

Automation Software Sysmac Studio

Startup Guide for Project Version Control Function

SYSMAC-SE2
SYSMAC-TA4

Startup Guide



P125-E1-03

- NOTE -

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means, mechanical, electronic, photocopying, recording, or otherwise, without the prior written permission of OMRON.

No patent liability is assumed with respect to the use of the information contained herein. Moreover, because OMRON is constantly striving to improve its high-quality products, the information contained in this manual is subject to change without notice. Every precaution has been taken in the preparation of this manual. Nevertheless, OMRON assumes no responsibility for errors or omissions. Neither is any liability assumed for damages resulting from the use of the information contained in this publication.

Trademarks

- Sysmac and SYSMAC are trademarks or registered trademarks of OMRON Corporation in Japan and other countries for OMRON factory automation products.
- Microsoft, Windows, Excel, and Visual Basic are either registered trademarks or trademarks of Microsoft Corporation in the United States and other countries.
- EtherCAT® is registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany.
- ODVA, CIP, CompoNet, DeviceNet, and EtherNet/IP are trademarks of ODVA.
- The SD and SDHC logos are trademarks of SD-3C, LLC.
- NVIDIA, the NVIDIA logo, GeForce, and the GeForce logo are the trademarks or registered trademarks of NVIDIA Corporation in the USA and other countries.
- ATI[™], Radeon[™] is a trademark of Advanced Micro Devices, Inc..
- · Celeron, Intel and Intel Core are trademarks of Intel Corporation in the U.S. and / or other countries.
- 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.

Other company names and product names in this document are the trademarks or registered trademarks of their respective companies.

Copyrights

Microsoft product screen shots reprinted with permission from Microsoft Corporation.

Table of Contents

INTF	RODUCI	[ION ·····	3
	Inten	ded Audience	. 3
	Applie	cable Products	. 3
	Term	s and Conditions Agreement	. 3
	Safet	y Precautions	.4
	Regu	lations and Standards	.4
		Software Licenses and Copyrights	.4
	Revis	ion History	. 4
1	OVE	RVIEW ·····	5
2	TERM	/INOLOGY ·····	6
3	SYST	EM CONFIGURATIONS	7
	3.1	Basic Configuration	.7
	3.2	Configurations to Share the Repository with Multiple Users	.7
	3.3	Basic operation of Version Control System	. 10
4	SCO	PE OF THE VERSION CONTROL ······	11
5	SOF	WARE SETUP AND BASIC SETTINGS	12
	5.1	Installing the Sysmac Studio	. 12
	5.2	Registering Sysmac Studio Team Development Option	. 12
	5.3	Installing Git	. 12
	5.4	Installing "TortoiseGit"	. 18
	5.5	Initial Setting of "TortoiseGit"	. 21
	5.6	Creating the shared folder and remote repository	. 23
	5.7	Additional Setting for "TortoiseGit"	
6	PRO	JECT RECORD CONTROL ON SYSMAC STUDIO	27
	6.1	Overview of Project Record Control Using the Version Control Function	. 27
	6.2	Operation Example of Record Control Function	. 28
	6.3	Preparing for Starting Project Control (Creating a Base Project and Saving	,
		6.3.1 Creating a Base Project	. 29
		6.3.2 Registering the project in the Local Repository	. 30
		6.3.3 Synchronizing the Local Repository and Remote Repository	. 33
	6.4	Updating the Project	. 37

		6.4.1	Changing the Project Data	
		6.4.2	Registering Changes to the Local Repository (Commit)	
		6.4.3	Registering the Changes to the Remote Repository (Push)	
	6.5	Searcl	hing for a Project	41
		6.5.1	Updating the Local Repository	41
		6.5.2	Displaying Project Logs	42
		6.5.3	Searching for the Revision	43
	6.6	Comp	aring the Projects	44
	6.7	Rever	ting the Project	45
7 OF S			INT BY MULTIPLE DEVELOPERS AND DERIVED DEVELOPM	
	7.1 and de		tion of version control function in development by multiple devel levelopment	
	7.2	Brancl	h and Merge	50
	7.3	Sysma	ac Studio Project Development by Multiple Developers	
		7.3.1	Sharing Edit Works to Concurrently Develop a Project by Mult	•
		Devel	lopers	
		Devel 7.3.2	Example of Dividing Tasks	
				53
	7.4	7.3.2 7.3.3	Example of Dividing Tasks	53 55
	7.4	7.3.2 7.3.3	Example of Dividing Tasks Operating Procedure	53 55 98
	7.4	7.3.2 7.3.3 Derive	Example of Dividing Tasks Operating Procedure ed Development of Sysmac Studio Projects	53 55 98 98

INTRODUCTION

Thank you for purchasing a Sysmac Studio Team Development Option.

This manual contains operating procedure of basic functions of the Sysmac Studio Team Development Option and project version control function.

Please read this manual and make sure you understand the operating procedure of the Sysmac Studio Team Development Option before you attempt to use it to build a control system.

Intended Audience

This manual is intended for the following personnel, who must also have knowledge of electrical systems (an electrical engineer or the equivalent).

- Personnel in charge of introducing FA systems.
- Personnel in charge of designing FA systems.
- Personnel in charge of installing and maintaining FA systems.
- · Personnel in charge of managing FA systems and facilities.

For programming, this manual is intended for personnel who understand the programming language specifications in international standard IEC 61131-3 or Japanese standard JIS B 3503.

Applicable Products

This manual covers the following products.

Sysmac Studio Team Development Option

Part of the specifications and restrictions for the CPU Units are given in other manuals.

Refer to Sysmac Studio Version 1 Operation Manual (Cat.No.W504) and Sysmac Studio Project Version Control Function Operation Manual (Cat.No. W589).

Terms and Conditions Agreement

- WARRANTY
 - The warranty period for the Software is one year from the date of purchase, unless otherwise specifically agreed.
 - If the User discovers defect of the Software (substantial non-conformity with the manual), and return it
 to OMRON within the above warranty period, OMRON will replace the Software without charge by
 offering media or download from OMRON's website. And if the User discovers defect of media which
 is attributable to OMRON and return it to OMRON within the above warranty period, OMRON will
 replace defective media without charge. If OMRON is unable to replace defective media or correct the
 Software, the liability of OMRON and the User's remedy shall be limited to the refund of the license
 fee paid to OMRON for the Software.
- LIMITATION OF LIABILITY
 - THE ABOVE WARRANTY SHALL CONSTITUTE THE USER'S SOLE AND EXCLUSIVE REMEDIES AGAINST OMRON AND THERE ARE NO OTHER WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO, WARRANTY OF MERCHANTABILITY OR FITNESS FOR PARTICULAR PURPOSE. IN NO EVENT, OMRON WILL BE LIABLE FOR ANY LOST PROFITS OR OTHER INDIRECT, INCIDENTAL, SPECIAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF USE OF THE SOFTWARE.

- OMRON SHALL HAVE NO LIABILITY FOR DEFECT OF THE SOFTWARE BASED ON MODIFICATION OR ALTERNATION TO THE SOFTWARE BY THE USER OR ANY THIRD PARTY. OMRON SHALL NOT BE RESPONSIBLE AND/OR LIABLE FOR ANY LOSS, DAMAGE, OR EXPENSES DIRECTLY OR INDIRECTLY RESULTING FROM THE INFECTION OF OMRON PRODUCTS, ANY SOFTWARE INSTALLED THEREON OR ANY COMPUTER EQUIPMENT, COMPUTER PROGRAMS, NETWORKS, DATABASES OR OTHER PROPRIETARY MATERIAL CONNECTED THERETO BY DISTRIBUTED DENIAL OF SERVICE ATTACK, COMPUTER VIRUSES, OTHER TECHNOLOGICALLY HARMFUL MATERIAL AND/OR UNAUTHORIZED ACCESS.
- OMRON SHALL HAVE NO LIABILITY FOR SOFTWARE DEVELOPED BY THE USER OR ANY THIRD PARTY BASED ON THE SOFTWARE OR ANY CONSEQUENCE THEREOF.
- APPLICABLE CONDITIONS USER SHALL NOT USE THE SOFTWARE FOR THE PURPOSE THAT IS NOT PROVIDED IN THE ATTACHED USER MANUAL.
- CHANGE IN SPECIFICATION The software specifications and accessories may be changed at any time based on improvements and other reasons.
- ERRORS AND OMISSIONS The information in this manual has been carefully checked and is believed to be accurate; however, no responsibility is assumed for clerical, typographical, or proofreading errors, or omissions.

Safety Precautions

Refer to Sysmac Studio Version 1 Operation Manual (Cat.No.W504) and Sysmac Studio Project Version Control Function Operation Manual (Cat.No. W589).

Regulations and Standards

Software Licenses and Copyrights

This product incorporates certain third party software. The license and copyright information associated with this software is available at <u>http://www.fa.omron.co.jp/nj_info_e/</u>.

Revision History

Revision code appears as a suffix to the catalog number on the front and back covers of the guide.

Revision code	Date	Revised content
01	October 2017	Original production
02	April 2019	Revised Terms and Conditions Agreement
03	January 2020	Revised for the support of Sysmac Studio (64 bit)

1 OVERVIEW

As the scale of production machines increases, the following problems in the development of production machines are becoming more serious.

- The development scale of controller programs has been increased and the development period has become longer.
- The workload of changing controller programs has increased due to the increased variation of production machines
- To offer solutions for these issues, new development environment with the following features is now required.
- Environment to develop controller programs with multiple developers
- Development environment where changes made to the common program can be applied to relevant machines with minimum operations

Sysmac Studio will offer functions to control Sysmac Studio project versions (hereinafter referred to as Version Control Function) as a solution for these issues. The version control function realizes various control capabilities by combining the Sysmac Studio with an open source software version control system commonly used in software development.

This document describes the procedures of installing the open source version control system, initial settings on Sysmac Studio, change record management which is the most basic function of the version control, development by multiple developers, and the steps to carry out derived development.

2 TERMINOLOGY

This section explains the terms used in this document.

