知らせる文法エラー表示です。プログラムを正しく記述すると消えます。



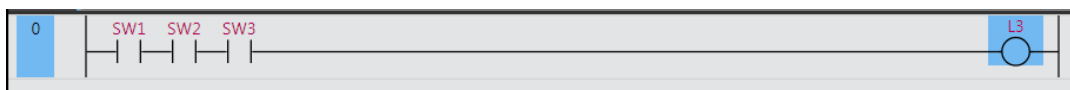
- (1) LD、AND 回路を入力します。
接点を入力し、変数名を入力します

1. ショートカット[C]キーを押すか、右クリックで[接点を挿入]を選択します。



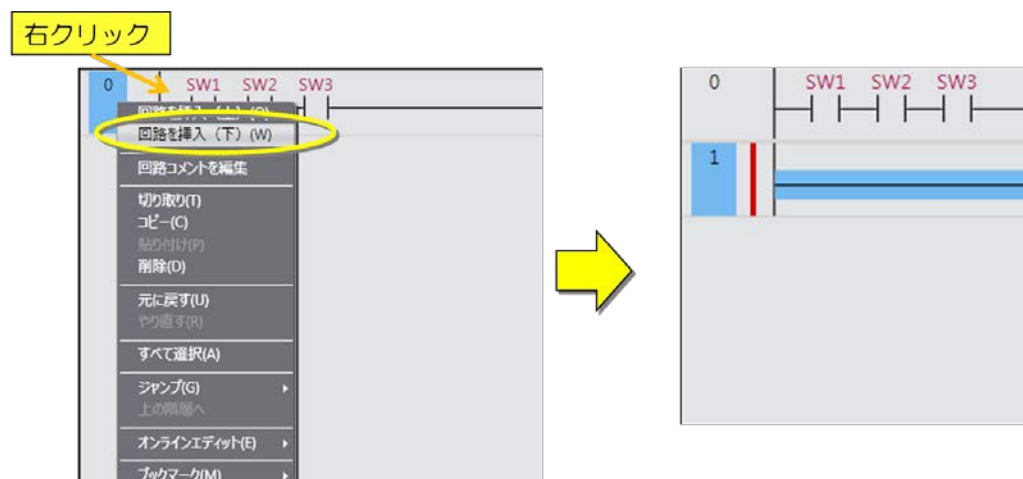
同様にして以下の回路を入力します。

2. 右側のコイルはショートカット[O]キーで入力するか、右クリック[コイルを挿入]を選択します。



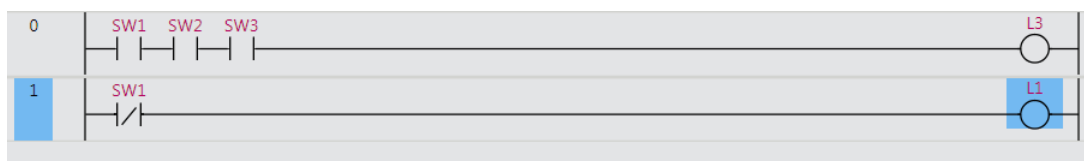
(2) 回路を下に追加します。

1. 回路上でショートカット[R]キーを押すか、左端で右クリック、メニューから[回路を挿入]を選択します。



(3) OR 回路を入力します。

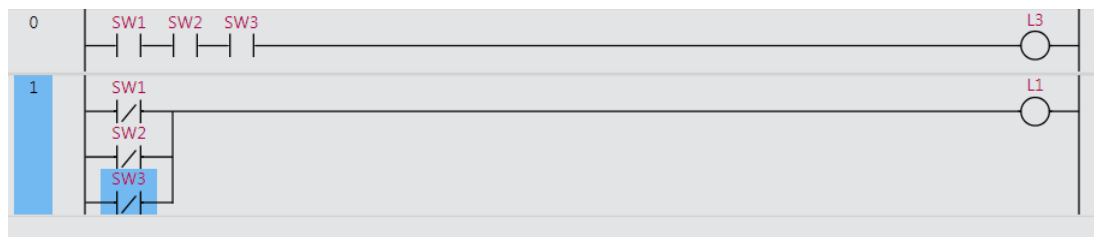
1. b 接点と出力を入力します。b 接点はショートカット[/]キー、または右クリックメニューで b 接点を選択します。



2. b 接点の並列回路(OR 回路)を挿入します。SW1 上でショートカット[X]キーを押すか、または始点から終点までドラッグ&ドロップで線を引き、メニューから b 接点を選択します。



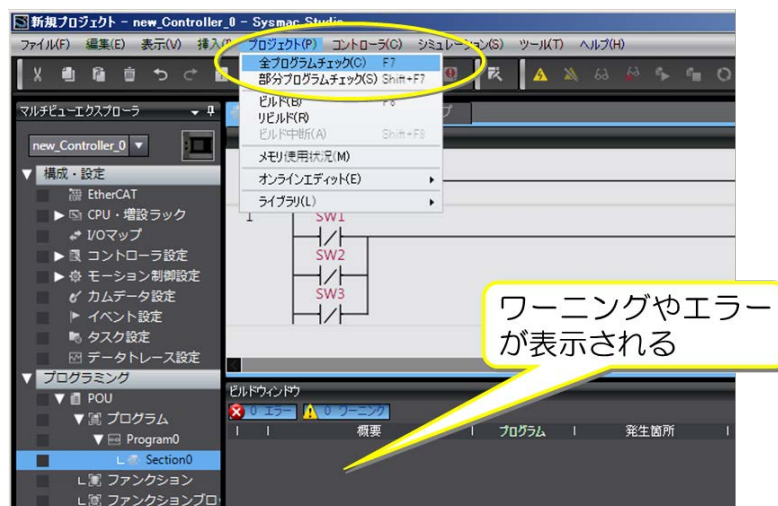
3. 同様に OR 回路を挿入します。



以上でプログラムが完成です。

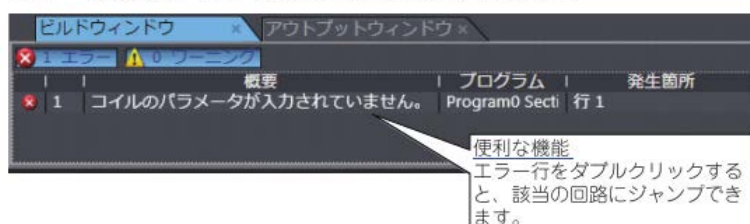
3-5-3 プログラムチェック

1. [プロジェクト]-[全プログラムチェック]を選択します。



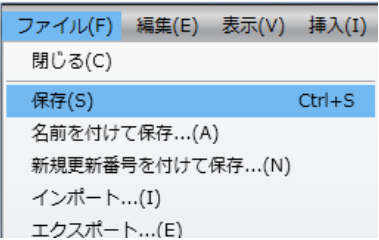
2. エラーがある場合はプログラムを修正します。

プログラムチェック結果がビルドウィンドウに表示されます。
エラーがある場合はプログラムを修正してください。

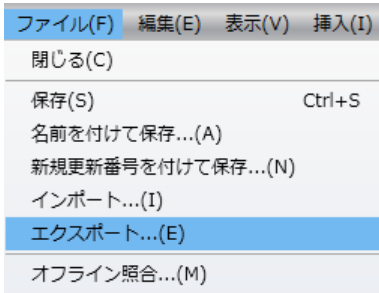


3-5-4 プログラムの保存

1. 次のステップに進む前にプロジェクトファイルの保存とエクスポートを行います。
メニューから「ファイル」-[保存]を選択します。



2. メニューから「ファイル」-「エクスポート」を選択します。ファイル名を入力し、デスクトップ上にエクスポートファイルを作成します。



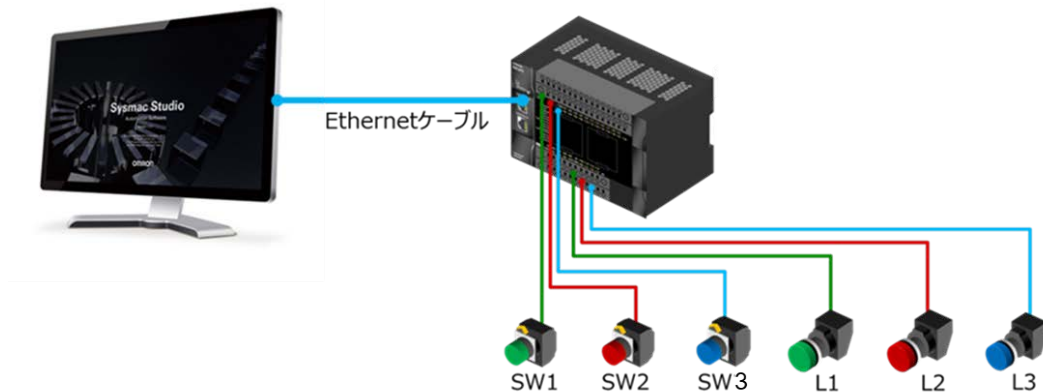
参考 保存とエクスポートの違い

保存	エクスポート
Sysmac Studio に固定で決められているフォルダに保存されます。 ユーザで保存されているフォルダを意識する必要はありません。 保存されたプロジェクトを開くときは、Sysmac Studio 起動時の初期画面から「プロジェクトを開く」で行います	ファイルとして保存する場合に使用します。 保存したファイルを開くときは、ファイル自身をダブルクリックするか、Sysmac Studio 起動時の初期画面から「インポート」で行います。
	

3-5-5 NX1P での動作確認

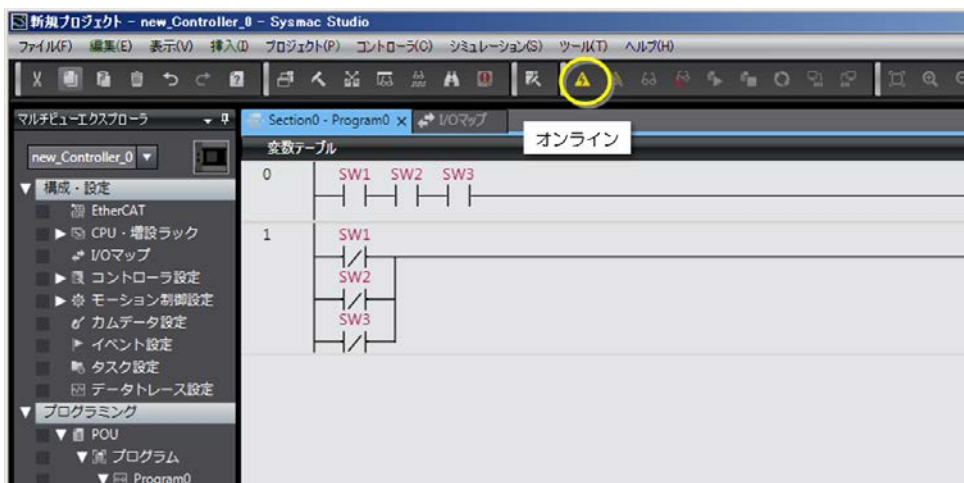
NX1P と接続し、プロジェクト(プログラムを含む全データ)を転送し、動作の確認をします。

1. NX1P と PC(Sysmac Studio)を Ethernet ケーブルで接続します。



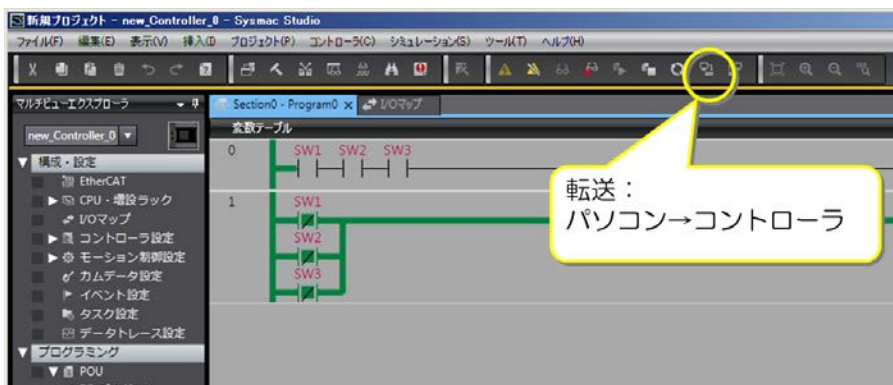
2. オンラインにします。

NX1P と Sysmac Studio をオンラインにします。



3. 転送します。

ツールバーのアイコンで転送ボタンを押します。オンラインにします。



4. 運転モードへ切り替えます。

右下のコントロールステータス表示がプログラムモードになっている場合は運転モードにし、動作をチェックします。



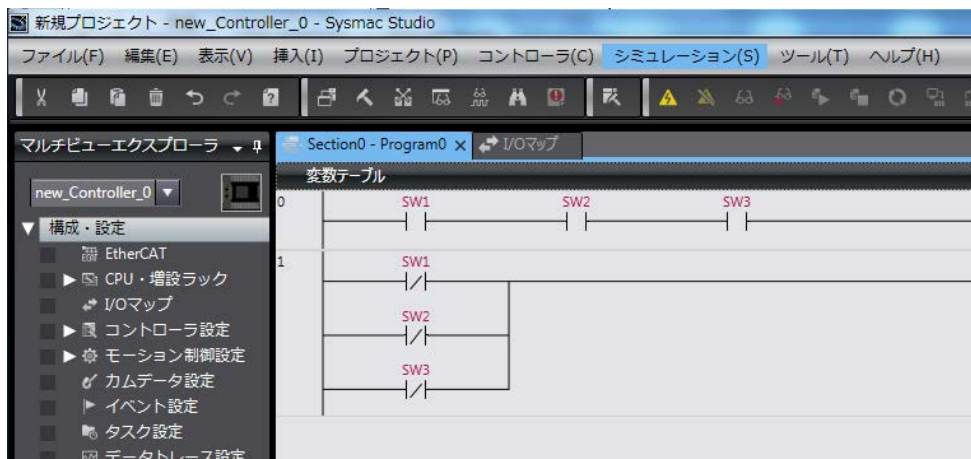
5. 動作を確認します。

SW1、SW2、SW3をオン/オフして、空車(L1)ランプ、空車(L3)ランプが点灯／消灯することを確認してください。

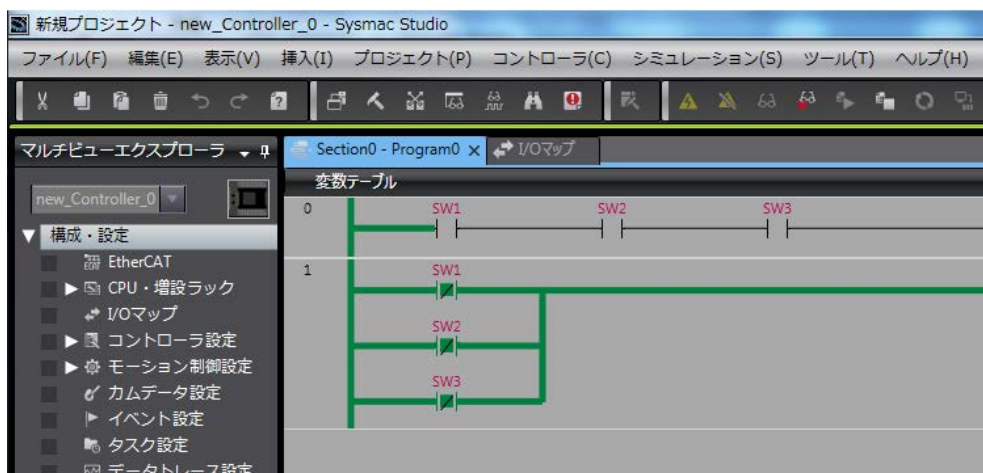
3-5-6 シミュレーションでの動作確認

NX1P 本体がない場合でも、Sysmac Studio のシミュレータ上で動作確認が可能です。(オフラインデバッグ)

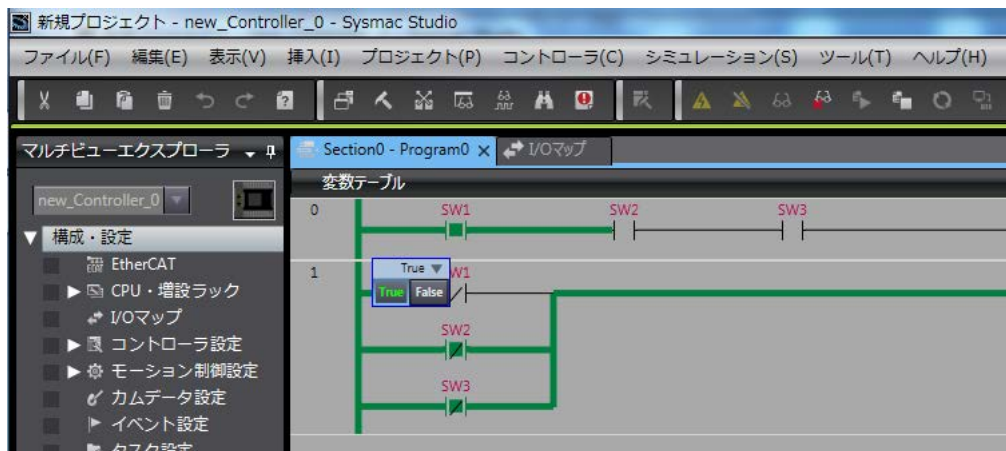
1. 「シミュレーション」-「実行」でシミュレータを起動します。



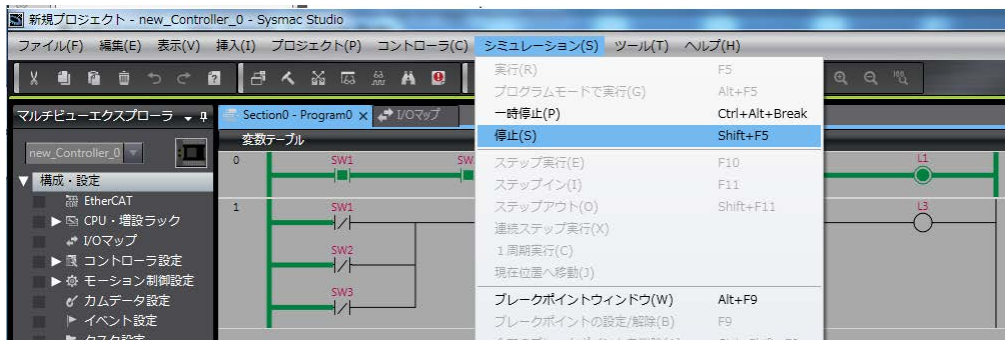
2. 「ビルト中」、「シミュレータ接続中」という表示がしばらく続いたのち、オンラインの状態になります。



3. 接点をダブルクリックすると、接点の True/False メニューが出てきます。接点をプログラム上で ON(True)/OFF(False)をすることができます。SW を押すかわりに、この操作でプログラムの動作を確認できます。



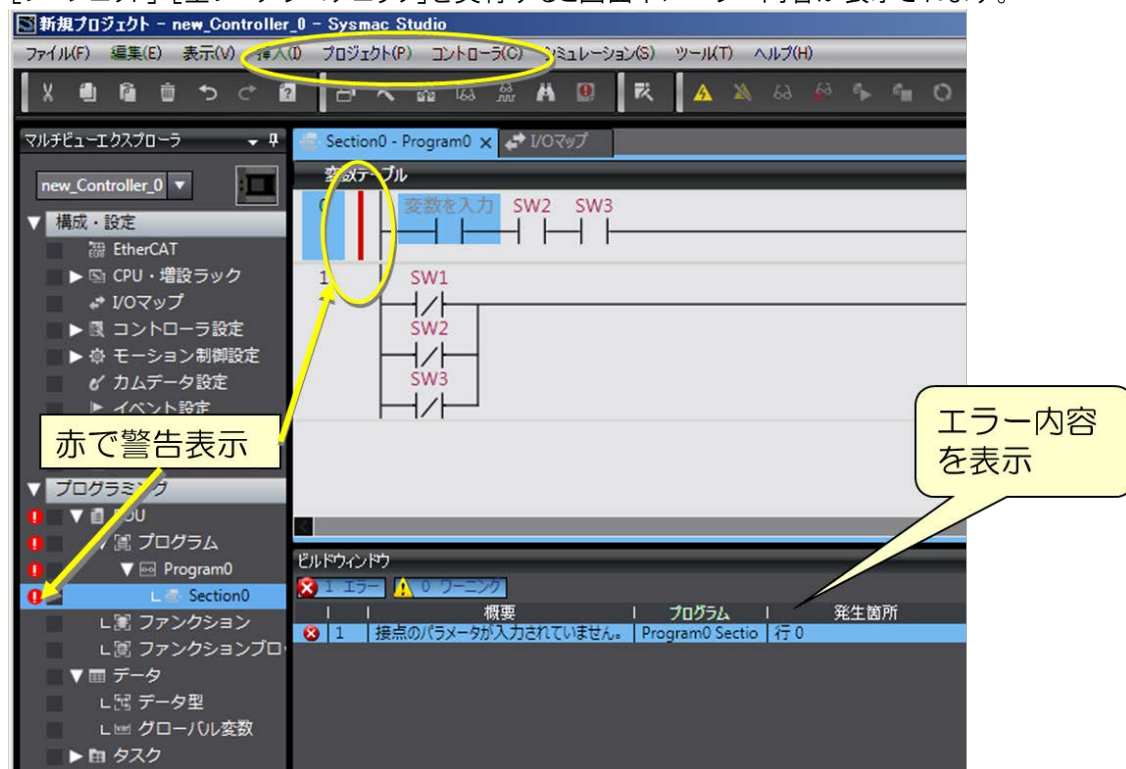
4. 「シミュレーション」-「停止」でシミュレータを停止します。



3-5-7 プログラム作成時のエラーの例(オフライン)

