取扱説明書

形式 BA1C－PAC／CODESYS 解説書

ご利用にあたってのご注意

ご利用に際しては下記をご理解下さい。
・本書とその他の付属書類（取扱説明書、仕様書、カタログ、マニュアルなど（電磁的方法で提供されるものも含む））をすべてご熟読いただき、安全上の注意事項などを遵守して弊社製品（BA1C－PAC）をご利用下さい。
・弊社製品をご利用の際には、お客様の責任の下、お客様ご自身で適合性など（お客様用途での弊社製品の（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、（e）各種規格の遵守など）をご確認いただき、ご利用の可否をご判断下さい。「弊社」は「適合性など」を一切保証いたしません。
・弊社製品が適切に配電・設置されていることをお客様の責任の下、お客様ご自身で、必ず事前にご確認下さい。
・弊社製品をご使用の際には、下記事項を実施して下さい。
（a）定格および性能に対し余裕のある弊社製品のご利用、冗長設計などの安全設計
（b）弊社製品が故障しても、危険を最小にする安全設計
（c）利用者に危険を知らせるための安全対策の構築
（d）弊社製品の定期的な保守
・弊社製品は、一般工業製品向けの汎用品として設計製造されており、次に掲げる用途での使用は意図していません。お客様が弊社製品をこれらの用途に使用される際には、「弊社」は弊社製品に対して一切保証をいたしかねます。
（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・電気などの供給システム、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）

【安全上の注意事項】

人への危害、財産の損害を防止するため、必ずお守りいただくことを説明しています。取り付け、運転、保守・点検の前に必ず本書とその他の付属書類をすべて熟読し、正しくご使用下さい。機器の知識、安全の情報そして注意事項のすべてについて習熟してご使用下さい。また、本書は最終保守責任者のお手元に必ず届くようにして下さい。

■アイコン表記について

誤った使い方をしたときに生じる危害や損害の程度を区分して、説明しています。

⚠ 警 告	「死亡や重傷を負うおそれがある内容」です。
⚠ 注 意	「軽傷を負うことや、財産の損害が発生するおそれがある内容」です。
⃠	「してはいけない内容」です。
❗	「しなければならない内容」です。

なお、⚠注意に記載した事項でも、状況によっては重大な結果に結びつく可能性があります。いずれも重要な内容を記載していますので、必ず守って下さい。

■取扱いについて

⚠ 警 告

❗ 弊社製品の故障や外部要因による異常が発生しても、システム全体が安全側に働くように弊社製品の外部で安全対策を行なって下さい。異常動作により、重大な事故につながるおそれがあります。
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プログラマブルオートメーションコントローラ（以下 PAC）の外部で構成して下さい。PAC の故障により、機械の破損や事故のおそれがあります。なお、リレー出力モジュールのリレー駆動用電源にて外部負荷とのインターロックは行わないで下さい。非常時に出力機器の電源を切る回路を必要とする場合は、弊社製品本体の外部に設けて下さい。モータの正転・逆転など相反する動作を制御する場合は、弊社製品本体の外部にインターロック回路を設けて下さい。
・信号線の断線、瞬時停電による異常信号などに備えて、ご使用者側でフェールセーフ対策を施して下さい。
・PAC は、自己診断機能で異常を検出したときや、運転停止命令を実行したとき、運転を停止して全出力を OFF にします（PAC の設定による）。ただし、PAC の自己診断機能では検出できない、入出力制御部や I／O メモリなどの異常時は、意図しない出力をすることがあります。これらのいずれのときでも、システムが安全側に動作するよう、PAC 外部で対策を施して下さい。

・出力リレーの溶着や焼損、出力トランジスタの破壊などによって、PAC の出力が ON または OFF になったままになることがあります。このとき、システムが安全側に動作するよう、PAC 外部で対策を施して下さい。
・PAC が運転を停止している状態(「停止」モード)においても、CPU ユニットは、I／O リフレッシュを行っています(PAC の設定による)。したがって、以下のいずれかの操作によって、出力ユニットに割り付けられた変数の値を変更する場合、十分に安全を確認してから行って下さい。出力ユニットに接続された負荷が思いがけない動作をするおそれがあります。
  (a) 周辺ツール（パソコンツール）による、PAC へのプログラム転送
  (b) 周辺ツールによる、現在値変更操作
  (c) 周辺ツールによる、強制セット／リセット操作
  (d) ネットワーク上の他の PAC または上位コンピュータから PAC へのアクセス
・次の操作は、設備に影響がないことを確認した上で行って下さい。
  (a) PAC の動作モード切替え（電源投入時の動作モード設定を含む）
  (b) 値の強制セット／リセット
  (c) 現在値や設定値の変更
・作成したユーザプログラム（フロー、ラダープログラムなど）は、十分な動作確認を行った後、本運転に移行して下さい。

⃠ 燃焼性ガスの雰囲気中は使用しないで下さい。爆発の原因となります。

・バッテリの＋／－の逆接続、充電、分解、加熱、火中に投入、ショートはしないで下さい。破裂、発火のおそれがあります。

⃠ 弊社製品を火中に投棄しないで下さい。電池や電子部品などが破裂する原因となります。

・通電中は、弊社製品を分解したり内部や端子に触れたりしないで下さい。感電のおそれがあります。

⚠ 注 意

❗ 異常発熱や発煙を防止するため、弊社製品の保証特性・性能の数値に対し余裕をもたせて使用して下さい。

⃠ 分解、改造はしないで下さい。また、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下さい。火災、故障、誤動作、異常発熱や発煙の原因となります。

なお、弊社製品を輸送するときは、輸送中に過剰な振動や衝撃が加わらないように専用の梱包箱を使用して下さい。

❗ 電線やコネクタは確実に接続して下さい。接続不十分な場合は、異常発熱や発煙の原因となります。

⃠ 電源を入れた状態では施工（接続、取り外しなど）しないで下さい。感電のおそれがあります。
ユニットの着脱は電源を OFF してから行って下さい。感電、誤動作、故障の原因となることがあります。

次のことを行うときは、必ず PAC 本体の電源を OFF にして下さい。
  (a) 弊社製品や電源ユニット、I／O ユニット、通信ユニットを着脱するとき
  (b) 装置を組み立てるとき
  (c) ディップスイッチやロータリスイッチを設定するとき
  (d) ケーブルを接続、配線するとき
  (e) コネクタを取り付けたり、取り外したりするとき

❗ 弊社が指定していない方法で使用すると、弊社製品の保護機能が損なわれることがあります。

・必ず接地（FE 端子）を行って下さい。接地しない場合、感電、誤動作のおそれがあります。
・本書とその他の付属書類に記載の環境で使用して下さい。感電、火災、誤動作の原因となることがありますので、次のような環境での使用は避けて下さい。
  (a) 直射日光が当たる場所
  (b) 急激な温度変化により結露が起こる可能性のある場所、周囲温度や相対湿度が仕様値の範囲を超える場所
  (c) 腐食性ガスや可燃性ガスのある場所
  (d) 塵埃、鉄粉、塩分などが多い場所
  (e) ベンジン、シンナーおよびアルコールなどの有機溶剤や、アンモニア、苛性ソーダなどの強アルカリ物質が付着する可能性のある場所、またはそれらの雰囲気中
  (f) 直接、振動や衝撃が伝わるような場所や水、油、薬品などの飛沫がかかる可能性のある場所
  (g) 高圧線・高圧機器・動力線・動力機器、あるいは無線などの送信部のある機器、または大きな開閉サージの発生する機器の周辺（最低 100 mm）

・強い高周波ノイズを発生する機器から離して取り付けて下さい。
　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下さい。
　(a) 静電気などによるノイズが発生する場所
　(b) 強い電界や磁界が生じる場所
　(c) 放射能を被曝するおそれのある場所
　(d) 電源線が近くを通る場所

配線について
・ケーブルの配線時には、次の点に注意して下さい。
　(a) ケーブルは、動力線、高圧線から離して下さい。
　(b) ケーブルは、折り曲げないで下さい。
　(c) ケーブルを引っ張らないで下さい。
　(d) ケーブルに物を載せないで下さい。
　(e) ケーブルは、必ずダクト内に配線して下さい。
・ご使用になる環境によっては、ノイズフィルタサージアブソーバフェライトコアを取り付けるなどの耐ノイズ対策が必要となる場合があります。
・PAC のベース取り付けねじ、端子台のねじ、ユニットの取り付けねじ、コネクタのねじは、本書で指定した規定トルクで締めて下さい。
・端子台、コネクタ、増設ケーブル、メモリカード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下さい。
・弊社製品側面のコネクタには、弊社製品シリーズ以外は接続しないで下さい。
・定格温度 90℃以上の銅電線を使用して下さい。

■設置について
・本書に従って取り付けて下さい。取り付けに不備があると落下、故障、誤動作の原因となることがあります。
・電線くずなどの異物を入れないで下さい。火災、故障、誤動作の原因となることがあります。
・定格にあった電源を接続して下さい。定格と異なった電源を接続すると火災の原因となることがあります。
・据え付け工事の際には、必ず接地抵抗 100 Ω 以下の D 種（第 3 種）接地として下さい。
・電接地点はできるだけ弊社製品本体の近くとし、接地線の距離を短くして下さい。
・接地を他の機器と共用すると逆効果となる場合がありますので、必ず専用接地として下さい。ご使用になる環境により、接地をすると逆に問題となる場合があります。プラス接地の場合は機能アース端子を接地しないで下さい。
・端子台、コネクタを十分確認してから、装着して下さい。
・配線は圧着端子を付けて下さい。撚り合わせただけの電線を、直接端子台に接続しないで下さい。
・接地された金属に触るなどして人体の静電気を放電されてから、弊社製品に触れて下さい。
・静電気破壊防止のため、コネクタ類のピンを直接さわらないで下さい。
・ノイズの少ない電源を使用するようにして下さい。
・電源線に重畳するノイズに対しては充分なノイズ耐量がありますが、絶縁トランス／絶縁型電源を介することにより、さらにノイズを減衰させることをお薦めします。
・電源事情が悪い場所では特に、定格の電圧や周波数の電源が供給できるようにしてご使用下さい。
・配線、スイッチなどの設定を十分確認してから通電して下さい。
・配線作業は、資格のある専門家が行って下さい。配線を誤ると火災、故障、感電のおそれがあります。
・運転中のプログラム変更、強制出力、運転、停止などの操作は十分安全を確認して行って下さい。操作ミスにより機械の破損や事故のおそれがあります。
・電源投入順序に従って投入して下さい。誤動作により機械の破損や事故のおそれがあります。
・弊社製品本体の起動は、入出力機器、動力機器が立ち上がってから行なって下さい。
・弊社製品本体を停止する場合も弊社製品本体の運転が停止してから入出力機器、動力機器を停止して下さい。
・弊社製品、入出力機器、動力機器への配線は、それぞれ系統を分離して下さい。

■その他

弊社製品は、工業環境に使用する目的で開発／製造された製品です。

・オンラインでのプログラム変更はサイクルタイムが延びても影響がないことを確認してから、実施して下さい。入力信号を読み取れないことがあります。

【はじめに】

このたびは、弊社製品をお買い上げいただき、誠にありがとうございます。
本書は、弊社製品を使用する上で、必要な情報を記載しています。
ご使用の前に、本書とその他の付属書類をすべてご熟読いただき、安全上の注意事項などを遵守して弊社製品をご利用下さい。また、お読みになった後も本書は大切に保管して、いつも手元においてお使い下さい。

■対象となる読者の方々

本書は、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
・PAC 機器の導入を担当される方
・PAC システムを設計される方
・PAC 機器を設置、接続される方
・PAC を導入後、管理される方

■著作権および商標に関する記述

・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です。
・CODESYS は、ドイツ CODESYS GmbH の登録商標です。
・Ethernet は富士フイルムビジネスイノベーション株式会社および米国 Xerox Corporation の登録商標です。
・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（Open DeviceNet Vendor Association）の登録商標です。
・OPC UA は OPC Foundation の登録商標です。
・PROFINET は PROFIBUS&PROFINET International（PI）の登録商標です。
・CANopen は CAN in Automation（CiA）の登録商標です。
・DeviceNet は、ODVA,INC. の登録商標です。
・SDHC、SD ロゴは、SD-3C、LLC の商標です。
・その他の会社および製品名は、各社の商標または登録商標です。

■ご注意

【セキュリティ上の注意事項】

弊社製品をご使用になる場合、例えば、下記の被害を受けるおそれがあります。
・弊社製品を経由した情報漏洩、情報流出の被害
・悪意を持った第三者による弊社製品が不正に操作されることによる被害
・悪意を持った第三者による弊社製品の妨害や停止による被害
・上記被害を防ぐため、お客様の責任の下、下記対策、及び、他のネットワークセキュリティ対策を十分に行なって下さい。
・パソコンが接続されているシステムで弊社製品を使用する場合には、コンピューターウイルスや不正プログラムの感染に対するチェックや駆除を定期的に実施する対策。
・安全性が確保されたネットワーク上で弊社製品を使用するためにファイアウォールなどを使用する対策。
・VPN（Virtual Private NetWork）や専用回線網を構築することにより、セキュリティを強化する対策。
・ユーザー名とパスワードを設定し、ログインできるユーザーを制限することにより、不正な攻撃から守るための対策。
・認証情報（ユーザー名、パスワード）、FTP サーバー情報などをネットワーク上に漏えいさせないため、ユーザー認証でアクセスを制限するなどの対策を実施する対策。
・管理者のパスワードを定期的に変更する対策。
・管理者権限で弊社製品にアクセスした後、すべてのブラウザを確実に閉じる対策。
・十分に安全な場所に弊社製品、ケーブルなどを設置する（容易に破壊される場所には設置しない）対策。
・弊社製品を利用した場合、端末に機密情報などの重要な情報が残る場合あります。弊社製品の使用を止める際には、お客様の責任の下、端末から弊社製品のアンインストール、関連ファイルを消去するなどの対策を実施して下さい。
・また、弊社製品を第三者に譲渡等する場合（例えば、譲渡、貸し渡し、修理、廃棄などの場合）には、お客様の責任の下、弊社製品やご使用するメモリに記録された機密情報などの重要な情報を消去するなど、重要な情報の取り扱いに十分に注意して下さい。

【保証内容】

■保証内容

・保証期間

弊社製品の保証期間は、弊社出荷日から 36 か月とさせていただきます。

・保証範囲

前項の保証期間内に、通常の設置環境での正常な使用状態において、ご購入いただいた弊社製品に万一故障が生じた場合は、納入した製品の代替品との交換または修理を製品の購入場所において無償で実施いたします。

ただし、故障の原因が以下のいずれかに該当する場合は、この保証の対象から除外させていただきます。

(a) お客様を含む弊社以外の者（以下「第三者」といいます）による不適当な使用または取り扱いによる場合
(b) 取扱説明書、仕様書、カタログ等に記載された設計仕様、設置条件などを逸脱した使用、取り扱い若しくは保管による場合
(c) 火災、風水害、地震、落雷その他の天災事変、若しくは公害、塩害、煙害、腐食性ガス、異常電圧などの不可抗力に起因する場合
(d) 第三者による当該製品への改造または修理に起因する場合
(e) 指定外の電源使用や他の接続機器の不具合など弊社製品以外の原因により生じた場合
(f) 法令で義務づけられた保安・保全業務を怠ったことに起因する場合
(g) 警報装置の動作時などに必要とされる措置を怠ったことに起因する場合
(h) 弊社の正規販売店以外から購入されたあるいは購入時に既使用の弊社製品の場合
(i) 部品若しくは消耗品の自然減耗、費消または寿命による場合
(j) 弊社出荷当時の科学・技術水準では、予見できなかった場合
(k) その他、弊社の客観的な判断により弊社の責に帰さないと判断される場合

なお、ここでいう保証は弊社製品単体の保証を意味するものであり、弊社製品の故障により誘発されるシステムおよび接続機器などに関する損害につきましては、補償はいたしかねます。

・弊社の保証範囲外の故障

(a) 前項の保証範囲に含まれない弊社製品の故障に関しては、特にご要望の場合、修理など有償にて対応させていただきます。
(b) 故障の原因調査および報告書作成は原則としてお受けいたしかねます。ただし、特にご要望の場合は、その実施の諾否を含めて協議させていただきます。なお、これにより原因調査等を実施する場合は原則として有償とさせていただき、別途実費を申し受けます。

■責任の制限

1. 弊社の製品に関する保証は、弊社製品単体の保証に限定されるものとし、代替品との交換または修理による対応に限らせていただきます。
2. 弊社製品の故障に起因して誘発される計測・制御システムや接続機器などについての損害に関しては、弊社は責任を負いません。製品のご返品につきましても、当該製品の販売価格を超えた金銭賠償等はいたしません。
3. 弊社製品の故障に起因して派生的に生じたいかなる損害（逸失利益、特別損害、間接損害、付随的損害を含む）に関しては、弊社は責任を負いません。
4. 前 3 項の責任の制限は、弊社に対する損害賠償またはその他の請求がこの保証規定、不法行為（過失責任および製造物責任を含む）、契約上の請求またはそれ以外の請求原因にもとづくものであるか否かに拘わらず適用いたします。ただし、法規上の強行規定により、上記の責任の制限が適用されない場合があります。

■製品ご使用時の注意事項

1. 弊社製品は一般産業機器として設計、製造されているものであるため、原子力制御設備、放射線関連機器、鉄道･航空･車両設備、航空・宇宙機器、海中設置機器、若しくは生命維持のための医療機器など、極めて高い信頼性と安全性が要求される用途には使用しないで下さい。
2. 使用されるシステムにおいて、お客様ご自身が、弊社製品の定格・性能に対し余裕をもった使い方や、システム全体に対する警報機器、安全機器の設置、安全性を確保した設計を行うなどの安全対策を講じて下さい。
3. 弊社の製品を他社の製品と組み合わせて使用される場合、関連する規格・法規または規制、ならびに、使用されるシステム・機械・装置への弊社製品の適合性は、お客様の責任においてご確認下さい。適合性に関する保証は一切いたしかねます。
4. 弊社製品が正しく使用されず不測の損害が生じることがないよう、取扱説明書ならびに仕様書を必ずご確認いただき、その安全に関する使用上の禁止事項および注意事項をすべてご理解いただいたうえご使用下さい。それらの禁止事項および注意事項に反する使用をされた場合、弊社は一切、当該製品の品質・性能・機能および安全性を保証いたしません。

■仕様の変更

弊社製品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。

■保証内容の変更
弊社が適当と判断する方法により、お客様に通知または周知することにより、本保証内容の一部若しくは全部を変更できるものとし、この場合、変更日以降は変更後の保証内容が適用されるものとします。

■サービスの範囲
弊社製品の価格には、技術員派遣などのサービス費用は含まれておりません。技術員の派遣などは、ご要望により別途ご相談させていただきます。
なお、原子力管理区域（放射線管理区域）および被爆放射能が原子力管理区域レベル相当の場所においての技術員派遣の対応はいたしません。

■表示価格
記載の表示価格には、消費税は含まれておりません。ご注文の際には、別途消費税を頂戴いたします。

■適用範囲
以上の保証規定は、弊社製品の日本国内での使用に限り適用されます。日本国外でのご使用につきましては、弊社カスタマセンターまでお問合せ下さい。

【輸出管理】

ご購入いただいた弊社製品が、「安全保障輸出管理」に関する輸出許可の対象貨物又は技術に該当する場合は、法令の定めに従って、輸出に際し事前に経済産業大臣の許可を得ることが必要です。許可なく又は不正に輸出されると刑事罰等の対象となります。

【動作確認済み環境】

動作確認は以下の環境で実施しています。

項　目	内　容
パソコン	以下の OS が正常に動作する PC／AT 互換機
OS	Windows 11 Pro（64bit 版）24H2
言語	日本語
ブラウザ	Google Chrome 137.0.7151.69（Official Build）（64 ビット） Microsoft Edge 137.0.3296.62（公式ビルド）（64 ビット）
CODESYS	3.5.21.10

【対応バージョン】

本書は、以下のバージョンに対応しています。

種　別	形　式	対応バージョン
プログラマブルオートメーションコントローラ	BA1C－PAC－A	1.0.0.16 以降
プログラミングツール	C－CDS35－21－10	3.5.21.10
ESSENTIAL パッケージ	MG_ESSENTIAL	3.0.0.0 以降
BA1C－PAC パッケージ	MG_BA1C－PAC	3.0.0.0 以降

目次

1. IEC61131

IEC61131 は、PLC（Programmable Logic Controller、プログラマブルロジックコントローラ）に関する国際標準規格であり、PLC の仕様、機能、プログラミング方法などを体系的に定めたものです。国際電気標準会議（IEC）によって策定され、以下のような複数のパートで構成されています。

1. 一般事項
   用語や基本概念、PLC システムの一般仕様を定義しています。
2. ハードウェア要件
   入出力特性や電気的仕様など、ハードウェアに関する要件を定義しています。
3. プログラミング言語
   5 種類の標準プログラミング言語、プログラム構造、データ型などソフトウェアに関する仕様を定義しています。
4. ユーザガイドライン
5. 通信
6. 機能安全

1.1 IEC61131-3

IEC61131-3 は、PLC のプログラミング言語の仕様を定めたものです。

1.1.1 目的

IEC61131-3 は PLC ごとに異なっていたプログラミング手法を統一し、プログラムの互換性や再利用性を高めることを目的としています。これにより、異なるメーカや機種間でも、共通のルールに基づいてプログラムを開発できるようになります。

1.1.2 内容

IEC61131-3 では、次のようなプログラミング要素と 5 つの言語を定義しています。この規格に準拠することで、保守性の高いプログラム設計、教育の標準化、異機種間での共通化が実現できます。

■プログラミング要素

・データ型や構造体の定義
・ファンクション（Function）、ファンクションブロック（Function Block）、プログラム（Program）の構造
・変数スコープと名前空間（Namespace）の管理

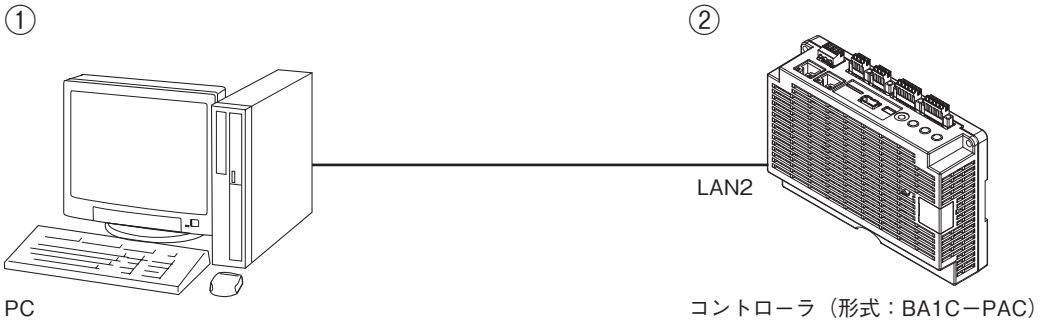
■言語

・FBD（Function Block Diagram）
・IL（Instruction List）
・LD（Ladder Logic Diagram）
・SFC（Sequential Function Chart）
・ST（Structured Text）

2. CODESYS 開発環境

2.1 CODESYS

CODESYS は国際規格 IEC61131-3 に準拠したプログラミングおよび実行が可能なツール（プログラミングシステム）です。プログラミングエディタ、HMI（Human Machine Interface）開発環境、オンラインデバッグ機能が統合された IDE System（Integrated Development Environment、統合開発環境）と実行エンジンである Target Runtime System（ターゲット）で構成されています。



項　目	内　容
① IDE System （統合開発環境）	プログラムの作成やデバッグ、デバイスの設定などを行うためのツールです。プログラミングエディタ、コンパイラ、HMI エディタ、オンラインデバッグ機能などが含まれています。
② Target Runtime System （ターゲット）	作成したソフトロジックを実行する環境です。

2.2 インストール

ここでは CODESYS で開発を行うために必要なソフトウェアのインストール方法を説明します。

■IDE System（統合開発環境）
PLC のプログラム開発、ビルド、デバッグを行うためのツールです。

■パッケージ
デバイスサポート、テンプレート、追加機能などを含む拡張コンポーネントで、IDE System に機能を追加します。

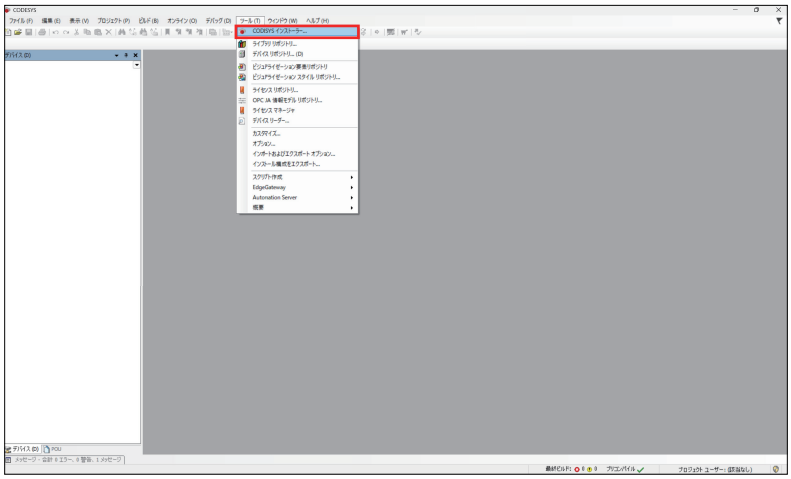
■ライブラリ
よく使う処理や制御ロジックを再利用可能なモジュールとしてまとめたもので、開発効率を向上させます。必要なライブラリはパッケージに含まれていますが、特定の用途に応じて別途インストールすることもできます。

2.2.1 IDE System（統合開発環境）

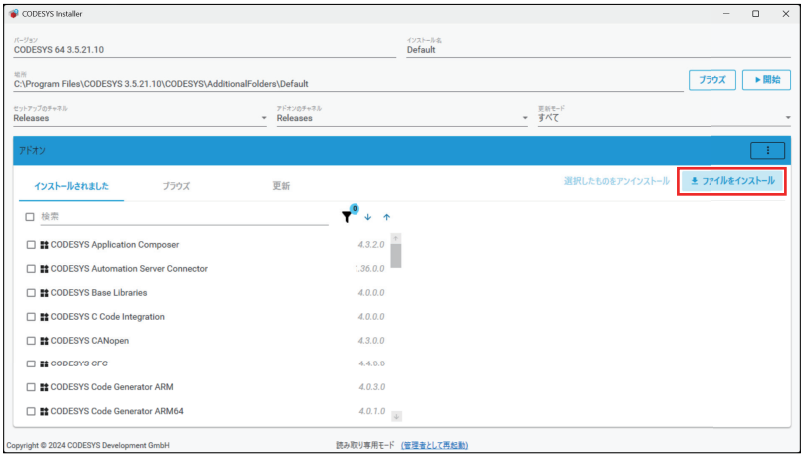
プログラミングツール（形式:C-CDS35）を弊社のホームページよりダウンロードし､任意のフォルダに展開して下さい。展開したフォルダ内の「CODESYS 64 3.5.21.10.exe」を実行して、ダイアログに従いインストールして下さい。

2.2.2 パッケージ

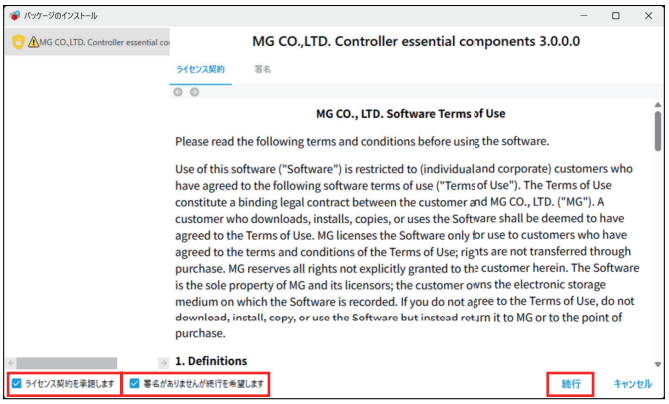
CODESYS IDE のメニューから「ツール」→「CODESYS インストーラ…」をクリックします。



CODESYS Installer の「ファイルをインストール」をクリックし、インストールするパッケージファイル（.package）を選択してパッケージをインストールして下さい。



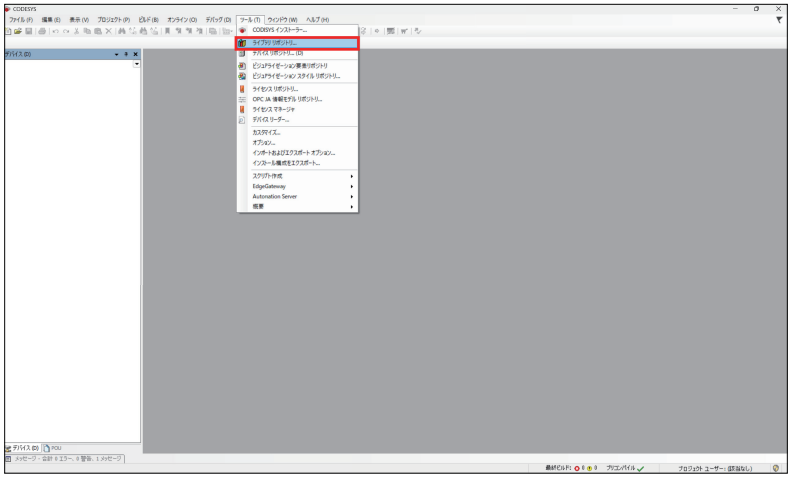
当社パッケージをインストールする際は画面の表示内容を確認のうえ、「ライセンス契約を承諾します」および「署名がありませんが続行を希望します」にチェックを付け、「続行」をクリックします。



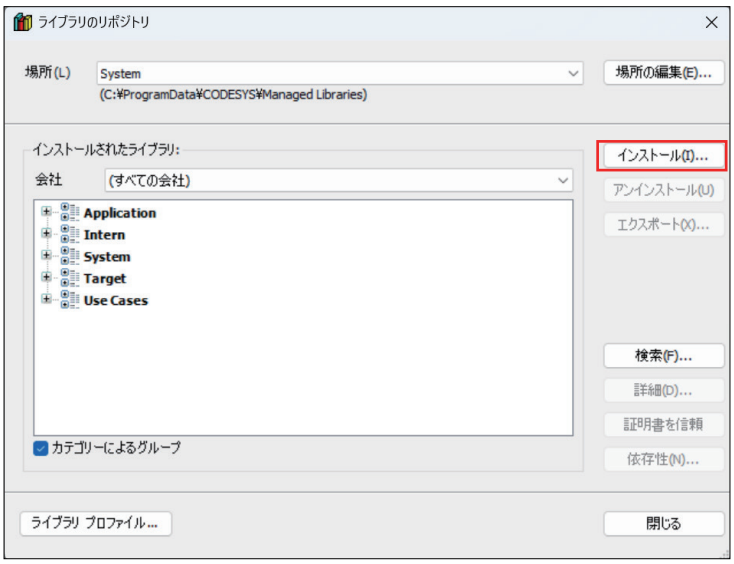
パッケージのインストールが完了すると、以下の画面が表示されます。

2.2.3 ライブラリ

CODESYS IDE のメニューから「ツール」→「ライブラリリポジトリ…」をクリックします。



ライブラリリポジトリ画面の「インストール…」をクリックし、インストールするライブラリファイル（.library）を選択してライブラリをインストールして下さい。



2.3 アンインストール

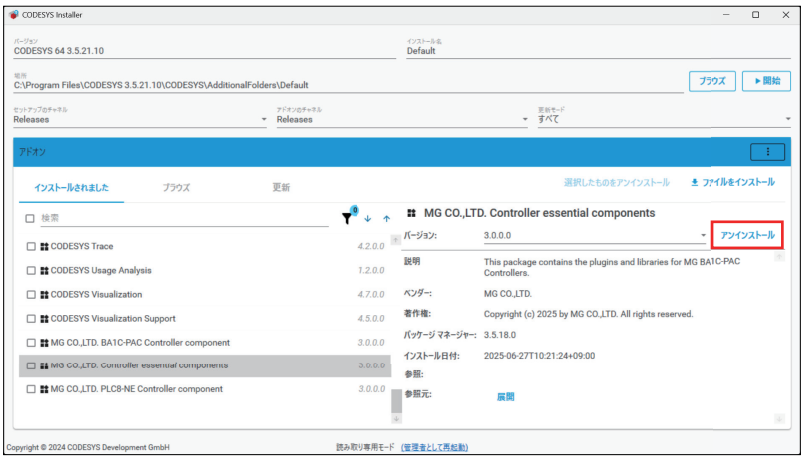
ここでは CODESYS IDE 本体や CODESYS IDE にインストールされているライブラリのアンインストール方法を説明します。

2.3.1 IDE System（統合開発環境）

コントロールパネルから、「プログラム」→「プログラムと機能」を選択します。一覧の中から「CODESYS 64 3.5.21.10」を選択し、アンインストールして下さい。

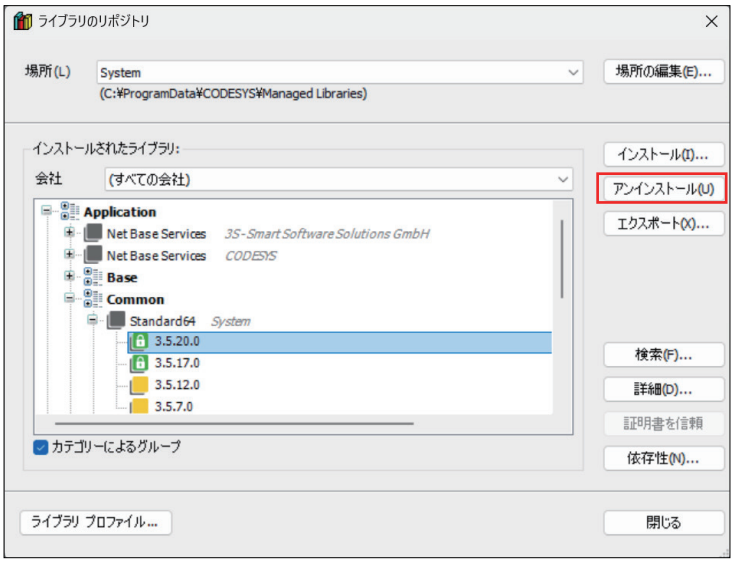
2.3.2 パッケージ

CODESYS IDE のメニューから「ツール」→「CODESYS インストーラ…」をクリックします。
CODESYS Installer でアンインストールしたいパッケージを選択し、「アンインストール」をクリックして下さい。

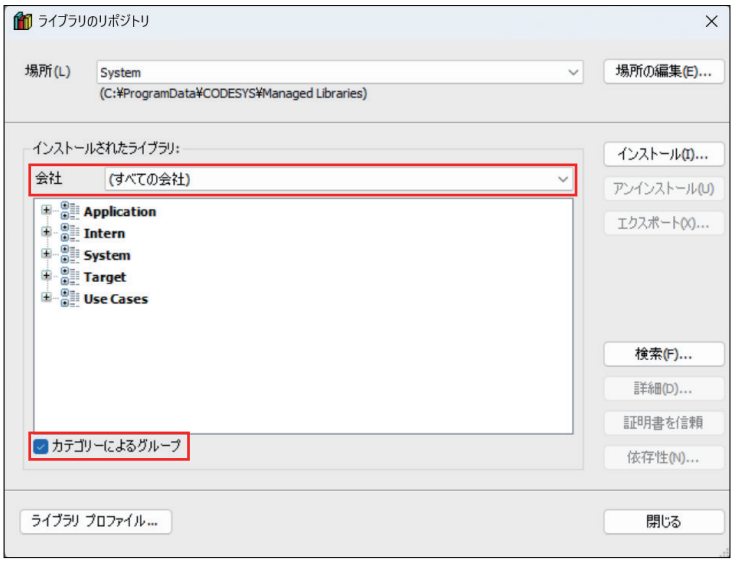


2.3.3 ライブラリ

CODESYS IDE のメニューから「ツール」→「ライブラリリポジトリ…」をクリックします。「インストールされたライブラリ」からアンインストールしたいライブラリを選択し、「アンインストール」をクリックして下さい。



表示されているライブラリの一覧は「会社」や「カテゴリーによるグループ」でフィルタリングされています。目的のライブラリが見つからない場合は適切なフィルタリングを選択して下さい。

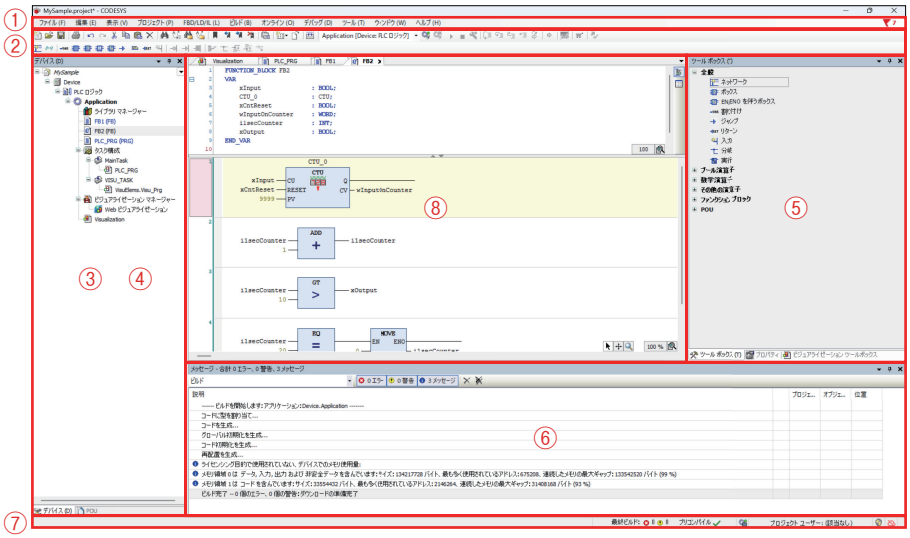


2.4 CODESYS 操作説明

2.4.1 起動

Windows スタートメニュー内の「すべてのアプリ」→「CODESYS」→「CODESYS V3.5 SP21 Patch 1」を起動します。
または、インストール時にデスクトップに作成される以下のショートカットからも起動できます。

2.4.2 画面構成



項　目	内　容
①メニューバー	「ファイル」「編集」「表示」「プロジェクト」などの各種メニューが並び、さまざまな操作機能を使用できます。
②ツールバー	よく使用するコマンドのショートカットアイコンが並び、ワンクリックで素早く操作を実行できます。
③デバイスツリー	プロジェクトに含まれるデバイスや通信設定などが階層的に表示され、各デバイスの設定や構成を編集できます。
④ POU ツリー	POU（プログラム構成ユニット）が階層的に表示され、各プログラム、ファンクション、ファンクションブロックを編集できます。
⑤サイドバー	プロパティやツールボックスなどの各種機能を、タブで切り替えて使用します。
⑥メッセージウィンドウ	コード生成時のエラー、警告、メッセージなどが一覧で表示され、トラブルシューティングに使用します。
⑦ステータスバー	現在のプロジェクト状態、通信状態など、作業状況や環境に関する情報が表示されます。
⑧エディタウィンドウ	プログラムコードや設定内容を表示・編集する領域で、使用する機能や言語に応じてエディタが切り替わります。

