

オートメーションソフトウェア Sysmac Studio

スタートアップガイド プロジェクトバージョン管理機能編

形SYSMAC-SE2□□□

形SYSMAC-TA4□□L

Startup
Guide

おことわり

- 本マニュアルの内容の一部または全部を無断で複写、複製、転載することを禁じます。
- 本マニュアルの内容に関しては、改良のため予告なしに仕様などを変更することがあります。あらかじめご了承ください。
- 本マニュアルの内容に関しては、万全を期しておりますが、万一ご不審な点や誤りなど、お気づきのことがありましたら、お手数ですが、弊社支店または営業所までご連絡ください。その際、巻末記載の**Man.No.** (マニュアルNo.) も併せてお知らせください。

商標

- **Sysmac**は、オムロン株式会社**FA**機器製品の日本およびその他の国における商標または登録商標です。
- **Microsoft**、**Windows**、**Excel**、**Visual Basic**は、米国**Microsoft Corporation**の米国およびその他の国における登録商標または商標です。
- **EtherCAT®**は、ドイツ**Beckhoff Automation GmbH**によりライセンスされた特許取得済み技術であり登録商標です。
- **ODVA**、**CIP**、**CompoNet**、**DeviceNet**、**EtherNet/IP**は**ODVA**の商標です。
- **SD**、**SDHC**ロゴは、**SD-3C**、**LLC**の商標です。 
- **NVIDIA**、**NVIDIA** ロゴ、**GeForce**、**GeForce** ロゴは、米国およびその他の国における**NVIDIA Corporation** の登録商標または商標です。
- **ATI™**、**Radeon™** は、**Advanced Micro Devices, Inc.** の商標です。
- **Celeron**、**Intel**、**Intel Core**は、米国およびその他の国における**Intel Corporation**の商標です。
- **Git** and the **Git logo** are either registered trademarks or trademarks of **Software Freedom Conservancy, Inc.**, corporate home of the **Git Project**, in the United States and/or other countries.

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

著作権

スクリーンショットはマイクロソフトの許可を得て使用しています。

目次

はじめに	3
対象となる読者の方々	3
対象となる製品	3
ご承諾事項	3
安全上の注意	4
法規と規格	4
海外でのご使用	4
ソフトウェアのライセンスと著作権	4
改訂履歴	4
1 概要	5
2 用語説明	6
3 システム構成	7
3.1 基本構成	7
3.2 複数のユーザでリポジトリを共有する構成	7
3.3 バージョン管理システムの基本操作	9
4 バージョン管理の対象範囲	11
5 ソフトウェアのセットアップおよび基本設定	12
5.1 Sysmac Studio のインストール	12
5.2 Sysmac Studio チーム開発オプションの登録	12
5.3 Git のインストール	12
5.4 “TortoiseGit” のインストール	18
5.5 “TortoiseGit” の初期設定	21
5.6 共有フォルダとリモートリポジトリの作成	23
5.7 “TortoiseGit” の追加設定	24
6 SYSMAC STUDIO プロジェクトの履歴管理	27
6.1 バージョン管理機能を使用した履歴管理機能の概要	27
6.2 履歴管理機能の実行例	28
6.3 プロジェクト管理の開始準備（ベースとなるプロジェクトの作成、保存）	28
6.3.1 ベースとなるプロジェクトの作成	28
6.3.2 ローカルリポジトリへの登録	30
6.3.3 ローカルリポジトリとリモートリポジトリの同期設定	33
6.4 プロジェクトの更新	36

6.4.1	プロジェクトのデータの変更	36
6.4.2	変更内容のローカルリポジトリへの登録（コミット）	36
6.4.3	変更内容のリモートリポジトリへの登録（プッシュ）	38
6.5	プロジェクトの検索	40
6.5.1	ローカルリポジトリの更新	40
6.5.2	プロジェクトのログの表示	41
6.5.3	リビジョンの絞り込み	42
6.6	プロジェクトの比較	43
6.7	プロジェクトの復元	44
7	SYSMAC STUDIO プロジェクトの複数人開発と派生開発	47
7.1	バージョン管理機能を使用した複数人開発および派生開発の概要	47
7.2	ブランチとマージ	48
7.3	Sysmac Studio プロジェクトの複数人開発手順	50
7.3.1	複数人で並行して開発するための編集の分担	50
7.3.2	分担例	51
7.3.3	操作手順	53
7.4	Sysmac Studio プロジェクトの派生開発	97
7.4.1	派生開発の例	97
7.4.2	バージョン管理システム上の操作の流れ	97
7.4.3	操作手順	99

はじめに

このたびは、Sysmac Studio チーム開発オプション をお買い上げいただきまして、ありがとうございます。
本ガイドは、Sysmac Studio チーム開発オプションおよびプロジェクトバージョン管理機能について、基本的な機能の操作手順を説明しています。本ガイドを参考に操作手順をご理解いただき、システムの構築にご活用ください。

対象となる読者の方々

対象となる読者の方々

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

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

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

対象となる製品

本マニュアルは以下の製品を対象にしています。

Sysmac Studio チーム開発オプション

各製品の仕様の一部または制約事項が、他のマニュアルに記載されている場合があります。

『Sysmac Studio Version 1 オペレーション マニュアル(SBCA-470)』および『Sysmac Studio プロジェクトバージョン管理機能 オペレーションマニュアル(SBCA-460)』をご確認ください。

ご承諾事項

- 保証内容
 1. 保証期間
本ソフトウェアの保証期間は、ご購入後 1 年といたします。
(ただし「カタログ等」に別途記載がある場合を除きます。)
 2. 保証範囲
 - A) 本ソフトウェアの使用許諾を受けたお客様が、上記保証期間中にコンピュータ・プログラムの瑕疵（マニュアルとの重要な不一致）を発見し、当社に返却した場合は、当社は瑕疵（当該マニュアルとの重要な不一致）のないコンピュータ・プログラムを記録した媒体と交換いたします。もしくは、当社の選択により、当社ホームページより瑕疵（当該マニュアルとの重要な不一致）のないコンピュータ・プログラムをダウンロードしていただく方法により提供いたします。また、当社の責任によるコンピュータ・プログラムの記録媒体の不良を発見し、当社に返却した場合、当社は、無償で、良品の媒体に記録したコンピュータ・プログラムと交換いたします。
 - B) 万一、当社がコンピュータ・プログラムの瑕疵（マニュアルとの重要な不一致）を除去できないと判断した場合は、お客様が本ソフトウェア購入代金として支払った金額をお返しいたします。
- 責任の制限
 1. 前条に定める交換または購入代金の返金は、本ソフトウェアの保証責任のすべてを定めるものであり、当社は本ソフトウェアの瑕疵により発生した、お客様の直接的、間接的あるいは波及効果による損害等いかなる損害に対しても一切の責任を負いません。
 2. 当社は、本ソフトウェアを当社以外の第三者が変更、改造することにより生じた瑕疵や、本ソフトウェアのインストールその他の使用によって当社商品、インストールされたソフトウェア、またはすべてのコン

ピュータ機器、コンピュータプログラム、ネットワーク、データベースが DDoS 攻撃（分散型 DoS 攻撃）、コンピュータウイルスその他の技術的な有害プログラム、不正アクセスを受けたことにより生じた損害につきましては、一切責任を負いません。

3. 当社は、本ソフトウェアに基づき、当社以外の第三者が開発したソフトウェアおよびそれにより生じた結果について一切の責任を負いません。

- 本ソフトウェアの用途
本ソフトウェアを本マニュアルに記載の用途以外の用途で使用しないでください。
- 仕様の変更
本ソフトウェアの仕様および付属品は改善またはその他の事由により、必要に応じて、変更される場合があります。
- 適用範囲
以上の内容は、日本国内での取引および使用を前提としております。日本国外での取引および使用に関しては、当社営業担当者までご相談ください。

安全上の注意

『Sysmac Studio Version 1 オペレーション マニュアル(SBCA-470)』および『Sysmac Studio プロジェクトバージョン管理機能 オペレーションマニュアル(SBCA-460)』をご確認ください。

法規と規格

海外でのご使用

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

ソフトウェアのライセンスと著作権

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

改訂履歴

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

Man.No. **SBCA-112C**

↑ 改訂記号

改訂記号	改訂年月	改訂理由・改訂ページ
A	2017 年 10 月	初版
B	2019 年 4 月	ご承諾事項の改定
C	2020 年 1 月	Sysmac Studio 64 ビット版対応に伴う改訂

1 概要

生産装置開発の現場では、生産装置の大規模化により次の問題があります。

- ・ 生産装置のコントローラのプログラム開発規模の増大、開発期間の長期化
- ・ 生産装置のバリエーションの増加に伴う、コントローラのプログラムの変更工数の増大

これらの問題への対策として、それぞれ次のような開発環境が求められています。

- ・ 複数人の開発者によるコントローラのプログラム開発環境
- ・ 装置バリエーション間の共通プログラムの変更を少ない操作で確実にできる開発環境

これらの課題の解決手段として、Sysmac Studio では装置のプロジェクトバージョン管理機能（以降、バージョン管理機能と呼ぶことがあります）をご提供します。バージョン管理機能は、ソフトウェア開発環境として標準となっているオープンソースソフトウェアのバージョン管理システムを Sysmac Studio と連動させることで実現します。

本書では、オープンソースソフトウェアのバージョン管理システムの導入の手順から Sysmac Studio での初期設定、バージョン管理の基本機能である履歴管理、複数人で開発する場合、派生開発を行う場合のそれぞれの操作方を説明します。

2 用語説明

本書で使用する用語は次のとおりです。

用語	説明
Sysmac Studio プロジェクトバージョン管理機能	Sysmac Studio のプロジェクトの変更内容や履歴を管理する機能です。本書では「Sysmac Studio バージョン管理機能」または「バージョン管理機能」と呼ぶことがあります。
Git™	オープンソースソフトウェアの名称で、バージョン管理システムのひとつです。Sysmac Studio と連動して、Sysmac Studio プロジェクトのバージョン管理機能を実現します。
“TortoiseGit” *1	バージョン管理システムのクライアントソフトウェアの名称です。Sysmac Studio から“TortoiseGit”のメニューや画面を使用してバージョン管理の操作を行います。
リポジトリ	Git が管理するデータの格納場所です。バージョン管理対象の Sysmac Studio のプロジェクトデータとその変更履歴情報が格納されます。
リモートリポジトリ	サーバーなどネットワーク上や、共有フォルダにあるリポジトリのことです。
ローカルリポジトリ	各ユーザのパソコン内にあるリポジトリのことです。
複数人開発	複数のユーザで分担して一つのプロジェクトのプログラムを開発することです。各人の変更の内容やタイミングを管理することが重要になります。
派生開発	装置のバリエーションに応じて、ある装置のプログラムから別の装置のプログラムを開発することです。装置間で共通の機能のプログラムの変更を、すべての装置のプログラムへ漏れなく反映したり、装置固有の機能の変更を管理したりすることが重要になります。
プッシュ	ローカルリポジトリの変更をリモートリポジトリへ反映する操作です。
プル	リモートリポジトリの変更をローカルリポジトリに取り込む操作です。
チェンジセット	プロジェクトの変更情報のかたまりです。Sysmac Studio のプロジェクトにおいては、たとえば、グローバル変数を追加し、ラダープログラムを追加する、といったプロジェクトの変更内容をまとめた情報のことです。Sysmac Studio バージョン管理機能では、プロジェクトのデータの変更情報を記憶し、変更内容を時系列に一覧表示したりプロジェクトを特定の状態に戻す際の情報のかたまりとして、本用語を用います。
コミット	プロジェクトの変更情報のかたまりであるチェンジセットを、Git が管理するローカルリポジトリに登録することを言います。Sysmac Studio には従来の保存機能もありますが、これは作業フォルダへの保存です。一方でローカルリポジトリへの登録にはコミットという操作が必要です。 コミットを行う際、チェンジセットの情報に加え、その操作を行った日付・時刻情報、ユーザ名が自動で登録されます。加えて、変更内容を容易に検索できるようにするためにコメント（ログメッセージ）を入力して登録することもできます。
リビジョン	コミットを行うと、チェンジセットの検索を容易にするための情報を付加してリポジトリに登録されます。その際、チェンジセットを一意に区別するためのリビジョンという番号（リビジョン識別番号と呼びます）がコミット時に自動で付加されます。編集集中のプロジェクトを特定のチェンジセットに戻したり、特定のチェンジセットと編集集中のプロジェクトの比較、プロジェクトの分岐、プロジェクトの結合をしたりする場合には、このリビジョン識別番号を用いてチェンジセットを指定することになります。Git でのリビジョン識別番号は、SHA-1 というアルゴリズムで生成された 16 進数 40 桁の値です。

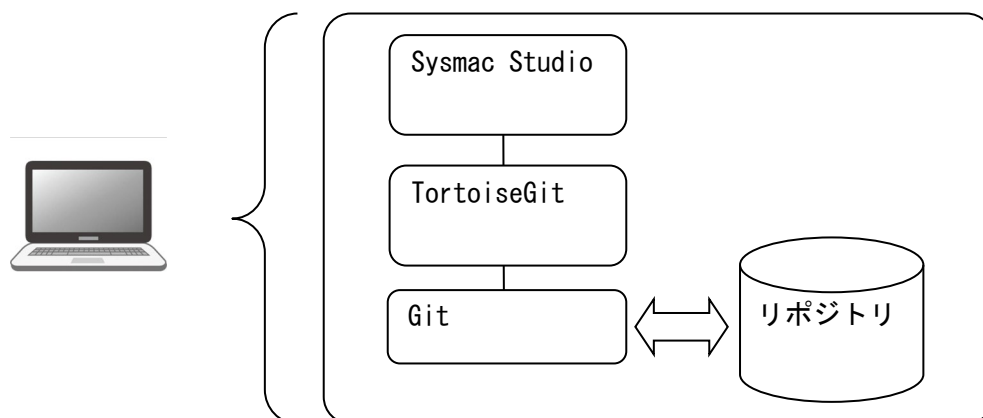
*1 “TortoiseGit”はオープンソースであり、(<https://tortoisegit.org/download/>)からダウンロードしました。

3 システム構成

3.1 基本構成

Sysmac Studio バージョン管理機能は、Sysmac Studio、バージョン管理システムの” Git”、および Git のクライアントソフトウェアの” TortoiseGit” と、Git が管理するフォルダである” リポジトリ” とで構成された環境で動作します。

下の図は、Sysmac Studio プロジェクトにアクセスするユーザが 1 名の場合の最小構成です。



Sysmac Studio バージョン管理機能環境の基本構成（1 パソコン内）

3.2 複数のユーザでリポジトリを共有する構成

Sysmac Studio バージョン管理機能は、分散型バージョン管理システムという特徴を持つ Git と連携することで、複数のユーザでリポジトリを共有する形態で運用することができます。

ユーザごとに作業をするための各ユーザのパソコンのローカルリポジトリと、ユーザ間で共有するリポジトリであるリモートリポジトリを持ち、各ユーザの作業のタイミングでローカルリポジトリとリモートリポジトリの同期をとることができます。

ローカルリポジトリでの変更を他のユーザと共有するときはリモートリポジトリへのプッシュ操作を、他のユーザの変更をローカルリポジトリに反映するときはリモートリポジトリからのプル操作を行います。

実際の運用形態には、リモートリポジトリの共有方法の違いにより次の 3 つの形態があります。

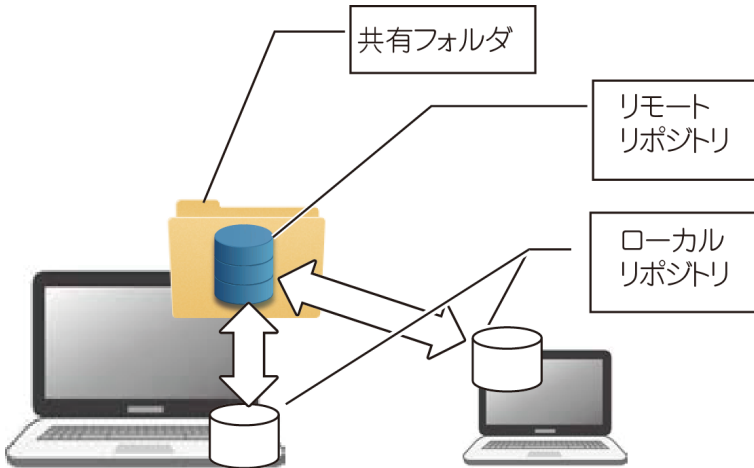
- (1) リモートリポジトリをパソコンの共有フォルダで共有する
- (2) Git 専用のサーバを構築し運用する
- (3) インターネット上の Git サーバサービスを活用して運用する

(1) リモートリポジトリをパソコンの共有フォルダとして設置して運用する

最も簡単に構築できる方法です。

以下の図のように、ローカルリポジトリと同期するリモートリポジトリを、パソコンの共有フォルダとしてローカルネットワークに公開し、他のパソコンからアクセスできるようにする方式です。

以降の説明では、本形態の構成を用いて説明します。

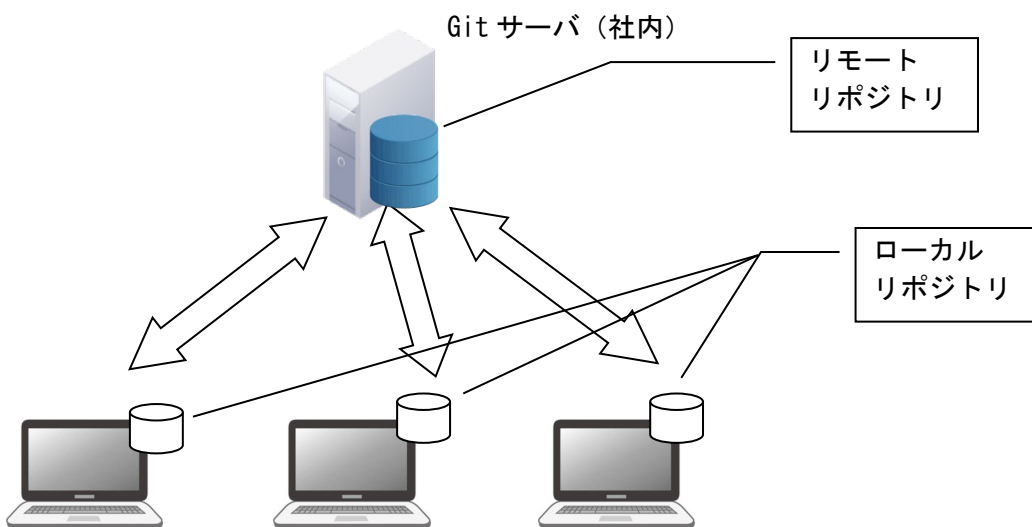


リモートリポジトリをパソコンの共有フォルダで共有して運用する

(2) Git 専用のサーバを構築し運用する

Git 専用のサーバを構築し、リモートリポジトリを運用することができます。

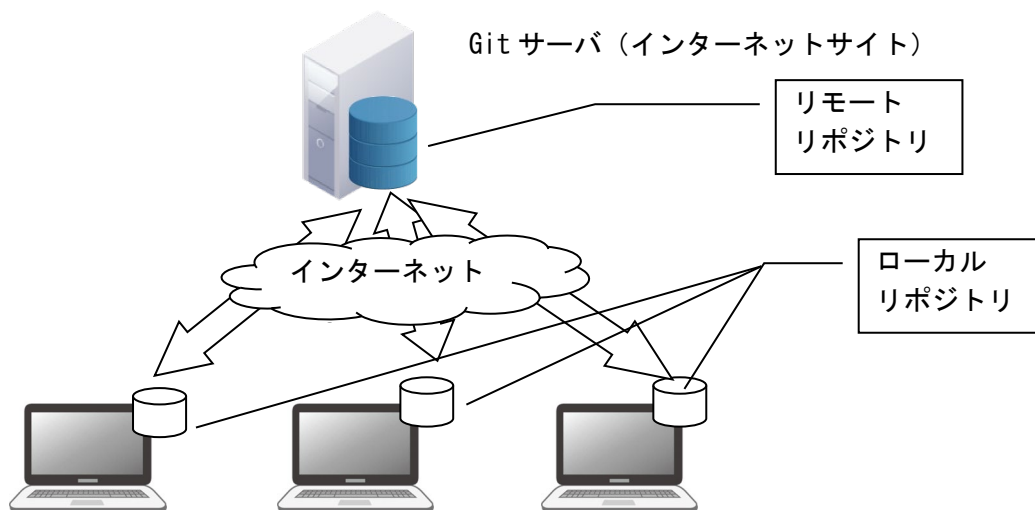
Git サーバソフトウェアには、Windows 用の Gitbucket、Linux 用の GitLab、Gitbucket、Gitblit、Gogs などがあります。サーバの構築や保守メンテナンスコストがかかりますが、自社内に閉じているため、データ流出のリスクを低減できるメリットがあります。



各ユーザのパソコンの Sysmac Studio バージョン管理システム

(3) インターネット上の Git サーバサービスを活用して運用する

インターネット上には、GitHub などの Git サーバのサービスがあります。商用では有償となりコストがかかりますが、サーバの保守メンテナンスが不要、社外の開発者と並行して開発ができるというメリットがあります。



各ユーザのパソコンの Sysmac Studio バージョン管理システム

3.3 バージョン管理システムの基本操作

Sysmac Studio での、バージョン管理システムの基本操作を説明します。

1) 変更内容のローカルリポジトリへの登録（コミット）

SysmacStudio で編集したプロジェクトをローカルリポジトリに登録する場合は、コミットを行います。詳細手順については、「6. 4. 2 変更内容のローカルリポジトリへの登録（コミット）」を参照してください。

2) 変更内容のリモートリポジトリへの登録（プッシュ）

ローカルリポジトリに登録したプロジェクトを、リモートリポジトリに反映します。これをプッシュと呼びます。詳細手順については、「6. 4. 3 変更内容のリモートリポジトリへの登録（プッシュ）」を参照してください。

3) ローカルリポジトリの更新（プル）

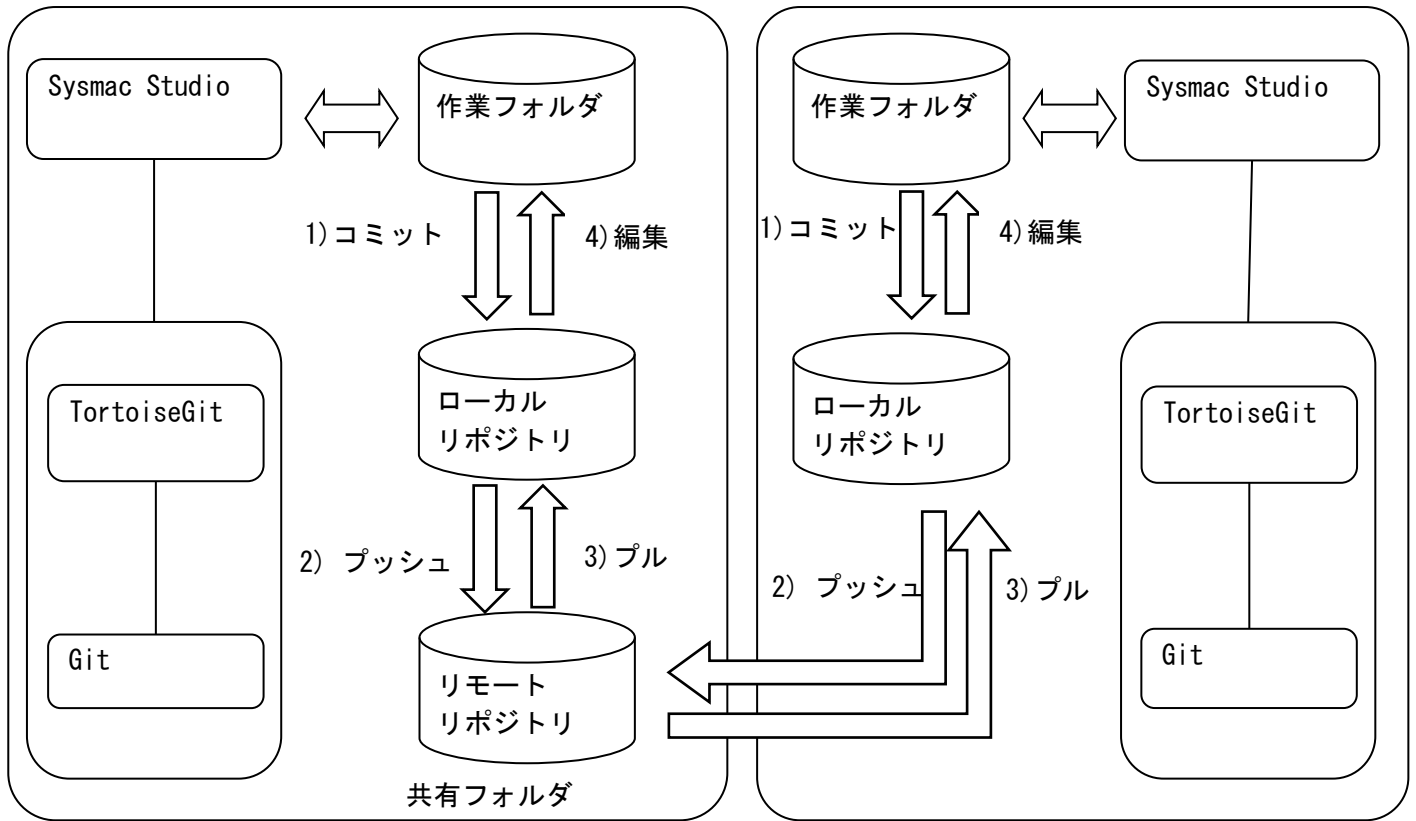
リモートリポジトリの内容を、ローカルリポジトリに反映します。これをプルと呼びます。詳細手順については、「6. 5. 1 ローカルリポジトリの更新」を参照してください。

4) プロジェクトの編集

ローカルリポジトリ内の選択したプロジェクトを、Sysmac Studio で表示・編集します。詳細手順については、「6. 4. 1 プロジェクトのデータの変更」を参照してください。

ユーザ A のパソコンの Sysmac Studio バージョン管理システム

ユーザ B のパソコンの Sysmac Studio バージョン管理システム



Sysmac Studio での、バージョン管理システムの基本操作
(リモートリポジトリをパソコンの共有フォルダとして設置して運用するケース)

4 バージョン管理の対象範囲

プロジェクトに登録されたデバイス、および各デバイスの次のデータが、バージョン管理対象となります。

- ・ マルチビューエクスプローラの「構成・設定」以下のデータ
- ・ マルチビューエクスプローラの「プログラミング」（「プログラミング」以下にデータを持つデバイスのみ）、または「HMI」以下のデータデバイスによっては、上記以外にバージョン管理対象となるデータ、上記項目のなかでバージョン管理対象とならないデータがあります。

該当するデータを持つデバイスについては、「Sysmac Studio プロジェクトバージョン管理機能 オペレーションマニュアル (SBCA-460)」を参照してください。

メインウィンドウの各ウィンドウの配置など、ウィンドウの表示設定は、バージョン管理対象とはなりません。

コントローラのユニットバージョンは、1.16 以上が対象となります。1.15 以下では、バージョン管理機能は使用できません。

5 ソフトウェアのセットアップおよび基本設定

ここでは、リモートリポジトリをパソコンの共有フォルダで共有する構成でソフトウェアのセットアップと基本設定の手順を説明します。

5.1 Sysmac Studio のインストール

Sysmac Studio Ver. 1.20 以上をインストールします。

Sysmac Studio のインストール方法については、『Sysmac Studio Version 1 オペレーションマニュアル (SBCA-470)』を参照してください。

5.2 Sysmac Studio チーム開発オプションの登録

バージョン管理機能を有効にするために、Sysmac Studio に Sysmac Studio チーム開発オプションライセンスを登録します。

チーム開発オプションライセンスの登録方法については、『Sysmac Studio プロジェクトバージョン管理機能 オペレーションマニュアル (SBCA-460)』を参照してください。

5.3 Git のインストール

1. Git を以下の URL のサイトからダウンロードします。

<https://git-scm.com/downloads>

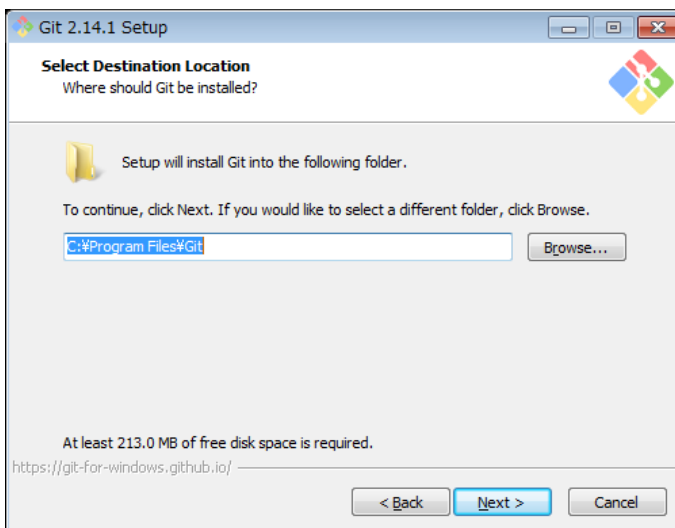
パソコンの Windows の種類に応じて、32 ビット版または 64 ビット版のインストーラをダウンロードしてください。

2. インストーラを起動します。
Windows のバージョン、ユーザの権限によっては [セキュリティの警告] ダイアログボックスが表示されます。
3. [セキュリティの警告] ダイアログボックスで [実行] ボタンをクリックします。
Windows のバージョン、ユーザの権限によっては [ユーザーアカウント制御 (User Account Control)] ダイアログボックスが表示されます。
4. [ユーザーアカウント制御] ダイアログボックスで [はい] ボタンをクリックします。
Git の Setup ダイアログボックスが表示されます。

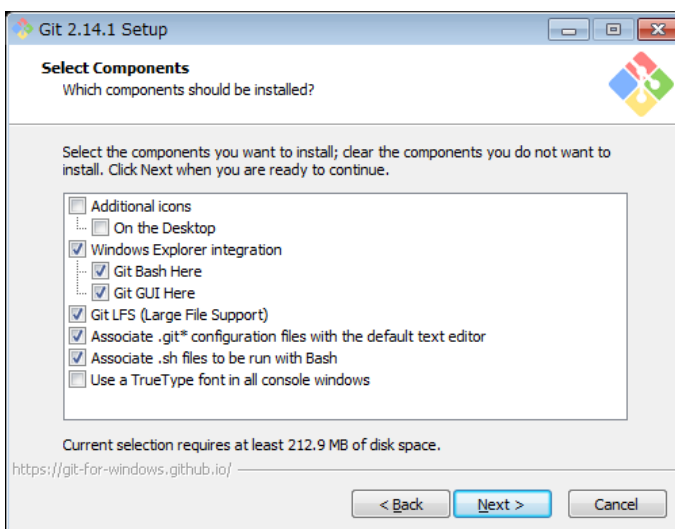
ここからは、Git 2.14.1 の画面を例に説明します。



5. [Next] ボタンをクリックします。
インストール先の選択ダイアログボックスが表示されます。

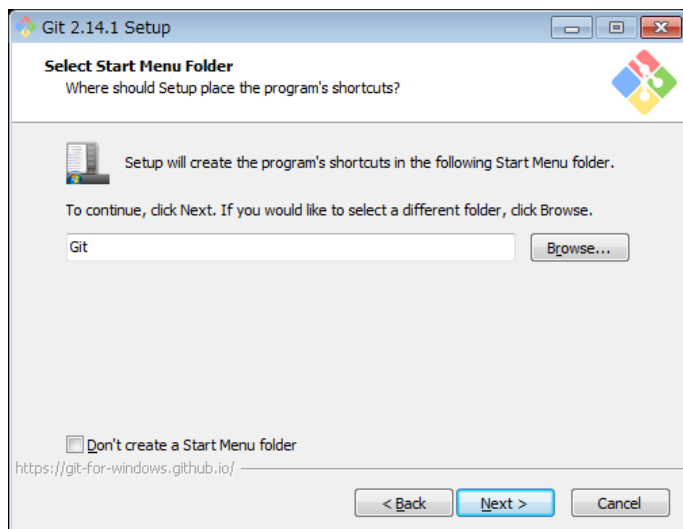


6. Git をインストールするフォルダを設定し、[Next] ボタンをクリックします。
インストールするコンポーネントを選択するダイアログボックスが表示されます。

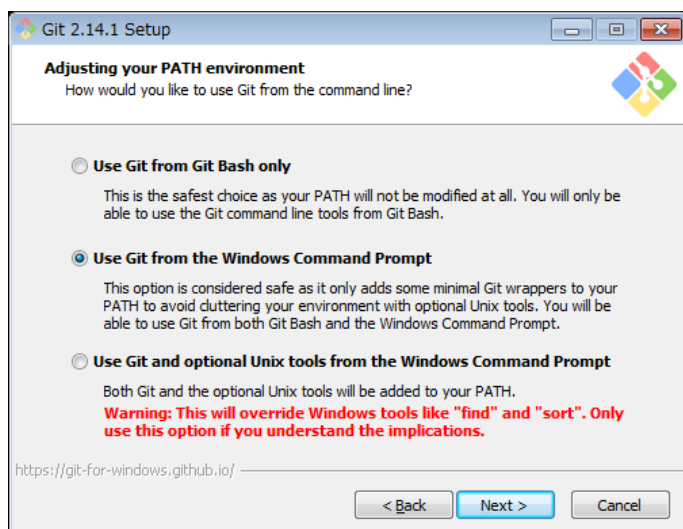


Sysmac Studio バージョン管理機能を使用するためにオプションを変更する必要はありません。

7. [Next] ボタンをクリックします。
スタートメニューの選択ダイアログボックスが表示されます。

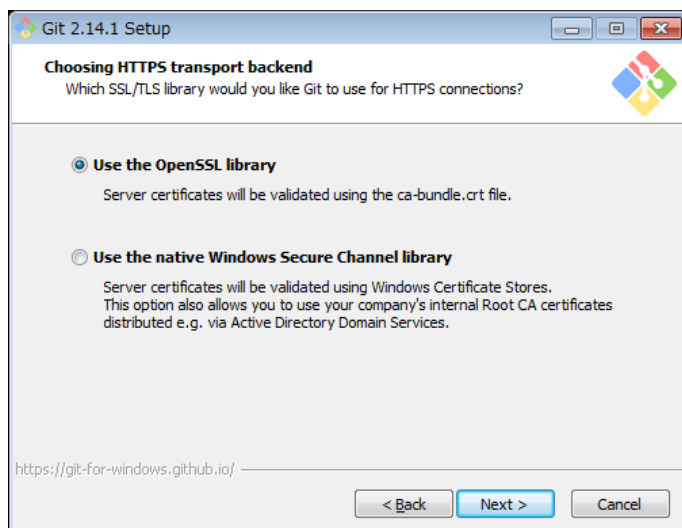


8. スタートメニューのフォルダ名を指定して [Next] ボタンをクリックします。
コマンドラインでの利用を確認するダイアログボックスが表示されます。



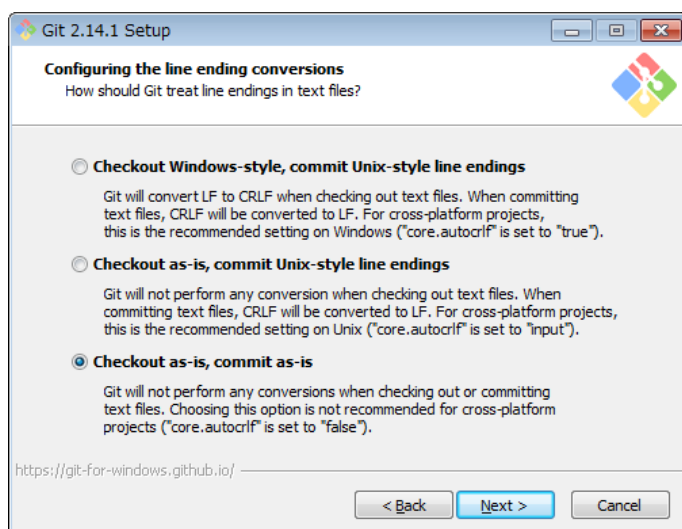
ここでは必ず 2 番目の[Use Git from the Windows Command Prompt]を選択してください。

9. [Next] ボタンをクリックします。
ユーザ認証のためのライブラリの選択ダイアログボックスが表示されます。



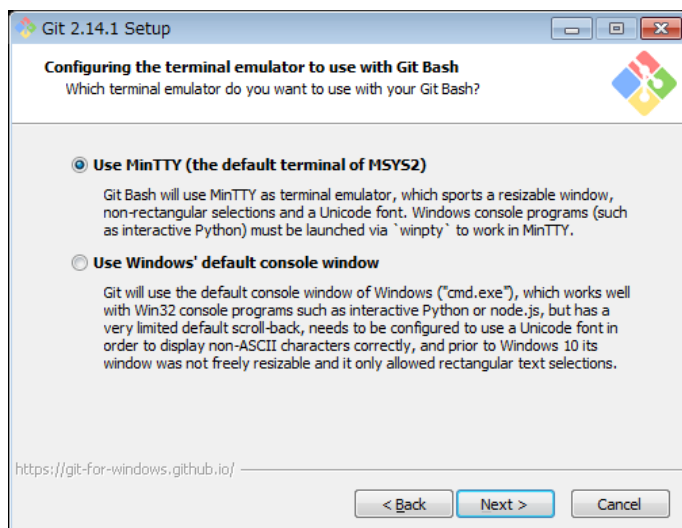
リモートリポジトリを Windows の共有で行う方式の場合はどちらの項目を選択してもかまいません。Git サーバを使用する場合は、サーバのシステム管理者に確認してください。ここでは 1 番目の [Use the OpenSSL Library] を選択します。

10. [Next] ボタンをクリックします。
テキストファイルの改行コードの選択ダイアログボックスが表示されます。



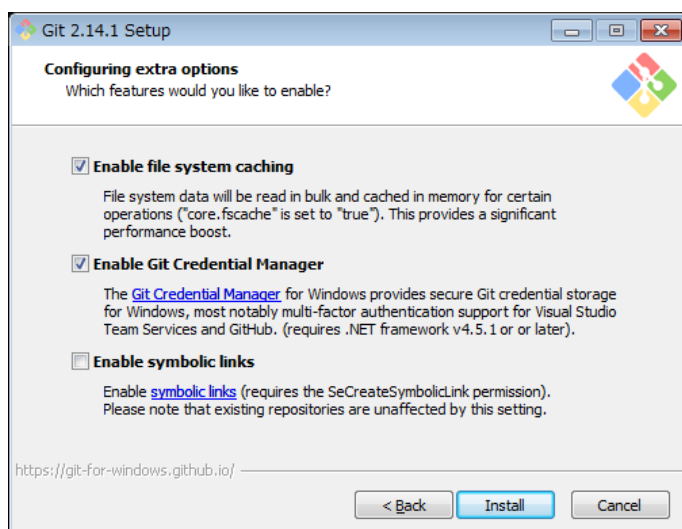
テキストファイルの改行コードの扱いを決めます。Sysmac Studio バージョン管理機能を使用するために、必ず 3 番目の [Checkout as-is, commit as-is] を選択してください。

11. [Next] ボタンをクリックします。
ターミナルソフトウェアの選択ダイアログボックスが表示されます。



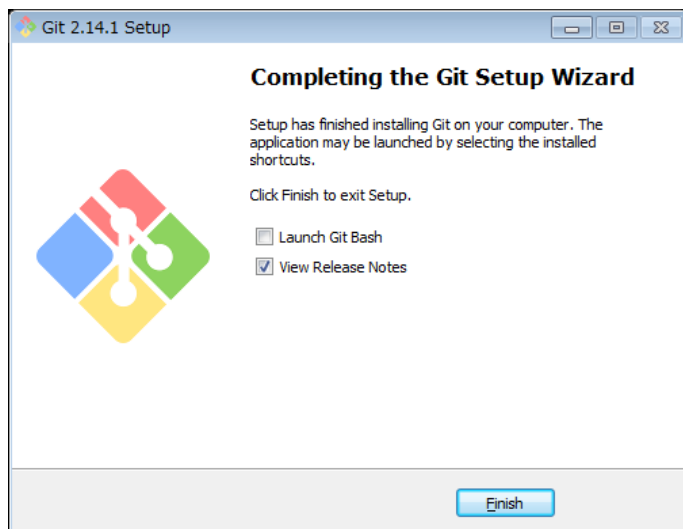
Git Bash というコマンドラインツールを使う場合のターミナルソフトウェアを選択します。Sysmac Studio バージョン管理機能を使用するためにはどのオプションでもかまいません。ここでは 1 番目の [Use MinTTY (the default terminal of MSYS2)] を選択します。

12. [Next] ボタンをクリックします。
追加オプションの選択ダイアログボックスが表示されます。



Sysmac Studio バージョン管理機能を使用するためにはどのオプションでもかまいません。ここでは 1 番目の [Enable file system caching] と 2 番目の [Enable Git Credential Manager] を選択します。

13. [Install] ボタンをクリックします。
インストール処理が行われ、完了すると次のダイアログボックスが表示されます。



14. [Finish] ボタンをクリックします。

5.4 “TortoiseGit” のインストール

1. “TortoiseGit” を以下の URL のサイトからダウンロードします。

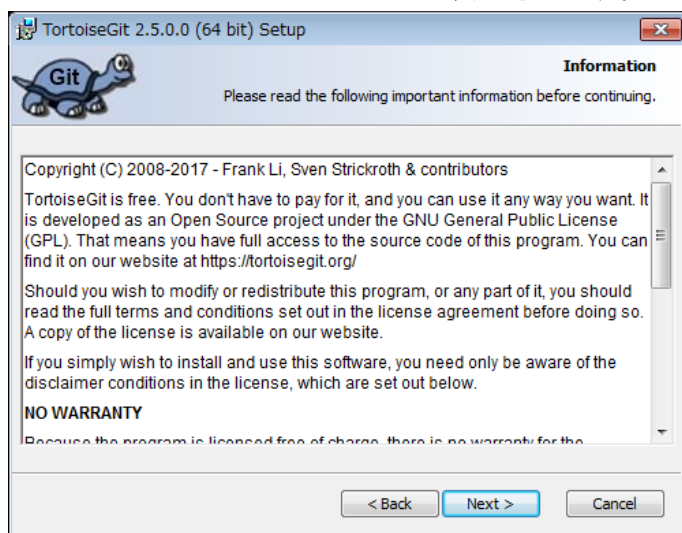
<https://tortoisegit.org/download/>

パソコンの Windows の種類に応じて、32 ビット版または 64 ビット版のインストーラをダウンロードしてください。また、同じ URL サイトに Language Packs も掲載されていますので、必要に応じて、適切な LanguagePacks をダウンロードしてください。

2. インストーラを起動します。
Windows のバージョン、ユーザの権限によっては [セキュリティの警告] ダイアログボックスが表示されず。
3. [実行] ボタンをクリックします。
“TortoiseGit” の Setup ダイアログボックスが表示されます。
ここからは、“TortoiseGit2.5.0.0” 64 ビット版を例に説明します。

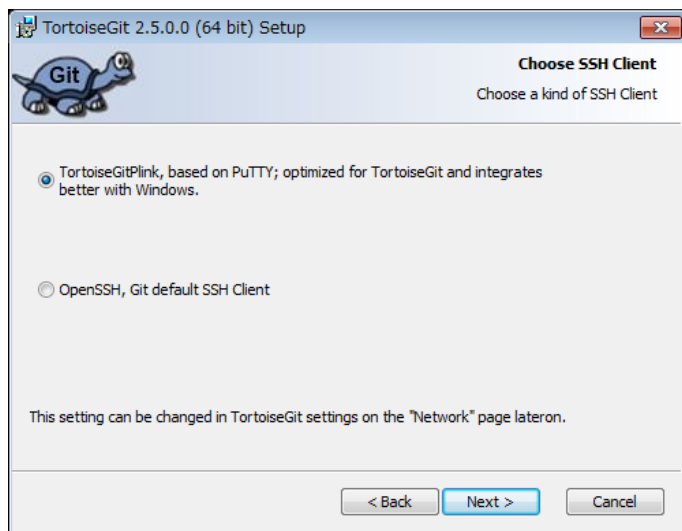


4. [Next] ボタンをクリックします。
Information ダイアログボックスが表示されます。



5. [Next] ボタンをクリックします。

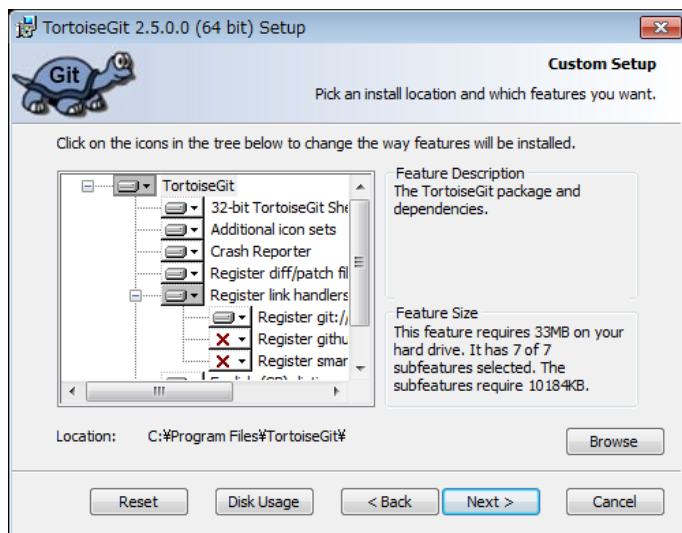
ユーザ認証のためのクライアントソフトウェアの選択ダイアログボックスが表示されます。



リモートリポジトリを Windows の共有で行う方式の場合はどちらを選択してもかまいません。Git サーバを使用する場合は、サーバのシステム管理者に確認してください。ここでは、1 番目の[TortoiseGitPlink based on PuTTY; optimized for TortoiseGit and integrates better with Windows.]を選択します。

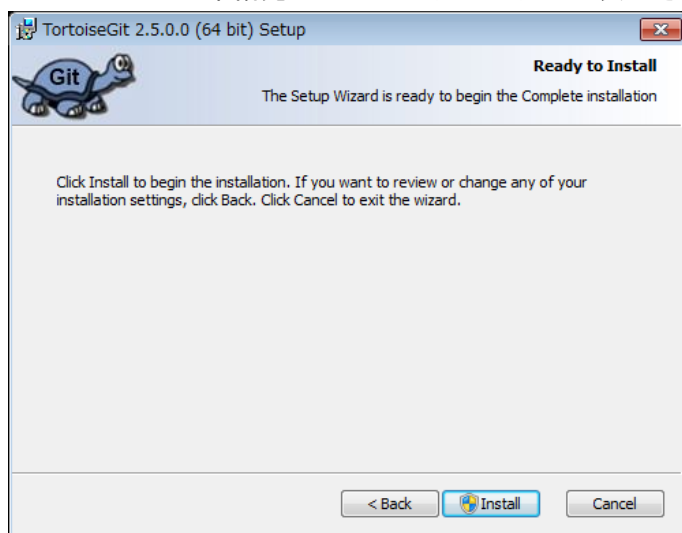
6. [Next] ボタンをクリックします。

カスタムセットアップのためのオプションダイアログボックスが表示されます。



Sysmac Studio バージョン管理機能を使用するためにはデフォルトの設定からオプションの選択を変更しないでください。

7. [Next] ボタンをクリックします。
インストールの準備完了ダイアログボックスが表示されます。



8. [Install] ボタンをクリックします。
Windows のバージョン、ユーザの権限によっては [ユーザアカウント制御] ダイアログボックスが表示されま
す。
9. [はい] ボタンをクリックします。
インストールが完了すると完了ダイアログボックスが表示されます。



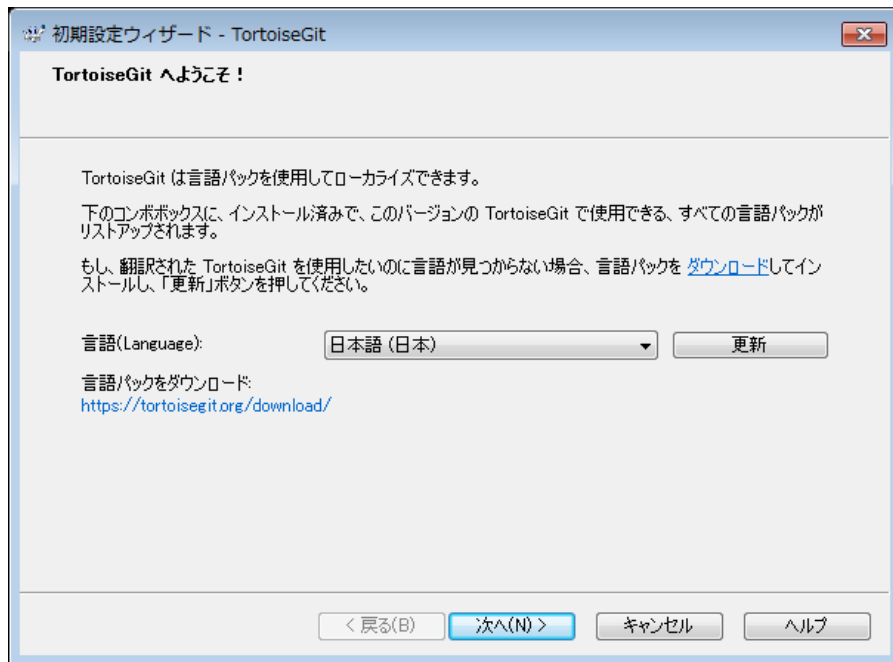
10. [Run first start wizard] のチェックをはずし、[Finish] ボタンをクリックします。
11. 必要に応じて、以下の URL のサイトから Language Packs をダウンロードし、セットアップを行ってください。