オフラインで、接点の変数名「SW1」を削除します。

[プロジェクト]-[全プログラムチェック]を実行すると画面下にエラー内容が表示されます。

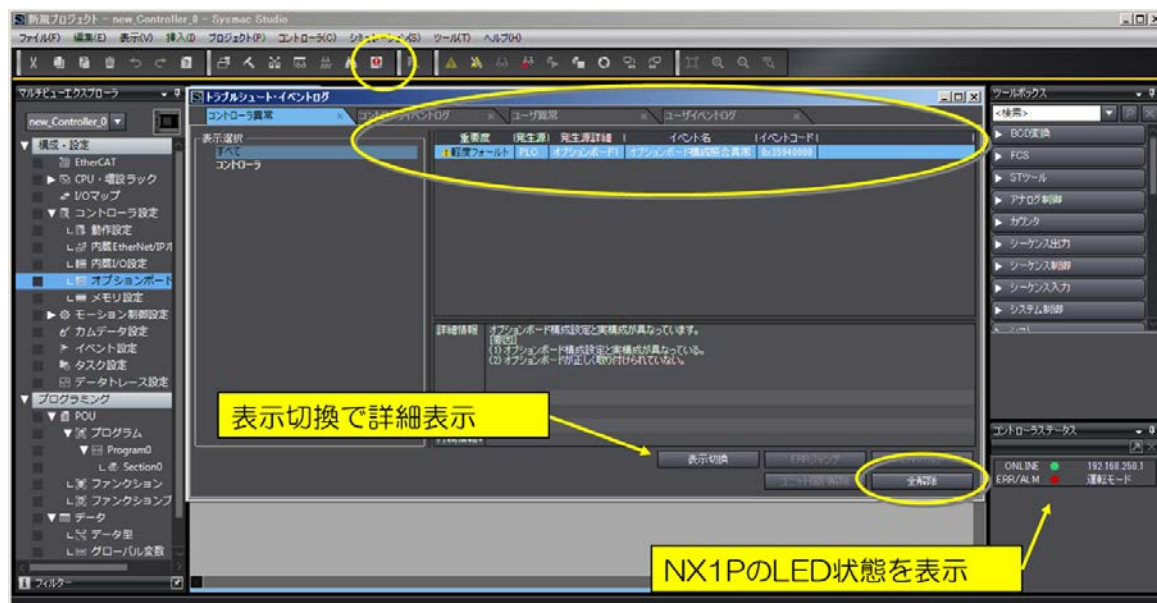


3-5-8 運転時のエラーの例

例えば、次のようなエラーが発生したときは、上部の！マークのトラブルシュートで確認します。

下記の例は、実際にはオプションボード(形 NX1W-CIF01)が装着されていないが、ツールでは装着した設定をしてしまった場合の照合エラーの例を示しています。

* オプションボードの設定は、[構成・設定]-[コントローラ設定]-[オプションボード設定]で行います。



3-6 タイマ命令を使用したラダープログラム例

3-6-1 自己保持回路

はじめに、タイマ命令で使用する、自己保持回路を作成します。

「SW1」ボタンを押すと、「L1」が点灯し続け、「SW2」で消灯する自己保持回路を作成しましょう。

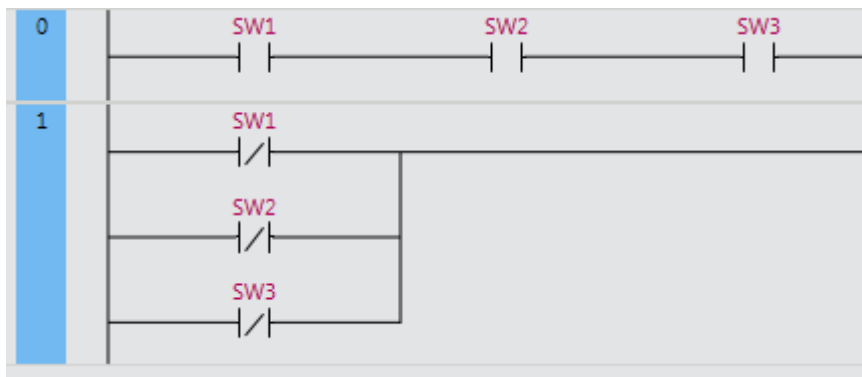


1. プログラム入力はおフラインで行います。



3. 「3-5 基本的なラダープログラム例」で作成したプログラムを削除します。

削除する回路の番号を「Ctrl」キーを押しながら、マウスで右クリックして選択し、「Delete」キーで削除します。

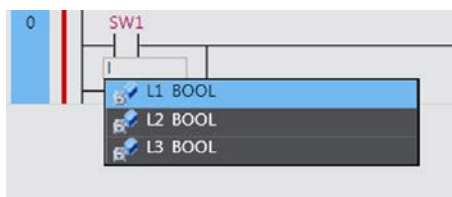


ラダープログラムは削除されますが、I/O マッピングは 3-3-2 I/O マップ設定で設定した状態のままになっています。

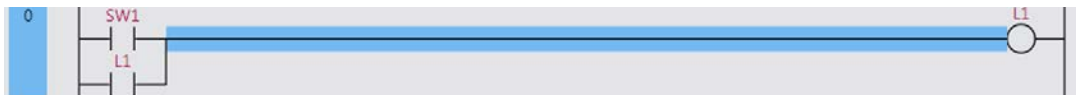
3. 下の回路を作成します。



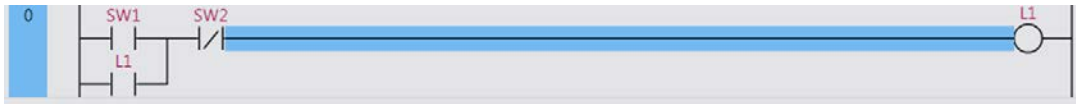
4. 「SW1」の接点をクリックし、[W]キーを押し、L1 を OR 回路で作成します。



5. b 接点を挿入する部分をクリックし、[/]を押し、「SW2」と入力します。下の回路を追加します。



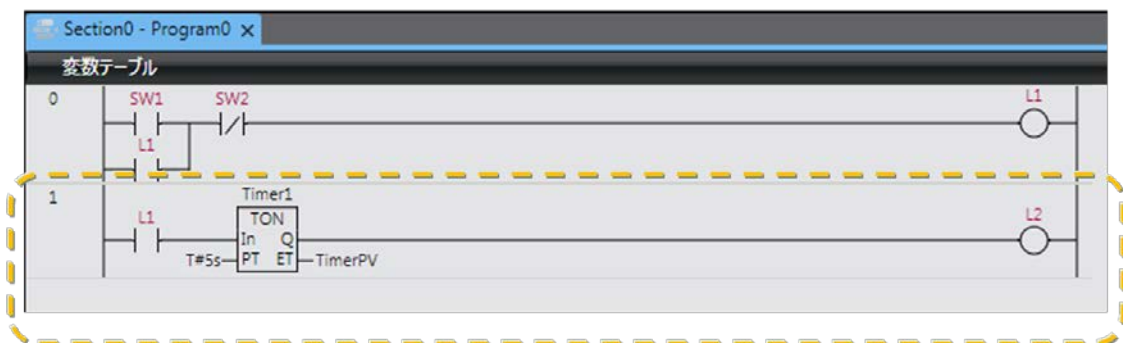
6. 自己保持回路が完成しました。



3-6-2 タイマ命令(TON)

「SW1」を押すと自己保持が働いて 5 秒後に「L2」が点灯する回路を作成しましょう。

追加するプログラム



● (1) ヘルプで TON 命令を確認してみましょう。

・ヘルプ⇒命令語リファレンス⇒タイマ⇒TON

TON

起動から設定時間経過後にTRUEを出力するタイマです。

命令	名称	FB/FUN	グラフィック表現	ST表現
TON	オンディレータイマ	FB		TON_instance (In, PT, Q, ET);

変数

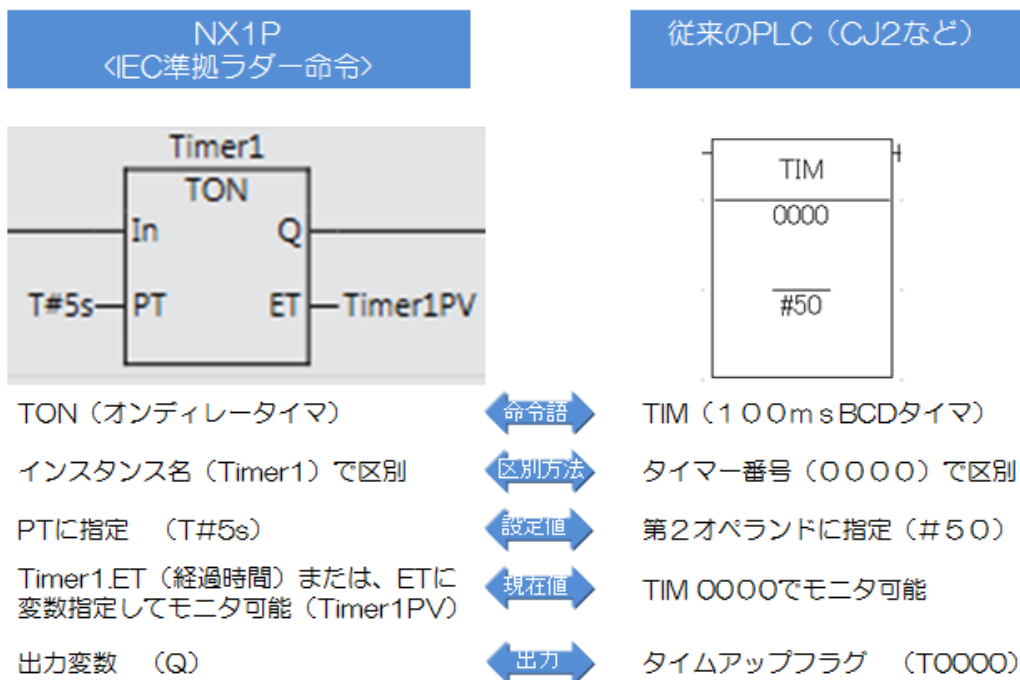
変数	名称	入/出力	内容	有効範囲	単位	初期値
In	タイマ入力	入力	TRUE : タイマ起動指示 FALSE : タイマリセット指示	データ型に依る	-	FALSE
PT	設定時間	入力	タイマ起動から、「Q」がTRUEになるまでの時間	(*)	ms	0
Q	タイマ出力	出力	TRUE : タイマ出力 FALSE : タイマ出力経過後			
ET	経過時間	出力	タイマ起動からの経過時間			

* T#0ms~T#106751d,23h,47m,16s,854,775807ms

ポイント

※命令語の上で、[F1]キーを押しても該当命令のヘルプが表示されます。

NX1P のタイマ(TON)命令と、従来の PLC(CJ2 など)のタイマ(TIM)命令の違い



TON 命令は、タイマ入力(In)が ON している間、時間を測定して、設定時間(PT)で指定した時間が経過するとタイマ出力(Q)が ON します。

タイマ入力(In)が OFF すると、経過時間(ET)やタイマ出力(Q)がリセットされます。

● 2) タイマ設定時間の入力方法

タイマ設定時間は入力パラメータの PT の部分に TIME 型で記述します。

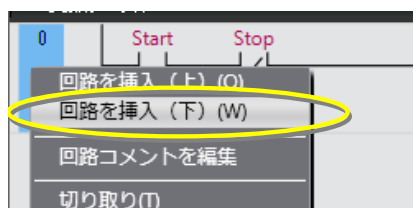
例えば 10.12 秒を設定する場合は T#10.12s と記述します。

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
持続時間 型 ¹⁾	TIME	64 ビット	8 バイト	T#-9223372036854.775808ms (T#-106751d_23h_47m_16s_8 54.775808ms) ~ T#+9223372036854.775807m s (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

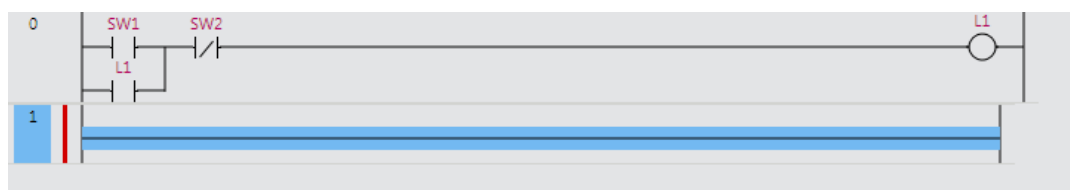
● オンディレイタイマ回路の追加

1. 新しい回路を下に追加します。

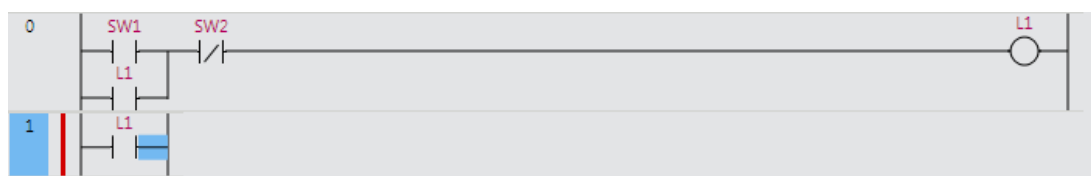
最初の回路の任意の箇所をクリックし、右クリックメニューで、[回路を挿入(下)]を選択、あるいは、[R]キーを押して、新規回路を追加します。



2. 新たに回路を作成する線が追加されます。

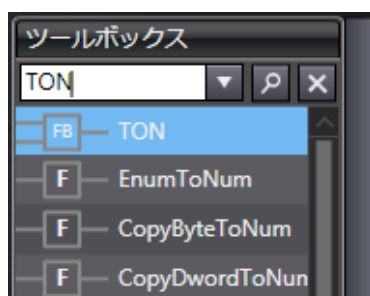


3. 追加した回路に下記のように a 接点「L1」を入力します。



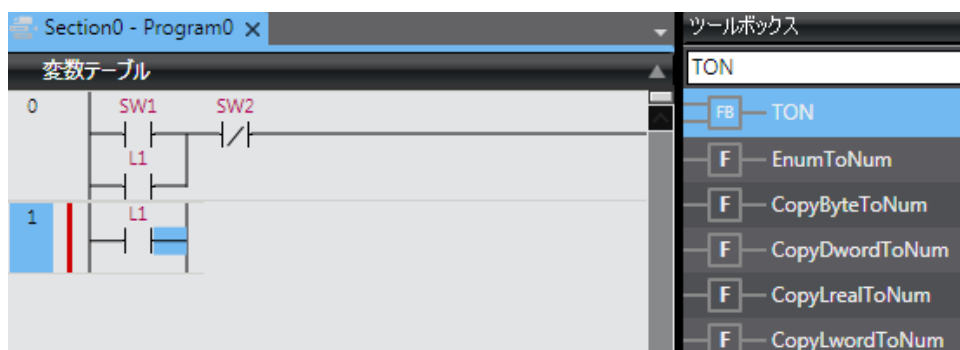
4. 画面右のツールボックスから「TON」命令を検索します。

あるいは、ツールボックスの[▼タイマ]から選択することもできます。

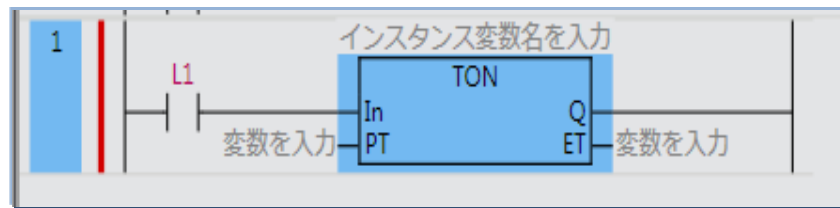


5. ツールボックスから TON 命令をドラッグ & ドロップします。

※もしくは、サブメニューから「ファンクションブロックを挿入」を選択し、「TON」と入力します。

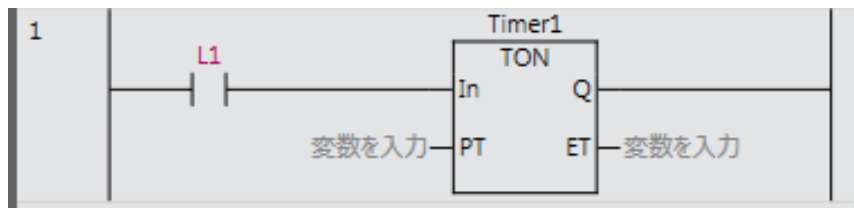


6. TON 命令が挿入されます。



7. TON 命令にインスタンス名をつけます。

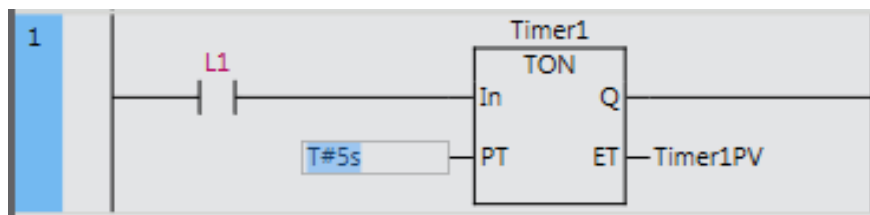
ここでは、「Timer1」とつけます。



8. TON 命令にパラメータを設定します。

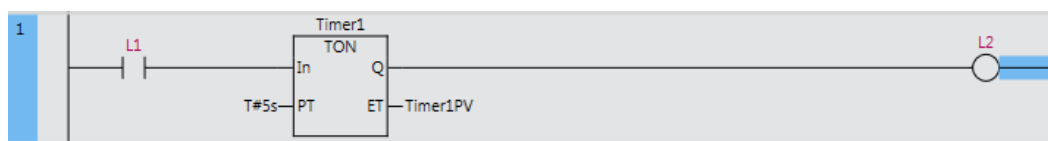
PT:T#5s

ET:Timer1PV



9 Timer1 がタイムアップしたときに ON するコイルを入力します。

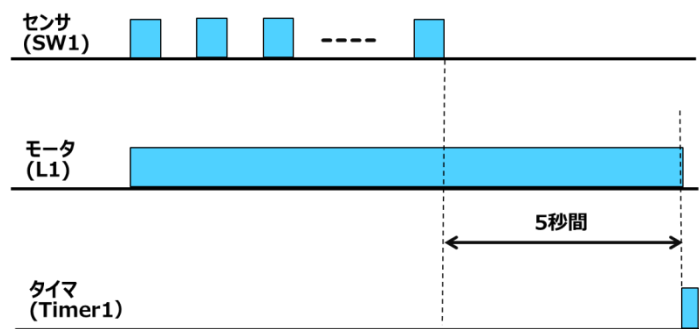
ここでは、コイルの変数名を「L2」とします



3-6-3 演習：省エネルギー対策のエスカレータ

タイマ命令(TON)の動きを理解しましょう。

このエスカレータは人が来るまでは止まっています。人がセンサ(SW1)の前を通過するとモータ(L1)が動き出し、最後の人が通過して5秒後にモータを止めて省エネ運転を行います。



ヒント

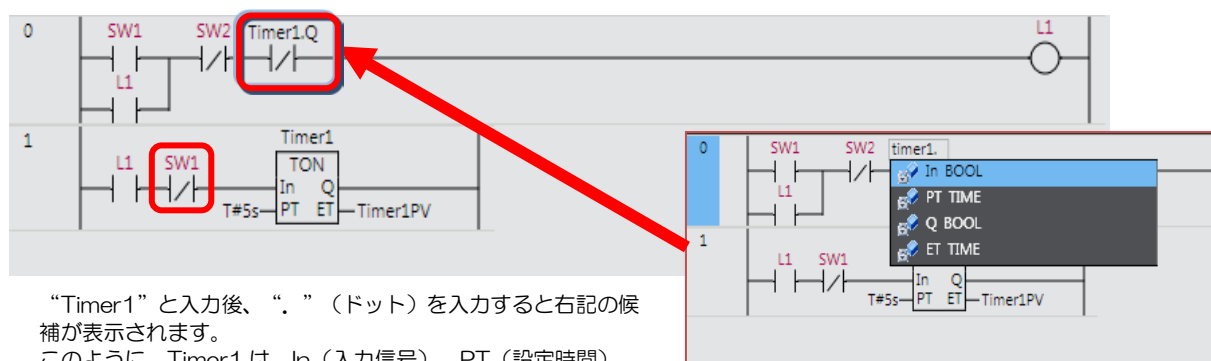
- (1)先ほどの回路を改造しましょう。
- (2)モータを切りたい回路にタイマ出力の b 接点を挿入します。b 接点の方法は2種類あります。
 - ①タイマ出力を内部補助リレー（例えば Timer1Up）で指定する
 - ②タイマ命令の出力状態を表す Timer1.Q を使用する
- (3)人が通過するたびに、タイマがリセットされる工夫をします。

1 回路目に Timer1Up が ON したときにランプ L1(ここではエスカレータのモータ)を停止させるための b 接点を追加します。また、タイマ命令の現在値をリセットするための b 接点(SW1)を追加します。これでプログラムは完成です。

■プログラム例(Timer1 のタイムアップを Timer1UP というコイル)



■プログラム例(別解:Timer1.Q を使用する例)

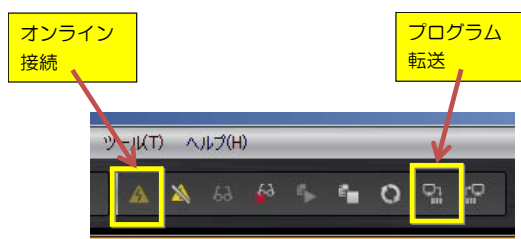


“Timer1”と入力後、“.”（ドット）を入力すると右記の候補が表示されます。
 このように、Timer1 は、In（入力信号）、PT（設定時間）、Q（タイムコアップフラグ）、ET（経過時間）の4つのデータを持っていることが分かります。