2.4.3 デバイスツリー

デバイス（コントローラ）で定義されているすべてのリソースが表示されます。

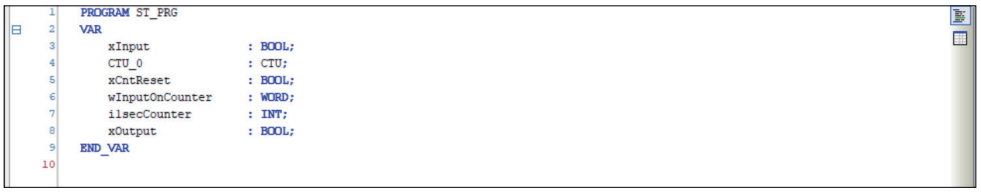
代表的な表示内容

「プロジェクト」：例 MySample
　「デバイス」：例 Device
　　「リソース」：例 PLCロジック
　　「アプリケーション」：例 Application
　　・ライブラリマネージャー
　　・グローバル変数定義：例 GVL_PLC
　　・POU（プログラム、ファンクションブロック、ファンクション）：例 PLC_PRG
　　・DUT（構造体、列挙など）
　　・タスク構成
　　など

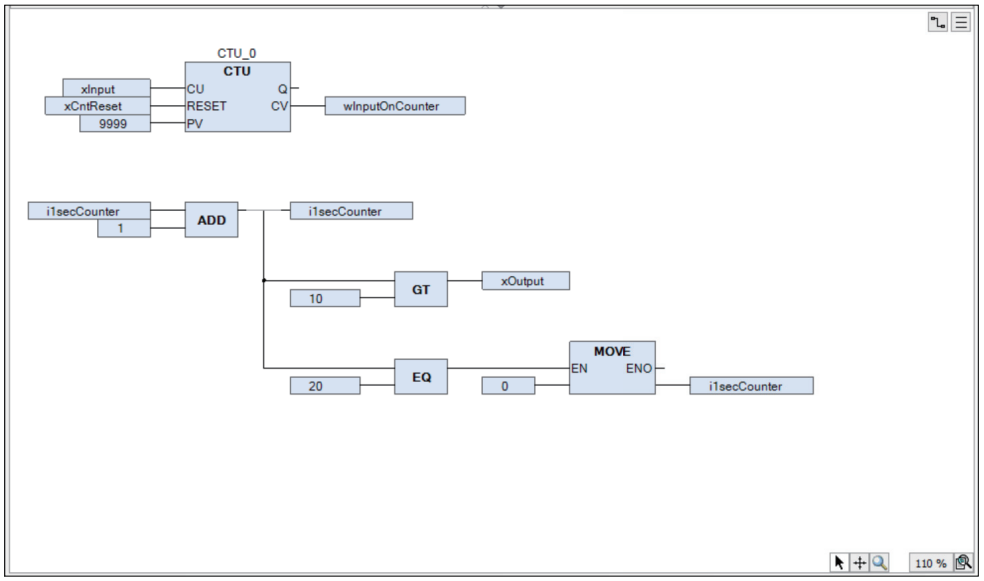
2.4.4 エディタウィンドウ

プログラムを作成する際に使用します。プログラム内で使用される変数を宣言する「変数宣言部」とプログラムの処理を記述する「プログラム本体部」の 2 つのウィンドウで構成されています。

「変数宣言部」の例



「プログラム本体部」の例
CFC 言語



ST 言語

```
1   CTU_0(CU := xInput, RESET := xCntReset, PV := 9999, CV => wInputOnCounter);
2
3   ilsecCounter := ilsecCounter + 1;
4
5   IF ilsecCounter > 10 THEN
6       xOutput := TRUE;
7   ELSE
8       xOutput := FALSE;
9   END_IF
10
11  IF ilsecCounter = 20 THEN
12      ilsecCounter := 0;
13  END_IF
14
```

2.4.5 オンラインモード

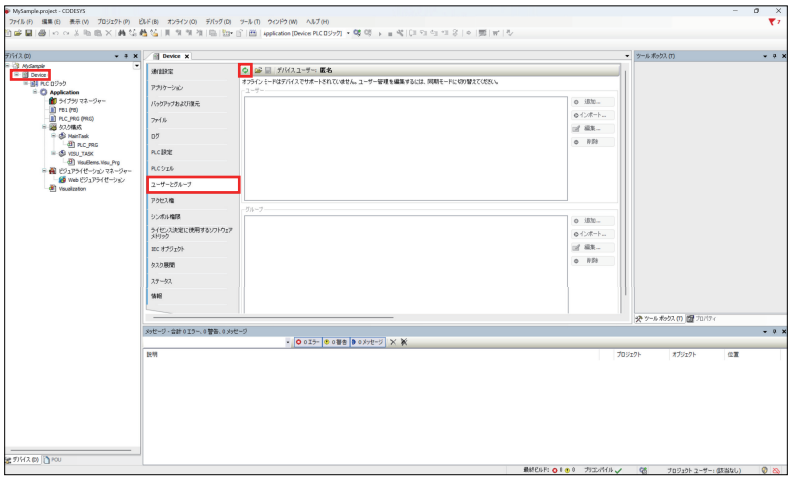
オンラインモードの情報は、画面下部のステータスバーに表示されます。

項　目	内　容
運転	プログラムを実行中です。
停止	プログラムを停止中です。
BPでの停止	プログラムがブレークポイントで停止中です。

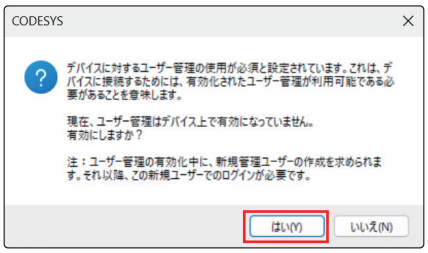
項　目	内　容
プログラムはロードされました	プログラムはデバイスにロード済みです。
プログラムは変更されていません	デバイス内のプログラムは CODESYS IDE 内のプログラムと一致しています。
プログラムの変更状態 (オンライン変更)	デバイス内のプログラムは CODESYS IDE 内のプログラムと異なるため、オンライン変更が必要です。
プログラムの変更状態 (完全ダウンロード)	デバイス内のプログラムは CODESYS IDE 内のプログラムと異なるため、完全なダウンロードが必要です。

2.4.6 ユーザ管理

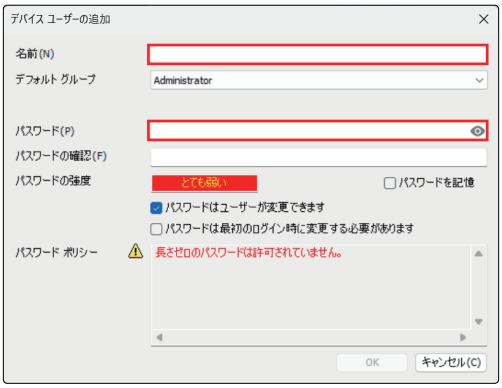
デバイスツリーの「Device（MG BA1C－PAC)」をダブルクリックして、デバイスの通信設定画面を開きます。「ユーザとグループ」タブを開き、「同期」ボタンをクリックしてコントローラと接続します。



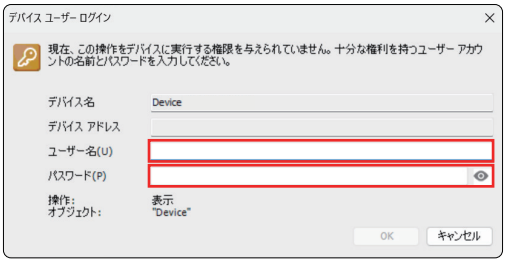
コントローラへ初回接続時は、新規にユーザ登録を行います。「はい」をクリックして、新規ユーザ登録を行います。



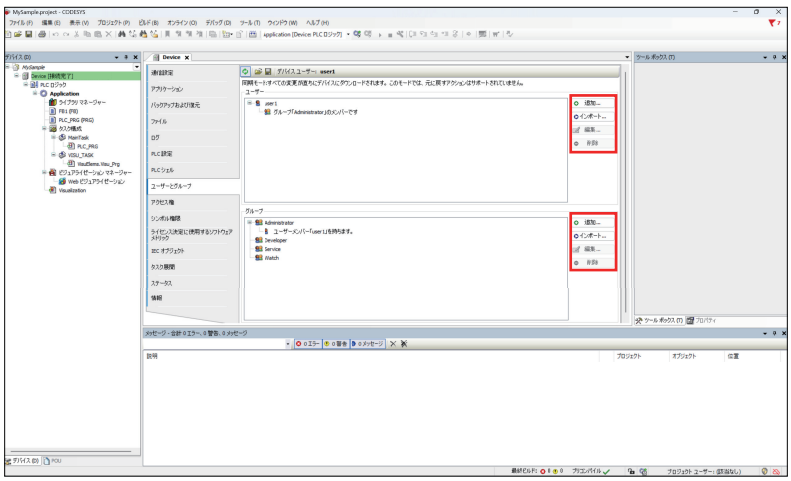
ユーザ名とパスワードを登録します。



登録したユーザ名とパスワードを入力して、コントローラに接続します。



「ユーザとグループ」ではユーザの追加や削除、グループの変更を行えます。



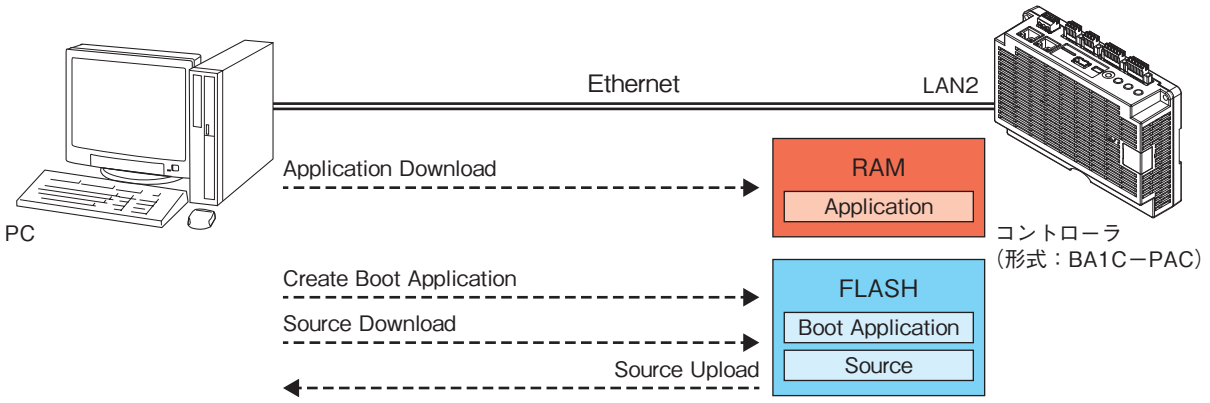
3. プログラミング

3.1 コントローラを使用する前に行っていただきたいこと

本コントローラは、RTC（Real Time Clock、リアルタイムクロック）機能を搭載しています。コントローラを初めて使用する場合や長期間未使用であったコントローラを再び使用する場合には、日付と時刻が正確でない可能性があります。使用開始前に日付と時刻を確認し、必要に応じて設定して下さい。
詳細な手順はコントローラの取扱説明書（NM-7345-A）を参照して下さい。

3.2 CODESYS IDE との接続

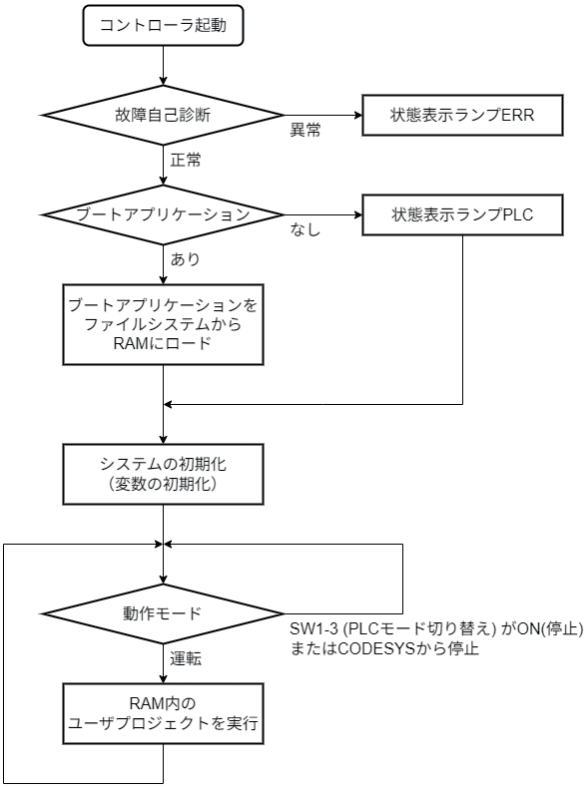
本コントローラは、CODESYS IDE がインストールされた PC と Ethernet 通信で接続します。
CODESYS IDE で接続する前に、PC からコントローラへの通信が可能であることを、ping コマンドなどを使用して事前に確認することをお勧めします。



3.3 アプリケーション

3.3.1 実行

起動時に故障自己診断が行われ、異常が検知されなければファイルシステムに格納されたブートアプリケーションがRAM に転送されます。その後、ユーザプロジェクトは保持変数（RETAIN）および持続変数（PERSISTENT）以外の変数が初期化され、「運転」モードに切り替わります。



3.3.2 サイクルタイム

サイクルタイムは、アプリケーション（IEC タスク）の開始から次の開始までの時間です。
アプリケーションにループ処理や時間待ち合わせファンクションの呼び出しがある場合は、動作時間およびサイクルタイムがその分だけ長くなります。
アプリケーションの実行中は、入出力データやタイマ値（ファンクションブロック TON など）の更新が行われません。これらの値の更新はアプリケーションの最後に行われるため、アプリケーション処理中のイベント待ちやループ処理中のタイムアウト検知にタイマを使用することはできません。

3.3.3 ブートアプリケーションの起動

ブートアプリケーションとして登録されたアプリケーションは、コントローラ起動時に自動的にファイルシステムからRAM に読み込まれて実行されます。

■ブートアプリケーションの登録
コントローラにログイン後、メニューから「オンライン」→「ブートアプリケーションの生成」をクリックすると、現在のアプリケーションがブートアプリケーションとして登録されます。

■ブートアプリケーションの解除
コントローラにログイン後、メニューから「オンライン」→「リセット（PLC 初期化）」をクリックすると、ブートアプリケーションが解除されます。

3.4 オンラインコマンドと変数の状態

コントローラにログイン後、メニュー「オンライン」または「デバッグ」の各コマンドを実行した後の変数の状態を以下の表に示します。

○＝値が保持されます　×＝値が初期化されます

オンラインコマンド	ローカル変数	保持変数（RETAIN）	持続変数（PERSISTENT）	アプリケーション	ブートアプリケーション	ソースファイル
ダウンロード	×	×	○	更新	○*1	○*1
オンライン変更	○	○	○	更新	○*1	○*1
停止	○	○	○	○	○	○
ウォームリセット	×	○	○	○	○	○
コールドリセット	×	×	○	○	○	○
電源再起動	×	○	○	×	○	○
リセット（PLC 初期化）	×	×	×	×	×	○

＊1、コントローラの初期設定では、ダウンロードおよびオンライン変更時にブートアプリケーションの生成は行わないため、値が保持されます。ブートアプリケーションの設定変更方法は「6.3.4 ターゲットにブートアプリケーションを生成する」を参照して下さい。

4. プログラミング要素

4.1 POU（Program Organization Unit）

POU（Program Organization Unit、プログラム構成ユニット）はプログラムの最小単位です。以下の 3 種類があります。

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

ファンクションは内部状態の記憶を持たず、すべての入力変数が同じであれば常に同じ結果を返します。一方、ファンクションブロックは内部状態を記憶（インスタンス化）するため、呼び出し毎に返す結果を変化させることができます。また、プログラムはユーザプログラム（アプリケーション）の最上位に位置し、唯一のインスタンスを持つファンクションブロックのようなものです。実行権（タスクからの呼び出し）を与えることができます。

ファンクションとファンクションブロックで使用（宣言）可能なパラメータの違いを以下の表に示します。
○＝使用可能　×＝使用不可

パラメータ	ファンクション（FUNCTION）	ファンクションブロック（FUNCTION_BLOCK）
入力変数	○	○
出力変数	×	○
入出力変数	×＊1	○
戻り値	○	×
ローカル変数と出力変数の保持	×	○

＊1、IEC61131-3 では使用できませんが、CODESYS では拡張機能として可能です。

例（ファンクションブロック）

4.2 VAR（Variable）

変数の一覧を以下に示します。

種　類	キーワード
ローカル変数	VAR
入力変数	VAR_INPUT
出力変数	VAR_OUTPUT
入力／出力変数	VAR_IN_OUT
グローバル変数	VAR_GLOBAL
一時変数	VAR_TEMP
静的変数	VAR_STAT
インスタンス変数	VAR_INST

属性の一覧を以下に示します。

種　類	キーワード
定数	CONSTANT
持続	PERSISTENT
保持	RETAIN

4.3 データ型

4.3.1 基本データ型

型	説　明	データ範囲
BOOL	真偽(ブール)	TRUE、FALSE
BIT[*1]	ビット	TRUE(1)、FALSE(0)
BYTE	長さ 8 のビット列(バイト)	16#00～16#FF
WORD	長さ 16 のビット列(ワード)	16#0000～16#FFFF
DWORD	長さ 32 のビット列(ダブルワード)	16#00000000～16#FFFFFFFF
LWORD	長さ 64 のビット列(クワッドワード)	16#0000000000000000～16#FFFFFFFFFFFFFFFF
SINT	単精度(8 ビット)整数	-128～+127
USINT	符号なし単精度(8 ビット)整数	0～+255
INT	(16 ビット)整数	-32,768～+32,767
UINT	符号なし(16 ビット)整数	0～+65,535
DINT	倍精度(32 ビット)整数	-2,147,483,648～+2,147,483,647
UDINT	符号なし倍精度(32 ビット)整数	0～+4,294,967,295
LINT	64 ビット整数	-9,223,372,036,854,775,808～+9,223,372,036,854,775,807
ULINT	符号なし 64 ビット整数	0～+18,446,744,073,709,551,615
TIME	32 ビット持続時間	T#0ms～T#49d17h2m47s295ms(4,294,967,295ms) 分解能 1ms
LTIME	64 ビット持続時間	LTIME#0ns～ LTIME#213503d23h34m33s709ms551us615ns (18,446,744,073,709,551,615ns) 分解能 1ns
DATE	32 ビット日付	D#1970-01-01～D#2106-02-07 分解能 1 日
LDATE	64 ビット日付	LD#1970-01-01～LD#2262-4-11 分解能 1 日
DATE_AND_TIME	32 ビット時刻	TOD#0:0:0.000～TOD#23:59:59.999 分解能 1s
LDATE_AND_TIME	64 ビット時刻	LTOD#0:0:0.000000000～LTOD#23:59:59.999999999 分解能 1ns
TIME_OF_DAY	32 ビット日付時刻	DT#1970-1-1-0:0:0～DT#2106-2-7-6:28:15 分解能 1ms
LTIME_OF_DAY	64 ビット日付時刻	LDT#1970-1-1-0:0:0.000000000～ LDT#2262-4-11-23:47:16.854775807 分解能 1ns
REAL	単精度実数(32 ビット浮動小数) (IEE754)	± 0.0000001～± 9,999,999 有効桁数 7 桁[*2]

LREAL	倍精度実数(64 ビット浮動小数) (IEE754)	± 0.000000000000001～± 999,999,999,999,999 有効桁数 15 桁*2
STRING	可変長 1 バイト文字列(ASCII 文字)	最大 255 文字(初期 80 文字)
WSTRING	可変長 2 バイト文字列(UNICODE 文字)	最大 255 文字(初期 80 文字)

＊ 1、CODESYS 固有のデータ型です。

＊ 2、演算結果には誤差（桁落ちや、小数点位置が異なる 2 つの値の演算による有効桁数の情報落ち）が発生することがあります。
例えば、REAL 型で 1.1 ＋ 2.2 を演算すると期待される結果は 3.3 ですが、実際には 3.3000002 になります。

4.3.2 配列（Array）

構文

```
<配列名> : ARRAY [<ll1> .. <ul1>, <ll2> .. <ul2>, <ll3> .. <ul3>] OF <データ型>
```

ll1, ll2, ll3 は各次元の下限値、ul1, ul2, ul3 は各次元の上限値であり、それぞれ整数で指定します。

使用例（変数宣言部）

```
aiTemp1    : ARRAY [1 .. 5] OF INT := [1, 2, 3, 4, 5];
aiTemp2    : ARRAY [1 .. 2, 3 .. 4] OF INT := [1, 3(7)];  * [1, 7, 7, 7]
aiTemp3    : ARRAY [1 .. 2, 2 .. 3, 3 .. 4] OF INT := [2(0), 4(4), 2, 3];  * [0, 0, 4, 4, 4, 4, 2, 3]
aiTemp4    : ARRAY [1 .. 10] OF INT := [1, 2];  * [1, 2, 0, 0, 0, 0, 0, 0, 0, 0]
```

使用例（構造体の配列）

```
TYPE ST_Sample
STRUCT
    iData1      : int;
    iData2      : int;
    dwData3     : dword;
END_STRUCT
END_TYPE

astSample       : ARRAY [1 .. 3] OF ST_Sample := [(iData1 := 1, iData2 := 10, dwData3 := 4723),
                                                  (iData1 := 2, iData2 := 0, dwData3 := 299),
                                                  (iData1 := 14, iData2 := 5, dwData3 := 112)];
```

4.3.3 構造体（Structure）

構文

```
TYPE <構造体名>:
STRUCT
    <変数名1>   : <データ型1>;
    <変数名2>   : <データ型2>;
    ...
END_STRUCT
END_TYPE
```

定義例

```
TYPE ST_PolygonLine:
STRUCT
    aiStart     : ARRAY [1 .. 2] OF INT;
    aiPoint1    : ARRAY [1 .. 2] OF INT;
    aiPoint2    : ARRAY [1 .. 2] OF INT;
    aiPoint3    : ARRAY [1 .. 2] OF INT;
    aiPoint4    : ARRAY [1 .. 2] OF INT;
    aiEnd       : ARRAY [1 .. 2] OF INT;
END_STRUCT
END_TYPE
```

使用例（変数宣言部）

```
stPoly1   : ST_PolygonLine := (aiStart := [3, 3],
                               aiPoint1 := [5, 2],
                               aiPoint2 := [7, 3],
                               aiPoint3 := [8, 5],
                               aiPoint4 := [5, 7],
                               aiEnd := [3, 5]);
```

構造体のメンバは次の構文でアクセスできます。
＜構造体名＞.＜メンバ名＞

使用例（プログラム本体部）

```
aiCurPos := stPoly1.aiStart;
```

BIT変数は構造体のメンバとしてのみ使用可能です。各BIT変数は構造体内で名前付きの1ビットとして定義され、1ビット分のメモリを占有します。
BIT 変数に対して参照（Reference）やポインタ（Pointer）を使った宣言はできません。BIT 変数を要素とする配列も宣言できません。

4.3.4 共用体（Union）

構文

```
TYPE ＜共用体名＞:
UNION
    ＜変数名1＞   :＜データ型1＞;
    ＜変数名2＞   :＜データ型2＞;
    ...
END_UNION
END_TYPE
```

定義例

```
TYPE U_Data:
UNION
    iData     : INT;
    strData   : STRING;
END_UNION
END_TYPE
```

使用例（変数宣言部）

```
uData : U_Data;
```

使用例（プログラム本体部）

```
uData.strData := 'ABC';
```

4.3.5 列挙（Enumeration）

構文

```
TYPE ＜列挙型名＞:
{
    ＜ラベル1＞,
    ＜ラベル2＞,
    ...
}＜データ型＞
END_TYPE
```

定義例

```
TYPE E_Color:
{
    Red := 16#FFFF0000,
    Green := 16#FF00FF00,
    Blue := 16#FF0000FF
}DWORD
END_TYPE
```

使用例（プログラム本体部）

```
dwColor := E_Color.Red;
```

4.3.6 参照（Reference）

構文

```
<変数名>        : REFERENCE TO <データ型> ;
```

使用例（変数宣言部）

```
riData    : REFERENCE TO INT ;
```

4.3.7 ポインタ（Pointer）

構文

```
<変数名>        : POINTER TO <データ型> ;
```

使用例（変数宣言部）

```
piData    : POINTER TO INT ;
pfbSum    : POINTER TO FB_SUM ;
```

4.3.8 範囲型（Subrange）

構文

```
<変数名>        : <データ型>（<ll> .. <ul>);
```

ll は変数の下限値、ul は上限値です。

使用例（変数宣言部）

```
iRange : INT (-100 .. 100);
uiLimit: UINT（10 .. 20);
```

4.3.9 エイリアス（Alias）

構文

```
TYPE <エイリアス名> : <データ型>;END_TYPE
```

定義例

```
TYPE MEMBER : ARRAY[ 0 .. 10] OF STRING(50); END_TYPE
```

使用例（プログラム本体部）

```
memberGroupA : MEMBER;
```

4.4 DUT（Data Unit Type）

DUT（Data Unit Type、データユニットタイプ）はユーザが定義できるデータ型（ユーザ型）です。基本型や既に定義されたユーザ型から新しいユーザ型を作成できます。

種　類	構　文
構造体 Structure	TYPE ＜構造体名＞: STRUCT 　＜変数名１＞　:＜データ型１＞; 　＜変数名２＞　:＜データ型２＞; 　... END_STRUCT END_TYPE
列挙 Enumeration	Enumeration TYPE ＜列挙型名＞: { 　＜ラベル１＞, 　＜ラベル２＞, 　... }＜データ型＞ END_TYPE
エイリアス Alias	TYPE ＜エイリアス名＞:＜データ型＞;END_TYPE
共用体 Union	TYPE ＜共用体名＞: UNION 　＜変数名１＞　:＜データ型１＞; 　＜変数名２＞　:＜データ型２＞; 　... END_UNION END_TYPE

4.5 リテラル

数値などの定数を直接表記する場合はリテラルを用います。リテラルは数値、文字列、時刻などを表記する際に必要です。また、デバック時などに値を入力する際もリテラルと同様の表記を使用する必要があります。
リテラルは次の形式で表記します。

[データ型]#[データ]

4.5.1 数値リテラル

数値リテラルは 2 進、8 進、10 進、16 進の数値とブール値です。

型	例
整数リテラル	-12、0、+986、123_456
実数リテラル	-12.0、0.0、0.4560、3.141_592_6
指数付実数リテラル	-1.34E-12、-1.34e-12、1.0E+6
2 進リテラル	INT#2#1111_1111
8 進リテラル	INT#8#377
16 進リテラル	INT#16#FF、SINT#16#59ac
ブール	FALSE、0、TRUE、1
インスタンス変数	VAR_INST

数値を読みやすくするために、区切りとして「 _ 」を挿入できます。
例：16#FFFFFFFF → 16#FFFF_FFFF

データ型のキーワードは省略して使用できます。
例：INT#16#FF → 16#FF
　　BOOL#FALSE → FALSE

4.5.2 文字列リテラル

文字列リテラルは 2 つの単引用符「'」または 2 重引用符「"」で囲まれた 0 またはそれ以上の連続する文字列です。

型	例
STRING 文字列	'', ' ', ' Hello'
WSTRING 文字列	"", " ", "こんにちは"

4.5.3 持続時間リテラル

持続時間リテラルは時間、分、秒、ミリ秒とその組み合わせで表される時間です。

型	例
短い接頭語	T#25d3h5s20ms t#2h_15m
長い接頭語	TIME#25d3h5s20ms time#2h_15m

最後の単位のみ小数点が使用できます。
例：T#4d1.5h → 4 日 1.5 時間
　　time#13.52s → 13.52 秒

数値はオーバーフローできます。
例：t#65m59s → t#1d5m59s
　　TIME#28h12m → TIME#1d4h12m

4.5.4 日付時刻リテラル

日付時刻リテラル年、月、日、時、分、秒、ミリ秒で表される日付と時刻です。

型	例
日付	DATE#1996-06-24、date#2000-08-10、D#1970-01-01、d#2030-12-31
時刻	TIME_OF_DAY#15:36:55、time_of_day#16:21:03.24、TOD#00:00:00.00、 tod#12:00:00
日付時刻	DATE_AND_TIME#1996-06-24-15:36:55、date_and_time#2000-08-10-16:21:03.24、 DT#1970-01-01-00:00:00.00、dt#2030-12-31-12:00:00

4.6 Direct I/O

Direct I/O は入出力カードやメモリの特定アドレス（絶対番地）に変数を割り当てる方法です。

種　類	記　号	宣言例
入力メモリ	%I	xInput1 : BOOL AT %IX0.0; bInput2 : BYTE AT %IB1; wInput3 : WORD AT %IW2;
出力メモリ	%Q	xOutput1 : BOOL AT %QX0.0; bOutput2 : BYTE AT %QB1; wOutput3 : WORD AT %QW2;
マーカメモリ	%M	xFlag1 : BOOL AT %MX0.0; bFlag2 : BYTE AT %MB1; wFlag3 : WORD AT %MW2;

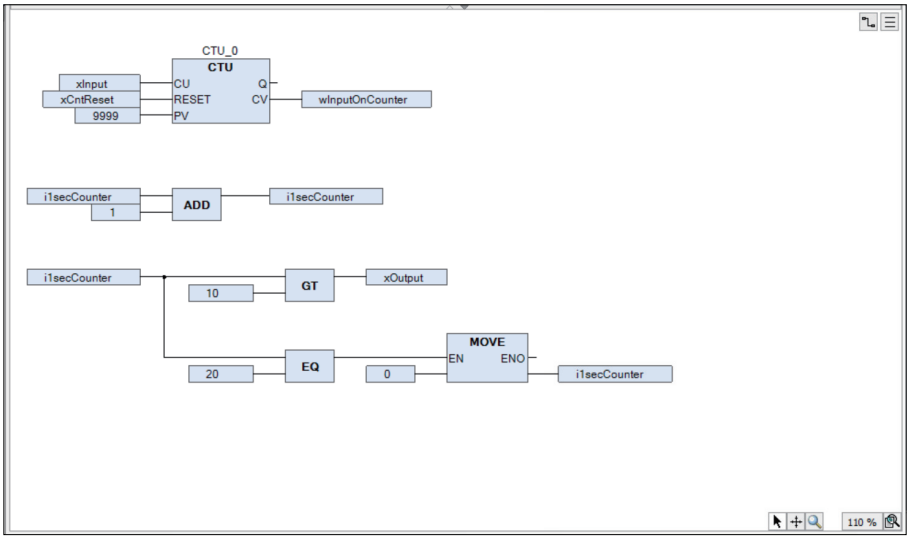
本コントローラでは特別な場合を除き、この割り当て方法（Direct I/O）を使用する必要はありません。

5. プログラミング言語

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

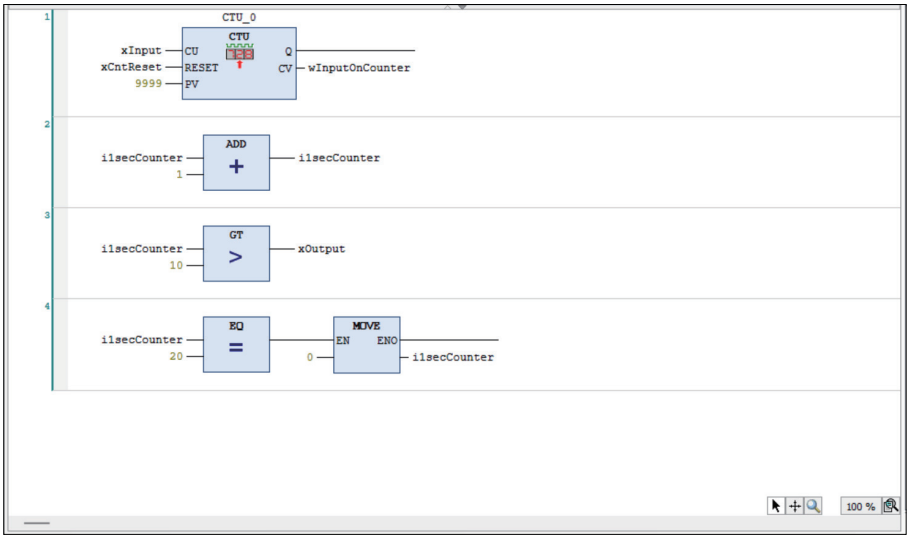
5.1 CFC（Continuous Function Chart）

CFC（Continuous Function Chart、コンティニアスファンクションチャート）は、FBD（Function Block Diagram）を元に拡張された、ファンクションブロックを任意の位置に自由に配置し、線で接続して処理を構築するグラフィック言語（IEC61131-3 非標準）です。



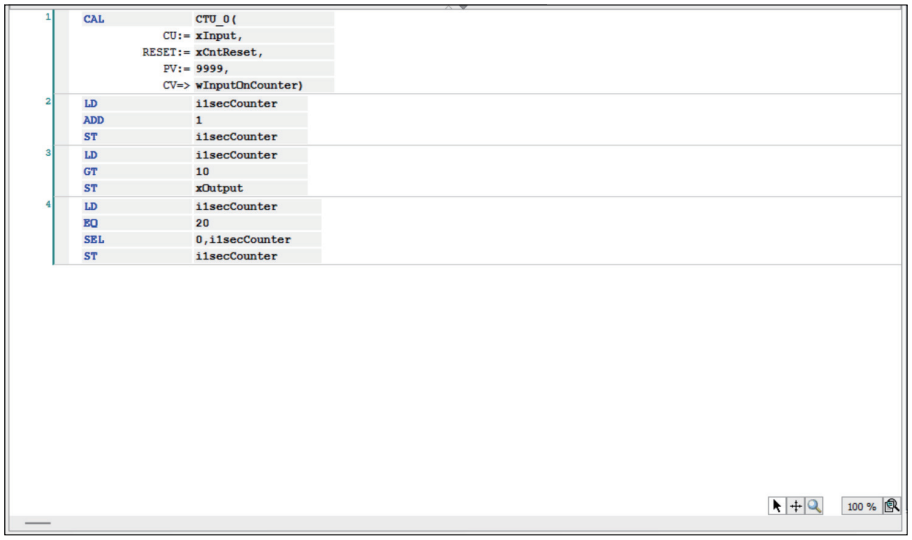
5.2 FBD（Function Block Diagram）

FBD（Function Block Diagram、ファンクションブロックダイアグラム）はファンクションブロックを線で接続して処理を構築するグラフィック言語です。

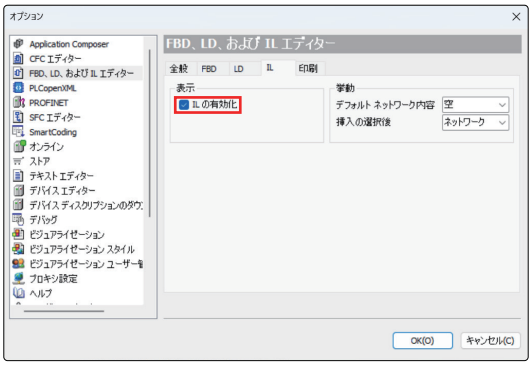


5.3 IL（Instruction List）

IL（Instruction List、命令リスト）はアセンブリ言語に類似した、逐次実行命令形式のテキスト言語です。

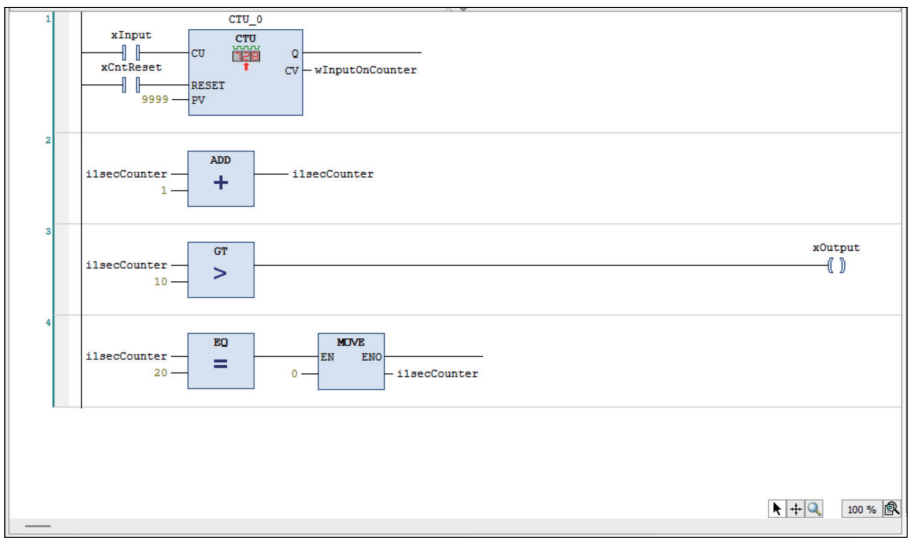


IL を使用する場合は、オプションの「FBD、LD、および IL エディター」タブで IL を有効に設定して下さい。



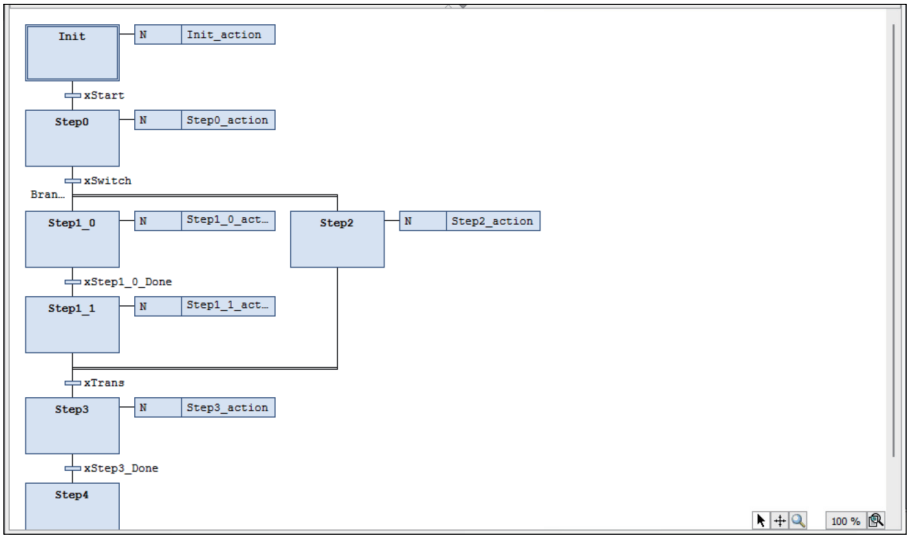
5.4 LD（Ladder Logic Diagram）

LD（Ladder Logic Diagram、ラダーロジックダイアグラム）はリレー回路記号をベースにしたグラフィック言語です。



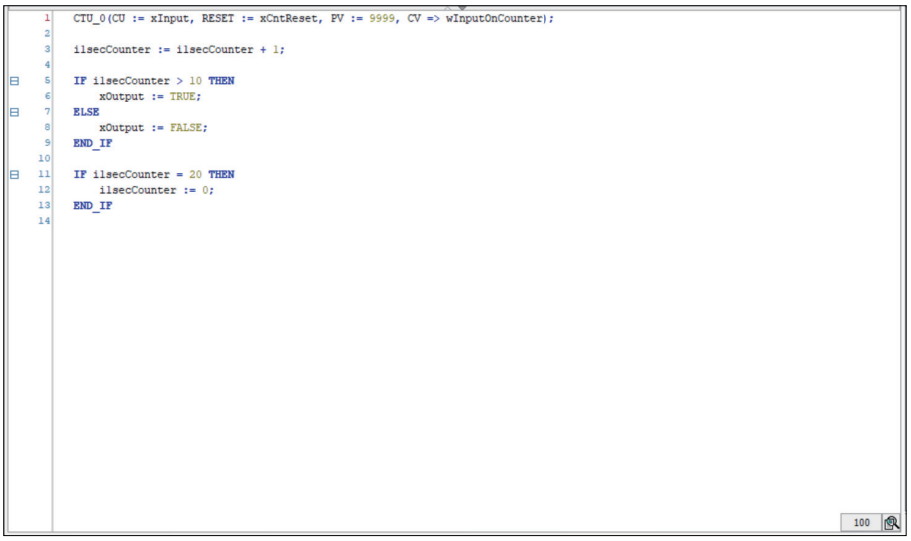
5.5 SFC（Sequential Function Chart）

SFC（Sequential Function Chart、シーケンシャルファンクションチャート）は処理の流れをステップと遷移で表現するグラフィック言語です。



5.6 ST（Structured Text）

ST（Structured Text、構造化テキスト）は構造化プログラミングをサポートするテキスト言語です。



6. プロジェクトの作成と実行

ここでは、サンプルプロジェクトの作成を通して、プログラミングからデバッグまでの手順を説明します。

6.1 プロジェクトの作成

6.1.1 パッケージのインストール

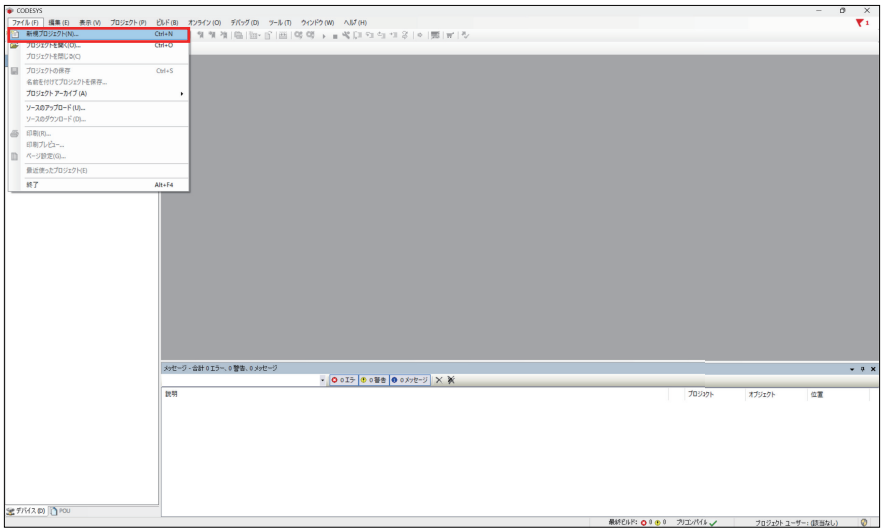
「2.2.2 パッケージ」を参照して、以下のパッケージをインストールします。

項　目	名　称
ESSENTIAL パッケージ	MG_ESSENTIAL_x_x_x_x_Rx.package
BA1C－PAC パッケージ	MG_BA1C－PAC_x_x_x_x_Rx.package