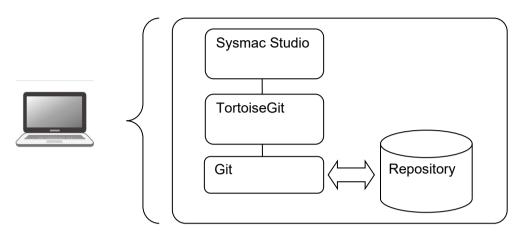
Terms	Description
Sysmac Studio Version Control Function	The function to control the changes and records of the Sysmac Studio projects. It is referred to as "Sysmac Studio version control function" or "version control function".
Git™	An open source software used for version control. This works with the Sysmac Studio to control the versions of the Sysmac Studio projects.
"TortoiseGit"*1	Client software used for the version control system. We operate the version control functions using the "TortoiseGit" menus called from the Sysmac Studio.
Repository	A location to save the data controlled by Git. The version controlled Sysmac Studio project data and its change records are saved.
Remote repository	A repository on the network such as a server, or in a shared folder.
Local repository	A repository created in the computer of each user.
Development by multiple developers	To develop a project program sharing the program among multiple developers. It is important to manage the change contents and operation timing of each developer.
Derived development	To develop a machine program using a program of another machine according to the machine variation. It is important to reflect all changes made for the program of common functions to all the machines and to manage the changes for a specific machine.
Push	Operation to apply the changes in the local repository to the remote repository
Pull	Operation to apply the changes in the remote repository to the local repository
Change set	Information of accumulated changes in a project. With a Sysmac Studio project, this means the information of changes such as adding a global variable or a ladder rung. The version control function of the Sysmac Studio uses "change set" as a chunk of changes to restore the project data to a certain state or to show the list of changes in the chronological order.
Commit	An operation to register a change set, a chunk of information of changes in the project data in the local repository which is controlled by Git. Sysmac Studio has its Save function but it saves data to the working folder. To register changes to the local repository, perform Commit. When you perform Commit, date/time and user name of the operation are automatically registered besides the change set. You can register comments (log messages) in addition, which makes it easy to search for change content.
Revision	When you perform Commit, a change set is registered in the repository with information that makes it easy to search for a change set. At that time, a revision number (a revision identification number) to uniquely identify the change set is automatically applied. When you revert the project to a specific change set, compare a change set with a project under editing, or branch and merge the project, use this revision number to specify the change set. The revision number used in Git is a hexadecimal value of 40 digits generated by the algorithm of SHA-1.

*1: "TortoiseGit" is open source software and can be downloaded from (https://tortoisegit.org/download/).

3 SYSTEM CONFIGURATIONS

3.1 Basic Configuration

The Sysmac Studio version control function operates in an environment that consists of the Sysmac Studio, Git (version control system), "TortoiseGit" (client software for Git), and repositories (folders managed by Git). The following figure illustrates the minimum configuration in which a single user has access to Sysmac Studio projects.



Basic configuration of the Sysmac Studio version control function environment (In a single computer)

3.2 Configurations to Share the Repository with Multiple Users

Sysmac Studio version control function works with Git which has a feature of distributed version control system and they offer a mechanism to share the repository with multiple users.

The configuration consists of local repositories registered in the computers of each user and the remote repository shared by multiple users. At a certain timing of each user, the local repository and remote repository can be synchronized.

To share changes in the local repository with other users, perform a push operation to the remote repository. To apply changes made by other users to the local repository, perform a pull operation from the remote repository.

There are the following three practical configurations depending on the difference in how the remote repository is shared.

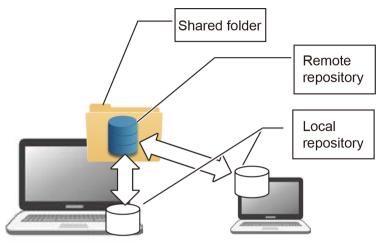
- (1) Using a shared folder on the computer to share it as the remote repository
- (2) Building a dedicated Git server to share it as the remote repository
- (3) Utilizing a Git Server Service on the Internet to Share the Remote Repository

(1) Using a shared folder on the computer to share it as the remote repository

This is the easiest way to build a remote repository.

In this method, you publish a Windows shared folder which works as a remote repository in the local network. The remote repository synchronizes with the local repositories and can be accessed from other computers.

In the following description, we use a remote repository that is built in this way.

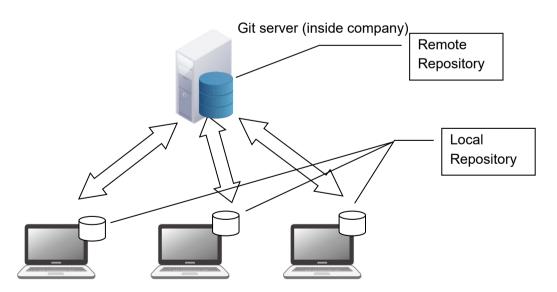


Using a shared folder on the computer to share it as the remote repository

(2) Building a Dedicated Git Server to Share It as the Remote Repository

You can build a dedicated Git server to share the remote repository.

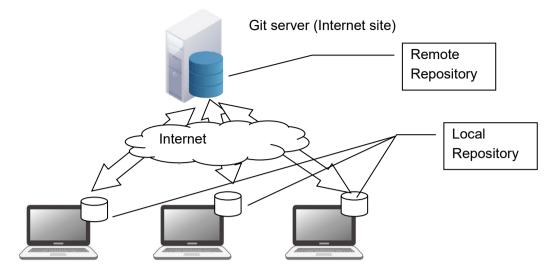
Various Git server software, Gitbucket (for Windows) or GitLab, Gitbucket, Gitblit, Gogs, and so on (for Linux) are available. Although this configuration incurs costs for building and maintaining the server, it has an advantage in reducing the risk of data leakage because the system is closed within the company.



Sysmac Studio version control system on each user's computer

(3) Utilizing a Git Server Service on the Internet to Share the Remote Repository

On the Internet, there are Git server services such as GitHub. Although these are commercial paid services that incur a cost, there are advantages that they require no server maintenance and allow development in parallel with external developers.



Sysmac Studio version control system on each user's computer

3.3 Basic operation of Version Control System

This section describes the basic operation of Version Control System on the Sysmac Studio.

1) Registering changes to the local repository (Commit)

Perform Commit to register a project edited on the Sysmac Studio to the local repository. Refer to *6.4.2 Registering Changes to the Local Repository (Commit)* for details.

2) Registering the changes to the remote repository (Push)

Apply the project registered in the local repository to the remote repository. This operation is called Push. Refer to *6.4.3 Registering the Changes to the Remote Repository (Push)* for details.

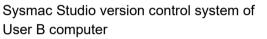
3) Updating the local repository (Pull)

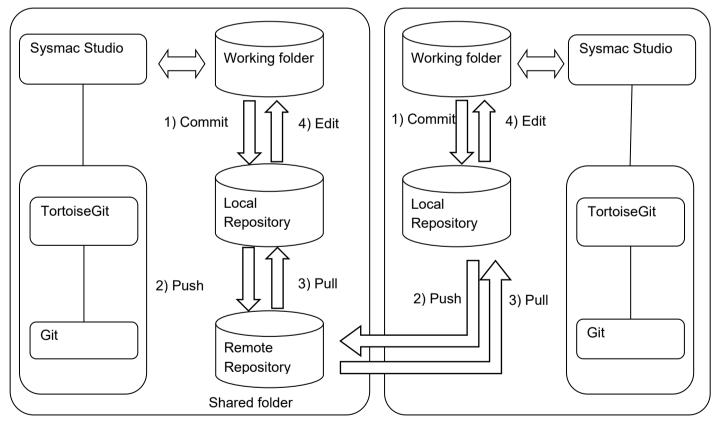
Apply the contents in the remote repository to the local repository. This operation is called Pull. Refer to 6.5.1 *Updating the Local Repository* for details.

4) Editing the project

Select a project in the local repository and open it on the Sysmac Studio to edit. Refer to *6.4.1 Changing the Project Data* for details.

Sysmac Studio version control system on User A computer





Basic operation of Version Control System on Sysmac Studio (Using a shared folder on the computer to share it as the remote repository)

4 SCOPE OF THE VERSION CONTROL

The version control function is applicable to devices that are registered in the project, as well as the following data of each device.

- Data in Configurations and Setup and lower-level folders in the Multiview Explorer
- Depending on the device, however, there is other version-controlled data in addition to the above, or some of the above data is not version-controlled.

Refer to the Sysmac Studio Project Version Control Function Operation Manual (Cat. No. W589) for information on devices with relevant data.

Note that display settings for windows, such as the layout of each pane in the main window, are not version-controlled.

The function is applicable to Controller unit version 1.16 or later. With the Unit of unit version 1.15 or earlier, version control function cannot be used.

5 SOFTWARE SETUP AND BASIC SETTINGS

This section describes the procedure to setup the software and basic settings in a configuration where the remote repository is shared in the shared folder in a computer.

5.1 Installing the Sysmac Studio

Please install the Sysmac Studio Ver.1.20 or later.

Refer to the Sysmac Studio Version 1 Operation Manual (Cat. No. W504) for information on installation of the Sysmac Studio.

5.2 Registering Sysmac Studio Team Development Option

To activate the version control function, register the Sysmac Studio Team Development Option license on the Sysmac Studio.

For details on the registration procedure of the Sysmac Studio Team Development Option, refer to Sysmac Studio Project Version Control Function Operation Manual (Cat.No. W589).

5.3 Installing Git

1. Please download Git from the following URL site.

https://git-scm.com/downloads

Depending on the operating system installed on the computer, download the 32-bit or 64-bit edition of the installer.

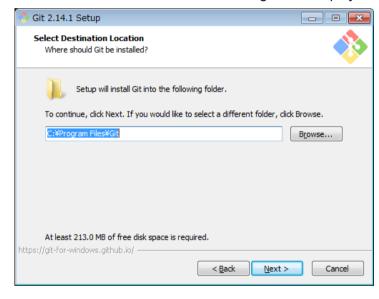
- Start the installer.
 The Windows Security dialog box is displayed according to the Windows version or user authority.
- Click the Execute button in the Windows Security dialog box.
 The User Account Control dialog box is displayed according to the Windows version or user authority.
- 4. Click the **Yes** button in the **User Account Control** dialog box. The **Setup** dialog of Git is displayed.

From here, the procedure to install Git 2.14.1 is described as an example.

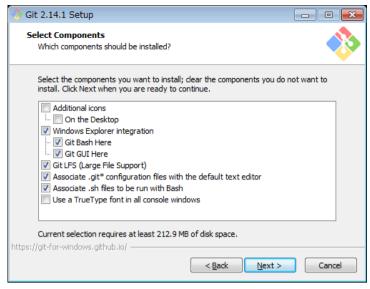


5. Click the **Next** button.

The Select Destination Location dialog box is displayed.



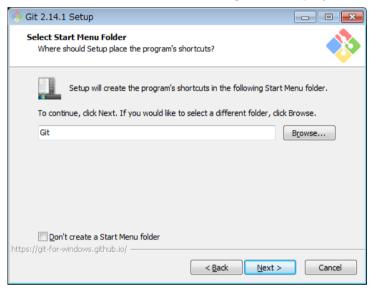
6. Select a folder to install Git, and then click the **Next** button. The **Select Components** dialog box is displayed.