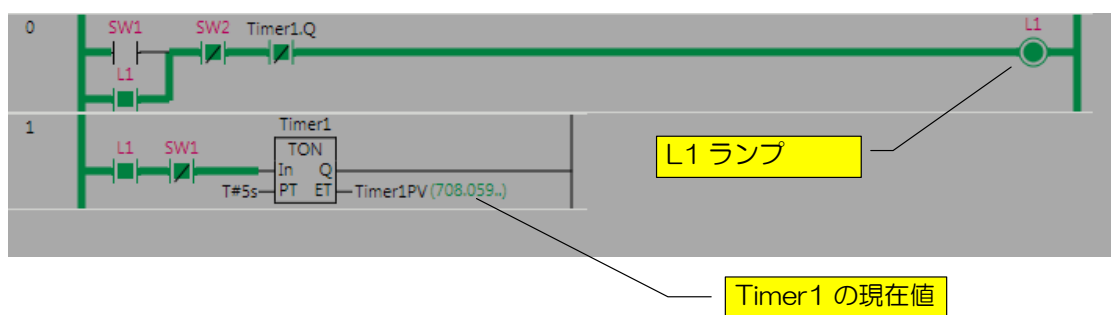
3-6-4 プログラムの動作を確認する

プログラムが正しく動作しているかをラダー導通モニタで確認します。

1. NX1PとPC(Sysmac Studio)をEthernet ケーブルで接続します。
2. オンライン接続で、オンラインにして、[転送(パソコン→コントローラ)]を行います。



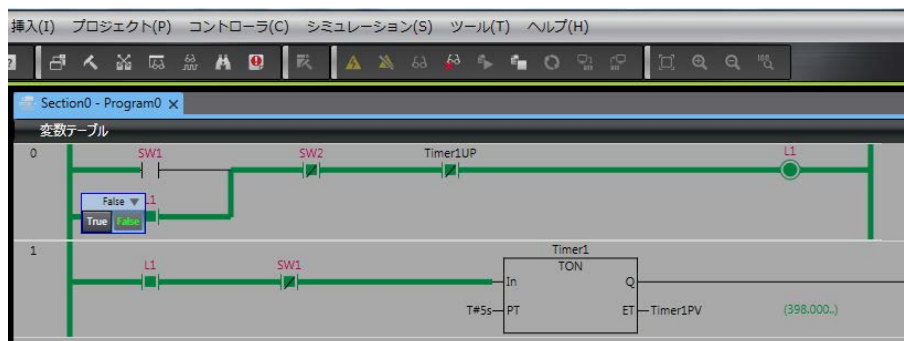
3. SW1(人の通過)を押すとランプ(L1)が点灯し、Timer1 現在値(Timer1PV)が計測を開始します。
4. 再度、SW1 スイッチを押すと、Timer1 現在値がリセットされることを確認します。
5. SW2 を押すとランプ L1 が消灯し、Timer1 現在値がリセットされることを確認します。
 (この場合、SW2 は、エスカレータの停止ボタンの役割になります。)
6. SW1を押して、5 秒間何もしないと自動的に L1 が消灯し Timer1 現在値がリセットされることを確認します。



●シミュレータ機能で動作確認を行う場合

1.「シミュレーション」-「実行」でシミュレータを起動します。

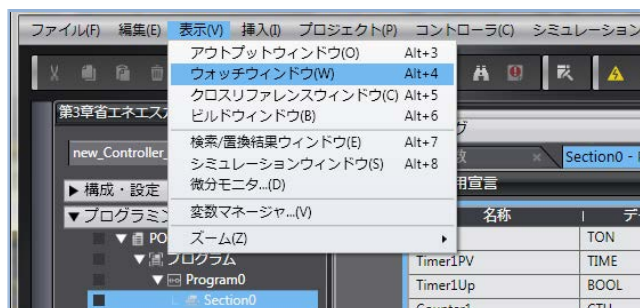
2.接点をダブルクリックすると、接点の True/False メニューが出てきます。SW を押すかわりに、このメニューで接点を ON/OFF して動作を確認できます。



3-6-5 プログラムの動作を確認する(ウォッチウィンドウ)

プログラムが正しく動作しているかをウォッチウィンドウで確認します。
実機でも、シミュレータでも同じ操作で使用できます。

1. 上部メニューから、[表示]-[ウォッチウィンドウ]を選択します。画面下部にウォッチウィンドウが表示されます。

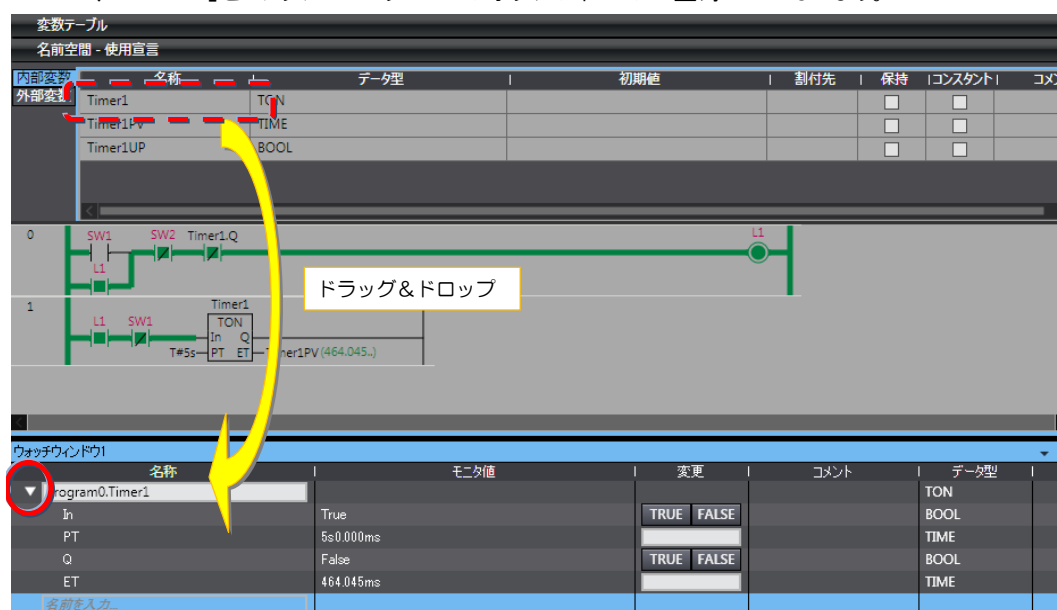


2.「ウォッチウィンドウ 1」をクリックします。



- 画面上部の「変数テーブル」をクリックすると変数テーブルが出てきます。プログラムで使用されている変数(外部変数、内部変数)からモニタしたい変数をウォッチウィンドウ1にドラッグ&ドロップします。

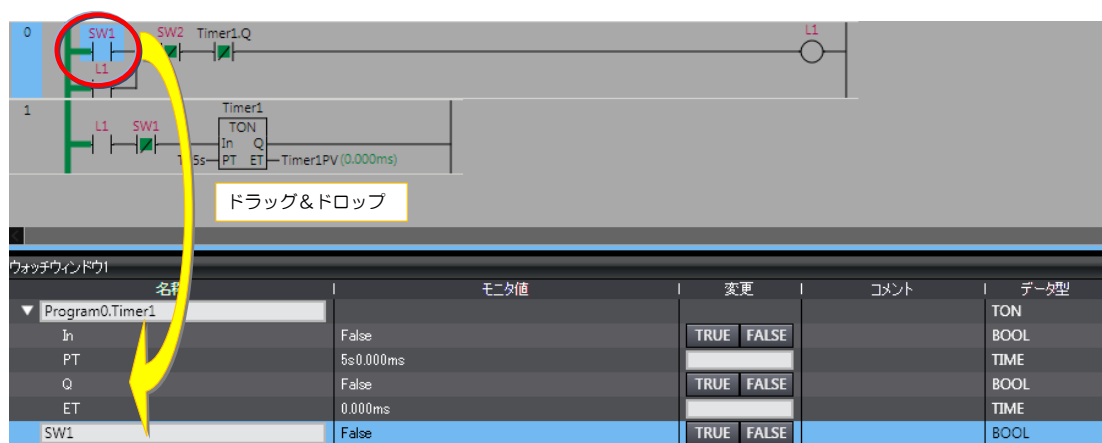
ここでは、「Timer1」をドラッグ&ドロップしてウォッチウィンドウに登録してみましょう。



ドラッグ&ドロップの後、▼マークをクリックすると、Timer1 に含まれるデータ(タイマ起動フラグ(In)、タイマ設定値(PT)、タイマ現在値(ET)、タイムアップフラグ(Q))のモニタができることを確認しましょう。

- 次に、ラダー回路からのドラッグ&ドロップでモニタしたい変数をウォッチウィンドウ1にドラッグ&ドロップします。

ここでは、「SW1」をドラッグ&ドロップしてウォッチウィンドウに登録してみましょう。



- この状態でプログラムを動作させてみましょう。それぞれの値をモニタすることができます。

- 次のステップに進む前にオフラインにし、

プロジェクトファイルの保存とエクスポートを行います。

- ・メニューから[ファイル]-[保存]を選択します。
- ・メニューから[ファイル]-[エクスポート]を選択し、デスクトップ上にエクスポートファイルを作成します。

3-7 時刻や時間を使ったラダープログラム例

3-7-1 NX1P での時刻・時間を使ったプログラミング

例えば、お弁当の消費期限を製造から 30 時間後として印字する場合を考えてみましょう。

そのためには、

- ・製造した時刻を取得する
- ・その時刻と指定された時間(30 時間)を加算し消費期限を算出するプログラムが必要です。



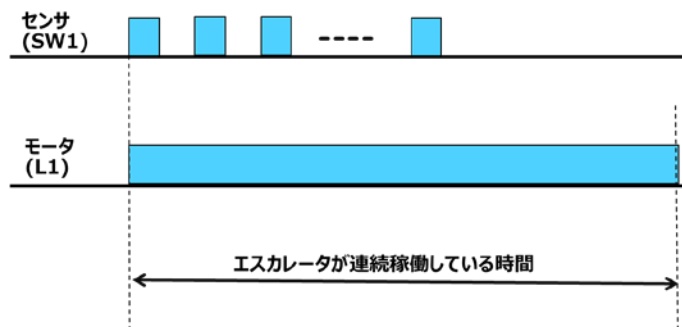
NX1P の変数プログラミングでは、時刻(年、月、日、時、分、秒)・時間のデータ型と命令語で、このような演算を簡単に作成することができます。

3-7-2 演習：エスカレータの連続移動時間

「3-6-3 省エネルギー対策のエスカレータ」演習を使って、時間を計測するプログラムを作成してみましょう。

【演習】 現在時刻の取得と経過時間の算出

エスカレータが連続稼働している時間(「L1」が連続点灯している時間)を計測します。「L1」が点灯した時刻から、「L1」が消灯した時刻を減算する回路を作成しましょう。



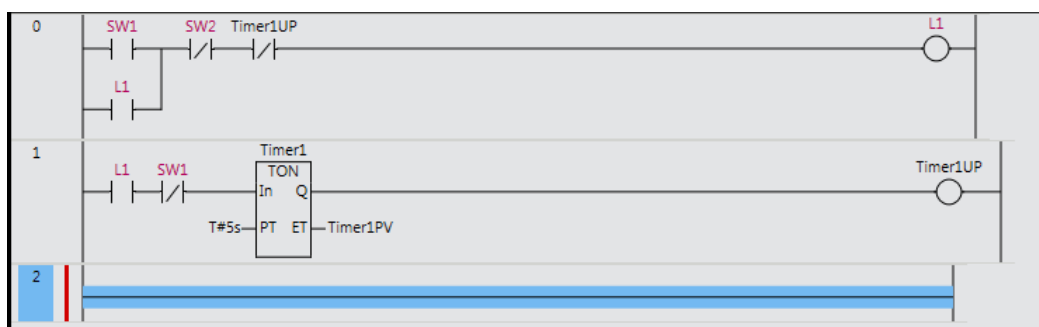
ヒント

- (1) 現在時刻を取得するには、GetTime ファンクションを使用します
- (2) 時刻の減算には、SUB_DT_DT ファンクションを使います。
- (3) SUB_DT_DT ファンクションの戻り値は TIME 型です。

「3-6-3 省エネルギー対策のエスカレータ」演習のプログラムにプログラムを追加します。

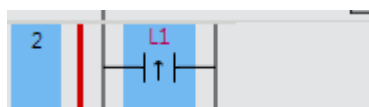
1. 回路を下に追加します

回路1の任意の箇所をクリックし、右クリックメニューで、[回路を挿入(下)]を選択、あるいは、[R]キーを押して、新規回路を追加します。

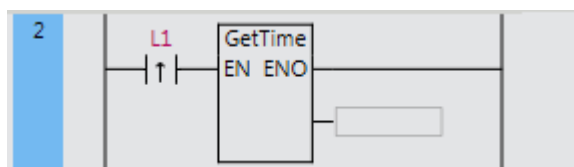


2. L1 の立上り微分を入力します。

ショートカット[C]キーを押すか、右クリックで[接点を挿入]を選択して、接点を入力したところで、[@]キーを押すか、右クリックで「立上り微分」を選択します。

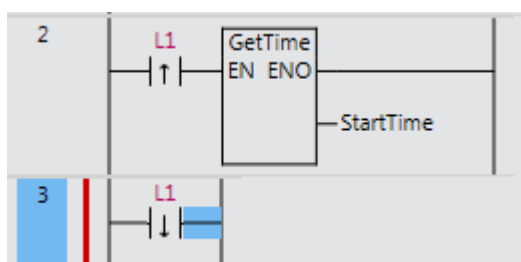


3. L1 が ON になった時点の時刻を取得するために、GetTime ファンクションを入力します。
[I]を押し、FUN 名「GetTime」を入力します。

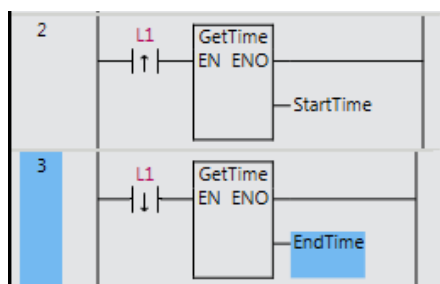


出力変数名を入力します。ここでは、StartTime という変数名にします。

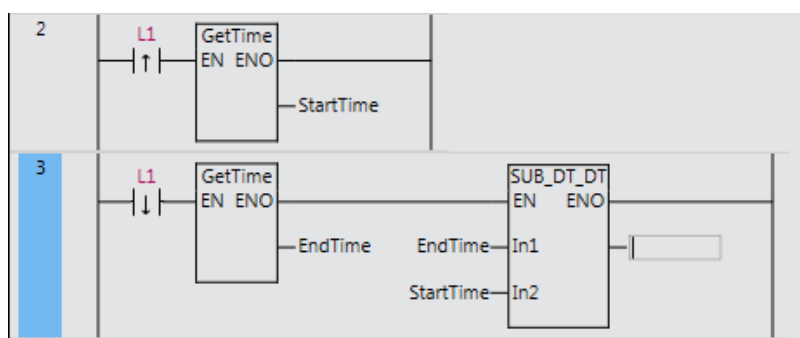
4. 同様に L1 の立下り微分で GetTime ファンクションを実行する回路を作成します。
立下り微分は[%]キーを押すか、右クリックで、「立下り微分」を選択します。



5. GetTime ファンクションを入力し、出力変数名として、EndTime を入力します。



6. EndTime から StartTime を日付時刻演算「SUB_DT_DT」命令で引き算する回路を追加します。



出力変数に LapTime を入力します。

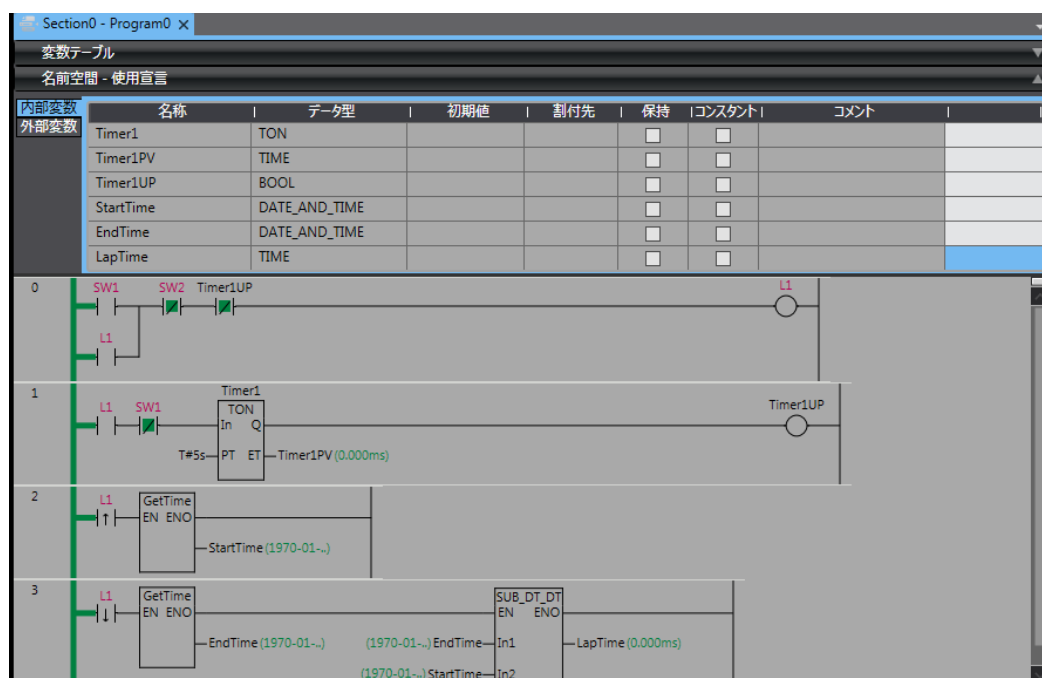
7. 画面上部の「変数テーブル」をクリックし、変数テーブルを確認してみましょう。

名前	データ型	初期値	割付先	保持	コンスタント	コメント
Timer1PV	TIME			☐	☐	
Timer1UP	BOOL			☐	☐	
StartTime	DATE_AND_TIME			☐	☐	
EndTime	DATE_AND_TIME			☐	☐	
LapTime	TIME			☐	☐	

先ほど入力した StartTime、EndTime が DATE_AND_TIME(日付時刻)型として、LapTime が、TIME(時間)型として登録されています。
この場合、データ型は、使用する命令語に応じて自動的に記載されています。

8. プログラムを動作させてみましょう。

NX1P にプログラムを転送するか、Sysmac Studio のシミュレーション機能で運転モードにします。



9. 「ウォッチウィンドウ1」をクリックします。

変数テーブルから、StartTime、EndTime、LapTime を「ウォッチウィンドウ1」にドラッグ&ドロップしてください。

SW1 を ON→OFF して、LapTime の値を確認します。

L1 が ON してから、OFF するまで時間が、LapTime に表示されます。

ドラッグ&ドロップ

名称	モニタ値	変更	コメント	データ型	割付先	表示形式
Program0.StartTime	2017-05-12-17:02:00.			DATE_AND_TIME		String
Program0.EndTime	2017-05-12-17:11:49.			DATE_AND_TIME		String
Program0.LapTime	9m49s2.000ms			TIME		String
名前を入力...						

10. 次のステップに進む前にオフラインにし、プロジェクトファイルの保存とエクスポートを行います。

3-8 開発効率 UP のためのプログラミング基礎知識

3-8-1 POU(Program Organization Unit)

- POU(Program Organization Unit)とは？



国際規格(IEC 61131-3)ではプログラムを構成する単位を POU と呼んでいます。POU の中には、「プログラム(PROGRAM)」、「ファンクション(FUN)」、「ファンクションブロック(FB)」

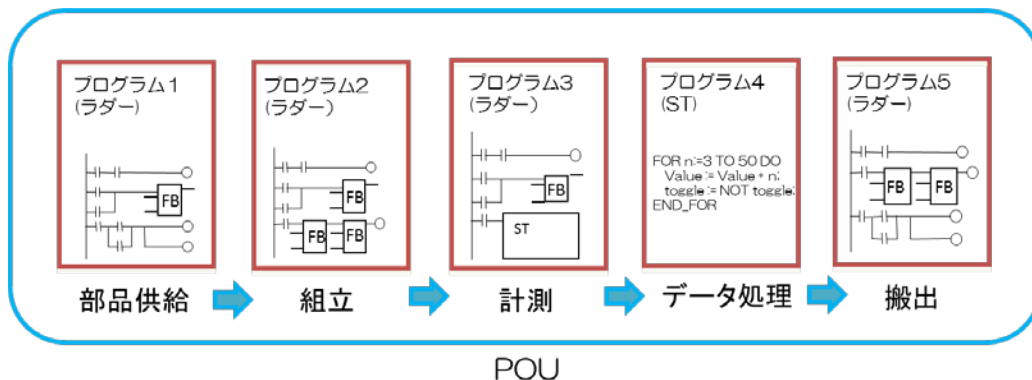
の3つがあります。

ファンクション(FUN)やファンクションブロック(FB)は、繰り返し使用するプログラムを再利用して効率的にプログラムを開発する仕組みです。

3-8-2 プログラムと実行優先順位(タスク)

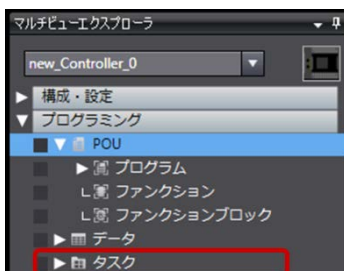
- プログラムとは？

工程ごとにプログラムを分けることにより、可読性や再利用性が向上します。



プログラムごとに使用する言語をラダーとSTを選択できるので、工程ごとに最適な言語を選択できます。ラダーを選択しても、ラダープログラムの中にST言語を使用することができます(インライン ST)。

- タスクとは？

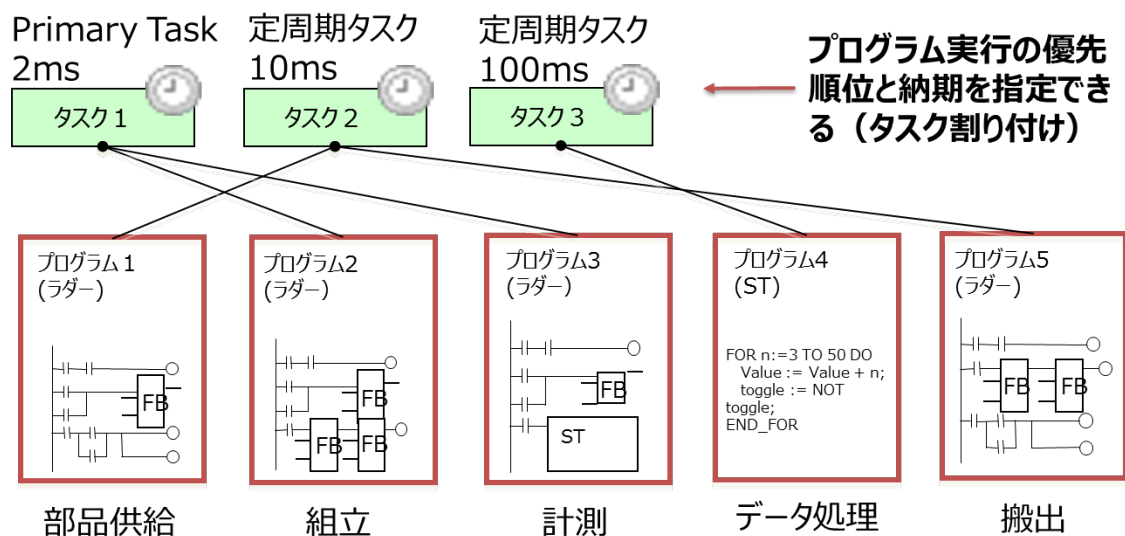


タスクは、各プログラムに対する実行の方法です。プログラムごとに、制御周期を設定することができます。

高速処理が必要な制御は、周期の速いプライマリタスク(Primary Task)を割り付けることにより、確実な高速処理を実現できます。

高速処理が必要ない制御は、遅い周期を割り付けることで、NX1P 処理の負荷を低減できます。

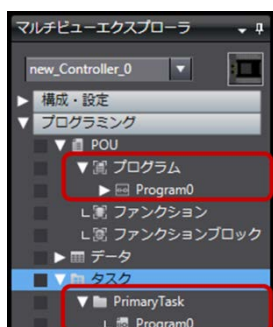
1つのタスクに複数のプログラムを割り付けることもできます。



- プログラムとタスクの初期設定とプログラムの追加の方法

1. Sysmac Studioで新規プロジェクトの作成を行う場合、プログラムには、Program0（ラダー言語）がすでに作成されています。また、タスクには、Program0は、プライマリ定周期タスク(PrimaryTask)が設定されています。