<https://tortoisegit.org/download/>

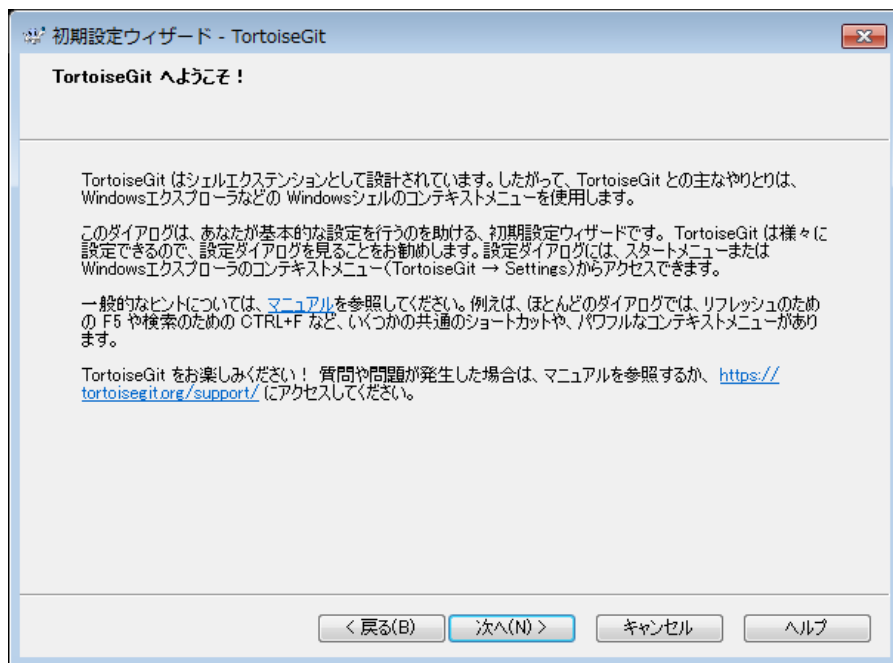
5.5 “TortoiseGit” の初期設定

“TortoiseGit” のインストールに続けて、“TortoiseGit” を使用するための初期設定を行います。

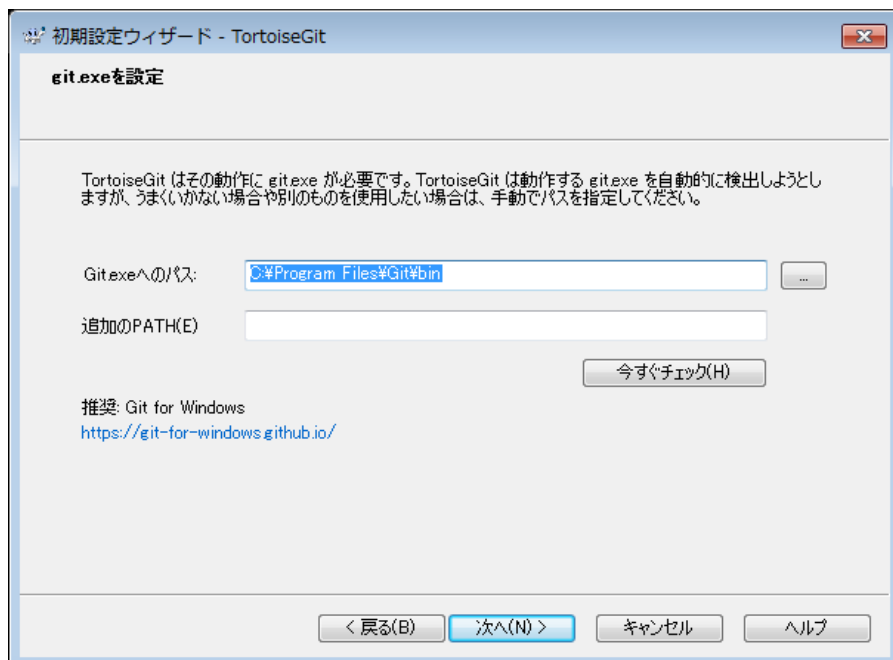
1. “TortoiseGit” の [設定] ダイアログボックスの [全般] - [初期設定ウィザードを再実施] を選択します。
[設定] ダイアログボックスは、スタートメニューから [すべてのプログラム] - [TortoiseGit] - [Settings] を選択して起動します。



2. [次へ] ボタンをクリックします。

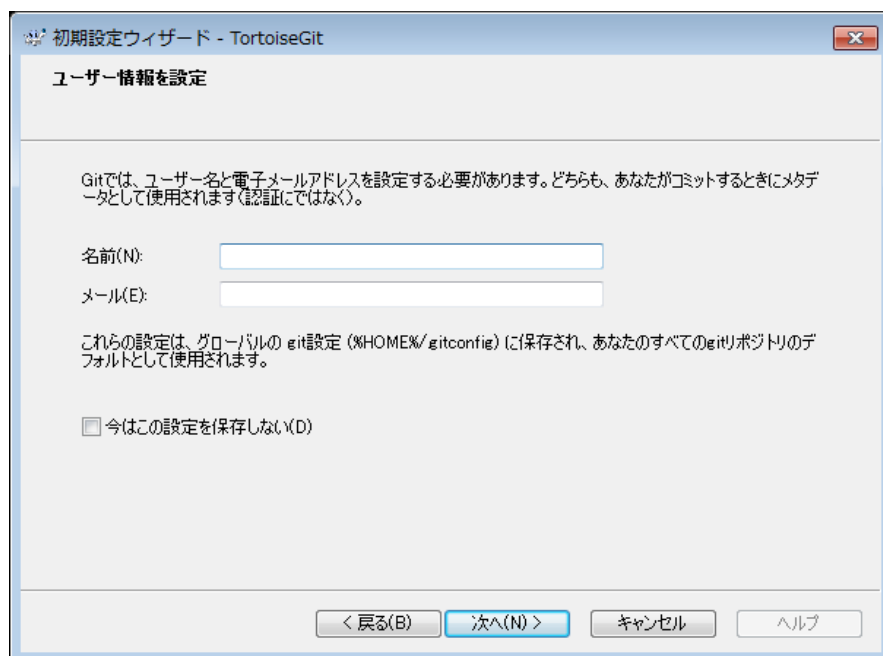


3. [次へ] ボタンをクリックします。
Git.exe のパスを設定するダイアログボックスが表示されます。



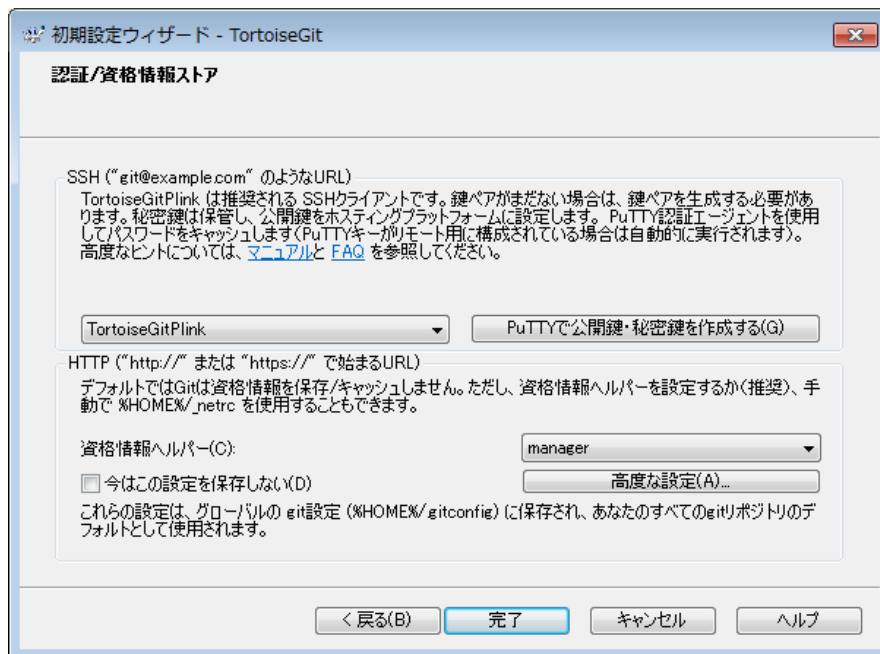
Git の実行モジュールである、Git.exe のパスを指定します。Git のインストール時にインストールパスを変更していない場合は、デフォルトのままでもかまいません。

4. 「次へ」 ボタンをクリックします。
ユーザ情報の入力ダイアログボックスが表示されます。



ユーザ名と Email アドレスは、変更履歴の情報として使用されます。

5. ユーザ名と Email アドレスを入力して「次へ」 ボタンをクリックします。
認証と資格情報の設定ダイアログボックスが表示されます。

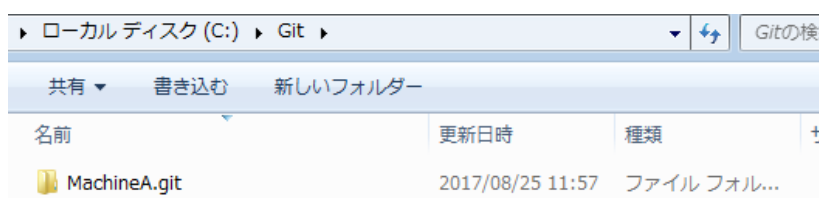


6. Sysmac Studio バージョン管理機能を使用するためにはデフォルトからオプションの選択を減らさないください。
7. [完了] ボタンをクリックします。
これで” TortoiseGit” の初期設定は完了です。

5.6 共有フォルダとリモートリポジトリの作成

本書では、リモートリポジトリをパソコンの共有フォルダで共有する方式で説明を行います。
この方式では、リモートリポジトリを配置するための共有フォルダを設定し、共有フォルダの中にリモートリポジトリとなるフォルダを作成します。

1. フォルダの作成
Windows Explorer で、新規にフォルダを作成します。
フォルダの位置、フォルダ名は任意でかまいません。ここでは C:\Git とします。
2. 共有フォルダの設定
手順1で作成したフォルダを右クリックして [プロパティ] を選択し、表示されるダイアログボックスの 共有タブにて 共有の設定を行います。
ここでは、Sysmac Studio バージョン管理機能を使用する他のユーザのパソコンからフルアクセスできるように設定します。
3. リポジトリとなるフォルダの作成
手順1で作成したフォルダの下に、新規にフォルダを作成します。
Git リポジトリとして使用するフォルダは、Git の慣例に従い、リポジトリ名の末尾に「.git」を追加した名称にします。リポジトリ名を「MachineA」とする場合、フォルダ名は「MachineA.git」とします。



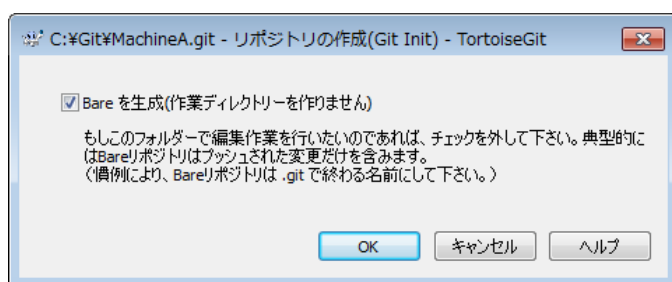
4. フォルダをリポジトリとして登録

手順 3 で作成したフォルダを右クリックして、[Git ここにリポジトリを作成] を選択します。

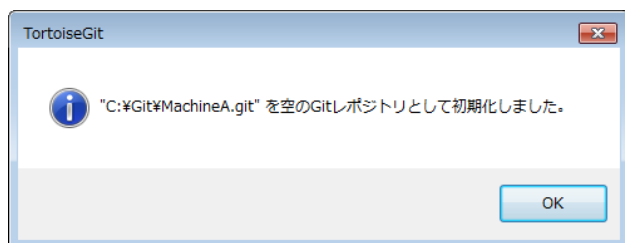


リポジトリ作成のオプションダイアログボックスが表示されます。

5. ダイアログボックスのチェックボックスにチェックを入れて [OK] ボタンをクリックします。



リポジトリが作成され、ダイアログボックスが表示されます。



6. [OK] ボタンをクリックします。

これでリモートリポジトリの作成は完了です。

ひとつのリポジトリで管理できる Sysmac Studio のプロジェクトは、ひとつです。バージョン管理するプロジェクトの数に応じて、それぞれリポジトリを作成します。

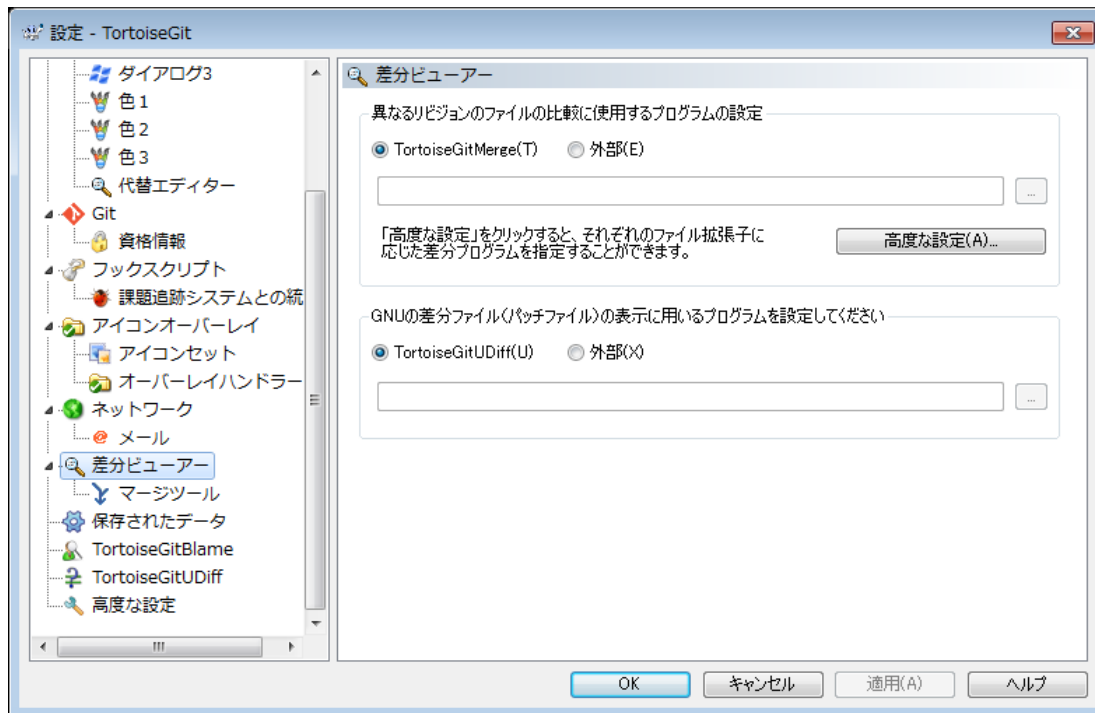
5.7 “TortoiseGit” の追加設定

“TortoiseGit” から Sysmac Studio のプロジェクトのグラフィカルな比較を行えるようにするために、“TortoiseGit” での設定を追加します。

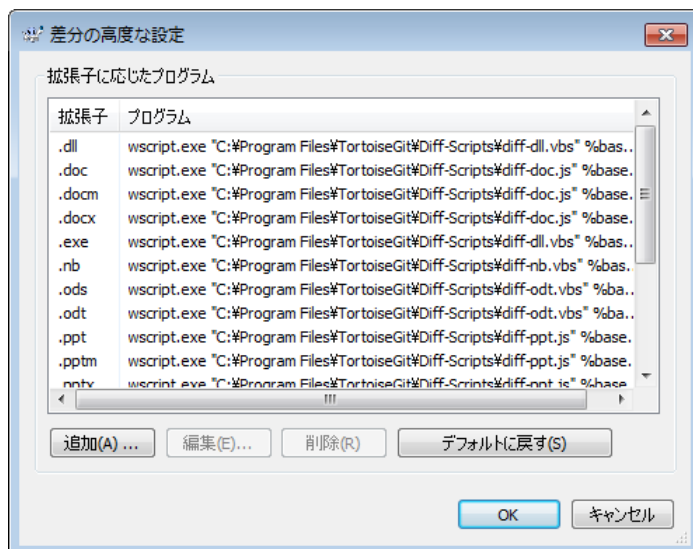
1. スタートメニューから [すべてのプログラム] -[TortoiseGit]- [Settings] を選択します。

“TortoiseGit” の [設定] ダイアログボックスが表示されます。

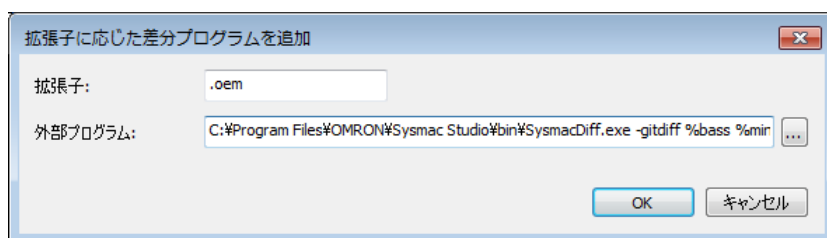
2. ツリーから「差分ビューアー」を選択します。
「差分ビューアー」画面が表示されます。



3. 「差分ビューアー」画面で、「高度な設定」ボタンをクリックします。
「差分の高度な設定」ダイアログボックスが表示されます。



4. 「追加」ボタンをクリックします。
「拡張子に応じた差分プログラムを追加」ダイアログボックスが表示されます。



5. 次の文字列を入力し、[OK] ボタンをクリックします。

項目	入力する文字列
Extension	.oem
External Program	(Sysmac Studio のインストールフォルダ*1) ¥SysmacDiff.exe -gitdiff %base %mine %bpath %brev %yrev

*1 Sysmac Studio のインストールフォルダはデフォルトでは次のとおりです。

Sysmac Studio 32 ビット版の場合

Windows 32bit 版の場合 : C:¥Program Files¥OMRON¥Sysmac Studio

Windows 64bit 版の場合 : C:¥Program Files (x86)¥OMRON¥Sysmac Studio

Sysmac Studio 64 ビット版の場合

Windows 64bit 版の場合 : C:¥Program Files¥OMRON¥Sysmac Studio

6. [差分の高度な設定] ダイアログボックスで [OK] ボタンをクリックします。

7. [設定] ダイアログボックスで [OK] ボタンをクリックします。

これで” TortoiseGit” の追加設定は完了です。

6 SYSMAC STUDIO プロジェクトの履歴管理

6.1 バージョン管理機能を使用した履歴管理機能の概要

Sysmac Studio と連携するバージョン管理システム「Git」には、次の 4 つの基本機能があります。

- ・ 保存したソフトウェアデータの変更を一意に特定できる識別番号を自動生成し、コメントをつけて保存できる
- ・ 日付情報、コメント、ユーザ名などで特定のソフトウェアデータの変更を検索できる
- ・ 任意のソフトウェアデータとの差分を把握（比較）できる
- ・ 特定の変更時点のソフトウェアデータに復元できる

これらの基本機能を使って、Sysmac Studio プロジェクトデータの履歴管理を行います。

From

マスタへの変更反映の度に、プロジェクトファイルをコピー&名前変更する手間が発生。いつ誰が何を変更したのか、すぐにわからない。

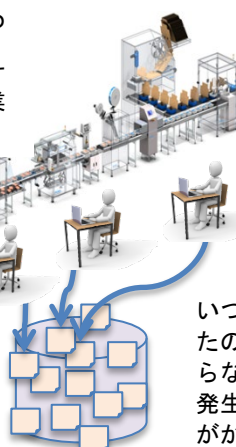


変更箇所を 1 つずつコピー&ペーストで、マスタに反映するので、マージ作業に時間がかかる。

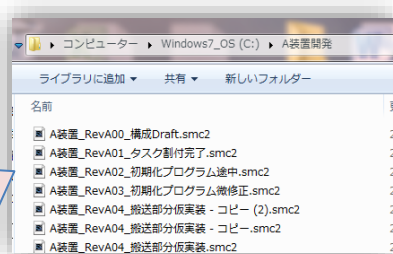


マスタプロジェクトにバージョンを付けるために、マスタへの変更反映の度に、プロジェクトファイルをコピー&名前変更する手間が発生。

プロジェクトファイルを複製して管理するものの、ファイル名だけでは管理しきれない。

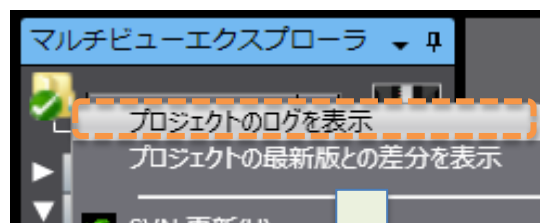


いつ誰が何を変更したのか、すぐにわからないため、不具合発生時の対処に時間がかかる。



To

バージョン管理機能により、プロジェクトの変更履歴を確実に管理できます。



ログを開くと

Revision	Actions	Author	Date	Message
9		010080037	2017年3月10日 7:30:46	搬送仮実装
8		010080037	2017年3月10日 7:29:28	初期化プログラム微修正
7		010080037	2017年3月10日 7:29:02	初期化プログラム途中
6		010080037	2017年3月10日 7:28:22	タスク割付完了
5		010080037	2017年3月10日 7:27:36	装置作成

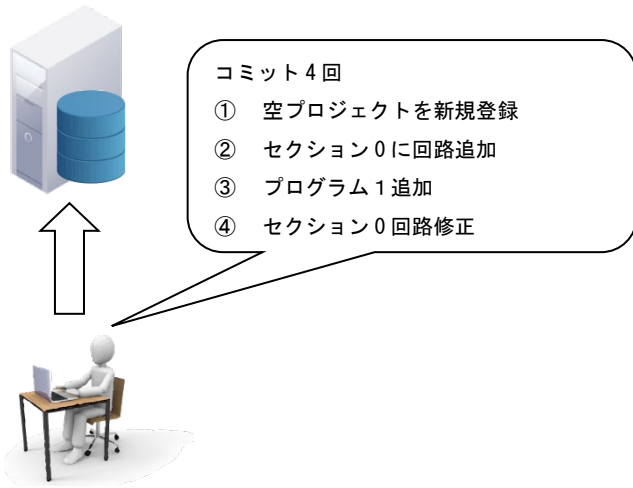
変更の履歴の確認

- ・ 昔との比較
- ・ 特定のバージョンに戻す
- ・ コメントの検索 etc

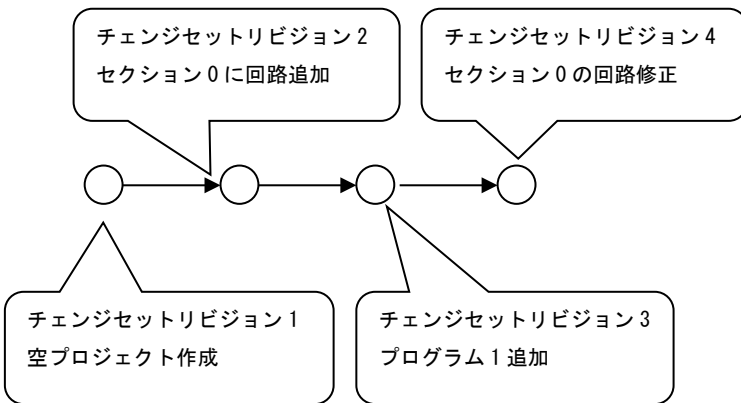
6.2 履歴管理機能の実行例

履歴管理機能を使用する際の操作を説明します。

例：次のように4回のコミットを行い、リポジトリの情報を更新する場合



これをバージョン管理システムの管理単位であるチェンジセットで表現すると、以下のように、起点となるリビジョン1(*)のチェンジセットのあとに、リビジョン2→リビジョン3→リビジョン4(*)と変更履歴情報がつながって管理されていることになります。特定のリビジョン間の差分情報をチェンジセット情報を合成して管理しているため、Sysmac Studioで開くプロジェクトを特定のリビジョンに戻すことができます。



*リビジョンは説明の簡略化のために1, 2, 3, 4と表現していますが、実際はSHA-1 アルゴリズムで生成される16進数40桁の数値であり、時系列に沿って値が一つずつ増えていくものではありません。

6.3 プロジェクト管理の開始準備（ベースとなるプロジェクトの作成、保存）

Sysmac Studio プロジェクトをバージョン管理を開始するために、ベースとなるプロジェクトの作成、ローカルリポジトリへの保存、およびローカルリポジトリとリモートリポジトリの同期設定を行います。

6.3.1 ベースとなるプロジェクトの作成

1. スタートメニューから [すべてのプログラム] - [OMRON] - [Sysmac Studio] - [Sysmac Studio] を選択して、Sysmac Studio を起動します。
Windows のバージョン、ユーザの権限によっては [ユーザーアカウント制御 (User Account Control)] ダイアログボックスが示されます。[はい] ボタンをクリックします。

2. ベースとなるプロジェクトを作成、登録します。

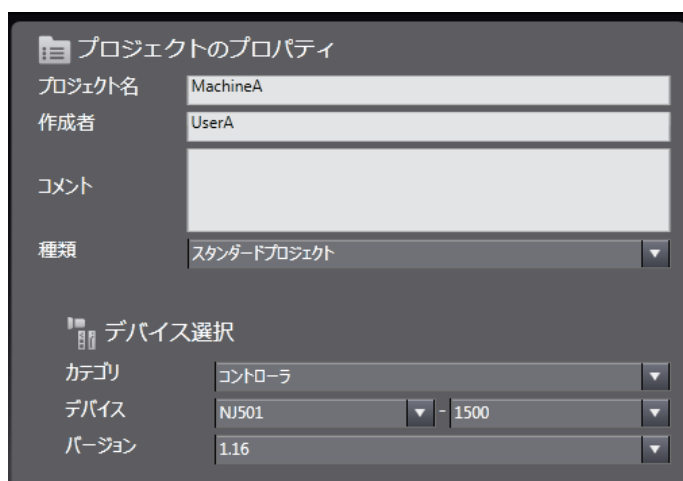
ここでは、新規プロジェクトを作成し、リポジトリに登録する手順を示します。作成済のプロジェクトに登録する場合も同様の手順で行います。

Sysmac Studio のスタートページで「新規プロジェクト」を選択し、必要事項を設定して「作成」ボタンをクリックします。

本説明では、例として次のように設定します。

項目	入力／選択値
プロジェクト名	MachineA
作成者	UserA
種類	スタンダードプロジェクト
カテゴリ	コントローラ
デバイス	NJ501-1500
バージョン	1.16 ^{*1}

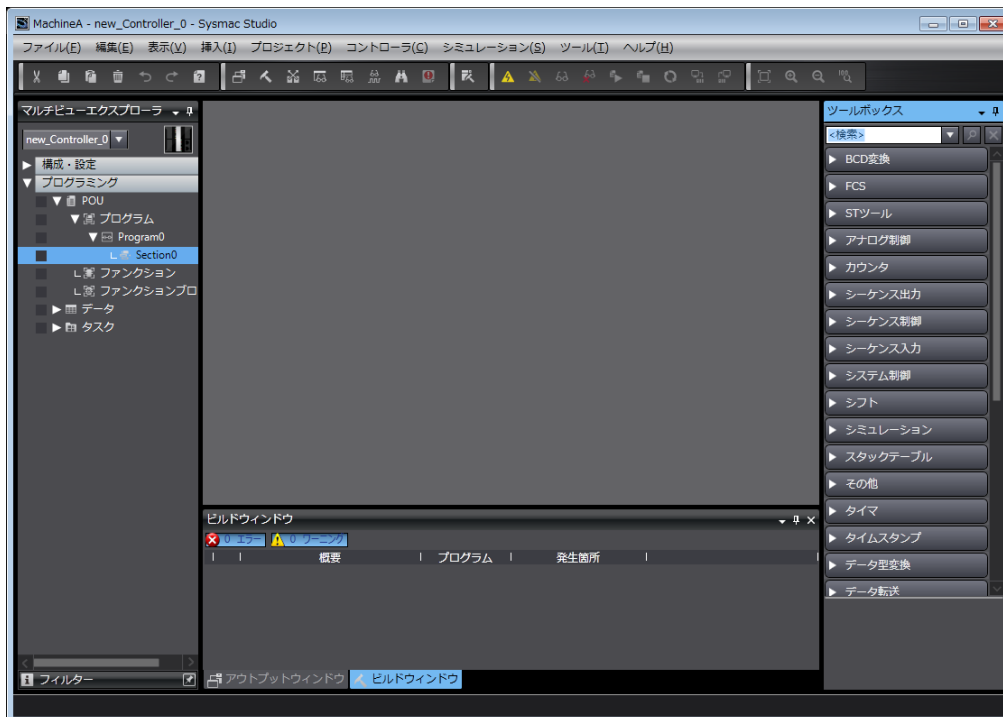
*1 カテゴリがコントローラの場合、Version 1.16 以上を選択してください。1.15 以下では、バージョン管理機能は使用できません。



The screenshot shows the 'プロジェクトのプロパティ' (Project Properties) window in Sysmac Studio. It contains the following fields and values:

- プロジェクト名 (Project Name): MachineA
- 作成者 (Creator): UserA
- コメント (Comment): (Empty text area)
- 種類 (Type): スタンダードプロジェクト (Standard Project)
- デバイス選択 (Device Selection):
 - カテゴリ (Category): コントローラ (Controller)
 - デバイス (Device): NJ501 - 1500
 - バージョン (Version): 1.16

プロジェクト名 MachineA のプロジェクトが開きます。

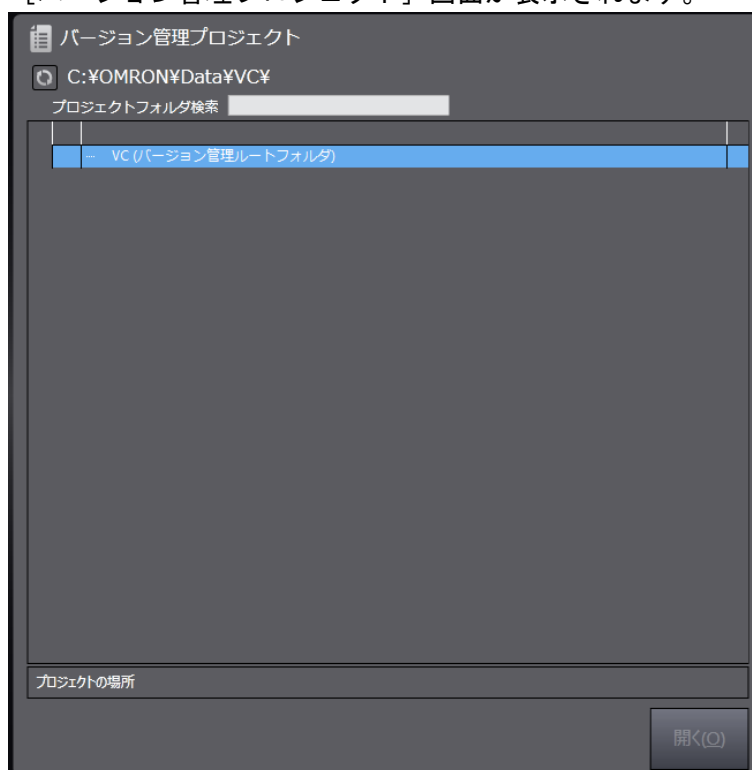


3. メインメニューから「ファイル」－「保存」を選択してプロジェクトを保存します。

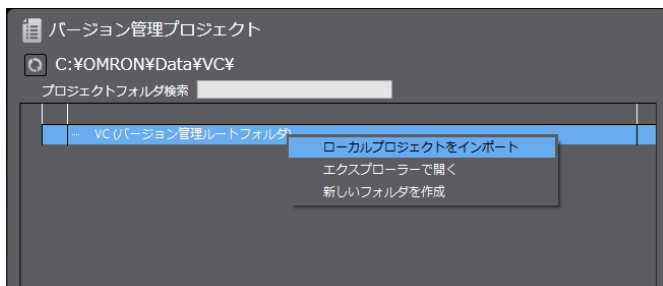
4. 「ファイル」－「閉じる」を選択してプロジェクトを閉じます。

6.3.2 ローカルリポジトリへの登録

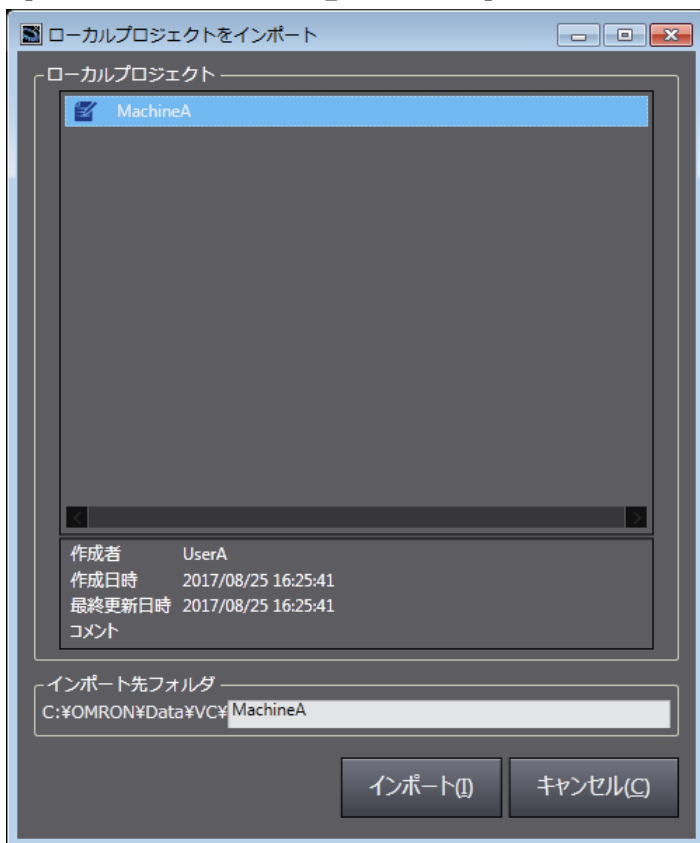
1. Sysmac Studio のスタートページで「バージョン管理エクスプローラ」をクリックします。
「バージョン管理プロジェクト」画面が表示されます。



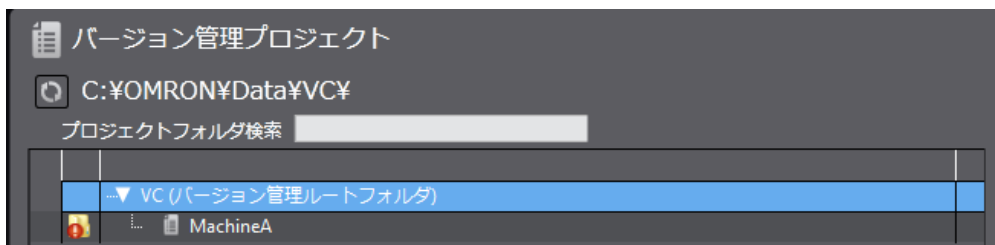
2. 作業フォルダ[VC (バージョン管理ルートフォルダ)]を右クリックして、[ローカルプロジェクトをインポート]を選択します。



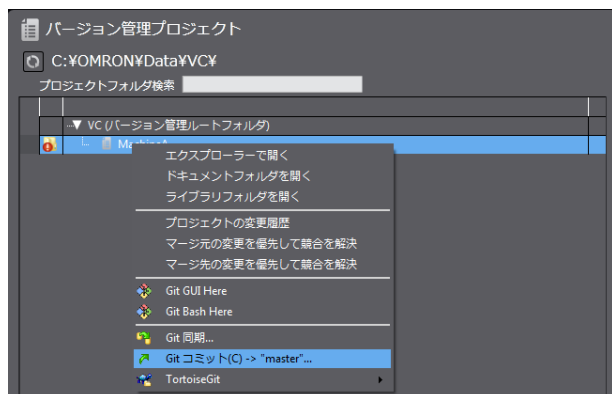
[ローカルプロジェクトをインポート] ダイアログボックスが表示されます。



3. 作成したプロジェクトを選択し、[インポート] ボタンをクリックします。
選択されたプロジェクトがバージョン管理の作業フォルダにコピーされます。

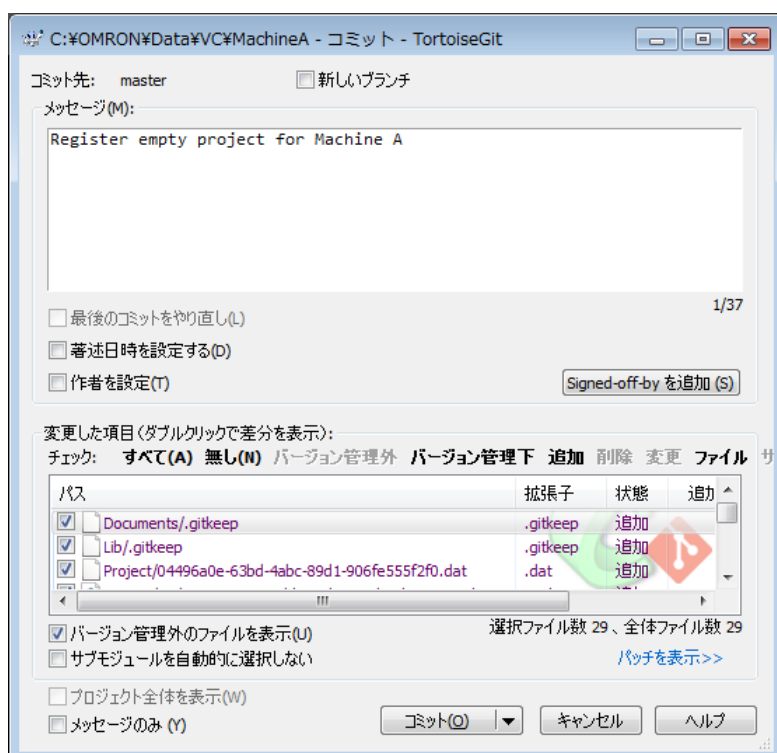


4. コピーされたプロジェクトを右クリックして、[Git コミット -> "master"] を選択します。

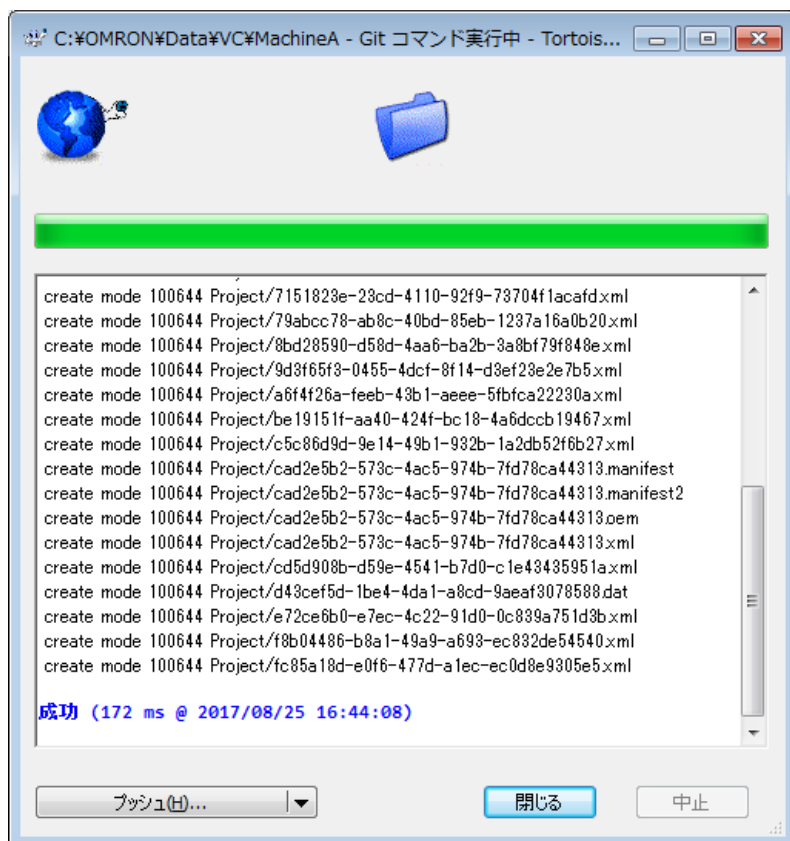


コミット操作時の注意を促すダイアログボックスが表示されます。
(以降、本書の手順説明では本ダイアログボックスの表示は省略します。)

5. 内容を確認し、[次回から表示しない] チェックボックスにチェックをいれて [OK] ボタンをクリックします。
Tortoise のコミットダイアログボックスが表示されます。
6. メッセージエリアに記録として残したいメッセージを入力し、[コミット] ボタンをクリックします。



コミット完了 ダイアログボックスが表示されます。



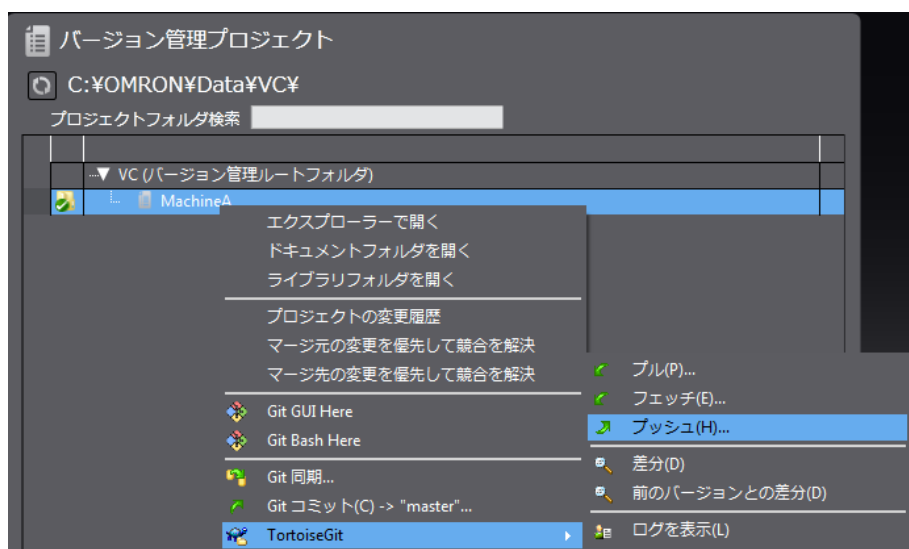
これでパソコン内のプロジェクトがローカルリポジトリに登録されました。

7. [閉じる] ボタンを押して終了します。

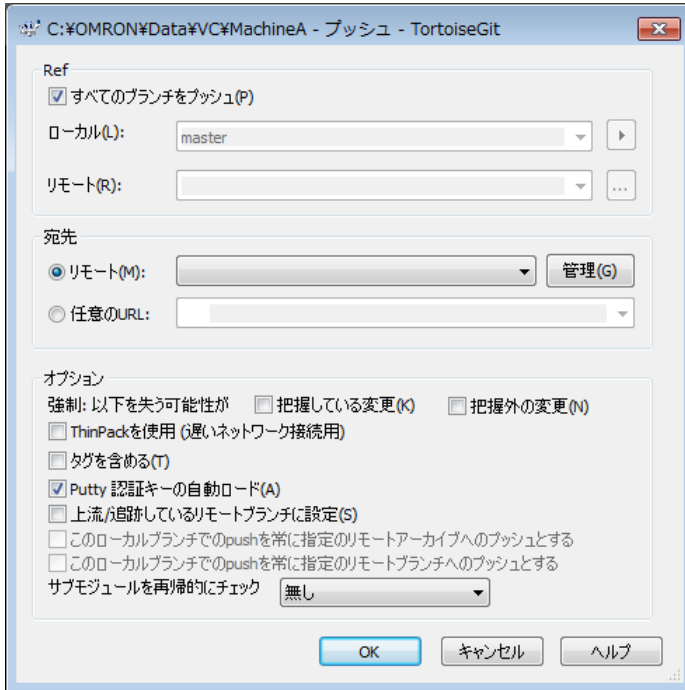
6.3.3 ローカルリポジトリとリモートリポジトリの同期設定

ローカルリポジトリからリモートリポジトリへバージョン管理対象のプロジェクトのデータをプッシュします。

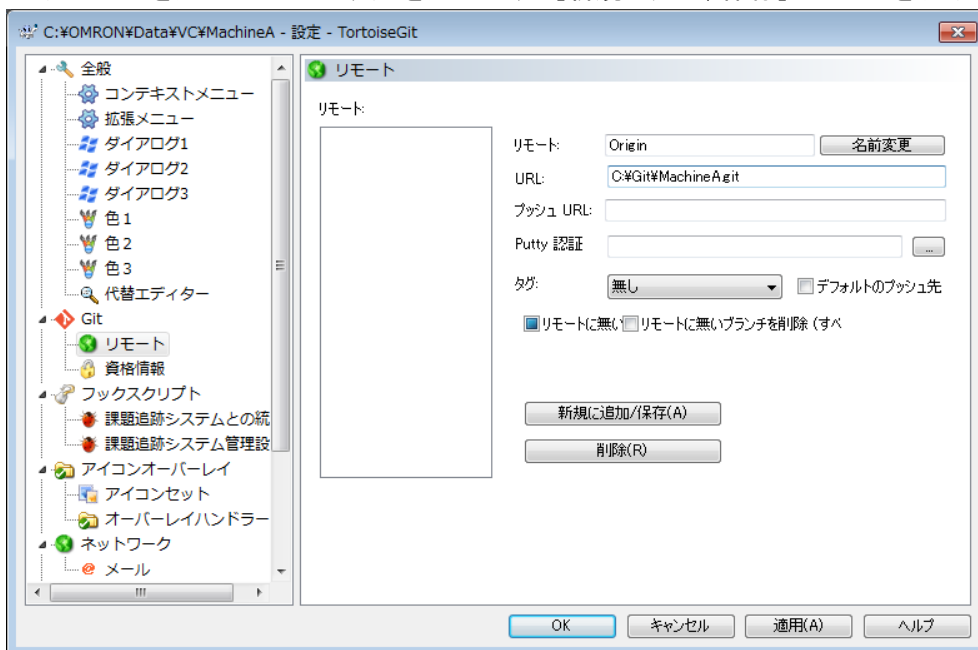
1. [バージョン管理プロジェクト] 画面で、作成したフォルダを右クリックして、[TortoiseGit]- [プッシュ] を選択します。



“TortoiseGit” のプッシュ ダイアログボックスが表示されます。



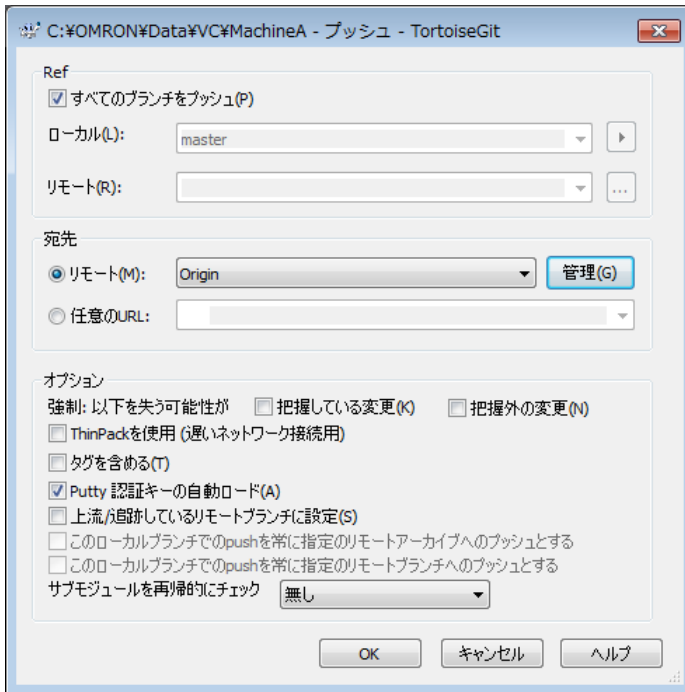
2. 「すべてのブランチをプッシュ」チェックボックスにチェックを入れ、[宛先] の項目の [管理] ボタンをクリックします。
リポジトリ「MachineA」のリモートリポジトリの場所を設定するためのダイアログボックスが表示されます。
3. [リモート] 項目にこの接続の名前（通常は「Origin」）と、URL（ここでは 5.6 項で作成したリモートリポジトリのパスを入力しています）を入力し、[新規に追加/保存] ボタンをクリックします。