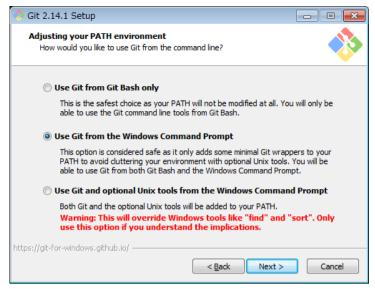
To use the Sysmac Studio version control function, you do not have to change the options selected by default.

7. Click the Next button.

The Select Start Menu Folder dialog box is displayed.



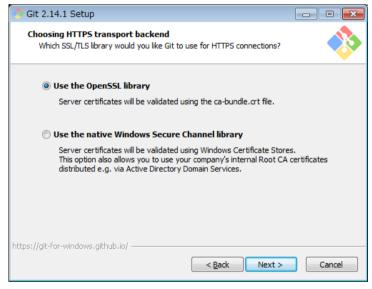
8. Enter the folder name for the Start menu, and click the **Next** button. The below dialog box is displayed to confirm whether to use Git in the command line.



Here, be sure to select the second option Use Git from the Windows Command Prompt.

9. Click the **Next** button.

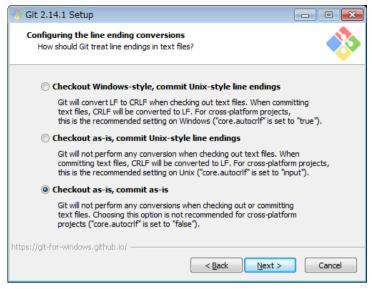
The dialog box to select the library for user authentication is displayed.



If you use a method of sharing the remote repository using the Windows shared folder, you can choose either of them. If you are using Git server, please contact the system administrator of the server. In this example, **Use the OpenSSL library** is selected.

10. Click the Next button.

The dialog box to select line ending of the text files is displayed.



Select the treatment of the line ending in the text files. To use the Sysmac Studio version control function, be sure to select the third option **Checkout as-is, commit as-is**.

11. Click the Next button.

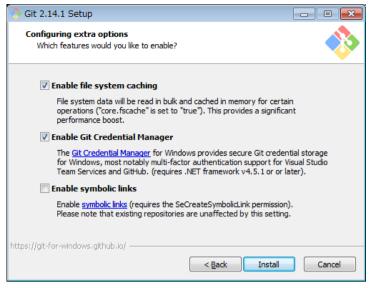
The dialog box to select the terminal software is displayed.



Select the terminal software to use the command line tool "Git Bash". To use the Sysmac Studio version control function, you can choose either of them. In this example, you select the first option **Use MinTTY (the default terminal of MSYS2)**.

12. Click the Next button.

The Configuring extra options dialog box is displayed.



To use the Sysmac Studio version control function, you can choose any option. In this example, first **Enable file system caching** and second **Enable Git Credential Manager** options are selected.

13. Click the Install button.

The following dialog box is displayed when installation is completed.

👌 Git 2.14.1 Setup	
	Completing the Git Setup Wizard
	Setup has finished installing Git on your computer. The application may be launched by selecting the installed shortcuts.
	Click Finish to exit Setup.
	 □ Launch Git Bash ☑ View Release Notes
	Einish

14. Click the Finish button.

5.4 Installing "TortoiseGit"

1. Please download "TortoiseGit" from the following URL site.

https://tortoisegit.org/download/

Depending on the operating system installed on the computer, download the 32-bit or 64-bit edition of the installer. In addition, language packs are also posted on the same URL site. Please download the appropriate one as necessary.

- Start the installer.
 The Windows Security dialog box is displayed according to the Windows version or user authority.
- 3. Click the **Execute** button.

The **Setup** dialog box of "TortoiseGit" is displayed. From here, the procedure to install "TortoiseGit" 2.5.0.0, 64 bit edition is described as an example.



4. Click the **Next** button.

The Information dialog box is displayed.



5. Click the **Next** button.

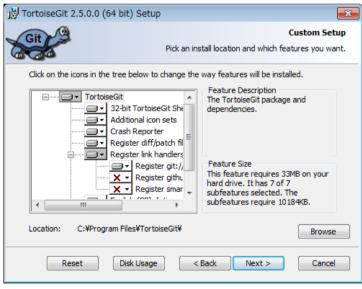
A dialog box to select the client software for user authentication is displayed.

😸 TortoiseGit 2.5.0.0 (64 bit) Setup	—
Cit	Choose SSH Client
GGG	Choose a kind of SSH Client
TortoiseGitPlink, based on PuTTY; optimized for TortoiseGit an better with Windows.	d integrates
◎ OpenSSH, Git default SSH Client	
This setting can be changed in TortoiseGit settings on the "Networ	k" page lateron.
< Back	Next > Cancel

If you use a method of sharing the remote repository using the Windows shared folder, you can choose either one. If you are using Git server, please contact the system administrator of the server. Here, select the first **TortoiseGitPink based on PuTTY; optimized for TortoiseGit and integrates better with Windows** option.

6. Click the **Next** button.

The Custom Setup dialog box is displayed.



To use the Sysmac Studio version control function, do not change the option selected by default.

7. Click the **Next** button.

The **Ready to Install** dialog box is displayed.

🛃 TortoiseGit 2.5.0.0 (64 bit) Setup	-X -
Git	Ready to The Setup Wizard is ready to begin the Complete in	
	llation. If you want to review or change any of your k. Click Cancel to exit the wizard.	
	< Back 🔞 Install 🛛	Cancel

8. Click the Install button.

The User Account Control dialog box is displayed according to the Windows version or user authority.

9. Click the Yes button.

The following dialog is displayed when installation is completed.



- 10. Deselect Run first start wizard option, and then click the Finish button.
- 11. Please download the applicable language pack from the following URL site and set it up as necessary.

https://tortoisegit.org/download/

5.5 Initial Setting of "TortoiseGit"

When installation of the "TortoiseGit" is completed, perform initial settings to use it.

 In the Settings dialog box of "TortoiseGit", select General and then Re-run First Start Wizard. You can open the Settings dialog box by selecting the Start menu, and then All programs - TortoiseGit -Settings.

🕸 First Start Wizard - TortoiseG	it	X
Welcome to TortoiseGit!		
TortoiseGit can be localized	using language packs.	
	ou can find all installed language packs which can be used by	
	ted version of TortoiseGit and your language is missing, ing language pack now and hit "Refresh" before continuing.	
Language:	English Refresh	
Download language packs: https://tortoisegit.org/down	load/	
	< Back Next > Cancel Hel	Þ

2. Click the Next button.

👾 First Start Wizard - TortoiseGit 🧮	3
Welcome to TortoiseGit!	
TortoiseGit is designed as a Shell Extension. Therefore the main interaction with	
TortoiseGit will be using the context menu of the Windows shell such as the Windows Explorer.	
This is the First Start Wizard which will help you configuring the basic settings. TortoiseGit is highly configurable, so it is advisable that you have a look into its settings dialog. The settings dialog is reachable using the start menu or the context menu of the Windows Explorer: TortoiseGit -> Settings.	
Please see the <u>manual</u> for general hints. E.g. in most dialogs there are some shared shortcuts such as F5 for refresh or CTRL+F for search and oftentimes there are powerfull context menus.	
We hope you enjoy the usage of TortoiseGit! In case of questions or problems, please consult our manual or go to https://tortoisegit.org/support/	
< Back Next > Cancel Help]

3. Click the **Next** button.

A dialog box is displayed to configure the path to Git.exe.

🎲 First Start Wizard - T	ortoiseGit	×
Configure git.exe		
	a git.exe for its operations. TortoiseGit tries to automatically detect a t if that doesn't work or you want to use a different one please specify	
Git.exe Path:	C:\Program Files\Git\bin)
Extra PATH:		
	Check now	
Recommended: Git		
https://git-for-wind	ows.github.io/	
	< Back Next > Cancel Help	

Set the path to Git execution module "Git.exe". If you did not change the install path when installing Git, leave it as default.

4. Click the **Next** button.

The Configure user information dialog box is displayed.

👾 First Start Wizard - TortoiseGit	×
Configure user information	
Git requires that you set up a user name and email address. Both are used as meta data	
for your commits (not for authentication).	
Name:	
Fmail:	
These settings will be stored to your global git configuration (%HOME%/.gitconfig) and will be used for all your git repositories as a default.	
Don't store these settings now.	
<back next=""> Cancel Help</back>	

The user name and email address that you enter here will be used as change record information.

5. Enter the user name and email address, and then click the **Next** button. The **Authentication and credential store** dialog box is displayed.

👾 First Start Wizard - TortoiseGit 📃	٢.
Authentication and credential store	
SSH (URLs look like "git@example.com")	
TortoiseGitPlink is recommended as SSH client. If you don't have a key pair yet, you should generate one. Keep the private one in a save place and set up the public key on your hosting platform. Use the PuTTY authentication agent for caching the password (done automatically if a PuTTY key is configured for a remote). For advanced tips & tricks see our <u>manual</u> and <u>FAQ</u> .	
TortoiseGitPlink Generate PuTTY key pair	
HTTP (URLs start with "http://" or "https://") By default Git does not save/cache credentials. However, you can configure a credential helper (recommended) or manually use %HOME%/_netrc.	
Credential helper:	
Don't store these settings now. Advanced	
These settings will be stored to your global git configuration (%HOME%/.gitconfig) and will be used for all your git repositories as a default.	
< Back Finish Cancel Help]

- 6. To use the Sysmac Studio version control function, do not reduce the option selected from default.
- 7. Click the **Finish** button.

This completes the initial setting for "TortoiseGit".

5.6 Creating the shared folder and remote repository

This section utilizes the configuration in which a shared folder on the computer is used as the remote repository as an example.

For this configuration, create a shared folder in which to store a remote repository, and then create a folder that serves as the remote repository in the shared folder.

1. Creating a new folder

In Windows Explorer, create a new folder. You can create this folder in any location with any name. Let's assume that it is C:\Git.

2. Settings for the shared folder

Right-click the folder that you created in step 1 and select **Properties** from the pop-up menu. Then, in the dialog box that is displayed, click the **Share** tab to perform the sharing settings. Here, you configure the folder to allow full access from other users' computers on which the Sysmac Studio version control function used.