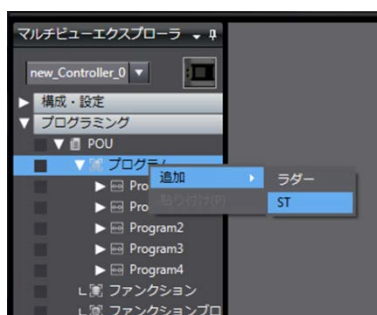
プログラムを新規で作成する場合は、Program0にプログラムを作成することで、プログラムの作成やタスクの設定を意識しなくても動作できます。



2. プログラムを追加する場合

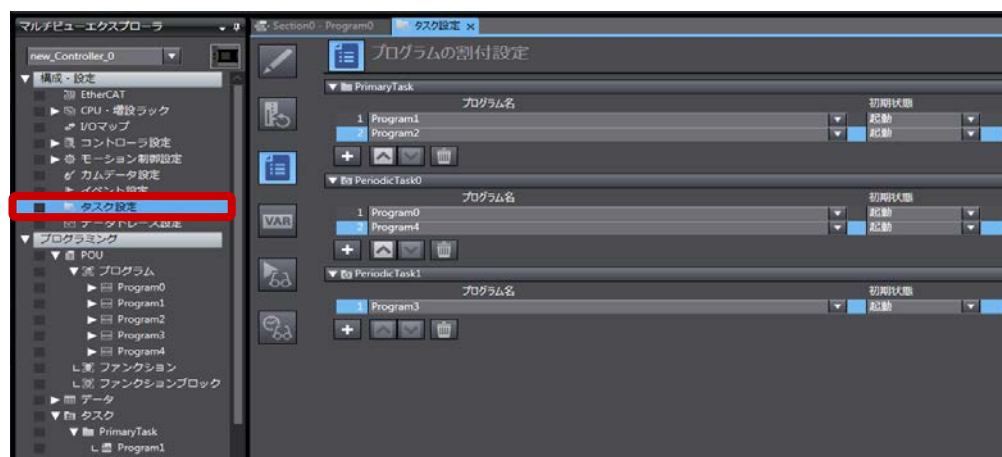
「プログラム」を右クリックして、「追加」で、「ラダー」か「ST」を選択します。

プログラムは、Program1, Program2 など、順番に作成されます。プログラムの名称を変更する場合は、Program*で右クリックをして、「名前の変更」で工程名などの名称にします。



3. 追加したプログラムにタスクを設定する場合

「構成・設定」-「タスク設定」で、「タスクの設定」と「プログラムの割付設定」によりタスクとプログラムの設定を行います。



3-8-3 ファンクション(FUN)とファンクションブロック(FB)

ファンクション FUN:Function やファンクションブロック FB:Function Block は、プログラムで使用する、命令語です。

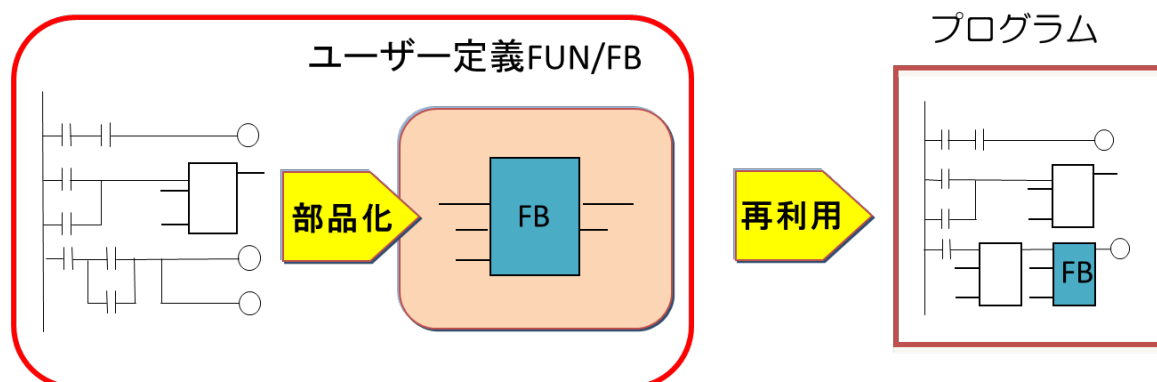
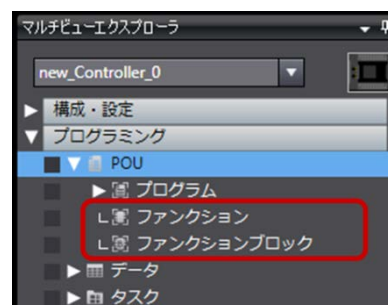
従来のPLC(CJ2 など)では、応用命令と呼ばれていたものです。(例:MOV 命令…FUN、TIM 命令…FB)

NX1P が標準で準備している FUN/FB 以外に、ユーザ自身で FUN/FB を作成することができます(ユーザ定義の FUN/FB)。

● ユーザ定義の FUN/FB

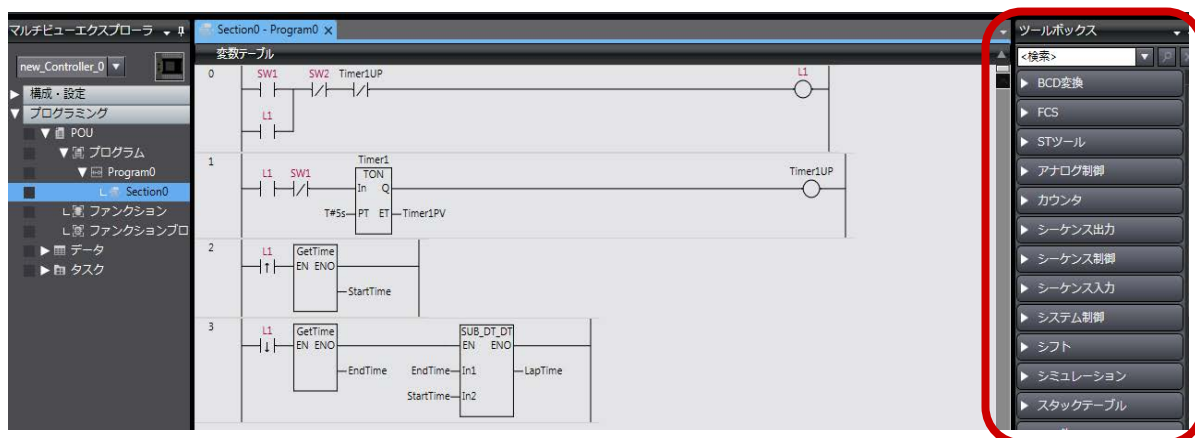
過去に作ったプログラムで他のプログラムでも繰り返し使用するプログラムを、ファンクションブロック FB やファンクション FUN として部品化することができます。

新規プログラムを作成時に、そのファンクションブロック FB やファンクション FUN を使用することで、プログラムの開発効率を向上することができます。

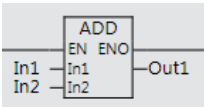
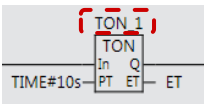


● NX1P で準備している FUN/FB

NX1P で準備している FUN/FB は、画面右側のツールボックスに一覧があります。
命令語をドラッグ&ドロップして、プログラミング回路に使用することができます。

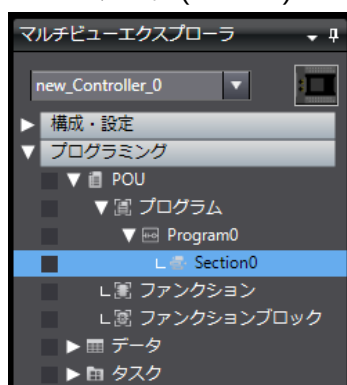


● ファンクション(FUN)とファンクションブロック(FB)の主な違い

	概要	命令語の形状	例
ファンクション(FUN)	単純な演算など、内部状態をもたない命令です。		ビット列型ファンクション(AND, OR, XOR, NOT) 数値計算ファンクション(ADD, SUB, MUL, DIV, SQRT, LN, LOG, EXP, SIN, COS, TAN) 比較ファンクション(GT, GE, EQ, LT, LE, NE)など
ファンクションブロック(FB)	タイマ命令など、命令実行後も条件などが完了するまで内部状態を記憶する必要がある命令です。	 名前が個別に必要です。(インスタンス名)	セット、リセット(SR, RS, ...) エッジ検出(微分)(R_TRIG, F_TRIG) カウンタ(CTU, CTD, CTUD) タイマ(TP, TON, TOF, ...) モーションファンクションブロック(MC_HOME, MC_MOVE)など

3-8-4 セクション

● セクション(Section)とは？



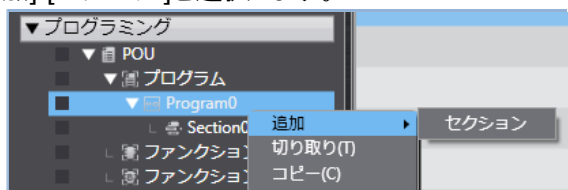
プログラムの複数回路ごとに、セクションで分割して、名称をつけることができます。プログラム内の回路を機能や動作ごとに名称をつけることで、プログラムの構成が見やすくなります。また、セクションごとに移動や削除することもできます。

プログラムは表示されるセクションの並び順に従って上から下へ順に実行されます。実行順序を変更するにはセクションの順序を変更します。

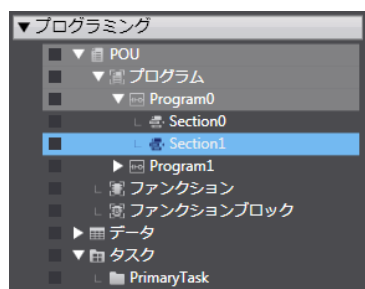
初期状態で Program0 には、Section0 が作成されています。

- セクションの追加の方法

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]-[Program0]を右クリックします。
[追加]-[セクション]を選択します。



2. [Program0]項目の下に[Section1]が追加されます。



3-8-5 変数の種類

- グローバル変数とは？

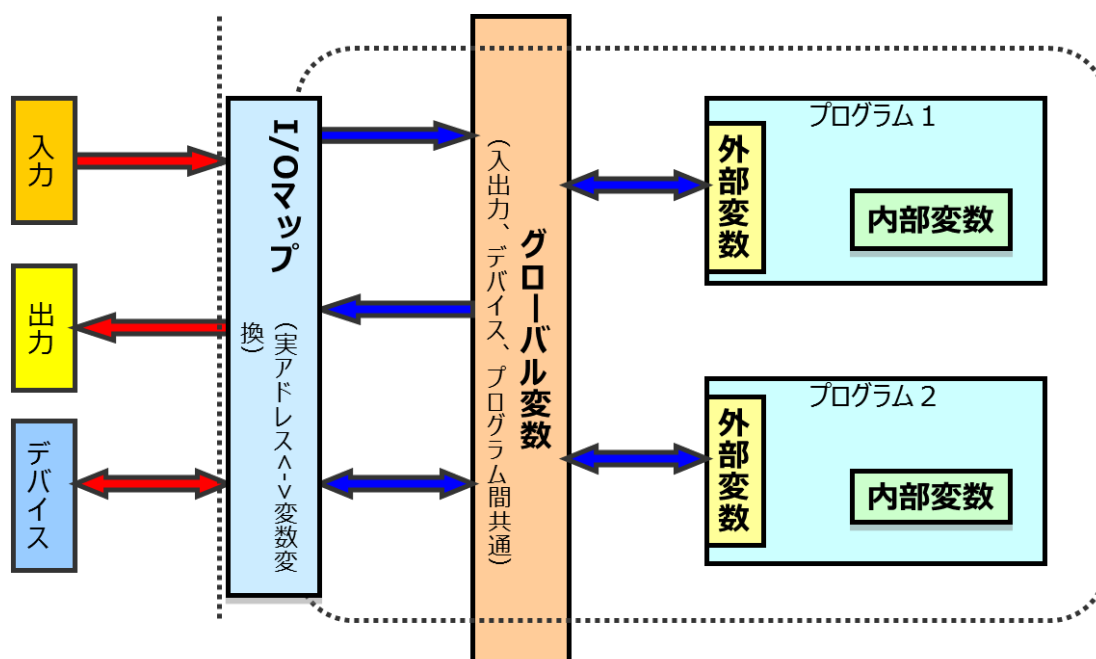
I/O マップで登録された変数やプログラム間で共通に使用する変数で、すべてのプログラムからアクセスできる変数です。

- 外部変数／内部変数とは？

外部変数／内部変数はプログラムの一部としてプログラムごとに管理される変数です。

外部変数は、プログラムでグローバル変数(SW1 や L1 など)を使用すると外部変数に登録されます。

内部変数は、そのプログラムで新たに登録した変数(Timer1 や Timer1PV など)で、内部変数に登録されます。



- グローバル変数の確認

「3-7 時刻や時間を使ったラダープログラム」例でグローバル変数の確認を行います
マルチビューエクスプローラの「グローバル変数」をクリックします。



I/O マップに登録した SW1 や L1 などの変数は、自動的にグローバル変数にも登録されていることを確認します。

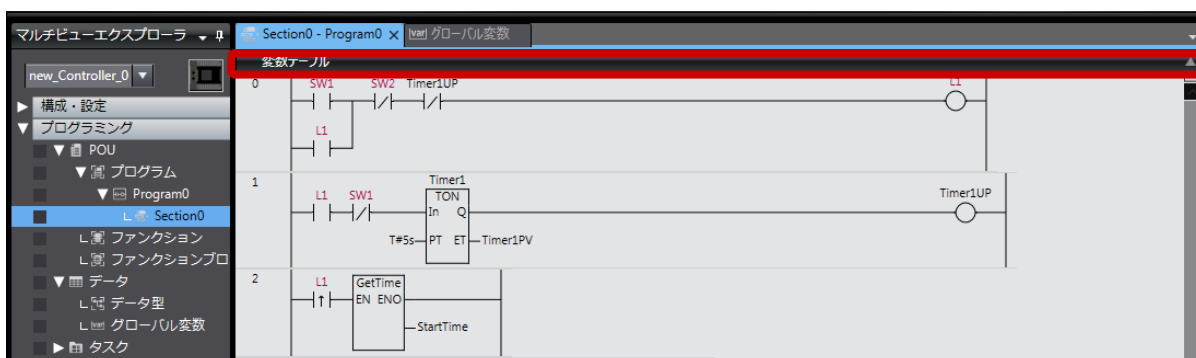


参考

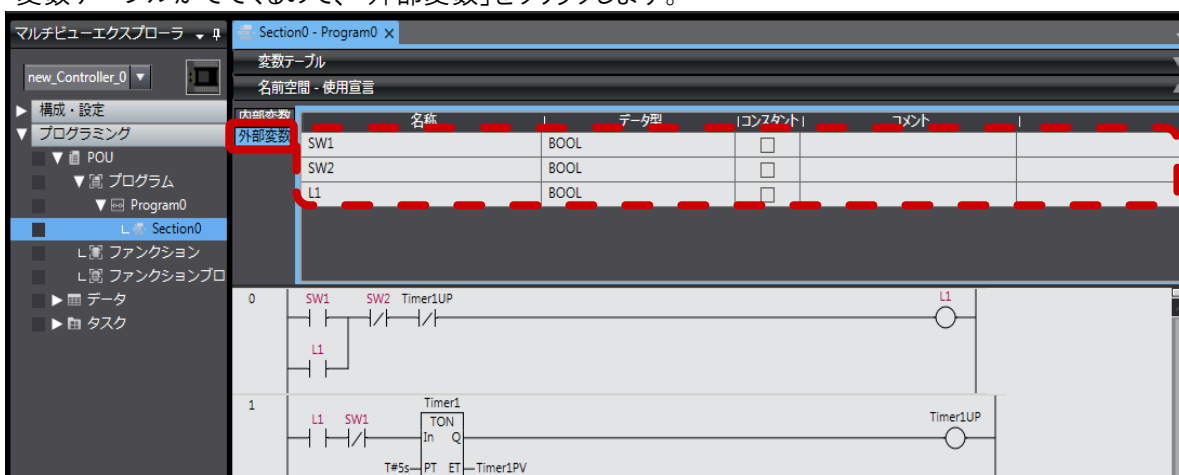
グローバル変数は紫色で表示されます。

- 変数テーブルの確認

プログラムのエディットウィンドウ上部の「変数テーブル」をクリックします。

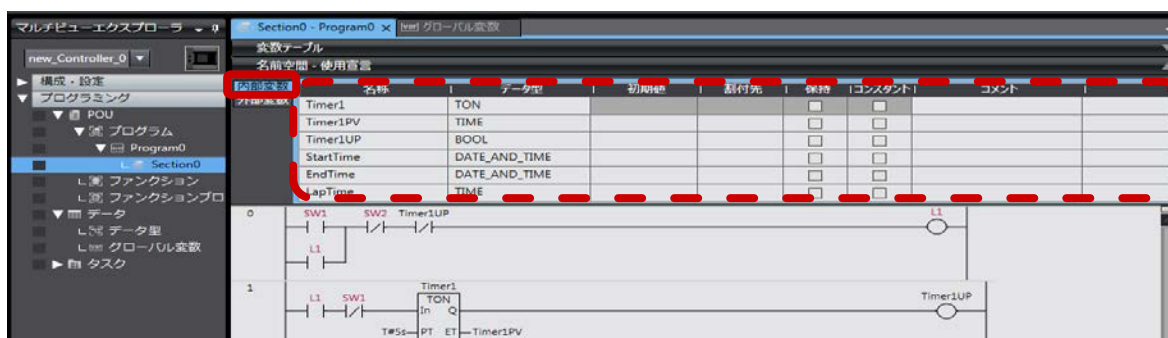


変数テーブルがでてくるので、「外部変数」をクリックします。



I/O マップで登録した変数(グローバル変数)をプログラムで使用すると、自動的に外部変数に登録されています。

「内部変数」をクリックします。



このプログラムで新たに使用した変数は、内部変数として自動的に登録されています。

4

4 データを扱うプログラミング

この章では、データを扱うプログラミングを説明します。

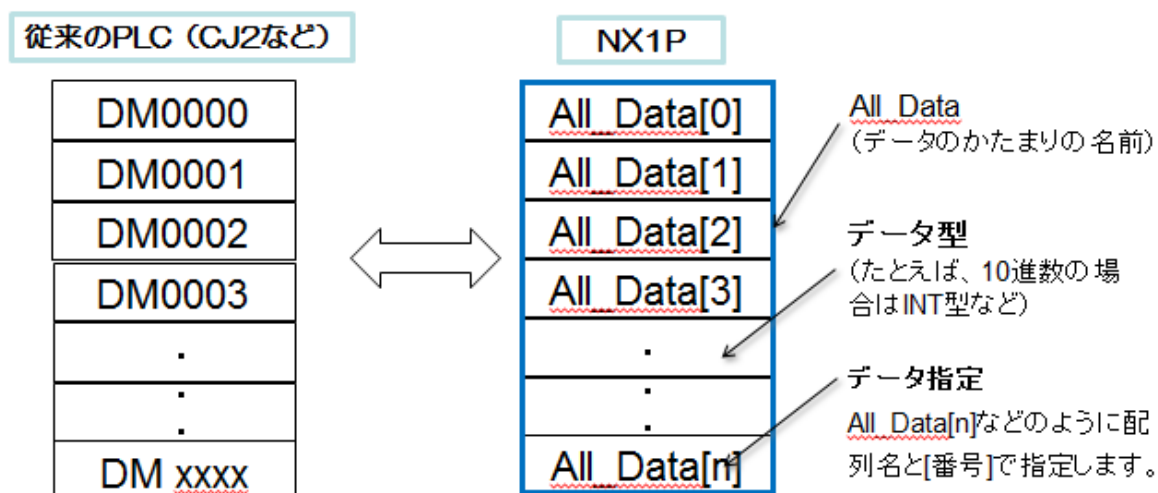
4-1	データ処理に使用する変数	4-74
4-1-1	配列	4-74
4-2	プログラミング実習	4-75
4-2-1	アプリケーション例	4-75
4-2-2	プログラミングの作成	4-75
4-2-3	新規プロジェクトの作成	4-76
4-2-4	アナログオプションボード設定	4-77
4-2-5	アナログ入力と入力端子に変数を割り付ける	4-77
4-2-6	プログラム例	4-78
4-2-7	配列の設定方法	4-79
4-2-8	プログラムの入力	4-80
4-2-9	プログラムの動作確認	4-81
4-2-10	配列データの参照の仕方	4-82