リモート名「Origin」が登録されます。

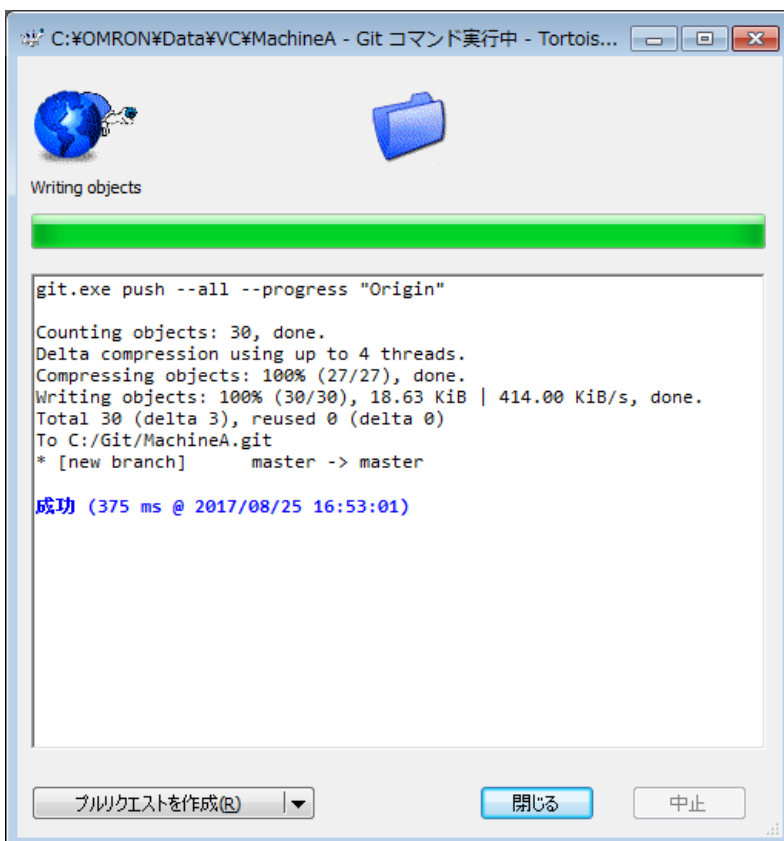
4. [OK] ボタンをクリックします。
プッシュ操作時の注意を促すダイアログボックスが表示されます。
(以降、本書の手順説明では本ダイアログボックスの表示は省略します。)
5. 内容を確認し、[このメッセージを再び表示しない] チェックボックスにチェックを入れて [はい] ボタンをクリックします。

プッシュダイアログボックスに「Origin」という接続がセットされます。



6. [OK] ボタンをクリックします。

ローカルリポジトリのプロジェクト MachineA が、リモートリポジトリの MachineA.git にコピーされました。



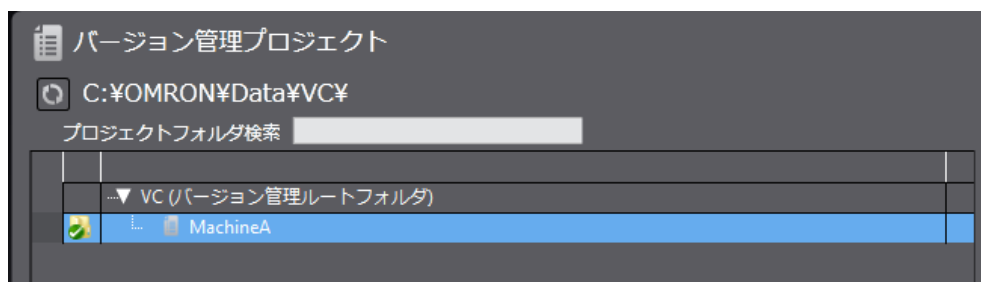
7. [閉じる] ボタンをクリックします。

6.4 プロジェクトの更新

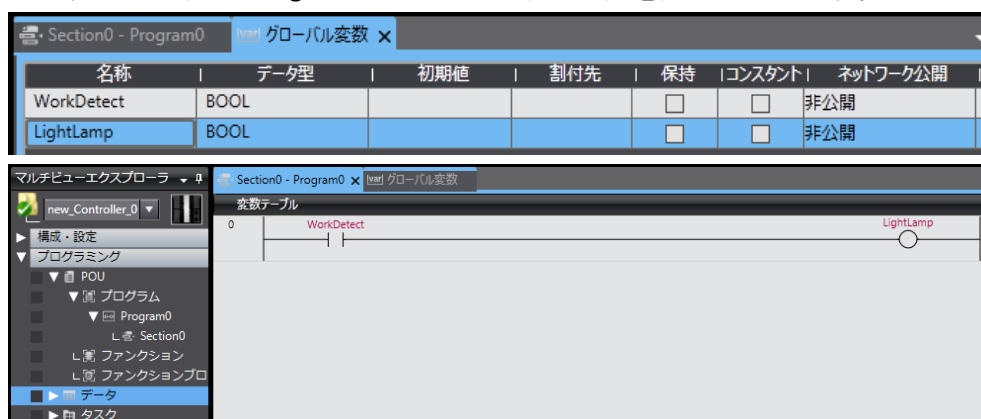
ここでは、プロジェクトのデータの変更を行い、その変更をリポジトリに反映する手順を示します。

6.4.1 プロジェクトのデータの変更

1. Sysmac Studio のスタートページで [バージョン管理エクスプローラ] を選択し、[バージョン管理プロジェクト] 画面から編集したいプロジェクトを選択して [開く] ボタンをクリックします。



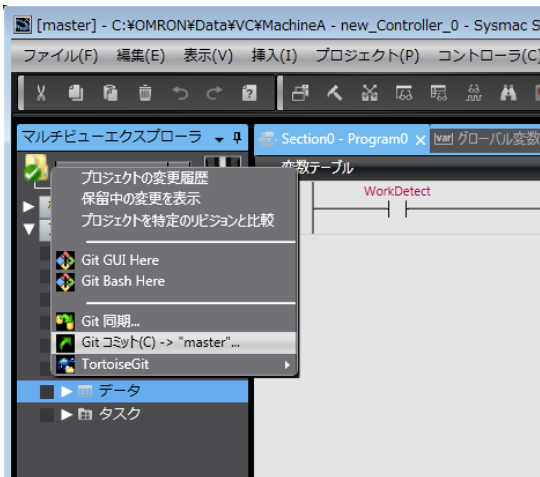
2. プロジェクトのデータを変更します。ここではグローバル変数に BOOL 型変数の WorkDetect と LightLamp を追加し、プログラム Program0-Section0 にラダー図を追加しています。



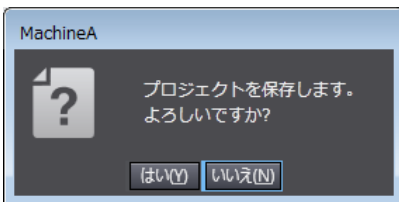
6.4.2 変更内容のローカルリポジトリへの登録（コミット）

プロジェクトの変更内容をローカルリポジトリに反映します。これをコミットと呼びます。

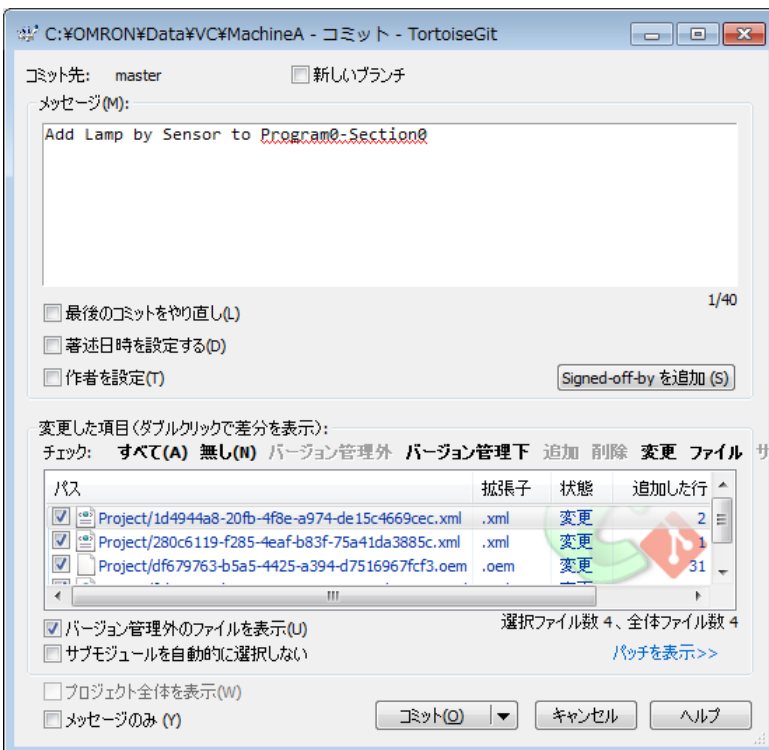
1. マルチビューエクスプローラのフォルダアイコンを右クリックして [Git コミット-> “master”] を選択します。



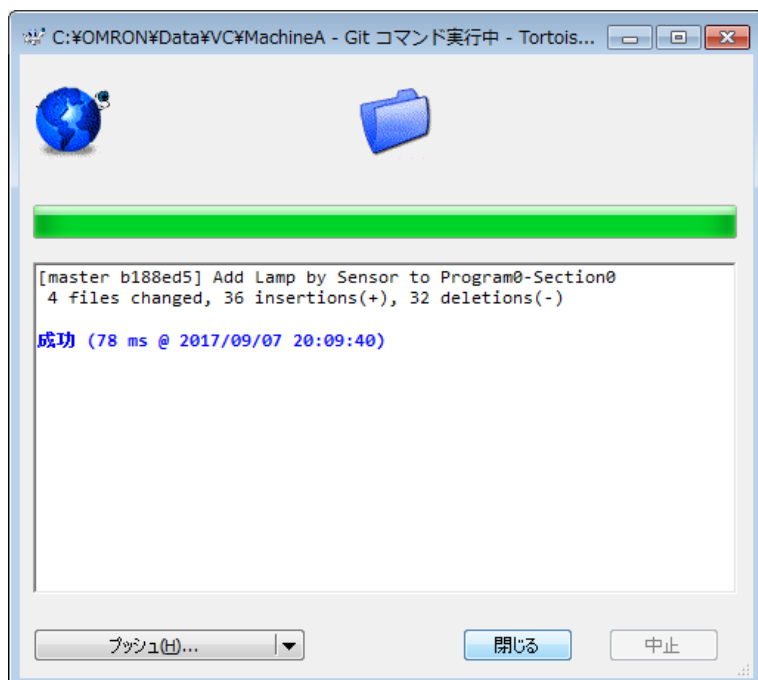
プロジェクトの保存確認ダイアログボックスが表示されます。



2. [はい] ボタンをクリックします。
“TortoiseGit” コミットダイアログボックスが表示されます。

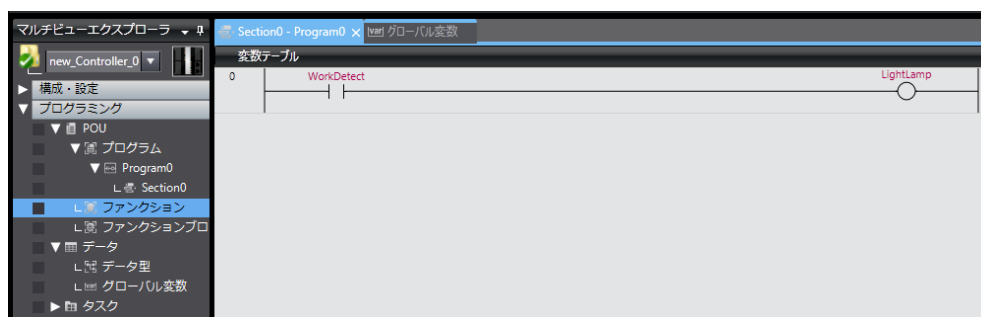


3. メッセージ エリアに記録として残したいメッセージを入力し、[コミット] ボタンをクリックします。
コミット完了 ダイアログボックスが表示されます。



これでパソコン内のプロジェクトの変更がローカルリポジトリに反映されました。

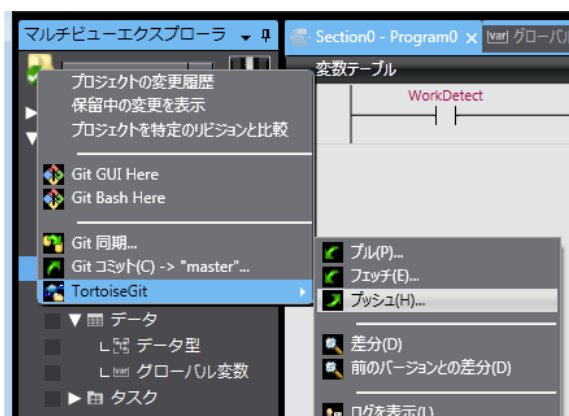
4. [閉じる] ボタンをクリックします。
編集していたプロジェクトが表示されます。



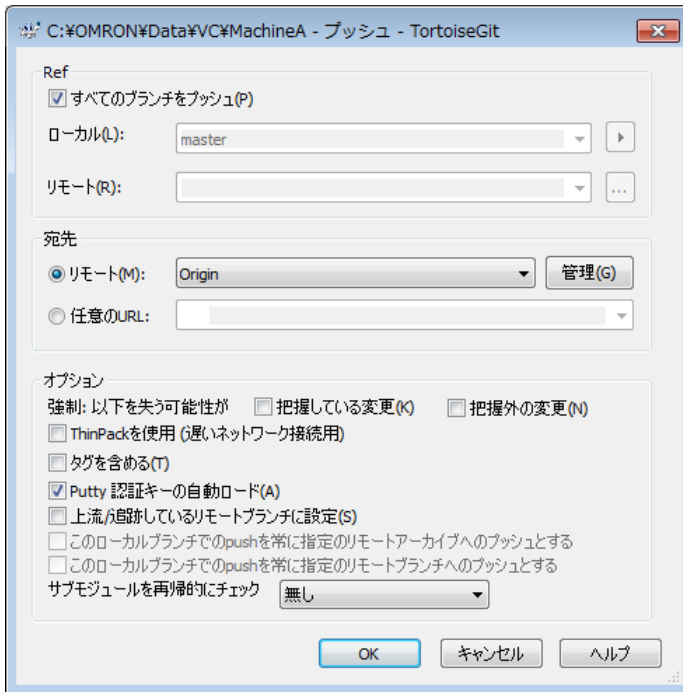
6.4.3 変更内容のリモートリポジトリへの登録（プッシュ）

ローカルリポジトリに登録したプロジェクトの変更内容を、リモートリポジトリに反映します。これをプッシュと呼びます。

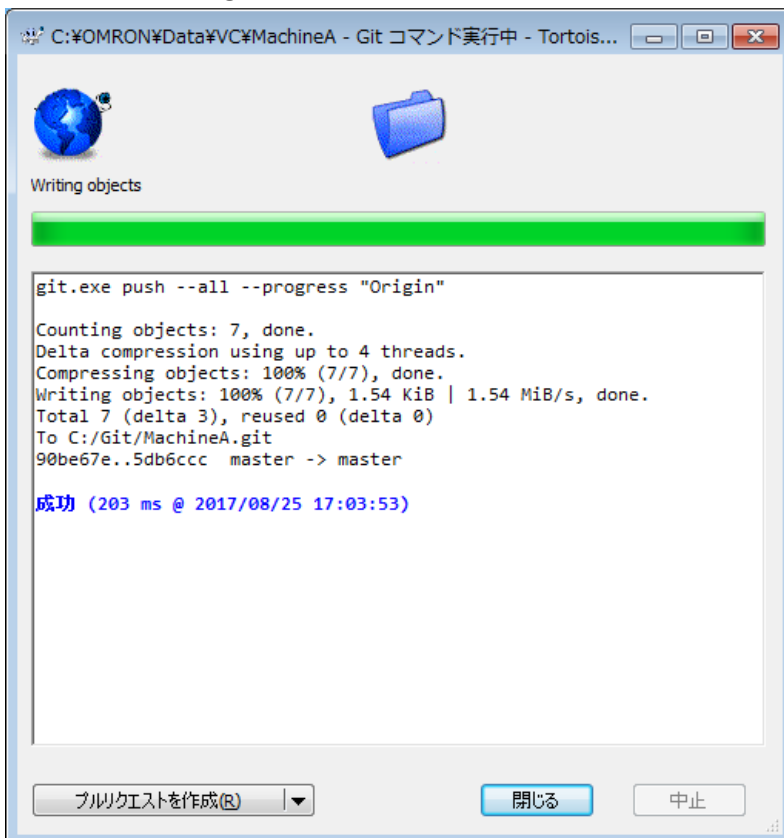
1. マルチビューエクスプローラのフォルダアイコンを右クリックして、[TortoiseGit]-[プッシュ] を選択します。



プッシュダイアログボックスが表示されます。



2. 「すべてのブランチをプッシュ」のチェックボックスにチェックが入っており、「リモート」項目がリモート名（ここではOrigin）になっていることを確認して、「[OK]」ボタンをクリックします。



これでプッシュ処理は完了です。

3. 「閉じる」ボタンをクリックします。

6.5 プロジェクトの検索

プロジェクト編集時に変更を誤ったために、変更前のプロジェクトに戻して編集を再開したいことがあります。このような場合、過去のプロジェクトの変更内容を日付や変更者、変更履歴情報を確認したり、プロジェクトを開いて内容を確認したりして、戻るべきプロジェクトのリビジョンを特定します。

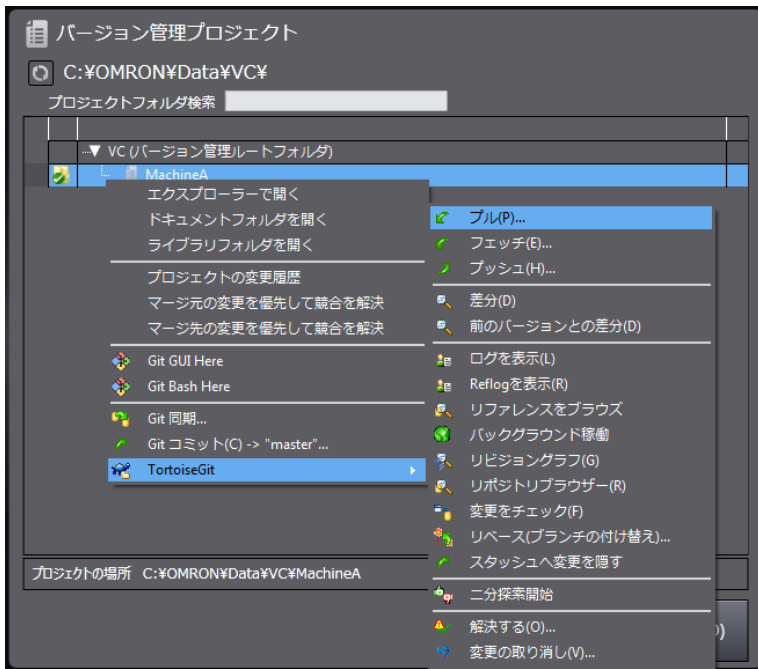
バージョン管理機能を使用すると、次の手順で過去のプロジェクトのリビジョンを検索することができます。

6.5.1 ローカルリポジトリの更新

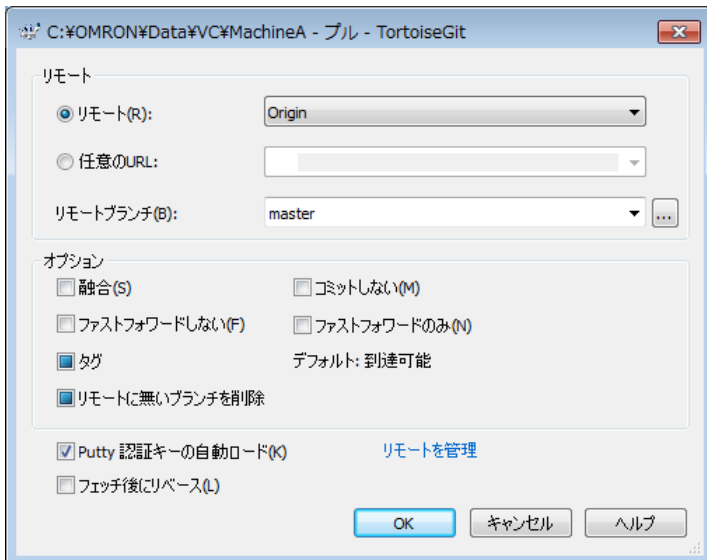
リモートリポジトリをマスタとする運用の場合は、作業をしているパソコンのローカルリポジトリを最新にする必要があります。

ここでは、「6.4 プロジェクトの更新」と同様の手順でいくつかの変更を行い、ローカルリポジトリとリモートリポジトリを更新したあとを想定しています。

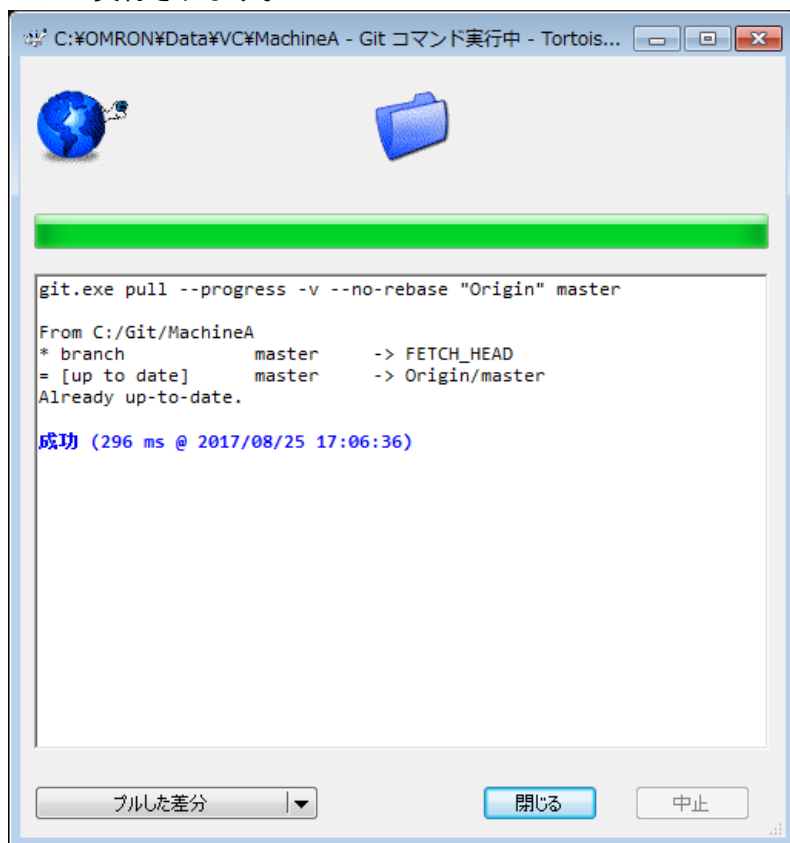
1. Sysmac Studio のスタートページの [バージョン管理プロジェクト] から目的のプロジェクトを右クリックして、[TortoiseGit]-[プル] を選択します。



“TortoiseGit” のプルダイアログボックスが表示されます。



2. 「リモート」項目が、登録していたリモート名（ここでは「Origin」）になっていることを確認して、[OK] ボタンをクリックします。
プルが実行されます。

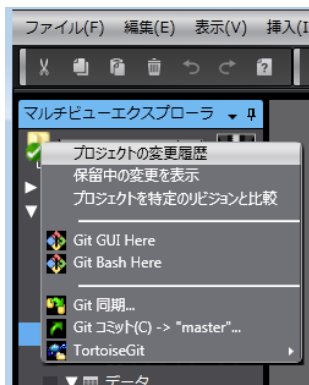


3. [閉じる] ボタンを押します。

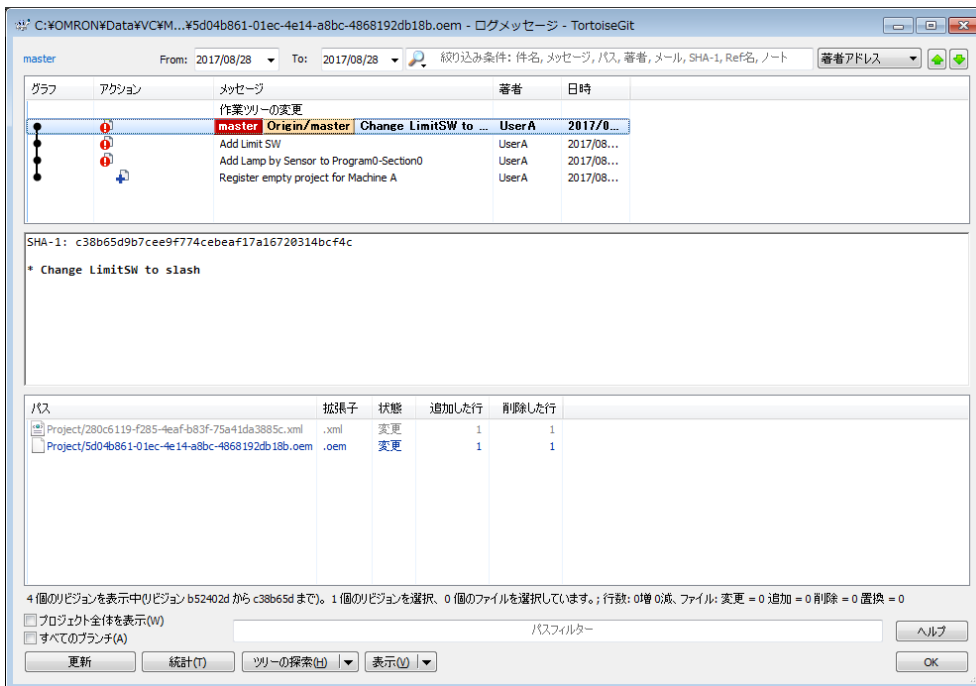
6.5.2 プロジェクトのログの表示

対象のプロジェクトの変更履歴情報を表示します。

1. Sysmac Studio のスタートページで [バージョン管理エクスプローラ] を選択し、[バージョン管理プロジェクト] 画面から対象のプロジェクトを選択して [開く] ボタンをクリックします。
プロジェクトが開きます。
2. マルチビューエクスプローラのフォルダアイコンを右クリックして、[プロジェクトの変更履歴] を選択します。

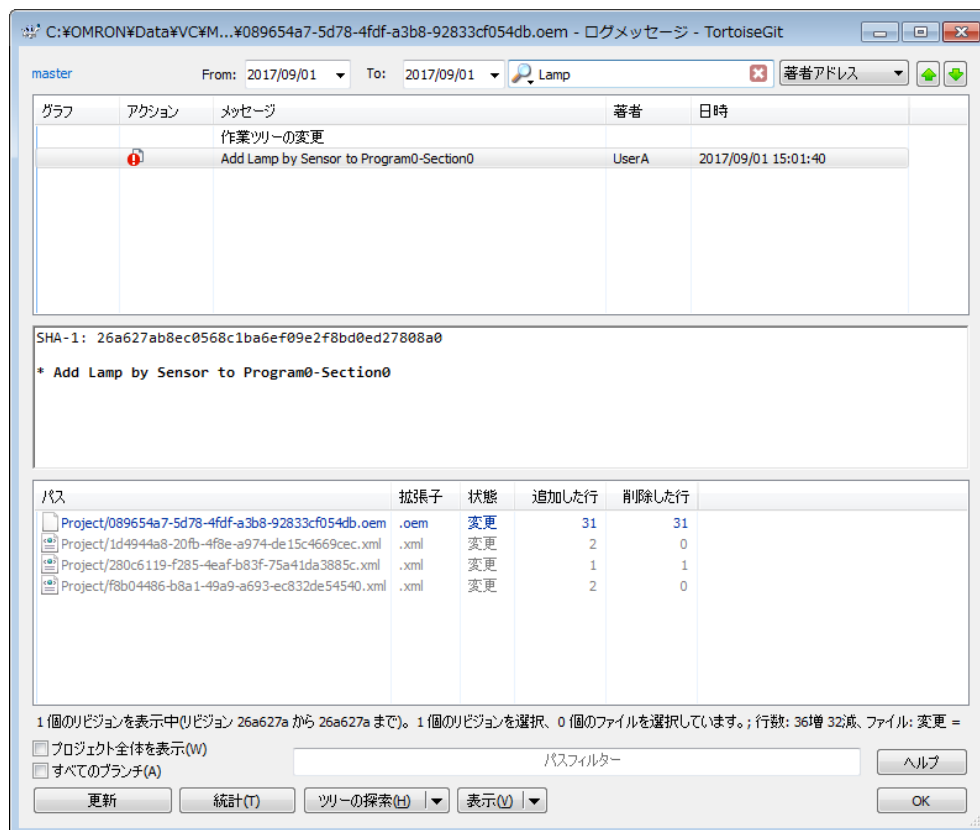


“TortoiseGit” のログメッセージダイアログボックスが表示されます。
上段の表示エリアに、コミットごとにアクション（変更、追加、削除など）、メッセージ（コミット時に入力したログメッセージ）、著者（ユーザ名）、日時が一覧で表示されます。



6.5.3 リビジョンの絞り込み

ログの最上段の検索フィールドに、絞り込みの条件となるキーワードを入力します。
たとえば、Lamp にかかわる変更を行ったりビジョンを特定するために、「Lamp」と入力すると、Lamp を含むメッセージを持つリビジョンのみが一覧に表示されます。

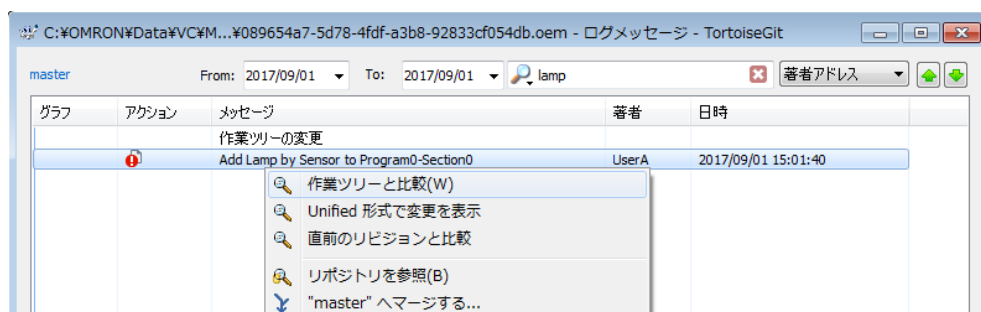


このようにメッセージの文字列で絞り込んだり、日付やユーザ名で絞り込み・ソートを行い、目的のリビジョンを絞り込むことができます。

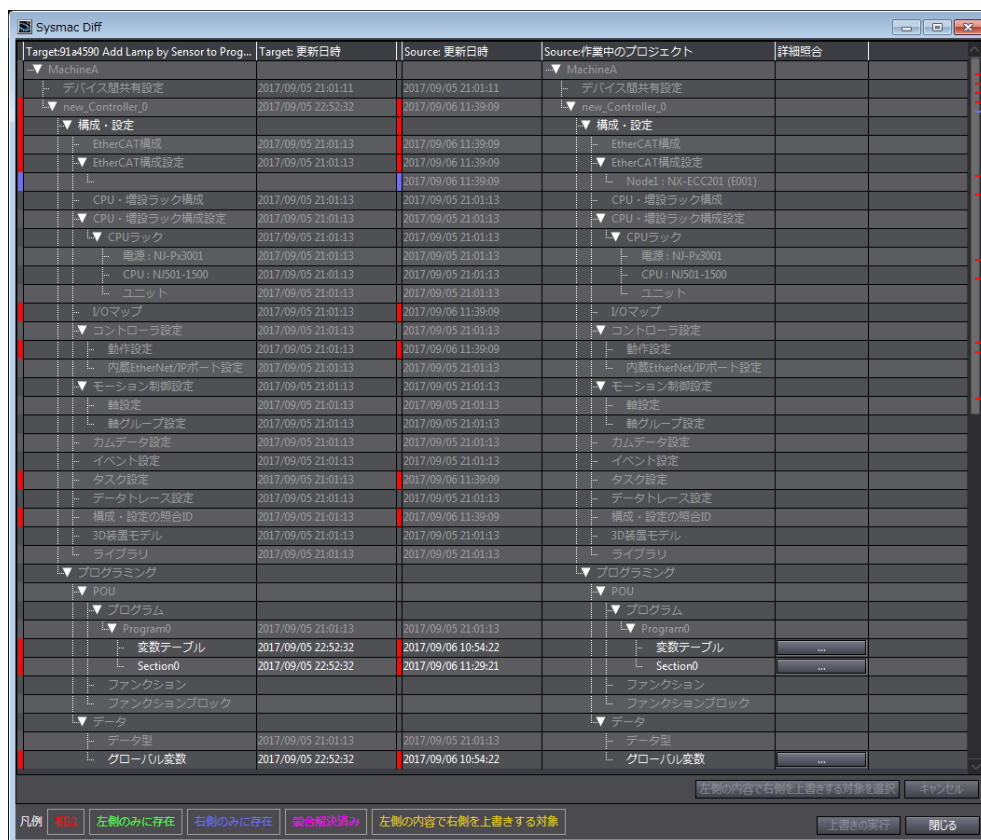
6.6 プロジェクトの比較

絞り込まれたリビジョンと開いているプロジェクトを比較します。

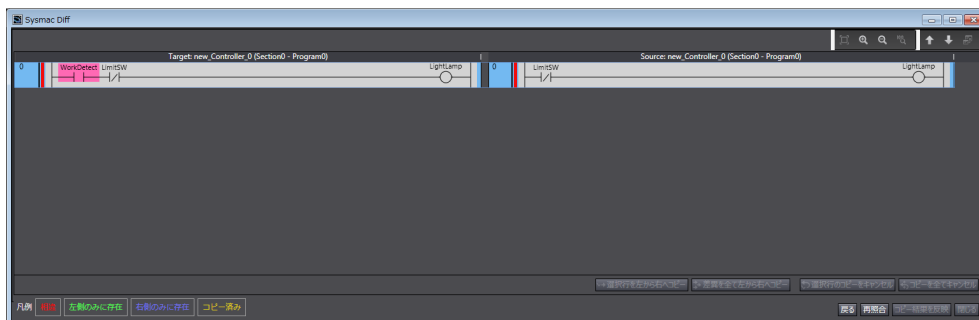
1. “TortoiseGit” のログメッセージダイアログボックスで対象のリビジョンを右クリックして、[作業ツリーと比較] を選択します。



開いているプロジェクトと、選択したリビジョンのプロジェクトの比較ダイアログボックスが表示されます。赤色になっている個所が差異がある個所です。



2. 詳細を参照するために「詳細照合」列の「…」ボタンをクリックします。
ラダー図での差異が表示され、リビジョン間の違いを把握できます。

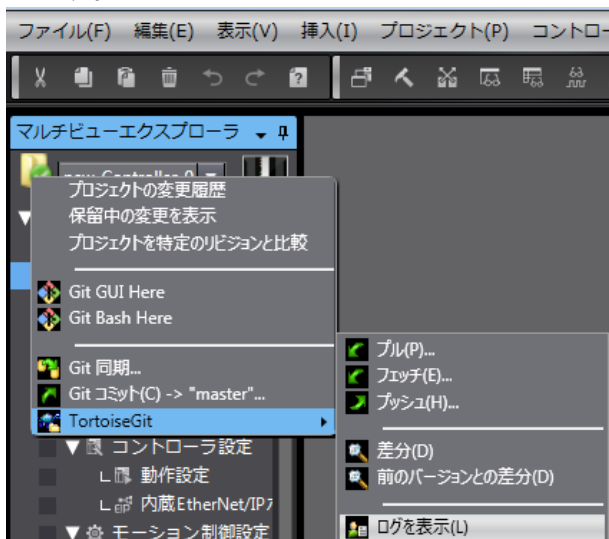


3. 差異を確認して、「閉じる」ボタンを押します。

6.7 プロジェクトの復元

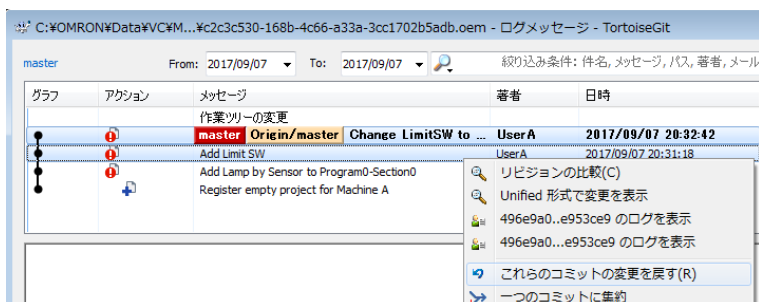
絞り込んだリビジョンのプロジェクトを開き、プロジェクトの復元を行います。

1. マルチビューエクスプローラのフォルダアイコンを右クリックして、「TortoiseGit」-「ログを表示」を選択します。

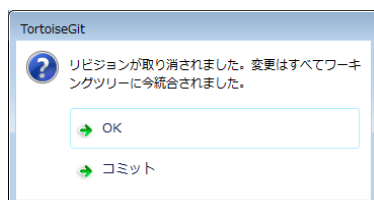


“TortoiseGit”のログメッセージダイアログボックスが表示されます。

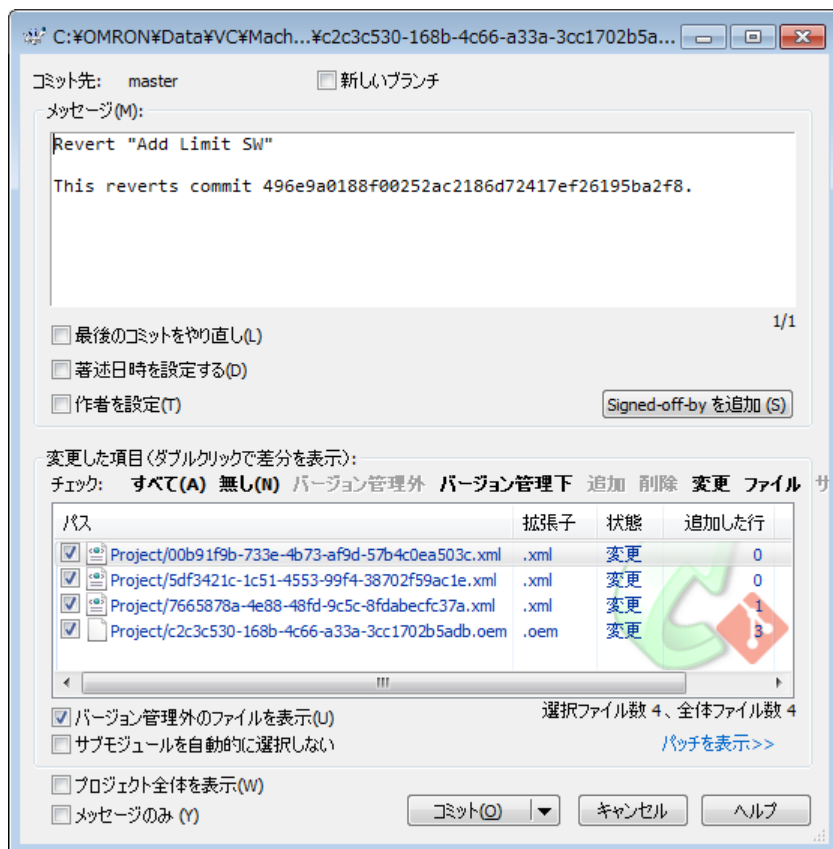
2. 戻したいリビジョンまで Shift キーを押しながら行を選択し、右クリックして、「これらのコミットの変更を戻す」を選択します。



“TortoiseGit”の結果ダイアログボックスが表示されます。

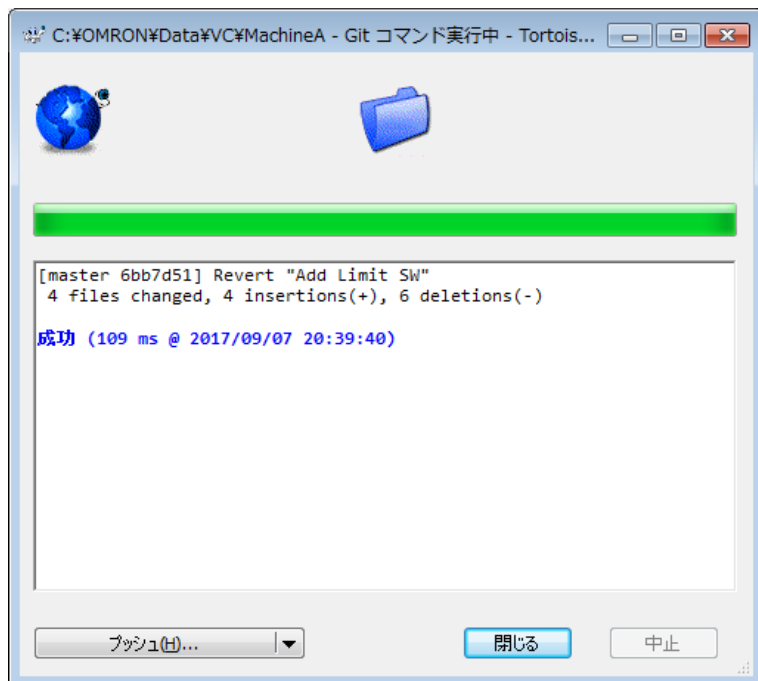


3. [コミット] ボタンをクリックします。
“TortoiseGit” の [コミット] ダイアログボックスが表示されます。



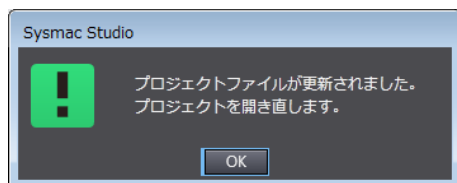
[メッセージ] には、復元処理のコメントが自動的に入ります。

4. [変更した項目] のすべてのファイルにチェックが入っていることを確認して、[コミット] ボタンをクリックします。
コミット処理の完了ダイアログボックスが表示されます。



5. [閉じる] ボタンをクリックし、つづけて [ログメッセージ] ダイアログボックスで [OK] ボタンをクリックします。

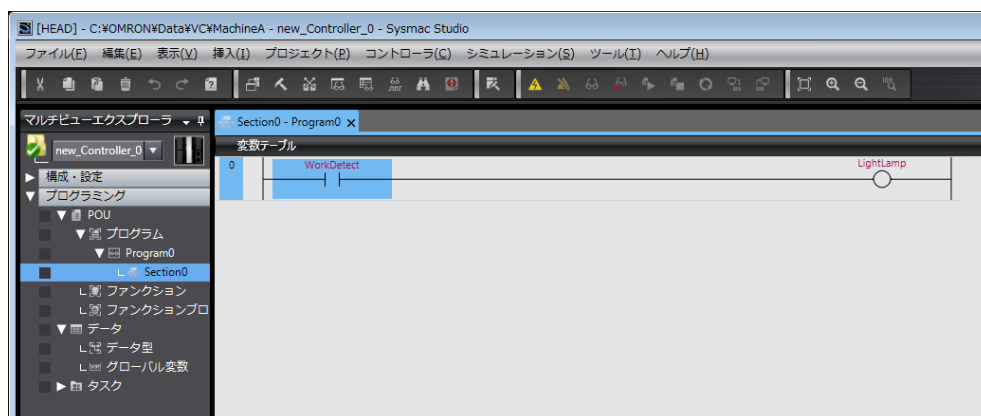
確認ダイアログボックスが表示されます。



6. [OK] ボタンをクリックします。

プロジェクトが保存され、復元されたプロジェクトが開きます。

以下のように指定したリビジョンのプロジェクトに戻っています。



7 SYSMAC STUDIO プロジェクトの複数人開発と派生開発

コントローラシステムの開発期間の短縮のために、複数人で並行して開発を行うことがあります。この開発方法を複数人開発と呼びます。また、生産装置のバリエーションの増加に伴い、装置のハードウェア構成が異なる複数の派生装置のプログラムを共通化することがあります。この開発方法を派生開発と呼びます。

この章では、バージョン管理機能を使用した Sysmac Studio プロジェクトの複数人開発および派生開発について説明します。

- ・ バージョン管理機能を使用した複数人開発および派生開発の概要
- ・ 複数人開発および派生開発を行うために必要な「ブランチ」および「マージ」
- ・ 複数人開発と派生開発を行う際の操作手順

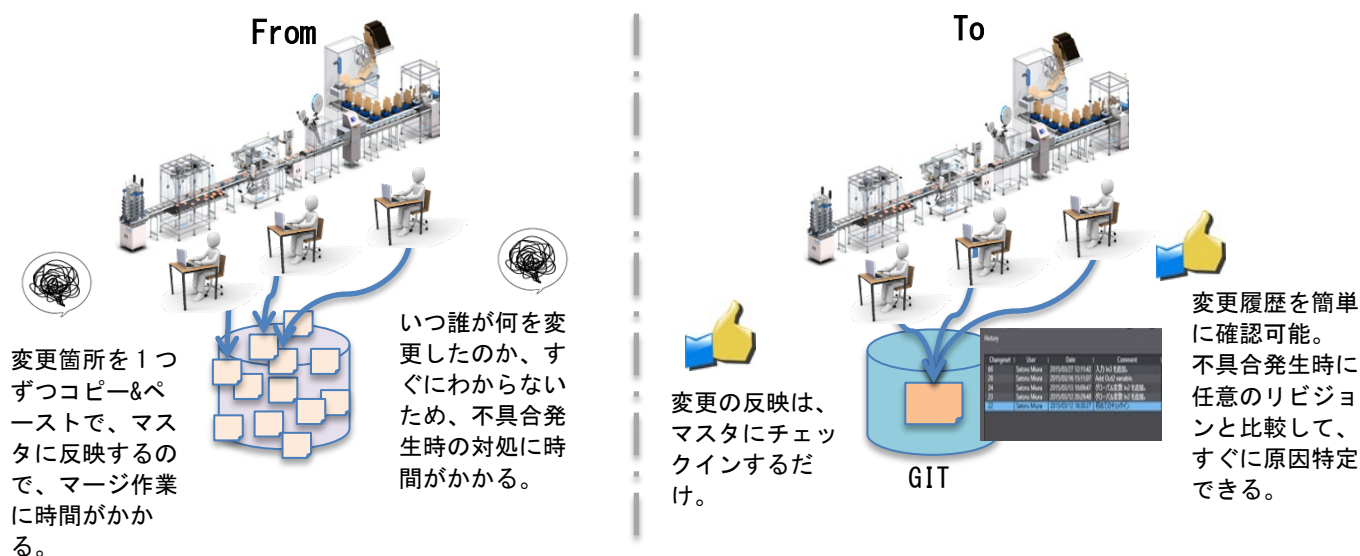
7.1 バージョン管理機能を使用した複数人開発および派生開発の概要

(1) 複数人開発

複数人開発では、各開発者のプロジェクト変更内容をマスタとなるプロジェクトへ反映する際、変更内容を漏れなく正しく反映する作業に時間がかかります。

たとえば、複数の開発者が同じ箇所を変更した場合、先にマスタとなるプロジェクトへ反映した内容が削除されることがあります。このとき、問題が発生した際の原因箇所の特定、対応に工数がかかってしまうことがあります。

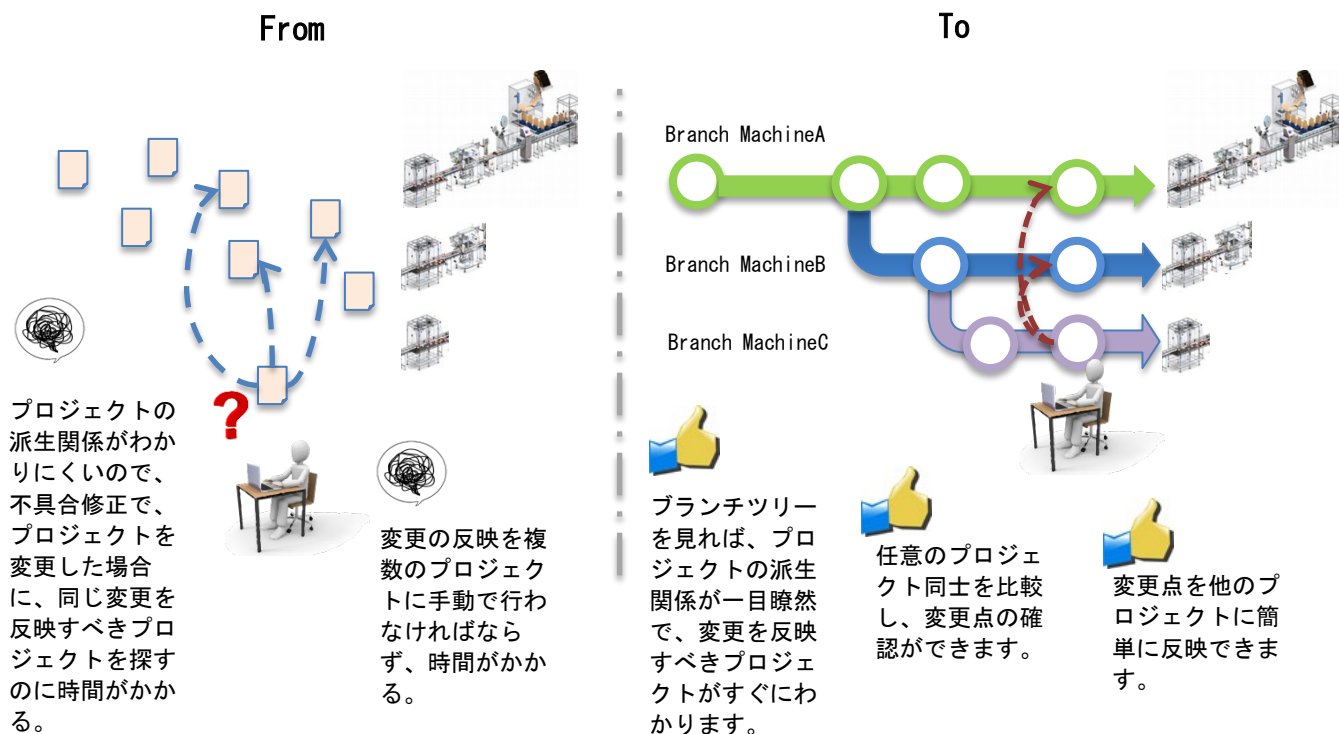
バージョン管理の履歴管理機能とマージ（結合）機能を用いることで、各開発者の変更内容を確実にマスタプロジェクトへ反映できるようになります。



(2) 派生開発

派生開発では、複数の派生装置のプロジェクトを個別に管理している場合、共通のプログラムの変更をそれぞれのプロジェクトに反映する際に、反映ミスや反映漏れが発生するという問題があります。