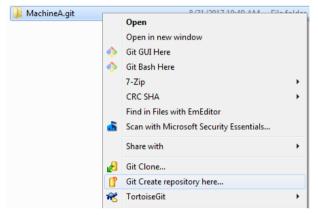
 Creating a folder to be used as a repository Create a new folder under the folder created in step1.
 According to the Git conventions, it is required that you name the folder to use as a Git repository to the repository name followed by ".git". For example, if the repository name is "MachineA," then name the folder

"MachineA.git."

► Local Disk (C:) ► Git ►		
Include in library 🔹 Share with 💌	Burn New folder	
Name	Date modified	Туре
\mu Machine.git	8/31/2017 9:45 AM	File folder

4. Registering the folder as a repository

Right-click the folder created in step 3 and select **Git Create repository here** from the pop-up menu.



A repository creation option dialog box is displayed.

5. Select the check box on the dialog box and click the OK button.



A repository is created and the following dialog box is displayed.



6. Click the **OK** button.

This completes the creation of a remote repository.

Note that you can control only one Sysmac Studio project per repository. Create a directory for each project to use version control function.

5.7 Additional Setting for "TortoiseGit"

In "TortoiseGit", add settings to enable graphical comparison of Sysmac Studio projects from "TortoiseGit".

1. From the Start menu, select **All programs - TortoiseGit - Settings.** The **Settings** dialog box of the "TortoiseGit" is displayed.

2. Select **Diff Viewer** from the tree.

The **Diff Viewer** pane is displayed.

🐲 Settings - TortoiseGit		— ×-
🖌 🔌 General	C Diff Viewer	
	Configure the program used for comparing different revis	ions of files
Dialogs 1	TortoiseGitMerge	
Colors 1 Colors 2 Colors 3 Colors 3 Colors 4 Colors 3	Click on "Advanced" to specify alternate diff programs based on file extension	Advanced
⊿ - ∲ Git Credential	Configure viewer program for GNU diff files (patch files)	
Generation Generation Generation Generation Generation Generation		
Overlay Handlers Solution		
✓ Email ✓ If Viewer		
Merge Tool		
	OK Cancel	Apply Help

3. In the **Diff Viewer** pane, click the **Advanced** button. The **Advanced diff settings** dialog box is displayed.

	Program	-
.dll	wscript.exe "C:\Program Files\TortoiseGit\Diff-Scripts\diff-dll.vbs" %bas.	
.doc	wscript.exe "C:\Program Files\TortoiseGit\Diff-Scripts\diff-doc.js" %bas	1
.docm	wscript.exe "C:\Program Files\TortoiseGit\Diff-Scripts\diff-doc.js" %bas	Ĩ
.docx	wscript.exe "C:\Program Files\TortoiseGit\Diff-Scripts\diff-doc.js" %bas	
.exe	wscript.exe "C:\Program Files\TortoiseGit\Diff-Scripts\diff-dll.vbs" %bas.	_
.nb	wscript.exe "C:\Program Files\TortoiseGit\Diff-Scripts\diff-nb.vbs" %ba	
.ods	wscript.exe "C:\Program Files\TortoiseGit\Diff-Scripts\diff-odt.vbs" %ba.	
.odt	wscript.exe "C:\Program Files\TortoiseGit\Diff-Scripts\diff-odt.vbs" %ba.	
.orm	C:\Program Files (x86)\OMRON\Sysmac Studio\SysmacDiff-gitdiff %bas	
.ppt	wscript.exe "C:\Program Files\TortoiseGit\Diff-Scripts\diff-ppt.js" %bas	
nntm	wscrint eve "C+\Program Files\TortoiseGit\Diff-Scrints\diff-ont is" %has III	
•		
·		_

4. Click the Add button.

The Add extension specific diff program dialog box is displayed.

Add extension specific o	liff program	3
Extension:	.oem	
External Program:	C:\Program Files\OMRON\\$ysmac Studio\bin\\$ysmacDiff.exe -gitdiff %base %mine)]
	OK Cancel]

5. Enter the following text string, and then click the **OK** button.

Items	Text string to enter
Extension	.oem
External Program	(Sysmac Studio install folder ^{*1})\ SysmacDiff.exe
	-gitdiff %base %mine %bpath %brev %yrev

*1: The Sysmac Studio installation folder is by default as follows.

Sysmac Studio (32 bit)

Windows 32bit edition: C:\Program Files\OMRON\Sysmac Studio

Windows 64bit edition: C:\Program Files (x86)\OMRON\Sysmac Studio

Sysmac Studio (64 bit)

Windows 64bit edition: C:\Program Files\OMRON\Sysmac Studio

- 6. Click the OK button in the Advanced diff settings dialog box.
- 7. Click the **OK** button in the **Settings** dialog box.

This completes the additional setting for "TortoiseGit".

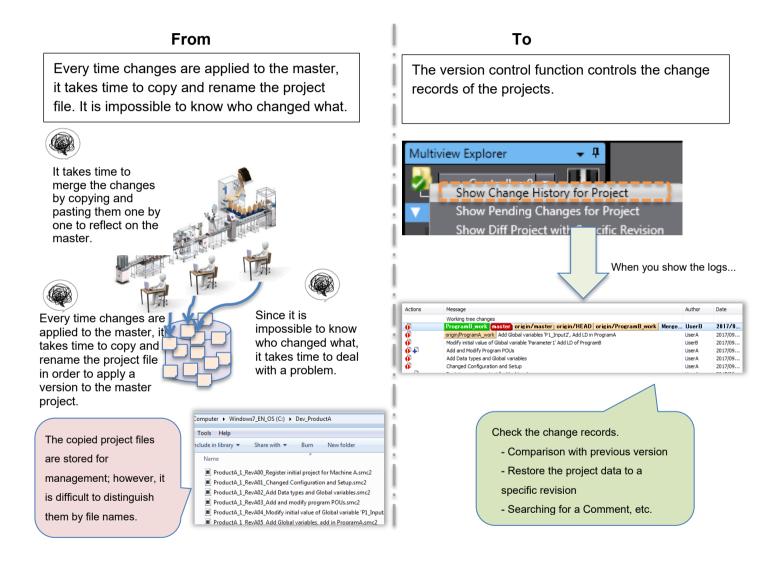
6 PROJECT RECORD CONTROL ON SYSMAC STUDIO

6.1 Overview of Project Record Control Using the Version Control Function

The version control system "Git" which works with the Sysmac Studio has the following four basic functions.

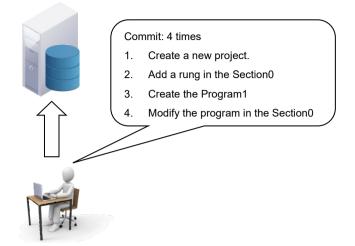
- Generates an identification number that uniquely identifies changes in saved software data and stores it with comments
- Searches for changes in specific software data by date information, comments, user name, etc.
- Comprehends (compares) differences from arbitrary software data
- Restores to software data at specific change time

The Git controls records of Sysmac Studio project data using above basic functions.

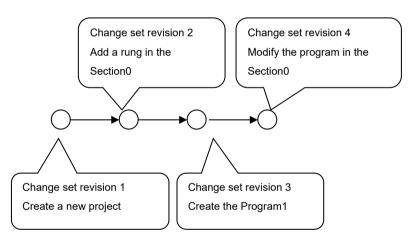


6.2 Operation Example of Record Control Function

This section describes an operation example to use the record control function. E.g. Executing Commit four times to update the information in the repository.



When above course of operation is explained with the idea of change set which is a control unit of the version control system, the change information is managed as a sequential information of revision 2 -> revision 3 -> revision 4 (*) that started from the starting point of revision 1 (*). Since the information of differences between revisions and change set are combined and managed, a project opened on the Sysmac Studio can be restored to a specific revision.



*The revisions are expressed as 1, 2, 3, and 4 for simplicity of explanation, but in reality it is a numeric value of 40 digits hexadecimal number generated by the SHA - 1 algorithm, and it is not serial numbers that increase one by one.

6.3 Preparing for Starting Project Control (Creating a Base Project and Saving It)

To start controlling the versions of a Sysmac Studio project, create a base project, save it in the local repository and configure the synchronization settings of the local and remote repository.

6.3.1 **Creating a Base Project**

1. From the Start menu, select All Programs - OMRON - Sysmac Studio - Sysmac Studio to start the Sysmac Studio.

The User Account Control dialog box is displayed according to the Windows version or user authority. Click the Yes button.

2. Create a new base project and register it.

This example uses the settings in the following table.

Here, the procedure to create a new project and register it in the repository is explained. Use the same procedure to register an already created project in the repository.

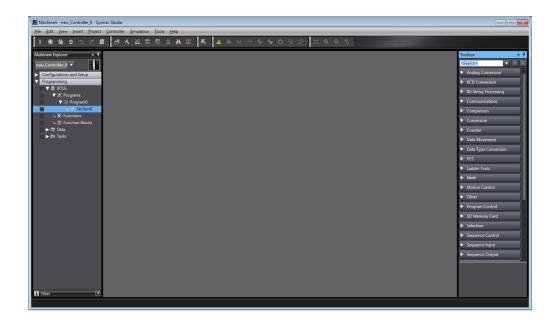
On the Start page of the Sysmac Studio, select New Project and configure the necessary setting items, and then select the Create button.

	<u> </u>
Items	Entry/Selection
Project name:	MachineA
Author	UserA
Туре	Standard Project
Category	Controller
Device	NJ501-1500
Version	1.16 ^{*1}

*1: When Controller is selected for Category, specify the Version 1.16 or later. With the Unit of unit version 1.15 or earlier, version control function cannot be used.

💼 Project Pro	operties
Project name	MachineA
Author	UserA
Comment	
Туре	Standard Project
Select D	Device
Category	Controller 🔹
Device	NJ501 🔽 - 1500 🔽
Version	1.16

The project with the project name MachineA is opened.



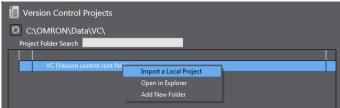
- 3. From the Main menu, select File Save to save the project.
- 4. Select File Close to close the project.

6.3.2 Registering the project in the Local Repository

1. On the Start page of the Sysmac Studio, select **Version Control Explorer**. The **Version Control Projects** window is displayed.

E Version Control Projects	
C:\OMRON\Data\VC\	
Project Folder Search	
Project Path	
	<u>O</u> pen

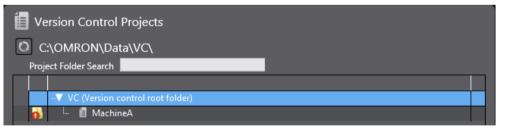
2. Right-click the working folder VC (Version control root folder) and select Import a Local Project.