4-1 データ処理に使用する変数

4-1-1 配列

従来の PLC (CJ2 など) では、データの演算や収集を行うプログラムのメモリ領域としてデータメモリ (DM) を使用していました。

NX1P ではデータメモリというメモリエリアはなく、接点と同様、変数をデータ処理に使用するメモリとして使用します。



- 配列

上記の All_Data[n] を、配列といいます。

かたまり全体の名前(変数名)の後に[番号]をつけて、複数のメモリ領域として使用します。

プログラムで使用する場合は、変数[番号](例えば、All_Data[3])を1つの変数として使用します。

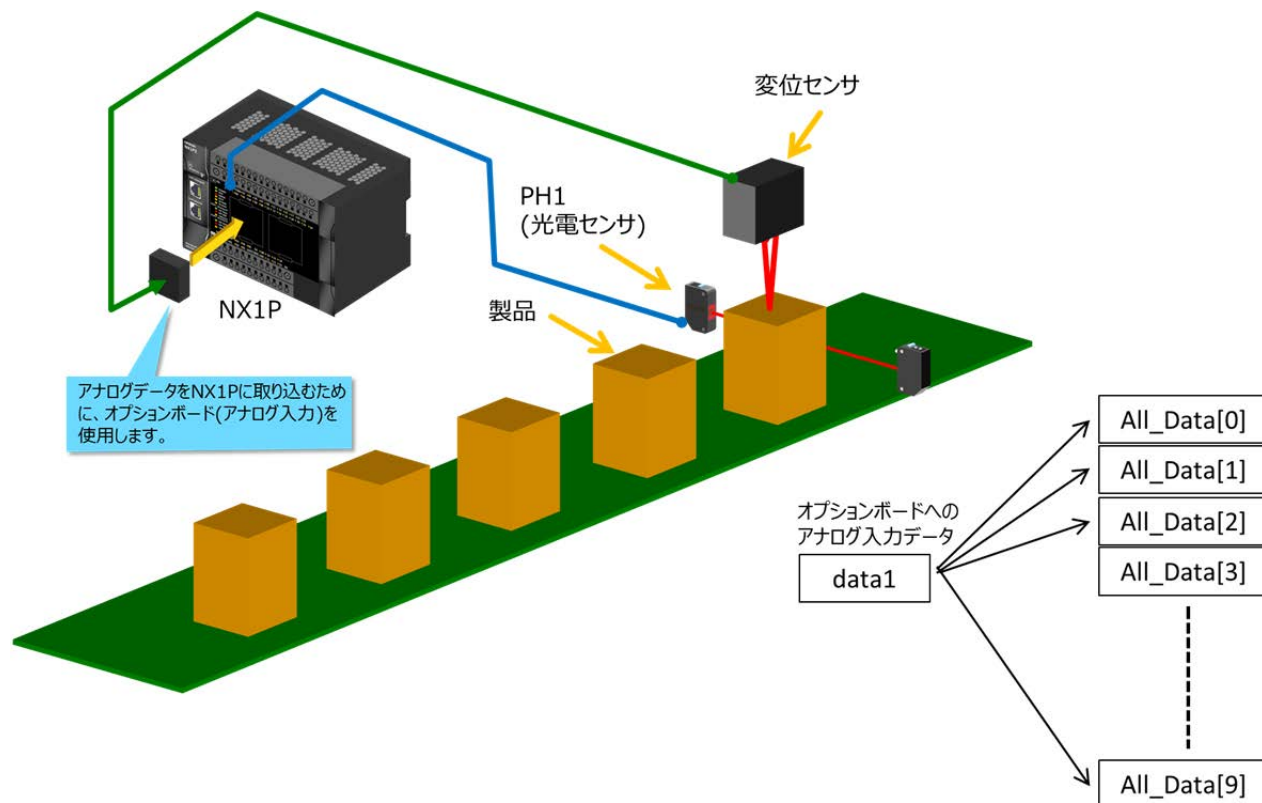
配列のデータ型は1種類のみ指定できます。

1つの名前で複数の変数を使用できるので、プログラム上での使用用途がわかりやすくなり、可読性が向上します。

4-2 プログラミング実習

4-2-1 アプリケーション例

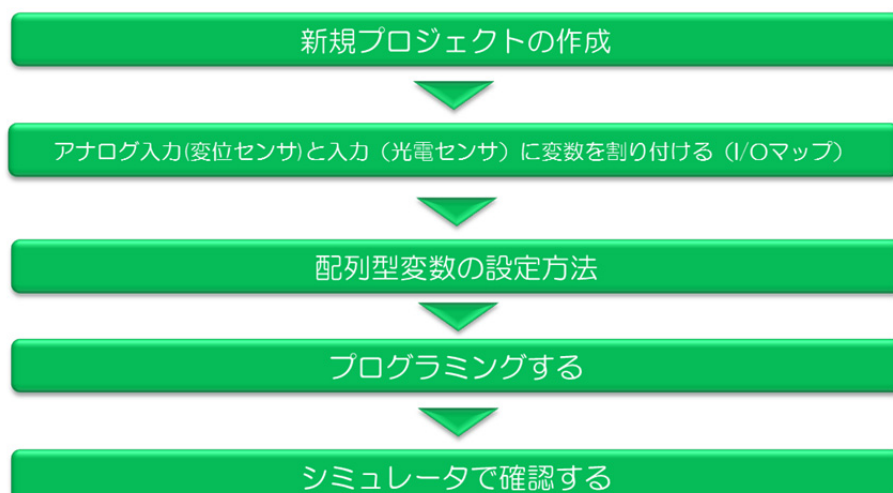
光電センサ(PH1)がONすると、変位センサで計測したデータを10個の配列の先頭から順番に格納されていくプログラムを考えてみましょう。



ここでは、変位センサで計測したデータをアナログデータとして「data1」という変数に格納し、そのデータを配列 `All_Data[0]`、`All_Data[1]`、.....と順番に格納していきます。

4-2-2 プログラミングの作成

この章では、実機を使わず、Sysmac Studioのシミュレータ機能を使って、動作確認を行う例を紹介します。

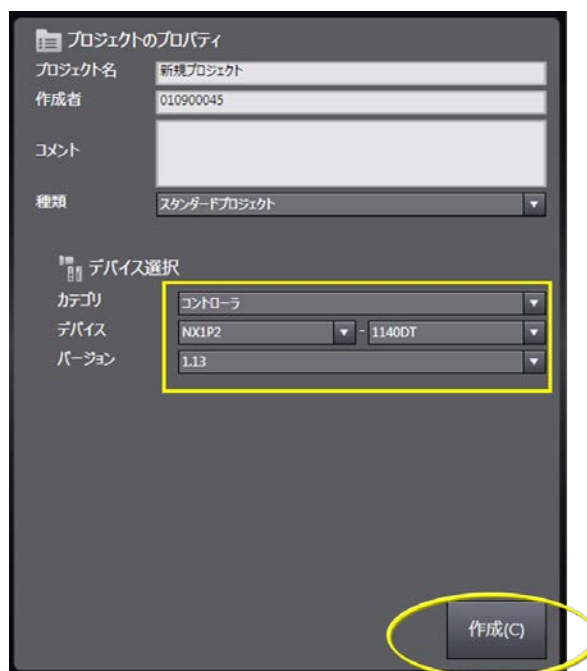


4-2-3 新規プロジェクトの作成

1. Sysmac Studio を起動します。

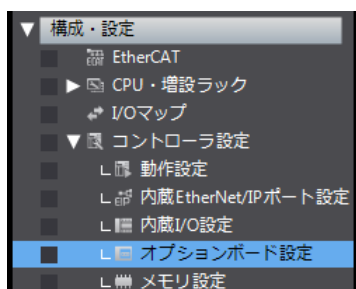


2. プロジェクト名を任意の名称をつけ、デバイス「NX1P2」および詳細形式(9024DT/1140DT など)、バージョン「1.13(NX1P 本体に記載のバージョン)」を選択し、[作成]ボタンをクリックします。

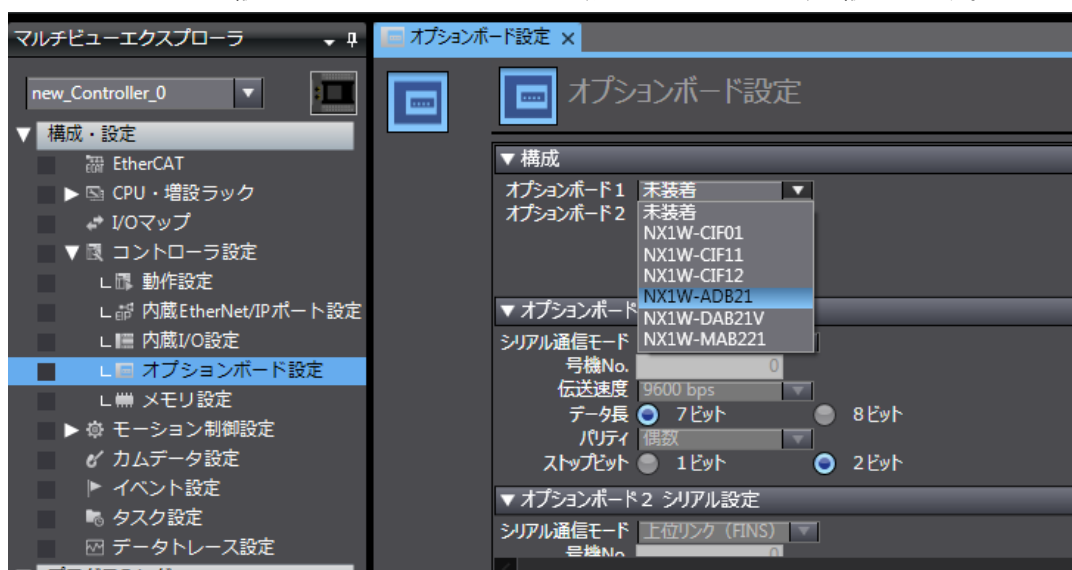


4-2-4 アナログオプションボード設定

1. マルチビューエクスプローラから、[構成・設定]-[コントローラ設定]-[オプションボード設定]を選択します。

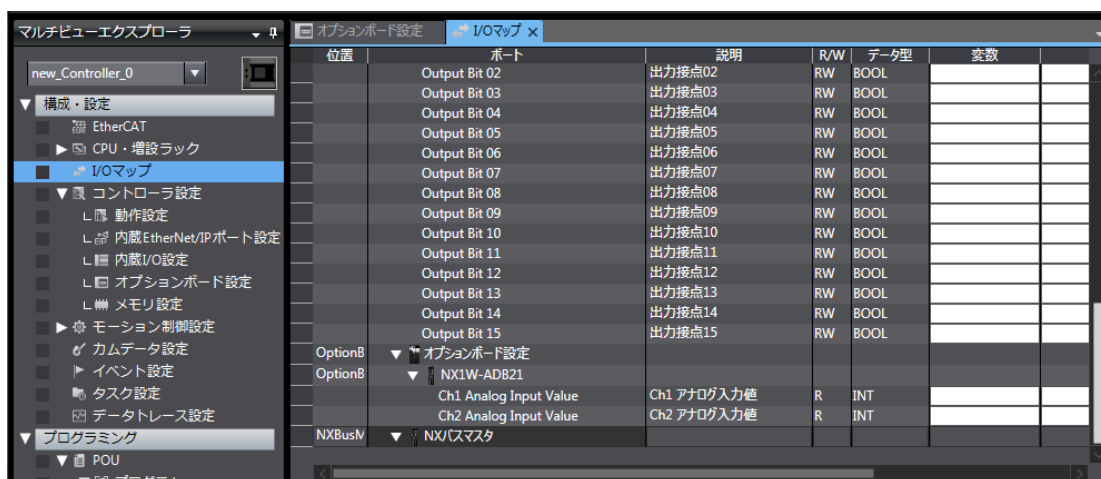


2. オプションボード設定にアナログ入力オプション(形 NX1W-ADB21)を設定します。



4-2-5 アナログ入力と入力端子に変数を割り付ける

1. 次に[I/O マップ]を選択します。
I/O マップの下部に、先ほど設定した NX1W-ADB21 のアナログ入力値の変数設定エリアが出てきます。



2. Ch1 Analog Input Value に変数「data1」を設定します。

	Output Bit 14	出力接点14	RW	BOOL		
	Output Bit 15	出力接点15	RW	BOOL		
OptionBoard	▼ オプションボード設定					
OptionBoard 1	▼ NX1W-ADB21					
	Ch1 Analog Input Value	Ch1 アナログ入力値	R	INT	data1	
	Ch2 Analog Input Value	Ch2 アナログ入力値	R	INT		
NXBusMaster	▼ NXバスマスタ					

変数 data1 は、変位センサのアナログ入力値 0-10V に応じて、0～4000 の整数(INT 型データ)が入力されます。

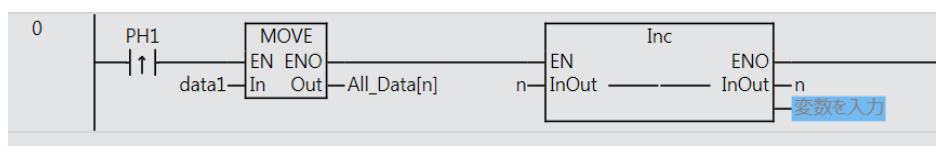
3. 光電センサが配線されている Input Bit 00 に変数「PH1」を設定します。

位置	ポート	説明	R/W	データ型	変数	変数二
	EtherCATネットワークコンフィグレーション					
	▼ CPU-増設ラック					
Built-in I/O	▼ 内蔵I/O設定					
	Input Bit 00	入力接点00	R	BOOL	PH1	
	Input Bit 01	入力接点01	R	BOOL		
	Input Bit 02	入力接点02	R	BOOL		
	Input Bit 03	入力接点03	R	BOOL		
	Input Bit 04	入力接点04	R	BOOL		
	Input Bit 05	入力接点05	R	BOOL		

光電センサの ON/OFF 状態により、変数 PH1 は、True(ON)、False(OFF)になります。
データ型は BOOL 型です。

4-2-6 プログラム例

PH1 が ON すると、MOVE 命令(データ転送命令)で data1 の数値を All_Data の n 番目の配列データを格納し、INC 命令(インクリメント命令)で、n を 1 加算します。(n=n+1)。
PH1 が ON するたびに、data1(アナログ入力値)を All_Data[0],All_Data[1], All_Data[2]....に格納します。

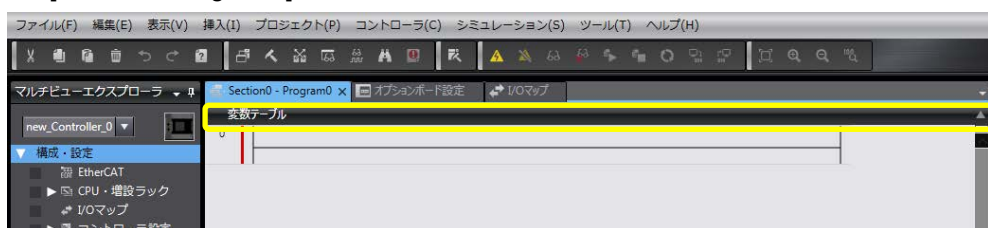


4-2-7 配列の設定方法

All_Data[n] の配列の設定を行います。

All_Data[0]
All_Data[1]
All_Data[2]
All_Data[3]
All_Data[n]

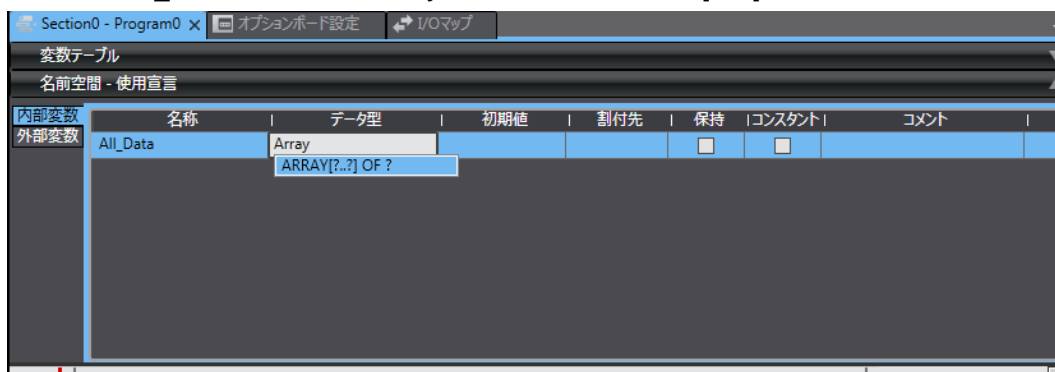
1. [Section0-Program0]のタブを選択し、「変数テーブル」をクリックして、変数テーブルを表示します。



2. 内部変数の新規登録を行います。

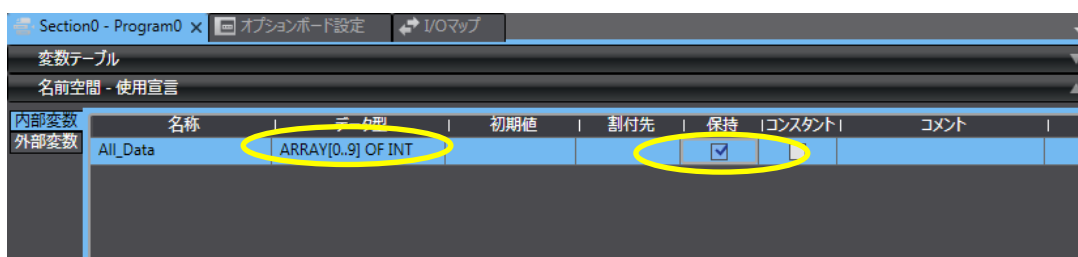
配列は、ARRAY 型(配列型)の変数として指定します。

名称に All_Data データ型に“Array”と入力します。ARRAY[?..?]OF? とガイドがでできます。



3. ARRAY [?..?]OF? の?のところに、ARRAY [0..9] OF INT と入力します。これにより All_Data[0] ~ All_Data[9]の INT(整数型)の配列変数が設定できました。

従来の PLC(CJ2 など)のデータメモリ(DM)ように PLC の電源をオフしてもデータを保持するようにするために「保持」オプションがあります。



4. 内部変数の表で右クリックし、メニューから[新規追加]を選択します。

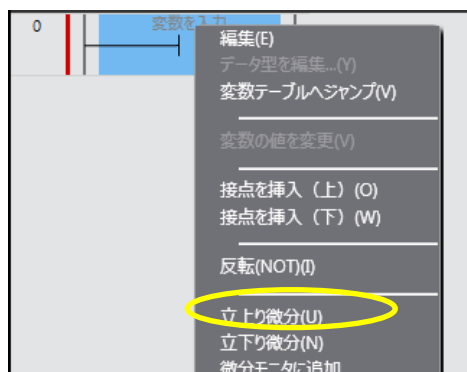
配列で使用する n: All_Data[n]の中で、カウントしていく変数(INT 型)を設定します。

変数テーブルで初期値を指定することができますので、0 を指定します。

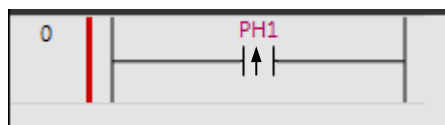
名称	データ型	初期値	割付先	保持	コンスタント	コメント
All_Data	ARRAY[0..9] OF INT			☒	☐	
n	INT	0		☐	☐	

4-2-8 プログラムの入力

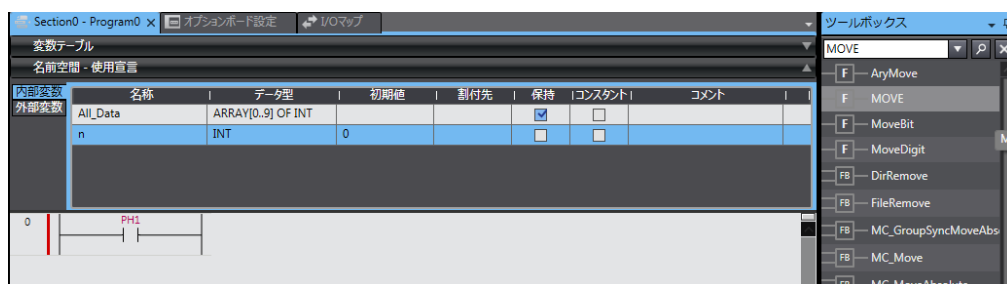
1. 接点 PH1 を立上り微分で入力します。回路で、[C]キー押し、接点が出たところで[右クリック]をし、メニューで[立上り微分]を選択します(もしくは[@]キー)。



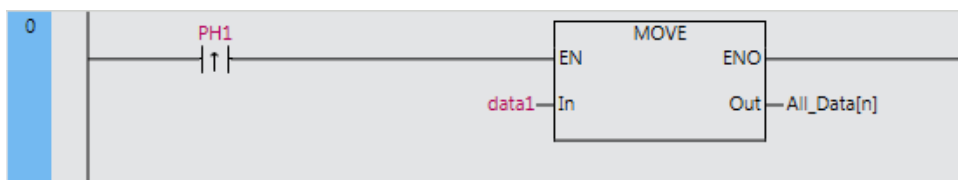
変数入力に、PH1 と入力します。



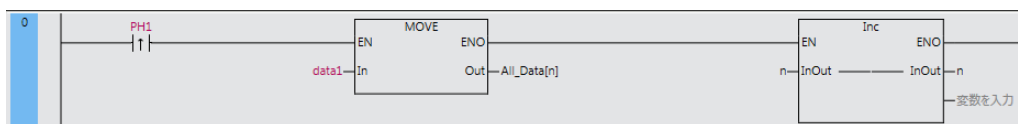
2. MOVE 命令を追加します。ツールボックスで“MOVE“を検索します。 MOVE のファンクション(FUN)が出てきますので、ドラッグ&ドロップで回路図まで持ってきます。(または、「I」キーを押し、“MOVE”と入力)