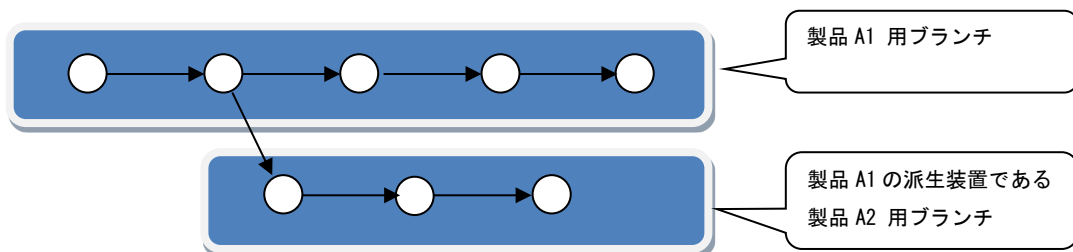
バージョン管理の履歴管理機能とマージ（結合）機能を用いることで、複数の派生装置のプロジェクトの変更点の把握、変更の反映を簡単にできるようになります。



7.2 ブランチとマージ

変更履歴情報の一連のつながりをブランチと呼びます。

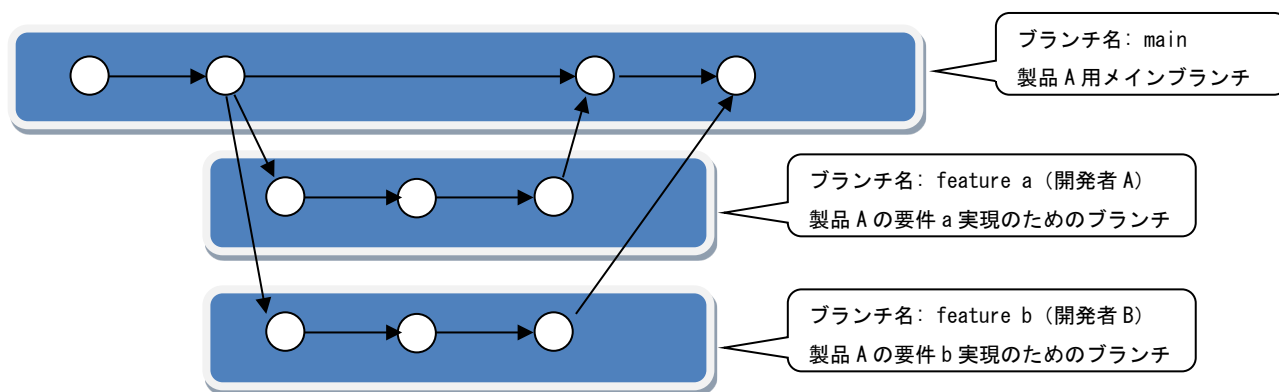
リポジトリを作成して使用開始した時点ではブランチは1つですが、ブランチのあるチェンジセットを起点として別のブランチを作成（分岐）し、元のブランチの変更に影響がない形でプロジェクトを管理することができます。



複数人での並行開発や製品のバージョンアップ開発を、ブランチを使用して行うことで、他の開発者が変更中のプロジェクトやすでにリリース済の製品のプロジェクトに影響がないように管理することができます。

複数人開発や派生開発では、分岐したブランチでの変更を元のブランチにも反映する場合があります。このとき、元のブランチと分岐したブランチを結合する「マージ」という機能を使用します。

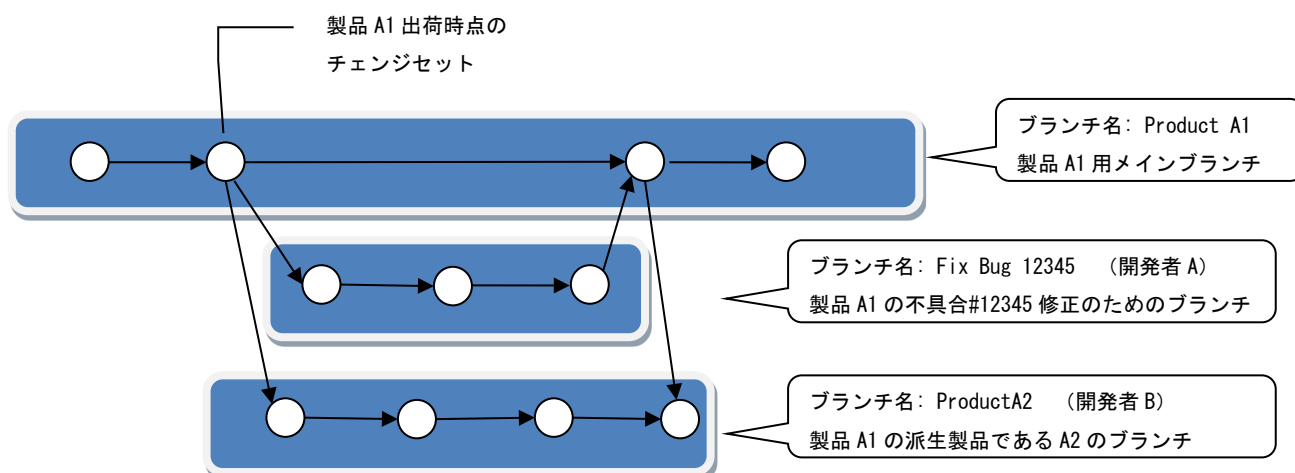
(1) 並行開発でのブランチ・マージの例



上図のように、最初に元となるブランチ「main」を作成します。要件 a と要件 b に対する開発を並行して行うためには、このブランチから、要件 a 用のブランチ「feature a」と要件 b 用のブランチ「feature b」を作成します。

それぞれのブランチの中で要件に対する開発が完了し品質確保ができた時点で、元のブランチ「main」へマージを行います。

(2) 派生開発でのブランチ・マージの例



上図のように、製品 A1 の派生商品である A2 の開発を行う場合は、製品 A1 のブランチ「Product A1」の、製品出荷時点のチェンジセットからブランチ「Product A2」を作成し、そのブランチの中で開発を進めます。製品 A1 で不具合が発生し、プロジェクトの修正が必要となった場合、製品 A1 の出荷時点のチェンジセットから不具合修正のためのブランチ「Fix Bug 12345」を作成し、そのブランチの中で変更内容の品質確保ができた時点で、元のブランチ「Product A1」へマージを行います。

この修正を製品 A1 だけでなく、製品 A2 にも適用する必要がある場合は、不具合修正のブランチ「Fix Bug 12345」での変更内容を一括して製品 A2 のブランチ「Product A2」に適用することができます。不具合修正のブランチをマージした後の製品 A1 用ブランチのチェンジセットを、製品 A2 のブランチにマージします。

7.3 Sysmac Studio プロジェクトの複数人開発手順

Sysmac Studio プロジェクトを複数人で並行して開発するためのプロジェクト編集の分担の考え方と、分担の例、および操作手順を説明します。

7.3.1 複数人で並行して開発するための編集の分担

Sysmac Studio のバージョン管理機能は、複数人で 1 つのプロジェクトを同時に編集しリモートリポジトリへプッシュする際に、他の開発者の変更内容を結合する機能を持っています。

ただし、他の開発者が同じデータを同時に編集していた場合、変更をマージしようとする、変更内容に競合が発生しマージすることができません。この問題を回避するために、以下の表に示す複数人編集可能（○）のデータを複数人で分担し、複数人編集不可（×）のデータは特定の 1 人が編集します。

データ名	複数人での編集可否
EtherCAT	×
CPU・増設ラック	×
I/O マップ	×
コントローラ設定	×
軸設定	×
軸グループ設定	×
カムデータ設定	×
イベント設定	×
タスク設定	×
プログラム	○
ファンクション	○
ファンクションブロック	○
データ型	×
グローバル変数	×

7.3.2 分担例

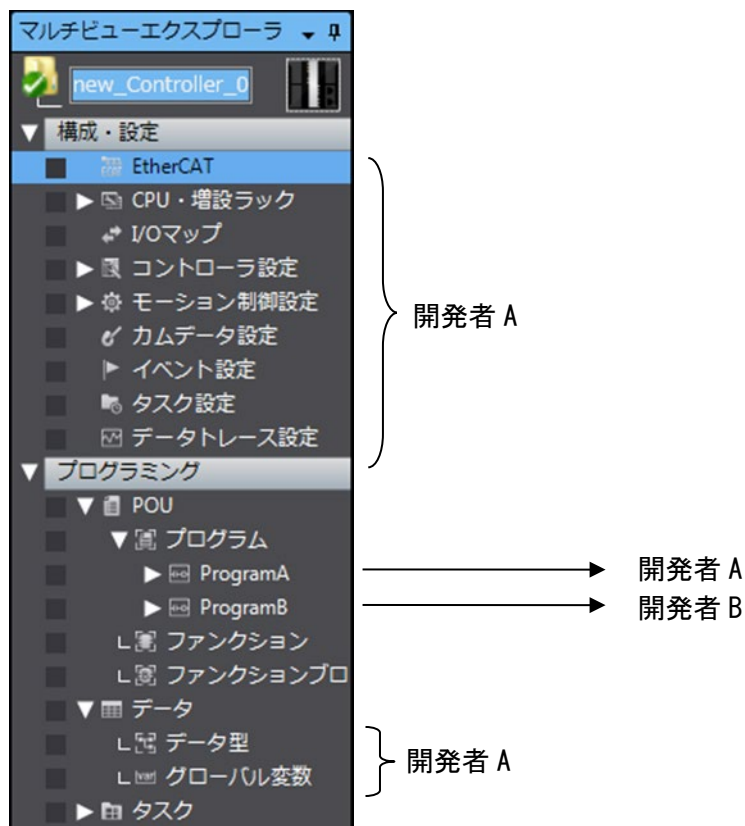
ここでは、次章の操作手順のための開発者の役割分担とシナリオを記載します。

7.3.2.1 開発者とその役割分担

開発者は2名で、開発者A、開発者Bとします。

開発者Aは、Sysmac Studio プロジェクトの構成・設定全般と、データ型、グローバル変数、ProgramA の編集を担当します。

開発者Bは、ProgramB の編集を担当します。



7.3.2.2 シナリオ

前述した役割分担で、開発者 A、開発者 B は、以下の表の手順で作業を進めます。

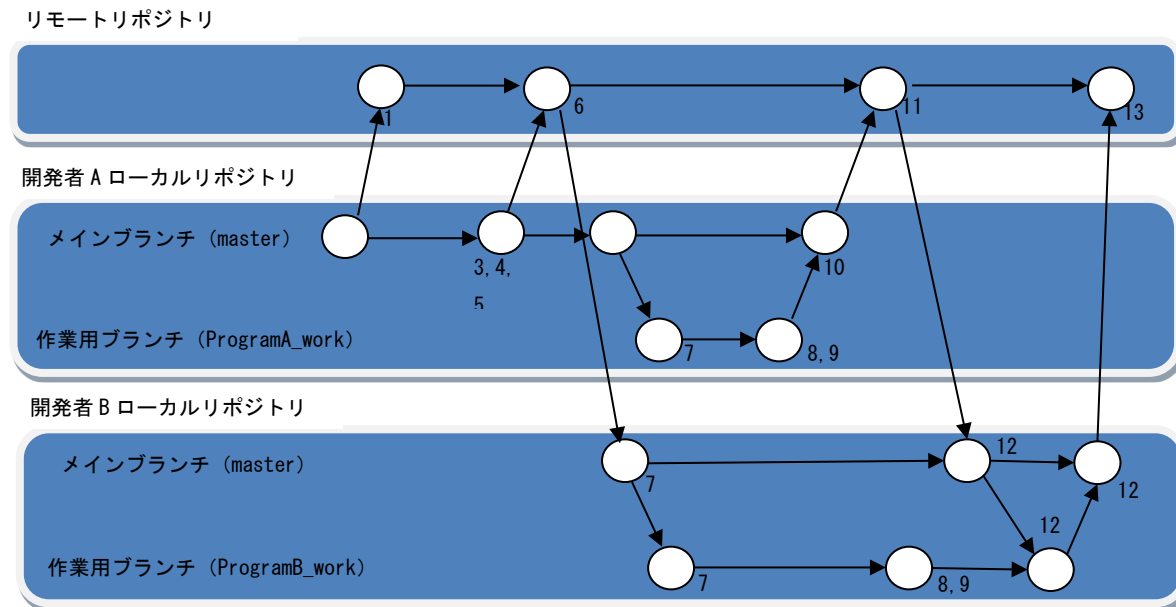
No.	開発者 A の作業	開発者 B の作業	作業手順の参照先
1	リモートリポジトリにプロジェクトを登録する ※ベースとなるプロジェクトを開発者 A がリポジトリに登録します。	-	7.3.3.1 リモートリポジトリにプロジェクトを登録する (No. 1 : 開発者 A)
2	リモートリポジトリからプロジェクトを読み出す	-	7.3.3.2 リモートリポジトリからプロジェクトを読み出す (No. 2 : 開発者 A)
3	構成・設定情報を入力する	-	7.3.3.3 構成・設定情報、データ型、グローバル変数、プログラム POU が登録されたプロジェクトをコミットする (No. 3、4、5 : 開発者 A)
4	データ型、グローバル変数、プログラム POU を入力する	-	
5	構成・設定情報、データ型、グローバル変数、プログラム POU が登録されたプロジェクトをローカルリポジトリにコミットする	-	
6	ローカルリポジトリの変更をリモートリポジトリへプッシュする ※この操作のあと、開発者 A と開発者 B で並行して編集可能です。	-	7.3.3.4 ローカルリポジトリの変更をリモートリポジトリへプッシュする (No. 6 : 開発者 A)
7	開発者 A の作業用のブランチをローカルリポジトリに作成する	・ リモートリポジトリの対象プロジェクトを開発者 B のパソコンにコピーし、ローカルリポジトリを作成する ・ 作業用のブランチを作成する	<開発者 A> 7.3.3.6 開発者 A の作業用のブランチを作成する (No. 7 : 開発者 A) <開発者 B> 7.3.3.5 開発者 B のパソコンにローカルリポジトリを作成する (No. 7 : 開発者 B) 7.3.3.9 開発者 B の作業用のブランチを作成する (No. 7 : 開発者 B)
8	ProgramA を入力する グローバル変数を追加する	ProgramB を入力する グローバル変数の初期値を変更する	7.3.3.7 ProgramA およびグローバル変数の変更を作業用のブランチにコミットする (No. 8、9 : 開発者 A) 7.3.3.10 ProgramB およびグローバル変数の変更をブランチにコミットする (No. 8、9 : 開発者 B)
9	ProgramA およびグローバル変数の変更を作業用のブランチにコミットする	ProgramB およびグローバル変数の変更を作業用のブランチにコミットする	
10	メインブランチに作業用のブランチの変更をマージする ※No. 1 で作成したローカルリポジトリのプロジェクトが開発者 A のメインブランチです。	-	7.3.3.8 メインブランチに作業用のブランチの変更をマージする (No. 10 : 開発者 A)
11	開発者 A のローカルリポジトリの変更をリモートリポジトリへプッシュする ※この操作のあと、開発者 A の変更と開発者 B の変更とをマージ可能です。	-	7.3.3.11 開発者 A のローカルリポジトリの変更をリモートリポジトリへプッシュする (No. 11 : 開発者 A)
12	-	メインブランチから最新のデータを取得して開発者 B の作業用のブランチにマージし、マージ後のデータをメインブランチに再度マージする	7.3.3.12 メインブランチに開発者 B の作業用のブランチの変更をマージする (No. 12 : 開発者 B)
13	-	開発者 B のメインブランチの変更をローカルリポジトリからリモートリポジトリへプッシュする	7.3.3.13 開発者 B のメインブランチの変更をリモートリポジトリへプッシュする (No. 13 : 開発者 B)

- : 作業なし

リポジトリとブランチの関係は次のとおりです。

リモートリポジトリを介して各開発者のパソコンのローカルリポジトリのメインブランチの変更をやり取りし、各開発者のローカルリポジトリ上で作業用ブランチの作成や変更のマージを行います。

図中の○の横の番号は、前述の表に示した開発者 A、B の作業 No. に対応します。



7.3.3 操作手順

「7.3.2.2 シナリオ」の内容に沿って、操作手順を説明します。

7.3.3.1 リモートリポジトリにプロジェクトを登録する (No. 1 : 開発者 A)

1. リモートリポジトリの作成

複数人でリポジトリを共有するために、リモートリポジトリを作成します。

ここでは 開発者 A のパソコンに「C:\Git\MachineA.git」というフォルダ名のリモートリポジトリを作成します。

手順の詳細は、「5.6 共有フォルダとリモートリポジトリの作成」を参照してください。

2. ローカルリポジトリの登録とベースとなるプロジェクトの作成、保存

新規のプロジェクトを作成し、そのプロジェクトをローカルリポジトリに登録します。

ここでは、開発者 A のパソコンで「MachineA」というプロジェクトを作成してローカルリポジトリにコミットし、リモートリポジトリへプッシュします。

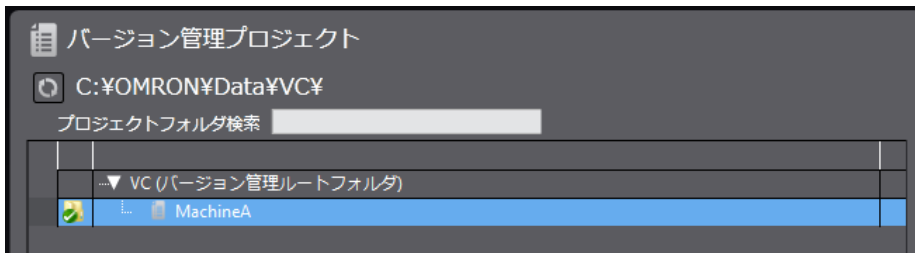
手順の詳細は、「6.3 プロジェクト管理の開始準備（ベースとなるプロジェクトの作成、保存）」を参照してください。

7.3.3.2 リモートリポジトリからプロジェクトを読み出す (No. 2 : 開発者 A)

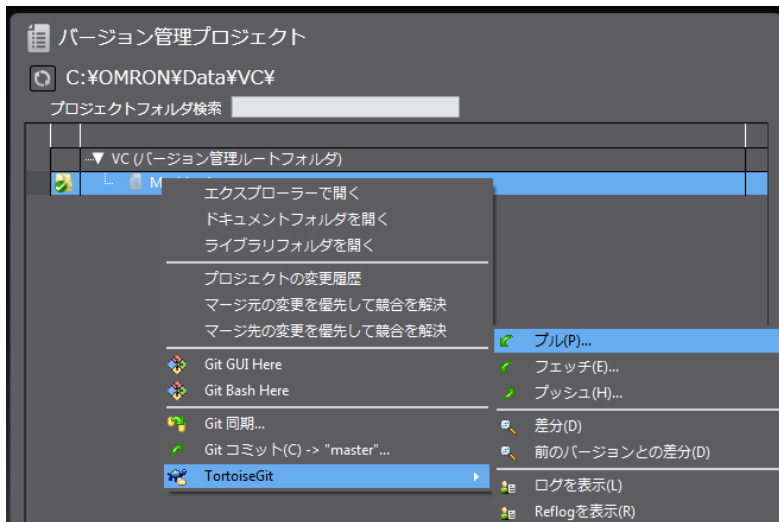
リモートリポジトリにプロジェクトを登録したあとは、他の開発者がリモートリポジトリの対象プロジェクトを更新している可能性があります。マスタとなるリモートリポジトリと開発者 A のパソコンのローカルリポジトリの同期をとるために次の操作を行います。

なお、本説明のシナリオでは開発者 A 以外はリモートリポジトリにアクセスしていないため、「7.3.3.1 リモートリポジトリにプロジェクトを登録する (No.1: 開発者 A)」の手順の直後は、この操作は必要ありません。

1. Sysmac Studio のスタートページで [バージョン管理エクスプローラ] を選択し、[バージョン管理プロジェクト] 画面を開きます。

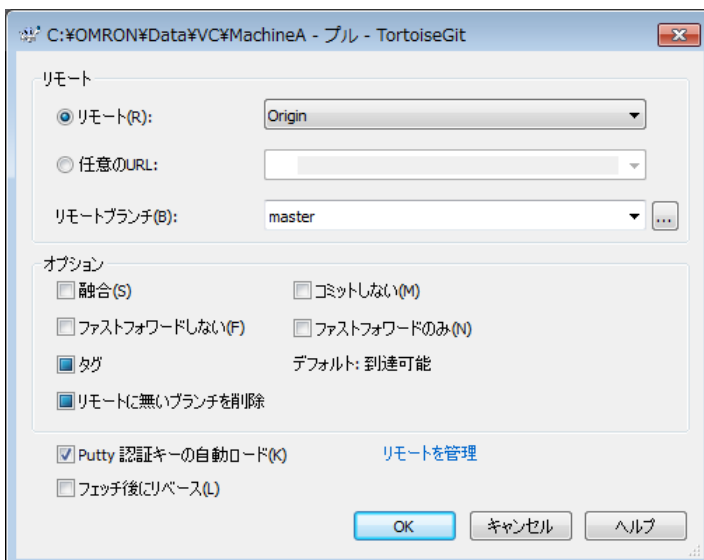


2. MachineA を右クリックして、[TortoiseGit] - [プル] を選択します。

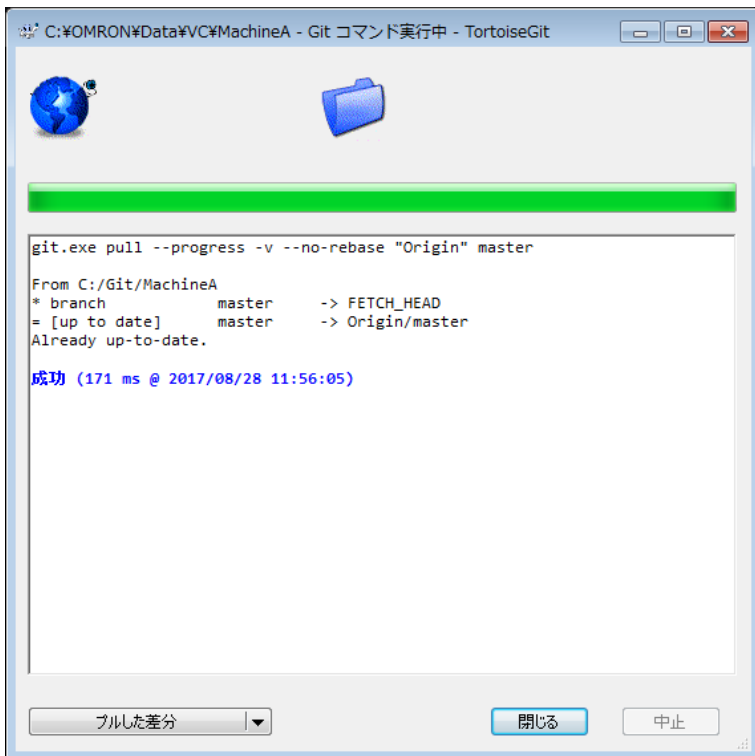


プルダイアログボックスが表示されます。

3. [リモート] に [Origin] (リモートリポジトリのパスを「C:\¥Git¥MachineA.git」と設定) が選択されていることを確認し、[OK] ボタンをクリックします。



プル処理が完了します。

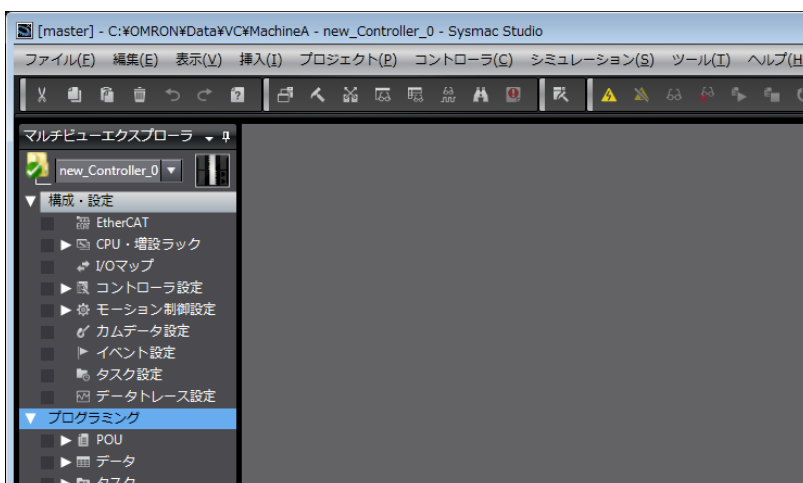


4. 「閉じる」 ボタンを押します。

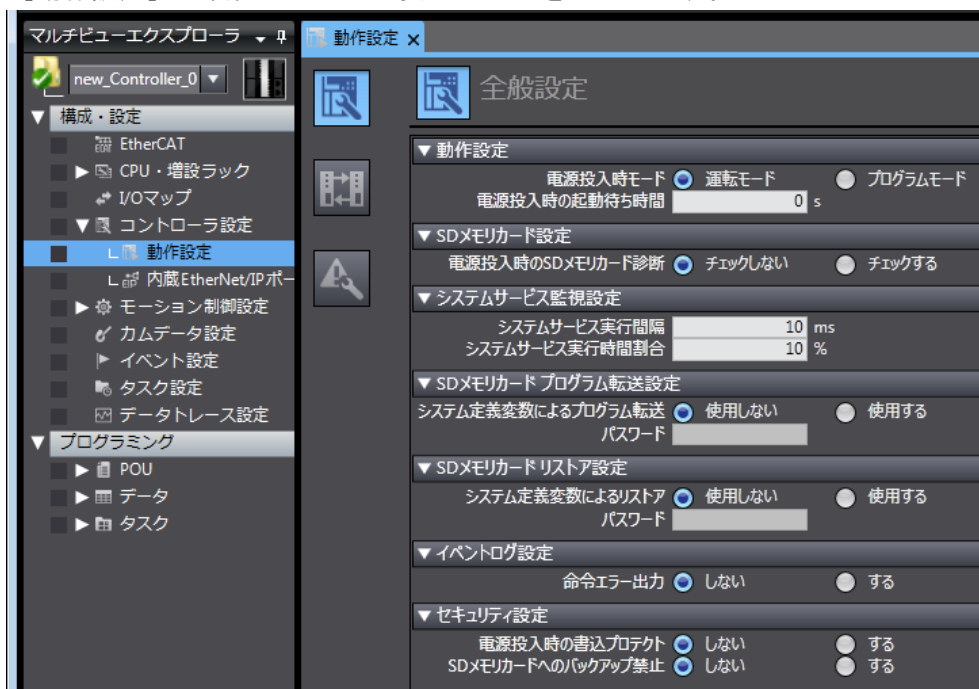
7.3.3.3 構成・設定情報、データ型、グローバル変数、プログラム POU が登録されたプロジェクトをコミットする (No. 3、4、5: 開発者 A)

1. 「バージョン管理エクスプローラ」でプロジェクト「MachineA」を選択し、「開く」ボタンをクリックします。

新規作成のプロジェクトが開きます。



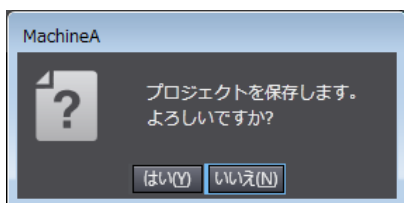
2. 「動作設定」の項目について必要なデータを入力します。



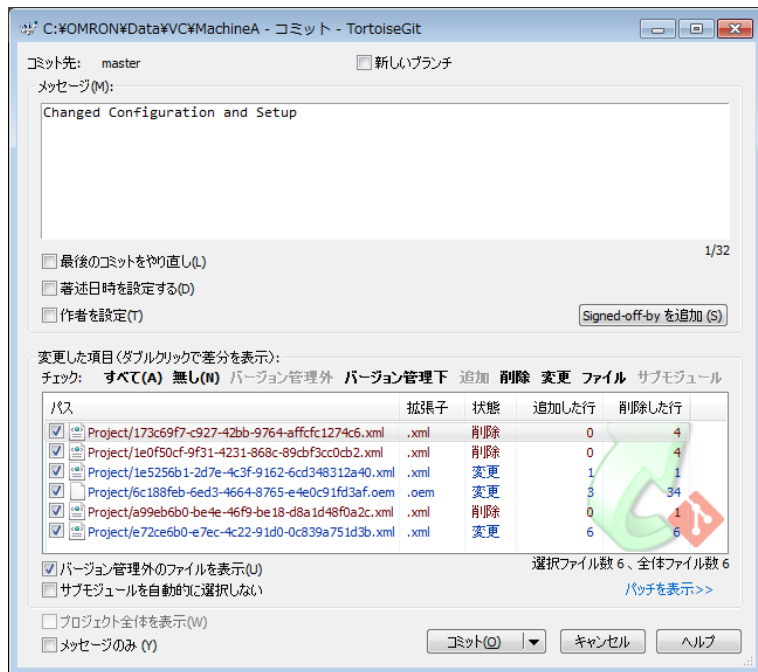
3. ローカルリポジトリにコミットするために、マルチビューエクスプローラのフォルダアイコンを右クリックして、[Git コミット -> "master"] を選択します。



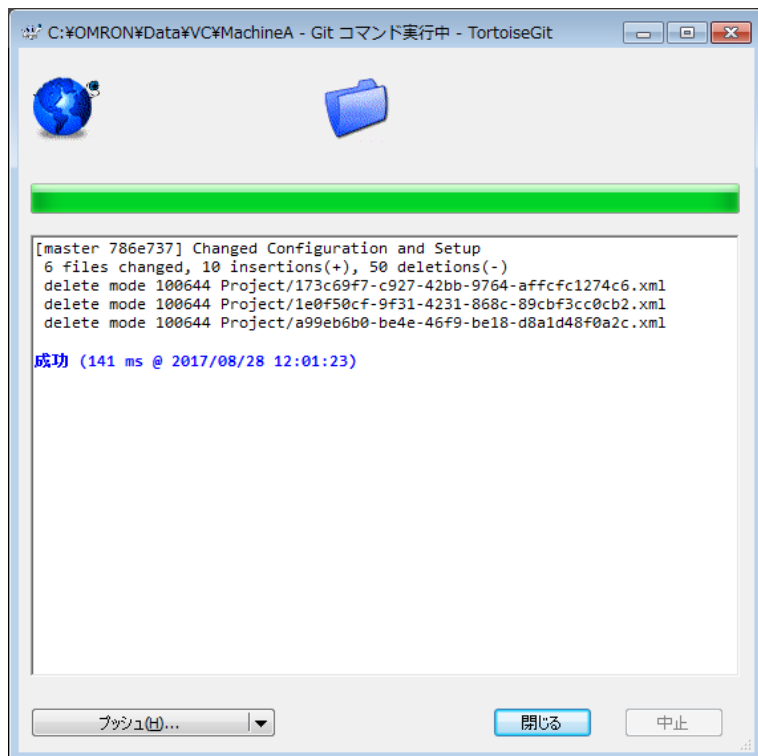
保存確認のダイアログボックスが表示されます。



4. [はい] ボタンをクリックします。
[コミット] ダイアログボックスが表示されます。

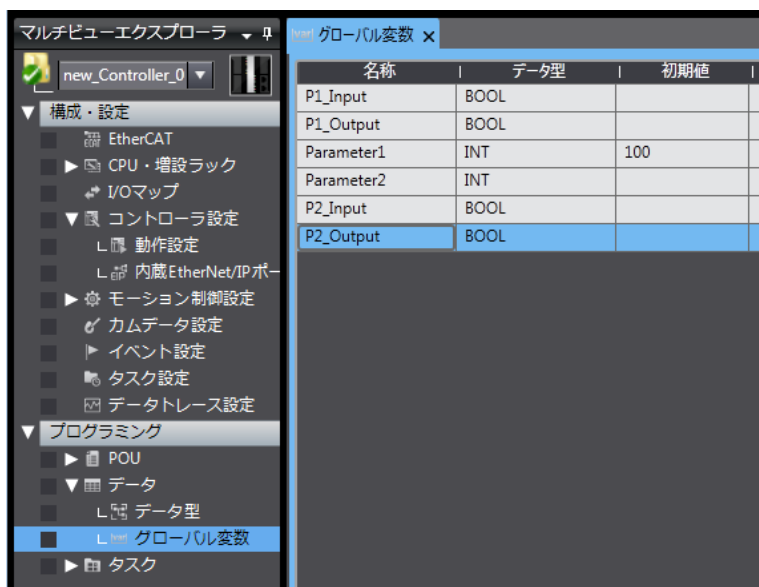


5. [メッセージ] エリアにコメントを記入して、[コミット] ボタンをクリックします。
コミットが完了します。

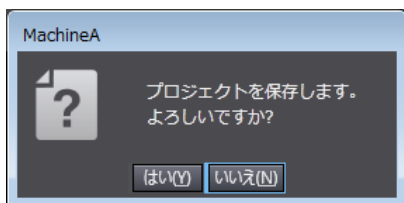


6. [閉じる] ボタンをクリックします。

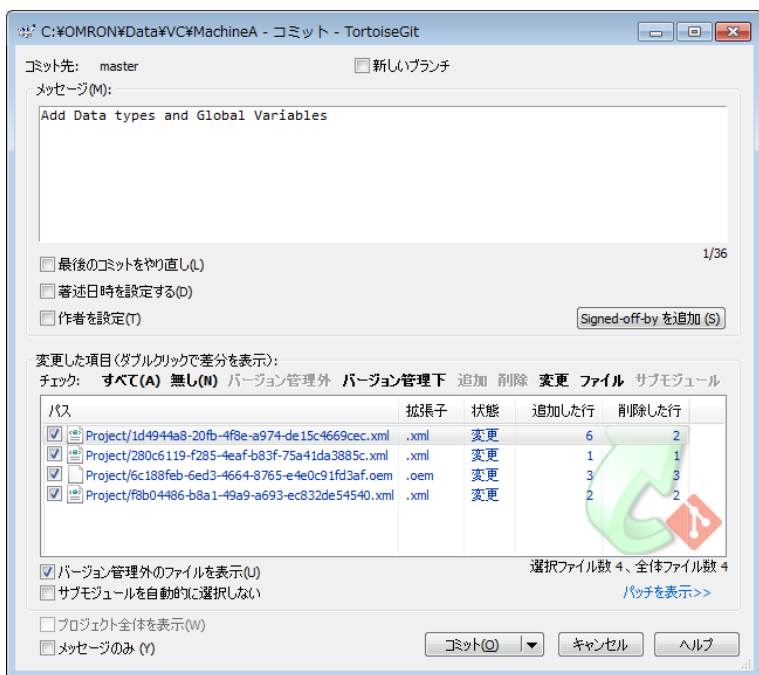
7. 次に、データ型、グローバル変数を入力します。



8. ローカルリポジトリにコミットするために、マルチビューエクスプローラのフォルダアイコンを右クリックして、[Git コミット -> “master”] を選択します。
保存確認のダイアログボックスが表示されます。



9. [はい] をクリックします。
コミットダイアログボックスが表示されます。

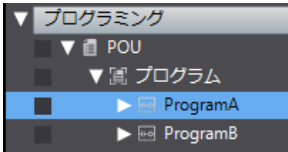


10. [メッセージ] エリアにコメントを記入して、[コミット] ボタンをクリックします。

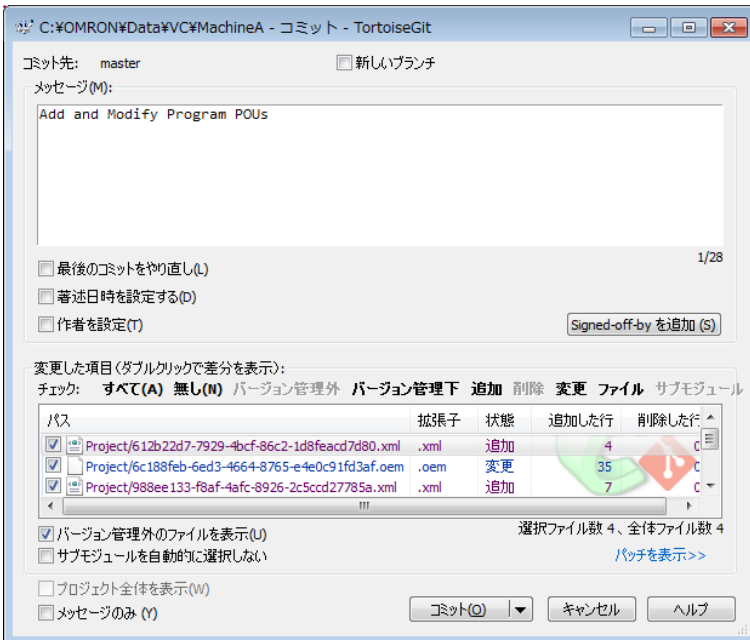
11. コミットの完了ダイアログボックスで [閉じる] ボタンをクリックします。

12. つづけて、プログラム POU を作成します。

この例では、開発者 A が ProgramA を、開発者 B が ProgramB を作成します。ここでは開発者 A が事前に両方のプログラム POU を作成します。



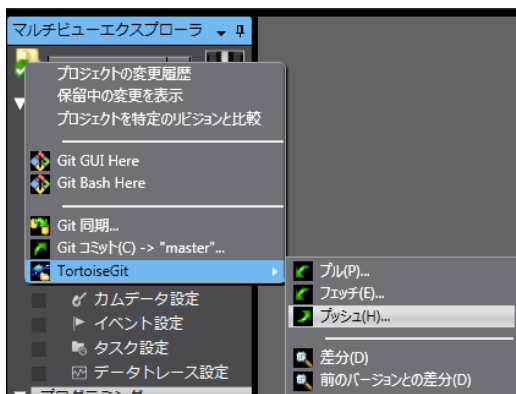
13. データ型、グローバル変数と同様に、コミットします。



7.3.3.4 ローカルリポジトリの変更をリモートリポジトリへプッシュする (No. 6 : 開発者 A)

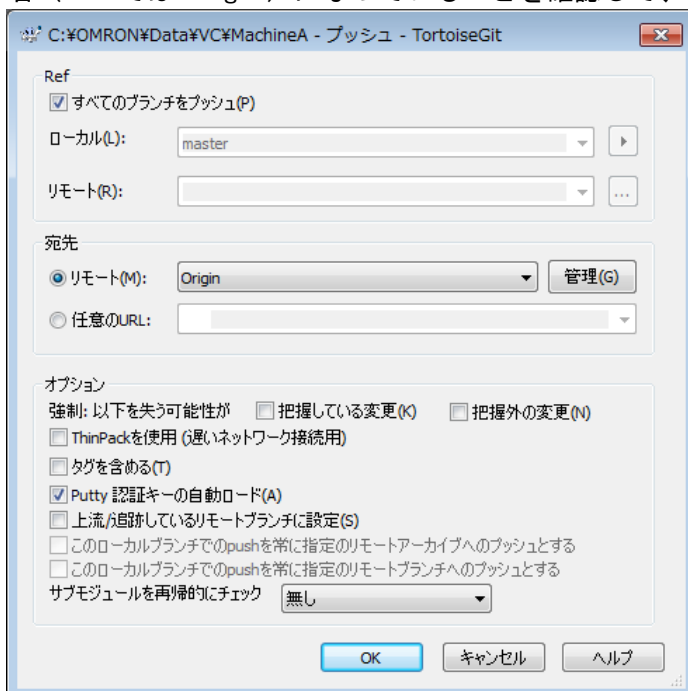
開発者 B と並行して作業する前に、開発者 A が担当していた共通の作業（構成・設定、データ型、グローバル変数、プログラム POU の登録）の結果をリモートリポジトリへプッシュします。

1. マルチビューエクスプローラのフォルダアイコン、または [バージョン管理プロジェクト] 画面でプロジェクトを右クリックして、[TortoiseGit] - [プッシュ] を選択します。

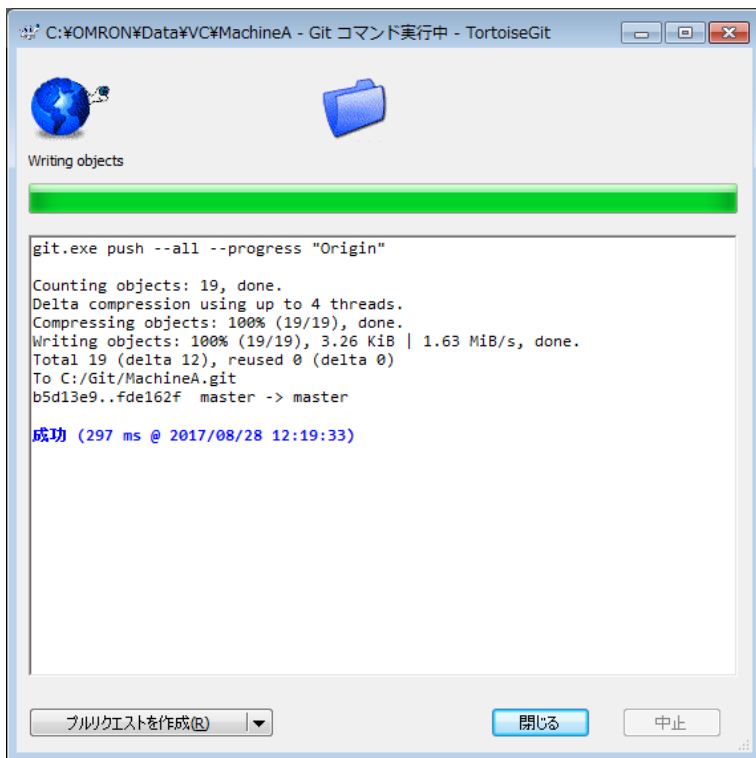


プッシュダイアログボックスが表示されます。

2. [すべてのブランチをプッシュ] チェックボックスにチェックを入れ、[リモート] が登録していたリモート名（ここでは Origin）になっていることを確認して、[OK] ボタンをクリックします。



プッシュ処理が完了します。



3. [閉じる] ボタンをクリックします。

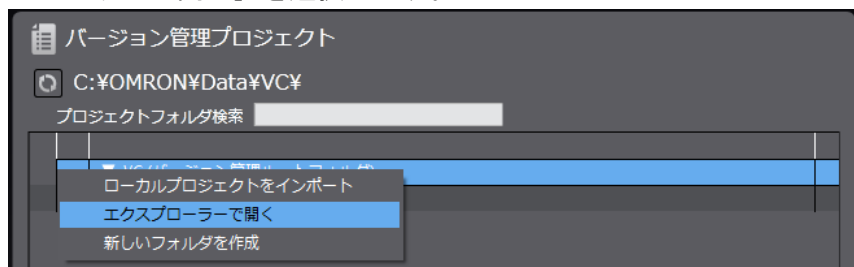
7.3.3.5 開発者 B のパソコンにローカルリポジトリを作成する (No. 7 : 開発者 B)

ここまでの操作で、開発者 A が編集（構成・設定、データ型、グローバル変数の登録）したデータがリモートリポジトリに登録されています。

これ以降、開発者 B も並行して開発作業をできるようにするために、リモートリポジトリから対象プロジェクトを取得し、開発者 B のパソコンにローカルリポジトリを作成します。

開発者 B のパソコンのソフトウェアのセットアップおよび基本設定が完了しているものとします。

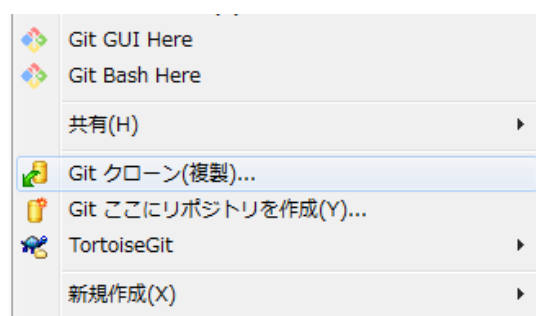
1. [バージョン管理プロジェクト] 画面で [VC(バージョン管理ルートフォルダ)] を右クリックして、[エクスプローラーで開く] を選択します。



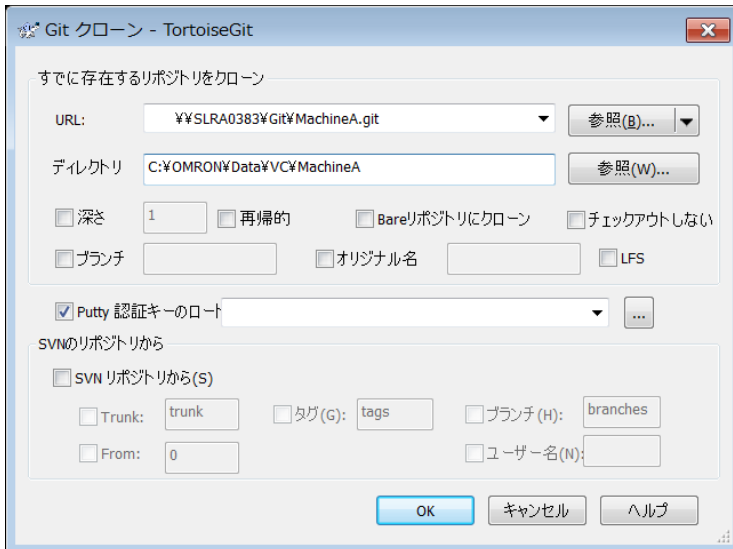
Windows エクスプローラーで「VC」フォルダが開きます。



2. VC フォルダ内の任意の場所で右クリックして、[Git クローン(複製)] を選択します。



[Git クローン] ダイアログボックスが表示されます。

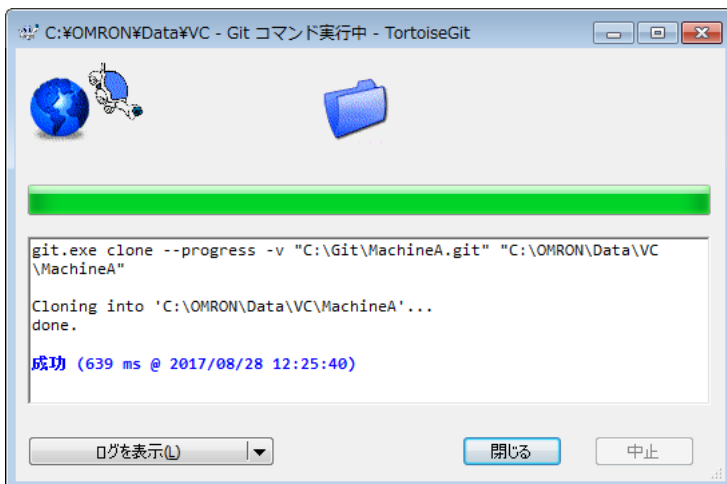


3. [Git クローン] ダイアログボックスで次の情報を入力します。

- [URL] にリモートリポジトリのパス
(本説明の例では¥¥SLRA0383¥Git¥MachineA.git、パソコン名 SLRA0383、共有名 Git、リポジトリ名 MachineA.git)
- [ディレクトリ] にローカルリポジトリのパス (ここでは「C:¥OMRON¥Data¥VC¥MachineA」)

4. [OK] ボタンをクリックします。

Git クローンコマンドにより、リモートリポジトリが開発者 B のパソコンに複製されました。以後、このリポジトリをローカルリポジトリと呼びます。

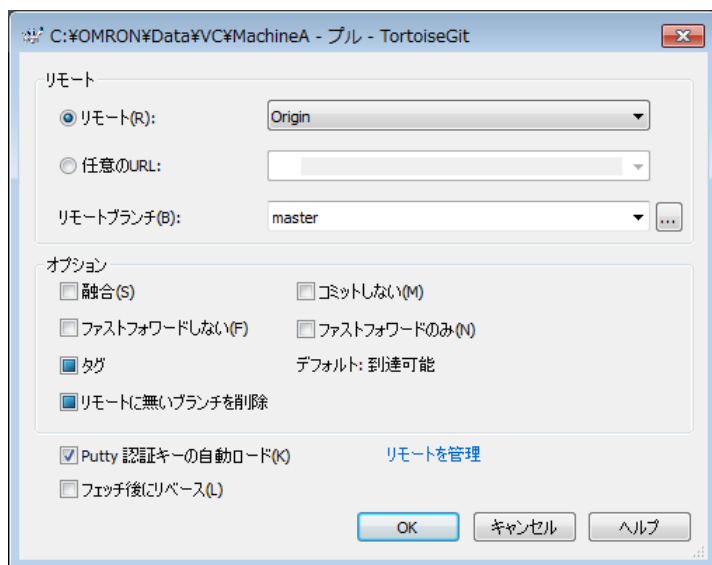


7.3.3.6 開発者 A の作業用のブランチを作成する (No. 7 : 開発者 A)