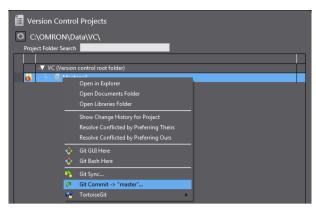
The Import a Local Project dialog box is displayed.

Minport a Local Project	
CLocal Projects	
🗹 MachineA	
	>
Author UserA	
Created 8/31/2017 10:20:29 AM Last Modified 8/31/2017 10:20:29 AM	
Comment	
Destination Folder C:\OMRON\Data\VC\ MachineA	
<u>I</u> mport	<u>C</u> ancel

3. Select the project which you created and then click the **Import** button. The selected project is copied to the working folder for version control.



4. Right-click the copied project and select **Git commit -> "master"** from the pop-up menu.

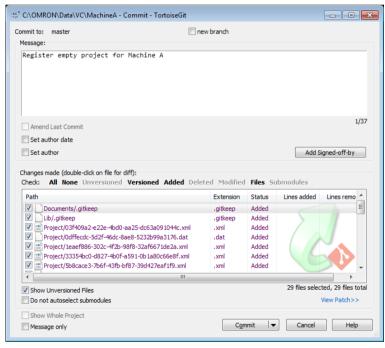


The dialog box asking for attention during Commit operation is displayed. (Hereafter, this dialog box is omitted in the explanation of operating procedure in this manual.)

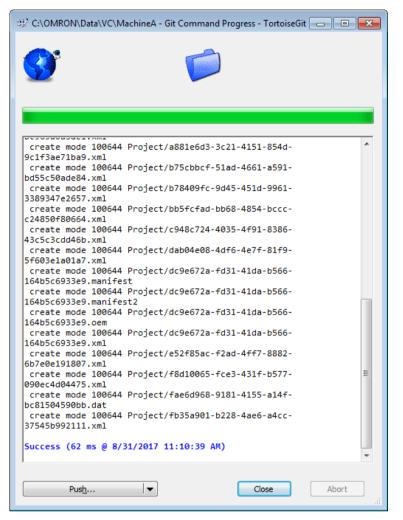
5. Confirm the description on the dialog box, select the **Don't show this again** check box, and then click the **OK** button.

The "TortoiseGit" **Commit** dialog box is displayed.

6. Enter a message that you want to leave as a record in the message area and click the **Commit** button.



The below dialog shows that Commit is successfully completed.



This completes the registration of the project in the computer to the local repository.

7. Click the **Close** button to exit.

6.3.3 Synchronizing the Local Repository and Remote Repository

"Push" the project data for version control from the local repository to the remote repository.

1. Right-click the target project in the **Version Control Projects** window and select **TortoiseGit - Push** from the pop-up menu.

Version	Control Projects		
O C:\OM	RON\Data\VC\		
Project Fold	er Search		
	C (Version control root folder)		
	Open in Explorer		
	Open Documents Folder		Pull
	Open Libraries Folder		Fetch
	Show Change History for Project	2	Push
	Resolve Conflicted by Preferring Theirs	۵,	Diff
	Resolve Conflicted by Preferring Ours	۹,	Diff with previous version
	🚸 Git GUI Here	20	Show log
	 Git GUI Here Git Bash Here 	ie ie	Show log Show Reflog
	Git Bash Here		
	Git Bash Here	* E	Show Reflog
	 Git Bash Here Git Sync 	- 4	Show Reflog Browse References

The "TortoiseGit" **Push** dialog box is displayed.

Ref		
Push all branch	hes	
Local:	master	-
Remote:		•
estination		
• Remote:		Manage
Arbitrary URL:	:	
Force: May discar	d known changes unknown (For slow network connections)	changes
	· · · · · · · · · · · · · · · · · · ·	
Include Tags	``````````````````````````````````````	
Include Tags	y Key	
Include Tags Autoload Putty Set upstream/	``````````````````````````````````````	
Include Tags Autoload Putty Set upstream/ Always push to	y Key Track remote branch o the selected remote archive for this local branch o the selected remote branch for this local branch	

- 2. Select the **Push all branches** check box, and then click the **Manage** button under **Destination**. A dialog box appears for setting the location of the remote repository of the repository "MachineA".
- 3. In the **Remote** field, enter the name of this connection (usually "Origin") and the URL (here, enter the path of the remote repository created in Section 5.5) and click the **Add New / Save** button.

양같 C:\OMRON\Data\VC\MachineA - S	ettin	ıgs - TortoiseGit		—
 Hook Scripts Issue Tracker Integration Issue Tracker Config Issue Tracker Config Icon Overlays Icon Set Overlay Handlers Network Email Email Off Viewer Merge Tool Saved Data TortoiseGittBlame TortoiseGittBlame 	E	Remote:		Origin Rename C:\Git\MachineA.git Reachable Prune (All remotes)
			ОК	Cancel Apply Help

The remote name "Origin" is registered.

4. Click the **OK** button.

The dialog box asking for attention during Push operation is displayed. (Hereafter, this dialog box is omitted in the explanation of operating procedure in this manual.)

5. Confirm the description on the dialog box, select the **Don't show this message again** check box, and then click the **Yes** button.

In the "TortoiseGit" Push dialog box, the connection named "Origin" is set.

🔽 Push all brar	ICRES
Local:	master 👻
Remote:	· · · · · · · · · · · · · · · · · · ·
Destination	
Remote:	Origin Manage
Arbitrary UR	u:
Options	
Force: May disc	card 📃 known changes 📃 unknown changes
Use Thin Pag	ck (For slow network connections)
Include Tags	s
📝 Autoload Pu	
	m/track remote branch
	the theorem is the descent the condition. Can this is well been also
Always push	n to the selected remote archive for this local branch n to the selected remote branch for this local branch

6. Click the **OK** button.

The project "MachineA" in the local repository is copied to the remote repository "MachineA.git".

👾 C:\OMRON\Data\VC\MachineA - Git Command Progress - TortoiseGit
Writing objects
while objects
git.exe pushallprogress "Origin"
Counting objects: 30, done. Delta compression using up to 4 threads. Compressing objects: 100% (27/27), done. Writing objects: 100% (30/30), 18.42 KiB 0 bytes/s, done. Total 30 (delta 4), reused 0 (delta 0) To C:/Git/MachineA.git * [new branch] master -> master Success (436 ms @ 8/31/2017 11:52:50 AM)
Create pull request Abort

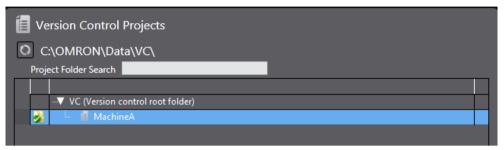
7. Click the **Close** button.

6.4 Updating the Project

This section describes the procedure to change project data and reflect it in the repository.

6.4.1 Changing the Project Data

1. On the Start page of the Sysmac Studio, select **Version Control Explorer** to open the **Version Control Projects** window. Select the project to edit and click the **Open** button.



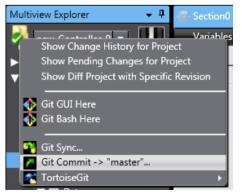
2. Modify the project data. In this example, BOOL variables "WorkDetect" and "LightLamp" are added as global variables and ladder rungs are added in the Program0-Section0.

🖶 Section0 - Program	0 International Variat	oles X				
Name	Data Type	Initial Value	I AT	Retain	Constant	Network Publish
WorkDetect	BOOL					o not publish
LightLamp	BOOL					o not publish
Multiview Explorer		Ver Global Variables				LightLamp

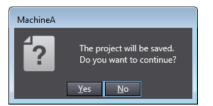
6.4.2 Registering Changes to the Local Repository (Commit)

Apply the changes made to the project to the local repository. This operation is called Commit.

1. Right-click the folder icon in the Multiview Explorer and select **Git Commit -> "master"** from the pop-up menu.



A dialog box is displayed to ask if you need to save the project.

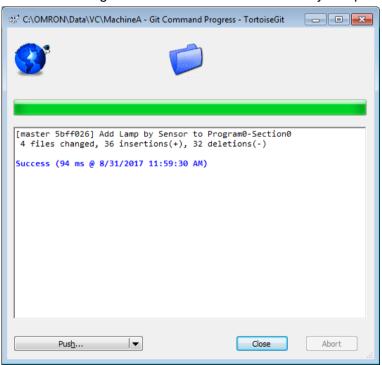


2. Click the Yes button.

The "TortoiseGit" Commit dialog box is displayed.

ిజీ C:\OMRON\Data\VC\MachineA - Commit - TortoiseGit			[- • •
Commit to: master new Message:	branch			
Add Lamp by Sensor to <u>Program@_Section@</u>				
Amend Last Commit				1/40
Set author			Add Sig	ned-off-by
Changes made (double-click on file for diff): Check: All None Unversioned Versioned Added D	eleted Mo	dified Files	Submodules	
Path	Extension	Status	Lines added	Lines rem 📥
Project/5b8cace3-7b6f-43fb-bf87-39d427eaf1f9.xml	.xml	Modified	2	E
Project/c948c724-4035-4f91-8386-43c5c3cdd46b.xml	.xml	Modified	1	
Project/dc9e672a-fd31-41da-b566-164b5c6933e9.oem	.oem	Modified	31	
<				4
Show Unversioned Files			4 files selec	ted, 4 files total
Do not autoselect submodules			V	iew Patch>>
Show Whole Project				
Message only	Com	mit 🛛 🔻	Cancel	Help

3. Enter a message that you want to leave as a record in the message area and click the **Commit** button. The below dialog shows that Commit is successfully completed.



The changes in the project in the computer were registered in the local repository.

4. Click the **Close** button.

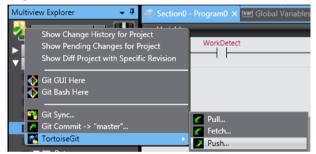
The project under editing is displayed.

Multiview Explorer 🗸 🗜	Section0 - Program0 × 🗺 Global Variables	
New_Controller_0 🔻 📊	Variables	
Configurations and Setup	0 WorkDetect LightLamp	_
Programming		
V 🗐 POUs		
V 🕅 Programs		
🔻 🖂 Program0		
∟ 🛃 Section0		
L Functions		
L 🕼 Function Blocks		
💙 🥅 Data		
L 5명 Data Types		
L 🔤 Global Variables		
► 🖿 Tasks		

6.4.3 Registering the Changes to the Remote Repository (Push)

Apply the changes made to the project in the local directory to the remote repository. This operation is called Push.