3. MOVE 命令に、data1 と All_Data[n] を記載します。



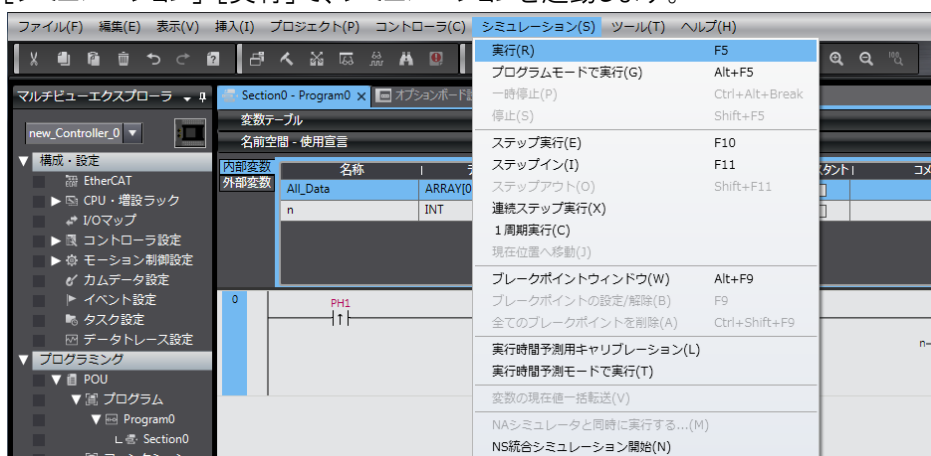
4. MOVE 命令に、d 同様に Inc 命令を追加して、変数にnを記載します。



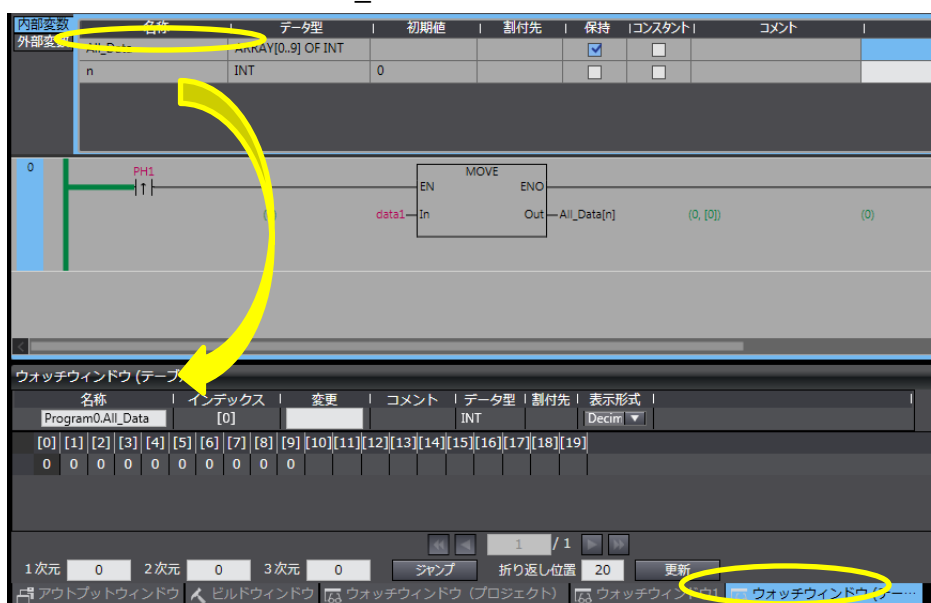
4-2-9 プログラムの動作確認

シミュレーション機能を使用して動作の確認を行います。

1. [シミュレーション]-[実行]で、シミュレーションを起動します。

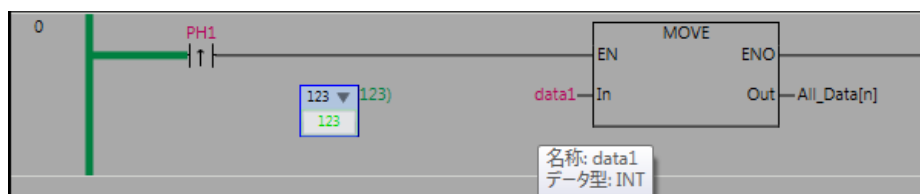


2. [表示]-[ウォッチウィンドウ]から、[ウォッチウィンドウ(テーブル1)]を開きます。
その後、変数テーブルから「All_Data」をウォッチウィンドウにドラッグ＆ドロップします。



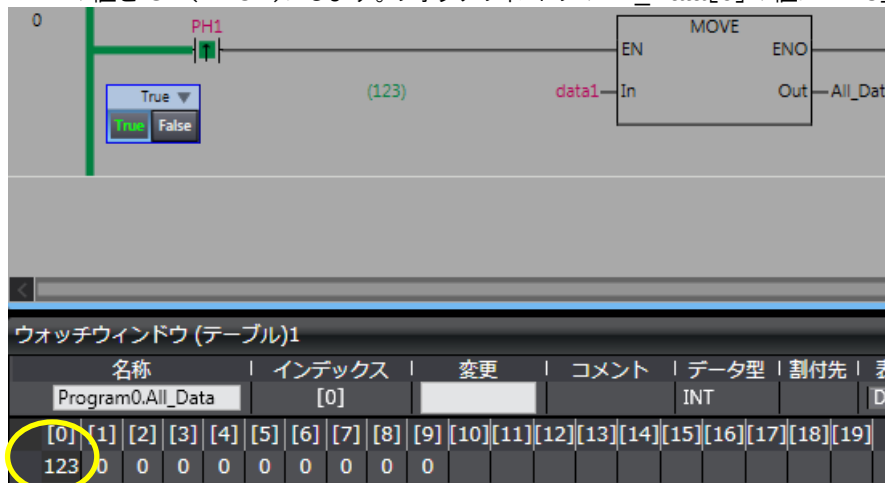
3. MOVE 命令の「data1」をダブルクリックすると値を設定することができます。

ここでは、値を「123」と入力してエンターキーを押します。

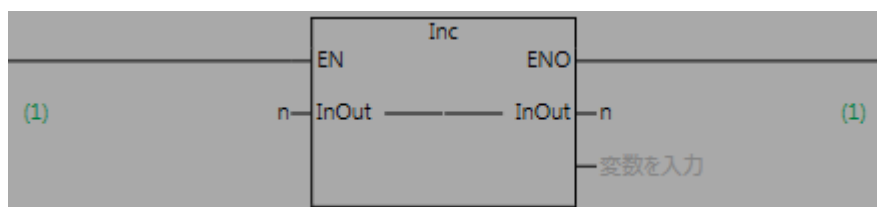


4. a 接点「PH1」の上でエンターキーを押すと、ON/OFF の値を変更することができます。

PH1 の値を ON (TRUE) にします。ウォッチウィンドウの All_Data[0] の値に「123」が格納されます。



また、INC 命令の「n」の値がインクリメントされて、1になっていることを確認します。



5. 同様に All_Data の配列 [1] から順に任意の値を登録してみましょう。

ウォッチウィンドウ (テーブル)1																				
名称		インデックス		変更		コメント		データ型		割付先		表								
Program0.All_Data		[0]																		
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]	[14]	[15]	[16]	[17]	[18]	[19]	
123	2000	100	700	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

上記のように、data1 の値が、All_Data という配列変数へ順番に値が登録されます。

1. 登録された配列のデータを参照するときは以下のように行います。

下記の例は、All_Data の添え字[3] (4 番目に登録した値) を取り出して、temp1 という新たな INT 型の変数に代入しているプログラム例です。

The screenshot shows a ladder logic program with a single step labeled '1'. It contains a 'MOVE' instruction. The input (EN) is a constant value '(700)' followed by the variable 'All_Data[3]'. The output (ENO) is the variable 'temp1', which is followed by a constant value '(700)'. Both '(700)' values are circled in yellow. Below the ladder logic is a 'ウォッチウィンドウ (テーブル)1' (Watch Window (Table) 1). It has columns for '名称' (Name), 'インデックス' (Index), '変更' (Change), 'コメント' (Comment), 'データ型' (Data Type), '割付先' (Assignment), and '表示形式' (Display Format). The first row shows 'Program0.All_Data' with index '[3]', data type 'INT', and display format 'Decim'. Below this is a table of values for indices [0] through [19]. The value for index [3] is '700', which is circled in yellow.

名称	インデックス	変更	コメント	データ型	割付先	表示形式													
Program0.All_Data	[3]			INT		Decim													
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]	[14]	[15]	[16]	[17]	[18]	[19]
123	2000	100	700		0	0	0	0	0										

2 次のステップに進む前にオフラインにし、

プロジェクトファイルの保存とエクスポートを行っておきましょう。

5

5 モーション FB プログラミング

この章では、モーション FB プログラミングについて説明します。

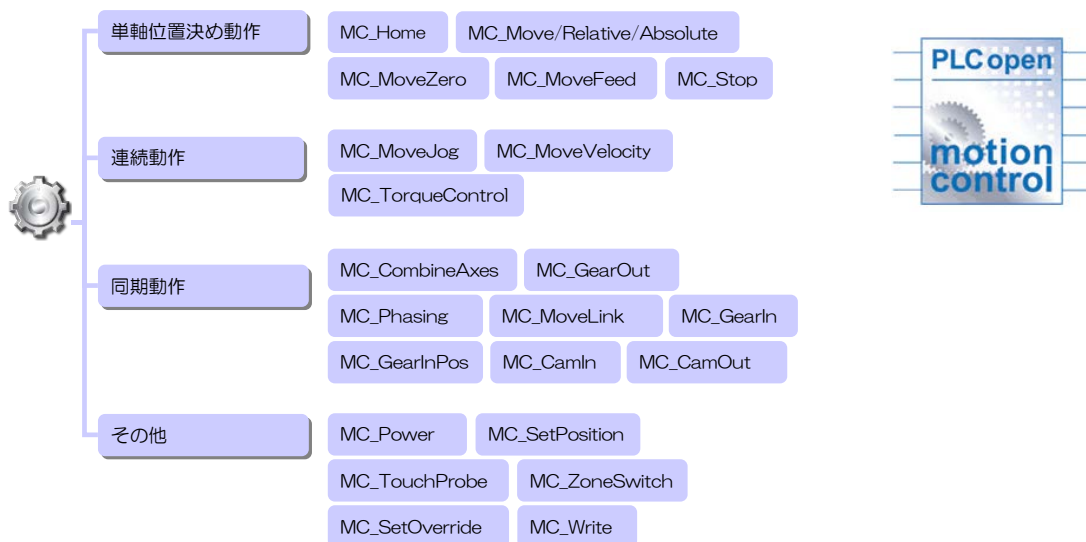
5-1	モーション FB プログラミング	5-85
5-1-1	モーション FB プログラミングとは	5-85
5-1-2	プログラミングの流れ	5-85
5-2	サーボドライバの追加とパラメータ設定	5-86
5-2-1	サーボドライバの登録	5-86
5-2-2	軸の登録	5-87
5-2-3	軸パラメータの設定	5-87
5-3	モーション FB プログラミング	5-89
5-3-1	ラダープログラムの概要	5-89
5-3-2	使用するモーション FB	5-89
5-3-3	ラダープログラムの作成	5-90
5-4	データトレース	5-93
5-4-1	データトレースによる動作確認	5-93
5-5	3D シミュレーション	5-95
5-5-1	3D シミュレーションの起動方法	5-95

5-1 モーション FB プログラミング

5-1-1 モーション FB プログラミングとは

ここでは、PLCopen®が提唱しているモーションファンクションブロック(以下、モーション FB)によるモーション制御プログラミングを学びます。

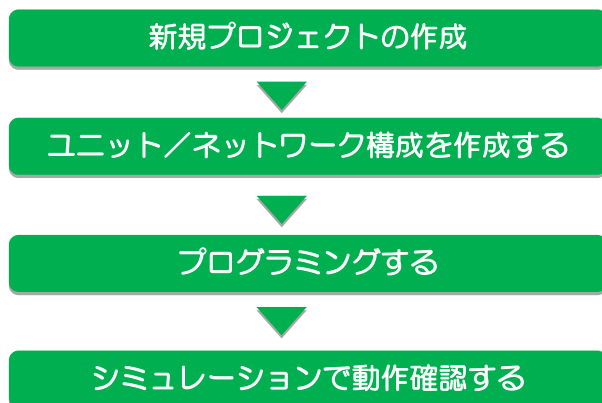
NX1P では、以下のようなモーション動作を実行するFBが用意されていますので、FBを組み合わせるだけで思い通りのモーション制御が可能です。



5-1-2 プログラミングの流れ

モーションFBを使った簡単な位置決めプログラムの作成を行います。

Sysmac Studio では、NX1P やサーボモータ等の実機がなくても、シミュレーション機能によりプログラムのデバッグやモーション動作を 3D 表示で確認することができます。

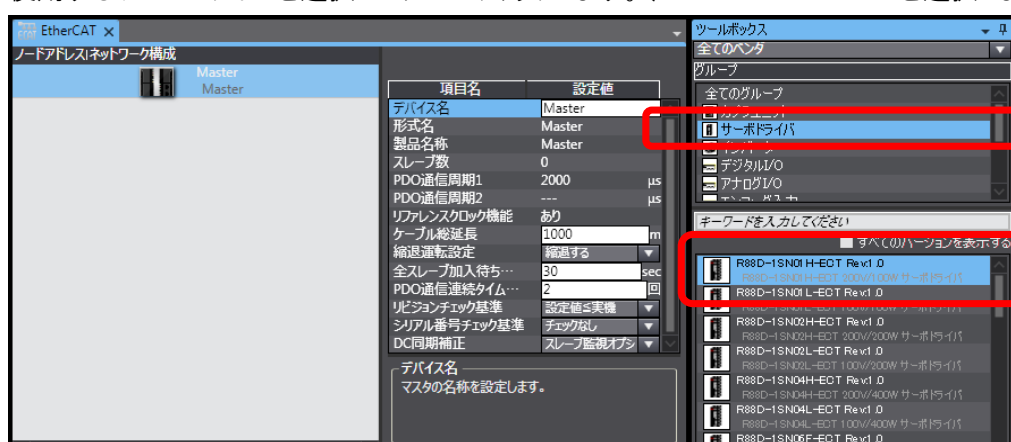


ここでは、モーションFBの使い方を中心に紹介しています。実際には、サーボモータの設定やアブソリュートエンコーダの設定などが必要ですが、省略して記載しています。

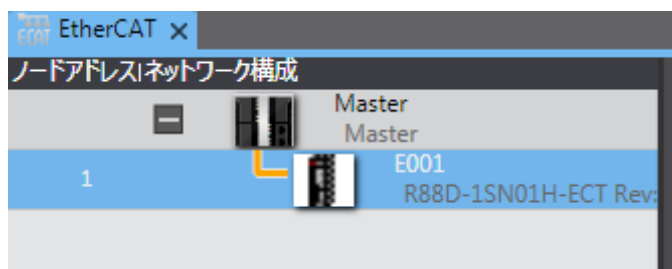
5-2 サーボドライバの追加とパラメータ設定

5-2-1 サーボドライバの登録

1. 新規プロジェクトを作成後、[構成・設定]-[EtherCAT]をダブルクリックし、EtherCAT 構成画面を表示させます。
2. サーボドライバを EtherCAT ネットワークに追加します。
上記のツールボックスでサーボドライバを選択し、サーボドライバの一覧が表示します。
使用するサーボドライバを選択してダブルクリックします。(ここでは R88D-1S□を選択します)



3. マスタの下にサーボドライバが追加されます。

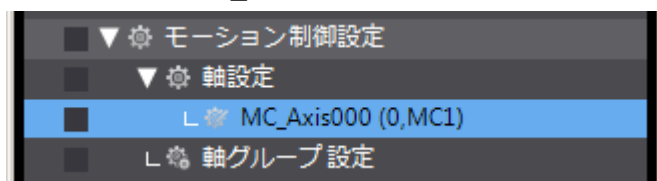


5-2-2 軸の登録

1. モーション制御を行うためには軸の登録が必要です。[構成・設定]-[モーション設定]-[軸設定]から右クリックして、[追加]-[単軸位置決制御]を選択します。



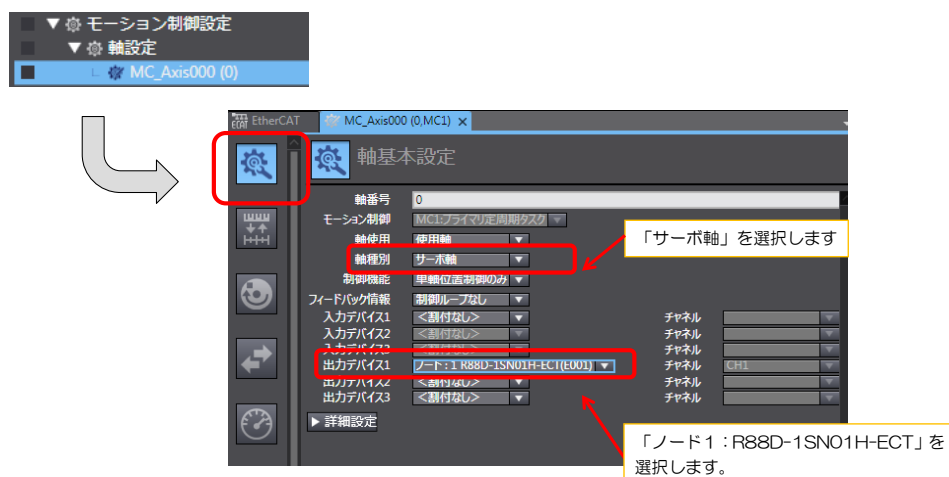
2. 以下のように、[MC_Axis000] が追加されます。



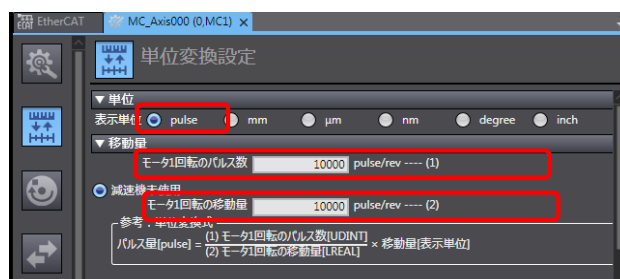
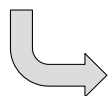
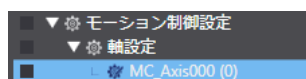
5-2-3 軸パラメータの設定

[MC_Axis000]をダブルクリックすると各種パラメータを設定する画面になります。本実習で必要となるパラメータのみを設定していきます。

1. 「軸基本設定」の内容を設定します。



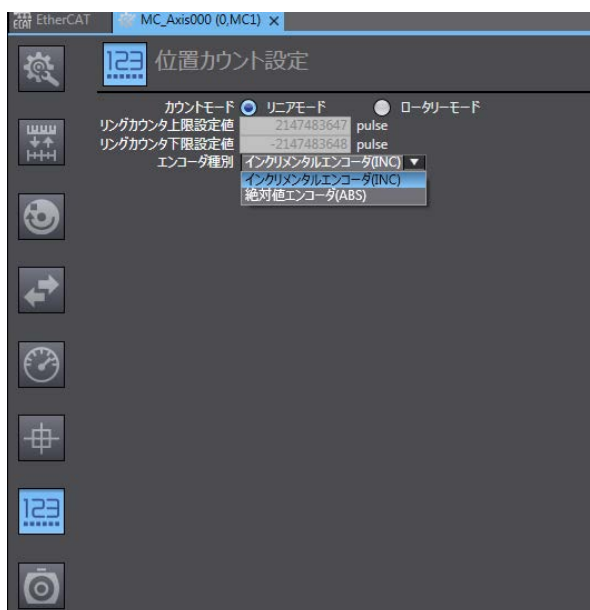
2. 「単位変換設定」の内容を以下のようにになっていることを確認します(初期設定値)



参考

サーボシステム 1S シリーズは絶対値エンコーダ搭載ですが、ここでは、初期設定のインクリメンタルエンコーダ設定で使用する例で紹介します。

絶対値エンコーダとして使用する場合は、[位置カウント設定]で絶対値エンコーダを選択してください。

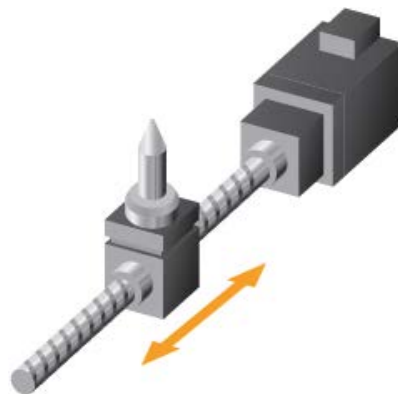


5-3 モーション FB プログラミング

5-3-1 ラダープログラムの概要

単軸動作のモーション FB プログラミングを行います。

下記の図のように、ボールねじを左右に移動するプログラムを、モーション FB を使って作成してみましょう。



5-3-2 使用するモーション FB

この実習で使用するモーション FB は以下の 3 種類です。

(Sysmac Studio 命令語ヘルプより)

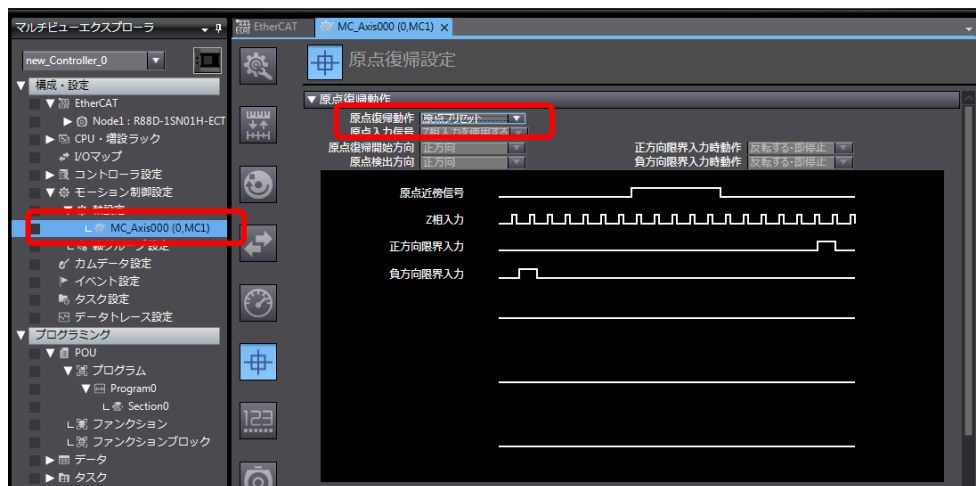
MC_Power サーボドライバを運転可能状態に切り替えます。			
命令	名称	FB/FUN	グラフィック表現
MC_Power	運転可	FB	MC_Power_instance MC_Power Axis Enable Axis Status Busy Error ErrorID
MC_MoveRelative 指令現在位置からの移動距離を指定して、位置決めを行います。			
命令	名称	FB/FUN	グラフィック表現
MC_MoveRelative	相対値 位置決め	FB	MC_MoveRelative_instance MC_MoveRelative Axis Execute Distance Velocity Acceleration Deceleration Jerk BufferMode Axis Done Busy Active CommandAborted Error ErrorID
MC_Home 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ます。			
命令	名称	FB/FUN	グラフィック表現
MC_Home	原点復帰	FB	MC_Home_instance MC_Home Axis Execute Axis Done Busy CommandAborted Error ErrorID