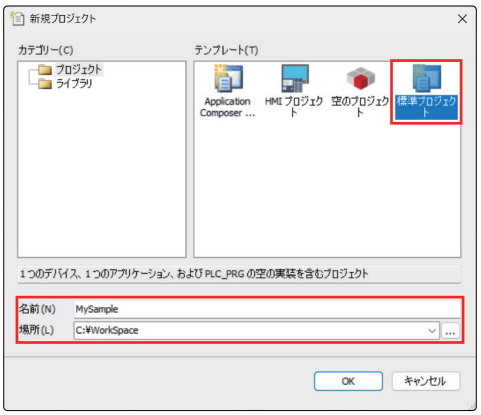
x はバージョン番号です。対応しているバージョンは「【対応バージョン】」を参照して下さい。

6.1.2 新規プロジェクトの作成

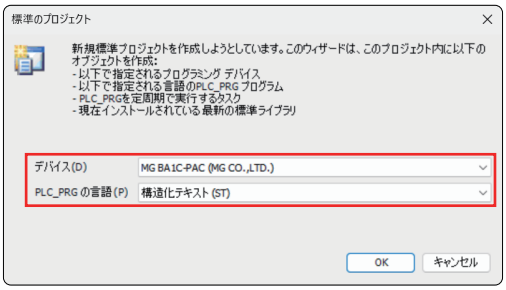
CODESYS IDE のメニューから「ファイル」→「新規プロジェクト…」をクリックします。



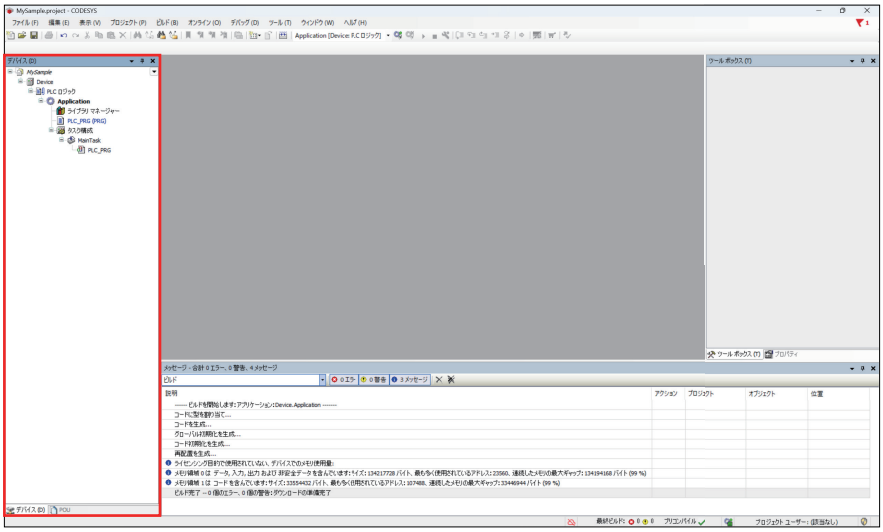
新規プロジェクト画面でカテゴリー「プロジェクト」、テンプレート「標準プロジェクト」を選択し、プロジェクトの名前と場所を入力します。



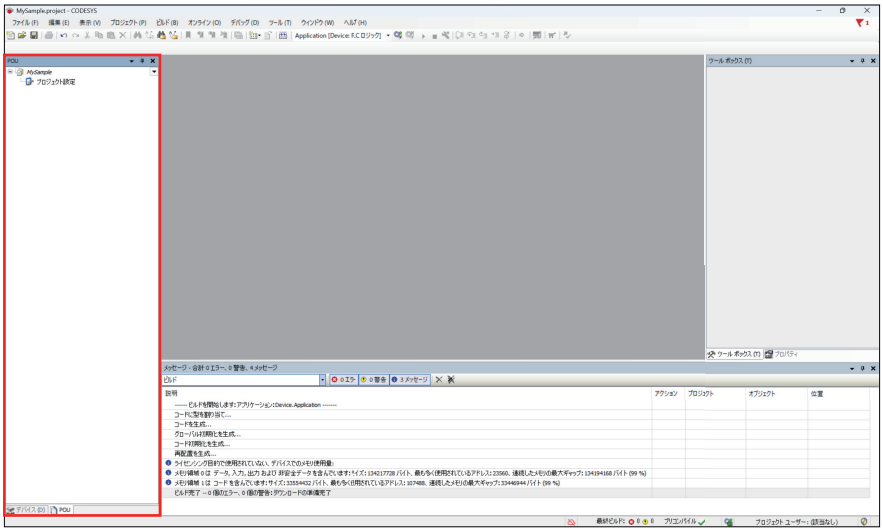
テンプレートで「標準のプロジェクト」を選択した場合は、作成したプロジェクトを実行するターゲット（MG BA1C－PAC）とプロジェクトに含まれる POU の IEC プログラミング言語を選択します。



プロジェクトが作成され、エディタウィンドウが表示されます。
「デバイスツリー」ではターゲット Device（MG BA1C－PAC）に属するコンポーネントをツリー構造で管理します。このツリーには、プログラム「PLC_PRG」とそれを呼び出すタスク「MainTask」、プロジェクトで使用するライブラリを管理する「ライブラリマネージャー」が含まれています。
標準プロジェクトの初期状態では、I/O 設定に必要な「IoStandard」ライブラリや IEC61131-3 に準拠したすべてのファンクションとファンクションブロックを提供する「Standard」ライブラリなどが読み込まれています。



「POU ツリー」は POU 管理やプロジェクト設定に使用します。



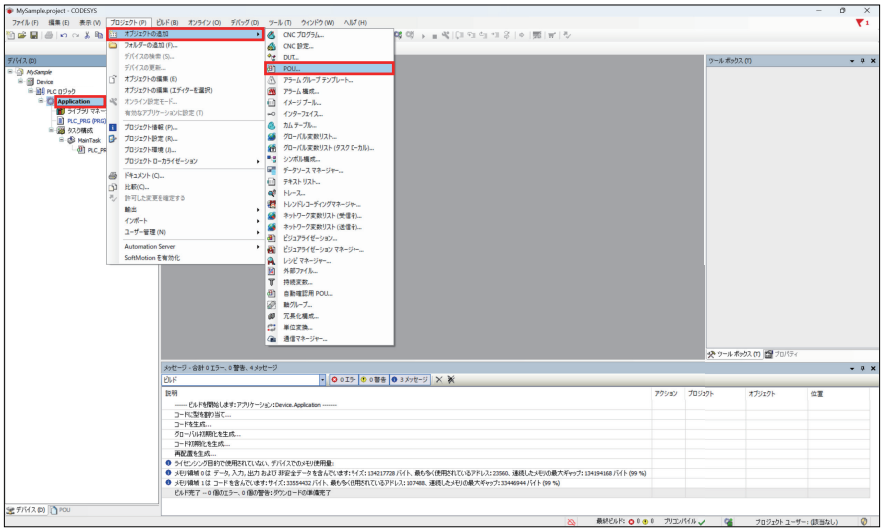
6.1.3 プロジェクト情報の設定

メニューから「プロジェクト」→「プロジェクト情報…」をクリックします。
プロジェクト情報画面では、プロジェクトの作成者やバージョンなどの情報を設定します。

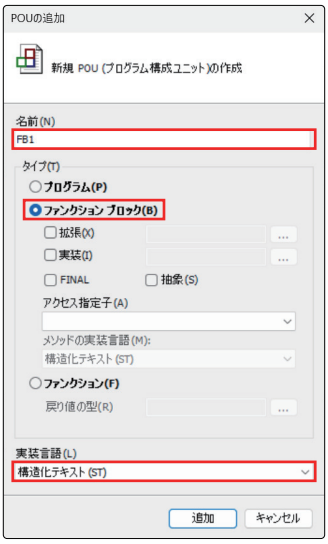
6.2 プログラミング

6.2.1 POU の作成

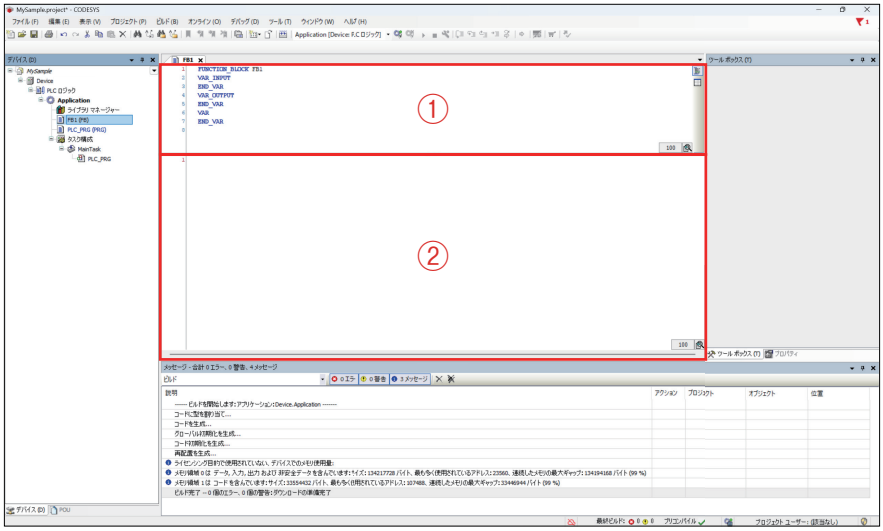
デバイスツリーの Application を選択した状態で、メニューの「プロジェクト」→「オブジェクトの追加」→「POU…」をクリックします。



名前を「FB1」、タイプを「ファンクションブロック」、実装言語を「構造化テキスト（ST）」に設定して、追加をクリックします。

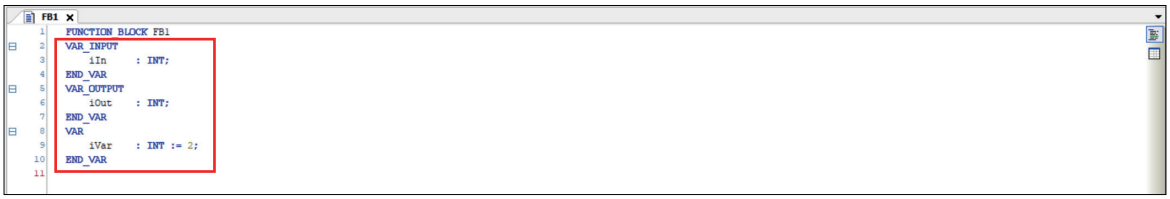


デバイスツリーの Application にファンクションブロック「FB1」が追加され、FB1 のプログラミング用エディタが開きます。このエディタは、変数宣言部とプログラム本体部の 2 画面に分割されています。



項　目	内　容
①変数宣言部	各種変数の定義や初期値を記述する領域です。 POU のタイプと名前（例:「FUNCTION_BLOCK FB1」）や、変数を宣言するためのキーワード「VAR_INPUT」「END_VAR」などを記述します。
②プログラム本体部	IEC プログラミング言語によるプログラムコードを記述する領域です。 初期状態では空で、行番号として「1」のみが表示されています。

エディタの変数宣言部で「VAR_INPUT」の後にカーソルを置き、エンターキーを押します。新しい行が追加されるので、整数型（INT 型）で「iIn」を宣言します。同じ方法で、「VAR_OUTPUT」には整数型（INT 型）で「iOut」を、「VAR」には整数型（INT 型）で初期値が 2 の「iVar」を宣言します。



プログラム本体部に、次のステートメントを入力します。

```
iOut := iIn + iVar;
```

これにより、入力変数「iIn」に定数「iVar＝2」を加えた結果を、出力変数「iOut」に出力するファンクションブロック「FB1」が定義されました。

6.2.2 PLC_PRG の変数宣言

デバイスツリーの「PLC_PRG」をダブルクリックして、PLC_PRG のプログラミング用エディタを開きます。
エディタの変数宣言部で、「VAR」に整数型（INT 型）の「iVar」と、事前に定義したファンクションブロック「FB1」のインスタンス「fbInst」を宣言します。

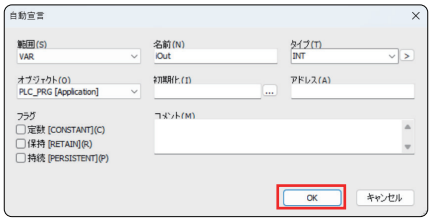
なお、変数宣言部で一つずつ宣言する代わりに、次に紹介する変数の自動宣言機能を使用することもできます。

6.2.3 変数の自動宣言機能

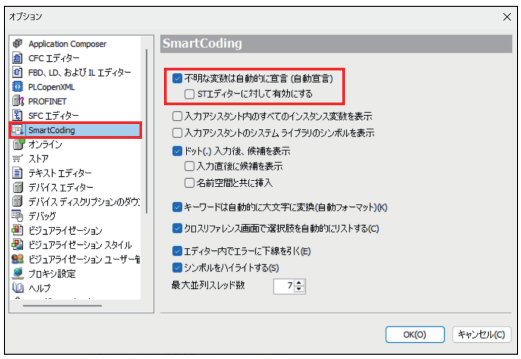
エディタのプログラム本体部に、次のステートメントを入力します。

```
iVar := iVar + 1;
fbInst(iIn := 11, iOut => iOut);
```

プログラム本体部でまだ宣言されていない変数を使用すると、変数宣言入力アシスタントが開きます。内容を確認して「OK」をクリックすると、未定義の変数「iOut」が変数宣言部に追加されます。

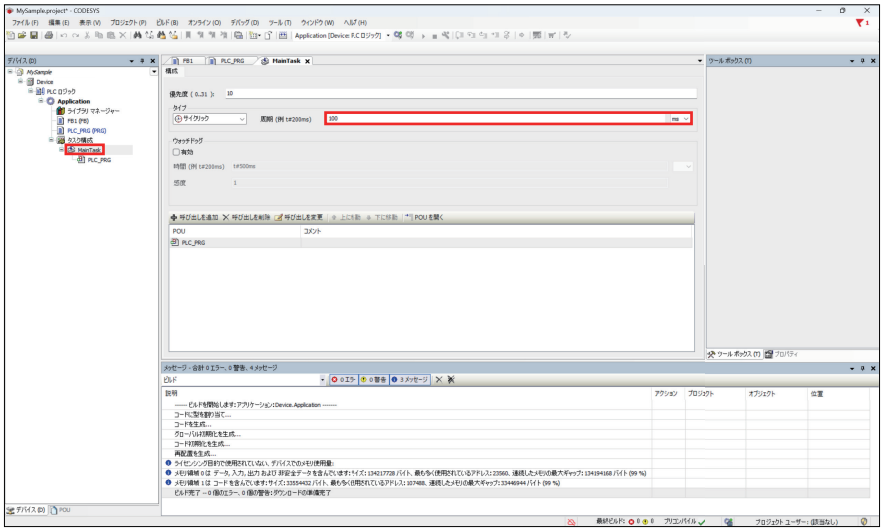


変数宣言入力アシスタント機能の設定はオプションの「SmartCording」タブで行います。



6.2.4 タスク設定

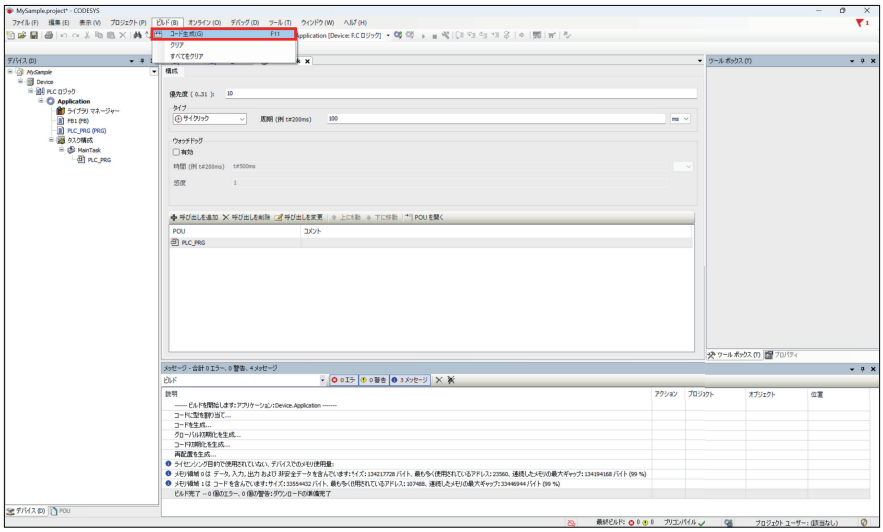
デバイスツリーの「MainTask」をダブルクリックして、タスク設定画面を開きます。
この画面では、タスク「MainTask」に登録されている POU「PLC_PRG」の実行周期を設定できます。本コントローラの初期設定では 100 ms 周期で呼び出されます。



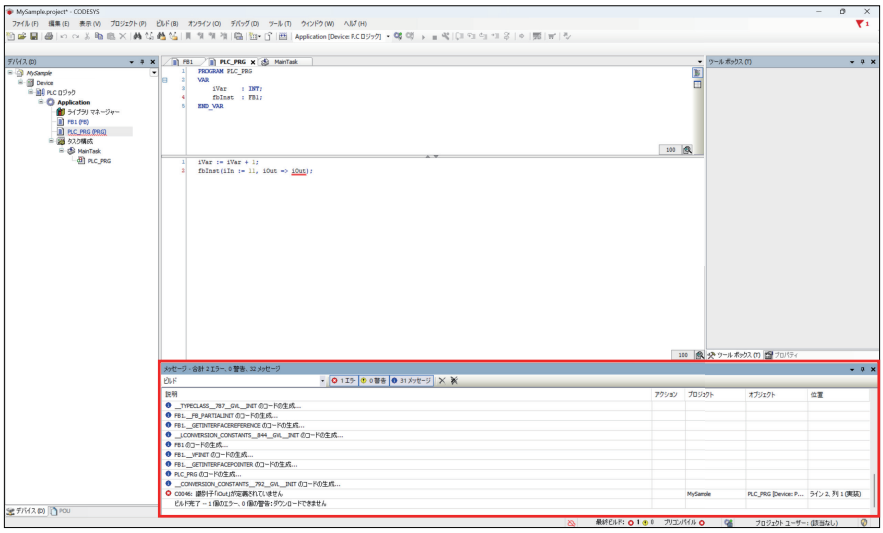
これでプログラムの作成は完了しました。次に、プログラムのエラーチェックを行い、ターゲットに転送するためのイメージを作成します。

6.2.5 コード生成

メニューから「ビルド」→「コード生成」をクリックします。



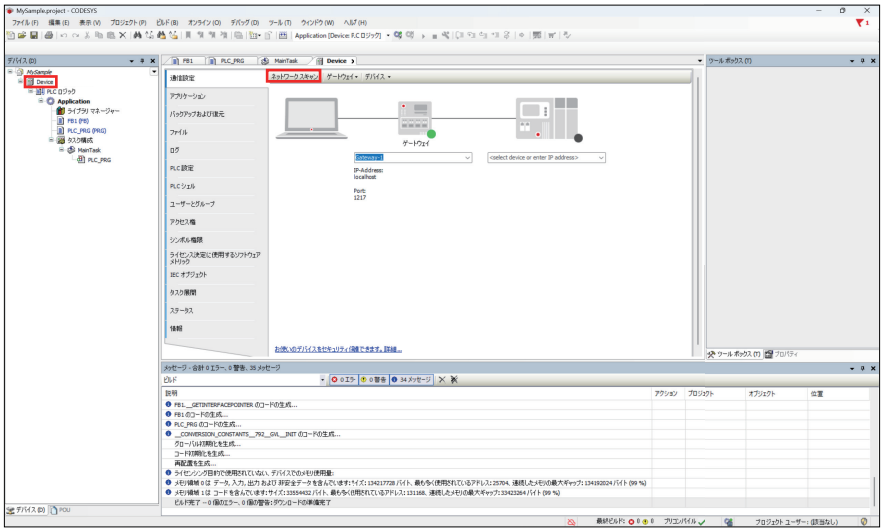
正常にコード生成が完了すると、メッセージウィンドウに「0 個のエラー、0 個の警告」と表示されます。
エラーや警告が検出された場合は、出現箇所がメッセージウィンドウに表示されます。エラーと警告が解消されるまで、問題個所の修正とコード生成作業を繰り返して下さい。



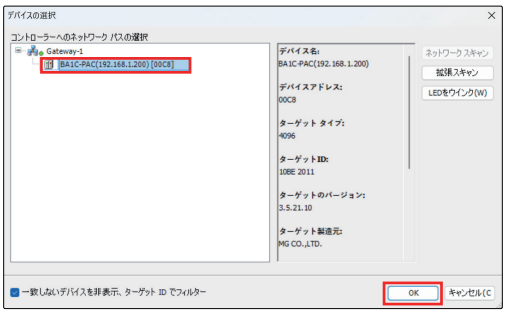
6.3 ターゲットへの接続とアプリケーションのダウンロード

6.3.1 ターゲットへの接続（ログイン）

デバイスツリーの「Device（MG BA1C－PAC）」をダブルクリックして、デバイスの通信設定画面を開きます。
この画面では、接続に使用するゲートウェイとターゲット（コントローラ）を設定します。

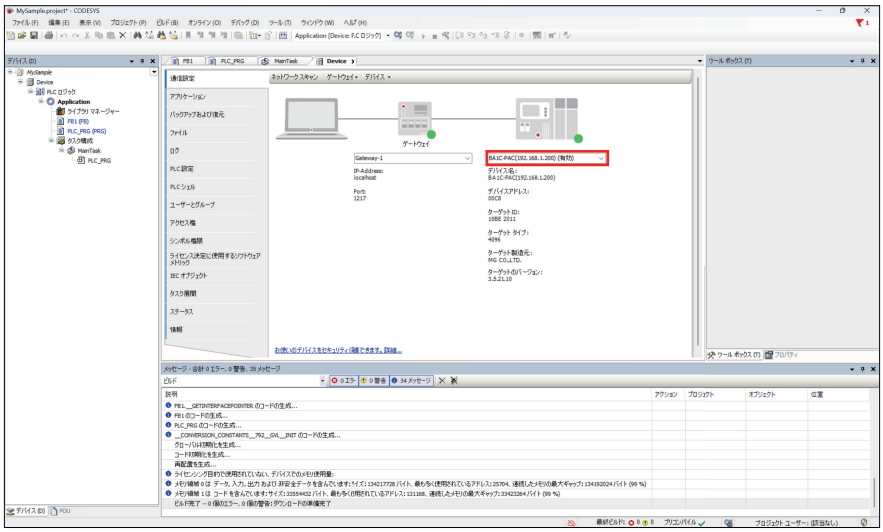


「ネットワークスキャン」をクリックすると、ネットワーク上の接続可能なターゲットが検出されます。検出されたターゲットの中から､接続するターゲット「BA1C－PAC（コントローラの IP アドレス）＊1」（例：BA1C－PAC（192.168.1.200））を選択し、「OK」をクリックします。
ターゲットが 1 台も検出されない場合は、接続対象のターゲットと PC が同一ネットワーク内にあること、および IP アドレスなどのネットワークパラメータが適切に設定されていることを確認して下さい。



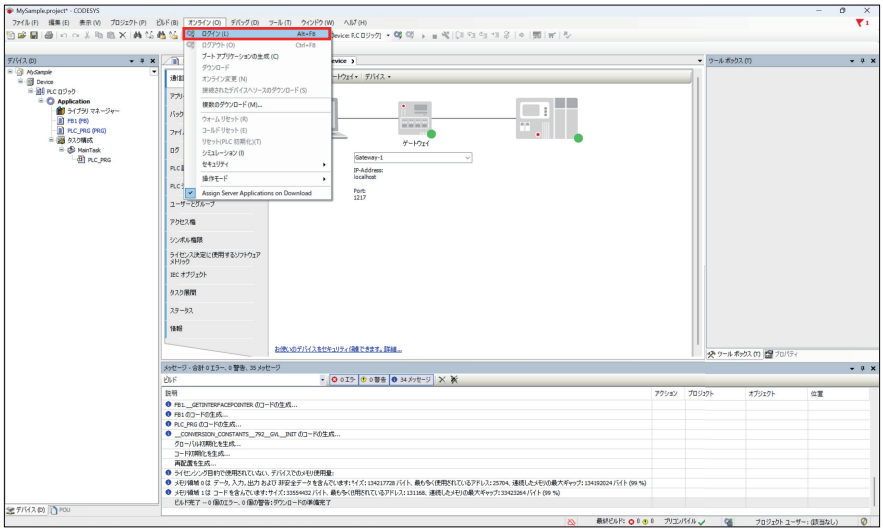
＊1、コントローラの IP アドレスは eth1（LAN2）、eth0（LAN1）、br0（LAN1＋LAN2）の順に検出されます。
CODESYS IDE と接続している IP アドレスとは異なるアドレスが表示される場合があります。

接続が完了すると、検出されたターゲットは「BA1C－PAC（コントローラの IP アドレス）」（例：BA1C－PAC（192.168.1.200））と表示されます。

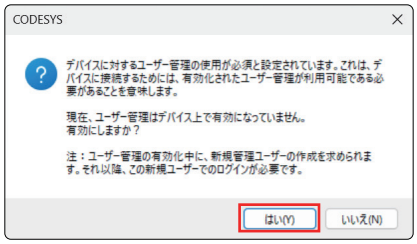


6.3.2 アプリケーションのダウンロード（転送）

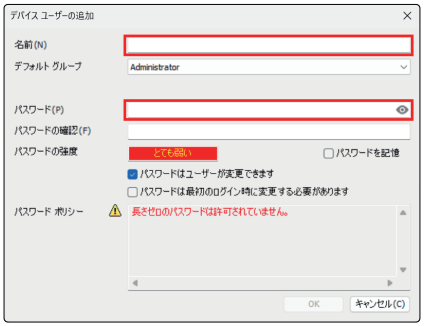
メニューから「オンライン」→「ログイン」をクリックします。



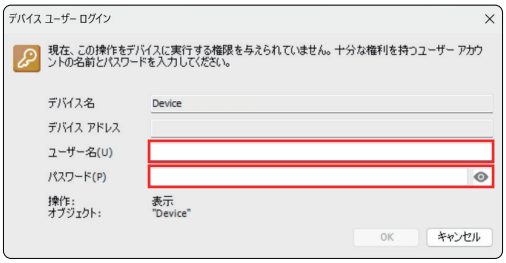
初回接続時は、以下のダイアログが表示されます。「はい」をクリックして、ユーザ管理設定を行います。



ユーザ名とパスワードを登録します。

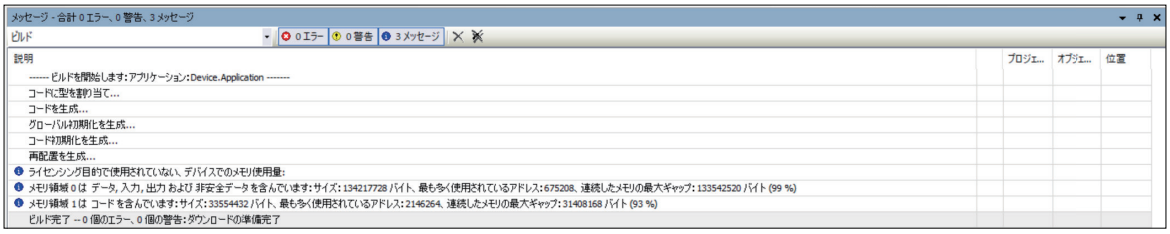


登録したユーザ名とパスワードを入力して、ターゲットに接続します。



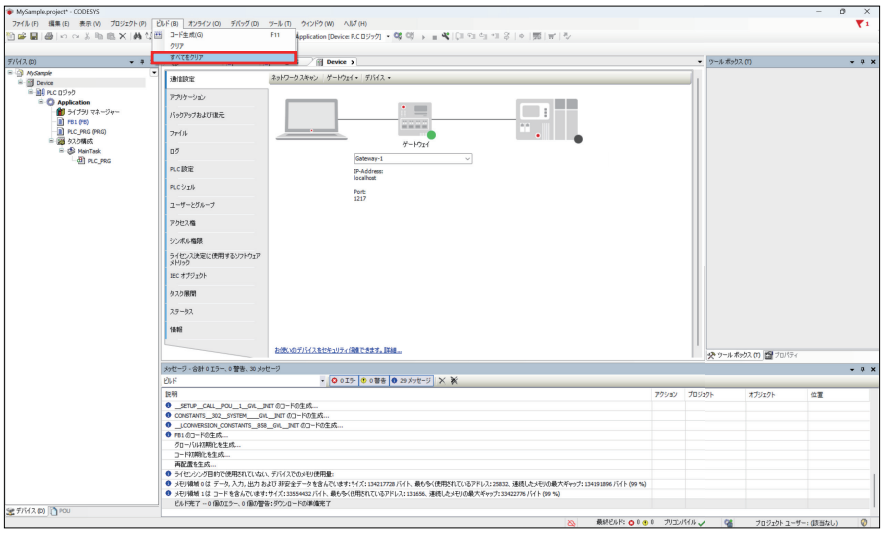
接続先のターゲットにアプリケーションが存在しない場合、以下のダイアログが表示されます。「はい」をクリックして、PC からターゲットへプログラムをダウンロード（転送）します。ターゲットにログインすると、CODESYS IDE は現在の状態や現在値をリアルタイムで表示するオンラインモードになります。

ダウンロードされたアプリケーションが必要とするメモリ容量や、ターゲット側の空きメモリ状況はコード生成時に表示されるメッセージから確認できます。

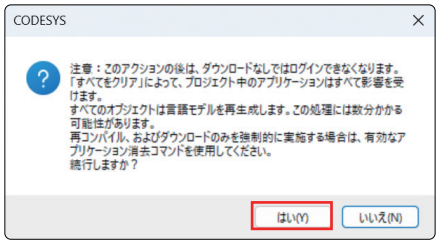


アプリケーションのダウンロードには、分断された空き領域の合計ではなく、一定サイズの連続した空きメモリ領域が必要です。繰り返しダウンロードを行うとメモリが断片化し、必要なサイズの連続領域が確保できず、ダウンロードが失敗する場合があります。このような場合には、アプリケーションを再生成して完全ダウンロードを行うことで、連続した空きメモリ領域の確保が可能となります。

ターゲットにログインしている場合は一度ログアウトして、メニューから「ビルド」→「すべてをクリア」をクリックします。



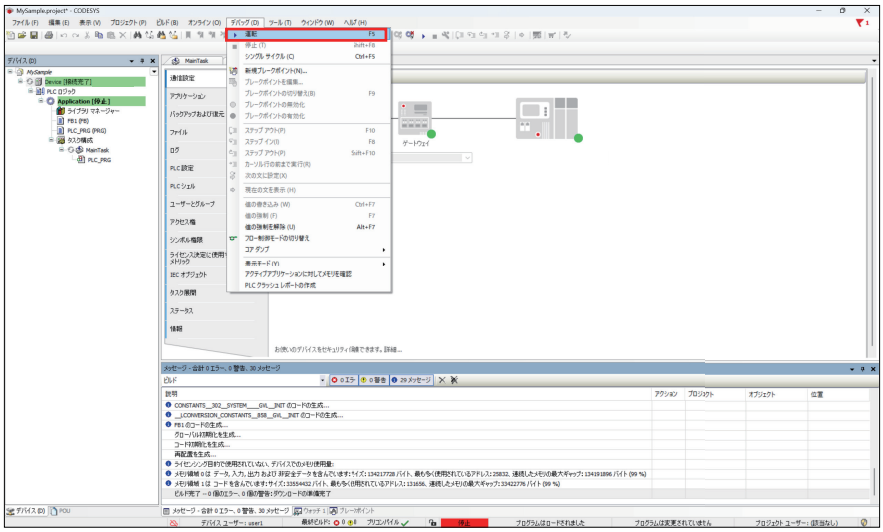
確認ダイアログが表示されたら「はい」をクリックします。



メニューから「ビルド」→「コード生成」をクリックしてアプリケーションを再生成します。その後、ターゲットに再びログインして、アプリケーションを再ダウンロードします。

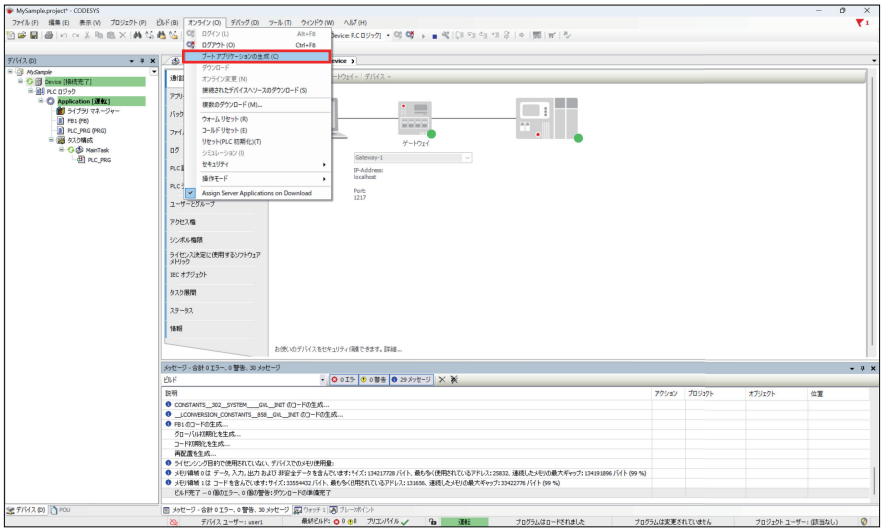
6.3.3 ターゲット内のアプリケーションを実行する

アプリケーションを新規にダウンロードした直後は「停止」状態になります。メニューから「デバッグ」→「運転」をクリックして、アプリケーションを「運転」状態にして下さい。



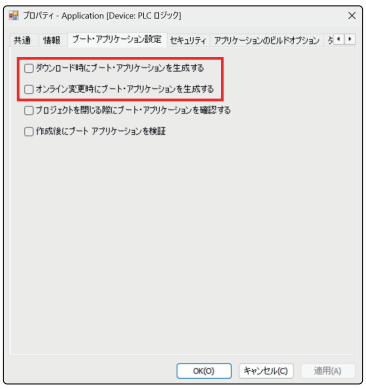
6.3.4 ターゲットにブートアプリケーションを生成する

ブートアプリケーションは、コントローラに電源が投入されると自動的に起動されるアプリケーションのことです。
オンライン（ログイン）状態で、メニューから「オンライン」→「ブートアプリケーションの生成」をクリックすると、現在ターゲットに転送されているアクティブなアプリケーションがブートアプリケーションとして登録されます。



ログインなどでダウンロード（転送）されただけのアプリケーションはブートアプリケーションではありません。そのため、次回ターゲットの電源が再投入された際には、以前に作成したブートアプリケーションが起動します。
ブートアプリケーションはファイルシステム（フラッシュメモリ）に格納されます。この書き込みには数秒かかるため、その間はファイルの破損を防ぐために「リセット」や「電源 OFF」を行わないで下さい。

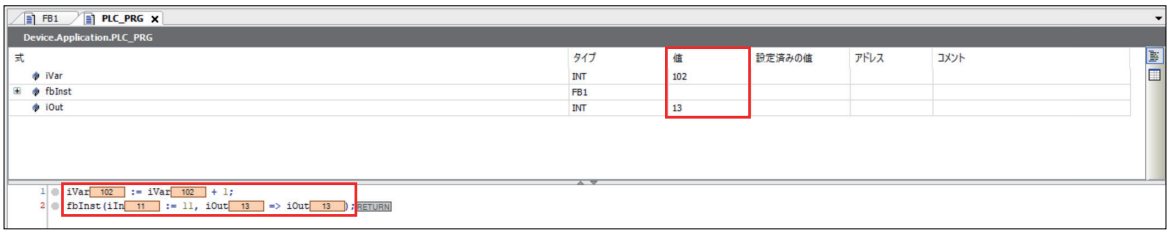
ブートアプリケーションの設定はデバイスツリーの「Application」のプロパティの「ブートアプリケーション設定」タブで行います。



6.4 デバッグ

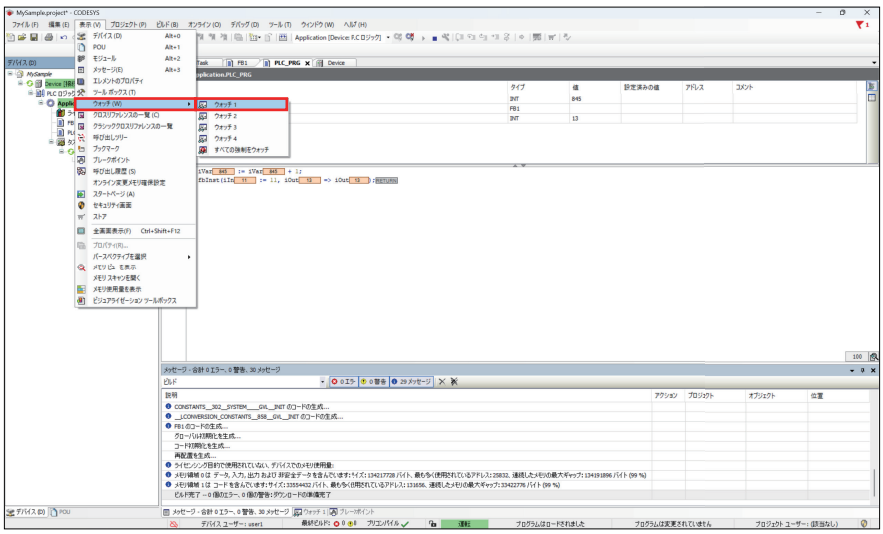
6.4.1 オンラインモードで変数の現在値をモニタ

オンラインモードでは、変数の現在値やファンクション、ファンクションブロックの入出力パラメータの現在値をリアルタイムに表示します。
プログラム本体部では、各変数にオレンジ色の四角が表示され、現在の値がリアルタイムで更新されます。変数宣言部では、プログラム内の変数一覧を確認できます。また、特定の変数をまとめて監視できるウォッチリスト機能もあります。