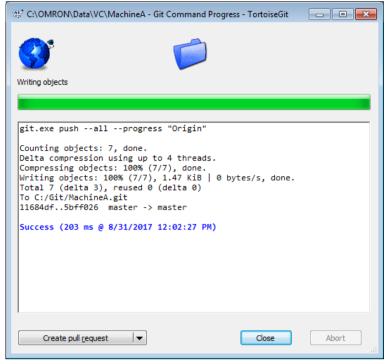
1. Right-click the folder icon in the Multiview Explorer and select **TortoiseGit - Push** from the pop-up menu.



The "TortoiseGit" Push dialog box is displayed.

Ref	
V Push all bran	iches
Local:	master 💌
<u>R</u> emote:	•
Destination	
• Re <u>m</u> ote:	Origin Manage
Arbitrary UR	L:
Options Force: May disc	ard 📃 known changes 🕅 unknown changes
Options Force: May disc	ard 📄 known changes 📄 unknown changes k (For slow network connections)
Dptions Force: May disc Use Thin Pac	ard 📄 known changes 📄 unknown changes k (For slow network connections)
Dptions Force: May disc Use Thin Pac Include Tags Autoload Put	ard Indexe Index I
Dptions Force: May disc Use Thin Pac Indude Tags Autoload Put Set upstrean Always push	ard I known changes I unknown changes ck (For slow network connections) tty Key n/track remote branch to the selected remote archive for this local branch
Dptions Force: May disc Use Thin Pac Indude Tags Autoload Put Set upstrean Always push	ard Interpretending Interprete

2. Make sure that the **Push all branches** check box is selected and **Remote** item is correctly set ("Origin" is set in this example), and then click the **OK** button.



This concludes the Push process.

3. Click the **Close** button.

6.5 Searching for a Project

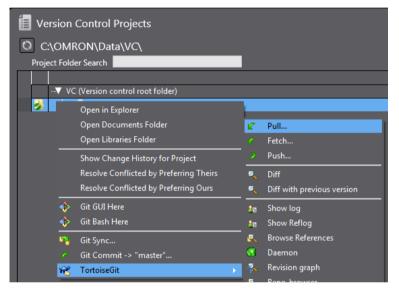
When you made a wrong change while editing a project, you might want to restore the previous project data and resume editing. In that case, you check the change contents of the past projects, change date, person who made changes, and change history information, or open the projects to identify the project revision to return. If you use the version control function, you can search for a revision of the project by the following procedure.

6.5.1 Updating the Local Repository

In a case that data in the remote repository is the master, it is necessary to update the local repository in the computer to the latest state.

This section describes the steps to be taken after updating the local repository and remote repository since some changes were made in the same procedure described in *6.4 Updating the Project*.

1. Right-click the target project in the Version Control Projects window and select TortoiseGit - Pull from the pop-up menu.



The "TortoiseGit" Pull dialog box is displayed.

🕸 C:\OMRON\Data\VC\Machir	neA - Pull - TortoiseGit	×
Remote		
Remote:	Origin]
⊘ Arbitrary <u>U</u> RL:	▼]
Remote <u>B</u> ranch:	master 🗸	
Options		
Squash	No Commit	
No East Forward	Fast Forward Only	
Tags	Default: Reachable	
Prune		
AutoLoad Putty Key	Manage Remotes	
🔲 Launch Rebase After Fetch	1	
	OK Cancel Help	.

2. Make sure that the registered remote name is correctly set (in this example, "Origin" is set) for the "Remote" item, and then click the **OK** button.

"Pull" is executed.

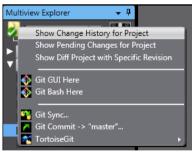
💥 C:\OMRON\Data\VC\MachineA - Git Command Progress - TortoiseGit 💦 💼 💷 💌
🔮 📁
git.exe pullprogress -vno-rebase "Origin" master
From C:/Git/MachineA * branch master -> FETCH_HEAD = [up to date] master -> Origin/master Already up-to-date.
Success (250 ms @ 8/31/2017 12:15:50 PM)
Pulled Diff Close Abort

3. Click the **Close** button.

6.5.2 Displaying Project Logs

Show the change history for the project.

- On the Start page of the Sysmac Studio, select Version Control Explorer and open the Version Control Projects window. Select the target project and click the Open button. The project is opened.
- 2. Right-click the folder icon in the Multiview Explorer and select **Show Change History for Project** from the popup menu.



The "TortoiseGit" Log Messages dialog box is displayed.

In the upper area, Actions (change, addition, or deletion), Message (log messages entered at Commit), Author

aster		From:	8/31/2017 -	To: 8/31/20	17 👻 issa	ges, Paths, Aut	hors, Emails, SF 🗛	Author Email 🔹	<u>_</u>]
Graph	Actions	Message	Message		Auth	nor Date			
згарп	Acuons	-	ree changes						
•	ø		origin/master	Thomas Limit CV	to lles	- 9/21/	2017 2:14:35		
I	Ä	Add Limit			User		017 2:14:20 PM		
I	ă		by Sensor to Pro	ram0-Section0	User		017 2:13:10 PM		
•	Ĩ₽		empty project for I		User		017 2:10:53 PM		
c	e LimitSW								
				Extension	Status	Lines added	Lines removed	1	
Path	t/3fa4049a-ca	a62-44ed-a	112-41dfa2cb7f7 08d-e731c0fa6cb	ł.oem .oem	Status Modified	Lines added 1 1	Lines removed 1 1		
Path Project	t/3fa4049a-cr /4d20cbb2-4 revision(s), fr	a <mark>62-44ed-</mark> a 44a-4cb6-a	112-41dfa2db7f7 08d-e731c0fa6db	4.oem .oem D.xml .xml	Modified Modified vision(s) sele	1	1		

(user names), and Date of each Commit are listed.

6.5.3 Searching for the Revision

In the search field at the top of the window, enter the keyword to search.

For example, if you enter "Lamp", only the revisions with message containing "Lamp" are displayed in the list.

aster		From:	8/31/2017	•	To:	8/31/201	7 🔻 🔑	lamp	X	Author Emai	
Graph	Actions	Message					Aut	nor Date			
	ø		tree changes by Sensor f		gram0-S	ection0	User	A 8/31/20)17 2:13:10 PM		
Path						Extension	Status	Lines added	Lines removed	1	
Path						Extension	Status	Lines added	Lines removed	1	
howing 1	revision(s), fr	om revisior	n cdc5df1 to	revisio				Lines added		8	

In this way, you can sort and search for the target revision by the character string of messages, date or user name.

6.6 Comparing the Projects

You can compare the revision searched and a project which is currently opened.

1. On the "TortoiseGit" **Log Messages** dialog box, right-click on the target revision and select **Compare with** working tree.

			8/31	1/2017 -	To:	8/31/2017	•	🔎 lamp	E	3	Author Email 🔻 🛧
Graph	Actions	Message						Author	Date		
		Working t	ree d	nanges							
	ø	Add Lamp	Q	Compare	with wo	orking tree			8/31/2017 2:13:10	PM	
			9	Show cha	anges as	unified diff					

The **Sysmac Diff** dialog box shows up and the project of the selected revision and one currently opened are compared on it.

The contents with differences are shown with red bars.

t:fcb2734 (HEAD -> master) Add Lam	Target: Update Date and	Source: Update Date and	Source:Working project	Detailed Comp	
1achineA	ĺ	li		i i	
	8/31/2017 12:29:39 PM	8/31/2017 12:29:39 PM	 Sharing setting between devices 		
new_Controller_0	8/31/2017 12:29:40 PM	8/31/2017 2:27:10 PM	w▼ new_Controller_0		
Configurations and Setup			Configurations and Setup		
	8/31/2017 12:29:40 PM	8/31/2017 12:29:40 PM	- EtherCAT Configuration		
	8/31/2017 12:29:40 PM	8/31/2017 12:29:40 PM	- EtherCAT Configuration and Set	1	
	8/31/2017 12:29:40 PM	8/31/2017 12:29:40 PM	- CPU/Expansion Racks Configurat		
V CPU/Expansion Racks Configurat	i 8/31/2017 12:29:40 PM	8/31/2017 12:29:40 PM	CPU/Expansion Racks Configurat	i	
CPU Rack	8/31/2017 12:29:40 PM	8/31/2017 12:29:40 PM	CPU Rack		
	8/31/2017 12:29:40 PM	8/31/2017 12:29:40 PM	- Power Supply : NJ-Px3001		
	8/31/2017 12:29:40 PM	8/31/2017 12:29:40 PM	CPU : NJ501-1500		
	8/31/2017 12:29:40 PM	8/31/2017 12:29:40 PM	Units		
- I/O Map	8/31/2017 12:29:40 PM	8/31/2017 12:29:40 PM	- I/O Map		
Controller Setup	8/31/2017 12:29:40 PM	8/31/2017 12:29:40 PM	-▼ Controller Setup		
- Operation Settings	8/31/2017 12:29:40 PM	8/31/2017 12:29:40 PM	- Operation Settings		
- Built-in EtherNet/IP Port Settin	8/31/2017 12:29:40 PM	8/31/2017 12:29:40 PM	Built-in EtherNet/IP Port Settin	1	
✓ Motion Control Setup	8/31/2017 12:29:40 PM	8/31/2017 12:29:40 PM	▼ Motion Control Setup		
	8/31/2017 12:29:40 PM	8/31/2017 12:29:40 PM	- Axis Settings		
- Axes Group Settings	8/31/2017 12:29:40 PM	8/31/2017 12:29:40 PM	Axes Group Settings		
	8/31/2017 12:29:40 PM	8/31/2017 12:29:40 PM	Cam Data Settings		
- Event Settings	8/31/2017 12:29:40 PM	8/31/2017 12:29:40 PM	- Event Settings		
···· Task Settings	8/31/2017 12:29:40 PM	8/31/2017 12:29:40 PM	- Task Settings		
Data Trace Settings	8/31/2017 12:29:40 PM	8/31/2017 12:29:40 PM	- Data Trace Settings		
- Comparison ID for Configuration	8/31/2017 12:29:40 PM	8/31/2017 2:27:10 PM	- Comparison ID for Configuration		
	8/31/2017 12:29:40 PM	8/31/2017 12:29:40 PM	- 3D equipment model		
Library	8/31/2017 12:29:40 PM	8/31/2017 2:28:21 PM	- Library		
Programming			V Programming		
V POUs			V POUs		
✓ Programs			-▼ Programs		
L.▼ Program0	8/31/2017 12:29:40 PM	8/31/2017 12:29:40 PM	Program0		
··· Variables	8/31/2017 2:25:32 PM	8/31/2017 2:25:32 PM	Variables		
Section0	8/31/2017 2:25:32 PM	8/31/2017 2:27:10 PM	Section0		
- Functions			- Functions		
- Function Blocks			- Function Blocks		
▼ Data			▼ Data		
Data Types	8/31/2017 12:29:40 PM	8/31/2017 12:29:40 PM	Data Types		
	8/31/2017 2:13:01 PM	8/31/2017 2:27:10 PM	Global Variables		
			Sel		th left Ci
Bifferent Left side only Right side		d Overwriting targets with	left	Overwr	rite C

2. To reference the details, select the · · · button in the **Detail Compare** column.

Then, the ladder diagrams are displayed, and you can grasp the difference between the revisions.