参考

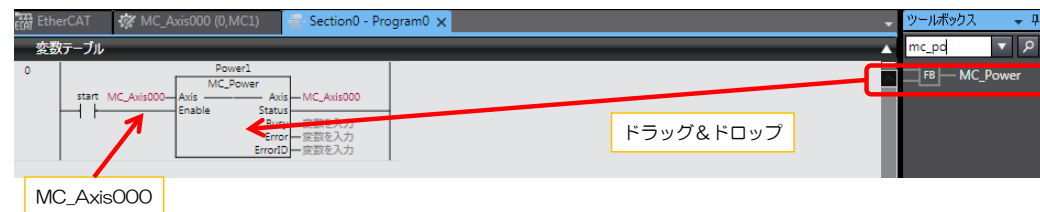
MC_HOME は原点復帰命令です。原点復帰命令のパターンは、[軸設定]-[MC_Axis000]の[原点復帰設定]で設定します。この例では 初期値の、[原点プリセット]で使します。

原点プリセット: MC_Home を実行すると、現在値を0にする。



5-3-3 ラダープログラムの作成

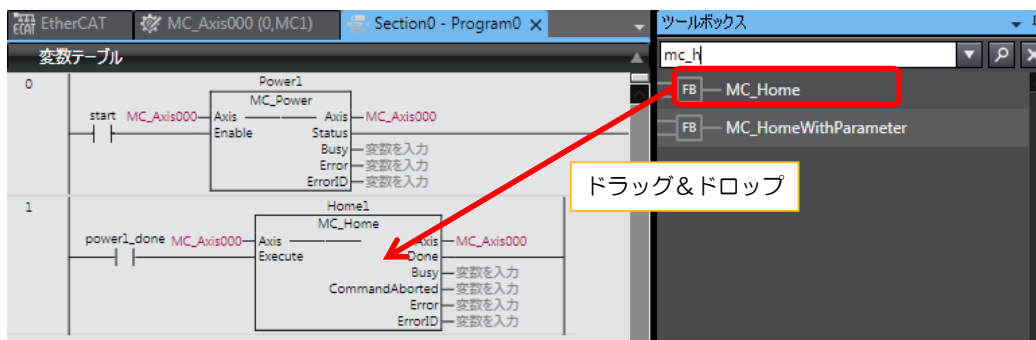
1. Sysmac Studio を起動し、新規プロジェクトを作成します。
[プログラミング]-[POU]-[プログラム]-[Program0]-[Section0]をダブルクリックして、ラダーエディタを開きます。
2. 1回路目のラダーを入力します。まず a 接点「start」を入力します。
次に、モーション FB「MC_Power」を入力し、インスタンス名を「Power1」とします。
(ツールボックスから「MC_Power」を検索してドラッグ&ドロップすると便利です)
「Axis」のパラメータに「MC_Axis000」を入力します。



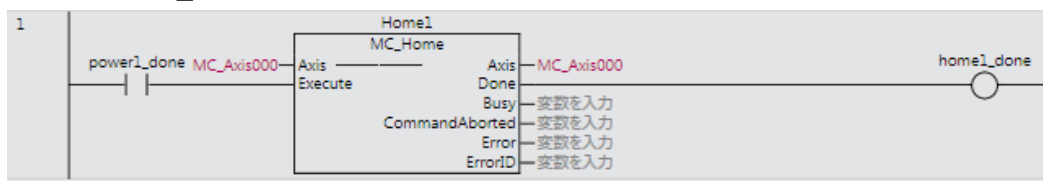
3. コイル「power1_done」(名前は任意)を入力します。



4. [R]キーで回路を下に追加し、2 回路目のラダーを入力します。
a 接点「power1_done」を入力後、モーション FB「MC_Home」を入力します。
インスタンス名は「Home1」とします。



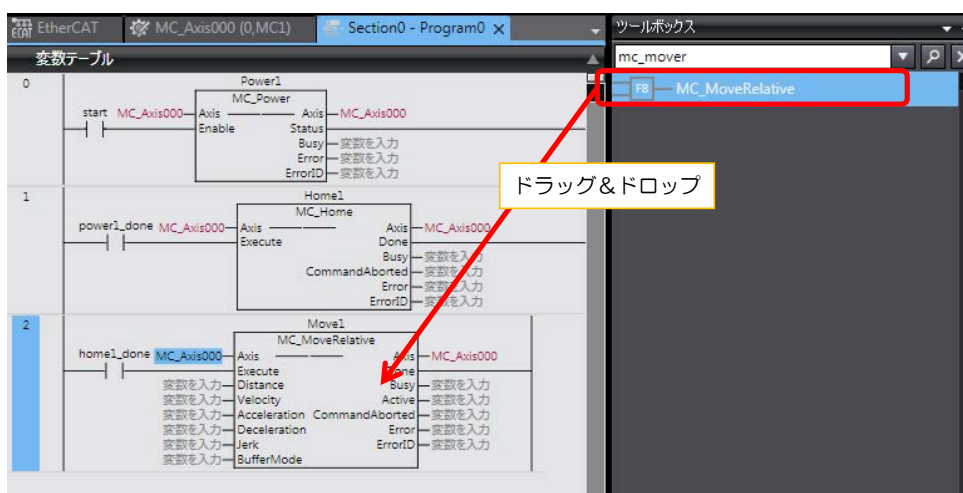
5. コイル「home1_done」(名前は任意)を入力します



6. [R]キーで回路を下に追加し、3 回路目のラダーを入力します。

a 接点「home1_done」を入力後、モーション FB「MC_MoveRelative」を入力します。

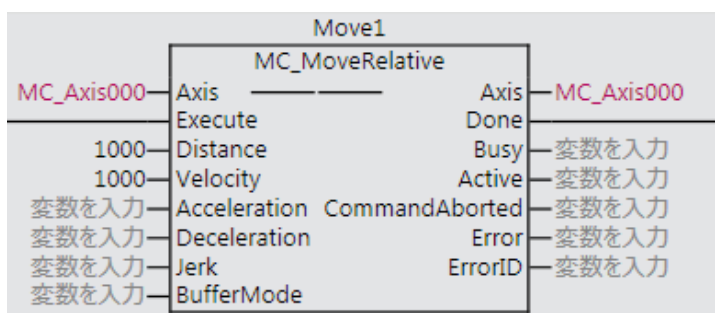
インスタンス名は「Move1」とします。



7. 以下のようにパラメータを設定します。(ここでは、その他のパラメータは設定不要です)

- ・Distance(距離): 1000 (Pulse)
- ・Velocity(速度): 1000 (Pulse/s)

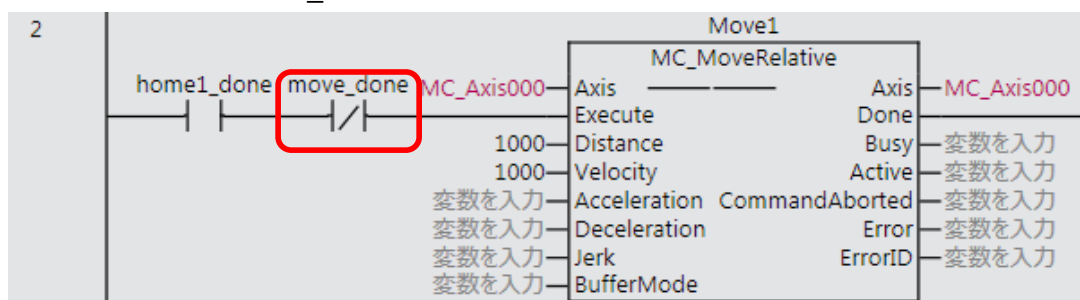
この設定により、指定した距離を1秒で移動する動作になります。



8. 上記 6. 7 と同様に、今度は同じで速度で逆向きに移動する動作を作ります。
インスタンス名は「Move2」とします。逆向きの動作にするために、Distance の値をマイナスで設定します。

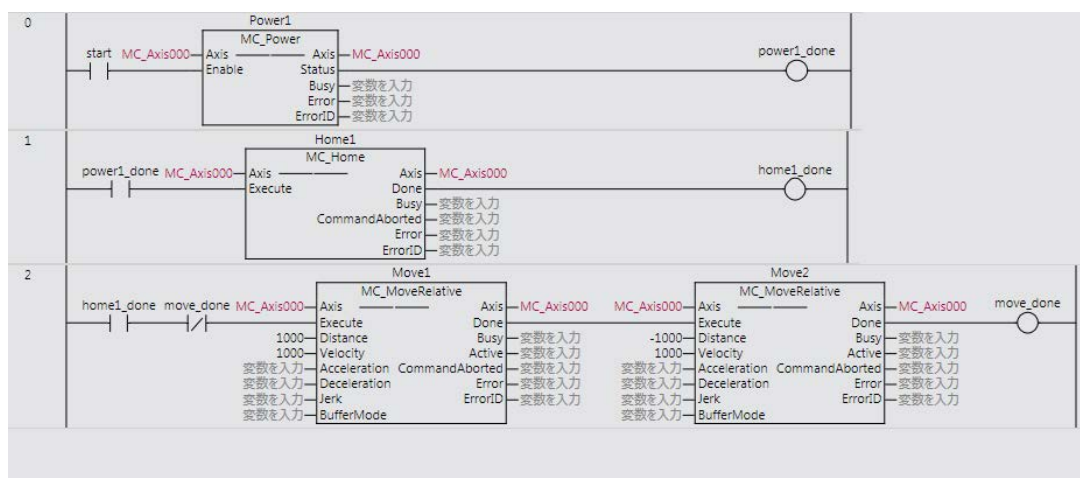


9. 以下のようにコイル「move_done」を入力します。



これでプログラム作成は完了です。

■ 完成プログラム



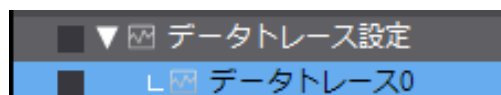
5-4 データトレース

5-4-1 データトレースによる動作確認

単軸モーション制御を動作において、位置情報をデータトレースで確認しましょう。

1. [構成・設定]-[データトレース]で右クリックして、[追加]-[データトレース]を選択します。

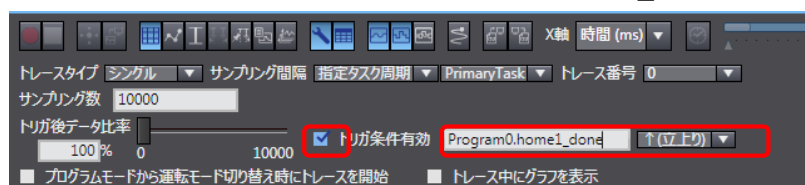
「データトレース 0」が追加されます。



2. 「データトレース 0」をダブルクリックして、以下の内容を設定します。

トリガ条件は、「Program0」と入力し、「.」(ピリオド)を入力すると変数の候補が表示されます。

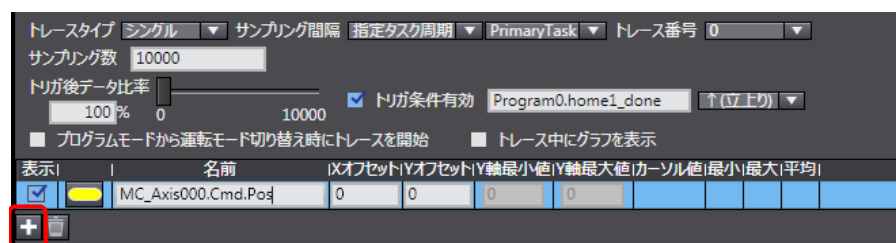
ここでは、モーション FB が起動する条件となる「home1_done」の「↑」(立上り)を選択します。



3. 次にトレースするデータを設定します。

「+」を押して、「MC_Axis000.Act.Pos」を指定します。

MC_Axis000 の後で「.」(ピリオド)を入力すると候補が表示されますので、「Cmd」(指令値)、および「Pos」(位置)を入力していきます。



4. [シミュレーション]-[実行]を選択します。これにより、NX1P の実機がなくてもプログラムデバッグができる状態になります。

画面の上側が黄緑色の線になるとシミュレーションが開始されている状態を表します。

5. 赤いボタンを押してトレースを開始します。

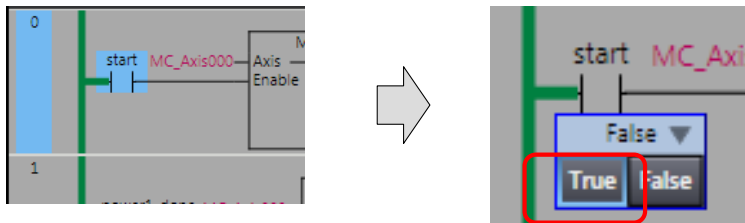
画面左下の表示が「トリガ待ち」となります。



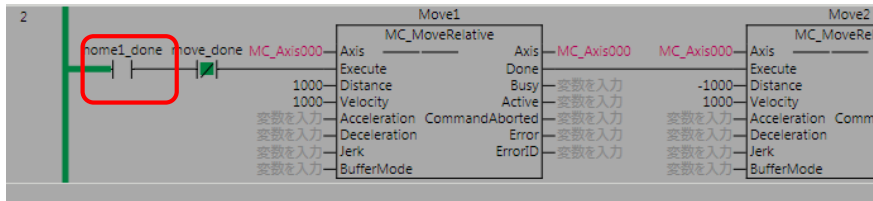
6. 「Section0」をクリックしてラダーを表示します。

セット／リセット機能で、仮想的に ON/OFF の信号を指定してきます。

先頭回路の「start」を選択して、ダブルクリックして、「True」を選択します。



7. 「start」をオン(True)にすると、トリガ条件の「home1_done」もオンし、トレースが開始されます。



画面右下の表示が水色のバー表示がトレースの進行状況を表します。

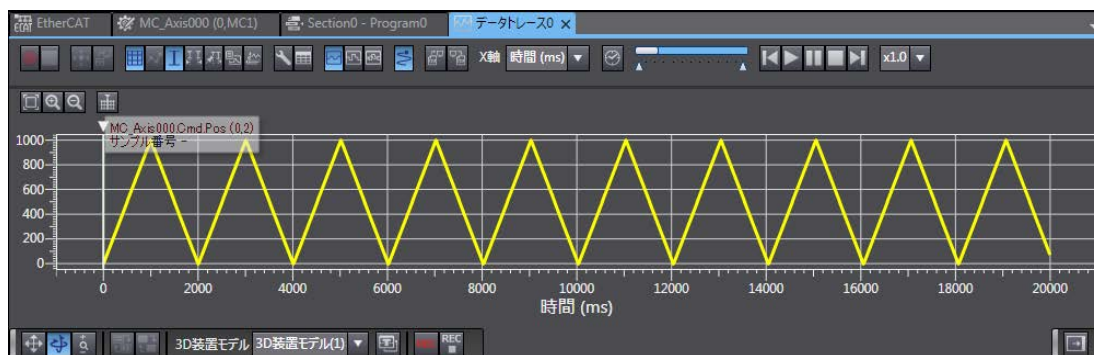
バーが消えるとトレースは完了です。



8. [構成・設定]-[データトレース]-[データトレース 0]をダブルクリックします。

以下のように位置データがトレースされていることを確認します。

これにより、三角の波形で、サーボモータの動作が 1 秒ごとに左右に動いていることが分かります。(横軸は時間、縦軸は移動距離(Pulse)です)



以下のアイコンで項目を表示／非表示できますので、見やすい画面レイアウトにしましょう。



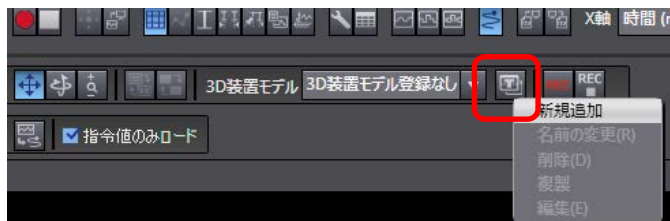
5-5 3D シミュレーション

5-5-1 3D シミュレーションの起動方法

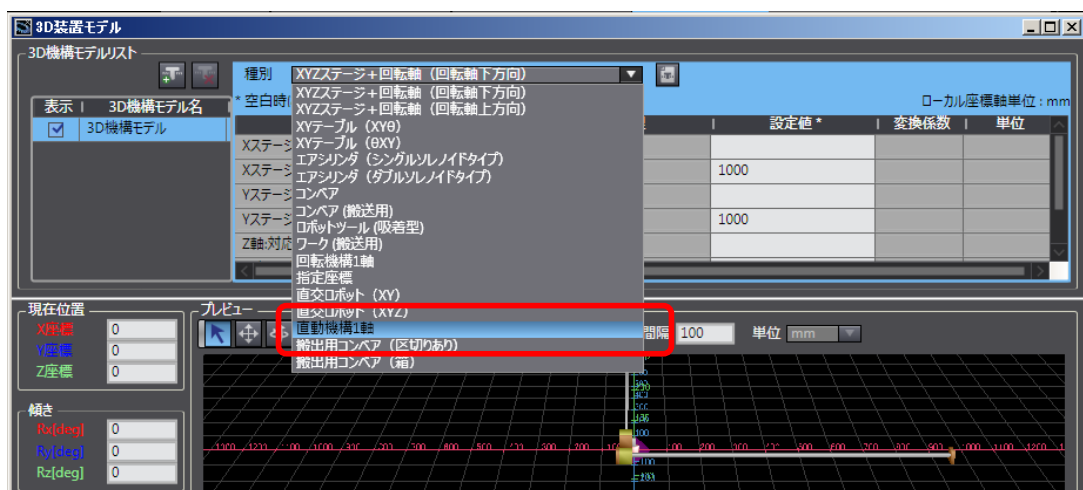
1. 以下のアイコンをクリックします。また、不要な画面を消します。



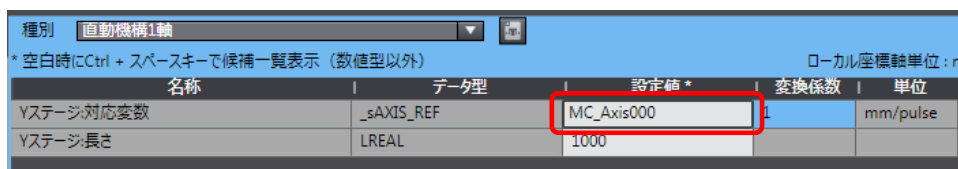
2. 以下のアイコンをクリックして、[新規追加]を選択します。



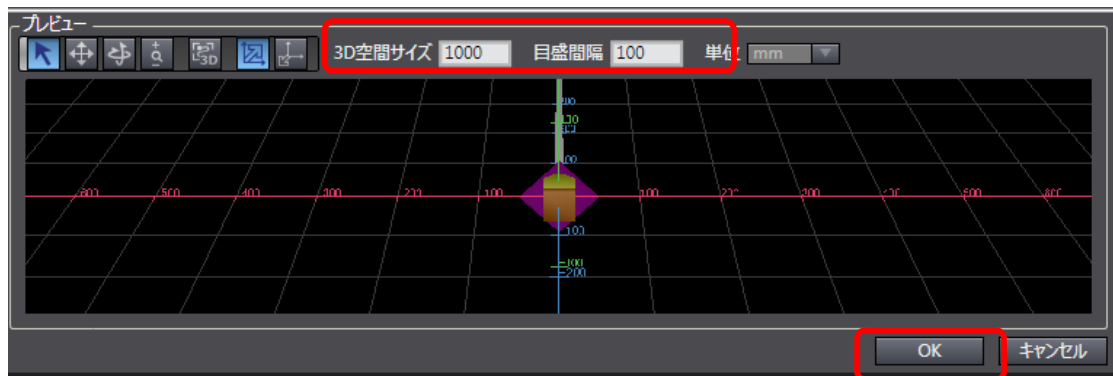
3. 3D 機構モデルリストの種別は、今回の動作に近い「直動機構 1 軸」モデルを選択します。