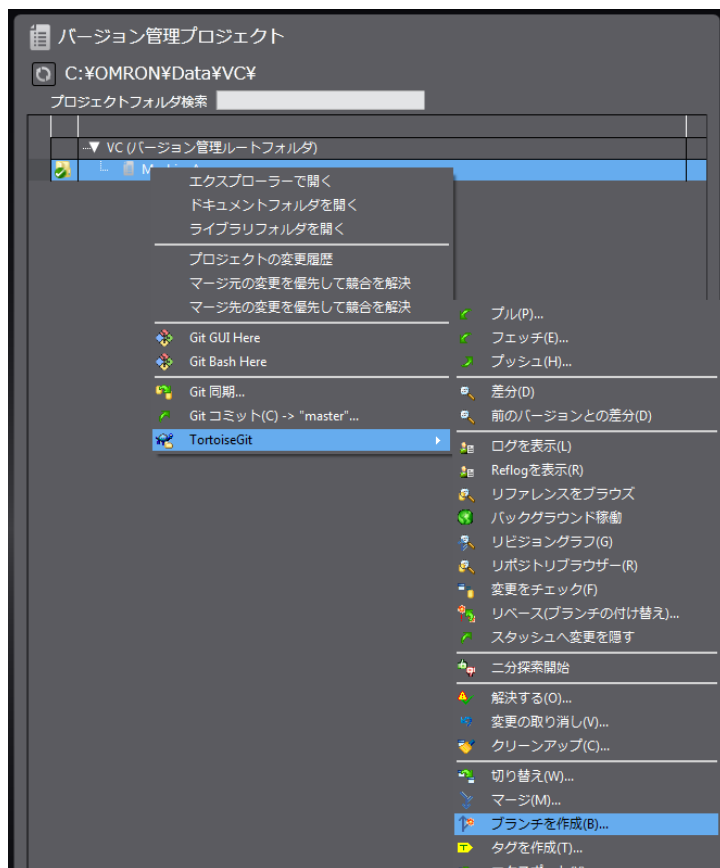
リモートリポジトリにプロジェクトを登録したあとは、他の開発者がリモートリポジトリの対象プロジェクトを更新している可能性があります。マスタとなるリモートリポジトリと開発者 A のパソコンのローカルリポジトリの同期をとるために次の操作を行います。

1. Sysmac Studio のスタートページで [バージョン管理エクスプローラ] を選択し、[バージョン管理プロジェクト] 画面を表示します。

- MachineA を右クリックして、[TortoiseGit] - [プル] を選択します。
プルダイアログボックスが表示されます
- [リモート] に [Origin] (リモートリポジトリのパスを「C:¥Git¥MachineA.git」と設定) が選択されていることを確認し、[OK] ボタンをクリックします。

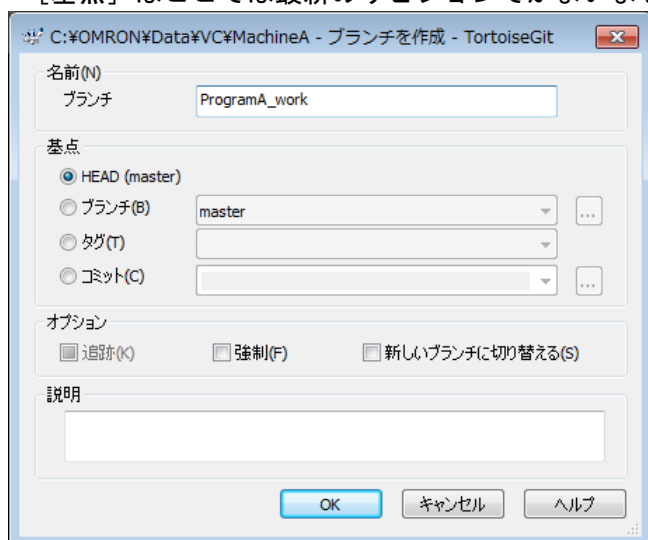


- プル処理の完了ダイアログボックスで [閉じる] ボタンをクリックします。
- ブランチを作成するために、マルチビューエクスプローラのフォルダアイコン、または [バージョン管理プロジェクト] 画面でプロジェクトを右クリックして、[TortoiseGit] - [ブランチを作成] を選択します。



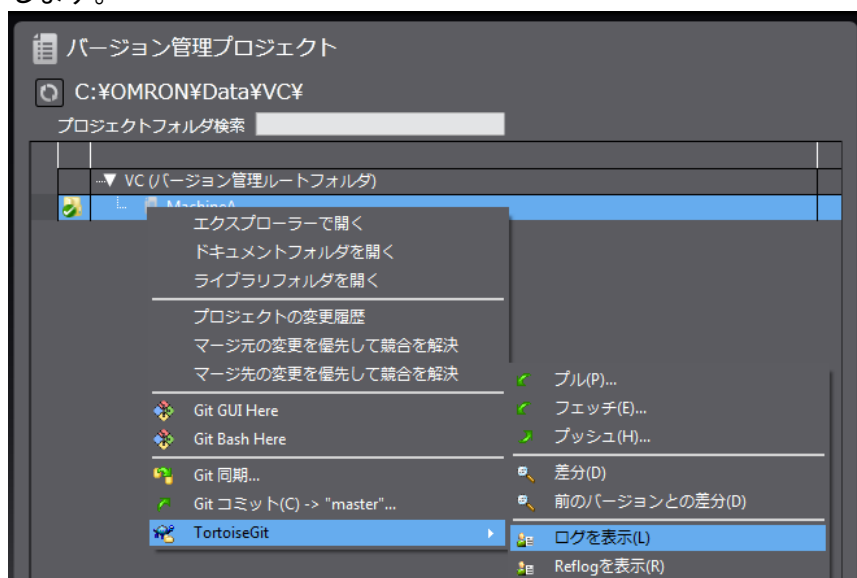
[ブランチを作成] ダイアログボックスが表示されます。

6. 適切なブランチ名（ここでは「ProgramA_work」）を入力し、[OK] ボタンをクリックします。
[基点] はここでは最新のリリースでかまいません。Head（master）を選択します。

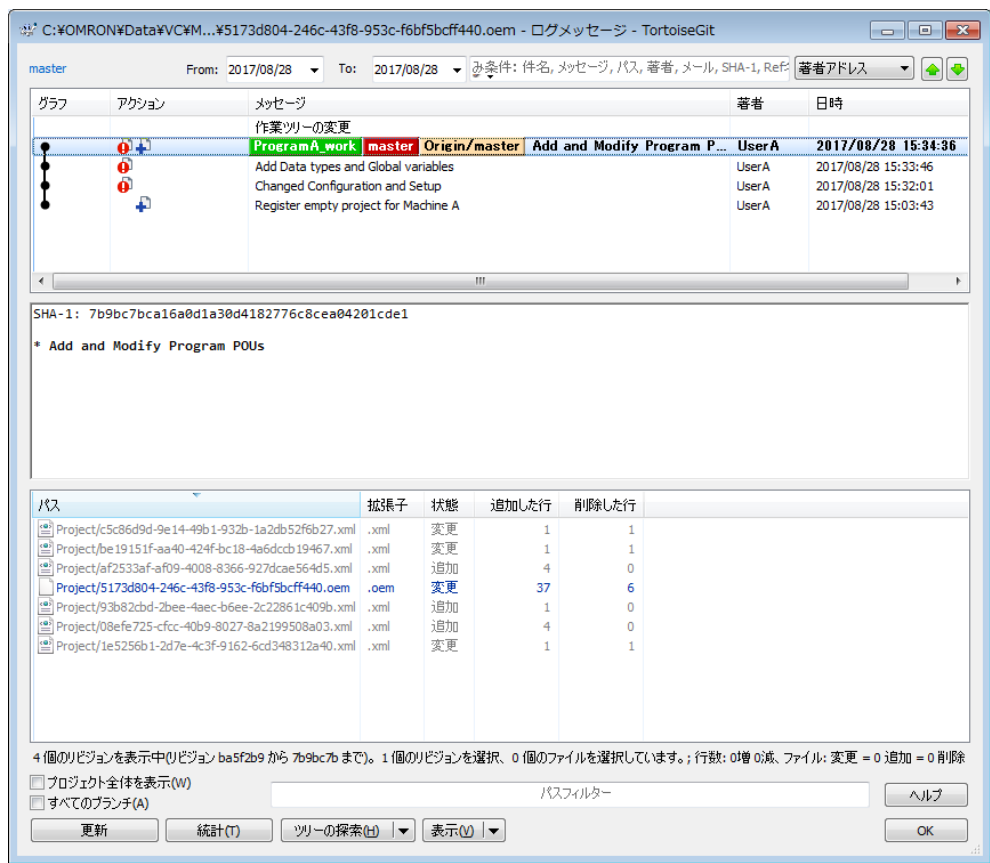


ブランチの作成が完了します。

7. ブランチが作成されたことを確認するために、マルチビューエクスプローラのフォルダアイコン、または [バージョン管理プロジェクト] 画面でプロジェクトを右クリックして、[TortoiseGit] - [ログを表示] を選択します。



ログメッセージダイアログボックスが表示されます。

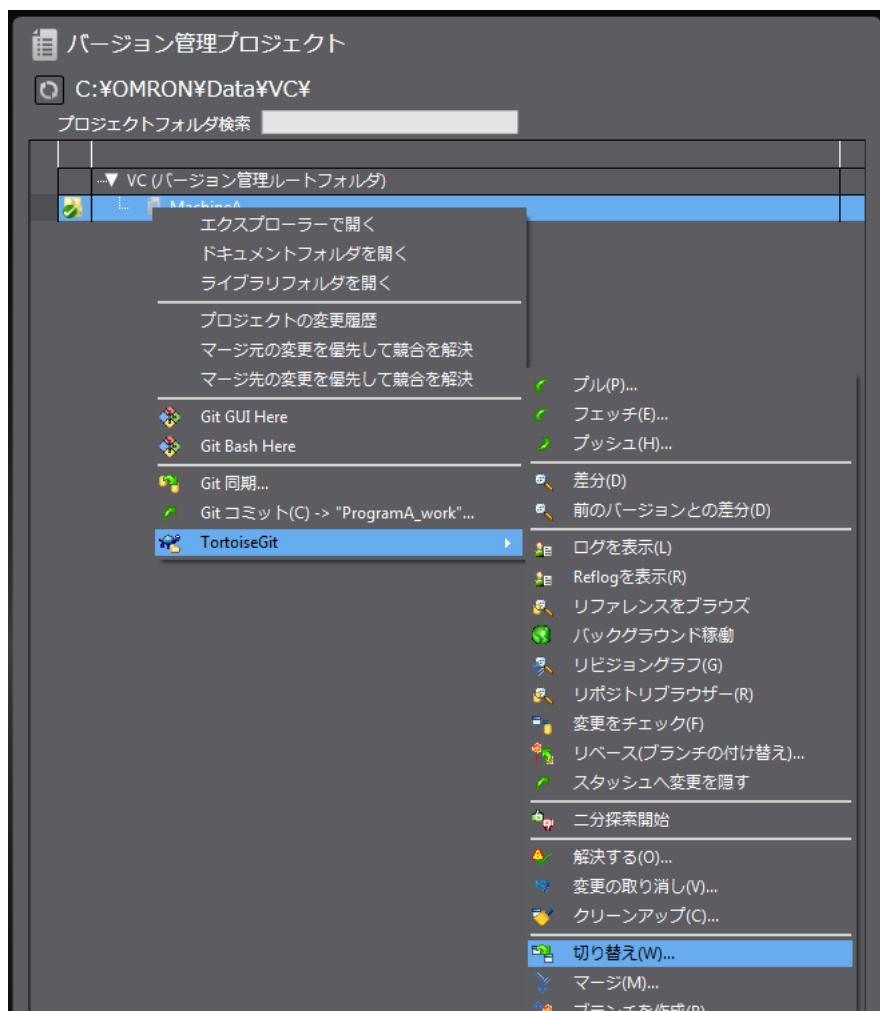


上部ペインの一番上の行に、「ProgramA_work」「master」「Origin/master」と表示されています。作業用のブランチ「ProgramA_work」が作成されていることが確認できます。
この行の表示は、最後のリビジョンである「Add and Modify Program P0Us」という変更が、ローカルリポジトリの作業用のブランチ「ProgramA_work」、メインブランチ「master」、およびリモートリポジトリ（接続先がOrigin）の「master」に存在していることを示しています。

7.3.3.7 ProgramA およびグローバル変数の変更を作業用のブランチにコミットする（No. 8、9：開発者 A）

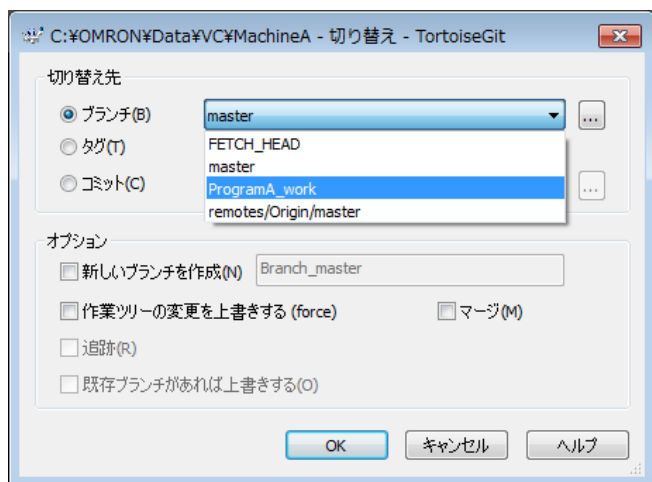
開発者 A のパソコンで作業をするブランチを切り替えて、ProgramA およびグローバル変数を変更します。

1. マルチビューエクスプローラのフォルダアイコン、または [バージョン管理プロジェクト] 画面でプロジェクトを右クリックして、[TortoiseGit] - [切り替え] を選択します。

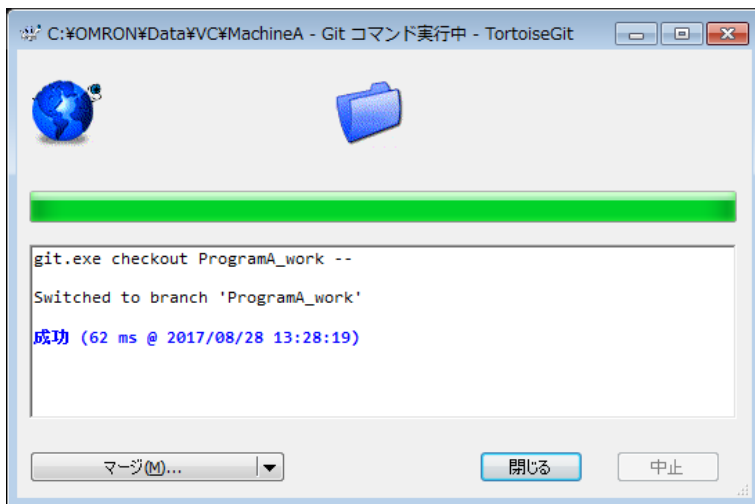


「切り替え」ダイアログボックスが表示されます。

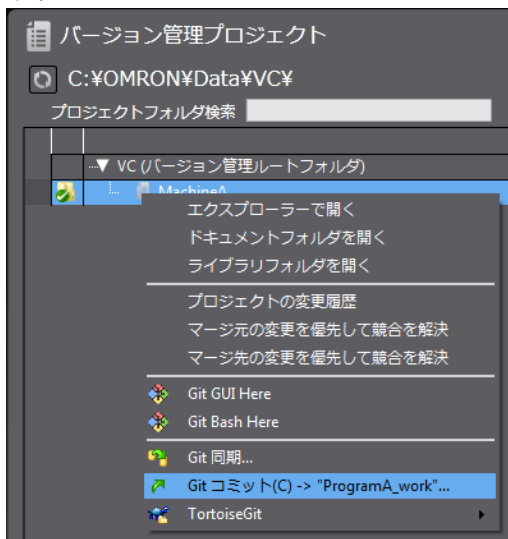
2. 「Branch」を選択し、「7.3.3.6 開発者 A の作業用のブランチを作成する (No. 7 : 開発者 A)」で作成した開発者 A の作業用のブランチである「ProgramA_work」を選択して、「OK」ボタンをクリックします。



ブランチの切り替えが完了します。



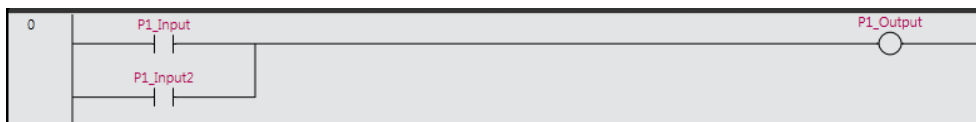
3. [閉じる] ボタンを押します。
4. マルチビューエクスプローラのフォルダアイコン、または [バージョン管理プロジェクト] 画面でプロジェクトを右クリックします。
メニューの [Git コミット->) に続くブランチ名が、切り替えたブランチに変わっていることを確認できます。



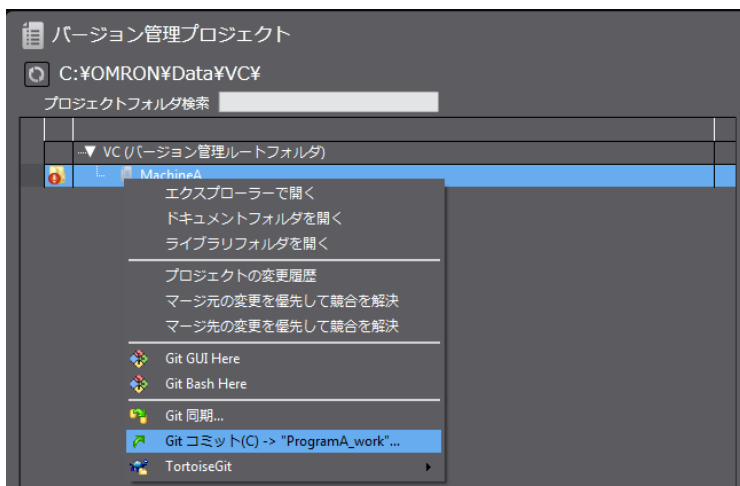
5. ProgramA を入力、グローバル変数を変更します。
以下の編集作業を行います。
グローバル変数に新たな変数 “P1_Input2” を追加

名称	データ型	初期値	割付先
P1_Input	BOOL		
P1_Output	BOOL		
P1_Input2	BOOL		
Parameter1	INT	100	
Parameter2	INT		
P2_Input	BOOL		
P2_Output	BOOL		

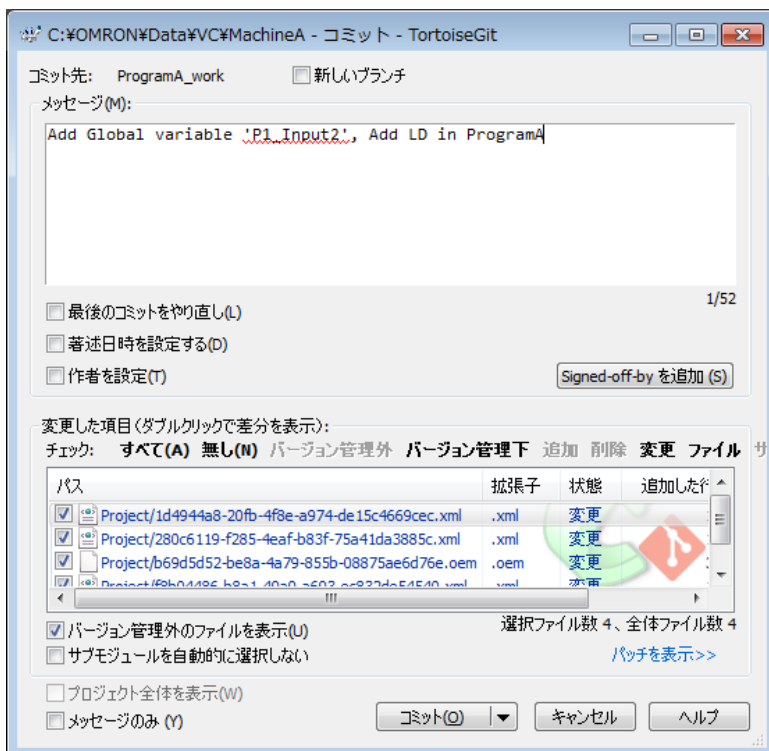
ProgramA-Section0 に以下のラダー図を入力



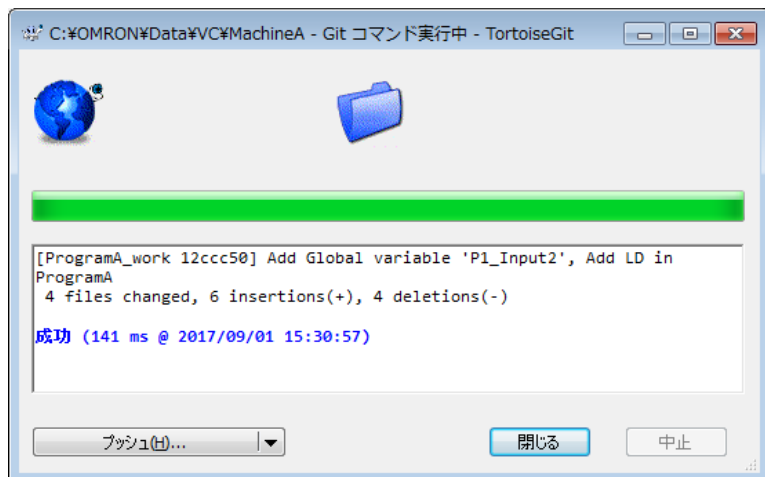
6. メインメニューから [ファイル] - [保存] を選択してプロジェクトを保存します。
7. [ファイル] - [閉じる] を選択してプロジェクトを閉じます。
8. [バージョン管理プロジェクト] 画面で対象プロジェクトを右クリックして、[Git コミット -> "ProgramA_work"] を選択します。



コミットダイアログボックスが表示されます。



9. [メッセージ] エリアにコメントを入力し、[コミット] ボタンをクリックします。
コミットが完了します。

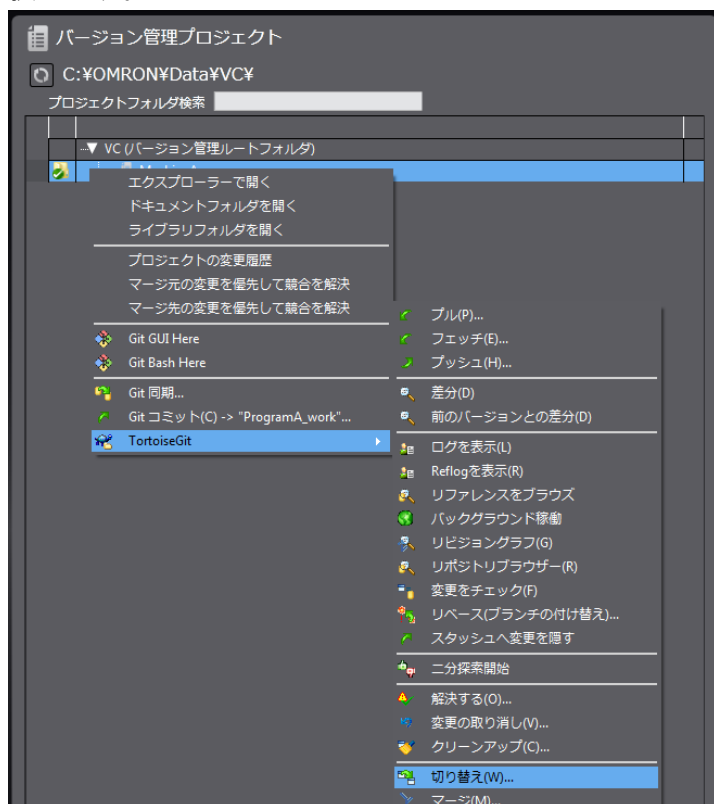


10. [閉じる] ボタンをクリックします。

7.3.3.8 メインブランチに作業用のブランチの変更をマージする (No. 10 : 開発者 A)

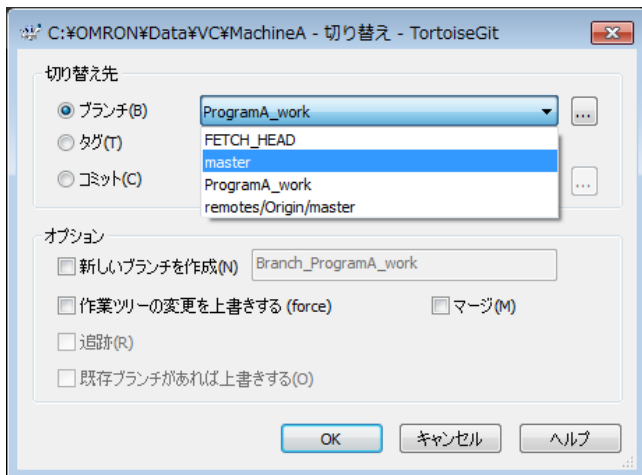
作業をするブランチをメインブランチに切り替えてから、作業用のブランチでの変更をメインブランチにマージします。

1. [バージョン管理プロジェクト] 画面でプロジェクトを右クリックして、[TortoiseGit] - [切り替え] を選択します。

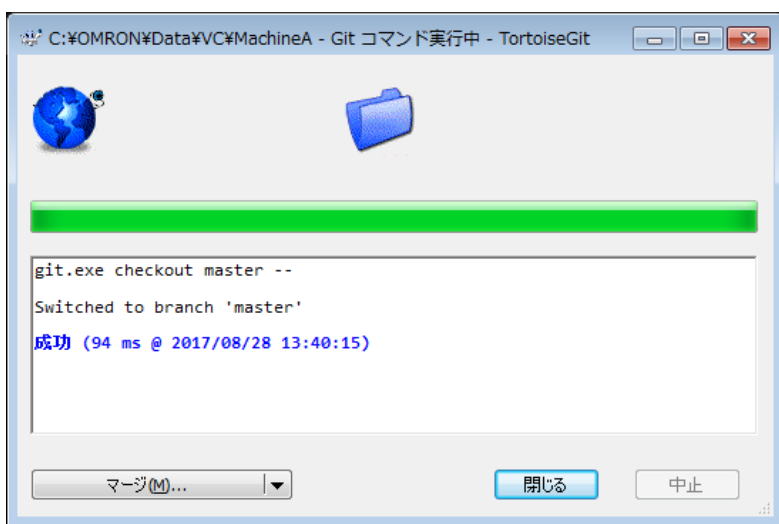


[切り替え] ダイアログボックスが表示されます。

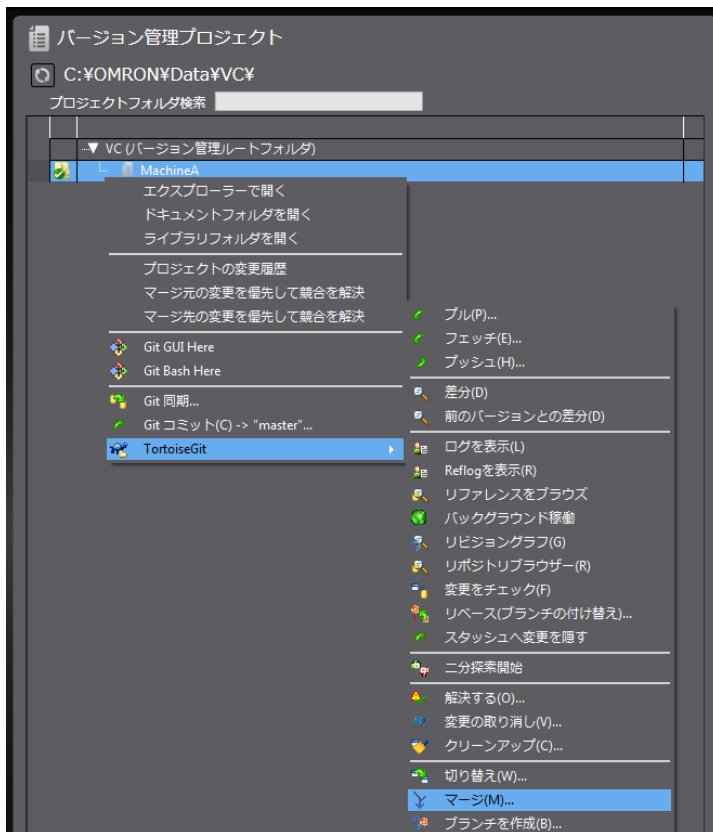
2. 「ブランチ」を選択し、メインブランチである「master」を選択して、「OK」ボタンをクリックします。



ブランチの切り替えが完了します。

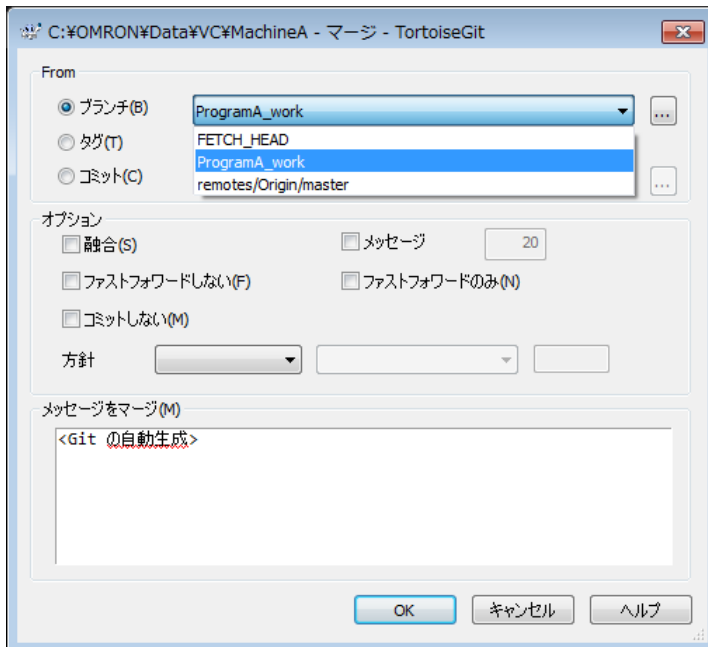


3. 「閉じる」ボタンをクリックします。
4. メインブランチにマージするため、「バージョン管理プロジェクト」画面で対象のプロジェクトを右クリックして、「TortoiseGit」-「マージ」を選択します。

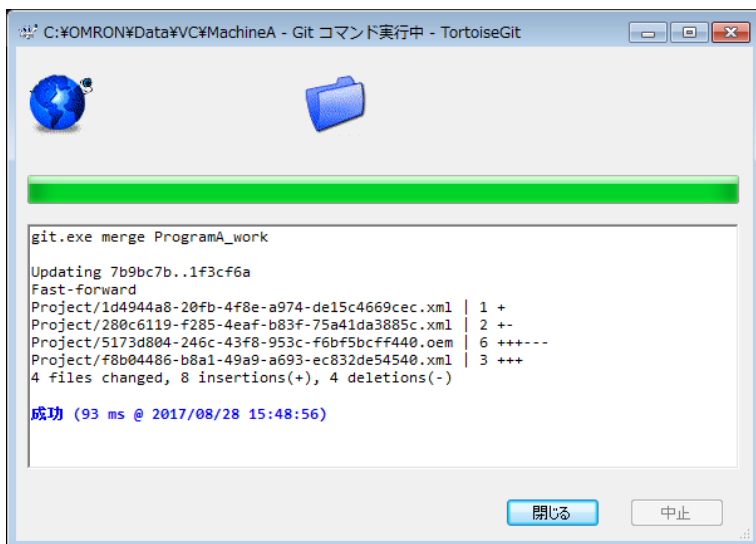


[マージ] ダイアログボックスが表示されます。

5. [ブランチ] を選択し、メインブランチ「master」へマージする作業用のブランチ「ProgramA_work」を選択して、[OK] ボタンをクリックします。

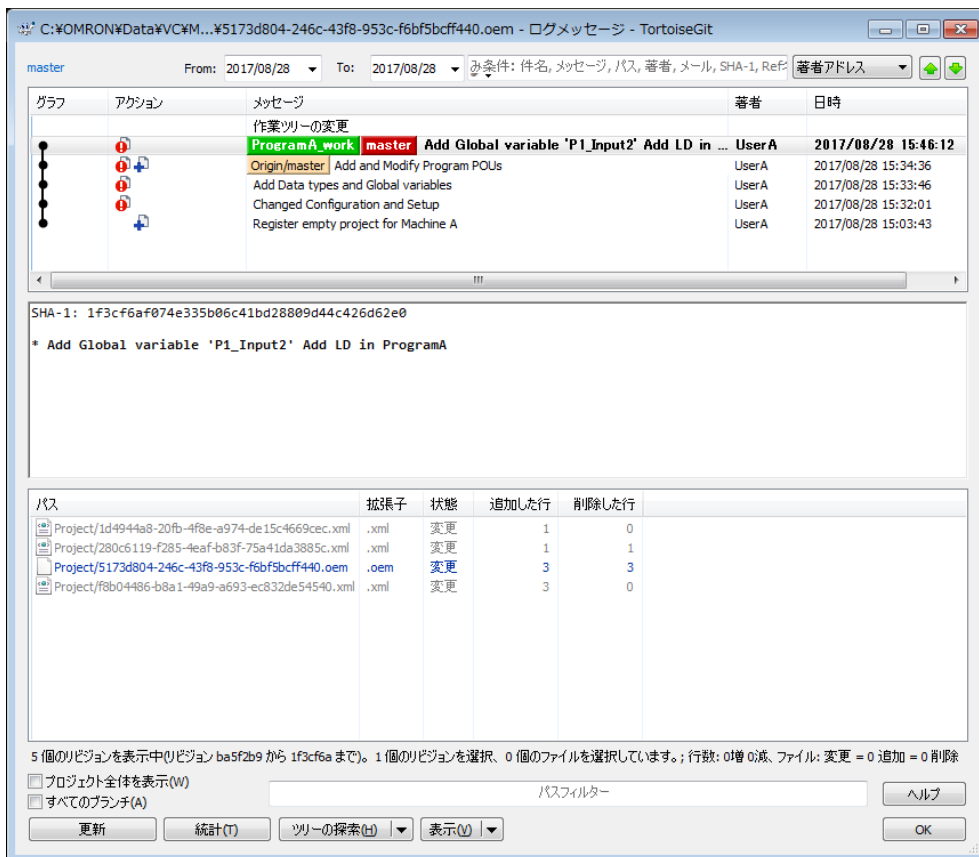


マージが完了します。



6. 「閉じる」 ボタンをクリックします。

7. マルチビューエクスプローラのフォルダアイコン、または「バージョン管理プロジェクト」画面でプロジェクトを右クリックして、「[TortoiseGit] - [ログを表示]」を選択します。
ログメッセージダイアログボックスが表示されます。

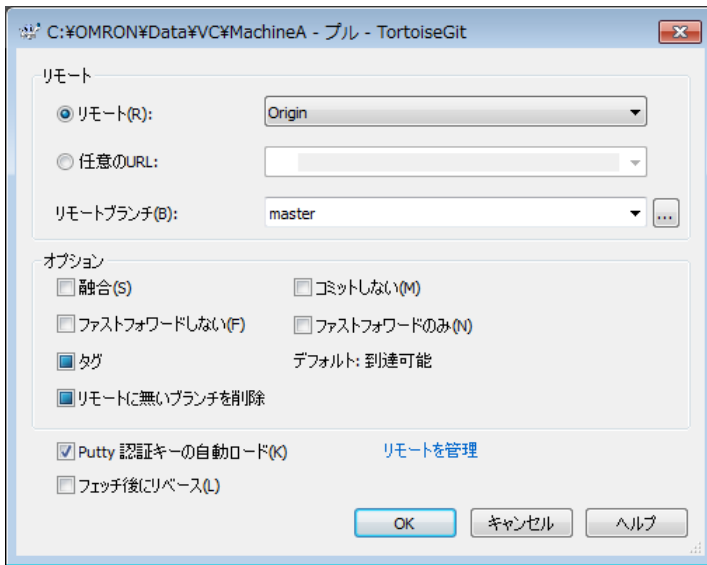


上部ペインの一番上の行に、「ProgramA_work」「master」と表示され、中段のペインに「Add Global variable 'P1_Input2', Add LD in ProgramA」が表示されています。作業用のブランチ「ProgramA_work」の最新のリリースであり、メインブランチ「master」が作業用のブランチ「ProgramA_work」と同じになっていることが確認できます。

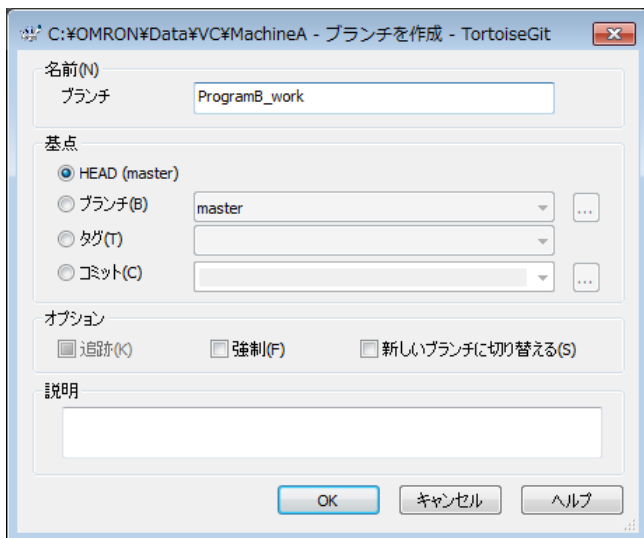
7.3.3.9 開発者 B の作業用のブランチを作成する (No. 7 : 開発者 B)

開発者 B のパソコンにローカルリポジトリを作成したあとに、他の開発者がリモートリポジトリの対象プロジェクトを更新している可能性があります。マスタとなるリモートリポジトリと開発者 B のパソコンのローカルリポジトリの同期をとるために次の操作を行います。

1. Sysmac Studio のスタートページで [バージョン管理エクスプローラ] を選択し、[バージョン管理プロジェクト] 画面を表示します。
2. MachineA を右クリックして、[TortoiseGit] - [プル] を選択します。
[プル] ダイアログボックスが表示されます
3. [リモート] に [Origin] (本説明ではリモートリポジトリのパスを「¥¥SLRA0383¥Git¥MachineA.git」と設定) が選択されていることを確認し、[OK] ボタンをクリックします。

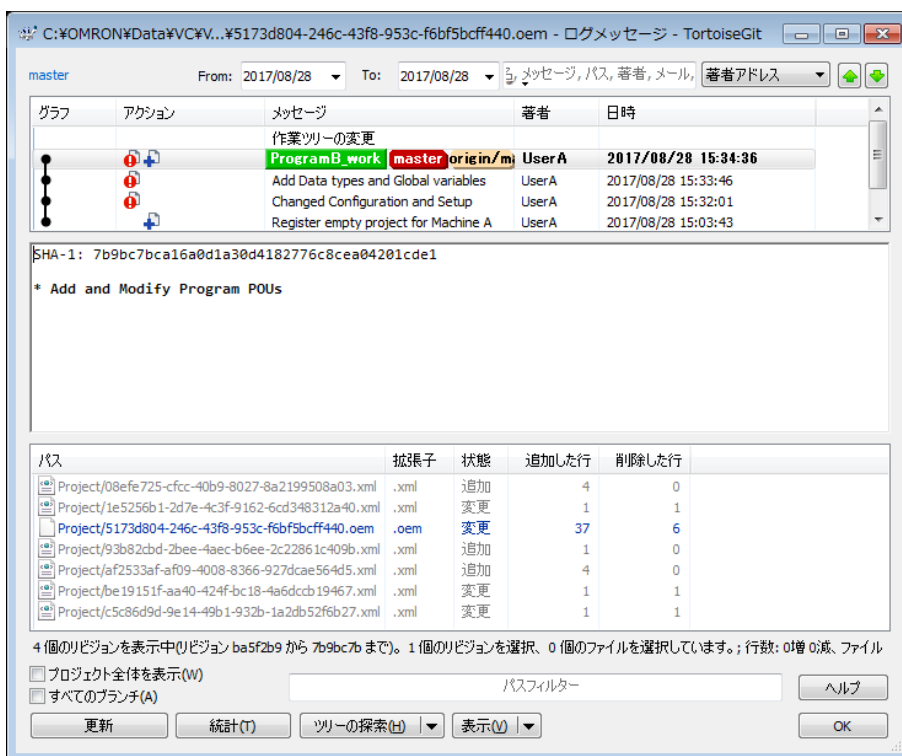


4. プル処理の完了ダイアログボックスで [閉じる] ボタンをクリックします。
5. ブランチを作成するために、マルチビューエクスプローラのフォルダアイコン、または [バージョン管理プロジェクト] 画面でプロジェクトを右クリックして、[TortoiseGit] - [ブランチを作成] を選択します。
[ブランチを作成] ダイアログボックスが表示されます。
6. 適切なブランチ名 (ここでは「ProgramB_work」) を入力し、[OK] ボタンをクリックします。
[基点] はここでは最新のリリースバージョンでかまいません。HEAD (master) を選択します。



ブランチの作成が完了します。

7. マルチビューエクスプローラのフォルダアイコン、または「バージョン管理プロジェクト」画面でプロジェクトを右クリックして、「TortoiseGit」-「ログを表示」を選択します。
「ログメッセージ」ダイアログボックスが表示されます。



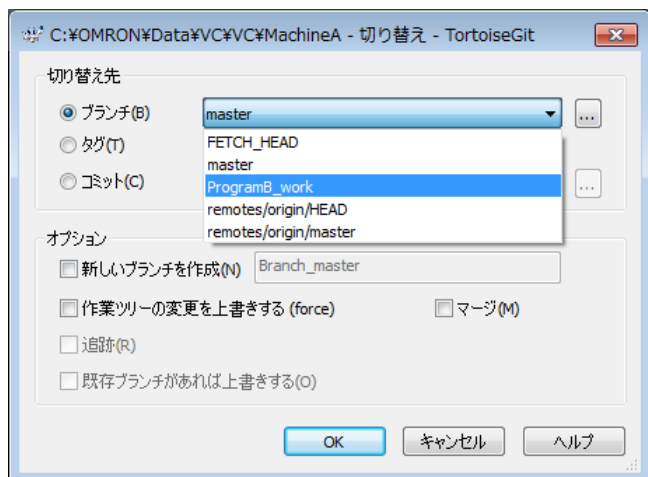
上部ペインの一番上の行に、「ProgramB_work」「master」「Origin/master」「origin/HEAD」と表示されています。作業用のブランチ「ProgramB_work」が作成されていることが確認できます。

この行の表示は、開発者 A が反映した最後のリビジョンである「Add and Modify Program POU's」という変更が、開発者 B のローカルリポジトリの作業用のブランチ「ProgramB_work」およびメイン「master」、およびリモートリポジトリ（接続先が Origin）の「master」に存在していることを示しています。

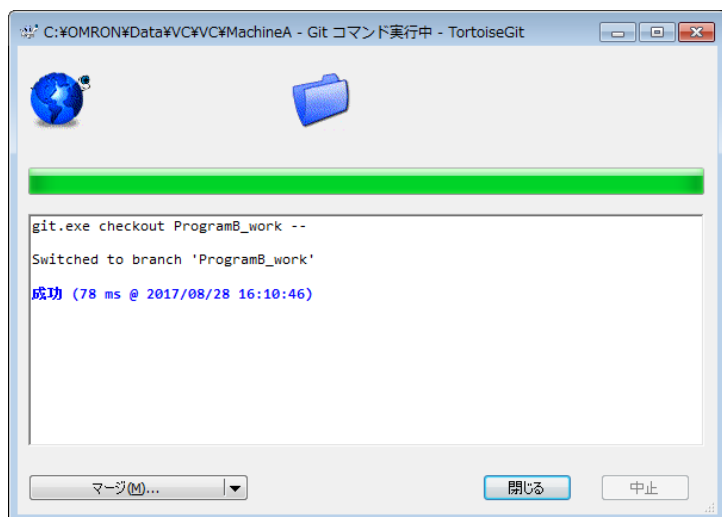
7.3.3.10 ProgramB およびグローバル変数の変更をブランチにコミットする (No. 8、9：開発者 B)

開発者 B のパソコンで作業をするブランチを切り替えて、ProgramB およびグローバル変数を変更します。

1. マルチビューエクスプローラのフォルダアイコン、または[バージョン管理プロジェクト]画面でプロジェクトを右クリックして、[TortoiseGit] - [切り替え] を選択します。
[切り替え] ダイアログボックスが表示されます。
2. [ブランチ] を選択し、「7.3.3.9 開発者 B の作業用のブランチを作成する (No.7: 開発者 B)」で作成した開発者 B の作業用のブランチである「ProgramB_work」を選択し、[OK] ボタンをクリックします。

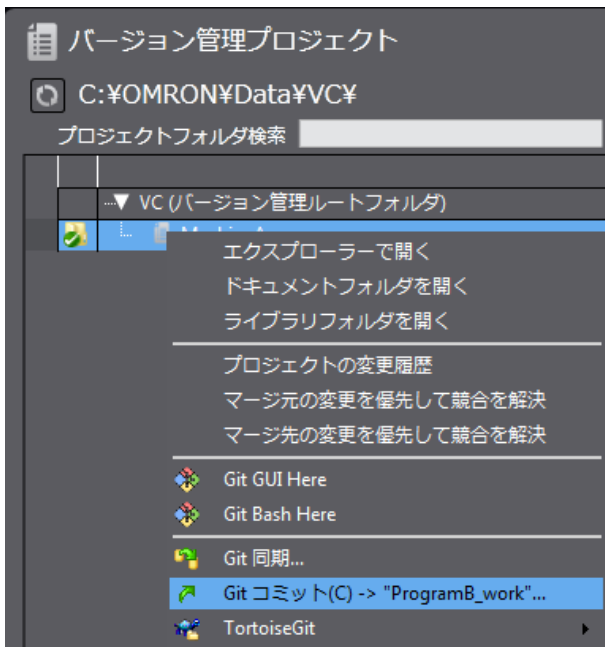


ブランチの切り替えが完了します。



3. [閉じる] ボタンをクリックします。
4. マルチビューエクスプローラのフォルダアイコン、または[バージョン管理プロジェクト]画面でプロジェクトを右クリックします。
メニューの [Git コミット ->] に続くブランチ名が、切り替えたブランチに変わっていることを確認できま

す

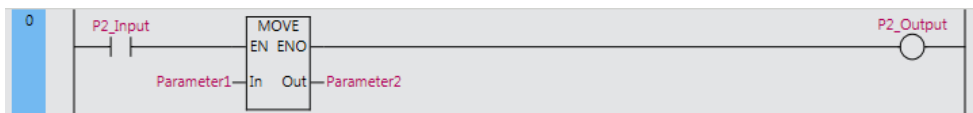


5. ProgramB を入力、グローバル変数を変更します。
以下の編集作業を行います。

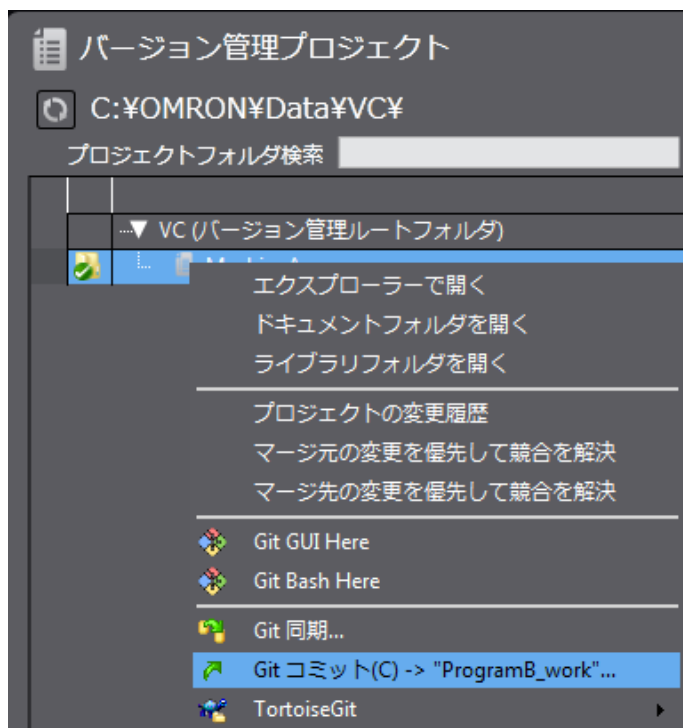
グローバル変数の Parameter1 の初期値を 200 に変更

P1_Input	BOOL	
P1_Output	BOOL	
Parameter1	INT	200
Parameter2	INT	
P2_Input	BOOL	
P2_Output	BOOL	

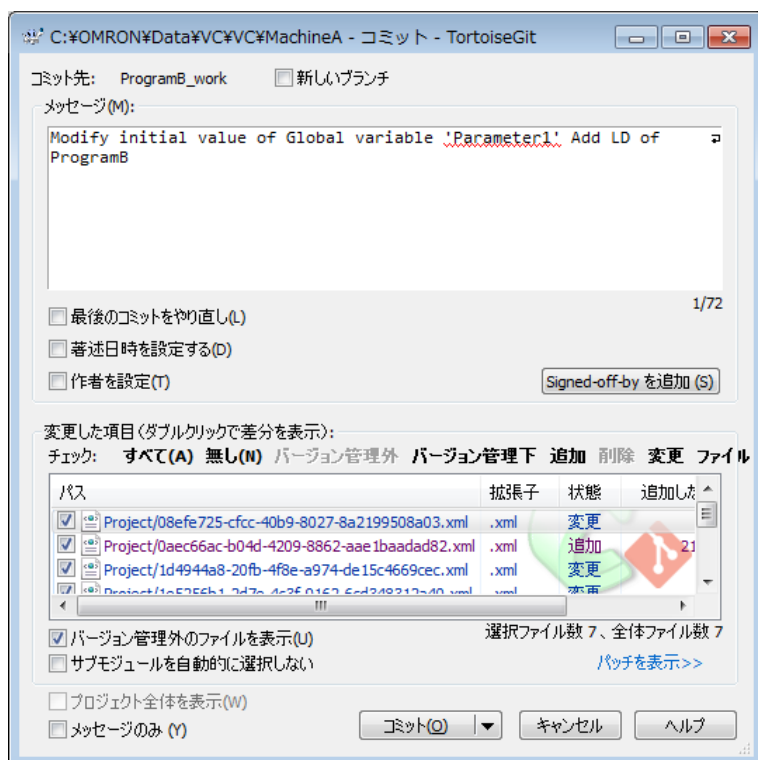
ProgramB-Section0 に以下のラダー図を入力



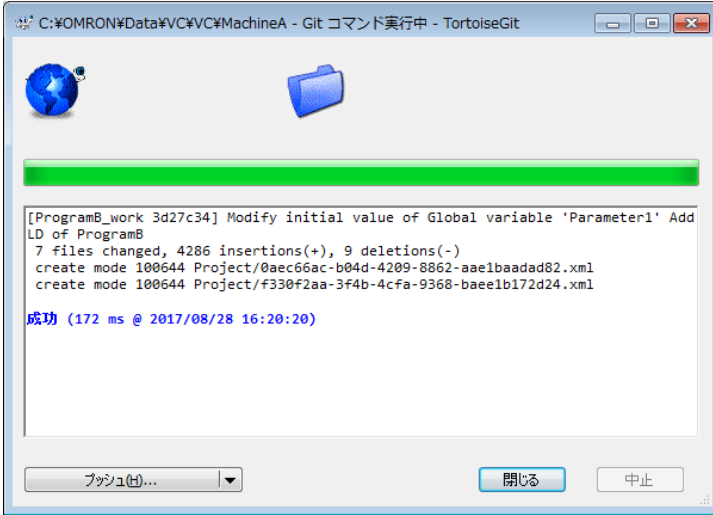
6. メインメニューから [ファイル] - [保存] を選択してプロジェクトを保存します。
7. [ファイル] - [閉じる] を選択してプロジェクトを閉じます。
8. [バージョン管理プロジェクト] 画面で対象プロジェクトを右クリックして、[Git コミット -> "ProgramB_work"] を選択します。