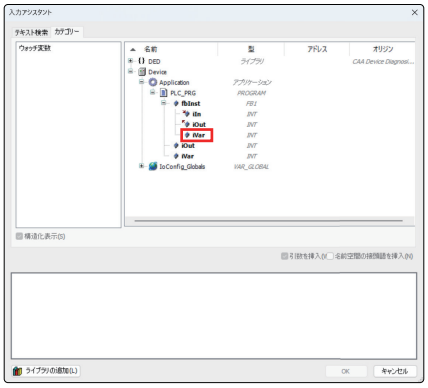


6.4.2 ウォッチウィンドウ

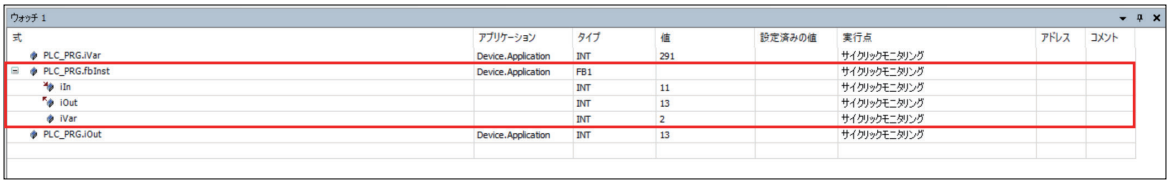
メニューから「表示」→「ウォッチ」→「ウォッチ 1」をクリックします。



開いたウォッチウィンドウに、監視する変数式（例：Device.Application.PLC_PRG.iVar）を入力します。変数式の入力欄の「…」をクリックするか、F2 キーを押すと、入力アシスタント機能を利用できます。



「FB1」のようなファンクションブロックでは、メンバ変数がツリー構造で表示されます。



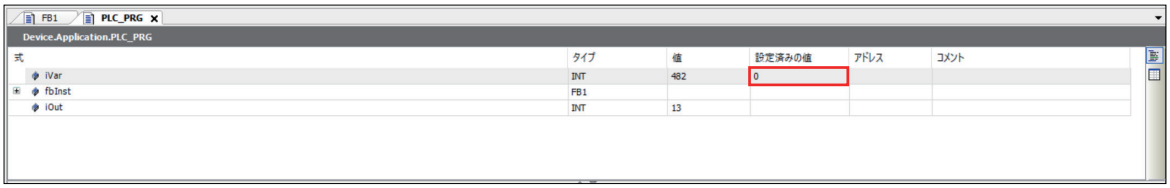
6.4.3 オンラインモードで変数の現在値を変更

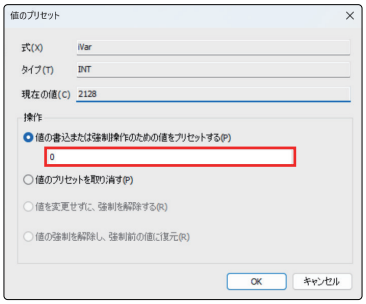
オンラインモードでは、必要に応じて変数の現在値を変更できます。
現在値の変更には次の方法があります。

種　類	手　順	内　容
値の書き込み	メニューから「デバッグ」→「値の書き込み」をクリックする、または Ctrl ＋ F7 キーを押す	次の実行周期の最初に、指定した値を 1 度だけ書き込みます。
値の強制	設定: メニューから「デバッグ」→「値の強制」をクリックする、または F7 キーを押す 解除: メニューから「デバッグ」→「値の強制を解除」をクリックする、または Alt ＋ F7 キーを押す	実行周期の最初と最後に、指定した値を毎周期書き込みます。 実行周期内の処理順序は以下のとおりです。 1. 強制値の書き込み 2. プログラムコードの実行 3. 強制値の書き込み

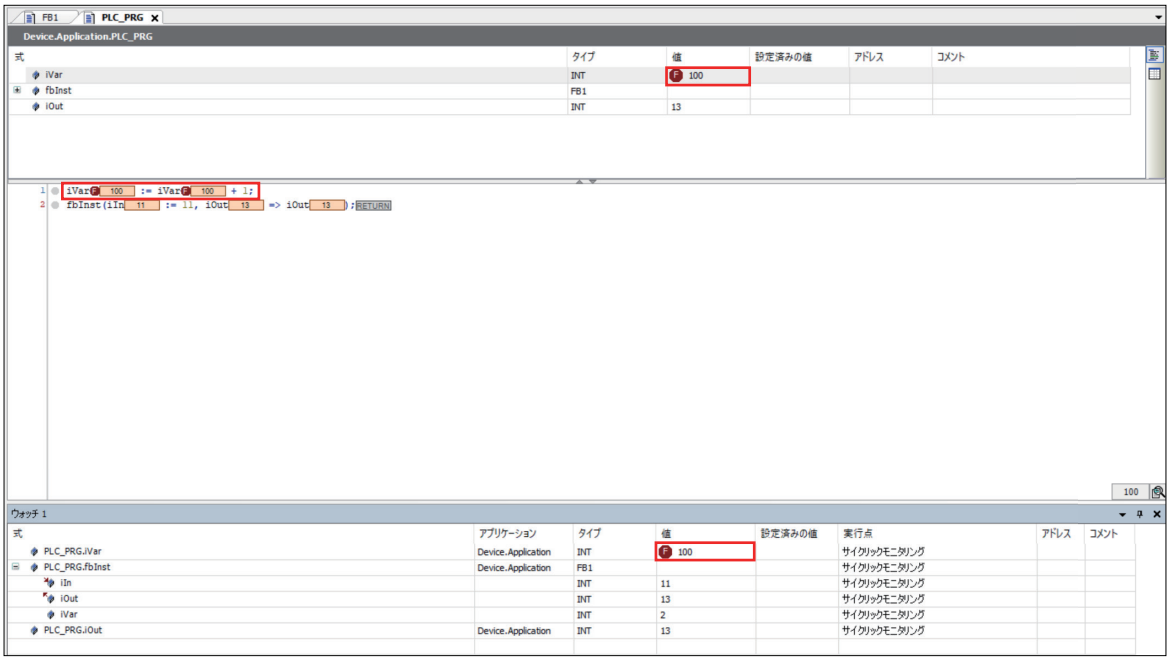
「値の書き込み」や「値の強制」は、実行中のプログラムの動作に予想外の影響を与える可能性があります。使用する際には、影響範囲を十分に考慮し、安全性に配慮して下さい。

「値の書き込み」や「値の強制」を行うには、「設定済みの値」に書き込む値を事前に入力する必要があります。





「設定済みの値」を入力したら、「値の書き込み」は Ctrl ＋ F7 キー、「値の強制」は F7 キーを押して実行します。
「値の強制」を行った場合、「値」欄に現在強制中であることを示す（F）が現在値の左側に表示されます。



6.4.4 プログラムコードを任意の位置で停止

オンラインモードでは、必要に応じてブレークポイントを設定できます。
プログラムコードの実行がブレークポイントに達すると、プログラムの実行が一時的に停止します。

停止状態からプログラムを再開するには次の方法があります。

種　類	手　順	内　容
運転	メニューから「デバッグ」→「運転」をクリックする	アプリケーションを運転状態にします。
ステップオーバー	メニューから「デバッグ」→「ステップオーバー」をクリックする、または F10 キーを押す	現在の命令の次の命令に進みます。
ステップイン	メニューから「デバッグ」→「ステップイン」をクリックする、または F8 キーを押す	命令のプログラムコードが表示可能な場合、現在の命令の内部に移動します。
ステップアウト	メニューから「デバッグ」→「ステップアウト」をクリックする、または Shift ＋ F10 キーを押す	現在のファンクションまたはファンクションブロックから抜けて、呼び出し元に戻ります。
カーソル行の前まで実行	メニューから「デバッグ」→「カーソル行の前まで実行」をクリックする	現在の停止位置からカーソル行の前の命令まで実行します。

ブレークポイントが設定された行の先頭には赤い丸が表示されます。

実行がブレークポイントに到達すると、プログラムが一時停止し、該当行が黄色く反転表示されます。

メニューから「表示」→「ブレークポイント」をクリックして開かれるブレークポイントウィンドウでは、プログラム中のブレークポイント一覧を確認できます。

6.5 ターゲットとの接続を終了

6.5.1 ターゲットとの接続を終了（ログアウト）

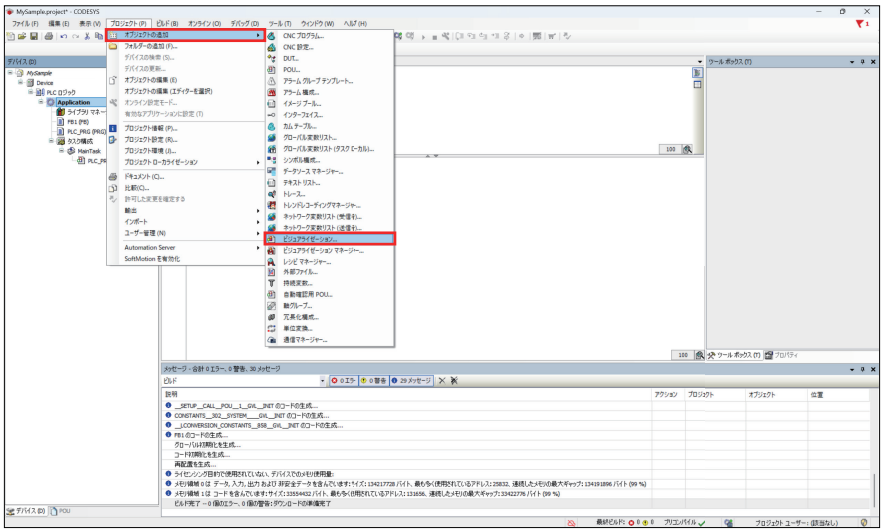
メニューから「オンライン」→「ログアウト」をクリックして、ターゲットとの接続を終了（ログアウト）します。デバッグを終了し、プログラムの修正や追加作業を行うには、ターゲットとの接続を一度終了する必要があります。

6.6 HMI

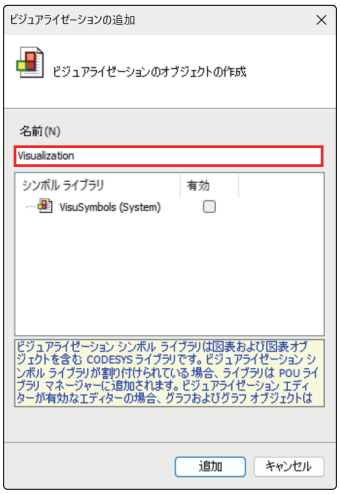
CODESYS IDE には、ユーザインターフェースの「ビジュアライゼーション」を作成する機能があります。このビジュアライゼーション機能では、アプリケーションで使用しているすべての内部変数にアクセスでき、ビジュアライゼーション画面から変数値の表示や書き換えが可能です。

6.6.1 HMI コンポーネントの追加

デバイスツリーの Application を選択した状態で、メニューの「プロジェクト」→「オブジェクトの追加」→「ビジュアライゼーション…」をクリックします。



名前を「Visualization」に設定して、追加をクリックします。



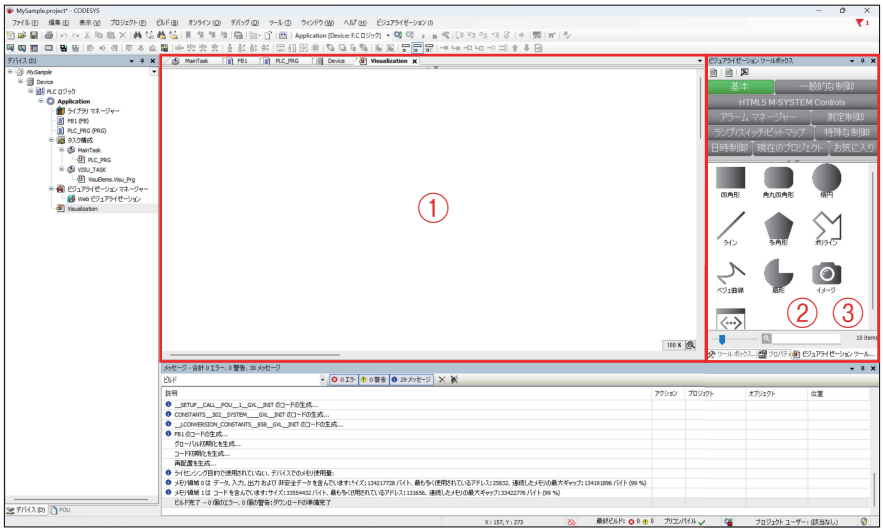
デバイスツリーの Application にビジュアライゼーション「Visualization」が追加され、ビジュアライゼーションのエディタ画面が開きます。

6.6.2 ビジュアライゼーションの種類

CODESYS が提供するビジュアライゼーションの種類を以下に示します。

種　類	内　容
CODESYS HMI	HMI は CODESYS IDE のオンラインモニタ機能を使用して PC 上で表示されます。
CODESYS ターゲットビジュアライゼーション	ディスプレイを搭載したターゲットの場合、HMI はターゲット自体のディスプレイに表示されます
CODESYS Web ビジュアライゼーション	HMI は HTML5 Canvas Element を利用できる Web ブラウザに表示されます。

6.6.3 ビジュアライゼーションエディタの説明



項　目	内　容
①画面エディタ	実際の表示イメージに、あらかじめ用意されている画面コンポーネント（部品）を配置する場所です。画面の作成は、ツールボックスから画面コンポーネントをドラッグして配置します。
②ツールボックス	用意されている各種画面コンポーネントが一覧で表示されます。
③プロパティ	エディタ画面に配置された部品のプロパティが一覧で表示され、設定値の変更ができます。

6.6.4 プログラムの修正

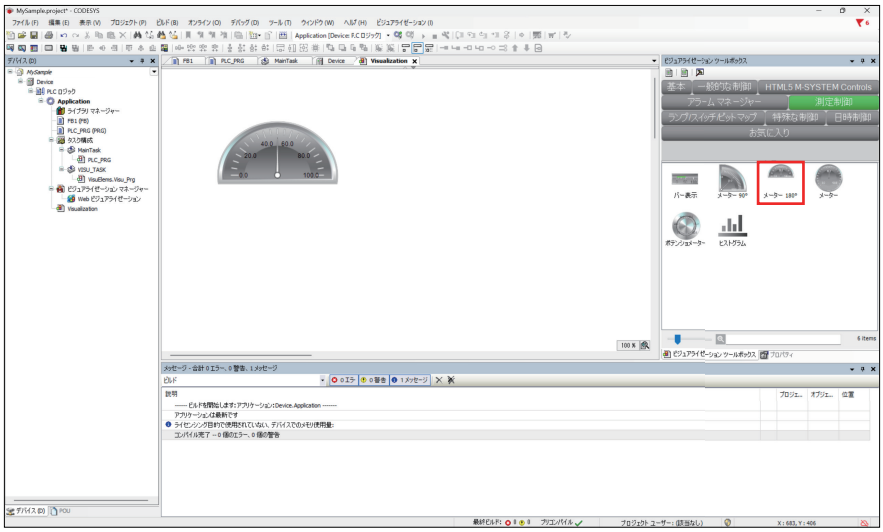
これから作成するビジュアライゼーション画面で表示を変化させるために、先で作成したプログラム「PLC_PRG」を次のように修正します。

・「VAR」にブール型（BOOL 型）の変数「bFlg」を追加
・「iVar」の値を 0 ～ 99 に制限する処理を追加
・「bFlg」を「iVar」の値が 50 以上で TRUE、それ以外で FALSE とする処理を追加

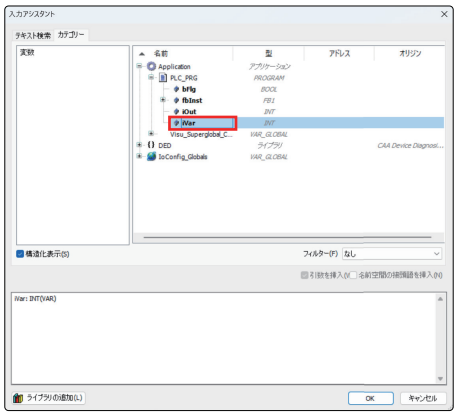
このプログラムを実行すると、「iVar」は初期値 0 から 1 ずつ加算され、99 になると 0 に戻ります。その結果、「bFlg」の値は FALSE と TRUE を繰り返す動作になります。

6.6.5 画面のデザイン

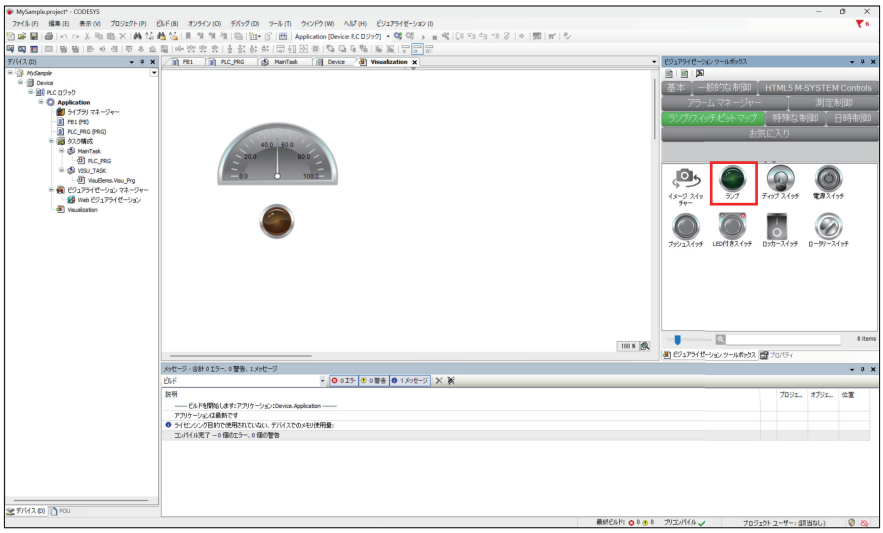
デバイスツリーの「Visualization」をダブルクリックして、Visualization のエディタ画面を開きます。
ビジュアライゼーションツールボックスから「メーター 180° 」をドラッグし、エディタ画面に配置します。



メーターと変数「iVar」を関連付けます。メーターの針が示す値はプロパティ「値」で指定できます。「値」欄の「…」をクリックするか、F2 キーを押して入力アシスタント画面を開き、「Application.PLC_PRG.iVar」を選択して「OK」をクリックします。



同様に、ツールボックスから「ランプ」ドラッグし、エディタ画面に配置します。点灯状態を指定するプロパティ「変数」に「Application.PLC_PRG.bFlg」を関連付けます。



6.6.6 実行

メニューから「オンライン」→「ログイン」をクリックして、アプリケーションを実行します。
前章で設定したブレークポイントがまだ残っている場合は、すべて解除してからアプリケーションを実行して下さい。

実行すると、CODESYS IDE 内のビジュアライゼーション画面がオンラインモードに切り替わります。
メーターの針は、関連付けられた変数「iVar」の値に従い、0 から 99 の範囲を繰り返し表示します。
また、ランプの色は関連付けられた変数「bFlg」の値に従い、FALSE（暗い黄色）と TRUE（明るい黄色）を繰り返し表示することが確認できます。

7. ライブラリ

ライブラリは、プログラムを再利用するための最も有効な方法です。作成したファンクションやファンクションブロックを他のアプリケーションで使用したり、他の開発者に提供したりすることが容易になります。
ここでは、ユーザライブラリの作成方法と既存のライブラリについて説明します。

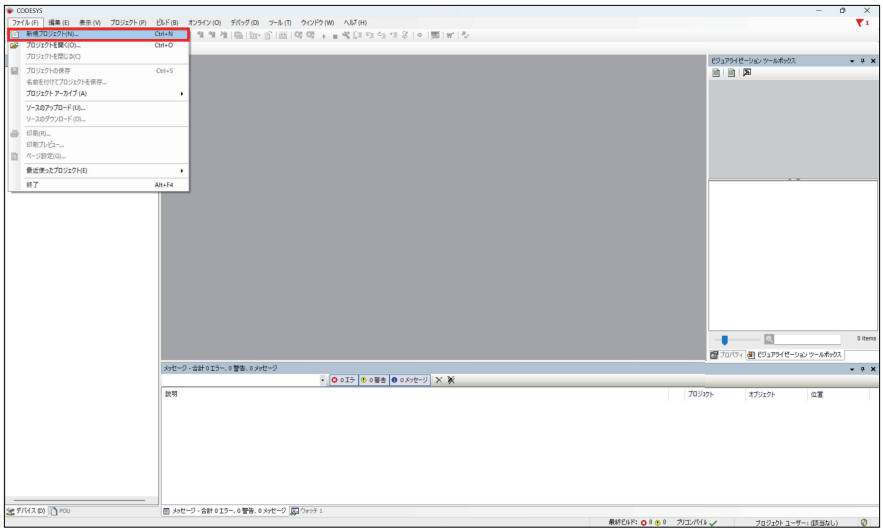
7.1 ユーザライブラリ

ライブラリの作成方法には､テンプレートを使用する方法と新規(空)から作成する方法の 2 種類があります。テンプレートには多くのモジュールが含まれているため、ライブラリ作成後は不要なモジュールを削除することをおすすめします。

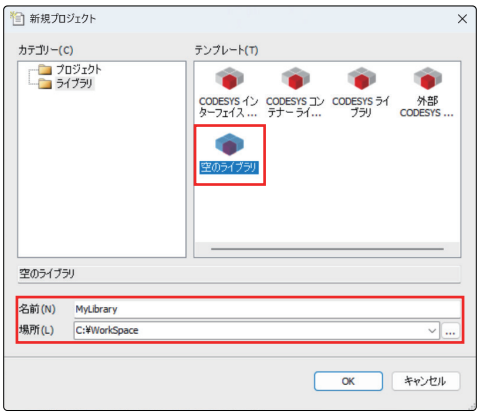
ここでは、新規（空）から作成し、最小限の要素を備えたライブラリを作成する方法を説明します。

7.1.1 ライブラリの作成

CODESYS IDE のメニューから「ファイル」→「新規プロジェクト…」をクリックします。

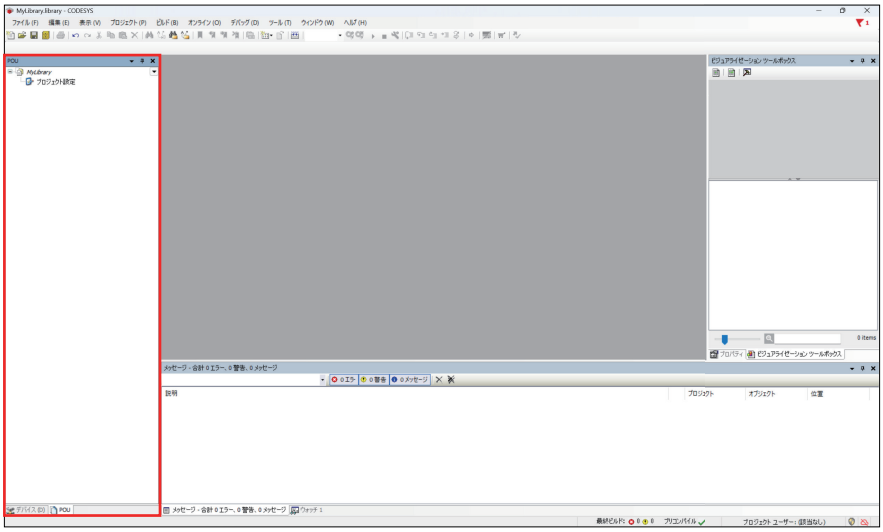


新規プロジェクト画面でカテゴリー「ライブラリ」、テンプレート「空のライブラリ」を選択し、ライブラリの名前と場所を入力します。

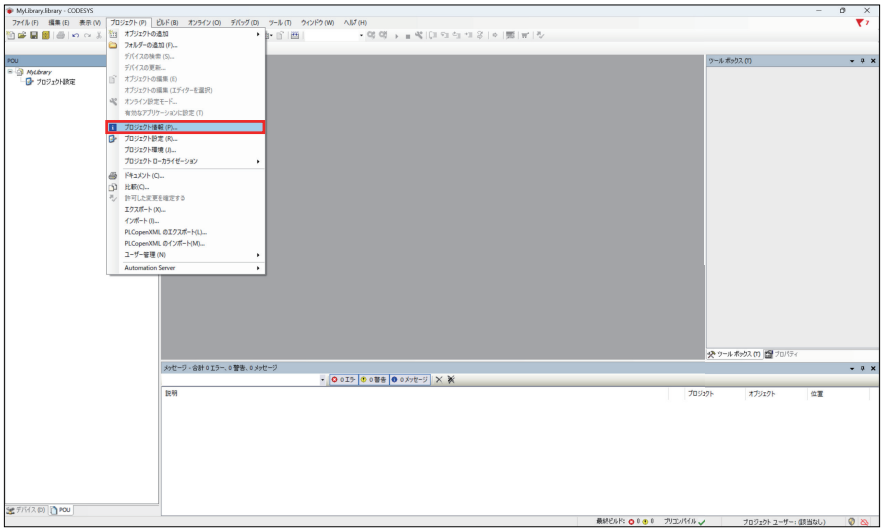


7.1.2 ライブラリのプロジェクト情報を設定

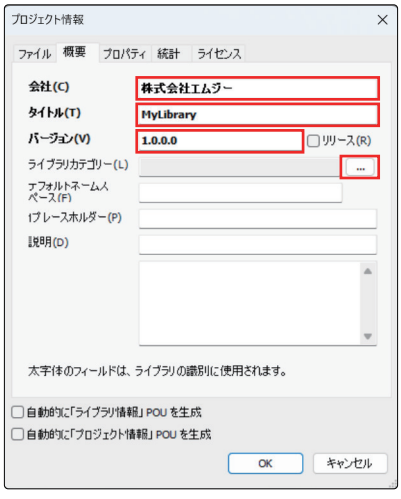
空のライブラリが作成されます。ライブラリの作成時は「POU ツリー」を表示して下さい。



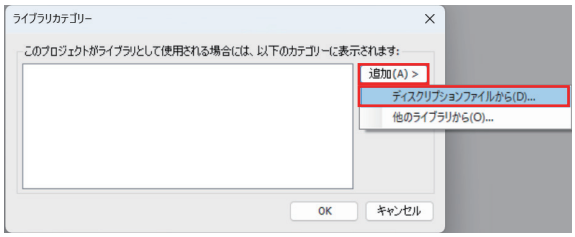
メニューから「プロジェクト」→「プロジェクト情報…」をクリックします。



プロジェクト情報画面の概要タブでは、プロジェクトの作成者やバージョンなどの情報を設定します。

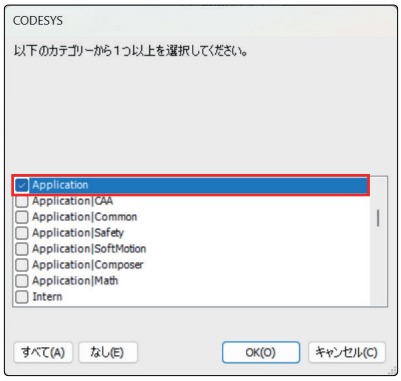


「ライブラリカテゴリー」欄の右端の「…」をクリックして、ライブラリカテゴリー画面を開きます。「追加」→「ディスクリプションファイルから…」をクリックして、以下のファイルを選択します。

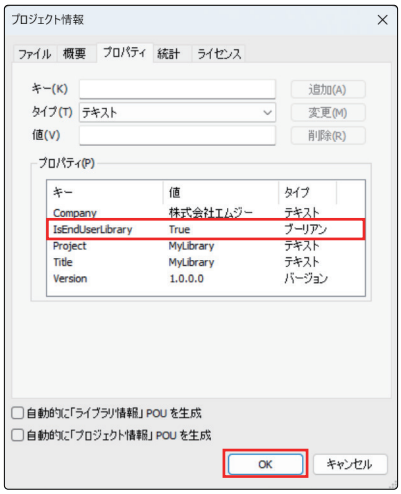


・ディスクリプションファイル
C:¥Program Files¥CODESYS 3.5.21.10¥CODESYS¥Templates¥Library_Template¥LibraryCategoryBase.libcat.xml

カテゴリーの一覧から「Application」のみを選択して「OK」をクリックします。

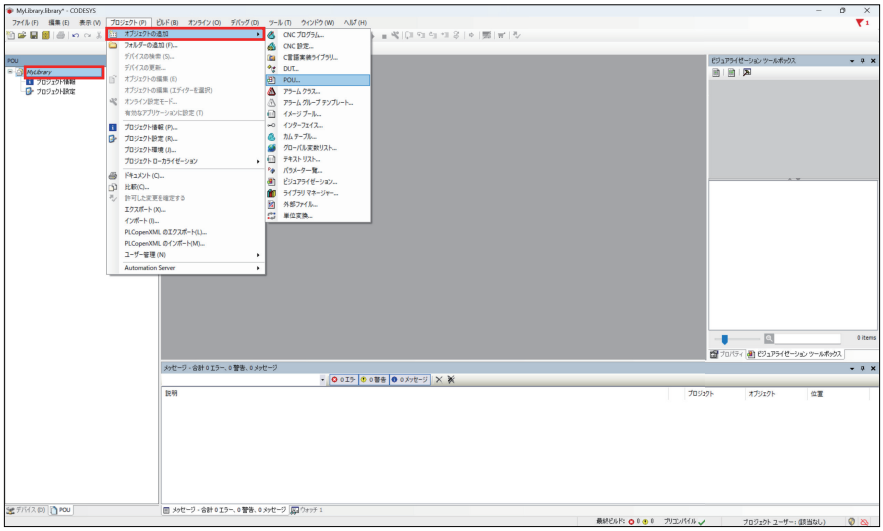


プロジェクト情報画面のプロパティタブでユーザライブラリキー「IsEndUserLibrary」を追加して、「OK」をクリックします。



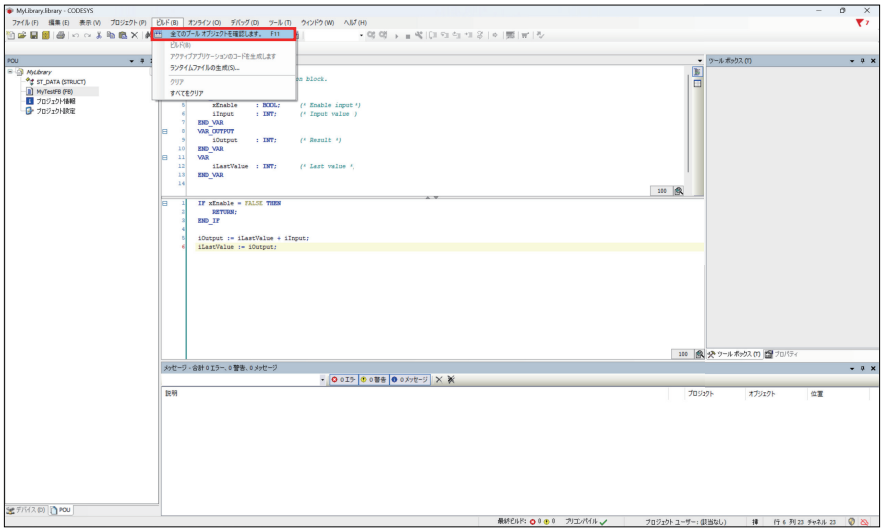
7.1.3 ライブラリにオブジェクトを追加

デバイスツリーのライブラリ（MyLibrary）を選択した状態で、メニューの「プロジェクト」→「オブジェクトの追加」をクリックします。表示されたオブジェクトの一覧から、DUT や POU などライブラリに追加したいオブジェクトを選択して追加します。



7.1.4 ライブラリのエラー確認

メニューから「ビルド」→「全てのプールオブジェクトを確認します。」をクリックして、ライブラリのエラー確認を行います。



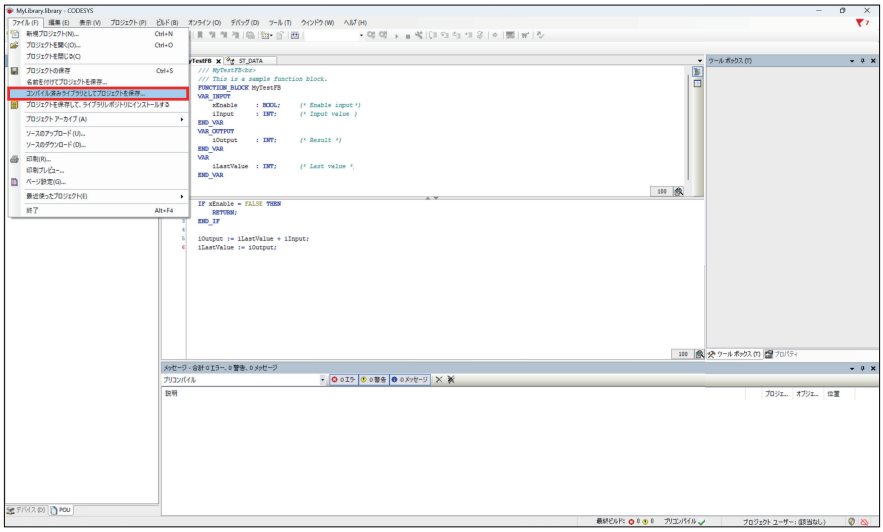
エラーや警告が検出された場合は、解消されるまで問題個所の修正とエラー確認を繰り返して下さい。

7.1.5 ライブラリの種類

ライブラリは、作成時に使用しているライブラリプロジェクトをそのまま使用･配布することもできますが、ソースコードを公開しない「コンパイル済みライブラリ」として作成することも可能です。
ライブラリには次の種類があります。

種　類	内　容
ライブラリプロジェクト (*.library)	ソースコード参照可能 (プロテクトオプションで変更可能)
コンパイル済みライブラリ (*.compiled-library)	ソースコード参照不可

コンパイル済みライブラリ（*.compiled-library）を作成するには、CODESYS IDE でライブラリプロジェクト（*.library）を開き、メニューから「ファイル」→「コンパイル済みライブラリとしてプロジェクトを保存…」をクリックします。



作成されたコンパイル済みライブラリは自動的にリポジトリへ登録されません。

7.1.6 ライブラリの公開（リポジトリ登録）

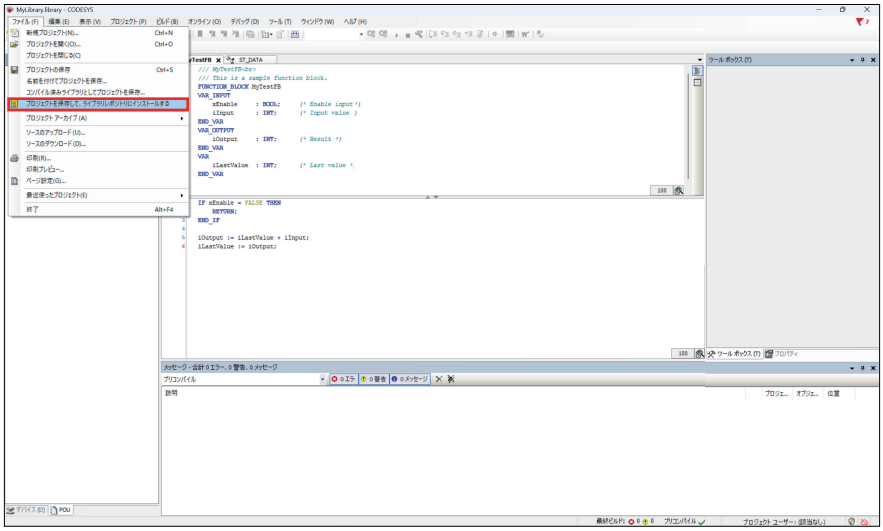
作成したライブラリや配布されたライブラリをアプリケーションで使用するには、ライブラリリポジトリへ登録する必要があります。

■外部から配布されたライブラリを登録する場合

メニューから「ツール」→「ライブラリリポジトリ…」をクリックします。表示されるライブラリリポジトリ画面の「インストール…」をクリックし、登録したいライブラリ（*.library または *.compiled-library）を選択します。詳細は「2.2.3 ライブラリ」を参照して下さい。

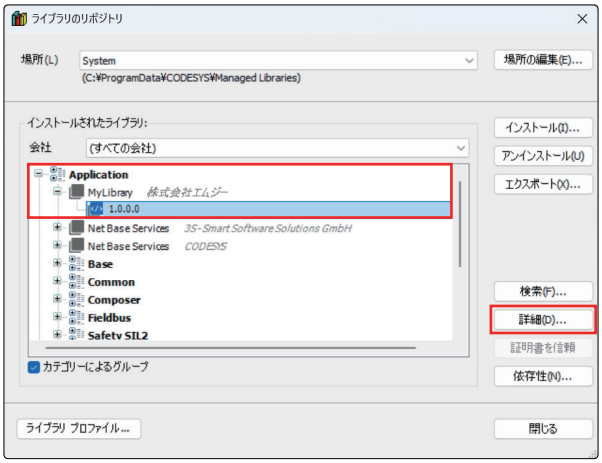
■CODESYS IDE で開いているライブラリプロジェクトを登録する場合

メニューから「ファイル」→「プロジェクトを保存して、ライブラリリポジトリにインストールする」をクリックします。

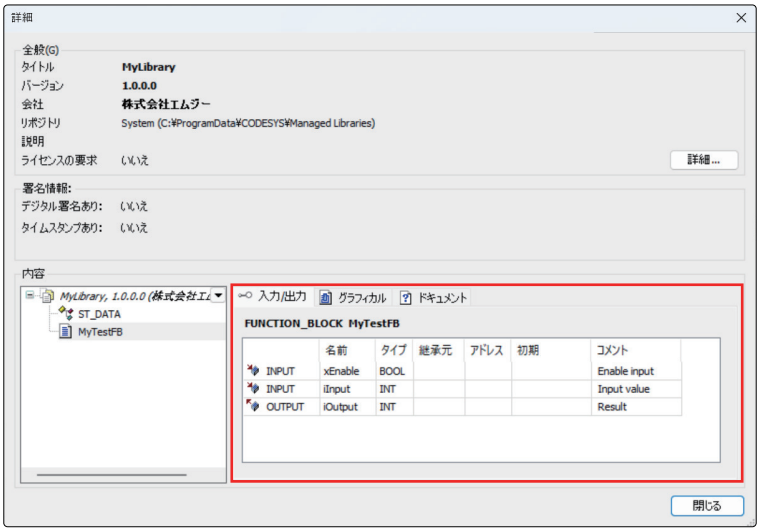


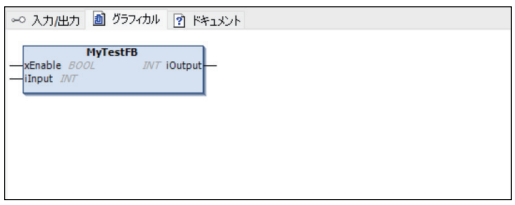
7.1.7 公開されているライブラリの確認

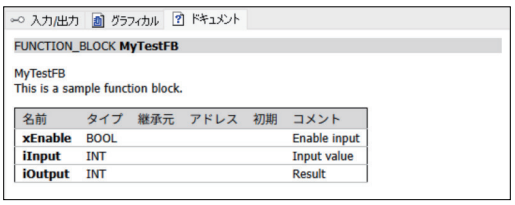
登録済みライブラリの情報は、「ライブラリリポジトリ」で確認できます。
メニューから「ツール」→「ライブラリリポジトリ…」をクリックすると、登録済みライブラリの一覧が表示されます。
作成した MyLibrary は「インストールされたライブラリ」の Application カテゴリーに登録されています。