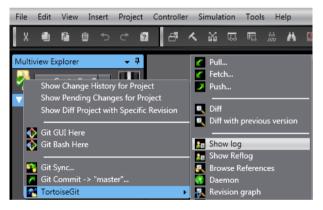
	×
II @ @ @ \ + + -	<u>م</u>
Target: new, Controller, 0 (Section0 - Program0) Source: new, Controller, 0 (Section0 - Program0)	
0 Workterst temp	
Sin Copy Selected from Left to Right 🐉 Oundo Selected 🐇 Oundo Selected	All
Legend Let side only Eight sid	se

3. After referencing the difference, click the **Close** button.

6.7 Reverting the Project

Open the project of the revision that was searched and restore the project data.

1. Right-click the folder icon in the Multiview Explorer and select **TortoiseGit – Show log** from the pop-up menu.



The "TortoiseGit" Log Messages dialog box is displayed.

2. Select one row after another up to the revision whose contents you want to restore while holding down the Shift key, right-click it and select **Revert change by these commits** from the pop-up menu.

C:\OMR	ON\Data\363c5	95c-f1cd-4e80-b2f7-2f5d48c9cfd6.oem - Log Messages - TortoiseGit	
master	F	om: 9/ 7/2017 🔻 To: 9/ 7/2017 👻 Messages, Paths, Authors, Emails, S	на
Graph	Actions	Message Author Date	
•	Ø	Working tree changes master Origin/master Change LimitSW to UserA 9/7/2017 12	2:44
ł	• •	Add Limit SW Q Compare revisions Add Lamp by Sensor to Program Q Compare revisions Register empty project for Mad Q Show changes as unified diff Sensor Sensor Show log of 3e978fbc56b79f Show log of 3e978fbc56b79f	
		Provide the sector of the s	

The result dialog box of "TortoiseGit" is displayed.



3. Click the **Commit** button.

The "TortoiseGit" Commit dialog box is displayed.

ommit to: Message: -	master		new branch					
Revert	"Add Limit	SW"						
This re	verts commi	t 3e978f	b6b78b05e4	lfd4366d	de60f162	260aef3a	cfb.	
Amend	Last Commit							1/
	thor date							
	thor date					Add	Signad of	fby
Set aut						Add	l Signed-ofi	f-by
🔲 Set aut	thor	k on file for a	liff):			Add	l Signed-ofi	f-by
Set aut				Added	Deleted		-	
Set aut	thor ade (double-clic			Added	Deleted	Modified	Files Su	
Set aut Changes m Check: Path	thor ade (double-clic	versioned	Versioned		Extensio	Modified	Files Su	ıbmod
Set aut Changes m Check: Path	thor Iade (double-clich All None Univ	versioned	Versioned		Extensio	Modified	Files Su	ıbmod
Set aut Changes m Check: Path	thor Iade (double-clich All None Univ	versioned	Versioned		Extensio	Modified	Files Su	ıbmod
Set aut	thor Iade (double-clich All None Univ	versioned -f1cd- 4e 80-b	Versioned		Extensio	Modified n Status Modifie	Files Su	ibmod s addeo
Set aut Changes m Check: Path Path Pr A Show U	thor ade (double-clici All None Unit oject/363c595c-	flcd-4e80-b	Versioned		Extensio	Modified n Status Modifie	Files Su	ibmod s addeo

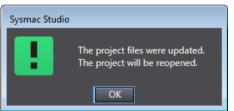
In the **Message** area, comments on the revert processing are automatically displayed.

4. Confirm that the check boxes for all files listed under **Changes made** are selected, and click the **Commit** button.

A dialog box is displayed to indicate the completion of the commit processing.

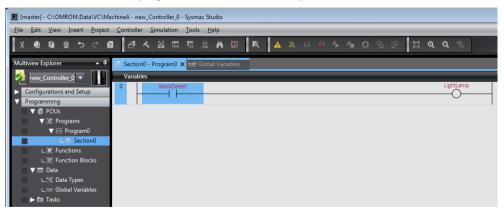
🗱 C:\OMRON\Data\VC\MachineE - Git Command Progress - TortoiseGit 💼 🔳 🗾
🔮 💋
[master ebd33cf] Revert "Add Limit SW"
1 file changed, 1 insertion(+), 1 deletion(-)
Success (62 ms @ 9/7/2017 12:47:18 PM)
Push V Close Abort

5. Click the **Close** button, and then on the "TortoiseGit" **Log Messages** dialog box click the **OK** button. A confirmation dialog box is displayed.



6. Click the **OK** button.

The project is saved and reverted project is opened. As shown below, the project is restored to the specified revision.



7 DEVELOPMENT BY MULTIPLE DEVELOPERS AND DERIVED DEVELOPMENT OF SYSMAC STUDIO PROJECT

In order to shorten the development period of controller systems, some projects are carried out concurrently by multiple developers. This development method is called "development by multiple developers". In addition, as variations of production machine increase, it is required to develop a common program for derived machines with different hardware configurations. This development method is called "derived development".

This section explains the development by multiple developers and derived development which utilize the version control function of the Sysmac Studio projects.

- Utilization of version control function in development by multiple developers and derived development
- "Branch" and "Merge" which are essential for development by multiple developers and derived development
- Operating procedure of development by multiple developers and derived development

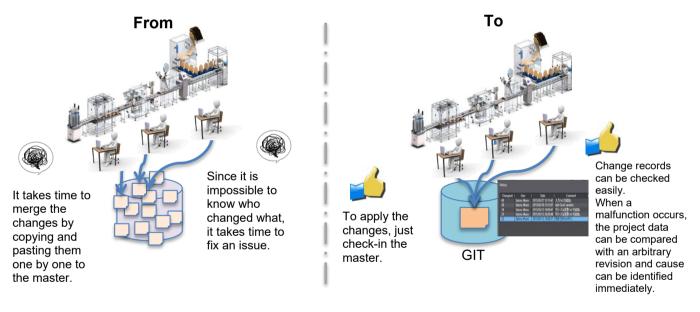
7.1 Utilization of version control function in development by multiple developers and derived development

(1) Development by multiple developers

In development carried out by multiple developers, it takes time to correctly reflect all the changes made by each developer to the master project.

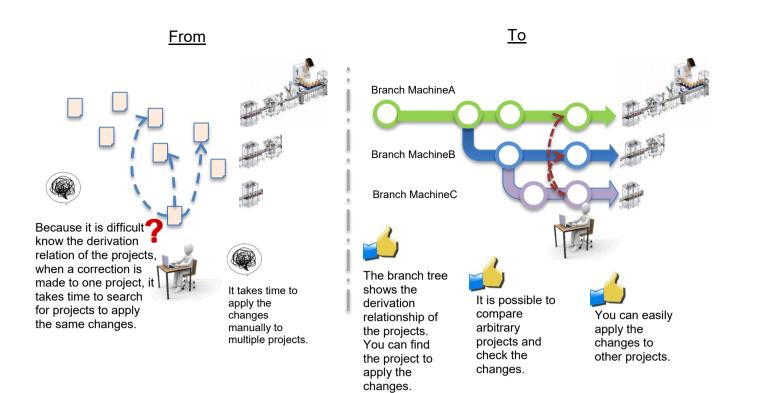
If different changes have been made to the same portion by multiple developers, the changes applied to the master project earlier might be canceled. For this case, errors might occur and it takes much labor hour to find the causes and correct the mistakes.

The record control function and merge function of the version control allow you to reliably apply changes of each developer to the master project.



(2) Derived development

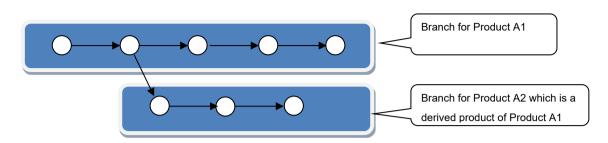
When projects for multiple machines are individually managed in derived development, mistakes might occur or some changes can be omitted when applying changes to the common program of each project. The record control function and merge function of the version control allow you to reliably find changes in multiple derived machine projects and apply the changes easily.



7.2 Branch and Merge

A branch represents a series of change record information.

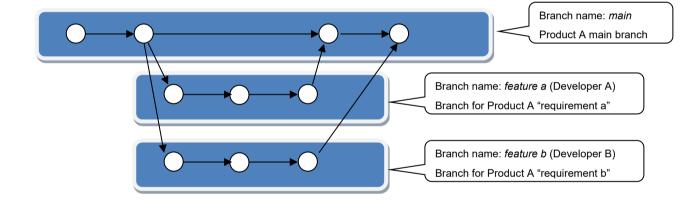
At the time of creating and starting to use the repository, there is one branch. However, another branch can be created (branching) from a change set, and the branched project is managed without affecting the change of the original branch.



If you use the branches in concurrent development by multiple developers or development for version upgrading, project data can be managed separately so that it does not affect other projects which other developers are editing or ones for already released products.

In development by multiple developers or derived development, changes made in branches may also be applied to the original branch.

At this time, we use the function called "merge" that combines the original branch with other branches.

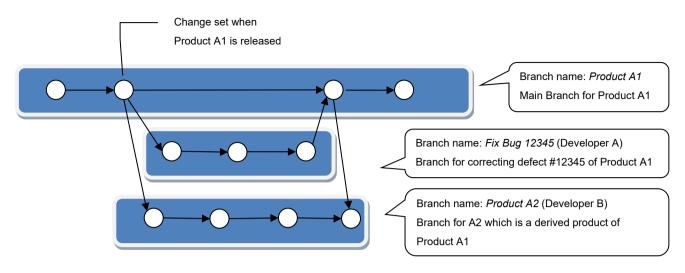


(1) Example of branching and merging in concurrent development

As shown in the above figure, a branch *main* is created to be the starting point. From the *main* branch, to carry out development for "requirement a" and" b" concurrently, create *feature a*, a branch for "requirement a" and *feature b*, a branch for "requirement b".