[コミット] ダイアログボックスが表示されます。



9. [メッセージ] エリアにコメントを入力し、[コミット] ボタンをクリックします。
コミットが完了します。



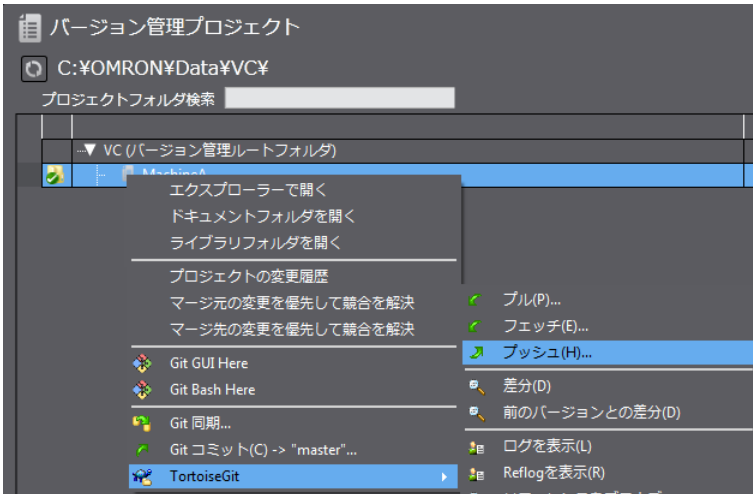
10. [閉じる] ボタンをクリックします。

7.3.3.11 開発者 A のローカルリポジトリの変更をリモートリポジトリへプッシュする (No. 11 : 開発者 A)

開発者 A の変更と開発者 B の変更をマージするためには、まず開発者 A のローカルリポジトリの変更をリモートリポジトリへプッシュし、次にリモートリポジトリの変更を開発者 B のローカルリポジトリに反映し、開発者 B の変更をマージする必要があります。

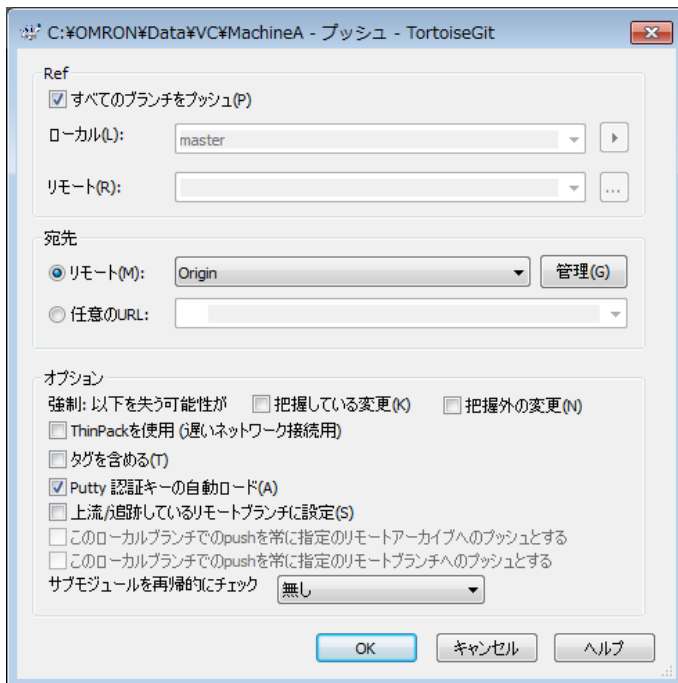
ここでは、開発者 A のローカルリポジトリの変更をリモートリポジトリへプッシュする手順を説明します。

1. [バージョン管理プロジェクト] 画面でプロジェクトを右クリックして、[TortoiseGit] - [プッシュ] を選択します。

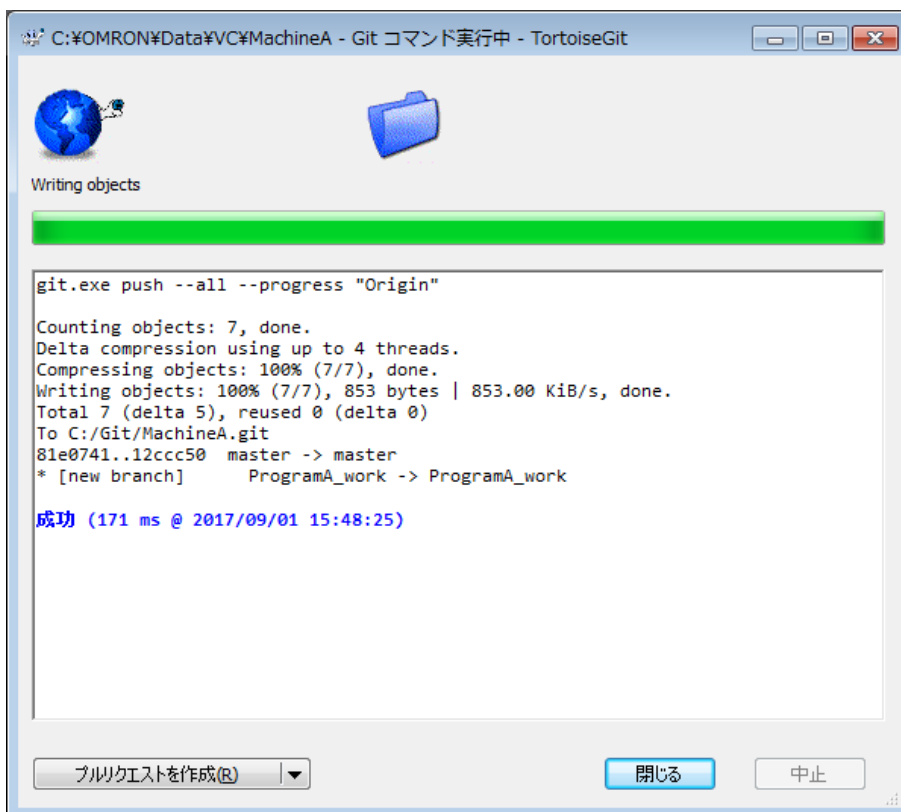


[プッシュ] ダイアログボックスが表示されます。

2. [すべてのブランチをプッシュ] チェックボックスにチェックを入れ、[宛先] の [リモート] が登録したリモート名 (ここでは Origin) になっていることを確認して、[OK] ボタンをクリックします。



プッシュ処理が完了します。



3. 「閉じる」ボタンをクリックします。

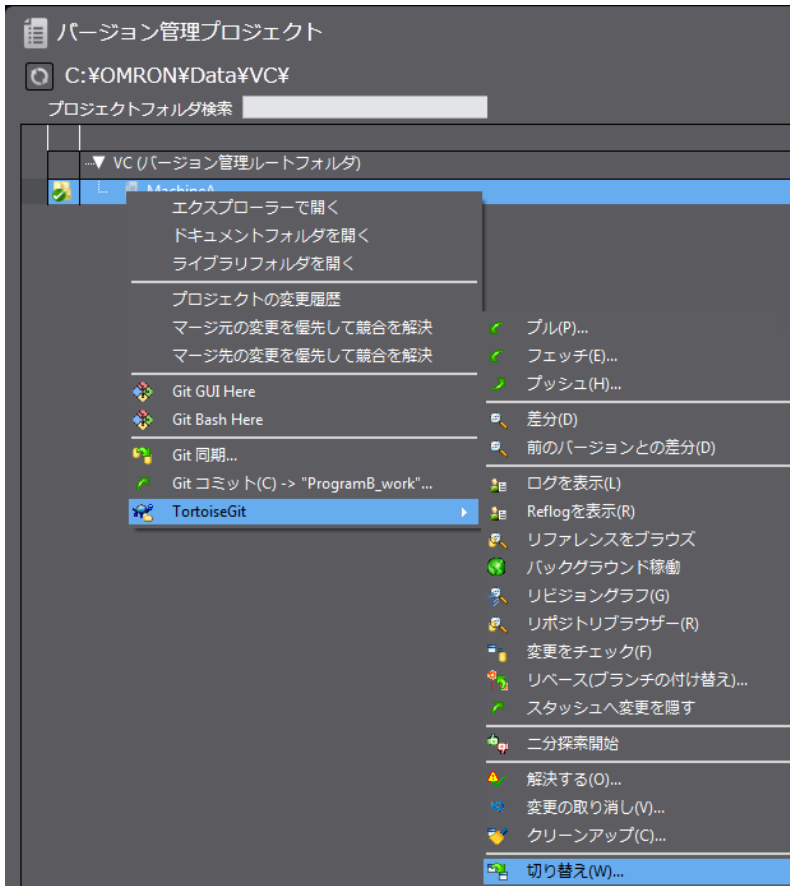
7.3.3.12 メインブランチに開発者 B の作業用のブランチの変更をマージする (No. 12 : 開発者 B)

作業をするブランチをメインブランチに切り替えてリモートリポジトリから開発者 A の変更を取得してから、開発者 B の作業用のブランチに切り替え、メインブランチの開発者 A の変更を開発者 B の作業用ブランチにマージしま

す。開発者 A の変更内容は、開発者 B の変更内容と競合するため、マージに失敗します。このとき、競合を解決してからマージを完了します。

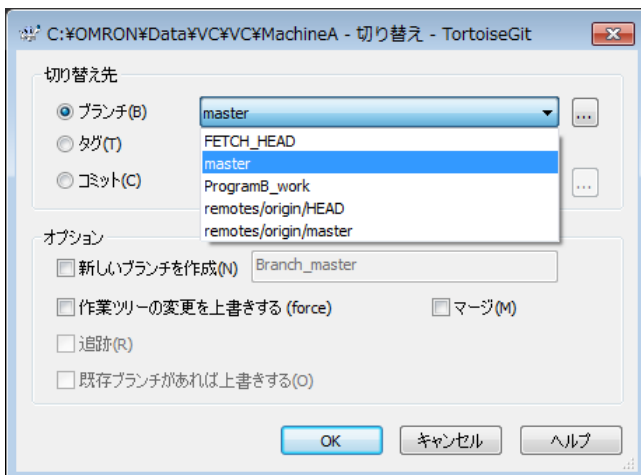
マージ完了後、作業するブランチをメインブランチに切り替えて、競合解決後のデータをメインブランチにマージします。これで一連のマージ作業は完了です。

1. 「バージョン管理プロジェクト」画面でプロジェクトを右クリックして、「TortoiseGit」 - 「切り替え」を選択します。

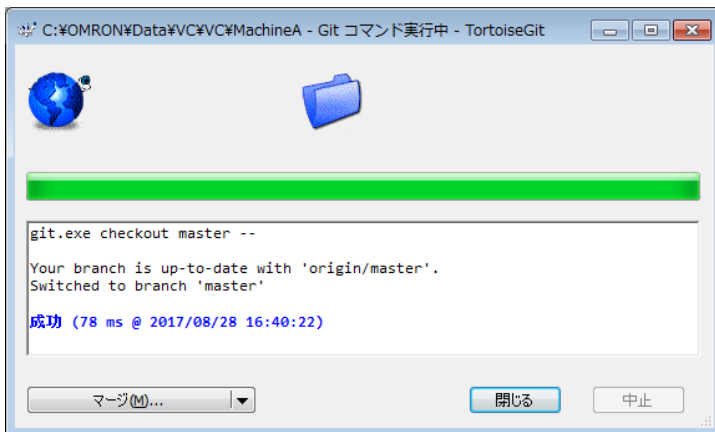


「切り替え」ダイアログボックスが表示されます。

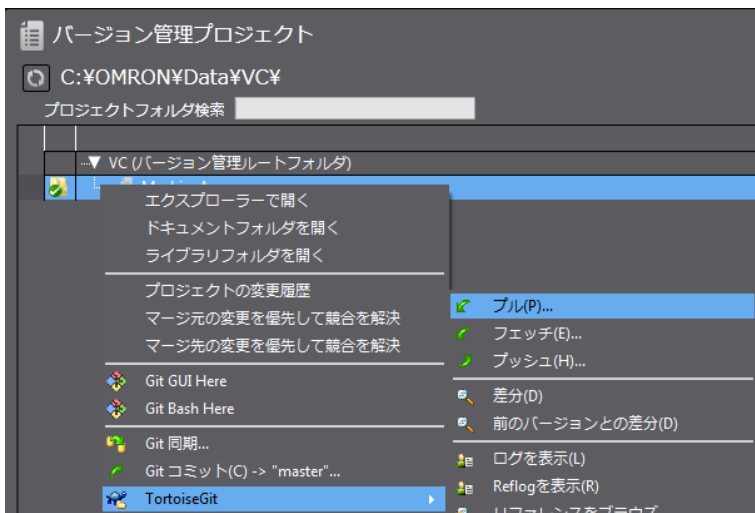
2. 「ブランチ」を選択し、メインブランチである「master」を選択して、「[OK]」ボタンをクリックします。



ブランチの切り替えが完了します。

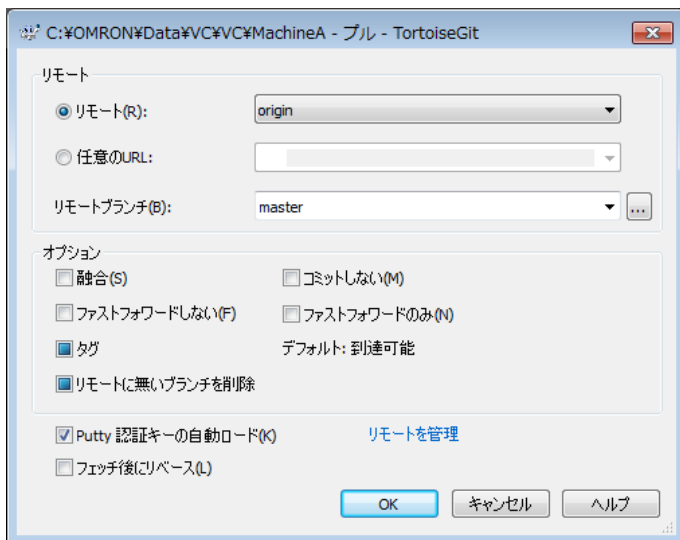


3. [閉じる] ボタンをクリックします。
4. 開発者 A の変更がリモートリポジトリに入っているため、リモートリポジトリと開発者 B のパソコンのローカルリポジトリの同期をとるために次の操作を行います。
Sysmac Studio のスタートページで [バージョン管理エクスプローラ] を選択し、[バージョン管理プロジェクト] 画面を開きます。
5. MachineA を右クリックして、[TortoiseGit] - [プル] を選択します。

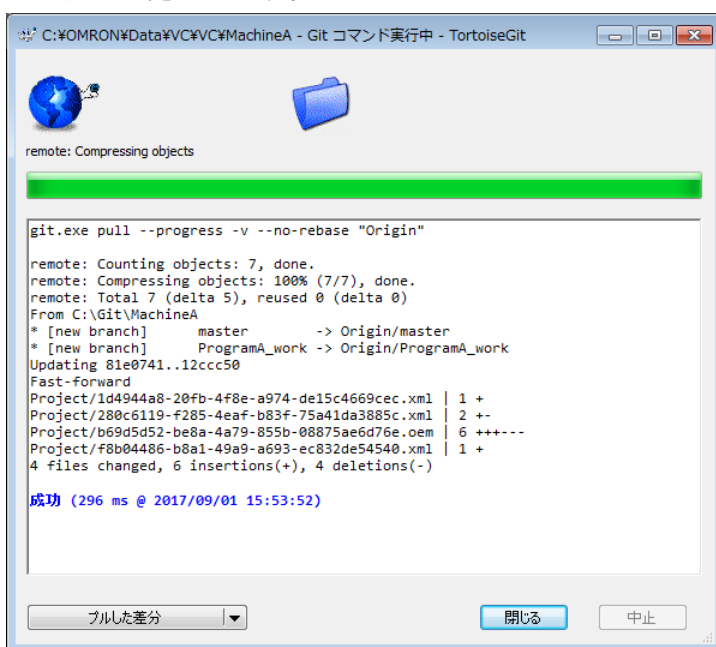


[プル] ダイアログボックスが表示されます。

6. [リモート] に [Origin] (本説明ではリモートリポジトリのパスを「¥¥SLRA0383¥Git¥MachineA.git」と設定) が選択されていることを確認し、[OK] ボタンをクリックします。

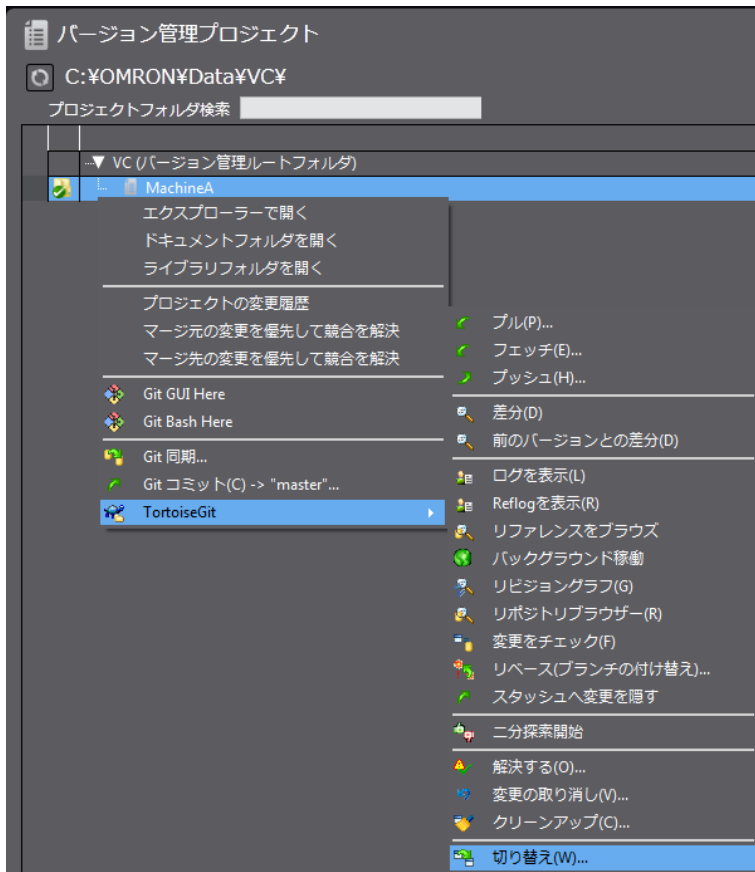


プル処理が完了します。



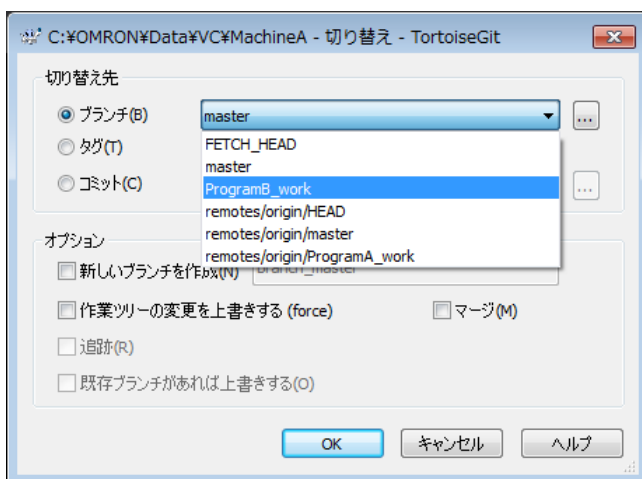
7. 「閉じる」 ボタンをクリックします。

8. 次に、メインブランチ「master」の開発者 A の変更を開発者 B の作業用のブランチ「ProgramB_work」へマージするために、ブランチ「ProgramB_work」へ切り替えます。[バージョン管理プロジェクト] 画面で対象のプロジェクトを右クリックして、[TortoiseGit] - [切り替え] を選択します。

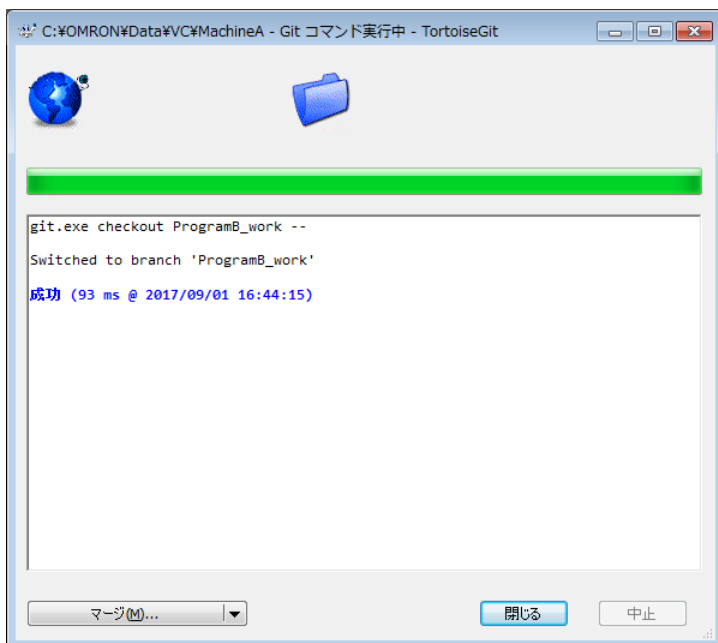


「切り替え」ダイアログボックスが表示されます。

9. 「ブランチ」を選択し、開発者 B の作業用ブランチである「ProgramB_work」を選択して、[OK] ボタンをクリックします。



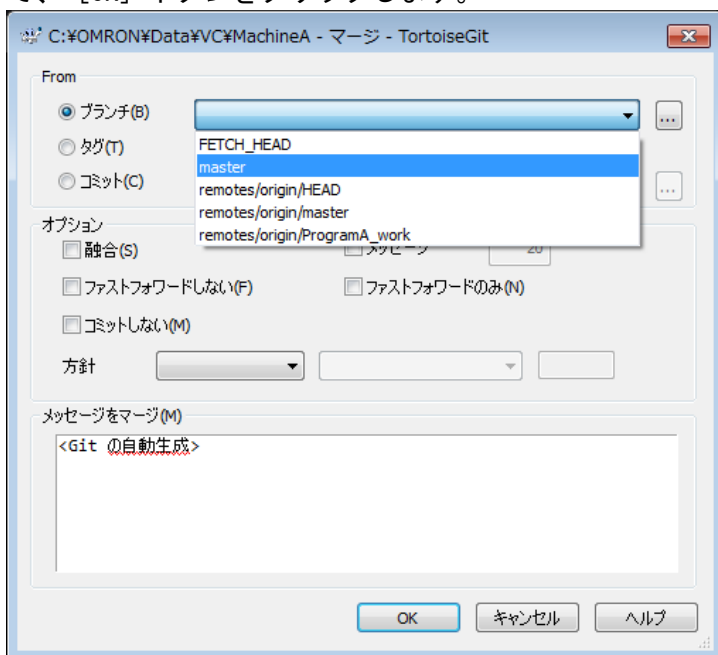
ブランチの切り替えが完了します。



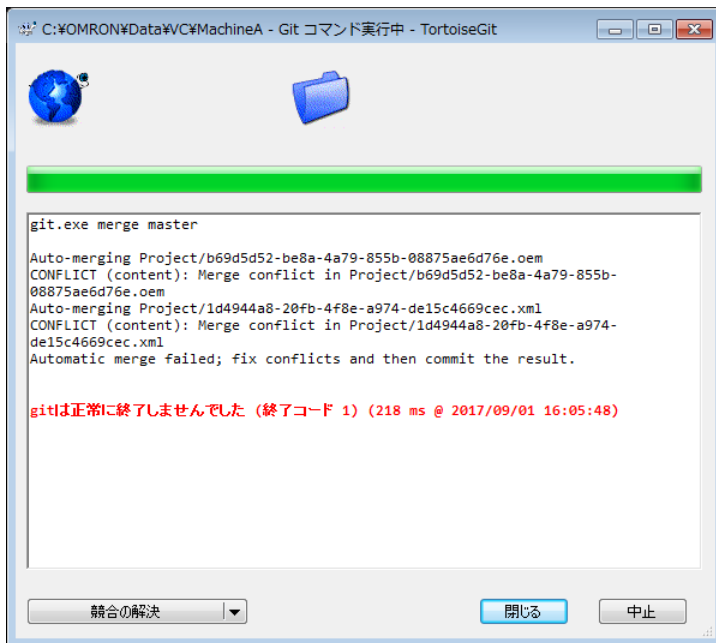
10. [閉じる] ボタンをクリックします。

11. [バージョン管理プロジェクト] 画面で対象のプロジェクトを右クリックして、[TortoiseGit] - [マージ] を選択します
[マージ] ダイアログボックスが表示されます。

12. [ブランチ] を選択し、作業用のブランチ「ProgramB_work」へマージするメインブランチ「master」を選択して、[OK] ボタンをクリックします。

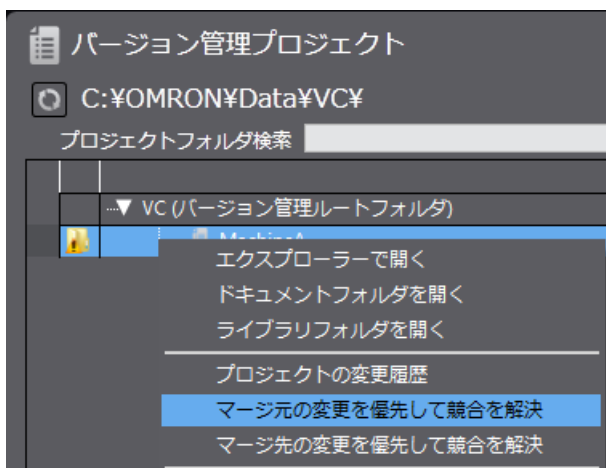


マージに失敗します。

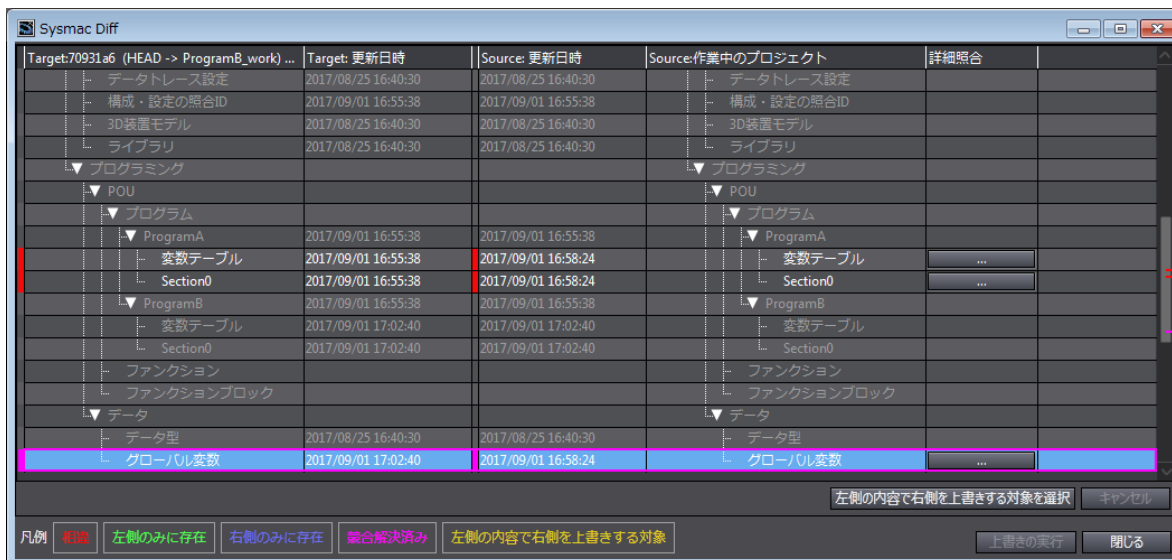


これは、開発者 A と開発者 B が同じグローバル変数に対して変更を行い、競合が発生したためです。

13. [閉じる] ボタンをクリックします。
14. つづけて、競合を解消するために [バージョン管理プロジェクト] 画面でプロジェクトを右クリックして、[TortoiseGit] - [マージ元の変更を優先して競合を解決] を選択します。

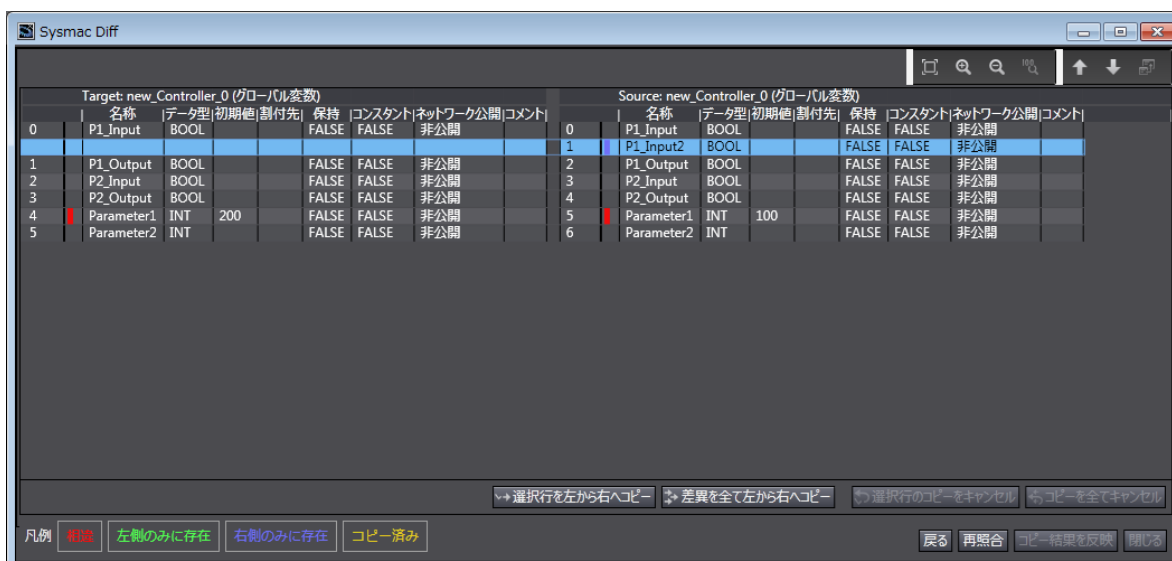


[マージ元の変更を優先して競合を解決] メニューは、マージ時に競合（ブランチ間で同じファイルが変更となっている）が発生したデータを強制的にマージするときに使用します。[マージ] ダイアログボックスの [From] セクションで指定するマージ対象のデータの変更を優先し、マージ元のデータの変更を無効なものとしてマージします。ここでは、[master] のデータの変更を優先してマージします。本説明においては、競合はグローバル変数で発生しているため、[マージ元の変更を優先して競合を解決] メニューを選択すると、マージされた結果が次の [Sysmac Diff] ダイアログボックスに表示されます。



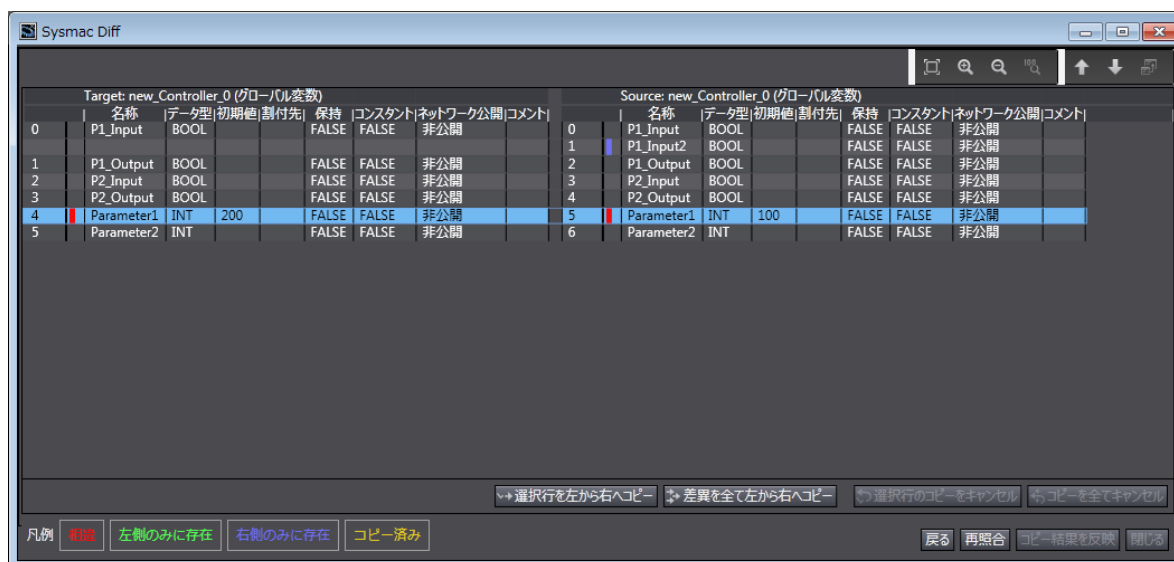
競合が発生しているデータは左端が紫色となっているグローバル変数です。左端が赤色のデータは違いがあったが競合がなくマージされているデータです。

- 競合が発生しているグローバル変数の [...] ボタンをクリックします。
[詳細照合] ダイアログボックスが表示されます。

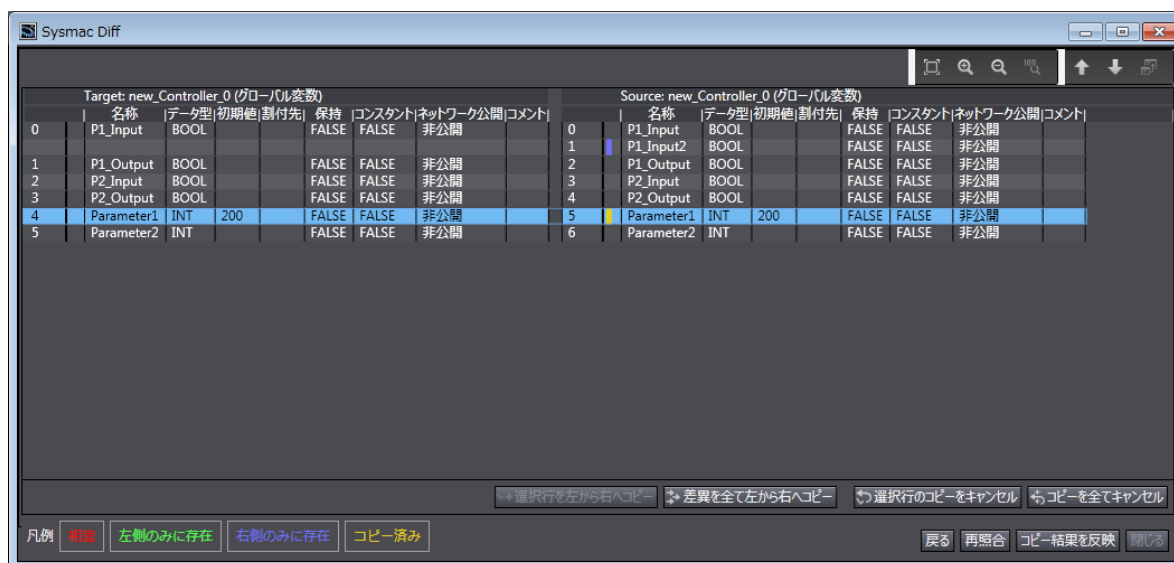


左側にマージ前の作業用の ブランチ「ProgramB_work」（開発者 B の変更）が表示され、右側にメインブランチのデータ（開発者 A の変更）のデータを優先させてマージされた編集集中のデータが表示されます。

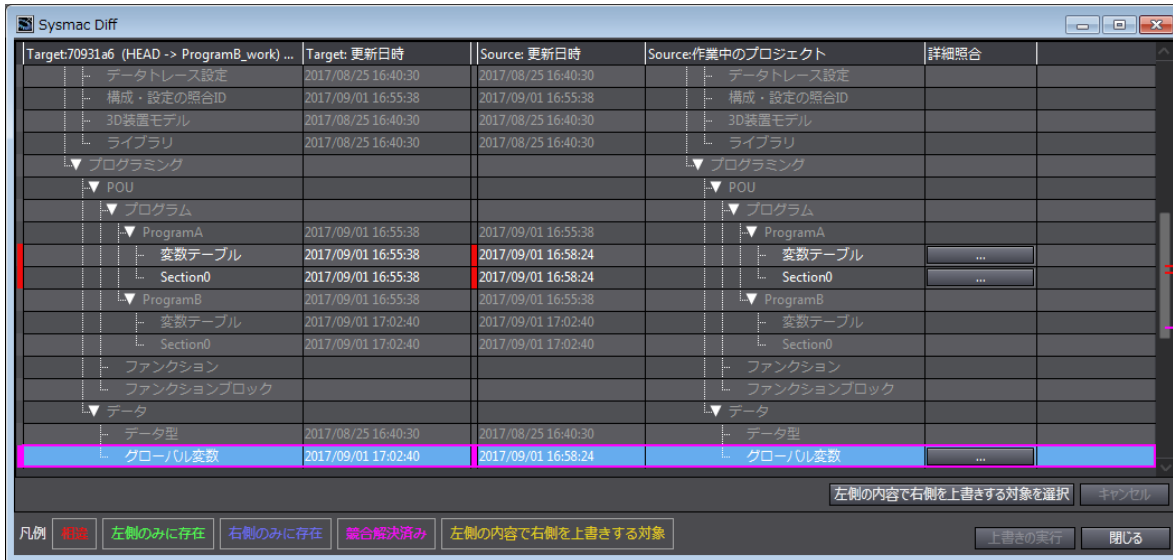
16. ここでは、4 行目の変数「Parameter1」の値を残す必要があります。そのために、まず 4 行目を選択します。



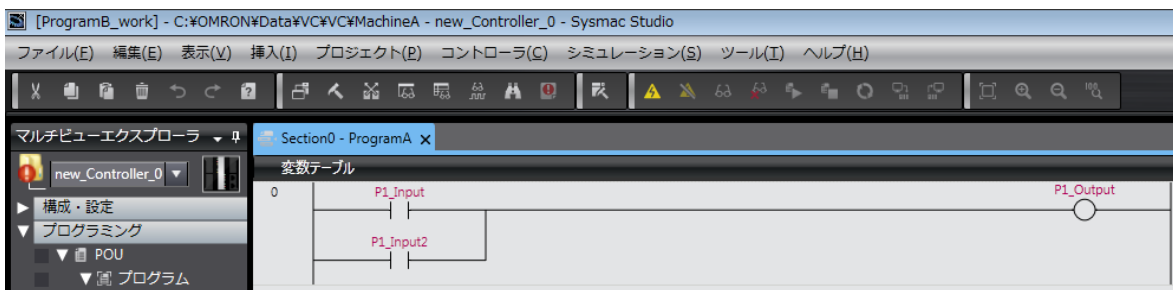
17. 「選択行を左から右へコピー」ボタンをクリックします。変数「Parameter1」の値が、右側の編集集中のデータの欄に入りました。



18. [コピー結果を反映] ボタンをクリックし、その後 [戻る] ボタンをクリックします。
[Sysmac Diff] ダイアログボックスに戻ります。

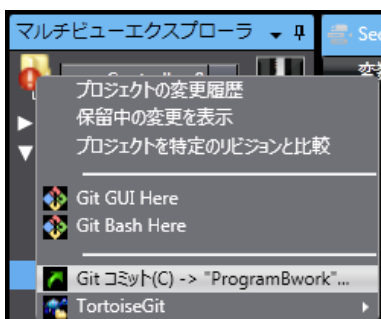


19. 他に競合が発生している箇所（紫色の箇所）がないか確認し、[閉じる] ボタンをクリックします。
プロジェクトを開くと、開発者 A が追加した変数「P1_Input2」が作業用の ブランチ「ProgramB_work」に追加されていることを確認できます。



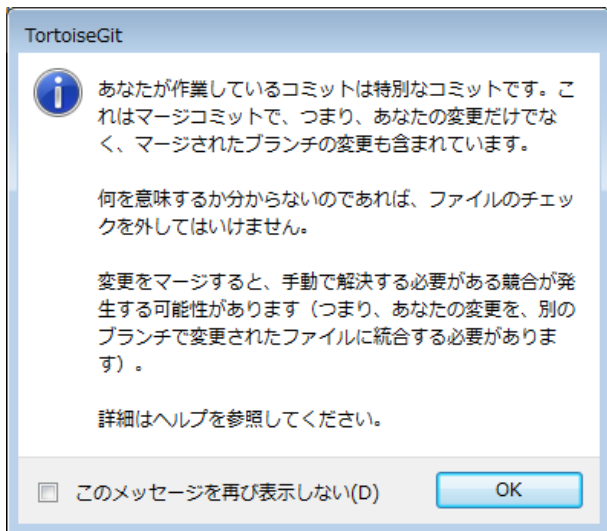
これで競合を解決し、設計者 A と設計者 B の変更のマージが完了しました。

20. つづけて、マージ結果をローカルリポジトリへコミットします。マルチビューエクスプローラのフォルダアイコンを右クリックして、[Git コミット -> "ProgramBwork" ...] を選択します。



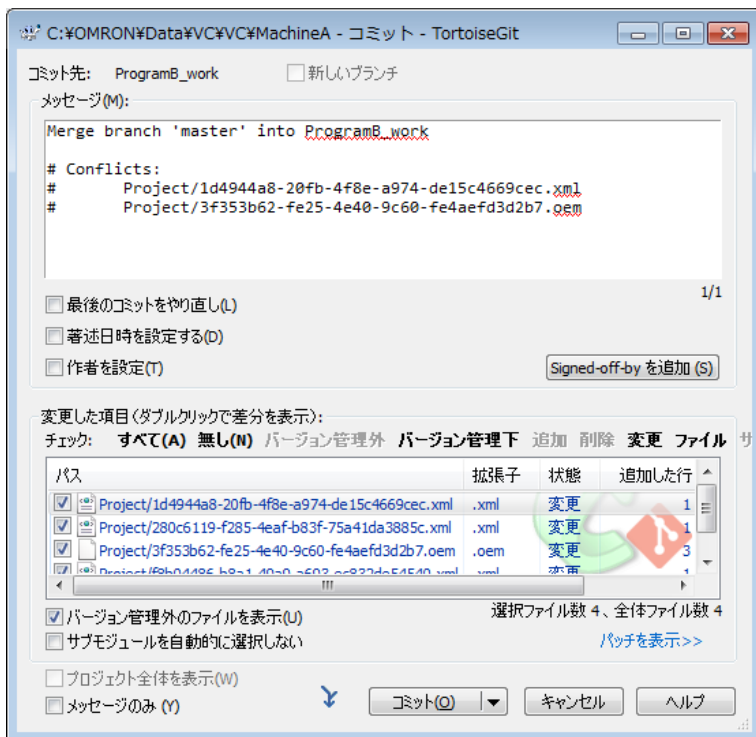
プロジェクトを保存していない場合は、プロジェクトの保存確認ダイアログボックスが表示されるので、[はい] ボタンをクリックします。

コミットダイアログボックスとともに、以下のダイアログボックスが表示されます。

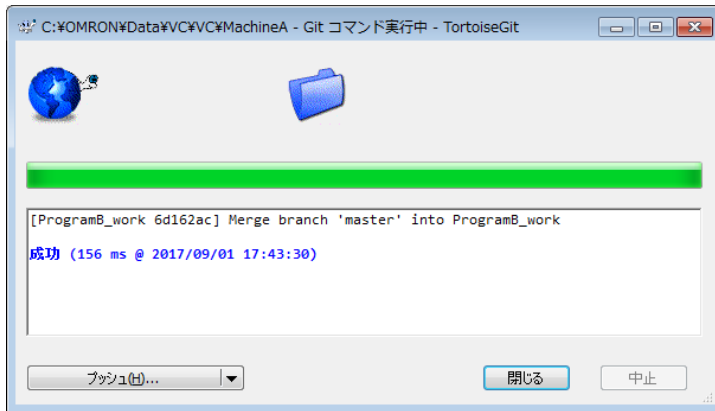


このダイアログボックスは、マージ時に競合が発生したブランチをマージする際に、注意を促すために表示されます。

21. [OK] ボタンをクリックします。
22. [コミット] ダイアログボックスの[メッセージ] エリアには、マージ時に競合が発生した旨のコメントが自動的に入力されます。必要によりコメントを追加・変更し、[コミット] ボタンをクリックします。



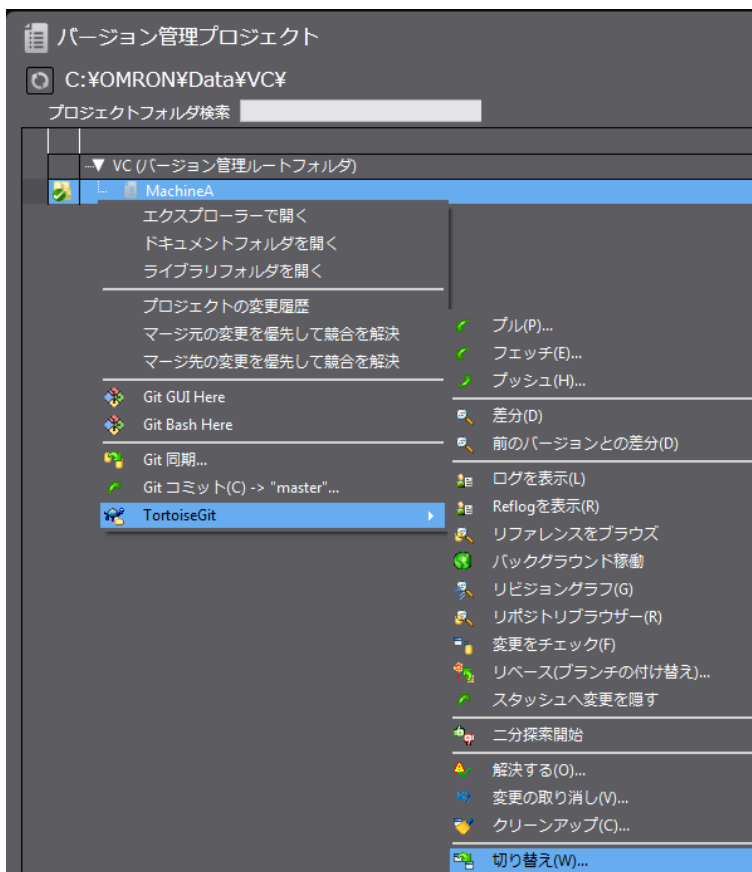
コミットが完了します。



23. [閉じる] ボタンをクリックします。

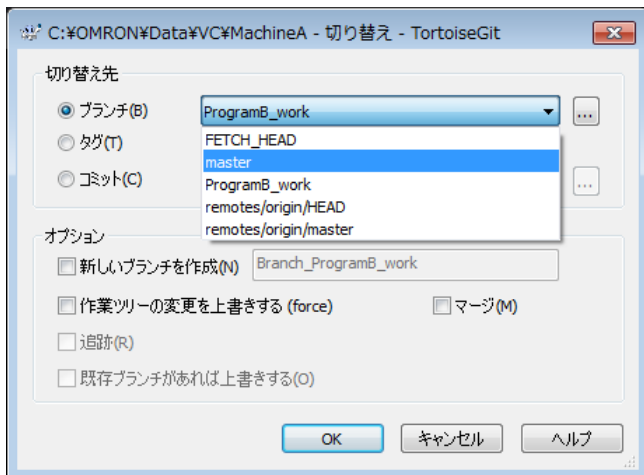
24. つづけて、作業をするブランチをメインブランチへ切り替えて、マージ後のブランチ「ProgramB_work」のデータをメインブランチへ反映します。