ライブラリの詳細を確認するには、対象ライブラリを選択して「詳細…」をクリックします。
詳細画面ではライブラリに含まれる POU や DUT がツリー形式で表示され、ファンクションブロックの入出力パラメータ、グラフィックエディタでの外観、および説明文を確認できます。







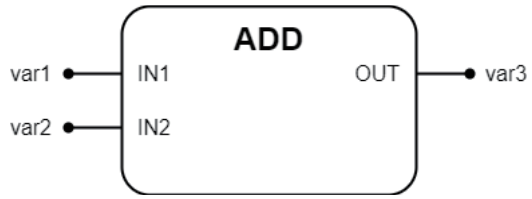
ファンクションブロックに記述したコメントは、説明欄に自動的に反映されます。
プログラム作成時にこの自動ドキュメント化機能を意識してコメントを記述することで、個別にドキュメントを作成する手間を省き、プログラムとドキュメントの一元管理が可能になります。

7.2 演算子

分　類	命　令	機　能
算術演算	ADD	加算
	SUB	減算
	MUL	乗算
	DIV	除算
	MOD	剰余
	MOVE	転送
	XSIZEOF *1	変数の占有サイズ取得
論理演算	AND	論理積
	OR	論理和
	XOR	排他的論理和
	NOT	ビット反転
	AND_THEN *1	短絡評価の論理積
	OR_ELSE *1	短絡評価の論理和
ビットシフト演算	SHL	左ビットシフト
	SHR	右ビットシフト
	ROL	左ビットローテイト
	ROR	右ビットローテイト
選択演算	SEL	データ選択
	MAX	最大値選択
	MIN	最小値選択
	LMIT	上下限制限
	MUX	マルチプレクサ
比較演算	GT	より大きい
	LT	より小さい
	LE	以下
	GE	以上
	EQ	等しい
	NE	等しくない
アドレス演算	ADR *1	アドレス取得
	BITADR *1	ビットアドレス取得
呼び出し演算	CAL	ファンクションブロック呼び出し
数値演算	ABS	絶対値
	SQRT	平方根
	LN	自然対数
	LOG	常用対数
	EXP	e のべき乗
	EXPT	べき乗
	SIN	正弦(サイン)
	ASIN	逆正弦(アークサイン)
	COS	余弦(コサイン)
	ACOS	逆余弦(アークコサイン)
	TAN	正接(タンジェント)
	ATAN	逆正接(アークタンジェント)

＊1、EC61131-3 非準拠

7.2.1 ADD：加算



■機能

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

■入出力パラメータ

入力パラメータ	説　明	備　考
IN1(var1)	被加数	
IN2(var2)	加数	

出力パラメータ	説　明	備　考
OUT(var3)	加算結果	

FBD エディタおよび LD エディタでは、ADD 演算子の入力数を拡張することが可能です。

■使用可能なデータ型

BYTE、WORD、DWORD、LWORD、SINT、USINT、INT、UINT、DINT、UDINT、LINT、ULINT、TIME、LTIME、DATE、LDATE、DATE_AND_TIME（DT）、LDATE_AND_TIME（LDT）、TIME_OF_DAY（TOD）、LTIME_OF_DAY（LTOD）、REAL、LREAL

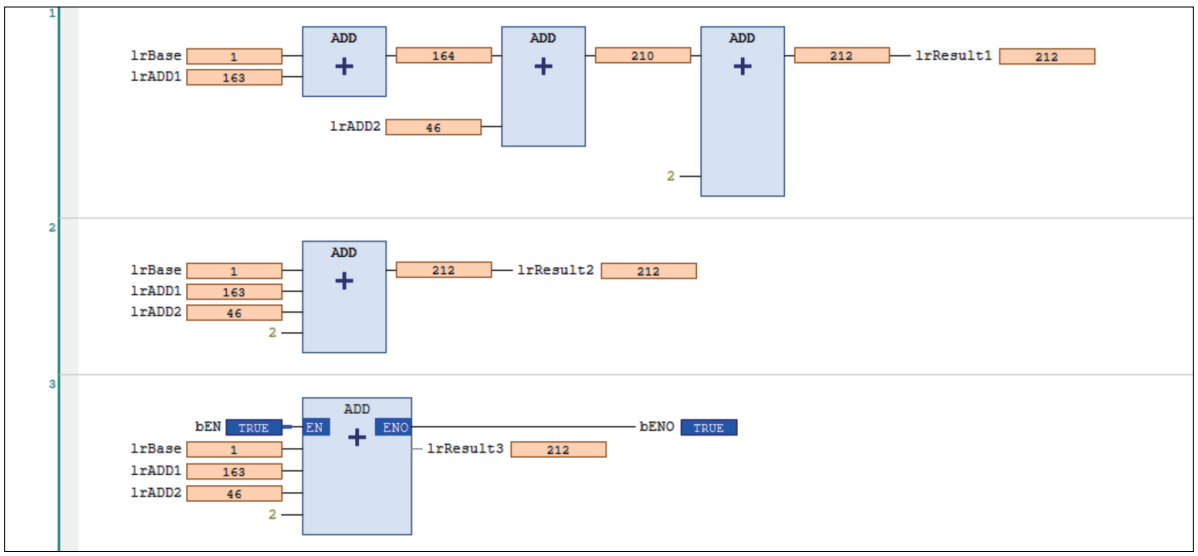
時間および日付時刻のデータ型では、以下のような組み合わせが使用可能です。

・TIME ＋ TIME ＝ TIME
・TIME ＋ LTIME ＝ LTIME
・LTIME ＋ LTIME ＝ LTIME

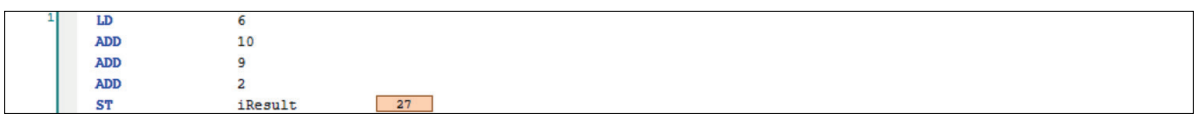
・TOD ＋ TIME ＝ TOD
・DT ＋ TIME ＝ DT
・TOD ＋ LTIME ＝ LTOD
・DT ＋ LTIME ＝ LDT
・LTOD ＋ TIME ＝ LTOD
・LDT ＋ LTIME ＝ LDT
・LTOD ＋ LTIME ＝ LTOD
・LDT ＋ LTIME ＝ LDT

■例

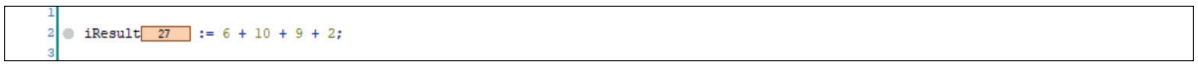
FBD 言語



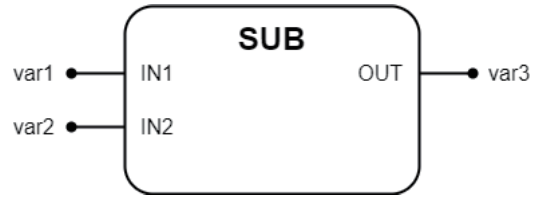
IL 言語



ST 言語



7.2.2 SUB：減算



■機能

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

■入出力パラメータ

入力パラメータ	説　明	備　考
IN1(var1)	被減数	
IN2(var2)	減数	

出力パラメータ	説　明	備　考
OUT(var3)	減算結果	

■使用可能なデータ型

BYTE、WORD、DWORD、LWORD、SINT、USINT、INT、UINT、DINT、UDINT、LINT、ULINT、TIME、LTIME、DATE、LDATE、DATE_AND_TIME（DT）、LDATE_AND_TIME（LDT）、TIME_OF_DAY（TOD）、LTIME_OF_DAY（LTOD）、REAL、LREAL、

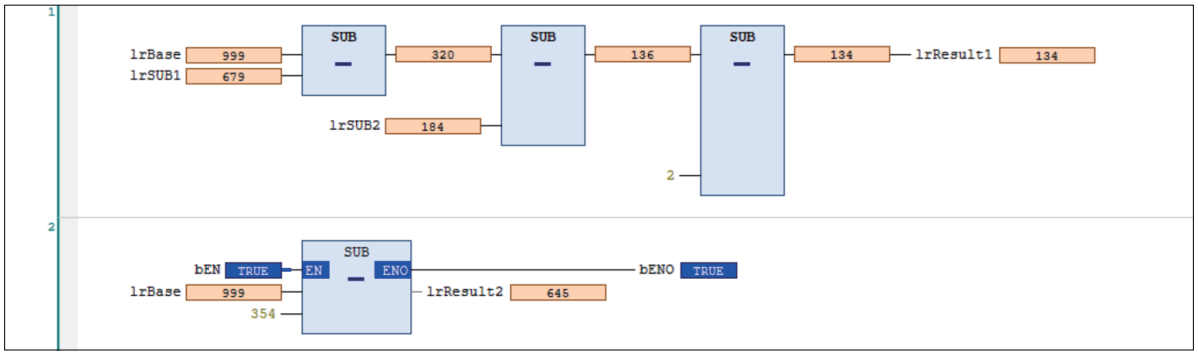
日付および時刻のデータ型では、以下のような組み合わせが使用可能です。
・TIME － TIME ＝ TIME
・LTIME － LTIME ＝ LTIME

・DATE － DATE ＝ TIME
・LDATE － LDATE ＝ LTIME
・TOD － TIME ＝ TOD
・LTOD － LTIME ＝ LTOD
・TOD － TOD ＝ TIME
・LTOD － LTOD ＝ LTIME
・DT － TIME ＝ DT
・LDT － LTIME ＝ LDT
・DT － DT ＝ TIME
・LDT － LDT ＝ LTIME

TIME 型および LTIME 型では、負の値は未定義となります。

■例

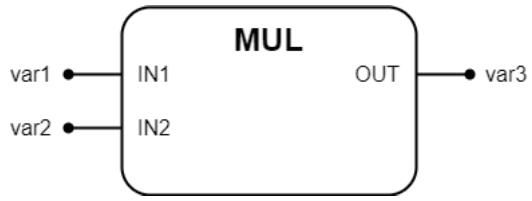
FBD 言語



IL 言語

ST 言語

7.2.3 MUL：乗算



■機能

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

■入出力パラメータ

入力パラメータ	説　明	備　考
N1(var1)	被乗数	
IN2(var2)	乗数	

出力パラメータ	説　明	備　考
OUT(var3)	乗算結果	

FBD エディタおよび LD エディタでは、MUL 演算子の入力数を拡張することが可能です。

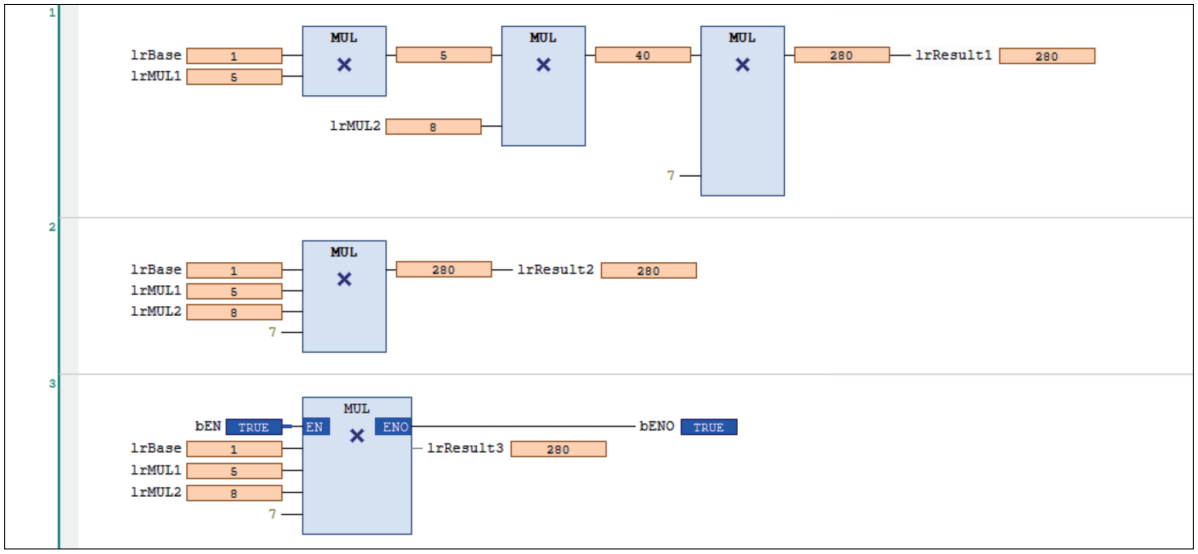
■使用可能なデータ型

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

TIME 型および LTIME 型は、整数型の値とのみ乗算が可能です。

■例

FBD 言語



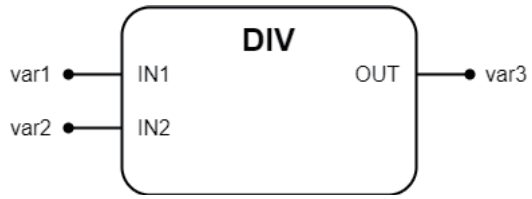
IL 言語

```
1   LD      9
    MUL     4
    MUL     3
    MUL     6
    ST      iResult     648
```

ST 言語

```
1
2   iResult 648 := 9 * 4 * 3 * 6;
3
```

7.2.4 DIV：除算



■機能

入力パラメータ IN1、IN2 に接続された値の除算結果（IN1 ÷ IN2）を出力パラメータ OUT に返します。
ゼロ除算については「8.1 ゼロ除算」を参照して下さい。

■入出力パラメータ

入力パラメータ	説　明	備　考
N1(var1)	被除数	
IN2(var2)	除数	

出力パラメータ	説　明	備　考
OUT(var3)	除算結果	

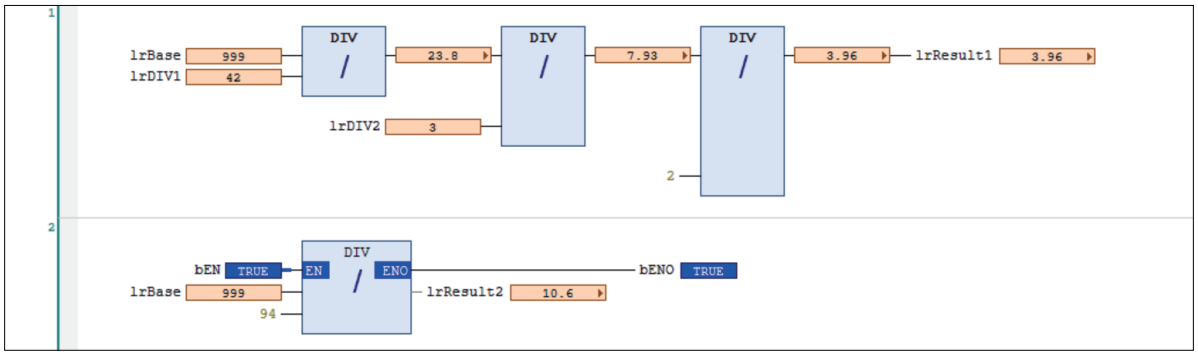
■使用可能なデータ型

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

TIME 型および LTIME 型は、整数型の値とのみ除算が可能です。

■例

FBD 言語



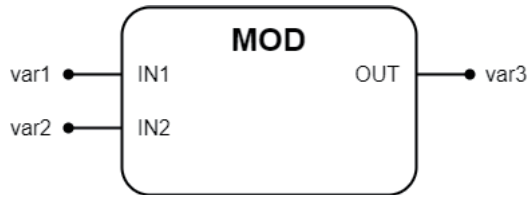
IL 言語

```
1   LD      9
    DIV     4
    ST      iResult     2
```

ST 言語

```
1
2  iResult 2 := 9 / 4;
3
```

7.2.5 MOD：剰余



■機能

入力パラメータ IN1、IN2 に接続された値の除算結果（IN1 % IN2）を出力パラメータ OUT に返します。
ゼロ除算については「8.1 ゼロ除算」を参照して下さい。

■入出力パラメータ

入力パラメータ	説　明	備　考
IN1(var1)	被除数	
IN2(var2)	除数	

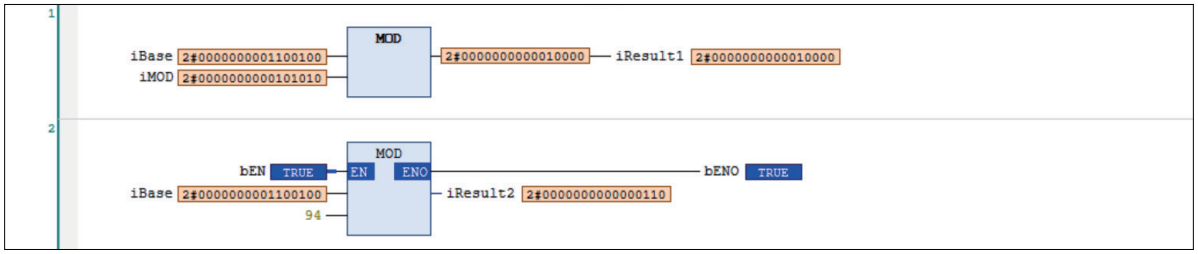
出力パラメータ	説　明	備　考
OUT(var3)	剰余結果	

■使用可能なデータ型

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

■例

FBD 言語



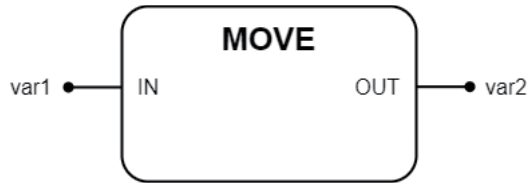
IL 言語

```
1  LD      9
   MOD     4
   ST      iResult  1
```

ST 言語

```
1
2  iResult 1 := 9 MOD 4;
3
```

7.2.6 MOVE：転送



■機能

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

■入出力パラメータ

入力パラメータ	説　明	備　考
IN(var1)	入力	

出力パラメータ	説　明	備　考
OUT(var2)	出力	IN と同じデータ型であること

■使用可能なデータ型

すべてのデータ型で使用可能です。

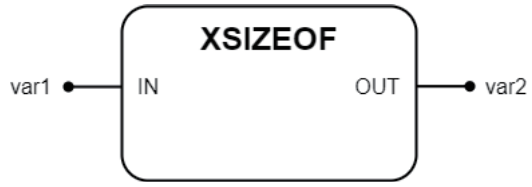
■例

FBD 言語

IL 言語

ST 言語

7.2.7 XSIZEOF：変数の占有サイズ取得



■機能

入力パラメータ IN が占有するメモリ領域のサイズを、バイト単位で出力パラメータ OUT に返します。

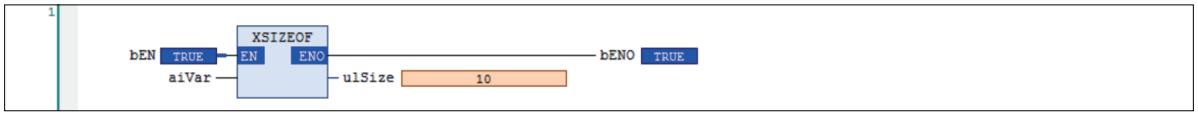
■入出力パラメータ

入力パラメータ	説　明	備　考
IN(var1)	変数	

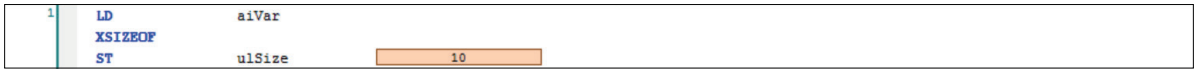
出力パラメータ	説　明	備　考
OUT(var2)	メモリ領域のサイズ	

■使用可能なデータ型
IN：すべてのデータ型で使用可能です。
OUT：ULINT

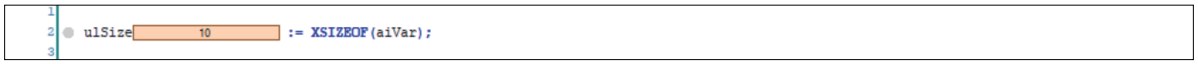
■例
FBD 言語



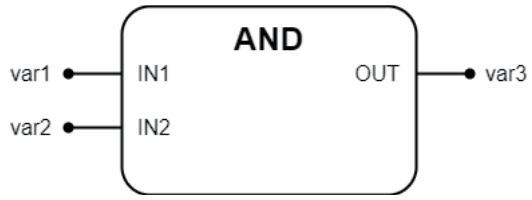
IL 言語



ST 言語



7.2.8 AND：論理積



■機能
入力パラメータ IN1、IN2 に接続された値の論理積（IN1 AND IN2）を出力パラメータ OUT に返します。

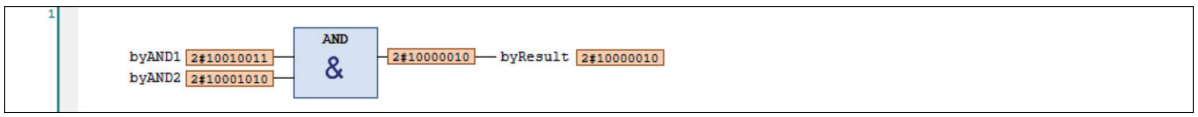
■入出力パラメータ

入力パラメータ	説　明	備　考
IN1(var1)	入力 1	
IN2(var2)	入力 2	IN1 と同じデータ型であること

出力パラメータ	説　明	備　考
OUT(var3)	論理積	IN1 と同じデータ型であること

■使用可能なデータ型
BOOL、BYTE、WORD、DWORD、LWORD

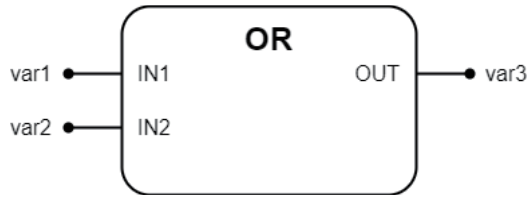
■例
FBD 言語



IL 言語

ST 言語

7.2.9 OR：論理和



■機能

入力パラメータ IN1、IN2 に接続された値の論理和（IN1 OR IN2）を出力パラメータ OUT に返します。

■入出力パラメータ

入力パラメータ	説　明	備　考
IN1(var1)	入力 1	
IN2(var2)	入力 2	IN1 と同じデータ型であること

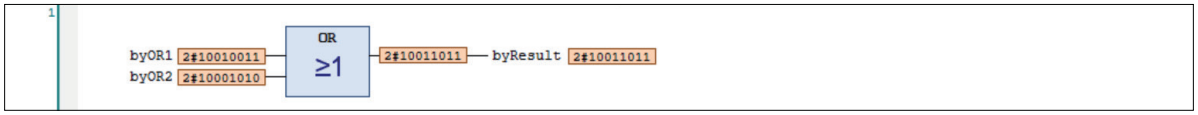
出力パラメータ	説　明	備　考
OUT(var3)	論理和	IN1 と同じデータ型であること

■使用可能なデータ型

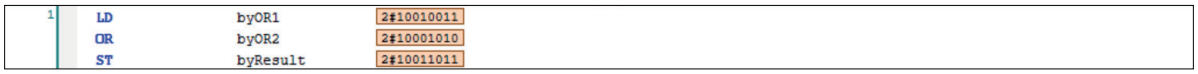
BOOL、BYTE、WORD、DWORD、LWORD

■例

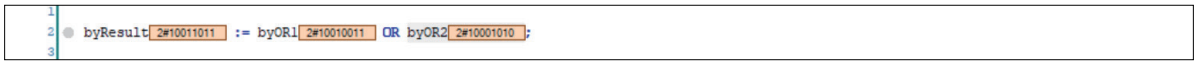
FBD 言語



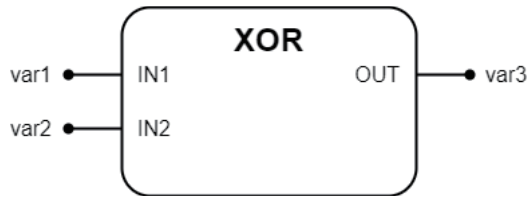
IL 言語



ST 言語



7.2.10 XOR：排他的論理和



■機能

入力パラメータ IN1、IN2 に接続された値の排他的論理和（IN1 XOR IN2）を出力パラメータ OUT に返します。

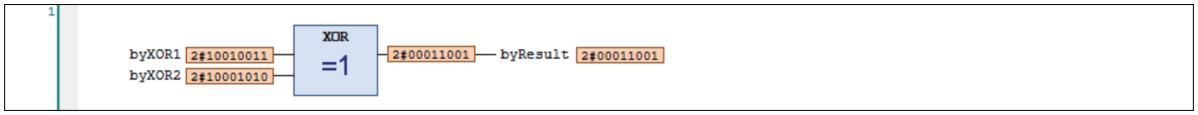
■入出力パラメータ

入力パラメータ	説　明	備　考
IN1(var1)	入力 1	
IN2(var2)	入力 2	IN1 と同じデータ型であること

出力パラメータ	説　明	備　考
OUT(var3)	排他的論理和	IN1 と同じデータ型であること

■使用可能なデータ型
BOOL、BYTE、WORD、DWORD、LWORD

■例
FBD 言語



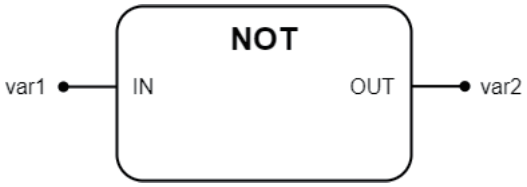
IL 言語

```
LD      byXOR1      2#10010011
XOR     byXOR2      2#10001010
ST      byResult    2#00011001
```

ST 言語

```
byResult 2#00011001 := byXOR1 2#10010011 XOR byXOR2 2#10001010;
```

7.2.11 NOT：ビット反転



■機能
入力パラメータ IN に接続された値のビット反転結果（NOT IN）を出力パラメータ OUT に返します。

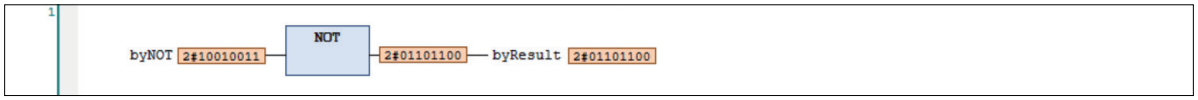
■入出力パラメータ

入力パラメータ	説　明	備　考
IN(var1)	入力	

出力パラメータ	説　明	備　考
OUT(var2)	ビット反転	IN と同じデータ型であること

■使用可能なデータ型
BOOL、BYTE、WORD、DWORD、LWORD

■例
FBD 言語



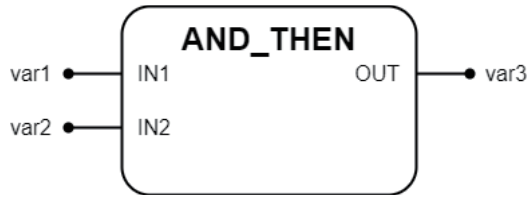
IL 言語

```
LD      byNOT       2#10010011
NOT
ST      byResult    2#01101100
```

ST 言語

```
byResult 2#01101100 := NOT byNOT 2#10010011;
```

7.2.12 AND_THEN：短絡評価の論理積



■機能

入力パラメータ IN1 に接続された値が FALSE の場合、入力パラメータ IN2 に接続された値の評価を行わずに出力パラメータ OUT に FALSE を返します。IN1 が TRUE の場合のみ、IN2 を評価してその結果を OUT に返します。AND 演算子では IN1 の値に関わらず、すべての入力が評価されます。

■入出力パラメータ

入力パラメータ	説　明	備　考
IN1(var1)	入力 1	
IN2(var2)	入力 2	

出力パラメータ	説　明	備　考
OUT(var3)	論理積	

■使用可能なデータ型

BOOL

■例

ST 言語（AND_THEN）

```
1
2   bX := (piVar^ <> 0) AND_THEN (piVar^ = 10);
3
```

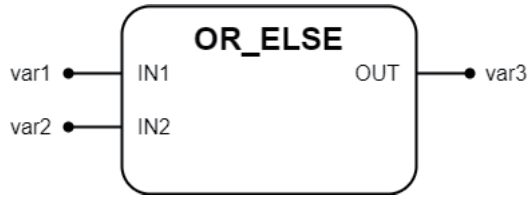
IN1（piVar＜＞0）が FALSE のため、IN2（piVar^＝10）の評価は行われません。

ST 言語（AND）

```
3
4   bX := (piVar^ <> 0) AND (piVar^ = 10);
5
```

IN1（piVar＜＞0）に関わらず、IN2（piVar^＝10）の評価を行います。

7.2.13 OR_ELSE：短絡評価の論理和



■機能

入力パラメータ IN1 に接続された値が TRUE の場合、入力パラメータ IN2 に接続された値の評価を行わずに出力パラメータ OUT に TRUE を返します。IN1 が FALSE の場合のみ、IN2 を評価してその結果を OUT に返します。OR 演算子では IN1 の値に関わらず、すべての入力が評価されます。

■入出力パラメータ

入力パラメータ	説　明	備　考
IN1(var1)	入力 1	
IN2(var2)	入力 2	

出力パラメータ	説　明	備　考
OUT(var3)	論理積	

■使用可能なデータ型
BOOL

■例
ST 言語（OR_ELSE）

```
bX := (wVar <> 16#0000) OR_ELSE (bFlg := TRUE);
```

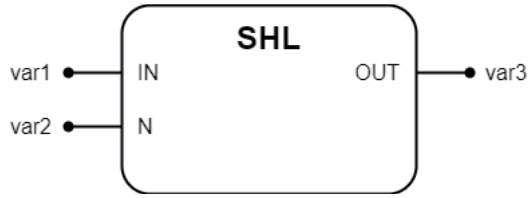
IN1（wVar＜＞16#0000）が TRUE のため、IN2（bFlg:＝TRUE）の評価は行われません。

ST 言語（OR）

```
bX := (wVar <> 16#0000) OR (bFlg := TRUE);
```

IN1（wVar＜＞16#0000）に関わらず、IN2（bFlg:＝TRUE）の評価を行います。

7.2.14 SHL：左ビットシフト



■機能
入力パラメータ IN に接続された値を、入力パラメータ N に接続されたビット数だけ左シフト（IN＜＜N）して、出力パラメータ OUT に返します。

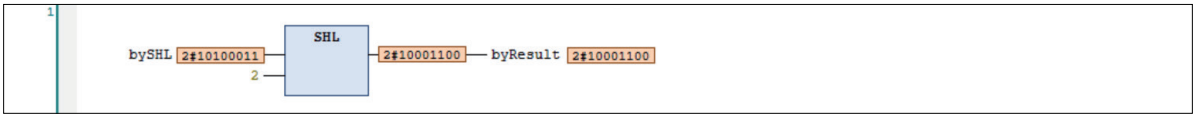
■入出力パラメータ

入力パラメータ	説　明	備　考
IN(var1)	シフトされるデータ	
N(var2)	シフトするビット数	

出力パラメータ	説　明	備　考
OUT(var3)	シフト演算結果	

■使用可能なデータ型
BYTE、WORD、DWORD、LWORD、SINT、USINT、INT、UINT、DINT、UDINT、LINT、ULINT

■例
FBD 言語



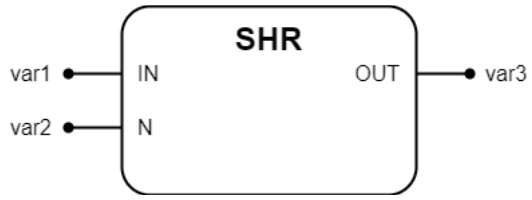
IL 言語

```
LD      bySHL       2#10100011
SHL     2
ST      byResult    2#10001100
```

ST 言語

```
byResult := SHL(bySHL, 2);
```

7.2.15 SHR：右ビットシフト



■機能

入力パラメータ IN に接続された値を、入力パラメータ N に接続されたビット数だけ右シフト（IN＞＞N）して、出力パラメータ OUT に返します。

■入出力パラメータ

入力パラメータ	説　明	備　考
IN(var1)	シフトされるデータ	
N(var2)	シフトするビット数	

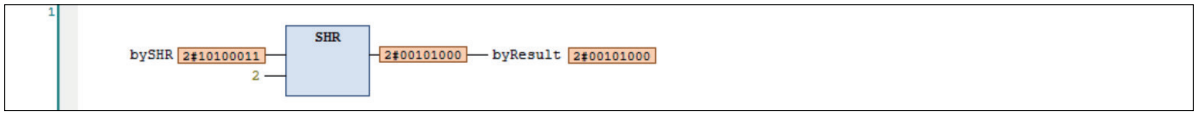
出力パラメータ	説　明	備　考
OUT(var3)	シフト演算結果	

■使用可能なデータ型

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

■例

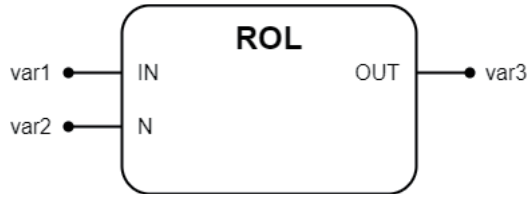
FBD 言語



IL 言語

ST 言語

7.2.16 ROL：左ビットローテイト



■機能

入力パラメータ IN に接続された値を、入力パラメータ N に接続されたビット数だけ左ローテイトして、出力パラメータ OUT に返します。

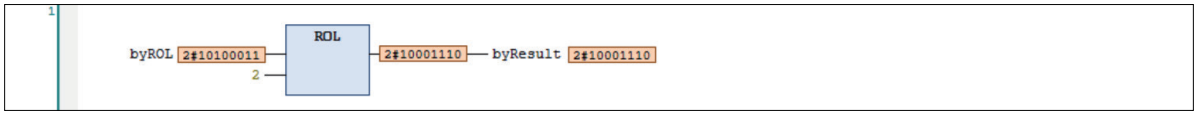
■入出力パラメータ

入力パラメータ	説　明	備　考
IN(var1)	ローテイトされるデータ	
N(var2)	ローテイトするビット数	

出力パラメータ	説　明	備　考
OUT(var3)	ローテイト演算結果	

■使用可能なデータ型
BYTE、WORD、DWORD、LWORD、SINT、USINT、INT、UINT、DINT、UDINT、LINT、ULINT

■例
FBD 言語



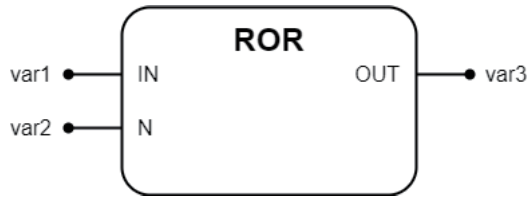
IL 言語

```
LD      byROL     2#10100011
ROL     2
ST      byResult  2#10001110
```

ST 言語

```
byResult := ROL(byROL, 2);
```

7.2.17 ROR：右ビットローテイト



■機能
入力パラメータ IN に接続された値を、入力パラメータ N に接続されたビット数だけ右ローテイトして、出力パラメータ OUT に返します。

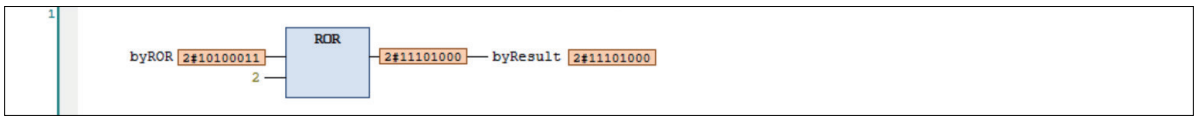
■入出力パラメータ

入力パラメータ	説　明	備　考
IN(var1)	ローテイトされるデータ	
N(var2)	ローテイトするビット数	

出力パラメータ	説　明	備　考
OUT(var3)	ローテイト演算結果	

■使用可能なデータ型
BYTE、WORD、DWORD、LWORD、SINT、USINT、INT、UINT、DINT、UDINT、LINT、ULINT

■例
FBD 言語



IL 言語

```
LD      byROR     2#10100011
ROR     2
ST      byResult  2#11101000
```

ST 言語

```
byResult := ROR(byROR, 2);
```

7.2.18 SEL：データ選択

■機能

入力パラメータ G に接続された値が FALSE の場合は入力パラメータ IN0、TRUE の場合は入力パラメータ IN1 に接続された値を出力パラメータ OUT に返します。

■入出力パラメータ

入力パラメータ	説　明	備　考
G(var1)	選択入力	
IN0(var2)	選択肢 0(G＝FALSE)	
IN1(var3)	選択肢 1(G＝TRUE)	IN0 と同じデータ型であること

出力パラメータ	説　明	備　考
OUT(var4)	選択結果	IN0 と同じデータ型であること

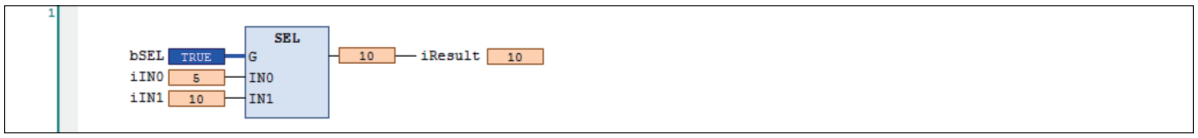
■使用可能なデータ型

G：BOOL

IN0、IN1、OUT：すべてのデータ型で使用可能です。

■例

FBD 言語

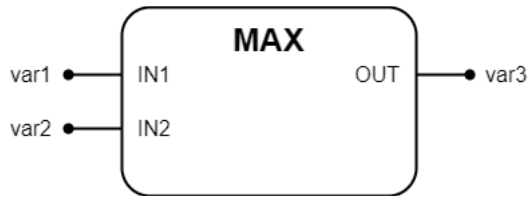


IL 言語

ST 言語

```
1
2  iResult 10 := SEL(bSEL TRUE, iIN0 5, iIN1 10);
3
```

7.2.19 MAX：最大値選択



■機能

入力パラメータ IN1、IN2 に接続された値のうち最も大きな値を出力パラメータ OUT に返します。

■入出力パラメータ

入力パラメータ	説　明	備　考
IN1(var1)	入力 1	
IN2(var2)	入力 2	IN1 と同じデータ型であること

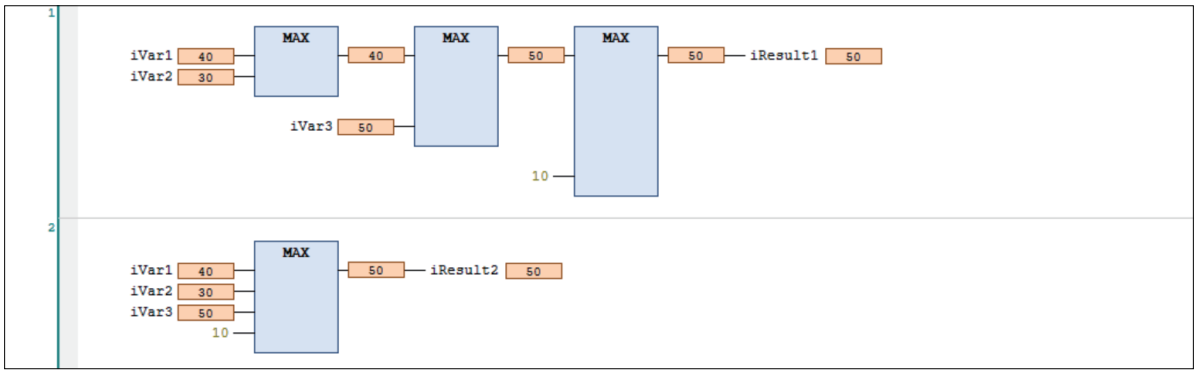
出力パラメータ	説　明	備　考
OUT(var3)	最大値	IN1 と同じデータ型であること