4. Y ステージ: 対応変数に「MC_Axis000」を指定します



5. 3D 空間サイズを 1000、目盛間隔を 100 程度に設定して、[OK]を押します。



6. 下記のアイコンを押して、トレースした波形データを読み込みます。



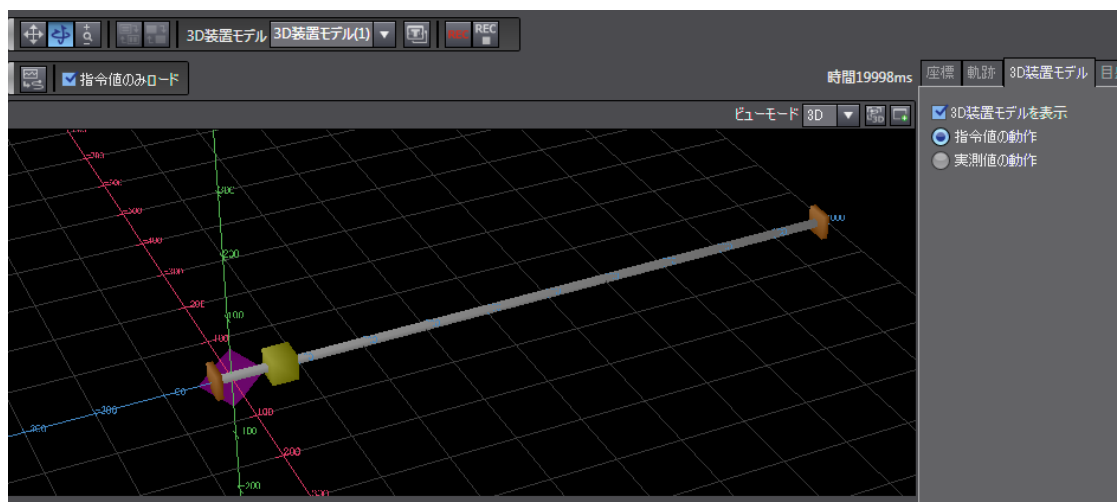
7. 以下のアイコンを押して 3D でモーション動作を確認します



8. 以下のアイコンで 3D 画面の移動や回転、拡大・縮小などが行えます



9. 以上で 3D シミュレーションデバッグは完了です。



10. 次のステップに進む前にオフラインにし、
プロジェクトファイルの保存とエクスポートを行っておきましょう。

6

6 はじめての ST 言語プログラミング

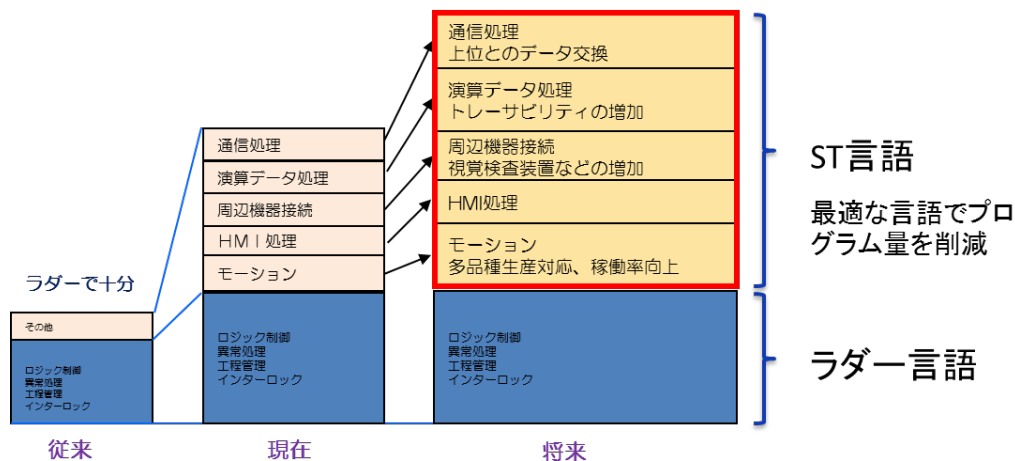
この章では、ST 言語を使ったプログラミングについて説明します。

6-1	ST プログラミングとは.....	6-98
6-1-1	ST 言語の利点.....	6-98
6-1-2	制御構文を含んだ ST プログラム.....	6-98
6-1-3	ST 記述の基本文法と使用例.....	6-99
6-1-4	演算子一覧.....	6-99
6-2	NX1P での ST プログラミング.....	6-100
6-2-1	NX1P での ST プログラミングの記述方法.....	6-100
6-3	ST プログラミング実習.....	6-101
6-3-1	数値計算プログラミング実習.....	6-101
6-3-2	プログラム手順.....	6-102
6-3-3	完成プログラムとプログラムチェック.....	6-104
6-3-4	ST プログラムの動作確認.....	6-104

6-1 ST プログラミングとは

6-1-1 ST 言語の利点

従来に比べ機械制御プログラムが大規模化、複雑化しています。これまではラダーで記述しやすかった状態制御やインターロック制御などの比率が少なくなり、複雑な演算処理やデータストレージのような情報処理などラダーが不得意とする処理が約 7 割を占めるようになってきています。この部分を ST 言語に移行することにより、容易にプログラムを記述できるだけでなく、プログラム量も削減できます。



6-1-2 制御構文を含んだ ST プログラム

ST 言語は、演算処理だけでなく、BASIC 言語や C 言語のように、条件分岐 (IF 文) や繰り返し処理 (FOR 文) といった制御も可能です。

従来ラダーで表現しにくかった演算や制御文を含むプログラム開発が容易になります。

```
IF NOT Fault THEN
  Ready:=True; // Green light in ON state
  AlarmSignal:=FALSE;
  (*Turn OFF the red light
    and enable machine for operation*)

  IF NOT HomingDone THEN
    ExecuteHoming:=TRUE; // Make homing for 1st time
    OpenGrip:=TRUE; // Open grip for enable grasping
  END_IF
ELSE
  ;
  Ready:=FALSE;
  ExecuteHoming:=FALSE;
  AlarmSignal:=TRUE;
  (*Turn red light ON*)
END_IF
```

// 以降はコメント

(* *) もコメント

IF~THEN~
END_IF;

6-1-3 ST 記述の基本文法と使用例

ST 言語は、前述のように一般的な数式でプログラミングできる言語であるため、新たに覚える知識はほとんどありません。

以下は、ST 言語の基本文法ですが、次の2項目を覚えるだけでプログラミングできます。

(1) 変数に値を代入する場合は、“:=” 記号を使用します。

(2) 式の最後には、必ずセミコロン“ ; ”が必要です。

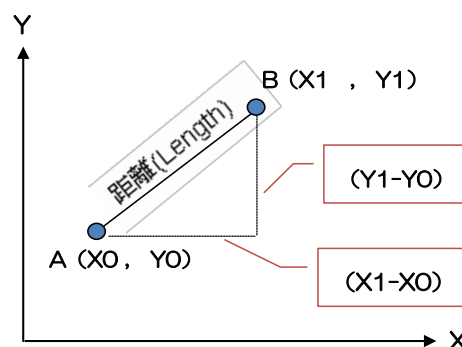
ST 言語の使用例

下記の例は、三平方の定理を使った2点間の距離を求めるプログラムです。

数式(公式)をプログラミングするだけです。

■一般的な数式(公式)

$$\text{Length} = \sqrt{(X1 - X0)^2 + (Y1 - Y0)^2}$$



■ST プログラム

```
Length := SQRT (( X1 - X0 )**2 + ( Y1 - Y0 )**2);
```

※SQRT:平方根(Square Root)

6-1-4 演算子一覧

演算子	意味	記述例	順位
()	演算順序の変更	Value:= (1+2)*(3+4); // Value は 21	1
-, +	符号	+100, -100	2
NOT	論理否定	Value:= NOT TRUE; // Value は FALSE	2
**	べき乗	Value:= 2**8 ; // Value は 256	3
*	乗算	Value:= 8 * 100; // Value は 800	4
/	除算	Value:= 200 / 25; // Value は 8	4
MOD	余り	Value:= 10 MOD 6; // Value は 4	4
+	加算	Value:= 200 + 25; // Value は 225	5
-	減算	Value:= 200 - 25; // Value は 175	5
< , > , <= , >=	比較	Value:= 60 > 10; // Value は TRUE	6
=	等式	Value:= 8=7; // Value は FALSE	7
< >	不等式	Value:= 8<>7; // Value は TRUE	7
& , AND	論理積	Value:= 2#1001 AND 2#1100; // Value は 2#1000	8
XOR	排他的論理和	Value:= 2#1001 XOR 2#1100; // Value は 2#0101	9
OR	論理和	Value:= 2#1001 XOR 2#1100; // Value は 2#1101	10

6-2 NX1P での ST プログラミング

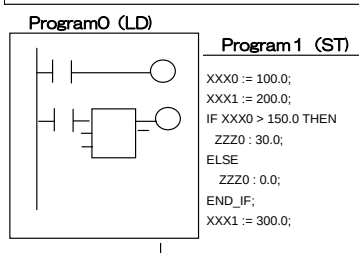
6-2-1 NX1P での ST プログラミングの記述方法

NX1P で ST プログラムを記述する場合、以下の2種類の方法でプログラミングすることができます。

1. ST 言語のみでプログラミングする
2. ラダー言語の一部だけを ST 言語でプログラミングする(インライン ST)

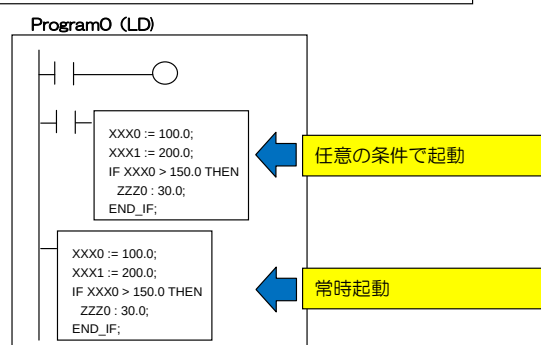
上記2. ではさらに、任意の条件(接点の ON など)でプログラムを実行させる方法と、常時 ST 実行させる方法があります。

1. ST 言語のみでプログラミングする



※複数のプログラムを登録する場合は追加したプログラムを**タスク**に割り付ける必要があります。(下記ページ参照)

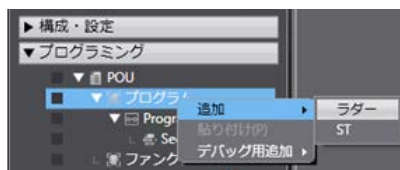
2. ラダー言語の一部を ST 言語でプログラミングする



参考

新たなプログラムの追加とタスクへの割り付け方法

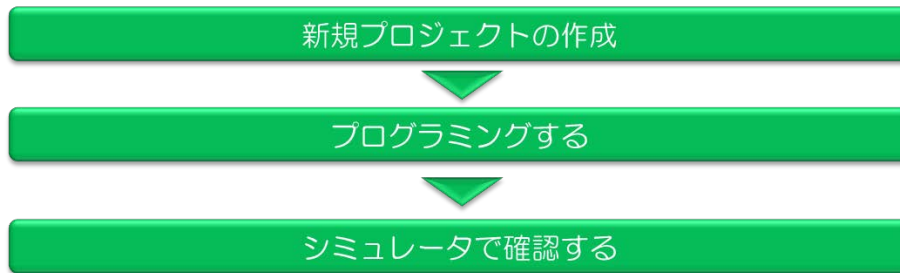
- (1) [プログラム]を右クリックし、メニューから[追加]を選択し、使用したい言語(ラダーまたは ST)を選択します。



- (2) [タスク設定]をダブルクリックし、[プログラムの割付設定]でタスク (Primary Task) に割り付け、初期条件を設定します。



6-3 ST プログラミング実習



6-3-1 数値計算プログラミング実習

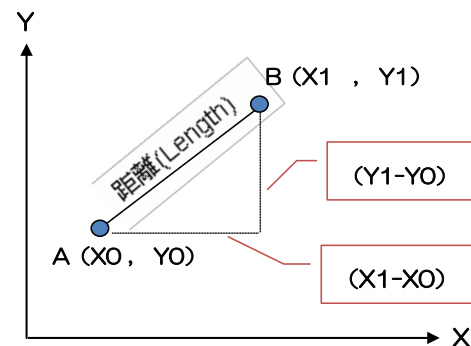
以下のような二点間の距離を計算する ST プログラムを作成しましょう。
ここでは、ラダーの一部だけを ST 言語で記述する「インライン ST」で作成してみましょう。

実習

SW3(スイッチ)を押すと、二点間の距離を求める ST プログラムが実行するプログラムを作成します。

【三平方の定理】

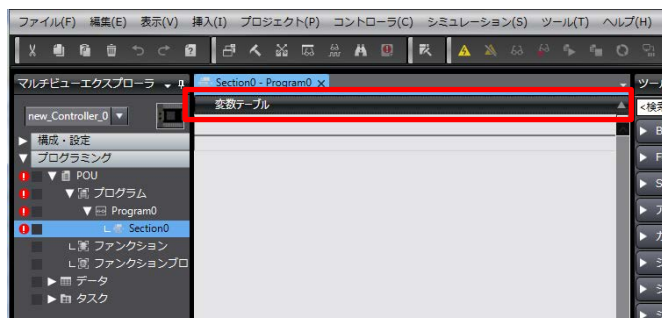
$$\text{Length} = \sqrt{(X1 - X0)^2 + (Y1 - Y0)^2}$$



6-3-2 プログラム手順

1. 新規プロジェクト作成後、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]-[Program0]-[Section0]をクリックしてラダーエディタを開きます。

2. 「変数テーブル」をクリックし表示します。



3. ST で使用する変数を変数テーブルに内部変数として登録します。



名前	データ型	初期値	割り先	保持	コンスタント	コメント
X0	LREAL			☐	☐	
Y0	LREAL			☐	☐	
X1	LREAL			☐	☐	
Y1	LREAL			☐	☐	
Length	LREAL			☐	☐	

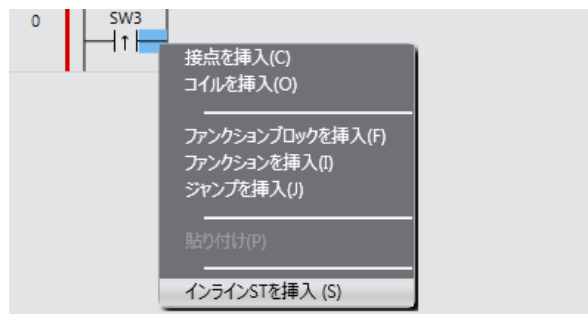
[Ins]キーで新しい行を挿入することもできます。

データ型は、LREAL(倍精度浮動小数点)とすることで、精度の高い小数点の演算が可能です

4. 新たなラダーの行を追加して、a 接点(SW3)を追加します。また、立上り微分にします。(a 接点上で“@”でも指定できます)



5. 下記の位置にカーソルを移動し、右クリックから[インライン ST を挿入]を選択します。

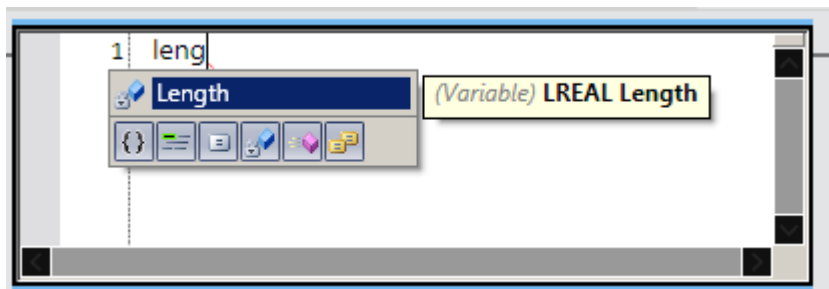


6. 以下ようになります。この箱の中に以下の ST プログラムを記述していきます。

Length := SQRT ((X1 - X0)**2.0 + (Y1 - Y0)**2.0);



7. 1 行目の「Length」を入力するために、「Leng…」あたりまで入力すると、以下のように変数の候補が表示されます



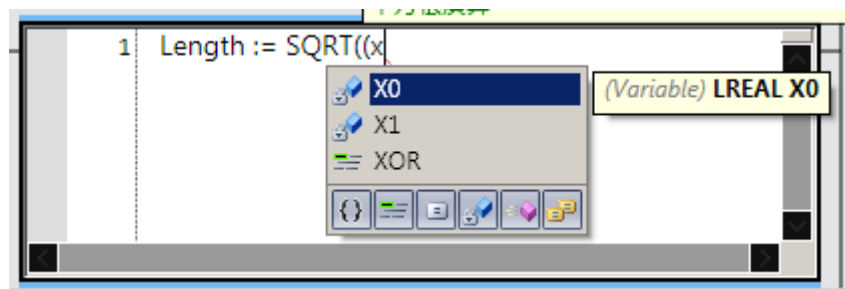
8. コロンを入力すると代入記号 := が自動で入力されます。

Length :=|

9. 平方根(SQRT)を入力します。引数のガイドを参照しながら入力します。



10 X1 を入力します。変数の候補が表示されるので、X1 を選択します。



式をすべて入力します。

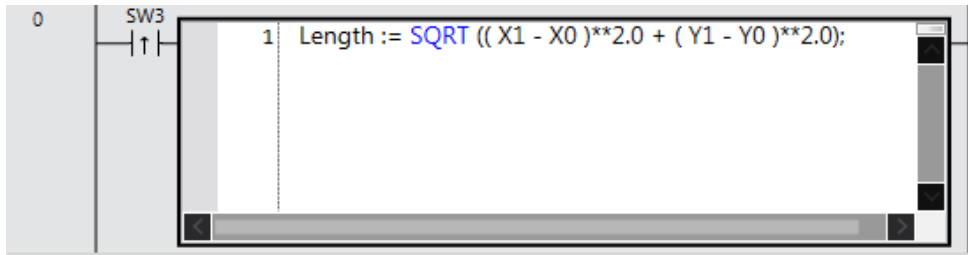
Length := SQRT((X1 - X0) ** 2.0 + (Y1 - Y0) ** 2.0)|

式の最後に ; セミicolonを入力します。

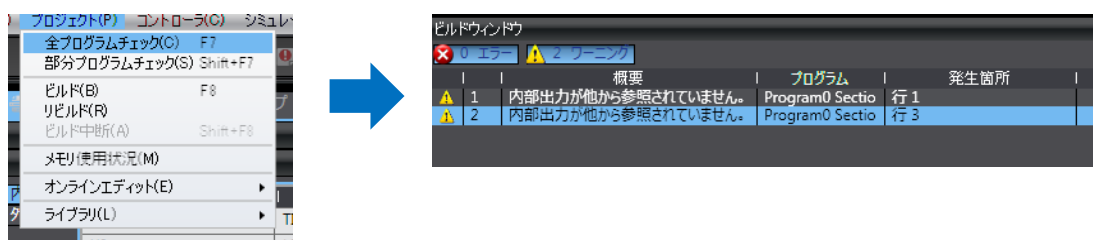
Length := SQRT((X1 - X0) ** 2.0 + (Y1 - Y0) ** 2.0);|

6-3-3 完成プログラムとプログラムチェック

以上でインライン ST プログラミングは完了です

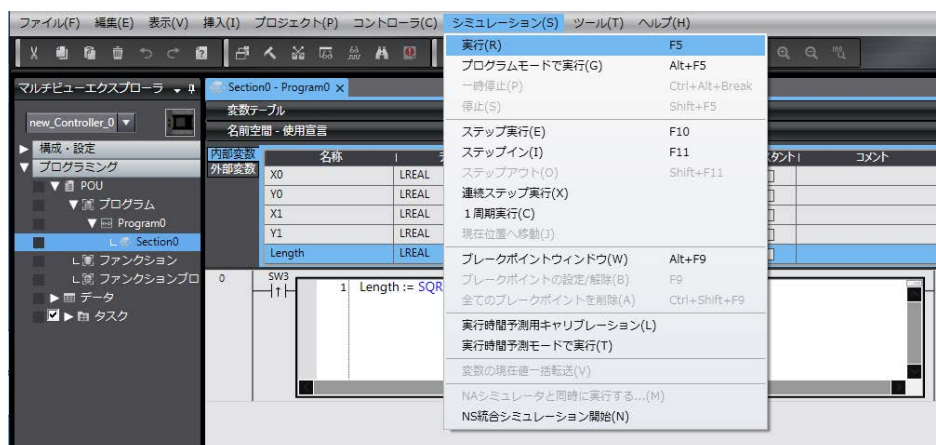


[プロジェクト]-[全プログラムチェック]で、エラーがないことを確認しましょう。
(2件のワーニングはここでは問題ありません)



6-3-4 ST プログラムの動作確認

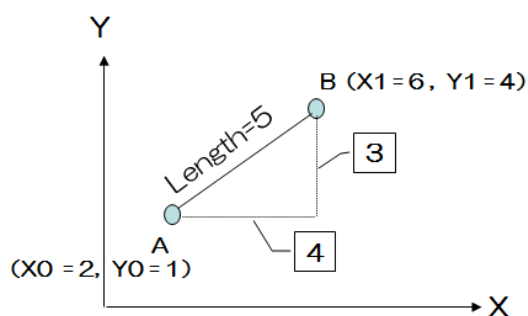
1. シミュレータを起動して動作確認しましょう。
[シミュレーション]-[実行]を選択します。



2. 動作確認の方法

プログラムを同期・転送後、ウォッチウィンドウで、事前に 2 点に値(X0、Y0、X1、Y1)を代入しておき、SW3 を押すと「Length」に計算結果が入るかを確認します。

ウォッチウィンドウ1		
名称	モニタ値	変更
Program0.X0	2	2
Program0.Y0	1	1
Program0.X1	6	6
Program0.Y1	4	4
Program0.Length	5	



3 オフラインにし、プロジェクトファイルの保存とエクスポートを行っておきましょう。

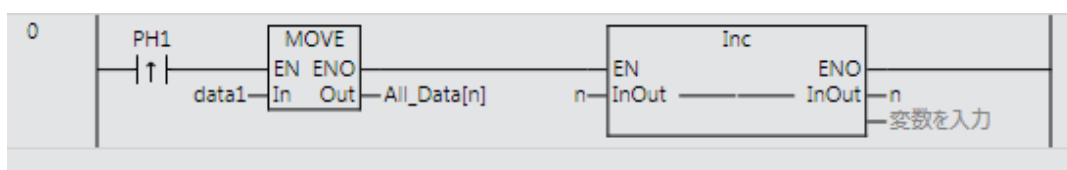


参考

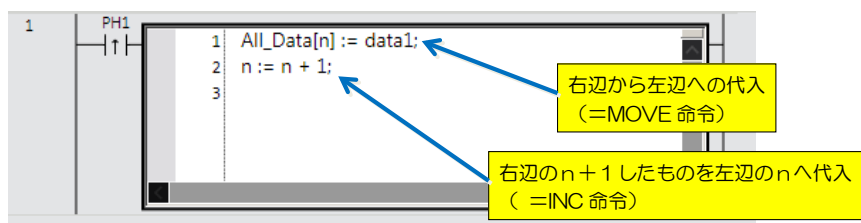
4 章のプログラムを、ST 言語で表記してみましょう。

MOVE 命令や INC 命令などのラダーを記述するのが不得意な方は、ST 言語を使うと、BASIC や C 言語のように数式で記述することもできます。

・ラダープログラミング



・STプログラミング(インラインST)



本誌には主に機種のご選定に必要な
内容を掲載し、ご使用上の注意事項等は
掲載していません。
ご使用上の注意事項等、
ご使用の際に必要な内容につきましては、
必ずユーザーズマニュアルをお読みください。

- 本誌に記載の標準価格はあくまで参考であり、確定されたユーザ購入価格を表示したものではありません。
本誌に記載の標準価格には消費税が含まれておりません。
- 本誌に記載されているアプリケーション事例は参考用ですので、ご採用に際しては機器・装置の機能や安全性を
ご確認の上、ご使用ください。
- 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



クイック オムロン
0120-919-066

携帯電話・PHS・IP電話などではご利用いただけませんので、下記の電話番号へおかけください。

電話 **055-982-5015** (通話料がかかります)

■営業時間：8:00～21:00 ■営業日：365日

●FAXやWebページでもお問い合わせいただけます。

FAX **055-982-5051** / www.fa.omron.co.jp

●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ください。

www.fa.omron.co.jp

緊急時のご購入にもご利用ください。

オムロン商品のご用命は