[バージョン管理プロジェクト] 画面で対象のプロジェクトを右クリックして、[TortoiseGit] - [切り替え] を選択します。

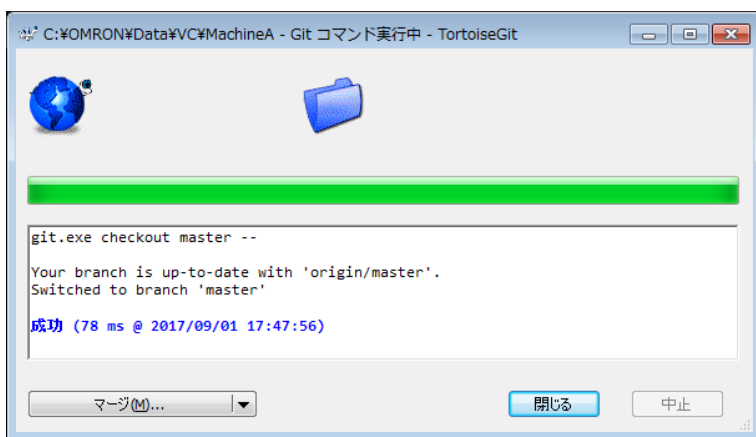


[切り替え] ダイアログボックスが表示されます。

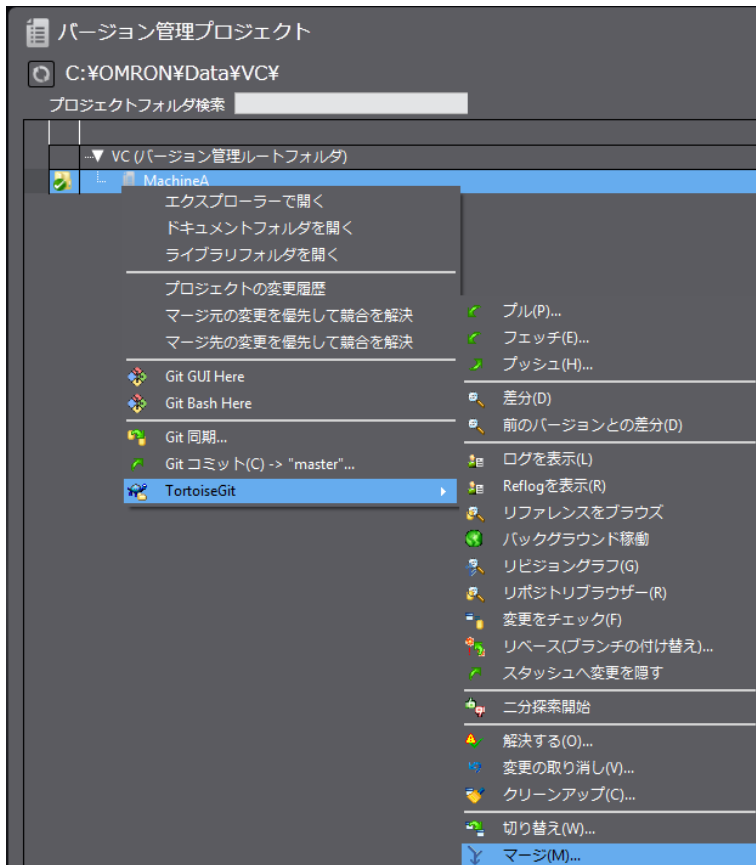
25. [ブランチ] を選択し、メインブランチ「master」を選択して、[OK] ボタンをクリックします。



ブランチの切り替えが完了します。

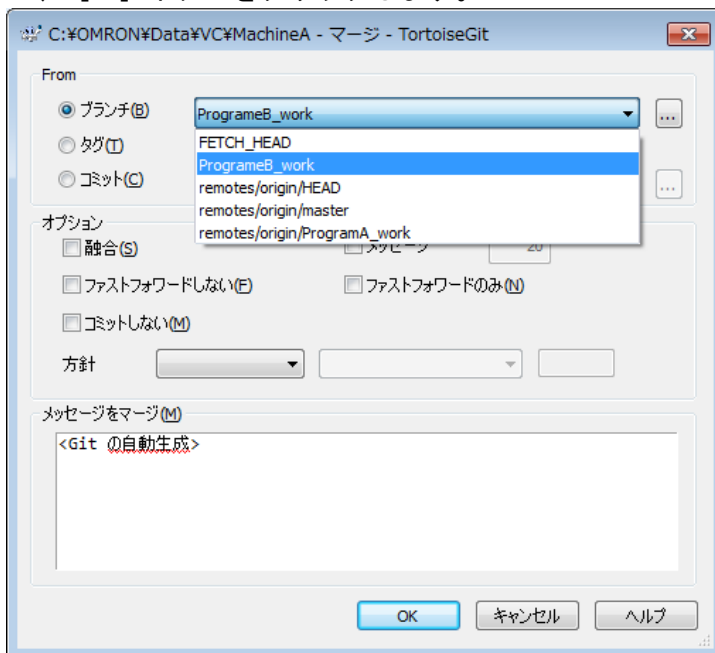


26. [閉じる] ボタンをクリックします。
27. [バージョン管理プロジェクト] 画面で対象のプロジェクトを右クリックして、[TortoiseGit] - [マージ] を選択します

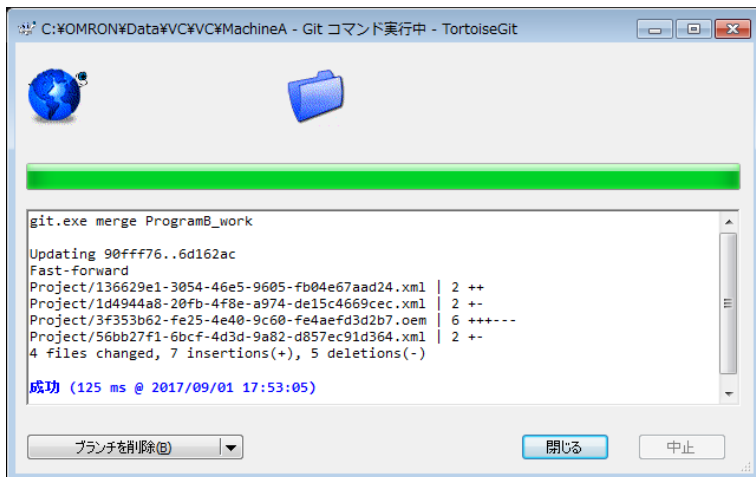


[マージ] ダイアログボックスが表示されます。

28. [ブランチ] を選択し、メインブランチ「master」へマージする作業用のブランチ「ProgramB_work」を選択して、[OK] ボタンをクリックします。



マージ処理の完了ダイアログボックスが表示されます。

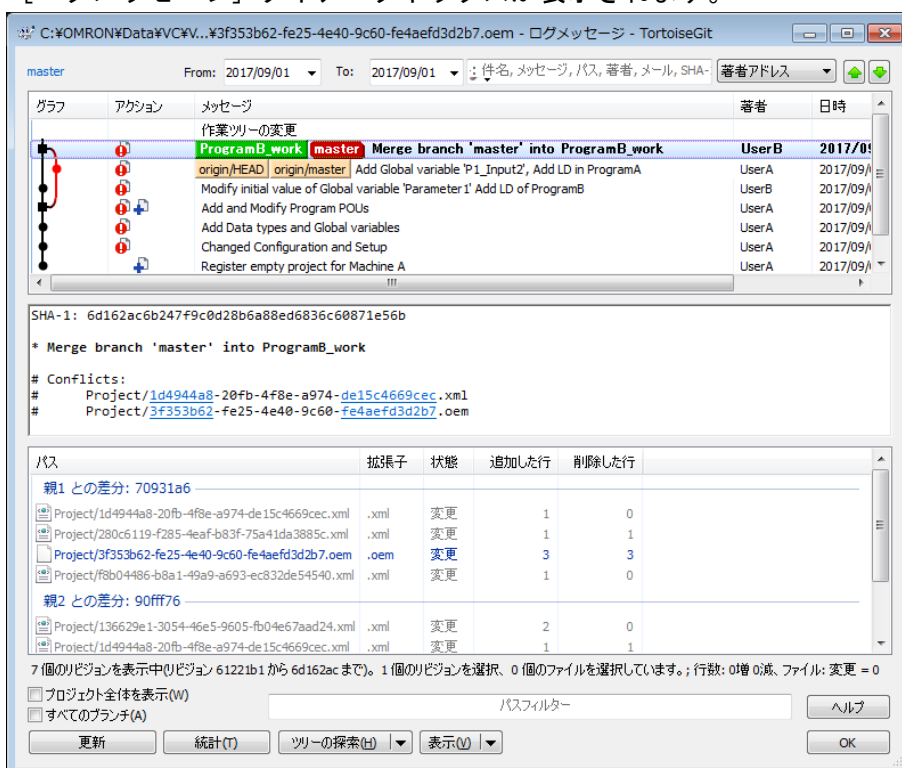


メインブランチの内容は、設計者 A と設計者 B の変更の競合を解決してマージが完了した後の内容に置き換えられます。

29. [閉じる] ボタンをクリックします。

30. マルチビューエクスプローラのフォルダアイコンを右クリックして、[TortoiseGit] - [ログを表示] を選択します。

[ログメッセージ] ダイアログボックスが表示されます。

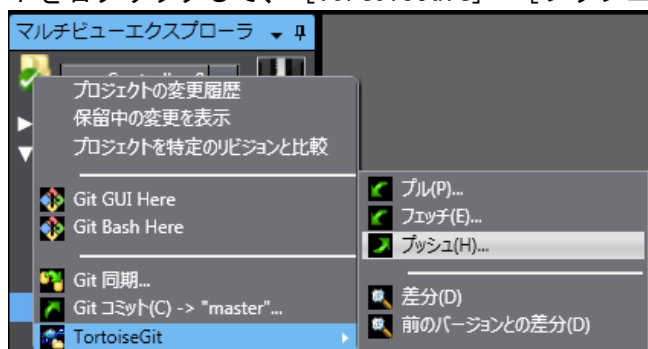


上部ペインを見ると、開発者 B が作業用のブランチ「PrgramB_work」で行った編集内容が、メインブランチ「master」に取り込まれていることが確認できます。[OK] ボタンをクリックしてダイアログボックスを閉じます。

7.3.3.13 開発者 B のメインブランチの変更をリモートリポジトリへプッシュする (No. 13 : 開発者 B)

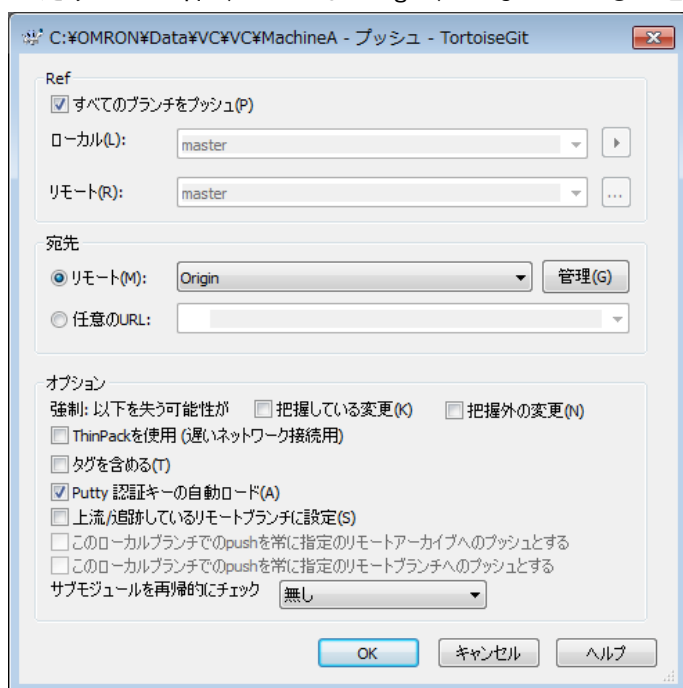
開発者 B のパソコンでマージしたプロジェクトの変更内容を、リモートリポジトリへ反映（プッシュ）します。

1. マルチビューエクスプローラのフォルダアイコン、または [バージョン管理プロジェクト] 画面でプロジェクトを右クリックして、[TortoiseGit] - [プッシュ] を選択します。

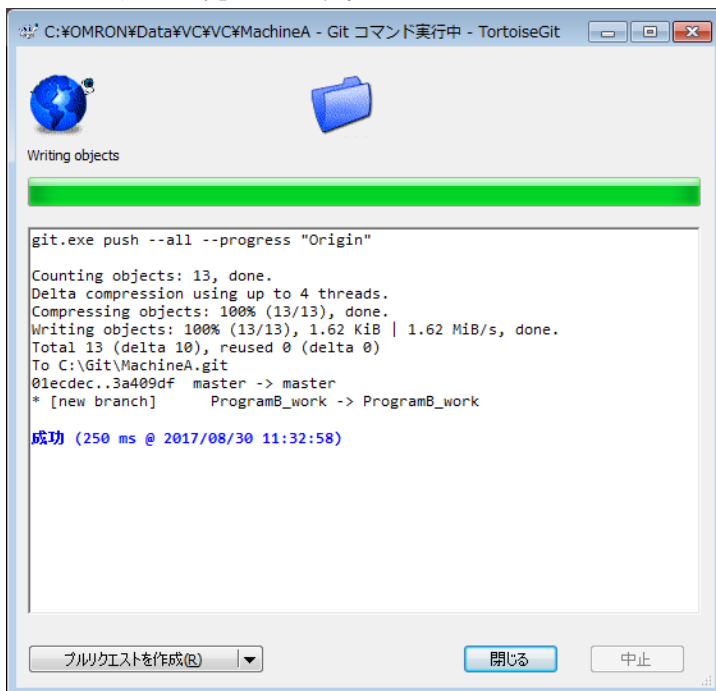


[プッシュ] ダイアログボックスが表示されます。

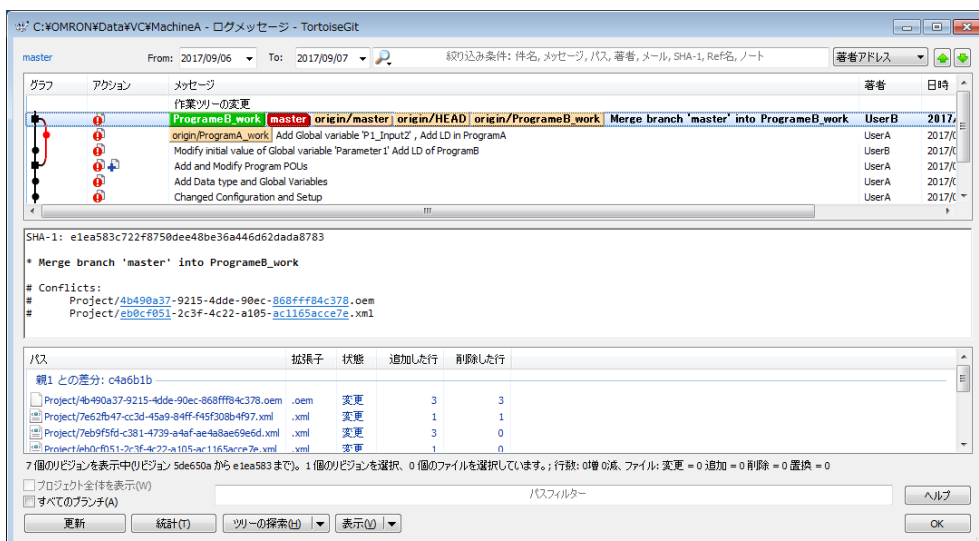
2. [すべてのブランチをプッシュ] チェックボックスにチェックを入れ、[宛先] の [リモート] が、登録していたリモート名（ここでは Origin）になっていることを確認して、[OK] ボタンをクリックします。



プッシュ処理が完了します。



3. 「閉じる」 ボタンをクリックします。
4. マルチビューエクスプローラのフォルダアイコンを右クリックして、[TortoiseGit] - [ログを表示] を選択します。
[ログメッセージ] ダイアログボックスが表示されます。



上部ペインを見ると、開発者 B が作業用のブランチ「PrgramB_work」で行った編集内容が、メインブランチ「master」に取り込まれており、かつ、リモートリポジトリ (origin) の「master」も同期していることが確認できます。[OK] ボタンをクリックしてダイアログボックスを閉じます。

7.4 Sysmac Studio プロジェクトの派生開発

Sysmac Studio プロジェクトの派生開発における操作の流れを、次の例で説明します。
ここでは、リモートリポジトリを共有フォルダを使用して共有する運用形態で説明します。

7.4.1 派生開発の例

装置 A から装置 B を派生開発します。装置 B の開発途中で判明した装置 A と装置 B の共通プログラムの不具合修正を、装置 B のプロジェクトへ反映して完成させます。

7.4.1.1 プロジェクトの説明

ここでは、一人の開発者が装置 A のプロジェクトを元に派生装置 B のプロジェクトを開発するものとします。
プロジェクト A: 装置 A のプロジェクトです。このプロジェクトを元に派生装置 B のプロジェクトを作成します。
プロジェクト B: 装置 A のプロジェクトを元にした、構成・設定が異なる装置 B のプロジェクトです。プログラムはプロジェクト A と共通とします。

7.4.1.2 シナリオ

装置のプロジェクトは、共通プログラム管理用の「master」、装置 A 用の「ProductA」、装置 B 用の「ProductB」というブランチを作成して管理します。共通プログラムの変更は「master」で行い、その変更を「ProductA」、「ProductB」へ反映します。

1. 装置 A の開発において、プロジェクト A を新規に作成し、バージョン 1.0 として装置 A に組み込みます。このプロジェクト A を派生開発のマスタプロジェクトとして master に登録しておきます。
2. 装置 B の開発において、プロジェクト A のバージョン 1.0 を元にプロジェクト B を作成し、[構成・設定] を変更します。
3. 装置 A のプログラムに不具合があることが判明したため、修正を行います。マスタプロジェクトのプログラムを修正し、修正内容を装置 A にバージョン 1.1 として組み込みます。
4. 同様に、開発途中のプロジェクト B に不具合修正の内容を組み込み、装置 B にバージョン 1.0 として組み込みます。

7.4.2 バージョン管理システム上の操作の流れ

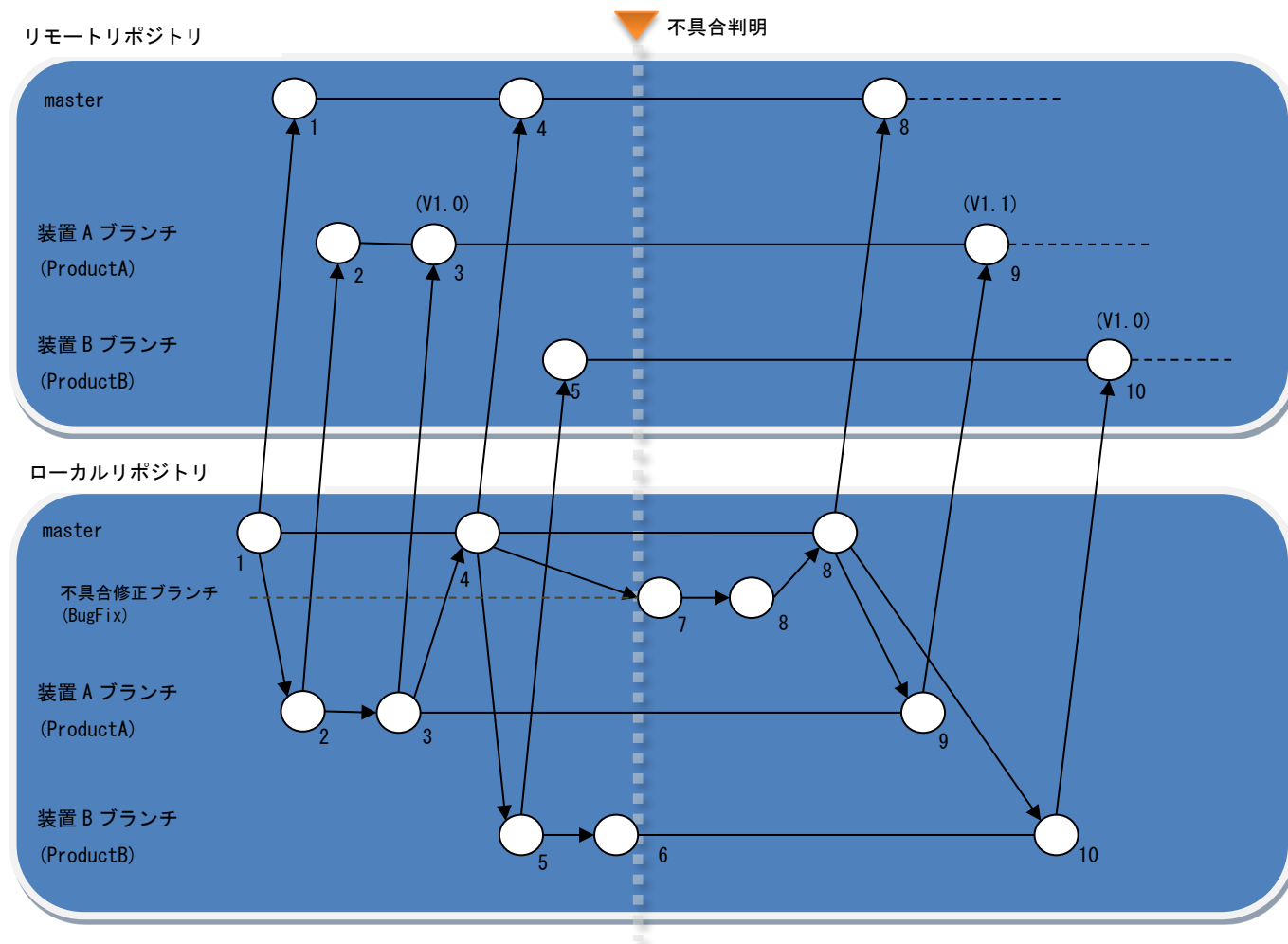
No.	装置 A 開発	装置 B 開発	不具合修正	作業手順の参照先
1	リモートリポジトリにプロジェクトを登録する	-	-	7.4.3.1 リモートリポジトリにプロジェクトを登録する (No.1: 装置 A 開発)
2	ローカルリポジトリに装置 A 用のブランチを作成して、リモートリポジトリへプッシュする ※装置 A の開発はこのブランチで行います。	-	-	7.4.3.2 ローカルリポジトリに装置 A 用のブランチを作成して、リモートリポジトリへプッシュする (No.2: 装置 A 開発)
3	装置 A のプログラムを作成してコミットし、リモートリポジトリへプッシュする ※このプロジェクトにバージョン 1.0 としてタグをつけて、装置 A に組み込みます。	-	-	7.4.3.3 装置 A のプログラムを作成してコミットし、リモートリポジトリへプッシュする (No.3: 装置 A 開発)

- : 作業なし

No.	装置 A 開発	装置 B 開発	不具合修正	作業手順の参照先
4	装置 A の変更をローカルリポジトリの master へマージし、リモートリポジトリへプッシュする ※これ以降の派生開発のマスタープロジェクトとして、装置 A のプロジェクトを master へ登録しておきます。	-	-	7.4.3.4 装置 A の変更をローカルリポジトリの master へマージし、リモートリポジトリへプッシュする (No. 4 : 装置 A 開発)
5	-	ローカルリポジトリに装置 B 用のブランチを作成して、リモートリポジトリへプッシュする ※装置 B の開発はこのブランチで行います。	-	7.4.3.5 ローカルリポジトリに装置 B 用のブランチを作成して、リモートリポジトリへプッシュする (No. 5 : 装置 B 開発) 装置 B の開発はこのブランチで行います。
6	-	装置 B の構成・設定を変更してコミットする ※この直後に共通プログラムに不具合が判明し、修正を行うものとします。	-	7.4.3.6 装置 B の構成・設定を変更してコミットする (No. 6 : 装置 B 開発) この直後に共通プログラムに不具合が判明し、修正を行うものとします。
7	-	-	ローカルリポジトリの master から、不具合修正のためのブランチを作成する	7.4.3.7 ローカルリポジトリの master から、不具合修正のためのブランチを作成する (No. 7 : 不具合修正)
8	-	-	不具合修正をコミットして master へマージし、リモートリポジトリへプッシュする	7.4.3.8 不具合修正をコミットして master へマージし、リモートリポジトリへプッシュする (No. 8 : 不具合修正)
9	不具合修正の内容を、装置 A 用のブランチにマージして、リモートリポジトリへプッシュする ※このプロジェクトにバージョン 1.1 としてタグをつけて、装置 A に組み込みます。	-	-	7.4.3.9 不具合修正の内容を、装置 A 用のブランチにマージして、リモートリポジトリへプッシュする (No. 9 : 装置 A 開発) このプロジェクトにバージョン 1.1 としてタグをつけて、装置 A に組み込みます。
10	-	不具合修正の内容を、装置 B 用のブランチにマージして、リモートリポジトリへプッシュする ※このプロジェクトにバージョン 1.0 としてタグをつけて、装置 B に組み込みます。	-	7.4.3.10 不具合修正の内容を、装置 B 用のブランチにマージして、リモートリポジトリへプッシュする (No. 10 : 装置 B 開発) このプロジェクトにバージョン 1.0 としてタグをつけて、装置 B に組み込みます。

- : 作業なし

リポジトリとブランチの関係は次のとおりです。



7.4.3 操作手順

「7.4.1.2 シナリオ」の内容に沿って、操作手順を説明します。

7.4.3.1 リモートリポジトリにプロジェクトを登録する (No. 1 : 装置 A 開発)

1. リモートリポジトリの作成

成果物を管理するために、リモートリポジトリを作成します。

ここでは 開発者のパソコンに「C:\¥Git¥MachineA.git」というフォルダ名のリモートリポジトリを作成します。

手順の詳細は、「5.6 共有フォルダとリモートリポジトリの作成」を参照してください。

2. ローカルリポジトリの登録と、ベースとなるプロジェクトの作成、保存

新規のプロジェクトを作成し、そのプロジェクトをローカルリポジトリに登録します。

ここでは、「MachineA」というプロジェクトを作成してローカルリポジトリにコミットし、リモートリポジトリへプッシュします。

手順の詳細は、「6.3 プロジェクト管理の開始準備（ベースとなるプロジェクトの作成、保存）」を参照してください。

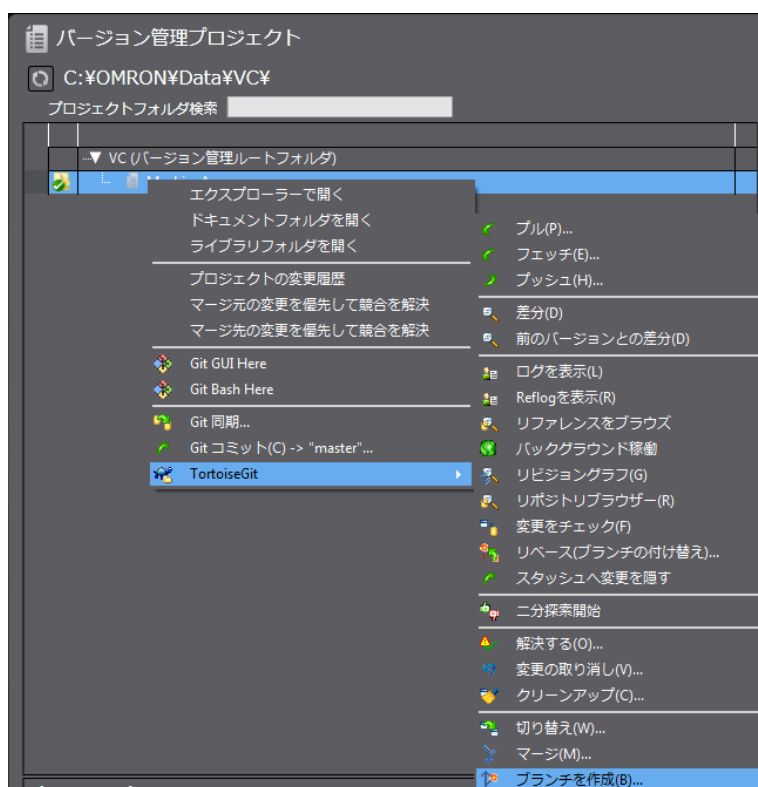
7.4.3.2 ローカルリポジトリに装置 A 用のブランチを作成して、リモートリポジトリへプッシュする (No.2 : 装置 A 開発)

1. リモートリポジトリとローカルリポジトリの同期

リモートリポジトリにプロジェクトを登録したあとは、他の開発者がリモートリポジトリの対象プロジェクトを更新している可能性がありますので、マスタとなるリモートリポジトリと開発者 A のパソコンのローカルリポジトリの同期をとるための操作を行います。手順の詳細は「7.3.3.2 リモートリポジトリからプロジェクトを読み出す (No.2 : 開発者 A)」を参照してください。

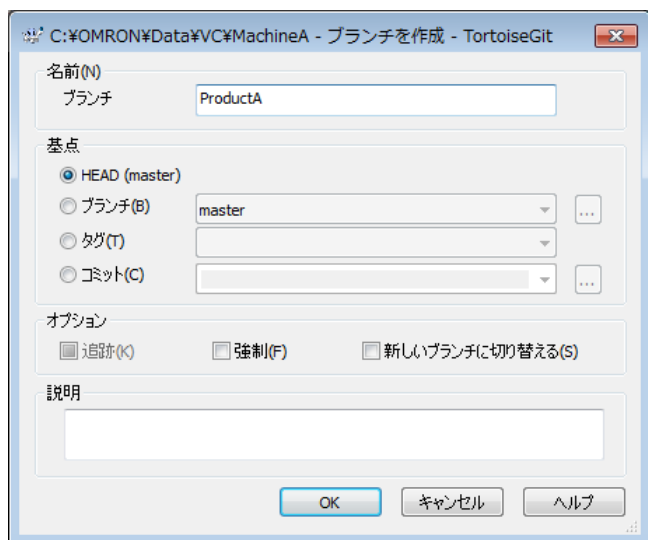
なお、本説明のシナリオでは開発者 A 以外はリモートリポジトリにアクセスしていないため、「7.4.3.1 リモートリポジトリにプロジェクトを登録する (No.1 : 装置 A 開発)」の手順の直後は、この操作は必要ありません。

2. ブランチを作成するために、マルチビューエクスプローラのフォルダアイコン、または [バージョン管理プロジェクト] 画面でプロジェクトを右クリックして、[TortoiseGit] - [ブランチを作成] を選択します。



「ブランチを作成」ダイアログボックスが表示されます。

3. 適切なブランチ名（ここでは「ProductA」）を入力し、[OK] ボタンをクリックします。 [基点] はここでは最新のリビジョンでかまいません。HEAD (master) を選択します。



ブランチの作成が完了します。

4. 作成されたブランチ「ProductA」を、ローカルリポジトリからリモートリポジトリへプッシュします。リモートリポジトリへプッシュする手順の詳細は「6. 4. 3 変更内容のリモートリポジトリへの登録（プッシュ）」を参照してください。

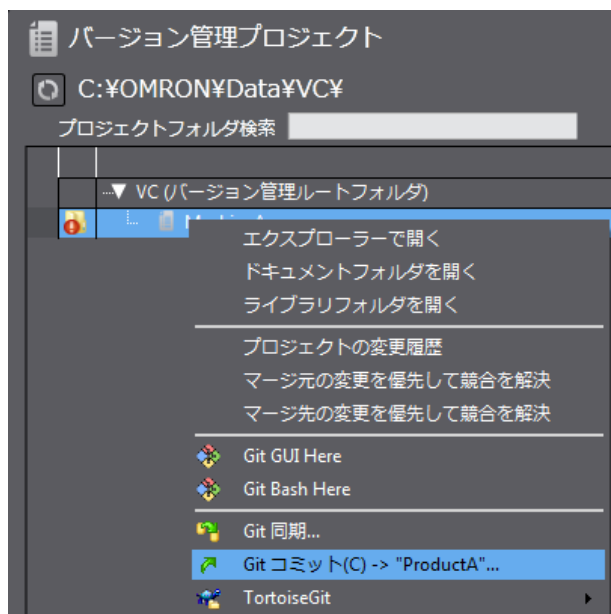
7. 4. 3. 3 装置 A のプログラムを作成してコミットし、リモートリポジトリへプッシュする (No. 3 : 装置 A 開発)

開発者のパソコンで、作業をするブランチを切り替えて、グローバル変数およびプログラムを入力します。作成後に、これを完成品と見なして「バージョン 1.0」としてタグをつけます。つづけて、リモートリポジトリへプッシュします。

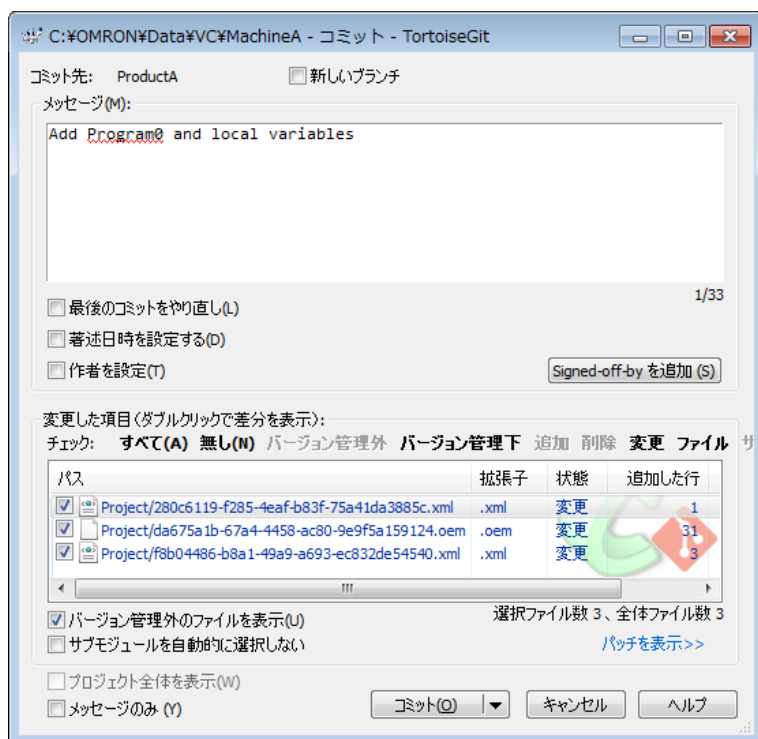
1. 編集対象のブランチを装置 A 用のブランチ「ProductA」に切り替えます。編集対象のブランチを切り替える手順の詳細は「7. 3. 3. 7 ProgramA およびグローバル変数の変更を作業用のブランチにコミットする (No. 8、9 : 開発者 A)」を参照してください。
2. 製品 A 用のプログラムを入力します。以下は入力例です。



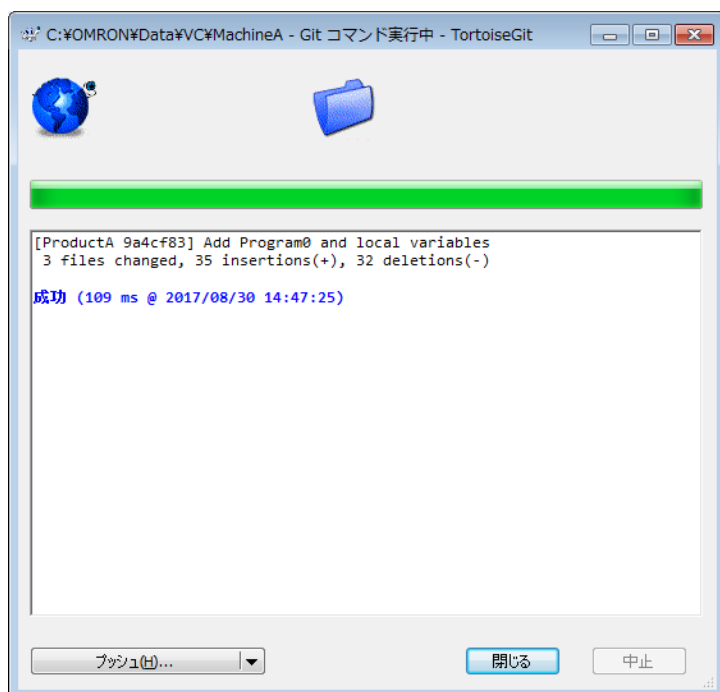
3. メインメニューから [ファイル] - [保存] を選択してプロジェクトを保存します。
4. [ファイル] - [閉じる] を選択してプロジェクトを閉じます。
5. [バージョン管理プロジェクト] 画面で対象プロジェクトを右クリックして、[Git コミット → “ProductA”] を選択します。



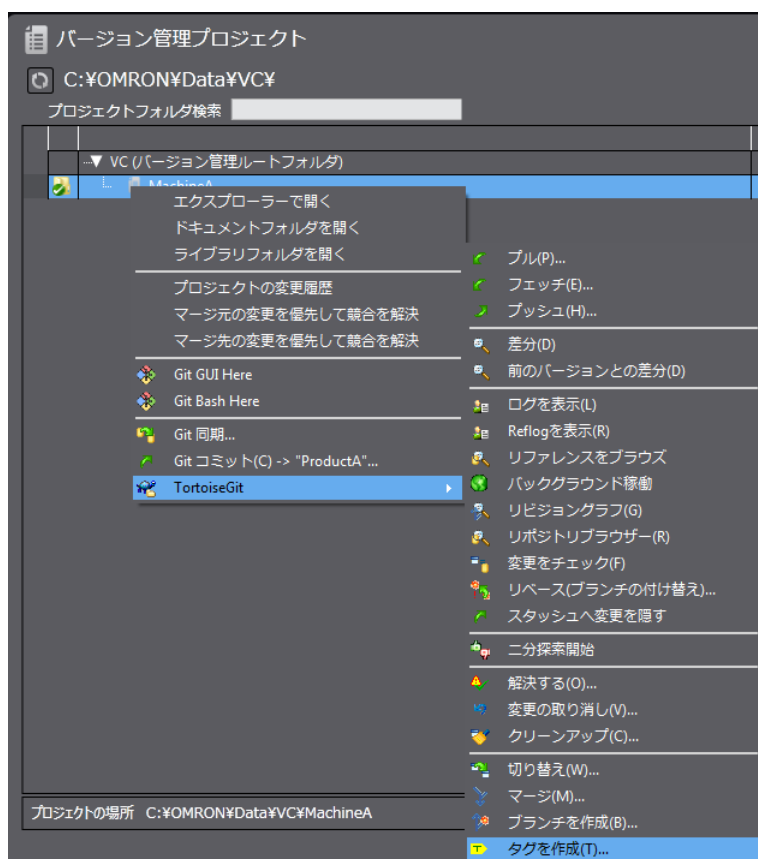
〔コミット〕 ダイアログボックスが表示されます。



6. 〔メッセージ〕 エリアにコメントを入力し、〔コミット〕 ボタンをクリックします。
コミットが完了します。

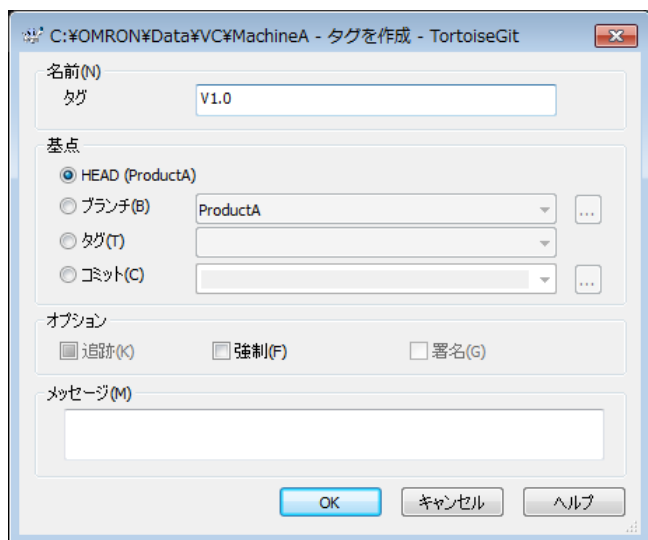


7. [閉じる] ボタンをクリックします。
8. [バージョン管理プロジェクト] 画面で対象プロジェクトを右クリックして、[TortoiseGit] - [タグを作成] を選択します。



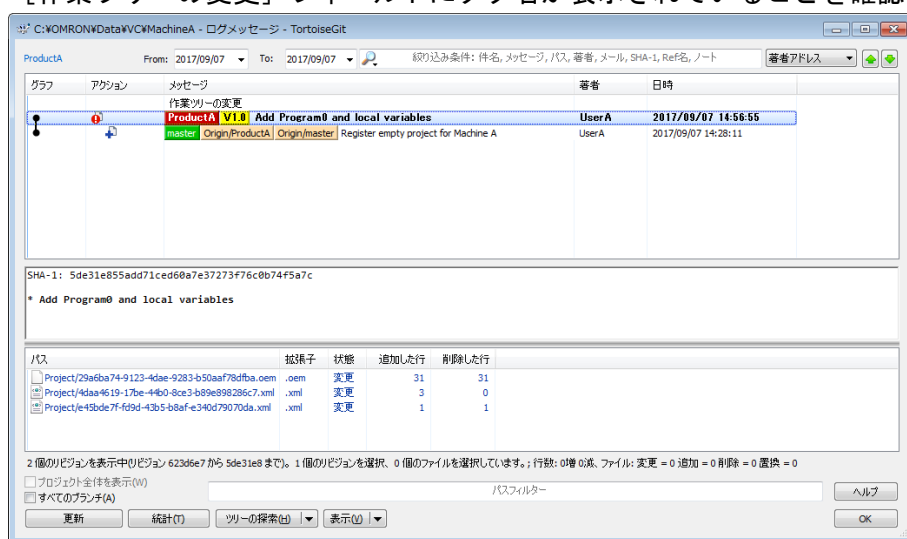
[タグを作成] ダイアログボックスが表示されます。

9. [タグ] テキストボックスにバージョン番号など区別しやすい名前を入力し、[OK] ボタンをクリックします。
ここでは「V1.0」と入力します。



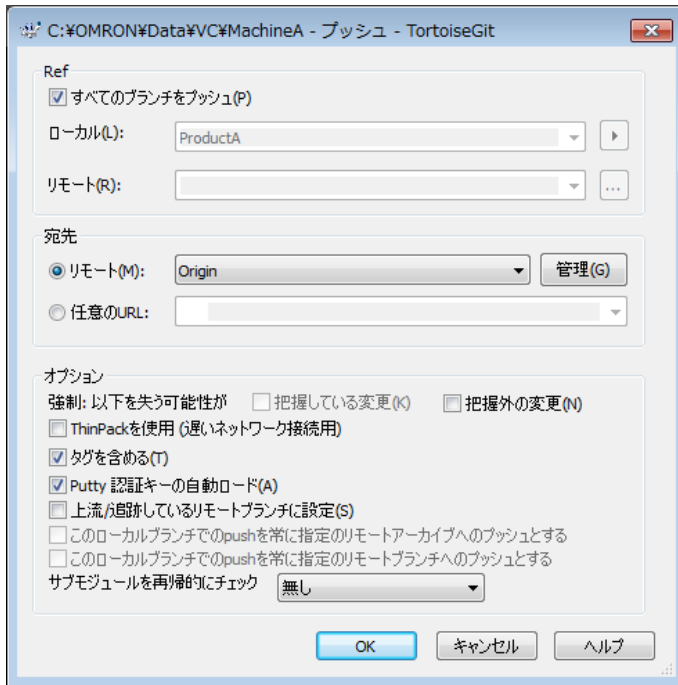
10. マルチビューエクスプローラのフォルダアイコン、または [バージョン管理プロジェクト] 画面でプロジェクトを右クリックして、[TortoiseGit] - [ログを表示] を選択します。
ログメッセージ ダイアログボックスが表示されます。

11. [作業ツリーの変更] フィールドにタグ名が表示されていることを確認できます。



12. ローカルリポジトリからリモートリポジトリへ装置 A 用ブランチの変更 (プログラムの入力) をプッシュします。手順の詳細は「6.4.3 変更内容のリモートリポジトリへの登録 (プッシュ)」を参照してください。
なお、[プッシュ] ダイアログボックスでは、[オプション] の [タグを含める] にチェックを入れてから

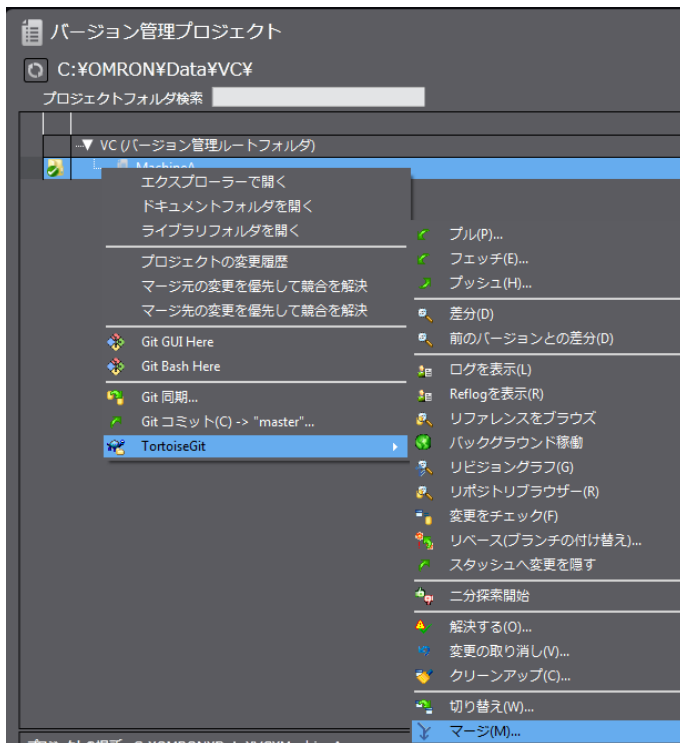
[OK] ボタンをクリックしてください。



7.4.3.4 装置Aの変更をローカルリポジトリのmasterへマージし、リモートリポジトリへプッシュする (No. 4 : 装置A開発)

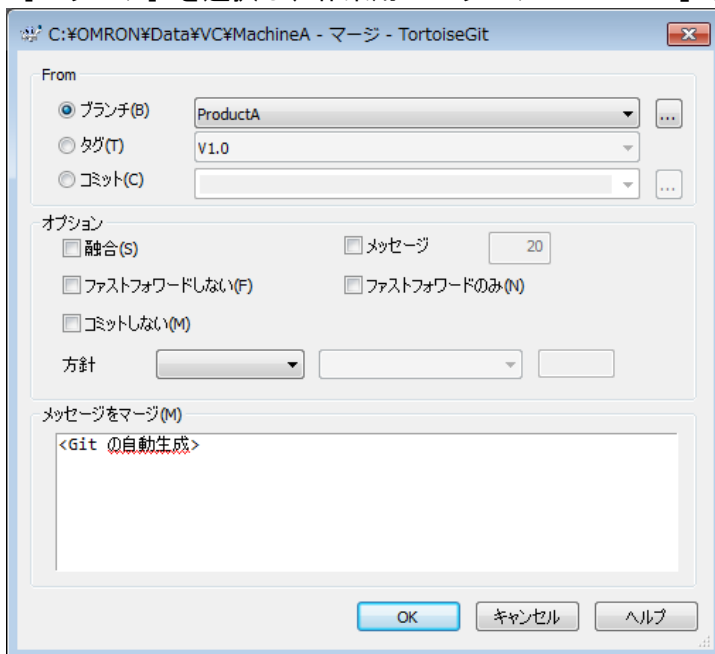
作業をするブランチをmasterブランチに切り替えてから、装置Aのブランチ「ProductA」での変更をmasterブランチにマージします。

1. 編集対象のブランチをmasterブランチに切り替えます。ブランチを切り替える手順の詳細は「7.3.3.7 ProgramA およびグローバル変数の変更を作業用のブランチにコミットする (No. 8、9 : 開発者A)」を参照してください。
2. masterブランチにマージするため、[バージョン管理プロジェクト]画面で対象のプロジェクトを右クリックして、[TortoiseGit] - [マージ]を選択します。



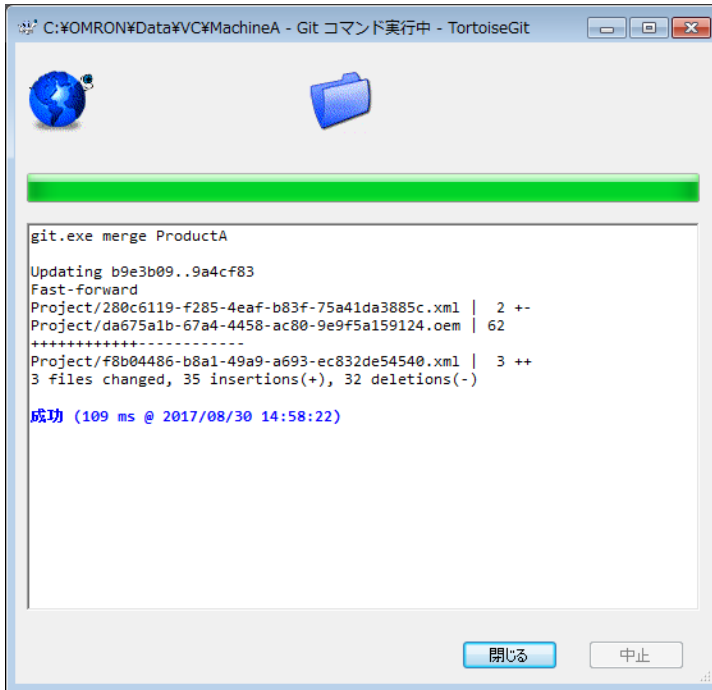
[マージ] ダイアログボックスが表示されます。

3. [ブランチ] を選択し、作業用のブランチ「ProductA」を選択して、[OK] ボタンをクリックします。



このシナリオでは ProductA ブランチの内容とタグ V1.0 の内容は同じなので、タグラジオボタンを選択し、V1.0 を選択してもかまいません。

マージが完了します。

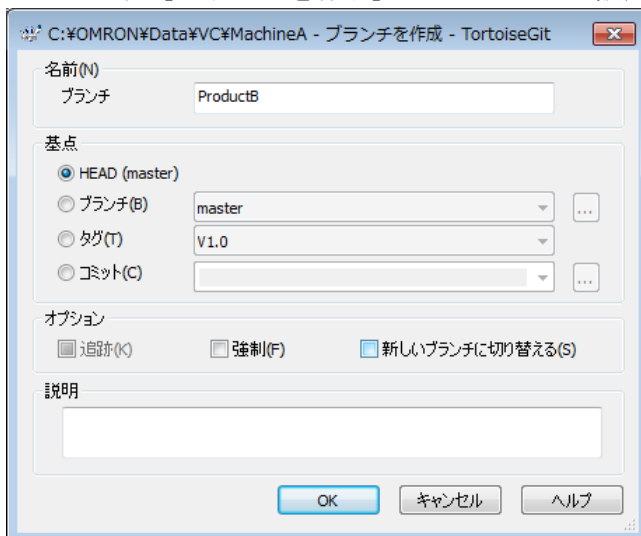


4. 「閉じる」 ボタンをクリックします。
5. ローカルリポジトリからリモートリポジトリへ装置 A 用ブランチの変更(プログラムの入力)をプッシュします。リモートリポジトリへプッシュする手順の詳細は「6. 4. 3 変更内容のリモートリポジトリへの登録 (プッシュ)」を参照してください。