■使用可能なデータ型

すべてのデータ型で使用可能です。

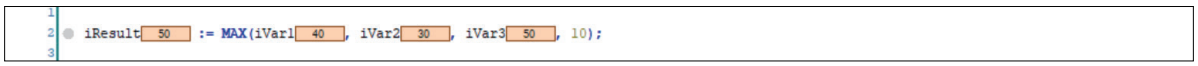
■例

FBD 言語

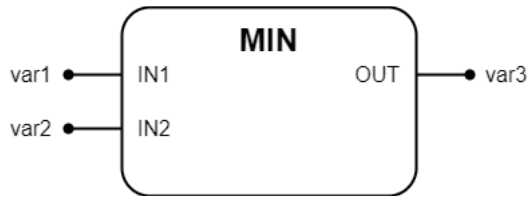


IL 言語

ST 言語



7.2.20 MIN：最小値選択



■機能

入力パラメータ IN1、IN2 に接続された値のうち最も小さな値を出力パラメータ OUT に返します。

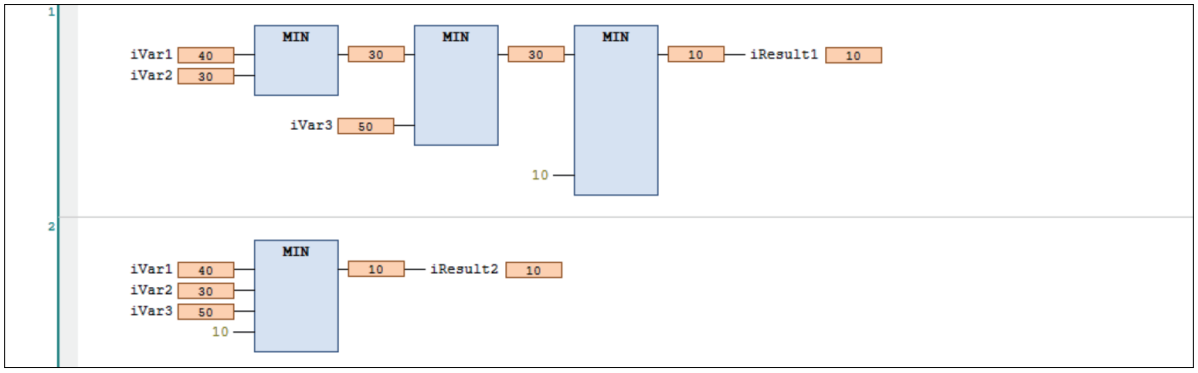
■入出力パラメータ

入力パラメータ	説　明	備　考
IN1(var1)	入力 1	
IN2(var2)	入力 2	IN1 と同じデータ型であること

出力パラメータ	説　明	備　考
OUT(var3)	最小値	IN1 と同じデータ型であること

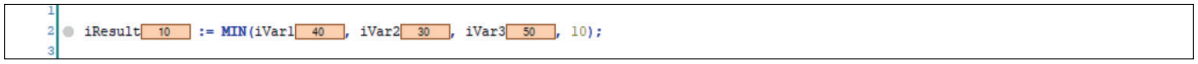
■使用可能なデータ型
すべてのデータ型で使用可能です。

■例
FBD 言語

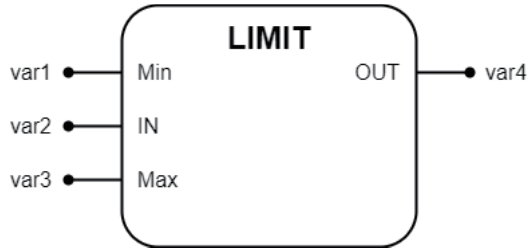


IL 言語

ST 言語



7.2.21 LMIT：上下限制限



■機能
入力パラメータ IN に接続された値を、入力パラメータ Min および Max の範囲内に制限して、出力パラメータ OUT に返します。

■入出力パラメータ

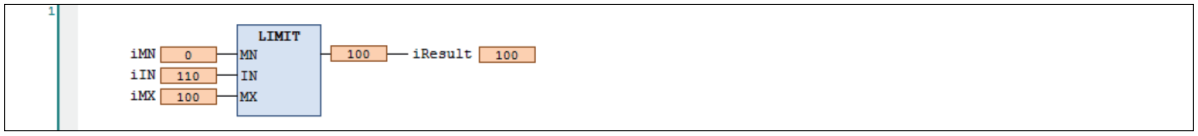
入力パラメータ	説　明	備　考
Min（var1）	最小値	IN と同じデータ型であること
IN（var2）	入力	
Max	最大値	IN と同じデータ型であること

出力パラメータ	説　明	備　考
OUT（var3）	上下限制限結果	IN と同じデータ型であること

■使用可能なデータ型
すべてのデータ型で使用可能です。

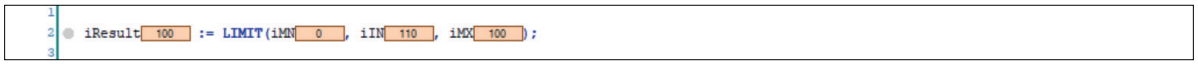
■例

FBD 言語

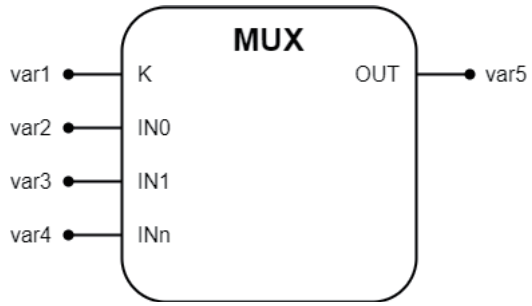


IL 言語

ST 言語



7.2.22 MUX：マルチプレクサ



■機能

入力パラメータ K に接続された値に応じて、入力パラメータ IN0、IN1、INn のいずれかの値を出力パラメータ OUT に返します。

■入出力パラメータ

入力パラメータ	説　明	備　考
K(var1)	選択入力	
IN0(var2)	選択肢 1(K=0)	
IN1(var3)	選択肢 2(K=1)	IN0 と同じデータ型であること
INn(var4)	選択肢 n(K=n－1)	IN0 と同じデータ型であること

出力パラメータ	説　明	備　考
OUT(var5)	選択結果	IN0 と同じデータ型であること

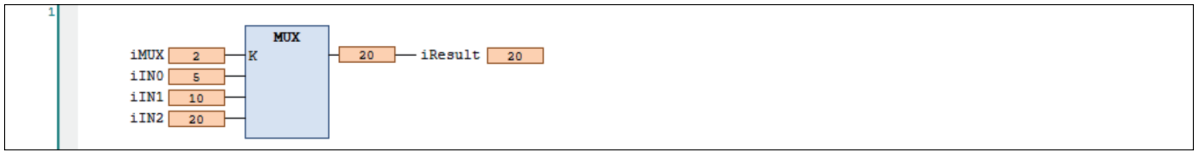
■使用可能なデータ型

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

IN0、IN1、INn、OUT：すべてのデータ型で使用可能です。

■例

FBD 言語



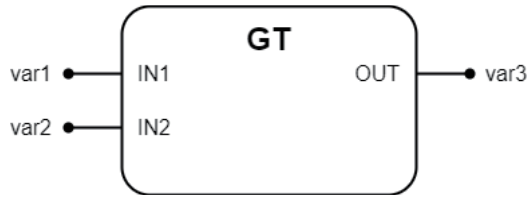
IL 言語

```
1   LD      iMUX    2
    MUX     iIN0    5
            iIN1    10
            iIN2    20
    ST      iResult 20
```

ST 言語

```
1
2   iResult 20 := MUX(iMUX 2, iIN0 5, iIN1 10, iIN2 20);
3
```

7.2.23 GT：より大きい



■機能

入力パラメータ IN1、IN2 に接続された値の比較結果（IN1＞IN2）を出力パラメータ OUT に返します。

■入出力パラメータ

入力パラメータ	説　明	備　考
IN1(var1)	入力 1	
IN2(var2)	入力 2	IN1 と同じデータ型であること

出力パラメータ	説　明	備　考
OUT(var3)	比較結果	TRUE：IN1＞IN2 FALSE：IN1＜＝IN2

■使用可能なデータ型

IN1、IN2：すべてのデータ型で使用可能です。
OUT：BOOL

■例

FBD 言語

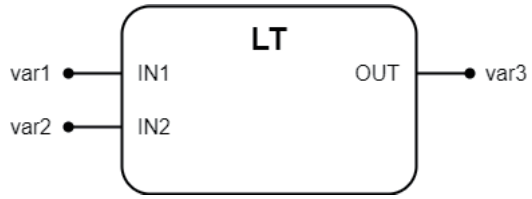
IL 言語

```
1   LD      iIN1    10
    GT      iIN2    20
    ST      bResult FALSE
```

ST 言語

```
1
2   bResult FALSE := iIN1 10 > iIN2 20;
3
```

7.2.24 LT：より小さい



■機能

入力パラメータ IN1、IN2 に接続された値の比較結果（IN1＜IN2）を出力パラメータ OUT に返します。

■入出力パラメータ

入力パラメータ	説　明	備　考
IN1(var1)	入力 1	
IN2(var2)	入力 2	IN1 と同じデータ型であること

出力パラメータ	説　明	備　考
OUT(var3)	比較結果	TRUE：IN1＜IN2 FALSE：IN1＞＝IN2

■使用可能なデータ型

IN1、IN2：すべてのデータ型で使用可能です。
OUT：BOOL

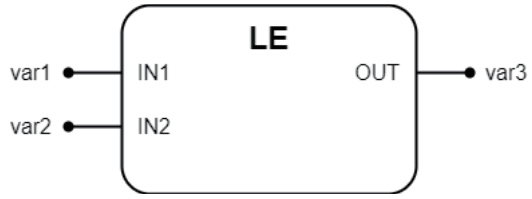
■例

FBD 言語

IL 言語

ST 言語

7.2.25 LE：以下



■機能

入力パラメータ IN1、IN2 に接続された値の比較結果（IN1＜＝IN2）を出力パラメータ OUT に返します。

■入出力パラメータ

入力パラメータ	説　明	備　考
IN1(var1)	入力 1	
IN2(var2)	入力 2	IN1 と同じデータ型であること

出力パラメータ	説　明	備　考
OUT(var3)	比較結果	TRUE：IN1＜＝IN2 FALSE：IN1＞IN2

■使用可能なデータ型
IN1、IN2：すべてのデータ型で使用可能です。
OUT：BOOL

■例
FBD 言語

IL 言語

ST 言語

7.2.26 GE：以上

■機能
入力パラメータ IN1、IN2 に接続された値の比較結果（IN1＞＝N2）を出力パラメータ OUT に返します。

■入出力パラメータ

入力パラメータ	説　明	備　考
IN1(var1)	入力 1	
IN2(var2)	入力 2	IN1 と同じデータ型であること

出力パラメータ	説　明	備　考
OUT(var3)	比較結果	TRUE：IN1＞＝IN2 FALSE：IN1＜IN2

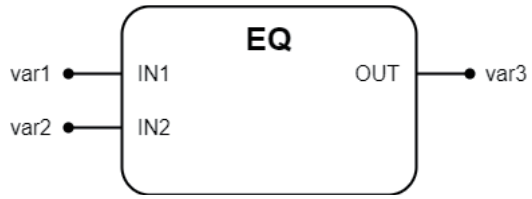
■使用可能なデータ型
IN1、IN2：すべてのデータ型で使用可能です。
OUT：BOOL

■例
FBD 言語

IL 言語

ST 言語

7.2.27 EQ：等しい



■機能

入力パラメータ IN1、IN2 に接続された値の比較結果（IN1＝N2）を出力パラメータ OUT に返します。

■入出力パラメータ

入力パラメータ	説　明	備　考
IN1(var1)	入力 1	
IN2(var2)	入力 2	IN1 と同じデータ型であること

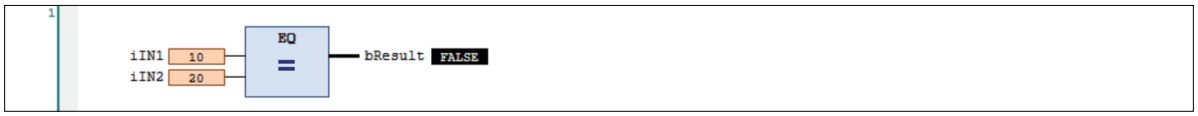
出力パラメータ	説　明	備　考
OUT(var3)	比較結果	TRUE：IN1＝IN2 FALSE：IN1＜＞IN2

■使用可能なデータ型

IN1、IN2：すべてのデータ型で使用可能です。
OUT：BOOL

■例

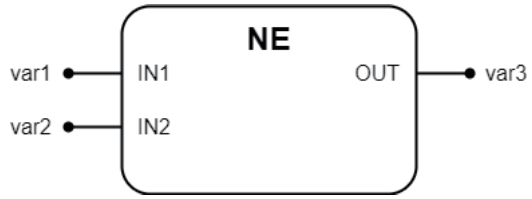
FBD 言語



IL 言語

ST 言語

7.2.28 NE：等しくない



■機能

入力パラメータ IN1、IN2 に接続された値の比較結果（IN1＜＞IN2）を出力パラメータ OUT に返します。

■入出力パラメータ

入力パラメータ	説　明	備　考
IN1(var1)	入力 1	
IN2(var2)	入力 2	IN1 と同じデータ型であること

出力パラメータ	説　明	備　考
OUT(var3)	比較結果	TRUE：IN1＜＞IN2 FALSE：IN1＝IN2

■使用可能なデータ型

IN1、IN2：すべてのデータ型で使用可能です。
OUT：BOOL

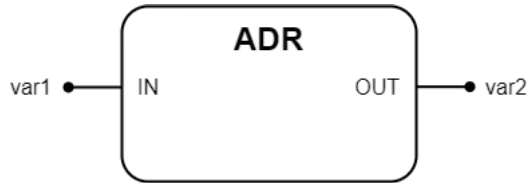
■例

FBD 言語

IL 言語

ST 言語

7.2.29 ADR：アドレス取得



■機能

入力パラメータ IN に接続された変数、ファンクション、ファンクションブロックのインスタンス、プログラムまたはメソッドのメモリアドレスを出力パラメータ OUT に返します。

オンライン変更時の注意点は「8.2 オンライン変更」を参照して下さい。

■入出力パラメータ

入力パラメータ	説　明	備　考
IN(var1)	変数 ファンクション ファンクションブロックのインスタンス プログラム メソッド	

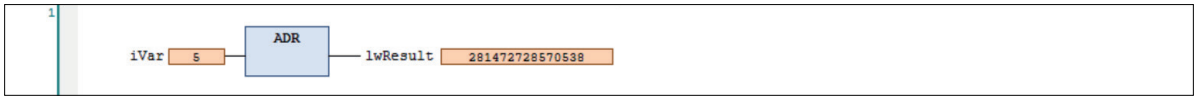
出力パラメータ	説　明	備　考
OUT(var2)	メモリアドレス	ポインタ

■使用可能なデータ型

IN：変数、ファンクション、ファンクションブロックのインスタンス、プログラム、メソッド
OUT：LWORD

■例

FBD 言語



IL 言語

ST 言語

7.2.30 BITADR：ビットアドレス取得

■機能

入力パラメータ IN に接続された BOOL 型変数のビットアドレス（メモリ範囲＋ビットオフセット）を出力パラメータに返します。
オンライン変更時の注意点は「8.2 オンライン変更」を参照して下さい。

■入出力パラメータ

入力パラメータ	説　明	備　考
IN(var1)	BOOL 型変数	

出力パラメータ	説　明	備　考
OUT(var2)	ビットアドレス	メモリ範囲 入力 :16#80000000 出力 :16#C0000000 マーカ :16#40000000

■使用可能なデータ型

IN：BOOL
OUT：DWORD

■例

FBD 言語

IL 言語

ST 言語

ビットアドレスの演算結果は I/O マッピングに依存します。

7.2.31 CAL：ファンクションブロック呼び出し

■機能

ファンクションブロックのインスタンスを呼び出します。

■入出力パラメータ

入力パラメータ	説　明	備　考
－		

出力パラメータ	説　明	備　考
－		

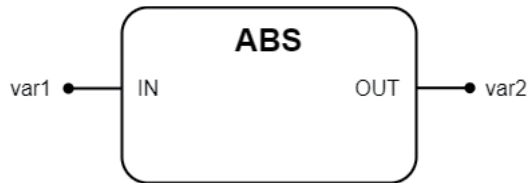
■使用可能なデータ型

－

■例

IL 言語

7.2.32 ABS：絶対値



■機能

入力パラメータ IN に接続された値の絶対値（|IN|）を出力パラメータ OUT に返します。

■入出力パラメータ

入力パラメータ	説　明	備　考
IN（var1）	入力	

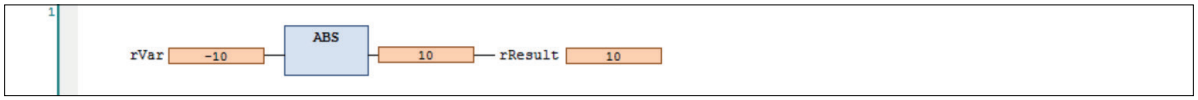
出力パラメータ	説　明	備　考
OUT（var2）	絶対値	IN と同じデータ型であること

■使用可能なデータ型

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

■例

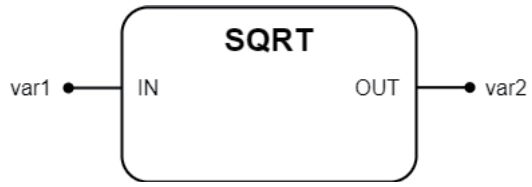
FBD 言語



IL 言語

ST 言語

7.2.33 SQRT：平方根



■機能

入力パラメータ IN に接続された値の平方根（$\sqrt{IN}$）を出力パラメータ OUT に返します。

■入出力パラメータ

入力パラメータ	説　明	備　考
IN(var1)	入力	

出力パラメータ	説　明	備　考
OUT(var2)	平方根	

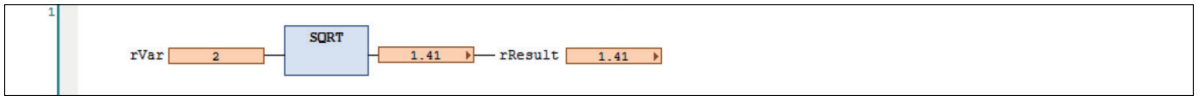
■使用可能なデータ型

IN：BYTE、WORD、DWORD、LWORD、SINT、USINT、INT、UINT、DINT、UDINT、LINT、ULINT、REAL、LREAL

OUT：REAL、LREAL

■例

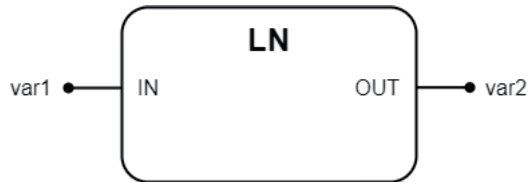
FBD 言語



IL 言語

ST 言語

7.2.34 LN：自然対数



■機能

入力パラメータ IN に接続された値の自然対数（ln(IN)）を出力パラメータ OUT に返します。

■入出力パラメータ

入力パラメータ	説　明	備　考
IN(var1)	入力	

出力パラメータ	説　明	備　考
OUT(var2)	自然対数	

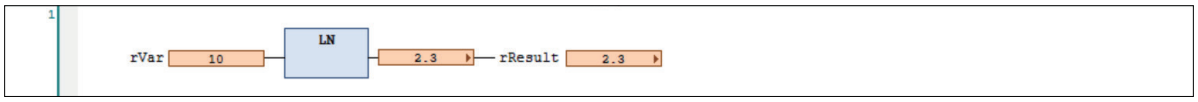
■使用可能なデータ型

IN：BYTE、WORD、DWORD、LWORD、SINT、USINT、INT、UINT、DINT、UDINT、LINT、ULINT、REAL、LREAL

OUT：REAL、LREAL

■例

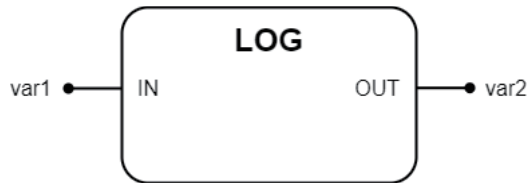
FBD 言語



IL 言語

ST 言語

7.2.35 LOG：常用対数



■機能

入力パラメータ IN に接続された値の常用対数（$\log_{10}$(IN)）を出力パラメータ OUT に返します。

■入出力パラメータ

入力パラメータ	説　明	備　考
IN(var1)	入力	

出力パラメータ	説　明	備　考
OUT(var2)	常用対数	

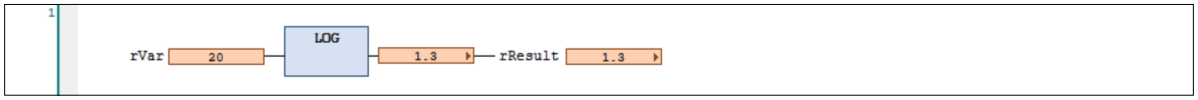
■使用可能なデータ型

IN：BYTE、WORD、DWORD、LWORD、SINT、USINT、INT、UINT、DINT、UDINT、LINT、ULINT、REAL、LREAL

OUT：REAL、LREAL

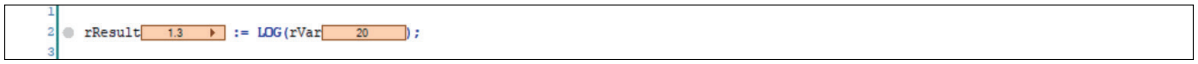
■例

FBD 言語



IL 言語

ST 言語



7.2.36 EXP：e のべき乗

■機能

入力パラメータ IN に接続された値の e のべき乗（e^{IN}）を出力パラメータ OUT に返します。

■入出力パラメータ

入力パラメータ	説　明	備　考
IN(var1)	指数	

出力パラメータ	説　明	備　考
OUT(var2)	e のべき乗	

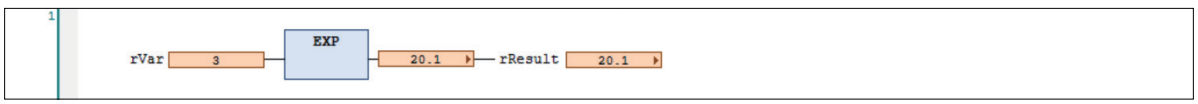
■使用可能なデータ型

IN：BYTE、WORD、DWORD、LWORD、SINT、USINT、INT、UINT、DINT、UDINT、LINT、ULINT、REAL、LREAL

OUT：REAL、LREAL

■例

FBD 言語



IL 言語

ST 言語

7.2.37 EXPT：べき乗

■機能

入力パラメータ IN1、IN2 に接続された値のべき乗（$IN1^{IN2}$）を出力パラメータ OUT に返します。

■入出力パラメータ

入力パラメータ	説　明	備　考
IN1（var1）	底	
IN2（var2）	指数	

出力パラメータ	説　明	備　考
OUT（var3）	自然対数	

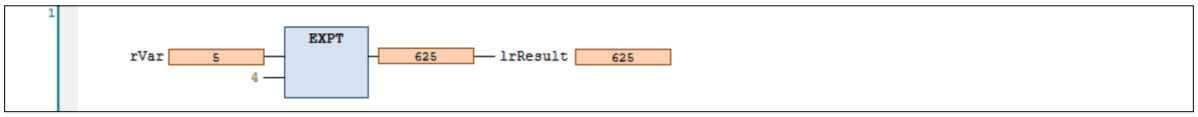
■使用可能なデータ型

IN：BYTE、WORD、DWORD、LWORD、SINT、USINT、INT、UINT、DINT、UDINT、LINT、ULINT、REAL、LREAL

OUT：REAL、LREAL

■例

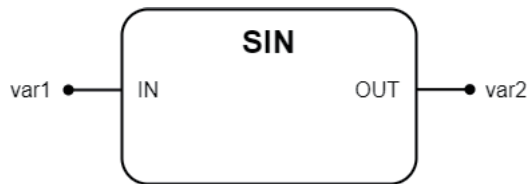
FBD 言語



IL 言語

ST 言語

7.2.38 SIN：正弦（サイン）



■機能

入力パラメータ IN に接続された値の正弦（sin（IN））を出力パラメータ OUT に返します。

■入出力パラメータ

入力パラメータ	説　明	備　考
IN（var1）	角度（ラジアン）	

出力パラメータ	説　明	備　考
OUT（var2）	正弦値	

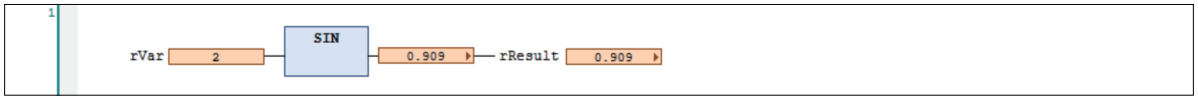
■使用可能なデータ型

IN：BYTE、WORD、DWORD、LWORD、SINT、USINT、INT、UINT、DINT、UDINT、LINT、ULINT、REAL、LREAL

OUT：REAL、LREAL

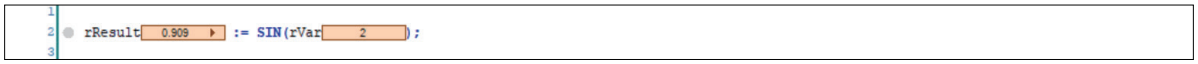
■例

FBD 言語

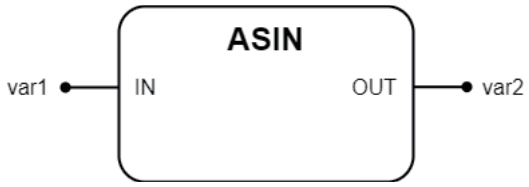


IL 言語

ST 言語



7.2.39 ASIN：逆正弦（アークサイン）



■機能

入力パラメータ IN に接続された値の逆正弦（arcsin(IN)）を出力パラメータ OUT に返します。

■入出力パラメータ

入力パラメータ	説　明	備　考
IN(var1)	正弦値	

出力パラメータ	説　明	備　考
OUT(var2)	角度(ラジアン)	

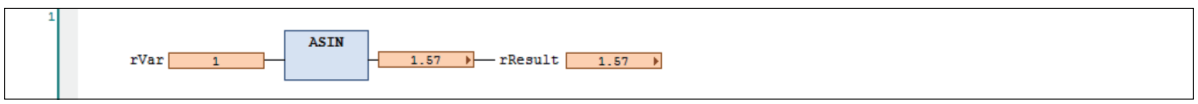
■使用可能なデータ型

IN：BYTE、WORD、DWORD、LWORD、SINT、USINT、INT、UINT、DINT、UDINT、LINT、ULINT、REAL、LREAL

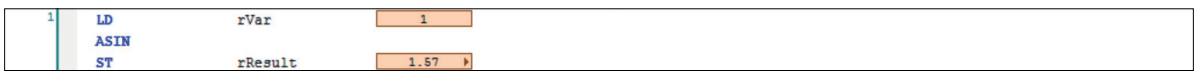
OUT：REAL、LREAL

■例

FBD 言語

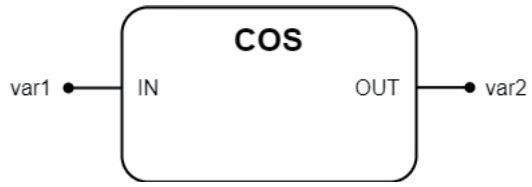


IL 言語



ST 言語

7.2.40 COS：余弦（コサイン）



■機能

入力パラメータ IN に接続された値の余弦（cos(IN)）を出力パラメータ OUT に返します。

■入出力パラメータ

入力パラメータ	説　明	備　考
IN(var1)	角度(ラジアン)	

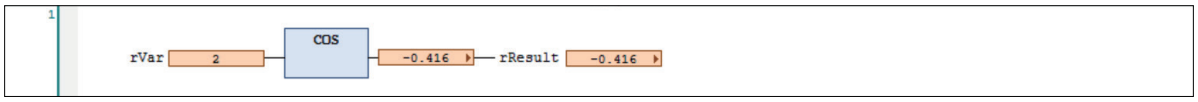
出力パラメータ	説　明	備　考
OUT(var2)	余弦値	

■使用可能なデータ型

IN：BYTE、WORD、DWORD、LWORD、SINT、USINT、INT、UINT、DINT、UDINT、LINT、ULINT、REAL、LREAL
OUT：REAL、LREAL

■例

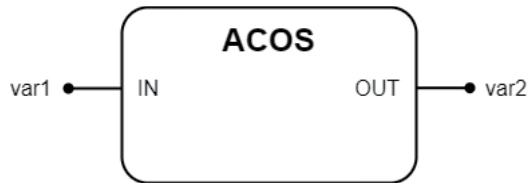
FBD 言語



IL 言語

ST 言語

7.2.41 ACOS：逆余弦（アークコサイン）



■機能

入力パラメータ IN に接続された値の逆余弦（arccos(IN)）を出力パラメータ OUT に返します。

■入出力パラメータ

入力パラメータ	説　明	備　考
IN(var1)	余弦値	

出力パラメータ	説　明	備　考
OUT(var2)	角度(ラジアン)	

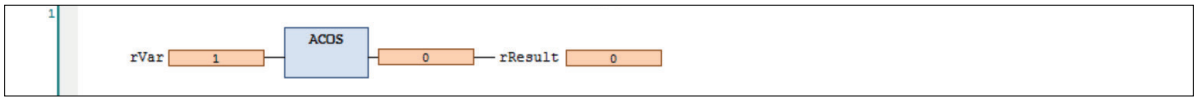
■使用可能なデータ型

IN：BYTE、WORD、DWORD、LWORD、SINT、USINT、INT、UINT、DINT、UDINT、LINT、ULINT、REAL、LREAL

OUT：REAL、LREAL

■例

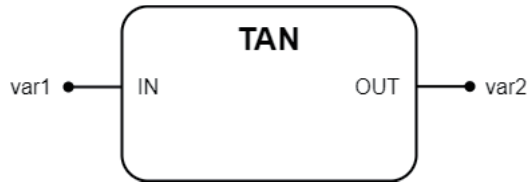
FBD 言語



IL 言語

ST 言語

7.2.42 TAN：正接（タンジェント）



■機能

入力パラメータ IN に接続された値の正接（tan(IN)）を出力パラメータ OUT に返します。

■入出力パラメータ

入力パラメータ	説　明	備　考
IN(var1)	角度(ラジアン)	

出力パラメータ	説　明	備　考
OUT(var2)	正接値	

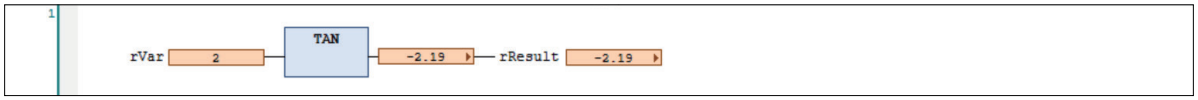
■使用可能なデータ型

IN：BYTE、WORD、DWORD、LWORD、SINT、USINT、INT、UINT、DINT、UDINT、LINT、ULINT、REAL、LREAL

OUT：REAL、LREAL

■例

FBD 言語

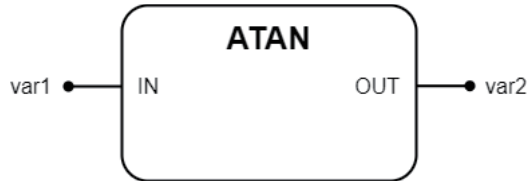


IL 言語

ST 言語

```
1
2  rResult -2.19 := TAN(rVar 2);
3
```

7.2.43 ATAN：逆正接（アークタンジェント）



■機能

入力パラメータ IN に接続された値の逆正接（arctan(IN)）を出力パラメータ OUT に返します。

■入出力パラメータ

入力パラメータ	説　明	備　考
IN(var1)	正接値	

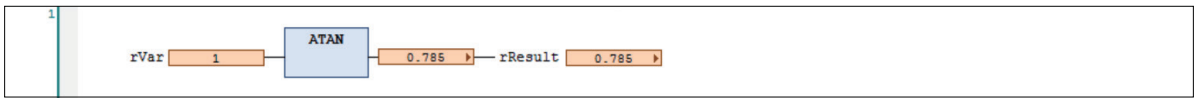
出力パラメータ	説　明	備　考
OUT(var2)	角度(ラジアン)	

■使用可能なデータ型

IN：BYTE、WORD、DWORD、LWORD、SINT、USINT、INT、UINT、DINT、UDINT、LINT、ULINT、REAL、LREAL
OUT：REAL、LREAL

■例

FBD 言語



IL 言語

```
1  LD      rVar      1
   ATAN
   ST      rResult   0.785
```

ST 言語

```
1
2  rResult 0.785 := ATAN(rVar 1);
3
```

7.3 型変換演算子

異なるデータ型の変換には、型変換演算子を使用します。
サイズや符号が異なるデータ型へ変換する際には、情報の一部が失われる可能性があるため注意が必要です。
また、＜データ型＞ _TO_STRING や＜データ型＞ _TO_WSTRING では、値は左詰めの文字列として格納され、文字数を超える場合は末尾が切り捨てられます。変換後の文字列が収まるように、十分な長さの戻り値変数を宣言して下さい。

構文

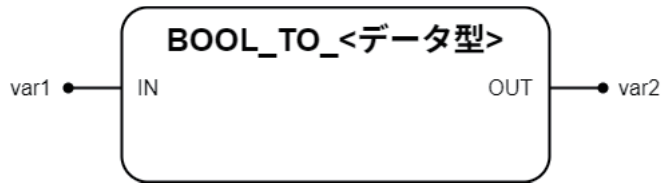
＜変換前の型＞_TO_＜変換後の型＞

例

INT_TO_STRING：INT型をSTRING型に変換 STRING_TO_INT：STRING型をINT型に変換

分類	命令	機能
BOOL 型	BOOL_TO_ ＜データ型＞	BOOL 型の値を指定されたデータ型に変換
符号付き整数型	SINT_TO_ ＜データ型＞ INT_TO_ ＜データ型＞ DINT_TO_ ＜データ型＞ LINT_TO_ ＜データ型＞	符号付き整数型の値を指定されたデータ型に変換
符号なし整数型	USINT_TO_ ＜データ型＞ UINT_TO_ ＜データ型＞ UDINT_TO_ ＜データ型＞ ULINT_TO_ ＜データ型＞	符号なし整数型の値を指定されたデータ型に変換
ビット列型	BYTE_TO_ ＜データ型＞ WORD_TO_ ＜データ型＞ DWORD_TO_ ＜データ型＞ LWORD_TO_ ＜データ型＞	ビット列型の値を指定されたデータ型に変換
実数型	REAL_TO_ ＜データ型＞ LREAL_TO_ ＜データ型＞	実数型の値を指定されたデータ型に変換
時間型	TIME_TO_ ＜データ型＞ LTIME_TO_ ＜データ型＞	時間型の値を指定されたデータ型に変換
日付時刻型	DATE_TO_ ＜データ型＞ LDATE_TO_ ＜データ型＞ DT_TO_ ＜データ型＞ LDT_TO_ ＜データ型＞ TOD_TO_ ＜データ型＞ LTOD_TO_ ＜データ型＞	日付時刻型の値を指定されたデータ型に変換
文字列型	STRING_TO_ ＜データ型＞ WSTRING_TO_ ＜データ型＞	文字列型の値を指定されたデータ型に変換
小数点以下切り捨て	TRUNC TRUNC_INT	REAL 型の値の小数点以下を切り捨てて、DINT 型または INT 型に変換

7.3.1 BOOL_TO_＜データ型＞：BOOL 型



■機能

入力パラメータ IN に接続された BOOL 型の値を、指定されたデータ型に変換して出力パラメータ OUT に返します。
数値型への変換では、IN が TRUE のとき 1、FALSE のとき 0 を OUT に返します。文字列型への変換では、IN が TRUE のとき’TRUE’、FALSE のとき’FALSE’を OUT に返します。

■入出力パラメータ

入力パラメータ	説　明	備　考
IN（var1）	入力	BOOL 型

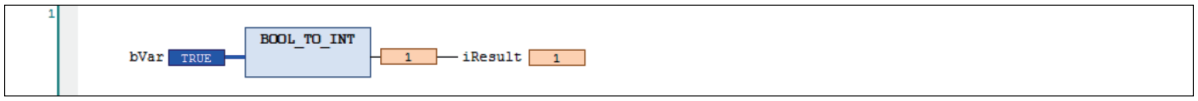
出力パラメータ	説　明	備　考
OUT（var2）	出力	

■使用可能なデータ型

IN：BOOL
OUT：BIT、BYTE、WORD、DWORD、LWORD、SINT、USINT、INT、UINT、DINT、UDINT、LINT、ULINT、TIME、LTIME、DATE、LDATE、DATE_AND_TIME（DT）、LDATE_AND_TIME（LDT）、TIME_OF_DAY（TOD）、LTIME_OF_DAY（LTOD）、REAL、LREAL、STRING、WSTRING

■例

FBD 言語

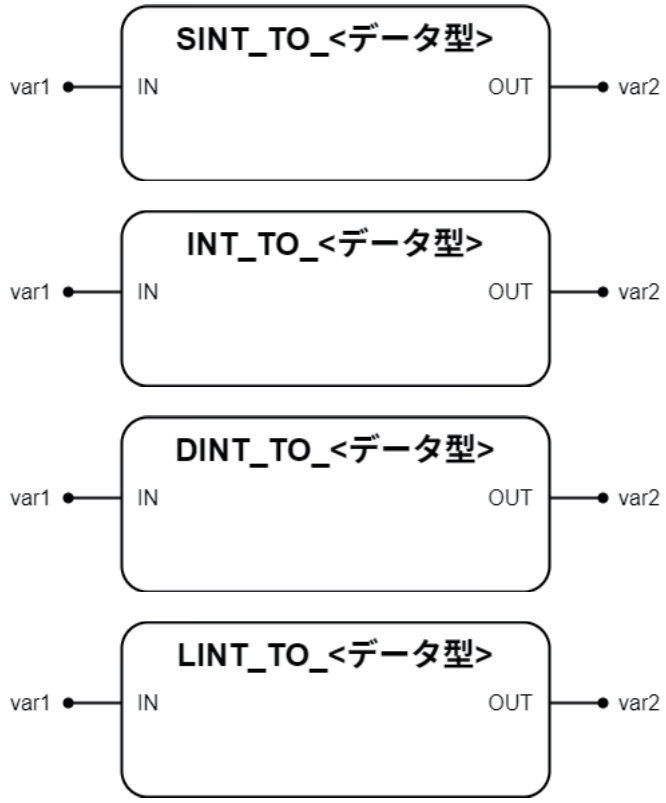


IL 言語

ST 言語

```
1
2   iResult_T 1 := BOOL_TO_INT(TRUE);
3   iResult_F 0 := BOOL_TO_INT(FALSE);
4
5   uiResult_T 1 := BOOL_TO_UINT(TRUE);
6   uiResult_F 0 := BOOL_TO_UINT(FALSE);
7
8   wResult_T 1 := BOOL_TO_WORD(TRUE);
9   wResult_F 0 := BOOL_TO_WORD(FALSE);
10
11  rResult_T 1 := BOOL_TO_REAL(TRUE);
12  rResult_F 0 := BOOL_TO_REAL(FALSE);
13
14  timeResult_T T#1ms := BOOL_TO_TIME(TRUE);
15  timeResult_F T#0ms := BOOL_TO_TIME(FALSE);
16
17  dateResult_T D#1970-1-1 := BOOL_TO_DATE(TRUE);
18  dateResult_F D#1970-1-1 := BOOL_TO_DATE(FALSE);
19
20  strResult_T 'TRUE' := BOOL_TO_STRING(TRUE);
21  strResult_F 'FALSE' := BOOL_TO_STRING(FALSE);
```

7.3.2 SINT_TO_＜データ型＞ / INT_TO_＜データ型＞ / DINT_TO_＜データ型＞ / LINT_TO_＜データ型＞：符号付き整数型



■機能

入力パラメータ IN に接続された符号付き整数型の値を、指定されたデータ型に変換して出力パラメータ OUT に返します。

IN の値が OUT のデータ型で表現可能な範囲を超える場合、上位バイトが切り捨てられた値（下位バイトのみ）を OUT に返します。

■入出力パラメータ

入力パラメータ	説　明	備　考
IN（var1）	入力	符号付き整数型

出力パラメータ	説　明	備　考
OUT（var2）	出力	

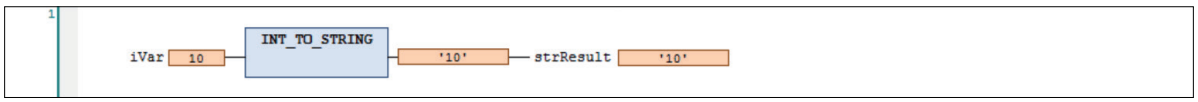
■使用可能なデータ型

IN：SINT、INT、DINT、LINT

OUT：BOOL、BIT、BYTE、WORD、DWORD、LWORD、SINT、USINT、INT、UINT、DINT、UDINT、LINT、ULINT、TIME、LTIME、DATE、LDATE、DATE_AND_TIME（DT）、LDATE_AND_TIME（LDT）、TIME_OF_DAY（TOD）、LTIME_OF_DAY（LTOD）、REAL、LREAL、STRING、WSTRING

■例

FBD 言語

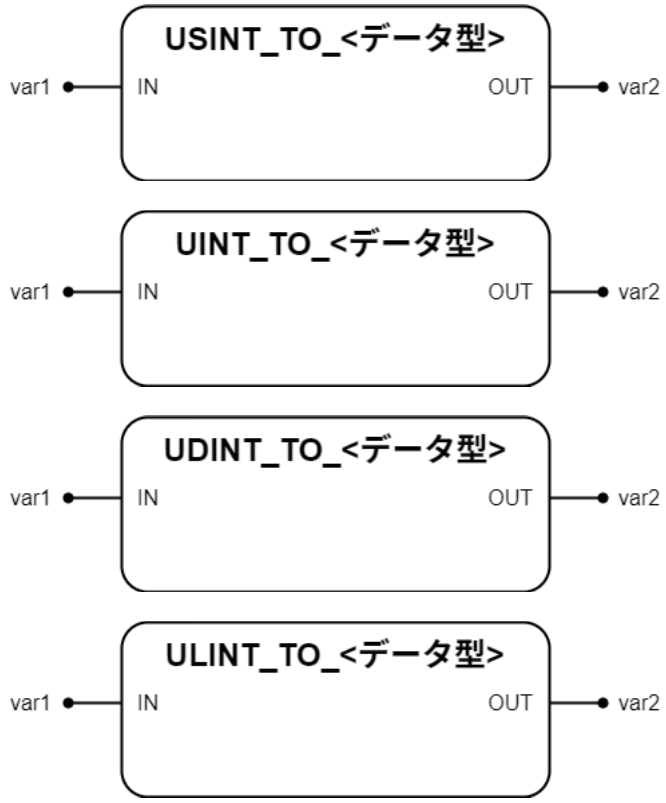


IL 言語

ST 言語

```
bResult := INT_TO_BOOL(150);
uiResult := INT_TO_UINT(150);
wResult := INT_TO_WORD(150);
rResult := INT_TO_REAL(150);
timeResult := INT_TO_TIME(150);
dateResult := INT_TO_DATE(150);
strResult := INT_TO_STRING(150);
```

7.3.3 USINT_TO_＜データ型＞ / UINT_TO_＜データ型＞ / UDINT_TO_＜データ型＞ / ULINT_TO_＜データ型＞：符号なし整数型



■機能

入力パラメータ IN に接続された符号なし整数型の値を、指定されたデータ型に変換して出力パラメータ OUT に返します。IN の値が OUT のデータ型で表現可能な範囲を超える場合、上位ビットが切り捨てられた値（下位ビットのみ）を OUT に返します。OUT の最上位ビット（MSB）が 1 の場合、OUT が符号付き整数型であれば負の値として扱われます。

■入出力パラメータ

入力パラメータ	説　明	備　考
IN（var1）	入力	符号なし整数型

出力パラメータ	説　明	備　考
OUT（var2）	出力	

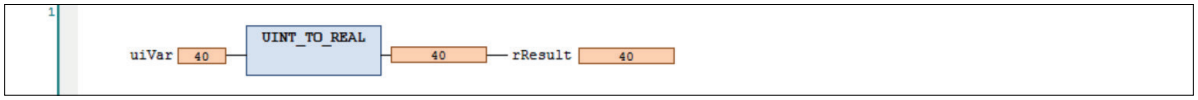
■使用可能なデータ型

IN：USINT、UINT、UDINT、ULINT

OUT：BOOL、BIT、BYTE、WORD、DWORD、LWORD、SINT、USINT、INT、UINT、DINT、UDINT、LINT、ULINT、TIME、LTIME、DATE、LDATE、DATE_AND_TIME（DT）、LDATE_AND_TIME（LDT）、TIME_OF_DAY（TOD）、LTIME_OF_DAY（LTOD）、REAL、LREAL、STRING、WSTRING

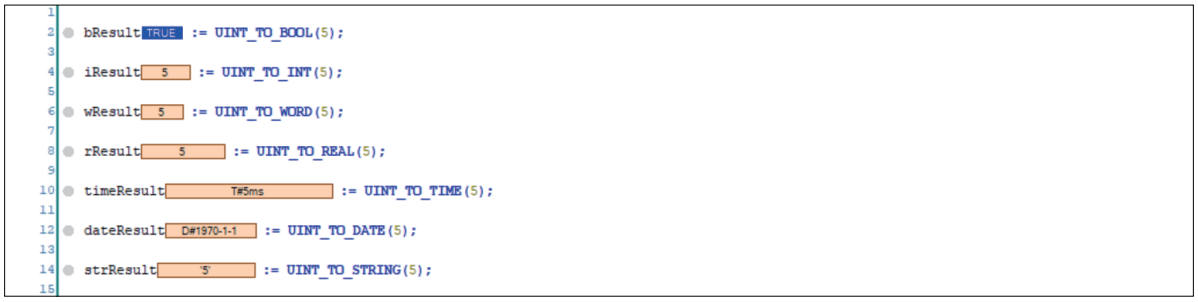
■例

FBD 言語

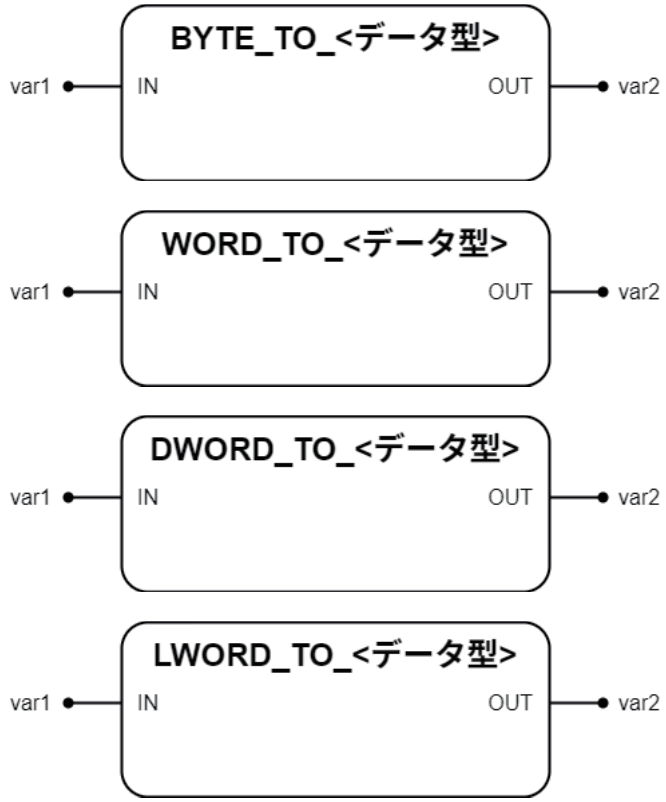


IL 言語

ST 言語



7.3.4 BYTE_TO_＜データ型＞ / WORD_TO_＜データ型＞ / DWORD_TO_＜データ型＞ / LWORD_TO_＜データ型＞：ビット列型



■機能

入力パラメータ IN に接続されたビット列型の値を、指定されたデータ型に変換して出力パラメータ OUT に返します。IN の値が OUT のデータ型で表現可能な範囲を超える場合、上位ビットが切り捨てられた値（下位ビットのみ）を OUT に返します。OUT の最上位ビット（MSB）が 1 の場合、OUT が符号付き整数型であれば負の値として扱われます。

■入出力パラメータ

入力パラメータ	説　明	備　考
IN(var1)	入力	ビット列型

出力パラメータ	説　明	備　考
OUT(var2)	出力	

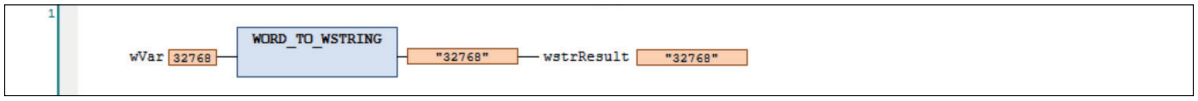
■使用可能なデータ型

IN：BYTE、WORD、DWORD、LWORD

OUT：BOOL、BIT、BYTE、WORD、DWORD、LWORD、SINT、USINT、INT、UINT、DINT、UDINT、LINT、ULINT、TIME、LTIME、DATE、LDATE、DATE_AND_TIME（DT）、LDATE_AND_TIME（LDT）、TIME_OF_DAY（TOD）、LTIME_OF_DAY（LTOD）、REAL、LREAL、STRING、WSTRING

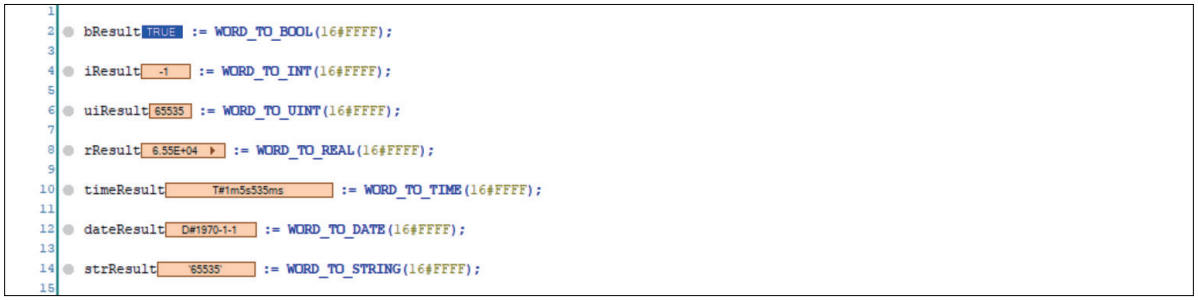
■例

FBD 言語



IL 言語

ST 言語



7.3.5 REAL_TO_＜データ型＞ / LREAL_TO_＜データ型＞：実数型

■機能

入力パラメータ IN に接続された実数型の値を、指定されたデータ型に変換して出力パラメータ OUT に返します。
REAL 型は UDINT 型に、LREAL 型は ULINT 型に変換された後、OUT のデータ型に変換されます。
IN の値が OUT のデータ型で表現可能な範囲を超える場合、上位ビットが切り捨てられた値（下位ビットのみ）を OUT に返します。OUT が整数型またはビット列型の場合、IN の値の小数点以下を四捨五入して OUT に返します。
OUT が文字列型の場合、仮数部は小数点以下 6 桁まで表示されます。

■入出力パラメータ

入力パラメータ	説　明	備　考
IN（var1）	入力	実数型

出力パラメータ	説　明	備　考
OUT（var2）	出力	

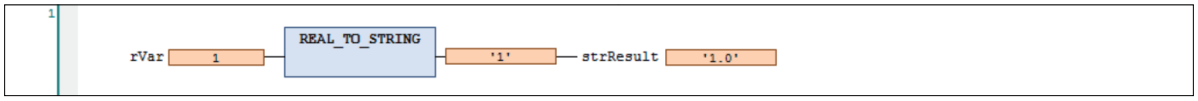
■使用可能なデータ型

IN：REAL、LREAL

OUT：BOOL、BIT、BYTE、WORD、DWORD、LWORD、SINT、USINT、INT、UINT、DINT、UDINT、LINT、ULINT、TIME、LTIME、DATE、LDATE、DATE_AND_TIME（DT）、LDATE_AND_TIME（LDT）、TIME_OF_DAY（TOD）、LTIME_OF_DAY（LTOD）、REAL、LREAL、STRING、WSTRING

■例

FBD 言語



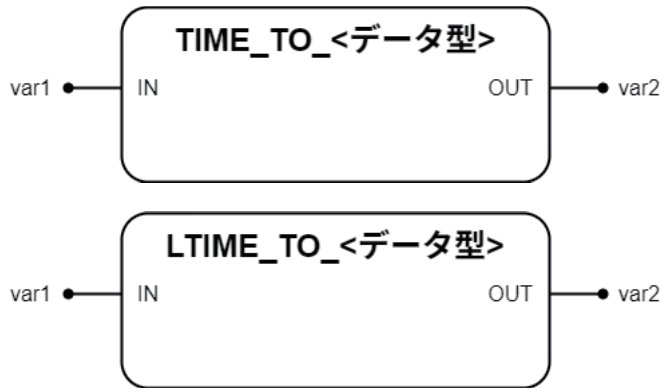
IL 言語

```
1   LD              rVar        1
    REAL_TO_STRING
    ST              strResult   '1.0'
```

ST 言語

```
1
2   bResult TRUE := REAL_TO_BOOL(1.234);
3
4   iResult 1 := REAL_TO_INT(1.234);
5
6   uiResult 1 := REAL_TO_UINT(1.234);
7
8   wResult 1 := REAL_TO_WORD(1.234);
9
10  timeResult T#1ms := REAL_TO_TIME(1.234);
11
12  dateResult D#1970-1-1 := REAL_TO_DATE(1.234);
13
14  strResult '1.234' := REAL_TO_STRING(1.234);
15
```

7.3.6 TIME_TO_＜データ型＞ / LTIME_TO_＜データ型＞：時間型



■機能

入力パラメータ IN に接続された時間型の値を、指定されたデータ型に変換して出力パラメータ OUT に返します。
IN の値が OUT のデータ型で表現可能な範囲を超える場合、上位ビットが切り捨てられた値（下位ビットのみ）を OUT に返します。

■入出力パラメータ

入力パラメータ	説　明	備　考
IN（var1）	入力	時間型

出力パラメータ	説　明	備　考
OUT（var2）	出力	

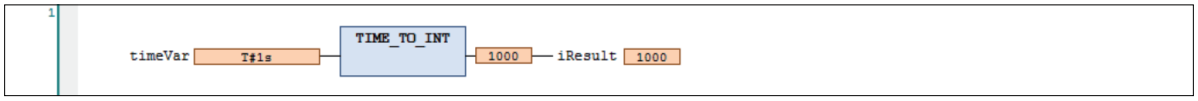
■使用可能なデータ型

IN：TIME、LTIME

OUT：BOOL、BIT、BYTE、WORD、DWORD、LWORD、SINT、USINT、INT、UINT、DINT、UDINT、LINT、ULINT、TIME、LTIME、DATE、LDATE、DATE_AND_TIME（DT）、LDATE_AND_TIME（LDT）、TIME_OF_DAY（TOD）、LTIME_OF_DAY（LTOD）、REAL、LREAL、STRING、WSTRING

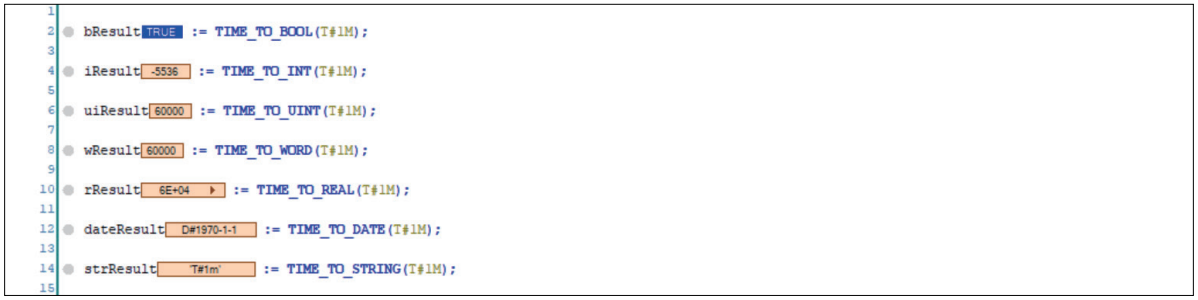
■例

FBD 言語

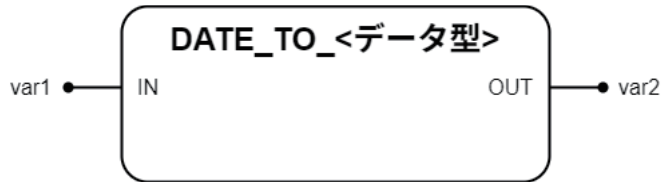


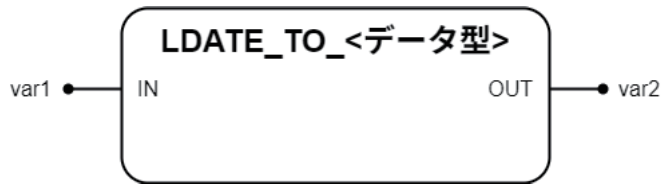
IL 言語

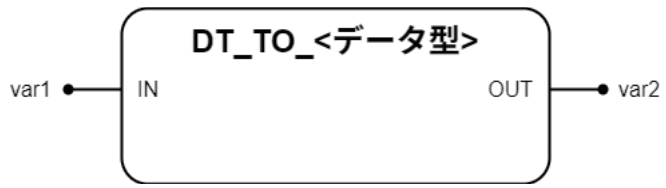
ST 言語

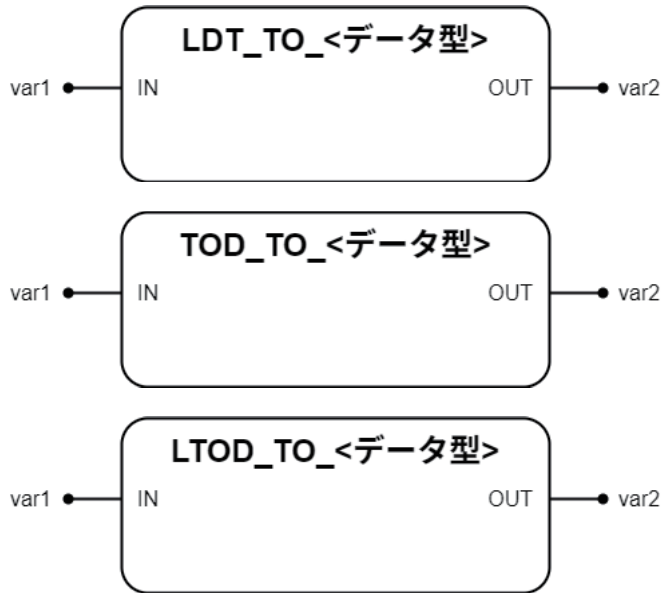


7.3.7 DATE_TO_＜データ型＞/ LDATE_TO_＜データ型＞/ DT_TO_＜データ型＞/ LDT_TO_＜データ型＞/ TOD_TO_＜データ型＞/ LTOD_TO_＜データ型＞：日付時刻型









■機能

入力パラメータ IN に接続された日付時刻型の値を、指定されたデータ型に変換して出力パラメータ OUT に返します。
DATE、DATE_AND_TIME（DT）、TIME_OF_DAY（TOD）型は DWORD 型に、LDATE、LDATE_AND_TIME（LDT）、LTIME_OF_DAY（LTOD）型は LWORD 型に変換された後、OUT のデータ型に変換されます。
IN の値が OUT のデータ型で表現可能な範囲を超える場合、上位ビットが切り捨てられた値（下位ビットのみ）を OUT に返します。

■入出力パラメータ

入力パラメータ	説　明	備　考
IN（var1）	入力	日付時刻型

出力パラメータ	説　明	備　考
OUT（var2）	出力	

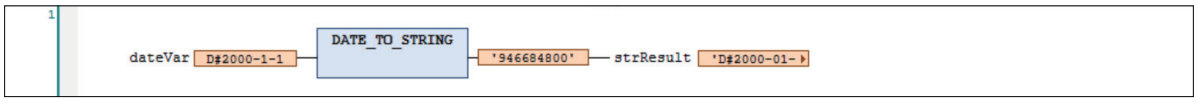
■使用可能なデータ型

IN：DATE、LDATE、DATE_AND_TIME（DT）、LDATE_AND_TIME（LDT）、TIME_OF_DAY（TOD）、LTIME_OF_DAY（LTOD）

OUT：BOOL、BIT、BYTE、WORD、DWORD、LWORD、SINT、USINT、INT、UINT、DINT、UDINT、LINT、ULINT、TIME、LTIME、DATE、LDATE、DATE_AND_TIME（DT）、LDATE_AND_TIME（LDT）、TIME_OF_DAY（TOD）、LTIME_OF_DAY（LTOD）、REAL、LREAL、STRING、WSTRING

■例

FBD 言語

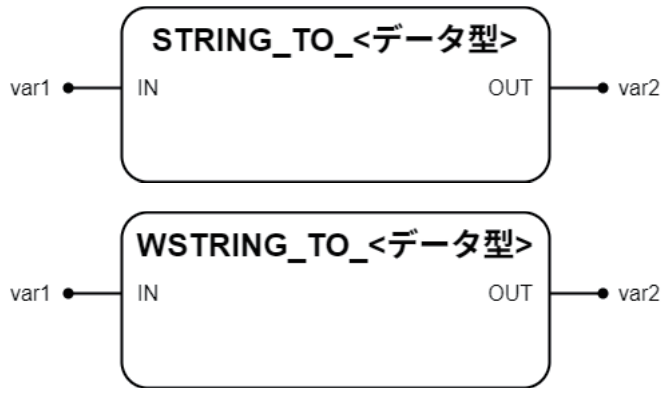


IL 言語

ST 言語

```
bResult TRUE := DATE_TO_BOOL(D#1970-12-31);
iResult -7680 := DATE_TO_INT(D#1970-12-31);
uiResult 57856 := DATE_TO_UINT(D#1970-12-31);
wResult 57856 := DATE_TO_WORD(D#1970-12-31);
rResult 3.14E+07 := DATE_TO_REAL(D#1970-12-31);
timeResult T#16d40m28s928ms := DATE_TO_TIME(D#1970-12-31);
strResult D#1970-12- := DATE_TO_STRING(D#1970-12-31);
```

7.3.8 STRING_TO_＜データ型＞ / WSTRING_TO_＜データ型＞：文字列型



■機能

入力パラメータ IN に接続された文字列型の値を、指定されたデータ型に変換して出力パラメータ OUT に返します。
BOOL 型への変換では、IN が’TRUE’または’true’のときは TRUE、それ以外の文字列（’True’）のときは FALSE を OUT に返します。

■入出力パラメータ

入力パラメータ	説　明	備　考
IN（var1）	入力	文字列型

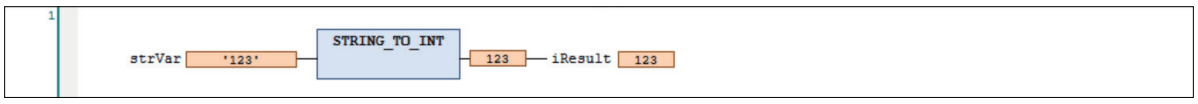
出力パラメータ	説　明	備　考
OUT（var2）	出力	

■使用可能なデータ型

IN：STRING、WSTRING
OUT：BOOL、BIT、BYTE、WORD、DWORD、LWORD、SINT、USINT、INT、UINT、DINT、UDINT、LINT、ULINT、TIME、LTIME、DATE、LDATE、DATE_AND_TIME（DT）、LDATE_AND_TIME（LDT）、TIME_OF_DAY（TOD）、LTIME_OF_DAY（LTOD）、REAL、LREAL、STRING、WSTRING

■例

FBD 言語



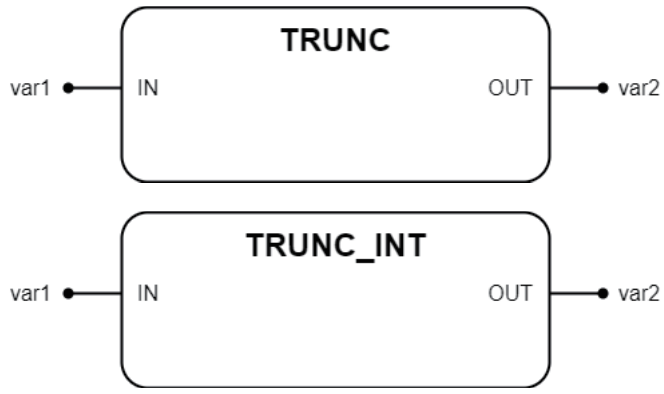
IL 言語

```
LD          strVar      '123'
STRING_TO_INT
ST          iResult     123
```

ST 言語

```
bResult := STRING_TO_BOOL('TRUE');

iResult := STRING_TO_INT('-1');

uiResult := STRING_TO_UINT('123');

wResult := STRING_TO_WORD('16#FFFF');

rResult := STRING_TO_REAL('1.234');

timeResult := STRING_TO_TIME('T#1h');

dateResult := STRING_TO_DATE('D#1990-7-7');
```

7.3.9 TRUNC / TRUNC_INT



■機能

入力パラメータ IN に接続された REAL 型の値の小数点以下を切り捨てて、出力パラメータ OUT に返します。

■入出力パラメータ

入力パラメータ	説　明	備　考
IN(var1)	入力	REAL 型

出力パラメータ	説　明	備　考
OUT(var2)	出力	TRUNC：DINT 型 TRUNC_INT：INT 型

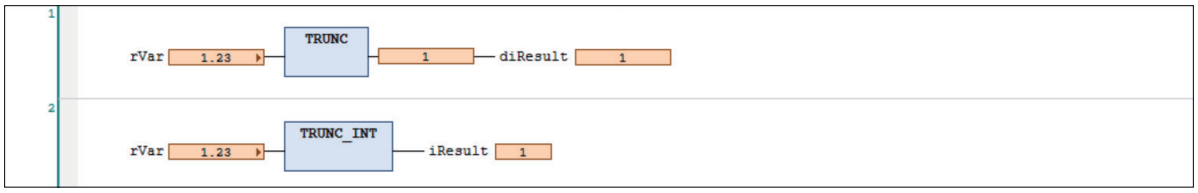
■使用可能なデータ型

IN：REAL

OUT：DINT、INT

■例

FBD 言語



IL 言語

```
LD      rVar        1.23
TRUNC
ST      diResult    1
LD      rVar        1.23
TRUNC_INT
ST      iResult     1
```

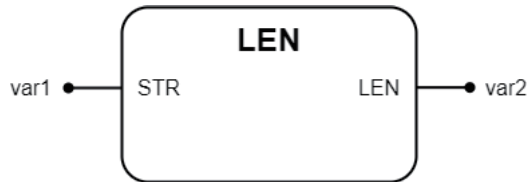
ST 言語

```
1
2  diResult 1 := TRUNC(rVar 1.23 );
3
4  iResult 1 := TRUNC_INT(rVar 1.23 );
5
```

7.4 文字列操作ファンクション

分　類	命　令	機　能
文字列操作	LEN	長さ(文字数)取得
	LEFT	左端から抽出
	RIGHT	右端から抽出
	MID	中間から抽出
	CONCAT	連結
	INSERT	挿入
	DELETE	削除
	REPLACE	置換
	FIND	検索(位置取得)

7.4.1 LEN：長さ（文字数）取得



■機能

入力パラメータ STR に接続された文字列の長さ（文字数）を出力パラメータ LEN に返します。

■入出力パラメータ

入力パラメータ	説　明	備　考
STR(var1)	文字列	最大 255 文字

出力パラメータ	説　明	備　考
LEN(var2)	文字列の長さ(文字数)	

■使用可能なデータ型

STR：STRING

LEN：INT

■例

FBD 言語

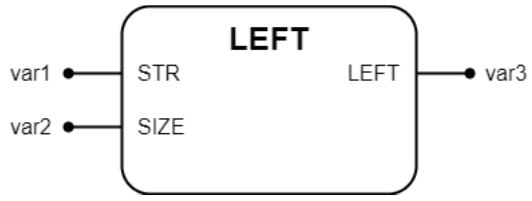
IL 言語

```
1  LD    strVar    'Function'
   LEN
   ST    iLEN      8
```

ST 言語

```
1
2  iLEN 8 := LEN(strVar 'Function' );
3
```

7.4.2 LEFT：左端から抽出



■機能

入力パラメータ STR に接続された文字列の左端から、入力パラメータ SIZE に接続された文字数だけ抽出して、出力パラメータ LEFT に返します。

■入出力パラメータ

入力パラメータ	説　明	備　考
STR(var1)	文字列	最大 255 文字
SIZE(var2)	文字数	

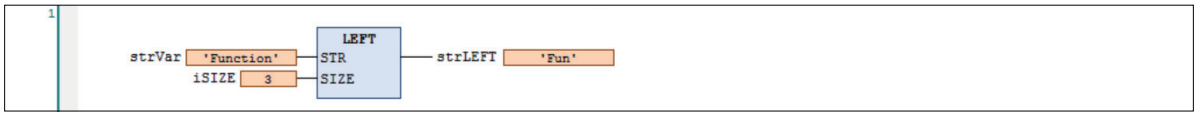
出力パラメータ	説　明	備　考
LEFT(var3)	左端から抽出	最大 255 文字

■使用可能なデータ型

STR、LEFT：STRING
SIZE：INT

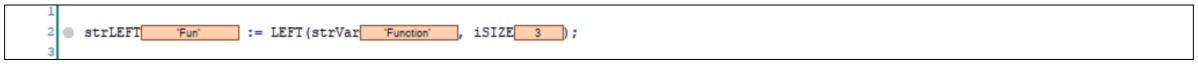
■例

FBD 言語

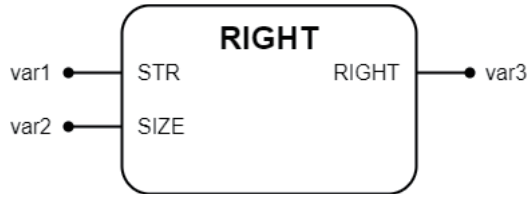


IL 言語

ST 言語



7.4.3 RIGHT：右端から抽出



■機能

入力パラメータ STR に接続された文字列の右端から、入力パラメータ SIZE に接続された文字数だけ抽出して、出力パラメータ RIGHT に返します。

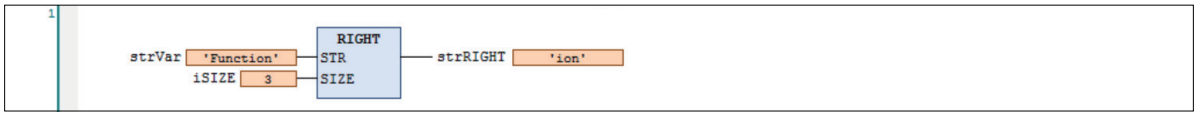
■入出力パラメータ

入力パラメータ	説　明	備　考
STR(var1)	文字列	最大 255 文字
SIZE(var2)	文字数	

出力パラメータ	説　明	備　考
RIGHT(var3)	右端から抽出	最大 255 文字

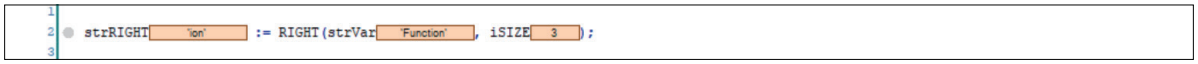
■使用可能なデータ型
STR、RIGHT：STRING
SIZE：INT

■例
FBD 言語

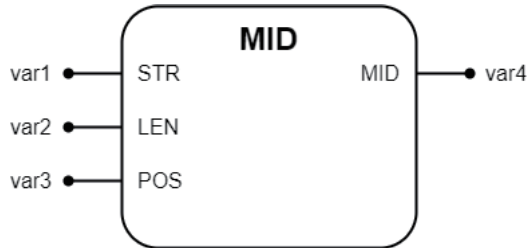


IL 言語

ST 言語



7.4.4 MID：中間から抽出



■機能
入力パラメータ STR に接続された文字列の入力パラメータ POS に接続された位置から、入力パラメータ LEN に接続された文字数だけ抽出して、出力パラメータ MID に返します。

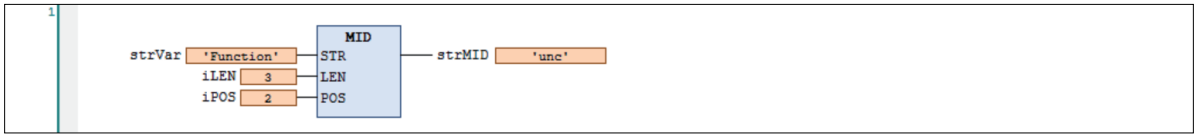
■入出力パラメータ

入力パラメータ	説　明	備　考
STR(var1)	文字列	最大 255 文字
LEN(var2)	文字数	
POS(var3)	位置	先頭文字位置は 1

出力パラメータ	説　明	備　考
MID(var4)	中間から抽出	最大 255 文字

■使用可能なデータ型
STR、MID：STRING
LEN、POS：INT

■例
FBD 言語



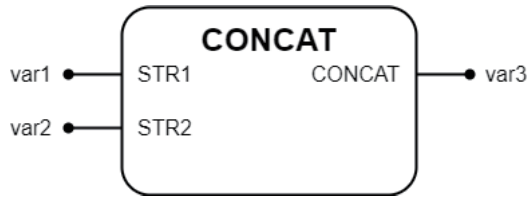
IL 言語

```
LD      strVar    'Function'
MID     iLEN      3
        iPOS      2
ST      strMID    'unc'
```

ST 言語

```
strMID 'unc' := MID(strVar 'Function', iLEN 3, iPOS 2);
```

7.4.5 CONCAT：連結



■機能

入力パラメータ STR1、STR2 に接続された文字列を連結して出力パラメータ CONCAT に返します。

■入出力パラメータ

入力パラメータ	説　明	備　考
STR1(var1)	文字列 1	最大 255 文字
STR2(var2)	文字列 2	最大 255 文字

出力パラメータ	説　明	備　考
CONCAT(var3)	連結	最大 255 文字

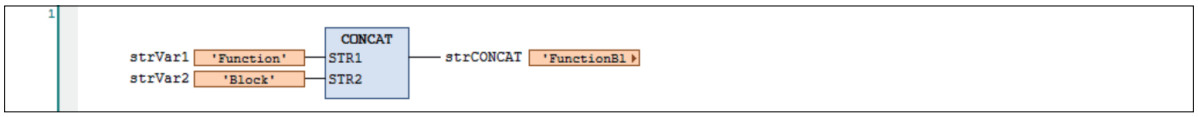
連結後の文字列が 255 文字を超える場合、CONCAT には先頭から 255 文字までが返されます。

■使用可能なデータ型

STR1、STR2、CONCAT：STRING

■例

FBD 言語



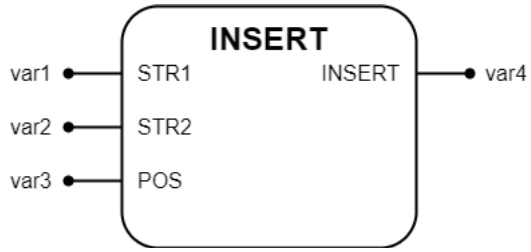
IL 言語

```
LD      strVar1   'Function'
CONCAT  strVar2   'Block'
ST      strCONCAT 'FunctionBl'
```

ST 言語

```
strCONCAT 'FunctionBl' := CONCAT(strVar1 'Function', strVar2 'Block');
```

7.4.6 INSERT：挿入



■機能

入力パラメータ STR1 に接続された文字列の入力パラメータ POS に接続された位置に、入力パラメータ STR2 に接続された文字列を挿入して、出力パラメータ INSERT に返します。

■入出力パラメータ

入力パラメータ	説　明	備　考
STR1(var1)	文字列 1	最大 255 文字
STR2(var2)	文字列 2	最大 255 文字
POS(var3)	位置	0: 先頭文字の前に挿入 1: 先頭文字の後に挿入

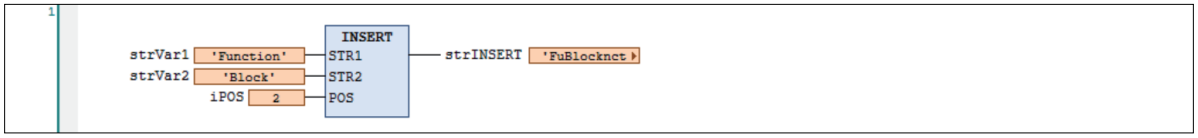
出力パラメータ	説　明	備　考
INSERT(var4)	挿入	最大 255 文字

■使用可能なデータ型

STR1、STR2、INSERT：STRING
POS：INT

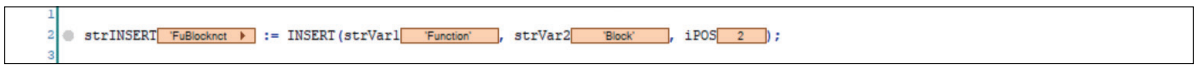
■例

FBD 言語

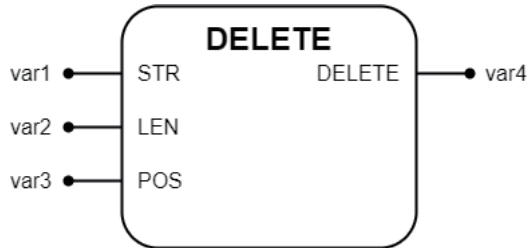


IL 言語

ST 言語



7.4.7 DELETE：削除



■機能

入力パラメータ STR に接続された文字列の入力パラメータ POS に接続された位置から、入力パラメータ LEN に接続された文字数を削除して、出力パラメータ DELETE に返します。

■入出力パラメータ

入力パラメータ	説　明	備　考
STR(var1)	文字列	最大 255 文字
LEN(var2)	文字数	
POS(var3)	位置	先頭文字位置は 1

出力パラメータ	説　明	備　考
DELETE(var4)	削除	最大 255 文字

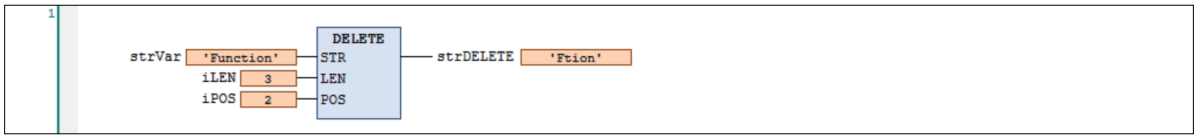
■使用可能なデータ型

STR、DELETE：STRING

LEN、POS：INT

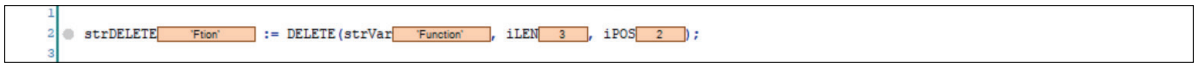
■例

FBD 言語

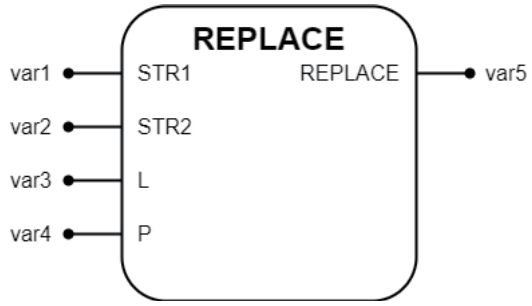


IL 言語

ST 言語



7.4.8 REPLACE：置換



■機能

入力パラメータ STR1 に接続された文字列の入力パラメータ P に接続された位置から、入力パラメータ L に接続された文字数分を、入力パラメータ STR2 に接続された文字列に置き換えて、出力パラメータ REPLACE に返します。