7. 4. 3. 5 ローカルリポジトリに装置 B 用のブランチを作成して、リモートリポジトリへプッシュする (No. 5 : 装置 B 開発)

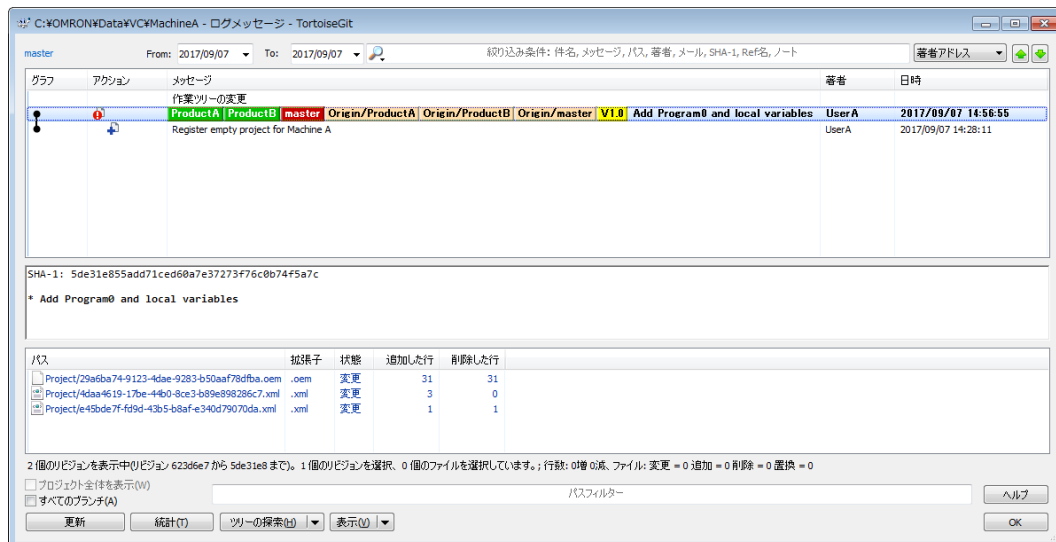
ここでは「ProductB」というブランチを master から作成し、その後リモートリポジトリへ変更内容をプッシュします。一連の手順は「7. 4. 3. 2 ローカルリポジトリに装置 A 用のブランチを作成して、リモートリポジトリへプッシュする (No. 2 : 装置 A 開発)」を参照してください。

このとき、「ブランチを作成」ダイアログでの設定は以下のようにします。



ローカルリポジトリに ProductB ブランチを作成した後、この変更をリモートリポジトリにプッシュします。

プッシュ後のログメッセージダイアログボックスは以下ようになります。

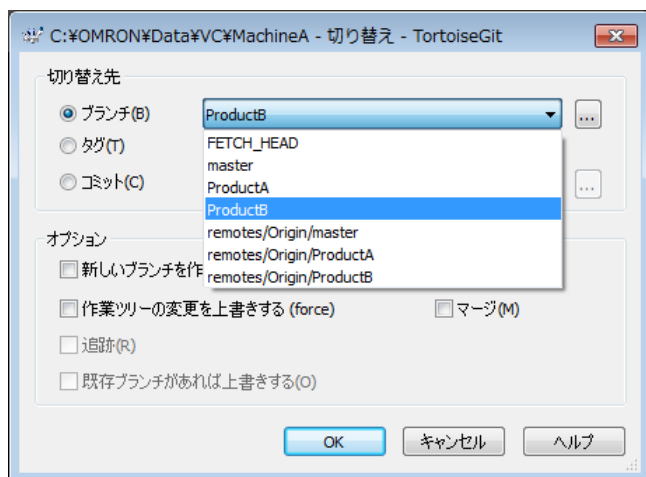


リモートリポジトリ、ローカルリポジトリともに master、ProductA、ProductB ブランチが同期していることを確認できます。

7.4.3.6 装置 B の構成・設定を変更してコミットする (No.6 : 装置 B 開発)

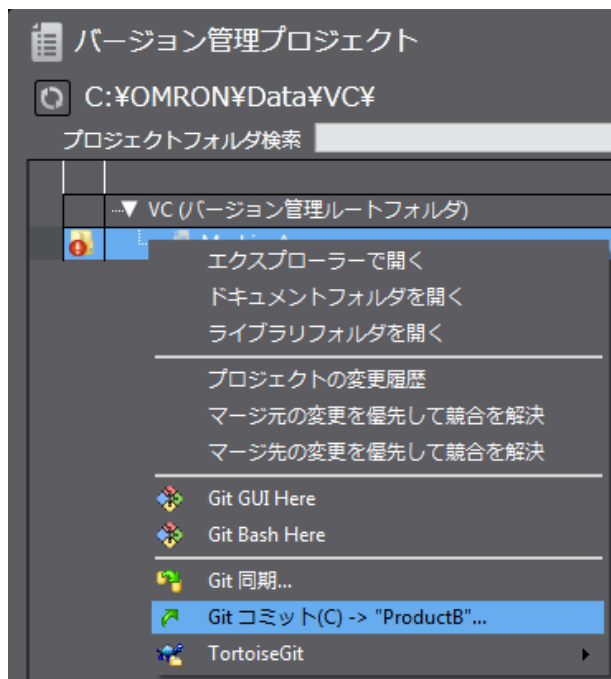
装置 B のプロジェクトを作成し、コミットします。このシナリオでは、装置 B のプロジェクトは装置 A のプロジェクトを元にしており、プログラムに変更はなく構成・設定だけが異なるものとします。

1. 編集対象のブランチを装置 B 用のブランチに切り替えます。ブランチを切り替える手順の詳細は「7.3.3.7 ProgramA およびグローバル変数の変更を作業用のブランチにコミットする (No.8、9 : 開発者 A)」を参照してください。
ここでは、[切り替え] ダイアログボックスの [ブランチ] で [ProductB] を選択します。

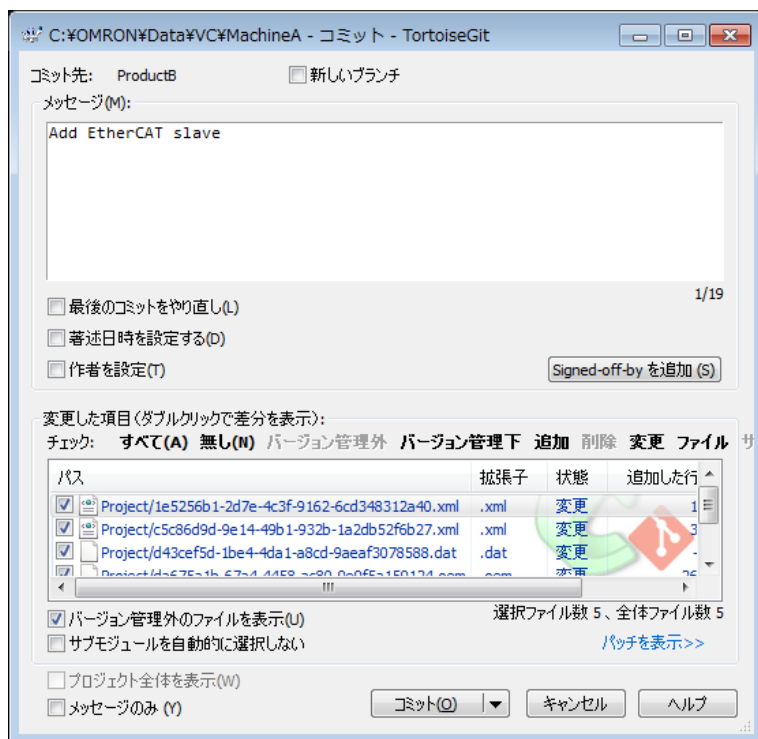


2. 装置 B のための構成・設定の変更を行います。たとえば、EtherCAT スレーブを追加します。
3. メインメニューから [ファイル] - [保存] を選択してプロジェクトを保存します。
4. [ファイル] - [閉じる] を選択してプロジェクトを閉じます。

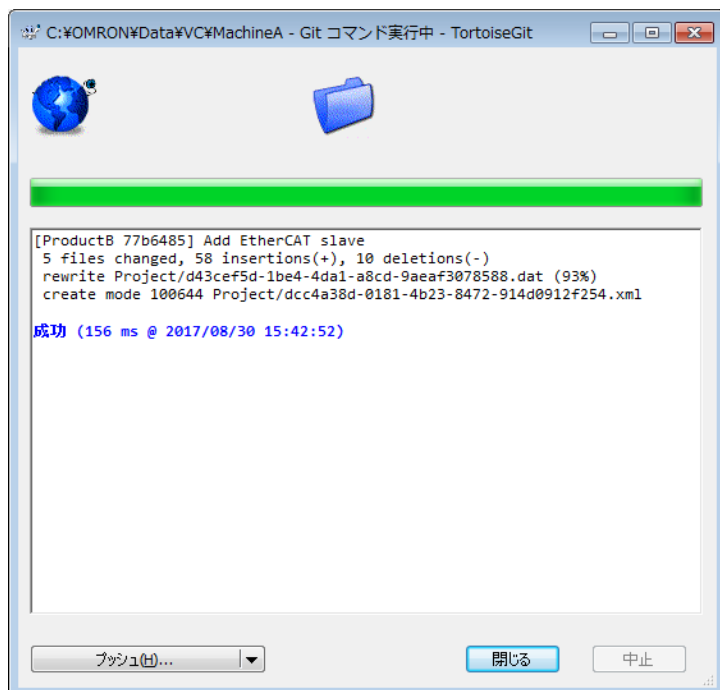
5. [バージョン管理プロジェクト] 画面で対象プロジェクトを右クリックして、[Git コミット -> "ProductB"] を選択します。



[コミット] ダイアログボックスが表示されます。



6. [メッセージ] エリアにコメントを入力し、[コミット] ボタンをクリックします。
コミットが完了します。

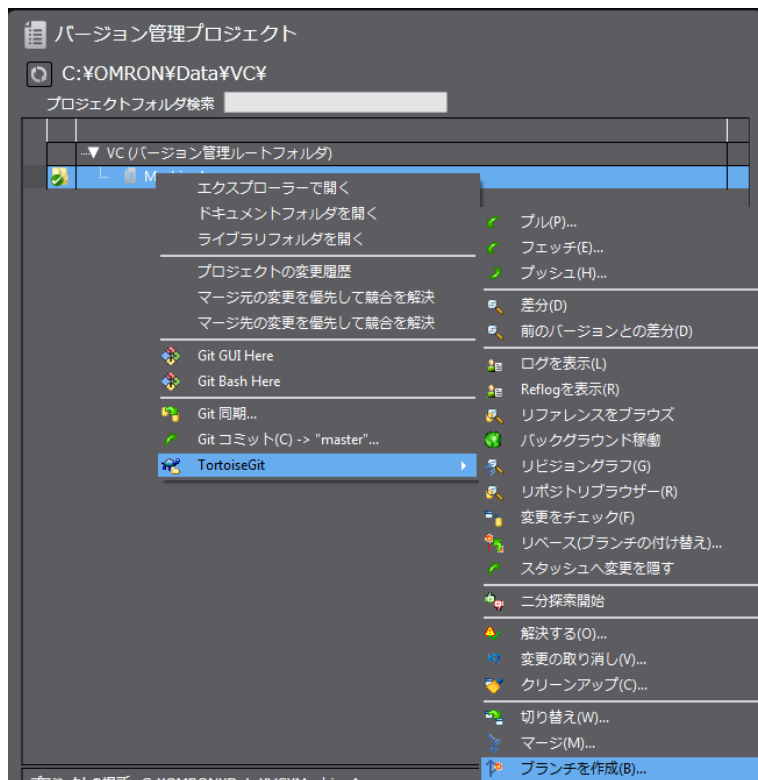


7. 「閉じる」 ボタンをクリックします。

7.4.3.7 ローカルリポジトリの master から、不具合修正のためのブランチを作成する (No. 7 : 不具合修正)

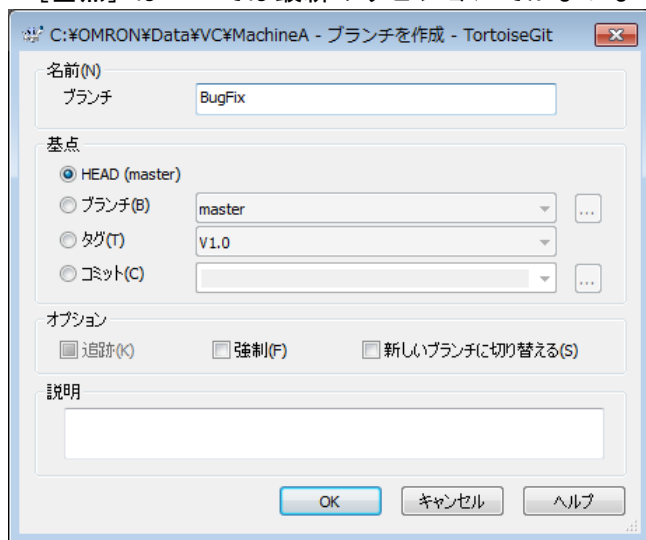
ここでは、装置 A で発見された不具合の修正内容を装置 B にも反映させるために、master から分岐させた BugFix ブランチで不具合修正を行います。まず BugFix ブランチを作成します。

1. 編集対象のブランチを master ブランチに切り替えます。ブランチを切り替える手順の詳細は「7.3.3.7 ProgramA およびグローバル変数の変更を作業用のブランチにコミットする (No. 8、9 : 開発者 A)」を参照してください。
2. ブランチを作成するために、マルチビューエクスプローラのフォルダアイコン、または [バージョン管理プロジェクト] 画面でプロジェクトを右クリックして、[TortoiseGit] - [ブランチを作成] を選択します。



「ブランチを作成」ダイアログボックスが表示されます。

3. 適切なブランチ名（ここでは「BugFix」）を入力し、[OK] ボタンをクリックします。
[基点] はここでは最新のリビジョンでかまいません。HEAD（master）を選択します。



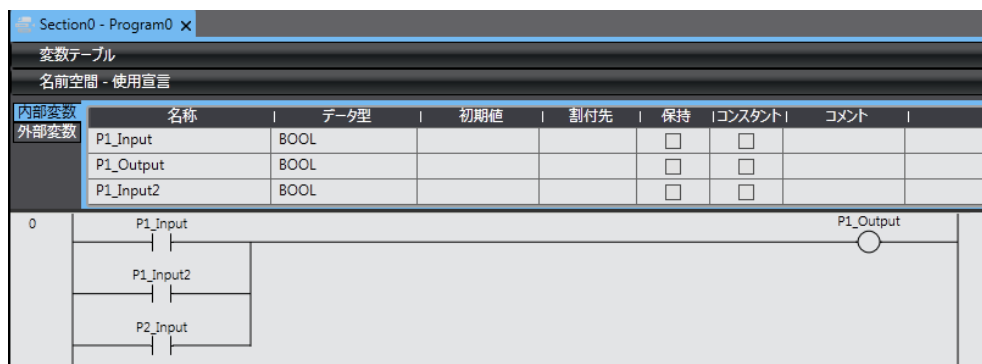
ブランチの作成が完了します。

7.4.3.8 不具合修正をコミットして master へマージし、リモートリポジトリへプッシュする（No.8：不具合修正）

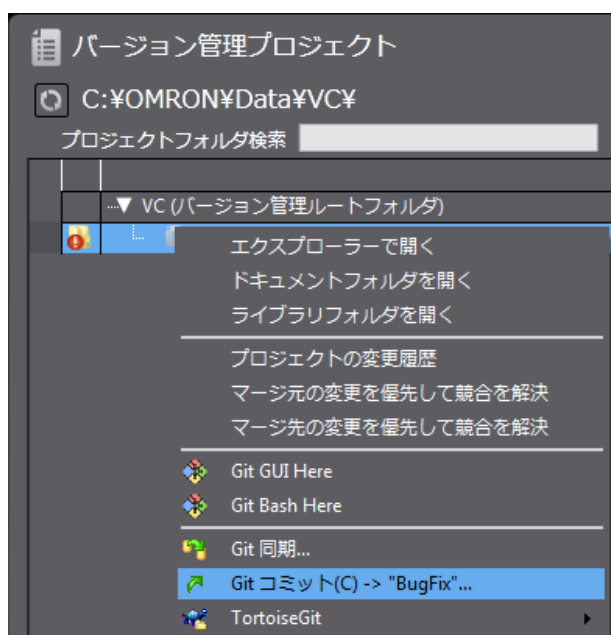
BugFix ブランチへ切り替えてプログラムを修正します。その後、修正内容を master へマージし、リモートリポジトリへプッシュします。

1. 編集対象のブランチを不具合修正用のブランチ「BugFix」に切り替えます。ブランチを切り替える手順の詳細は「7.3.3.7 ProgramA およびグローバル変数の変更を作業用のブランチにコミットする（No.8、9：開発者A）」を参照してください。

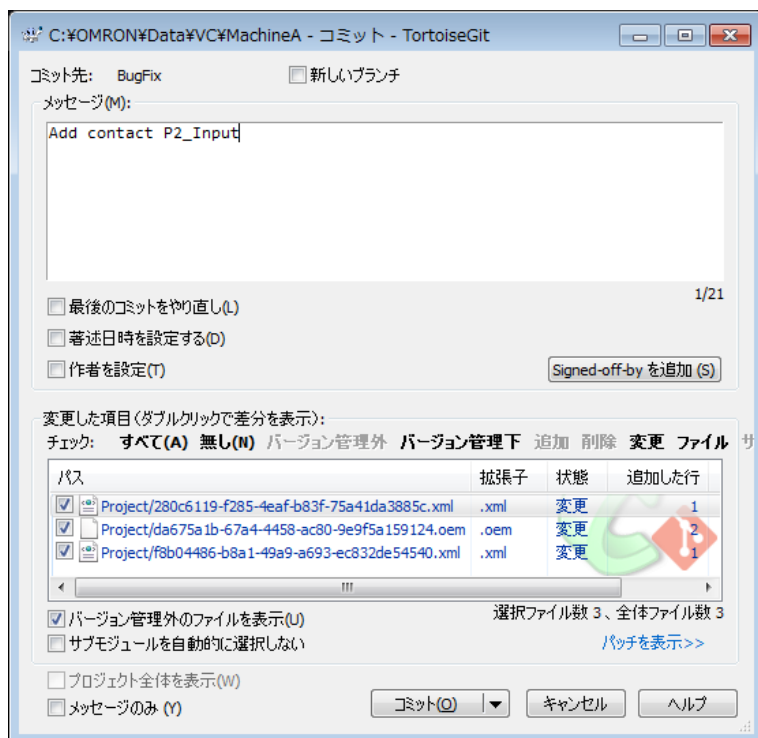
2. 不具合修正のためのプログラムの変更を行います。以下は入力例です。



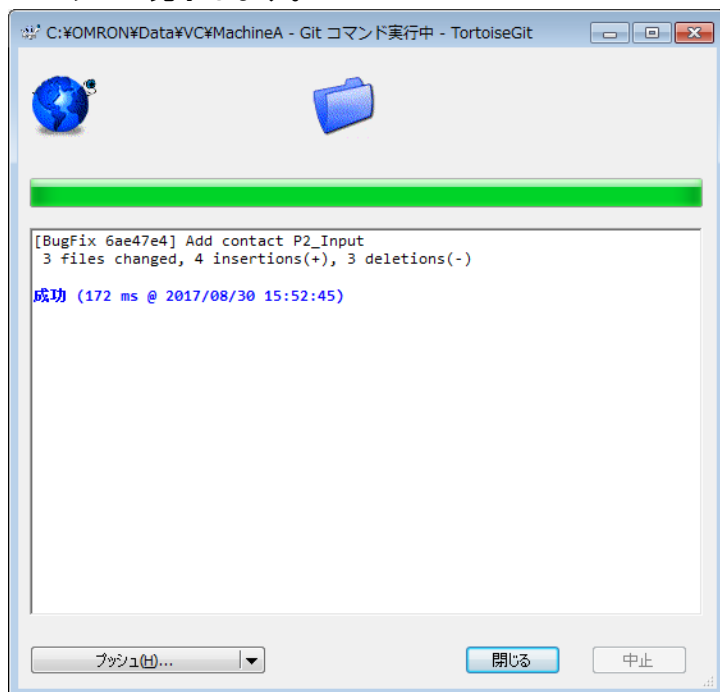
3. メインメニューから [ファイル] - [保存] を選択してプロジェクトを保存します。
4. [ファイル] - [閉じる] を選択してプロジェクトを閉じます。
5. [バージョン管理プロジェクト] 画面で対象プロジェクトを右クリックして、[Git コミット-> "BugFix"] を選択します。



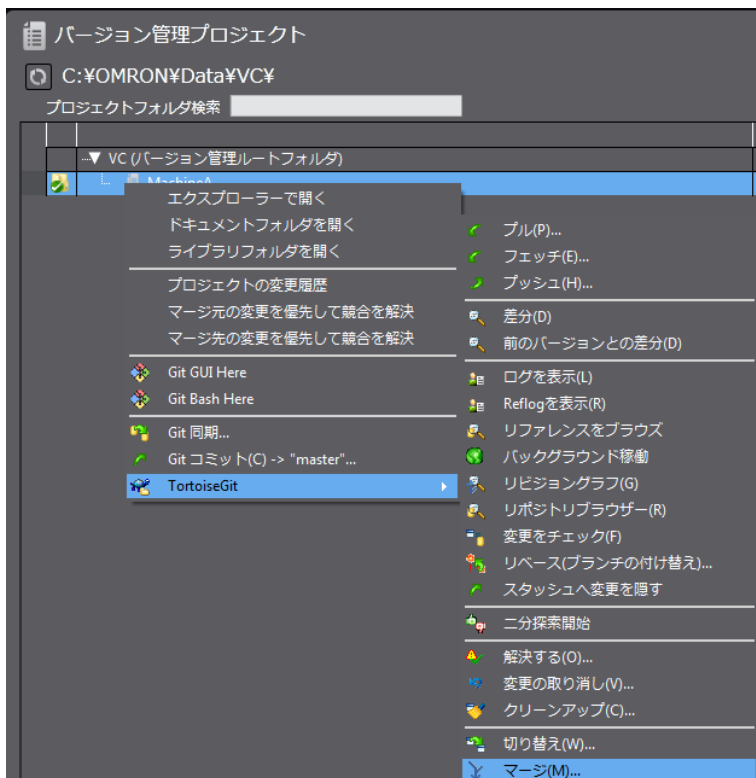
[コミット] ダイアログボックスが表示されます。



6. [メッセージ] エリアにコメントを入力し、[コミット] ボタンをクリックします。
コミットが完了します。

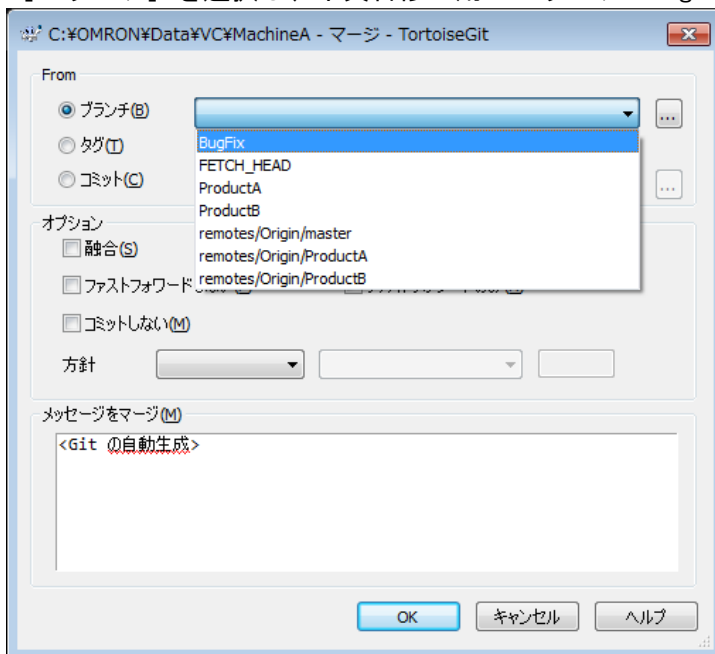


7. [閉じる] ボタンをクリックします。
8. 編集対象のブランチを master ブランチに切り替えます。ブランチの切り替え手順の詳細は「7.3.3.7 ProgramA およびグローバル変数の変更を作業用のブランチにコミットする (No. 8、9 : 開発者 A)」を参照してください。
9. BugFix ブランチの変更を master ブランチにマージするため、[バージョン管理プロジェクト] 画面で対象のプロジェクトを右クリックして、[TortoiseGit] - [マージ] を選択します。

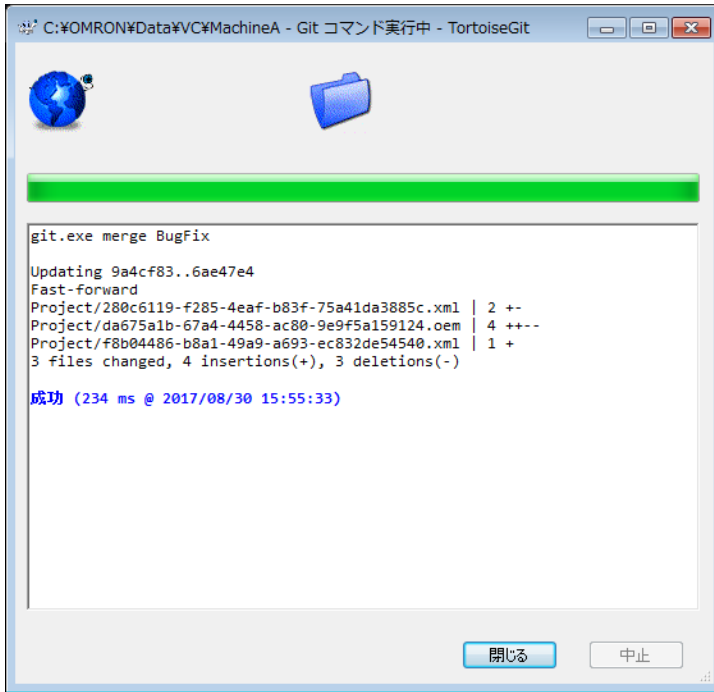


「マージ」ダイアログボックスが表示されます。

10. 「ブランチ」を選択し、不具合修正用のブランチ「BugFix」を選択して、「OK」ボタンをクリックします。



11. マージが完了します。



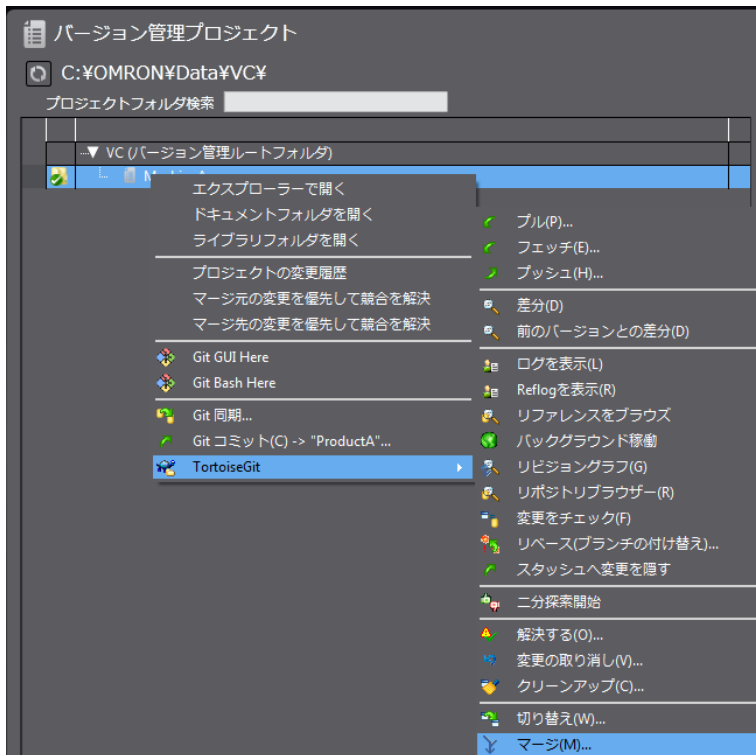
12. [閉じる] ボタンをクリックします。

13. ローカルリポジトリからリモートリポジトリへ不具合修正ブランチおよび master ブランチの変更をプッシュします。リモートリポジトリへプッシュする手順の詳細は「6. 4. 3 変更内容のリモートリポジトリへの登録（プッシュ）」を参照してください。

7. 4. 3. 9 不具合修正の内容を、装置 A 用のブランチにマージして、リモートリポジトリへプッシュする（No. 9：装置 A 開発）

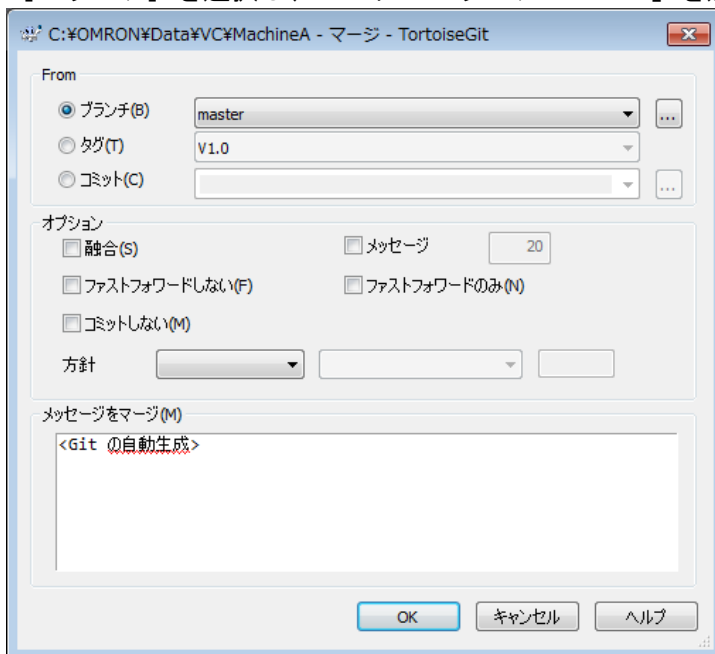
master にマージされた装置 A の不具合修正内容を、装置 A 用のブランチにマージします。マージ後に、これを修正後の完成品と見なして「バージョン 1.1」としてタグをつけます。つづけて、リモートリポジトリへプッシュします。

1. 編集対象のブランチを装置 A 用のブランチ「ProductA」に切り替えます。ブランチを切り替える手順の詳細は「7. 3. 3. 7 ProgramA およびグローバル変数の変更を作業用のブランチにコミットする（No. 8、9：開発者 A）」を参照してください。
2. ProductA ブランチにマージするため、[バージョン管理プロジェクト] 画面で対象のプロジェクトを右クリックして、[TortoiseGit] - [マージ] を選択します。

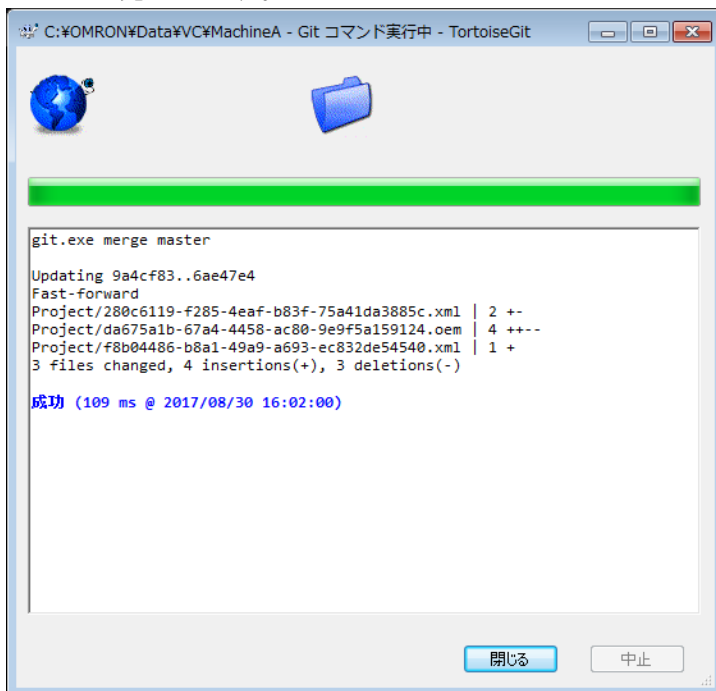


「マージ」ダイアログボックスが表示されます。

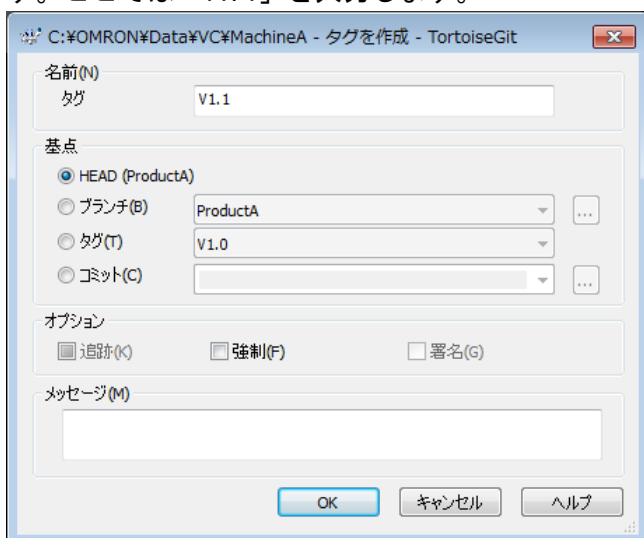
3. 「ブランチ」を選択し、マスタのブランチ「master」を選択して、「OK」ボタンをクリックします。



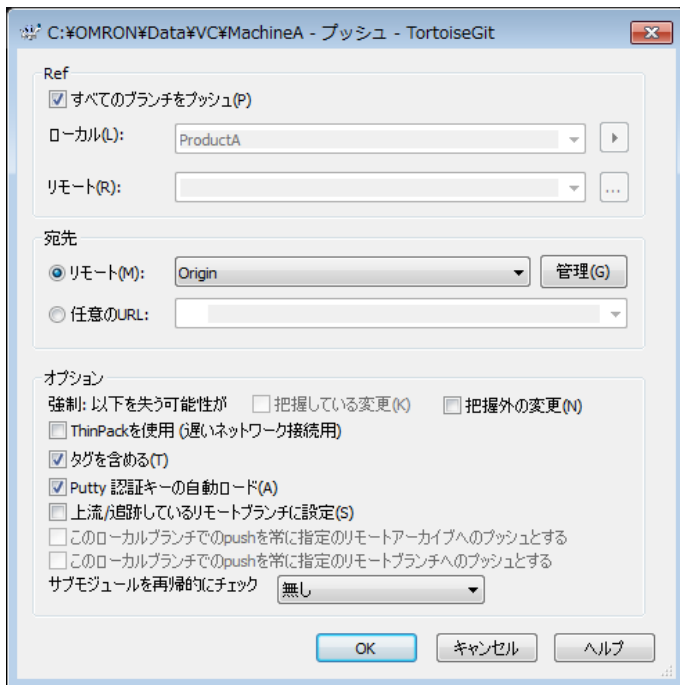
マージが完了します。



4. 「閉じる」 ボタンをクリックします。
5. 「バージョン管理プロジェクト」 画面で対象プロジェクトを右クリックして、[TortoiseGit] - 「タグを作成」を選択します。
「タグを作成」ダイアログボックスが表示されます。
6. 「タグ」 のテキストボックスにバージョン番号など区別しやすい名前を入力し、[OK]ボタンをクリックします。ここでは「V1.1」と入力します。



7. ローカルリポジトリからリモートリポジトリへ装置 A 用ブランチの変更(不具合修正の反映)をプッシュします。手順の詳細は「6.4.3 変更内容のリモートリポジトリへの登録(プッシュ)」を参照してください。
なお、「プッシュ」ダイアログボックスでは、「オプション」の「タグを含める」にチェックを入れてから [OK] ボタンをクリックしてください。

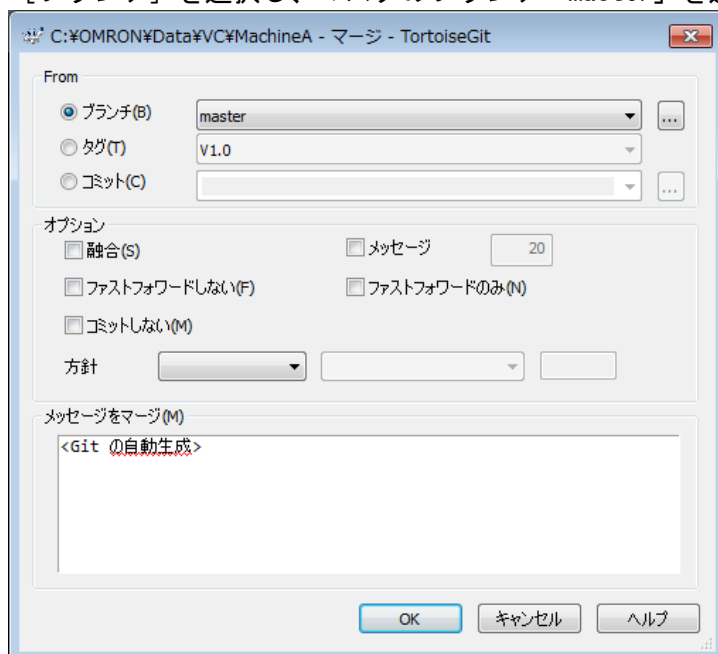


7.4.3.10 不具合修正の内容を、装置 B 用のブランチにマージして、リモートリポジトリへプッシュする (No. 10 : 装置 B 開発)

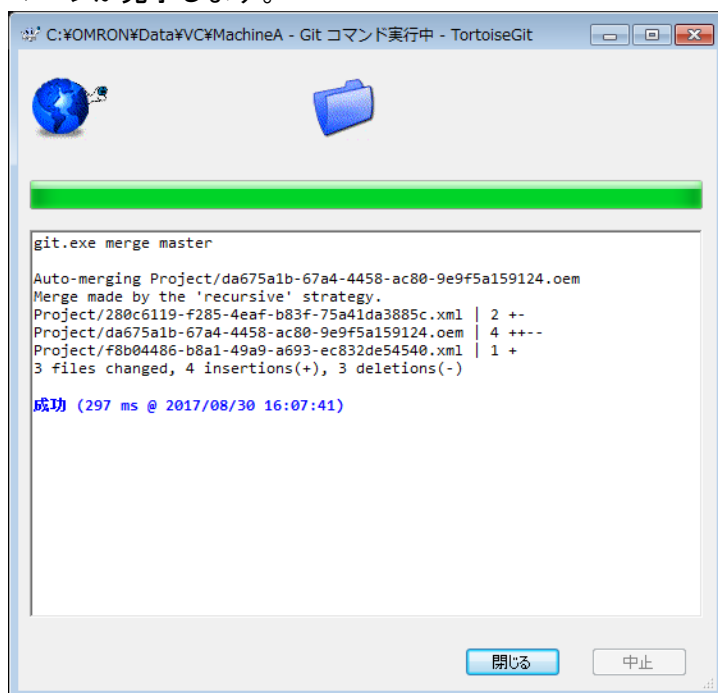
master にマージされた装置 A の不具合修正内容を、装置 B 用のブランチにマージします。マージ後に、これを装置 B のプログラムの完成品と見なして「バージョン 1.0」としてタグをつけます。つづけて、リモートリポジトリへプッシュします。

1. 編集対象のブランチを装置 B 用のブランチ「ProductB」に切り替えます。ブランチを切り替える手順の詳細は「7.3.3.7 ProgramA およびグローバル変数の変更を作業用のブランチにコミットする (No. 8、9 : 開発者 A)」を参照してください。
2. ProductB ブランチにマージするため、[バージョン管理プロジェクト] 画面で対象のプロジェクトを右クリックして、[TortoiseGit] - [マージ] を選択します。
[マージ] ダイアログボックスが表示されます。

3. 「ブランチ」を選択し、マスタのブランチ「master」を選択して、「OK」ボタンをクリックします。



マージが完了します。

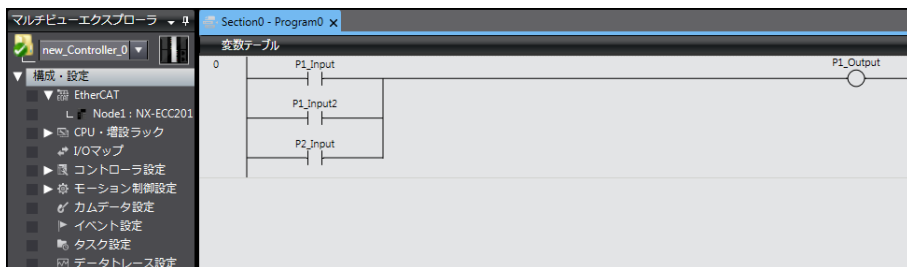


4. 「閉じる」ボタンをクリックします。

<使用上の注意>

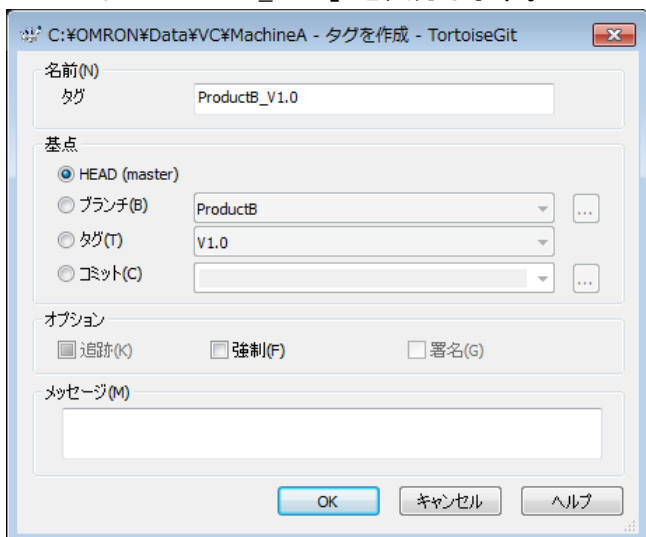
本説明の手順ではマージ時に競合は発生しませんが、装置 A と装置 B とで同じデータを変更していた場合は、競合が発生しマージに失敗します。このような場合は、装置 B の構成・設定の変更を保持するために、装置 B の ProductB ブランチを優先して競合を解決してください。競合の解決の手順については、「7.3.3.12 メインブランチに開発者 B の作業用のブランチの変更をマージする (No. 12 : 開発者 B)」を参照してください。

5. プロジェクトを開くと、以下のように装置 B で追加された構成・設定の変更を保持したまま、プログラムの修正が適用されていることを確認できます。

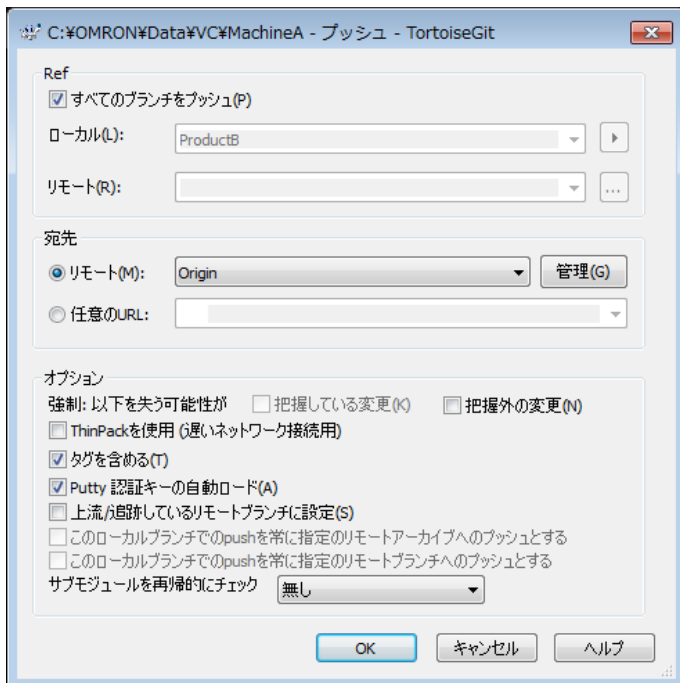


6. [バージョン管理プロジェクト] 画面で対象プロジェクトを右クリックして、[TortoiseGit] – [タグを作成] を選択します。
[タグを作成] ダイアログボックスが表示されます。

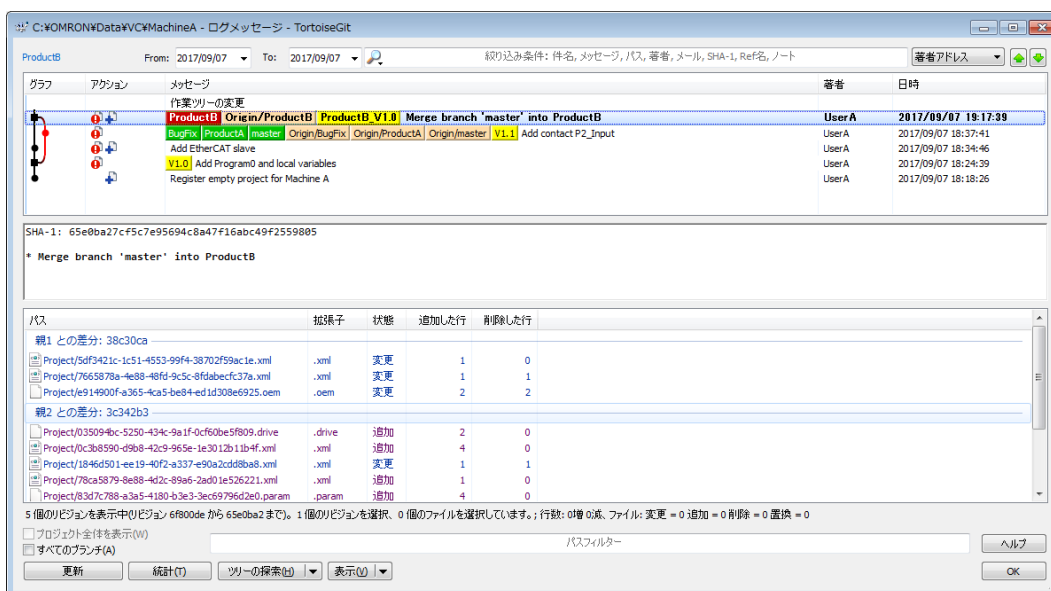
7. [タグ] テキストボックスにバージョン番号など区別しやすい名前を入力し、[OK] ボタンをクリックします。
ここでは「ProductB_V1.0」と入力します。



8. ローカルリポジトリからリモートリポジトリへ装置 B 用ブランチの変更(構成設定の変更および不具合修正の反映)をプッシュします。手順の詳細は「6.4.3 変更内容のリモートリポジトリへの登録(プッシュ)」を参照してください。
なお、[プッシュ] ダイアログボックスでは、[オプション] の [タグを含める] にチェックを入れてから [OK] ボタンをクリックしてください。



9. プッシュ後のログメッセージダイアログボックスは以下のようになります。



メッセージフィールドの内容から、BugFix ブランチの変更が master ブランチおよび ProductA ブランチに反映されており、master ブランチの内容 (BugFix ブランチの変更の反映) が ProductB ブランチに反映されていることを確認できます。

●本誌に記載のない条件や環境での使用、および原子力制御・鉄道・航空・車両・燃焼装置・医療機器・娯楽機械・安全機器、その他人命や財産に大きな影響が予測されるなど、特に安全性が要求される用途に使用される際には、当社の意図した特別な商品用途の場合や特別の合意がある場合を除き、当社は当社商品に対して一切保証をいたしません。
●本製品の内、外国為替及び外国貿易法に定める輸出許可、承認対象貨物(又は技術)に該当するものを輸出(又は非居住者に提供)する場合は同法に基づく輸出許可、承認(又は役務取引許可)が必要です。

オムロン株式会社 インダストリアルオートメーションビジネスカンパニー

●製品に関するお問い合わせ先

お客様相談室

 **0120-919-066** クイック オムロン

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

電話 **055-982-5015** (通話料がかかります)

■営業時間：8:00～21:00 ■営業日：365日

●FAXやWebページでもお問い合わせいただけます。

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

●その他のお問い合わせ

納期・価格・サンプル・仕様書は貴社のお取引先、または貴社担当オムロン販売員にご相談ください。
オムロン制御機器販売店やオムロン販売拠点は、Webページでご案内しています。

オムロン制御機器の最新情報をご覧ください。

www.fa.omron.co.jp

緊急時のご購入にもご利用ください。

オムロン商品のご用命は