■入出力パラメータ

入力パラメータ	説　明	備　考
STR1(var1)	文字列 1	最大 255 文字
STR2(var2)	文字列 2	最大 255 文字
L(var3)	文字数	
P(var4)	位置	先頭文字位置は 1

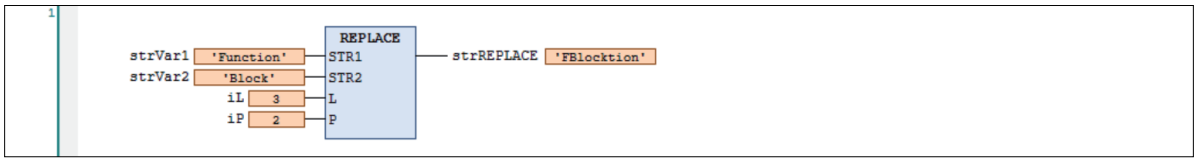
出力パラメータ	説　明	備　考
REPLACE(var5)	置換	最大 255 文字

■使用可能なデータ型

STR1、STR2、REPLACE：STRING
L、P：INT

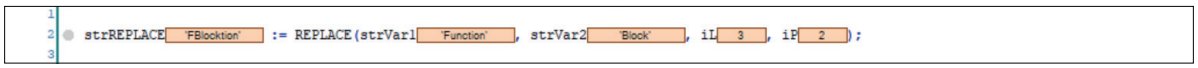
■例

FBD 言語

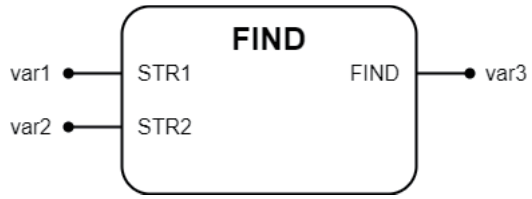


IL 言語

ST 言語



7.4.9 FIND：検索（位置取得）



■機能

入力パラメータ STR1 に接続された文字列内で入力パラメータ STR2 に接続された文字列を検索して、最初に出現する位置を出力パラメータ FIND に返します。

■入出力パラメータ

入力パラメータ	説　明	備　考
STR1(var1)	文字列 1	最大 255 文字
STR2(var2)	文字列 2	最大 255 文字

出力パラメータ	説　明	備　考
FIND(var3)	検索(位置)	先頭文字位置は 1

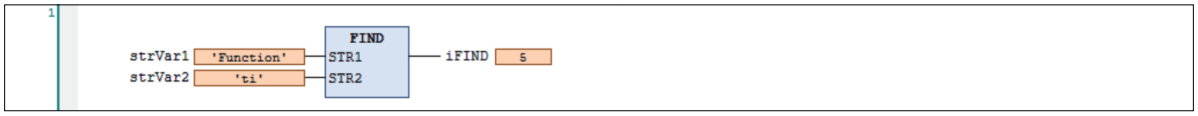
見つからなかった場合、FIND には 0 が返されます。

■使用可能なデータ型

STR1、STR2：STRING
FIND：INT

■例

FBD 言語



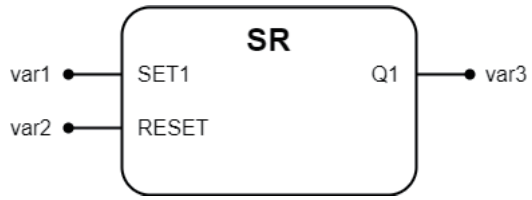
IL 言語

ST 言語

7.5 標準ファンクションブロック

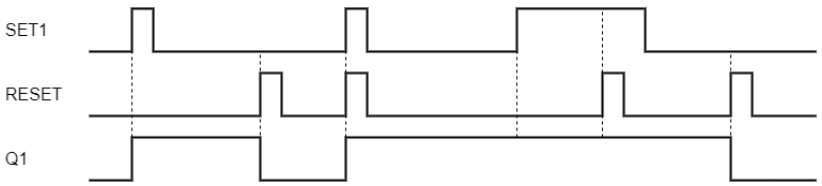
分　類	命　令	機　能
フリップフロップ	SR	セット優先フリップフロップ
	RS	リセット優先フリップフロップ
カウンタ	CTU	アップカウンタ
	CTD	ダウンカウンタ
	CTUD	アップダウンカウンタ
その他	RTC	リアルタイムクロック
タイマ	TON	オンディレイタイマ
	TOF	オフディレイタイマ
	TP	パルスタイマ
エッジ検出	R_TRIG	立ち上がりエッジ検出
	F_TRIG	立ち下がりエッジ検出

7.5.1 SR：セット優先フリップフロップ



■機能

セット優先のラッチ動作（状態の保持）を行います。入力パラメータ SET1 に接続された値が TRUE の場合は出力パラメータ Q1 を TRUE にし、入力パラメータ RESET に接続された値が TRUE の場合は Q1 を FALSE にします。SET1 と RESET が同時に TRUE の場合は SET1 が優先され、Q1 は TRUE となります。



■入出力パラメータ

入力パラメータ	説　明	備　考
SET1(var1)	セット入力	
RESET(var2)	リセット入力	

出力パラメータ	説　明	備　考
Q1(var3)	出力	初期値 FALSE

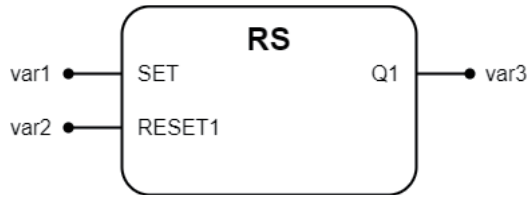
■使用可能なデータ型
SET1、RESET、Q1：BOOL

■例
FBD 言語

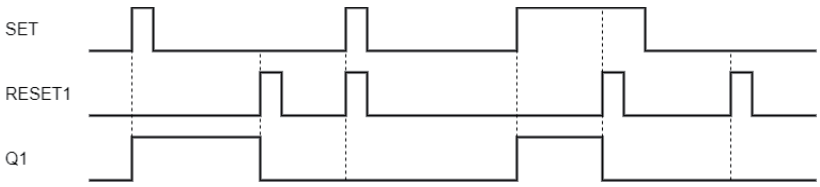
IL 言語

ST 言語

7.5.2 RS：リセット優先フリップフロップ



■機能
リセット優先のラッチ動作（状態の保持）を行います。入力パラメータ SET に接続された値が TRUE の場合は出力パラメータ Q1 を TRUE にし、入力パラメータ RESET1 に接続された値が TRUE の場合は Q1 を FALSE にします。SET と RESET1 が同時に TRUE の場合は RESET1 が優先され、Q1 は FALSE となります。



■入出力パラメータ

入力パラメータ	説　明	備　考
SET(var1)	セット入力	
RESET1(var2)	リセット入力	

出力パラメータ	説　明	備　考
Q1(var3)	出力	初期値 FALSE

■使用可能なデータ型
SET、RESET1、Q1：BOOL

■例
FBD 言語

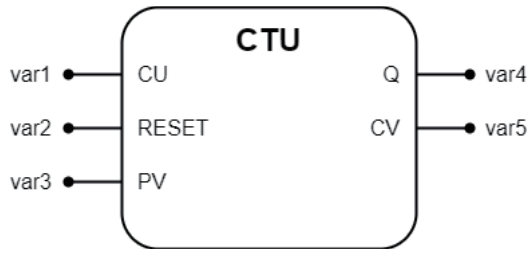
IL 言語

```
1   CAL          RS_0(
              SET:= bSET          TRUE
            RESET1:= bRESET1      TRUE
               Q1=> bQ1)          FALSE
```

ST 言語

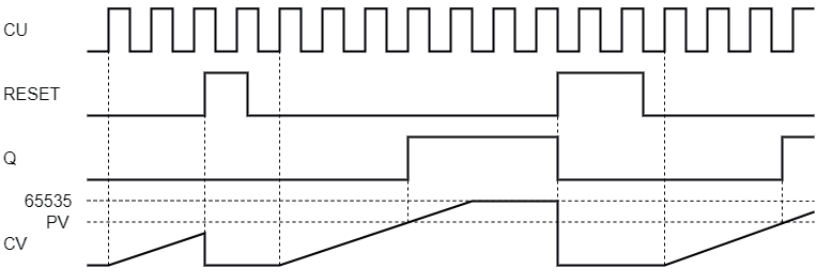
```
1
2   RS_0(SET TRUE := bSET TRUE, RESET1 TRUE := bRESET1 TRUE, Q1 FALSE => bQ1 FALSE);
3
```

7.5.3 CTU：アップカウンタ



■機能

入力パラメータ CU に接続された値の立ち上がりエッジを検出するたびにカウントアップし、出力パラメータ CV を更新します。CV が入力パラメータ PV に接続された値以上になると、出力パラメータ Q に TRUE を返します。入力パラメータ RESET が TRUE の間は、Q に FALSE、CV に 0 を返します。



■入出力パラメータ

入力パラメータ	説　明	備　考
CU(var1)	カウントアップトリガ入力	立ち上がりエッジでカウントアップ
RESET(var2)	リセット入力	
PV(var3)	目標値	

出力パラメータ	説　明	備　考
Q(var4)	出力	CV＞＝PV のとき TRUE 初期値 FALSE
CV(var5)	現在値	

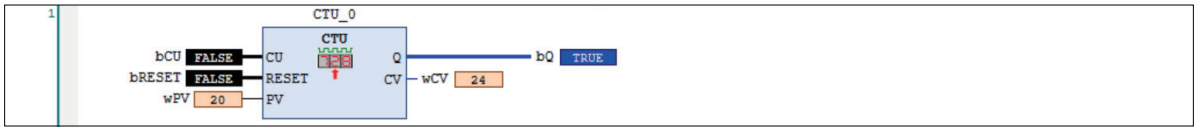
■使用可能なデータ型

CU、RESET、Q：BOOL

PV、CV：WORD

■例

FBD 言語



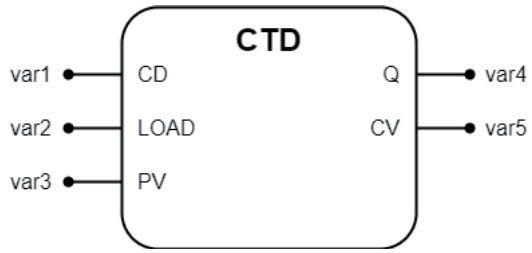
IL 言語

```
CAL          CTU_0(
     CU:= bCU          FALSE
     RESET:= bRESET    FALSE
     PV:= wPV          20
     Q=> bQ            TRUE
     CV=> wCV)         24
```

ST 言語

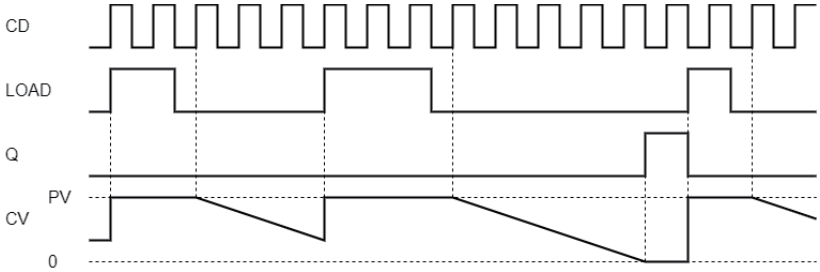
```
CTU_0(CU:= bCU, RESET:= bRESET, PV:= wPV, Q=> bQ, CV=> wCV);
```

7.5.4 CTD：ダウンカウンタ



■機能

入力パラメータ CD に接続された値の立ち上がりエッジを検出するたびにカウントダウンし、出力パラメータ CV を更新します。CV が 0 になると、出力パラメータ Q に TRUE を返します。入力パラメータ LOAD が TRUE の間は、Q に FALSE、CV に入力パラメータ PV に接続された値を返します。



■入出力パラメータ

入力パラメータ	説　明	備　考
CD(var1)	カウントダウントリガ入力	立ち上がりエッジでカウントダウン
LOAD(var2)	ロード入力	
PV(var3)	目標値	

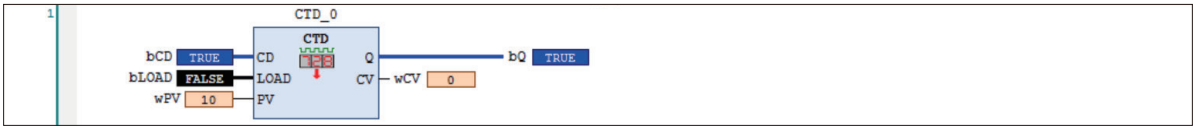
出力パラメータ	説　明	備　考
Q(var4)	出力	CV＝0 のとき TRUE 初期値 FALSE
CV(var5)	現在値	

■使用可能なデータ型

CD、LOAD、Q：BOOL
PV、CV：WORD

■例

FBD 言語



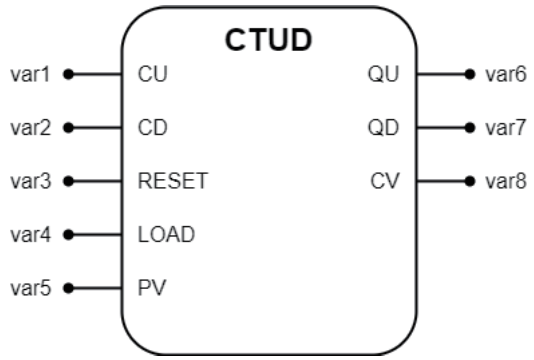
IL 言語

```
1   CAL          CTD_0(
                 CD:= bCD          TRUE
                 LOAD:= bLOAD      FALSE
                 PV:= wPV          10
                 Q=> bQ            TRUE
                 CV=> wCV)         0
```

ST 言語

```
1
2   CTD_0(CD TRUE := bCD TRUE, LOAD FALSE := bLOAD FALSE, PV 10 := wPV 10, Q TRUE => bQ TRUE, CV 0 => wCV 0);
3
```

7.5.5 CTUD：アップダウンカウンタ



■機能

入力パラメータ CU に接続された値の立ち上がりエッジを検出するたびにカウントアップし、入力パラメータ CD に接続された値の立ち上がりエッジを検出するたびにカウントダウンします。出力パラメータ CV が入力パラメータ PV に接続された値以上になると出力パラメータ QU に TRUE、CV が 0 になると QD に TRUE を返します。入力パラメータ RESET が TRUE の間は、QU および QD に FALSE、CV に 0 を返します。入力パラメータ LOAD が TRUE の間は、QU および QD に FALSE、CV に PV に接続された値を返します。

■入出力パラメータ

入力パラメータ	説　明	備　考
CU(var1)	カウントアップトリガ入力	立ち上がりエッジでカウントアップ
CD(var2)	カウントダウントリガ入力	立ち上がりエッジでカウントダウン
RESET(var3)	リセット入力	
LOAD(var4)	ロード入力	
PV(var5)	目標値	

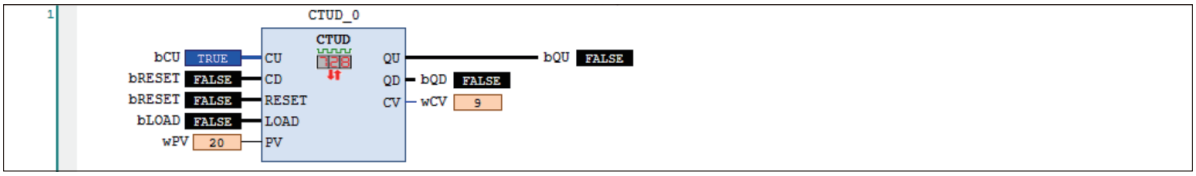
出力パラメータ	説　明	備　考
QU(var6)	出力	CV＞＝PV のとき TRUE 初期値 FALSE
QD(var7)	出力	CV＝0 のとき TRUE 初期値 FALSE
CV(var8)	現在値	

■使用可能なデータ型

CU、CD、RESET、LOAD、QU、QD：BOOL
PV、CV：WORD

■例

FBD 言語



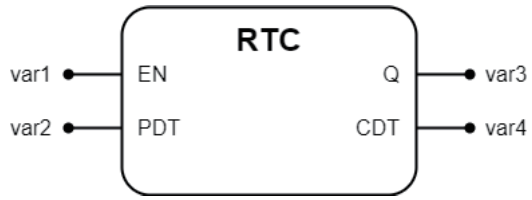
IL 言語

```
CAL          CTUD_0(
     CU:= bCU          TRUE
     CD:= bCD          FALSE
  RESET:= bRESET       FALSE
   LOAD:= bLOAD        FALSE
     PV:= wPV          20
     QU=> bQU          FALSE
     QD=> bQD          FALSE
     CV=> wCV)         9
```

ST 言語

```
CTUD_0(CU TRUE := bCU TRUE, CD FALSE := bCD FALSE, RESET FALSE := bRESET FALSE, LOAD FALSE := bLOAD FALSE, PV 20 := wPV 20,
       QU FALSE => bQU FALSE, QD FALSE => bQD FALSE, CV 9 => wCV 9 );
```

7.5.6 RTC：リアルタイムクロック



■機能

入力パラメータ PDT に接続された日時からの経過日時を出力パラメータ CDT に返します。入力パラメータ EN が TRUE の間は、出力パラメータ Q に TRUE、CDT に PDT からの経過日時を返します。EN が FALSE の場合は、Q に FALSE、CDT に DT#1970-01-01-00:00:00 を返します。

RTC は 32 ビットデータ型 DATE_AND_TIME（DT）で演算を行うため、2106 年 2 月 7 日にオーバーフローします。

■入出力パラメータ

入力パラメータ	説　明	備　考
EN（var1）	開始入力	
PDT（var2）	プリセット日時	

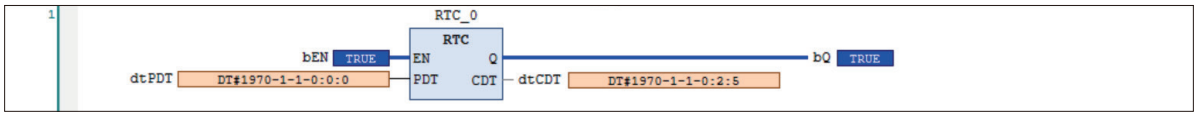
出力パラメータ	説　明	備　考
Q（var3）	出力	初期値 FALSE
CDT（var4）	経過日時	

■使用可能なデータ型

EN、Q：BOOL
PDT、CDT：DATE_AND_TIME（DT）

■例

FBD 言語



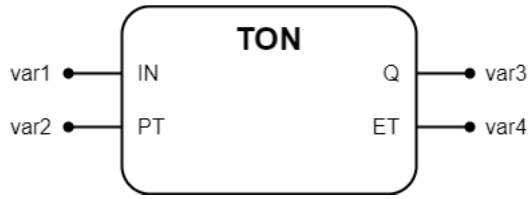
IL 言語

```
CAL          RTC_0(
     EN:= bEN          TRUE
    PDT:= dtPDT        DT#1970-1-1-0:0:0
      Q=> bQ           TRUE
    CDT=> dtCDT)       DT#1970-1-1-0:2:5
```

ST 言語

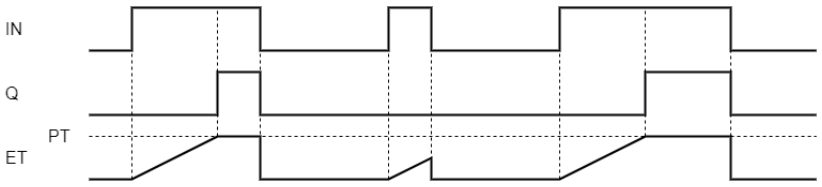
```
RTC_0(EN TRUE := bEN TRUE, PDT DT#1970-1-1-0:0:0 := dtPDT DT#1970-1-1-0:0:0,
      Q TRUE => bQ TRUE, CDT DT#1970-1-1-0:2:5 => dtCDT DT#1970-1-1-0:2:5 );
```

7.5.7 TON：オンディレイタイマ



■機能

入力パラメータ IN に接続された値の立ち上がりエッジから、入力パラメータ PT に接続された時間が経過すると、出力パラメータ Q に TRUE を返します。IN が TRUE の間は、経過時間が出力パラメータ ET に返されます。IN が FALSE になると、Q に FALSE、ET に 0 を返します。



■入出力パラメータ

入力パラメータ	説　明	備　考
IN(var1)	タイマ開始入力	立ち上がりエッジでタイマ開始
PT(var2)	プリセット時間	

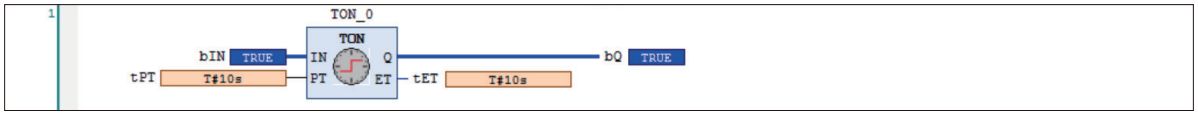
出力パラメータ	説　明	備　考
Q(var3)	出力	初期値 FALSE
ET(var4)	経過時間	

■使用可能なデータ型

IN、Q：BOOL
PT、ET：TIME

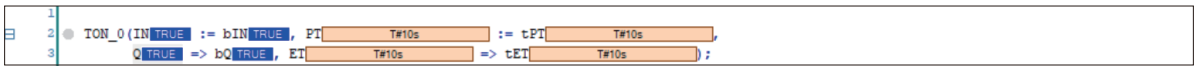
■例

FBD 言語

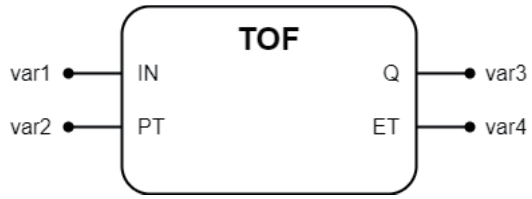


IL 言語

ST 言語

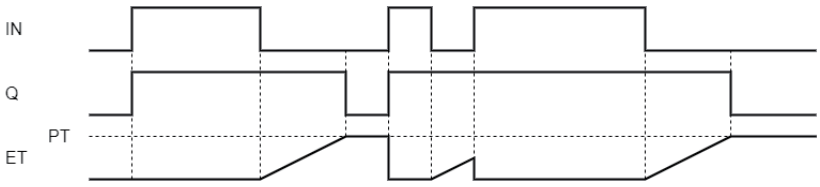


7.5.8 TOF：オフディレイタイマ



■機能

入力パラメータ IN に接続された値の立ち下がりエッジから、入力パラメータ PT に接続された時間が経過すると、出力パラメータ Q に FALSE を返します。IN が FALSE の間は、経過時間が出力パラメータ ET に返されます。IN が TRUE になると、Q に TRUE、ET に 0 を返します。



■入出力パラメータ

入力パラメータ	説　明	備　考
IN(var1)	タイマ開始入力	立ち下がりエッジでタイマ開始
PT(var2)	プリセット時間	

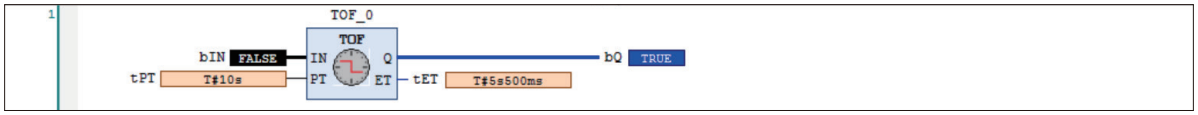
出力パラメータ	説　明	備　考
Q(var3)	出力	初期値 FALSE
ET(var4)	経過時間	

■使用可能なデータ型

IN、Q：BOOL
PT、ET：TIME

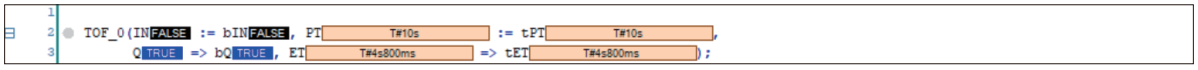
■例

FBD 言語

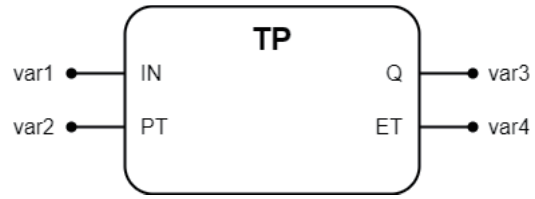


IL 言語

ST 言語

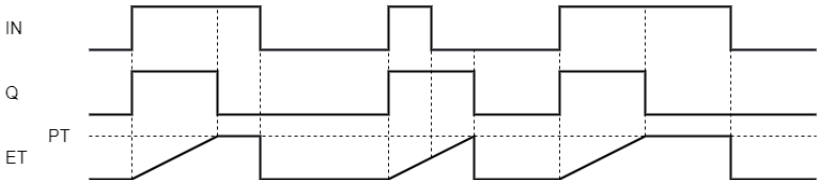


7.5.9 TP：パルスタイマ



■機能

入力パラメータ IN に接続された値の立ち上がりエッジから、入力パラメータ PT に接続された時間が経過するまで、出力パラメータ Q に TRUE、出力パラメータ ET に経過時間を返します。PT の時間が経過すると、Q に FALSE を返します。



■入出力パラメータ

入力パラメータ	説　明	備　考
IN(var1)	タイマ開始入力	立ち上がりエッジでタイマ開始
PT(var2)	プリセット時間	

出力パラメータ	説　明	備　考
Q(var3)	出力	初期値 FALSE
ET(var4)	経過時間	

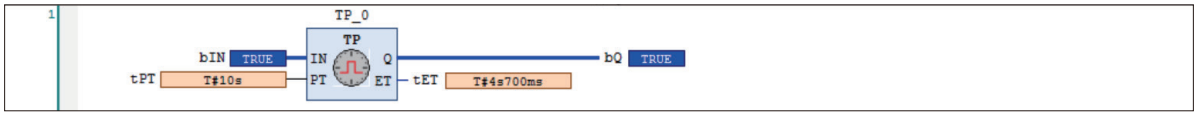
■使用可能なデータ型

IN、Q：BOOL

PT、ET：TIME

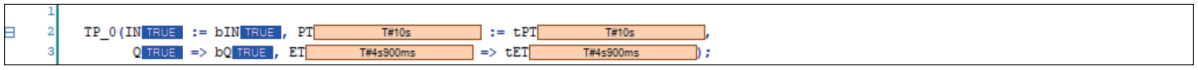
■例

FBD 言語

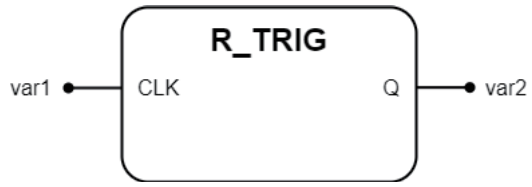


IL 言語

ST 言語

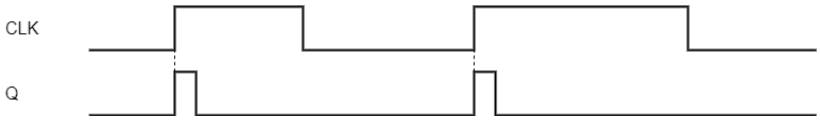


7.5.10 R_TRIG：立ち上がりエッジ検出



■機能

入力パラメータ CLK に接続された値の立ち上がりエッジを検出すると、出力パラメータ Q に TRUE（単一パルス）を 1 周期だけ返します。



■入出力パラメータ

入力パラメータ	説　明	備　考
CLK(var1)	入力	

出力パラメータ	説　明	備　考
Q(var2)	出力	初期値 FALSE

■使用可能なデータ型

CLK、Q：BOOL

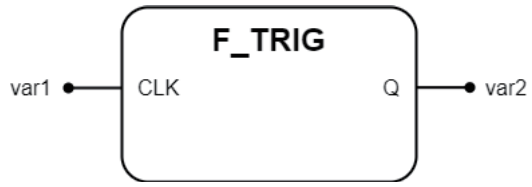
■例

FBD 言語

IL 言語

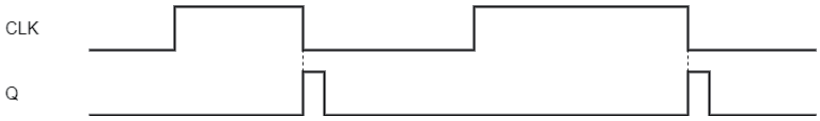
ST 言語

7.5.11 F_TRIG：立ち下がりエッジ検出



■機能

入力パラメータ CLK に接続された値の立ち下がりエッジを検出すると、出力パラメータ Q に TRUE（単一パルス）を 1 周期だけ返します。



■入出力パラメータ

入力パラメータ	説　明	備　考
CLK(var1)	入力	

出力パラメータ	説　明	備　考
Q(var2)	出力	初期値 FALSE

■使用可能なデータ型

CLK、Q：BOOL

■例

FBD 言語

IL 言語

ST 言語

8. 付録

8.1 ゼロ除算

ゼロ除算（0 で割る処理）は、実行時に異常動作やエラーの原因となるため注意が必要です。CODESYS では、動作環境によってその挙動が異なります。

■ターゲット（コントローラ）
ターゲット上でゼロ除算が発生した場合、挙動は以下のとおりです。
・整数型（INT など）：結果は 0 になります。
・実数型（REAL など）：結果は＋Infinity または－Infinity になります。
プログラムは異常終了せず、継続して動作します。

■シミュレーション
シミュレーションでゼロ除算が発生した場合、CODESYS IDE 上で例外処理が発生し、プログラムが停止する場合があります。

8.2 オンライン変更

オンライン変更は、実行中のプログラムを停止せずに、コードの修正や設定の変更を反映する機能です。メモリ構成やインスタンス管理に影響を及ぼす可能性があるため、以下の点には十分注意して下さい。

■アドレス・ポインタ
オンライン変更を行うと、内部のメモリ構成が再配置され、既存のポインタ変数の参照先が無効になる可能性があります。そのため、ポインタを使用している場合は、ポインタの初期化や更新処理が毎サイクル実行されるように実装して下さい。

■インスタンスの追加や削除
オンライン変更中にインスタンスの追加や削除を行うと、エラーや警告が発生することがあります。これらの操作を行ったあとは、コード生成の前にメニューから「ビルド」→「クリア」をクリックして、クリーン作業を行って下さい。

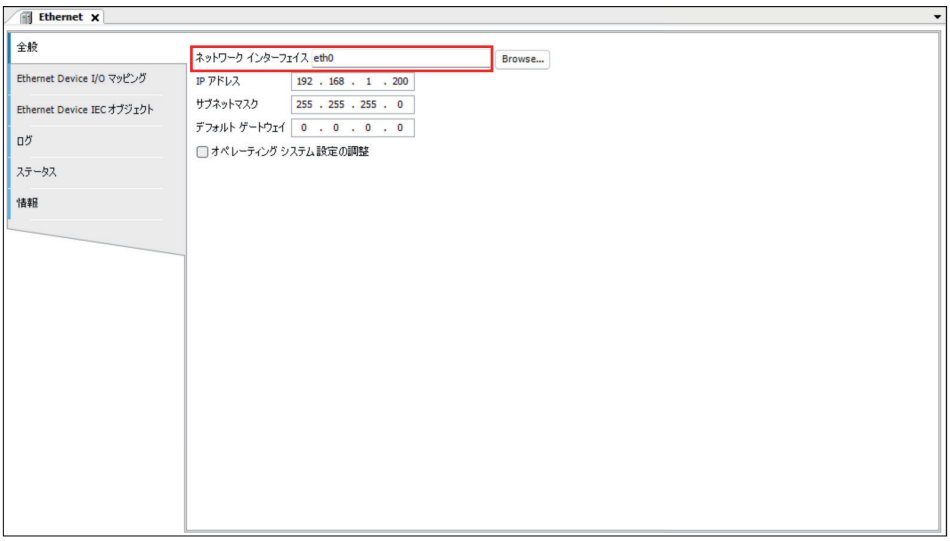
オンライン変更は、小規模な修正やパラメータ調整に有効です。ただし、データ構造の大幅な変更、複数の変数や POU をまたぐ修正、通信設定の変更などを含む場合には、オンライン変更による運用が不安定になる可能性があります。そのため、大幅な変更を行う際は、停止状態での完全ダウンロードをおすすめします。

8.3 ハードウェアとの対応

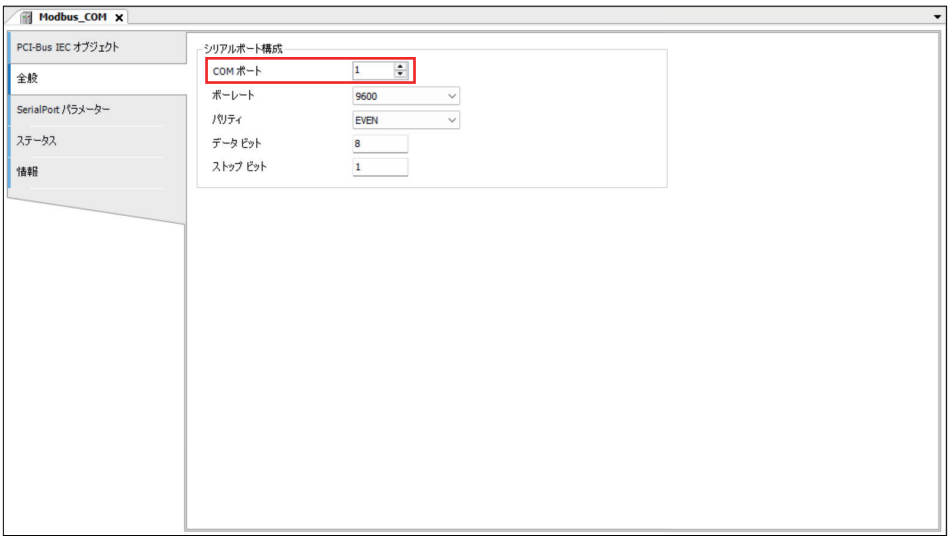
名　称		
RJ-45 モジュラジャック	LAN1	eth0
	LAN2	eth1
RS-485 用コネクタ	RS-485 1	COM ポート 1
	RS-485 2	COM ポート 2
CAN 用コネクタ	CAN1	ネットワーク 0
	CAN2	ネットワーク 1
SD カードスロット	SD	oem_mmc
ヘッドフォン出力用オーディオジャック	PHONE	使用不可
ライン入力用オーディオジャック	LINE	使用不可
マイク入力用オーディオジャック	MIC	使用不可
設定用ジャックコネクタ	Config	使用不可
Micro-USB コネクタ	USB1	使用不可
USB コネクタ	USB2	使用不可
機能設定用ディップスイッチ	SW1	[FB] MgHWGetSw（設定不可）
設定用ロータリースイッチ	SW2,3	[FB] MgHWGetSw
工場調整用スイッチ	SW4,5,6,7	使用不可
状態表示ランプ	RUN,ERR,SD,PLC	[FB] MgHWSetLed（設定不可）
	USR	[FB] MgHWSetLed
FLASH ディスク＊1	－	oem_flash
RAM ディスク ＊1	－	oem_tmp

＊ 1、ユーザが使用できる領域の目安は FLASH ディスクで 4 GB 以下、RAM ディスクで 512 MB 以下となります。

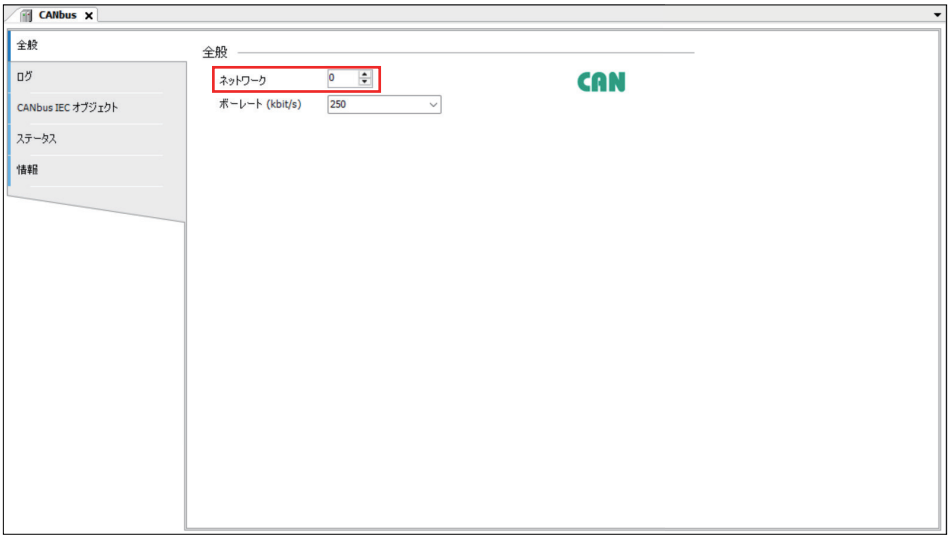
RJ-45 モジュラジャック



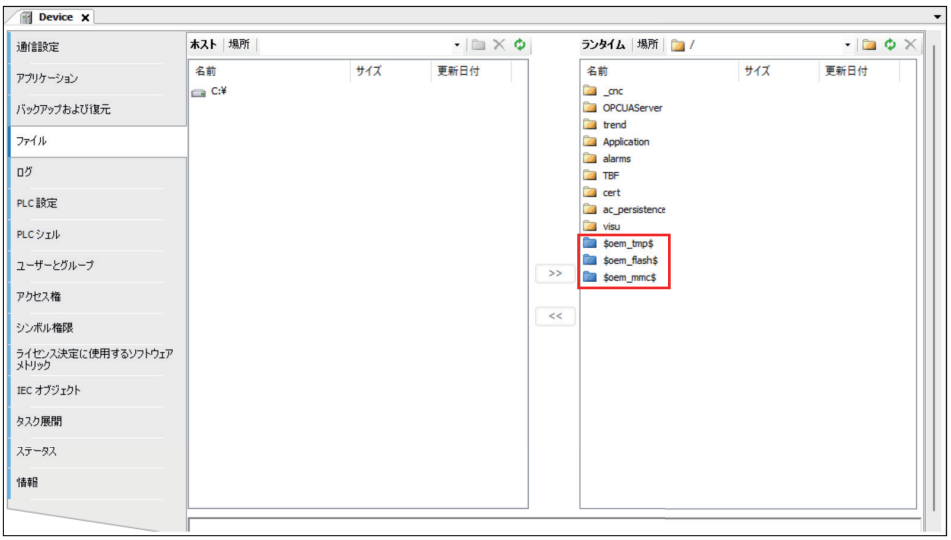
RS-485 用コネクタ



CAN 用コネクタ



SD カードスロット
FLASH ディスク
RAM ディスク



9. 履歴

改訂番号	バージョン	内　容
初版	1.0.0.16	初版

株式会社エムジー

〒541-0042
大阪市中央区今橋2丁目5番8号（トレードピア淀屋橋13F）
TEL 06－7525－8800
E-mail hotline@mgco.jp