When development in both branches is completed and satisfactory quality is achieved, merge them to the *main* branch.

(2) Example of branching and merging in derived development



As shown in above figure, when you develop A2 which is a derivative product of product A1, create a branch *Product A2* from the change set of branch *Product A1* at the time of product release.

When a problem occurs in product A1 and it is necessary to modify the project, create a branch *Fix Bug 12345* for defect correction from the change set at the time of product A1 release. When quality is secured in the branch, merge it to the original branch *Product A1*.

If it is necessary to apply this modification not only to Product A1 but also to Product A2, you can apply the changes in the branch *Fix Bug 12345* all together to the branch *Product A2*. Merge the change set of branch for Product A1 after merging fix bug branch into the branch of Product A2.

7.3 Sysmac Studio Project Development by Multiple Developers

This section describes how to develop the Sysmac Studio project concurrently by multiple developers, example of sharing the development works and operation procedure.

7.3.1 Sharing Edit Works to Concurrently Develop a Project by Multiple Developers

The version control function has a feature to help multiple developers to edit one project concurrently and to combine their changes when they "Push" their changes to the remote repository.

However, if multiple developers edit the same portion at the same time and try to merge the changes, a conflict occurs and merging fails. To avoid this problem, a specific developer edits the data which does not support the development by multiple developers ("No" in below table), and data which allows multiple developers to edit ("Yes" in below table) can be edited by multiple developers.

Data name	Edit by multiple developers
EtherCAT	No
CPU/Expansion Racks	No
I/O Map	No
Controller Setup	No
Axis Settings	No
Axes Group Settings	No
Cam Data Settings	No
Event Settings	No
Task Settings	No
Programs	Yes
Functions	Yes
Function Blocks	Yes
Data Types	No
Global Variables	No

7.3.2 Example of Dividing Tasks

This section describes how to divide the roles of developers and the scenario as a prerequisite of the next section.

7.3.2.1 How to Divide Tasks

In this example, two developers (Developer A and B) are working on a project. Developer A configures Configurations and Setup, Data Types, and Global Variables, and develops ProgramA. Developer B develops ProgramB.

Multiview Explorer	Developer A	
 Programming POUs Programs ProgramA ProgramB Functions Function Blocks E Data E Cobal Variables Tasks 	Developer A	Developer A Developer B

7.3.2.2 Scenario

The Developer A and B work on above mentioned tasks of the project in the procedure of the below table.

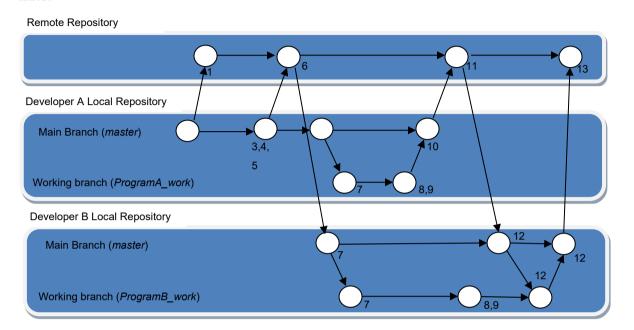
No. Develope D datas Notified bit in the transition of transit transit transit transition of transition of transition of transi		Peveloper A and B work on above mentic		
repository remote repository remote repository 2 Load the project from the remote repository - 7.3.2 Load the project from the remote repository (No.2: Developer A) 3 Configure Configurations and Setup. - 7.3.3 Commit the project in which Configurations and Setup. 4 Set the Data Types, Global Variables, and program POUs - 7.3.3 Commit the project in which Configurations and Setup. 5 Commit the project in which Configurations and Setup. - - 6 Push the changes in the local repository to the remote repository "After this operation. Developer A and B can concurrently edititic project data. - 7 Create the working branch occurrently edititic project data. - - 7 Create the working branch of Developer A in the local repository - - 8 Develop Program A Add Global Variables Develop Program B or Add Global Variables - 9 Commit changes of Program A and Global Variables Commit changes of Program A and Global Variables - 9 Commit changes of vorking branch of Developer B) - - - 7.3.3.10 Create the working branch of Developer B) - -<	No.	Developer A tasks	Developer B tasks	Reference for the tasks
Developer A registers the base project in the repository. Developer A) 2 Load the project from the remote repository. 7.3.2.2 Load the project from the remote repository (No.2: Developer A) 3 Configurations and Setup. - 4 Set the Data Types, Global Variables, and program POUs - 5 Commit the project in which Configurations and Setup, Data Types, Global Variables, and program POUs are registered (No. 3.4,5: Developer A) - 6 Push the changes in the local repository to the remote repository "After this operation. Developer A in the local repository to the remote repository to the computer of Developer B and create the local repository ' Create the working branch of Developer A (No.7: Developer B) - 7 Create the working branch of Developer A (No.7: Developer B) - 7.3.3.6 Commit changes of Program A Add Global Variables Develop Program B Global Variables to the working branch - 9 Commit changes of Program A and Blobal Variables to the working branch of Developer B) - - 11 Push the changes in the local repository of Developer A) - - 12 - Obtain the latest data from the main branch (No.10: Developer A) - 12 - Obtain the latest data from the main branch (No.10: Developer A) </td <td>1</td> <td>o</td> <td>-</td> <td></td>	1	o	-	
repository. Point the project from the remote 2 Load the project from the remote - 3 Configure Configurations and Setup - 4 Set the Data Types, Global Variables, - - 7 Additional Setup, Data Types, Global Variables, - - 7 Commit the project in which - - Comfigurations and Setup, Data Types, Global Variables, and program POUs are registered - 6 Push the changes in the local repository to the remote repository (No.6: Developer A) 7 Create the working branch of Developer A in the local repository to the remote for Developer B and create the local repository (No.6: Developer A) 7 Create the working branch of Developer A (No.7: Developer B and create the local repository (No.6: Developer A) 7 Create the working branch of Developer A (No.7: Developer B and create the local repository (No.6: Developer B) 8 Develop Program A Add Global Variables Change the initial values of Change the working branch of Developer A) 7.3.3.0 Create the working branch of Developer B 8 Develop Program A and Global Variables to the working branch of Developer B 9 Commit changes of Program A and Global Variables to the working branch of De				
2 Load the project from the remote repository - 7.3.2.2.0.41 the project from the remote repository (No.2: Developer A) 3 Configure Configurations and Setup, and program POUs - - 7.3.3.2.Comfit the project in which Configurations and Setup, Data Types, Global Variables, and program POUs are registered -				Developer A)
repository remote repository (No.2: Developer A) 3 Configure Configurations and Setup, and program POUs - 5 Commit the project in which Configurations and Setup, Dat Types, Global Variables, and program POUs are registered (No. 3.4,5: Developer A) 6 Push the changes in the local repository (Io the remote repository "After this operation, Developer A 7 Create the working branch of Developer A in the local repository "After this operation, Developer A 7 Create the working branch of Developer A in the local repository "After this operation, Developer A 8 Develop Program A Add Global Variables tranch Commit changes of Program A and Global Variables of Clobal Variables 8 Develop Program A Add Global Variables to the working branch of Developer B Create the working branch of Developer B 9 Commit changes of Program A and Global Variables to the working branch to the main branch "The project in the local repository created in No.1 is the main branch "The project in the local repository created in No.1 is the main branch "The project in the local repository created in No.1 is the main branch "The project in the local repository created in No.1 is the main branch "The project in the local repository created in No.1 is the main branch "The project in the local repository created in No.1 is the main branch (No.8, 9: Developer B) 10 Merge the changes of working branch to the main branch (No.10: Developer A) 11 Push	2		-	7.3.3.2 Load the project from the
3 Configure Configurations and Setup - 7 7 as 3 Cormit the project in which Configurations and Setup, Data Types, Global Variables, and program POUs 7 7 as 3 Cormit the project in which Configurations and Setup, Data Types, Global Variables, and program POUs are registered 7 7 as 3 Cormit the project in which Configurations and Setup, Data Types, Global Variables, and program POUs are registered 7 7 as 3 Program A 7 7 as 3 Program POUs are registered 7 7 as 3 Cormit the project in the local repository to the remote repository to the computer of Developer B and create the local repository to the remote repository to the computer of Developer B and create the local repository to the remote repository in the Developer B computer (No.7 Developer B) 7 Commit changes of Program A Add Global Variables Commit changes of Program A and Global Variables to the working branch of Developer B) 7 Commit changes of Program A and Global Variables to the working branch (No.8, 9: Developer B) 7 7 7 Add Global Variables Develop Program B and Global Variables to the working branch (No.8, 9: Developer B) 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7	-			
3 Configure Configurations and Setup		lopolitory		
4 Set the Data Types, Global Variables, and program POUs which Configurations and Setup, Data Types, Global Variables, and program POUs are registered (No. 3,4,5; Developer A) 6 Commit the project in which - repository solutions and Setup, Data Types, Global Variables, and program POUs are registered (No. 3,4,5; Developer A) 7,3,3,4 Push the changes in the local - repository to the remote repository to the remote repository to the remote repository or No.6; Developer A) 7 Create the working branch of Developer A in the local repository • Create the working branch of Developer B and create the local repository • Create the working branch of Developer B and create the local repository • Create the working branch of Developer B and create the local repository • Create the working branch of Developer B and create the local repository • Create the working branch of Developer B and create the local repository • Create the working branch of Developer B and create the local repository • Create the working branch of Developer B and create the working branch of Developer B and Global Variables to the working branch of Developer B and Global Variables to the working branch of Developer B and Global Variables to the working branch of Developer A) • 7.3.3 Commit changes of Program A and Global Variables to the working branch (No.8, 9; Developer B) 10 Merge the changes of working branch ho No.10; Developer A) • 7.3.3.4 Merges the changes of working branch of Developer A) 11 Push the changes in the local repository of Developer A to the remote repository of Developer A to the remote repository of Developer A to the remote repo	3	Configure Configurations and Setup	_	
and program POUs Data Types, Global Variables, and program POUs are registered (No. 3,4,5: Developer A A) 6 Push the changes in the local repository to the remote repository oncurrently edit the project data. - 7.3.4 Push the changes in the local repository to the remote repository oncurrently edit the project data. 7 Create the working branch Developer A in the local repository of Developer A in the local repository oreate the local repository to the computer of Developer Band create the local repository oreate the working branch of Developer B Change the initial values Global Variables to the working branch - 7.3.7 Commit changes of Program A and Global Variables on the Developer A) 8 Develop Program A Add Global Variables to the working branch Develop Program B and Global Variables to the working branch (No.8, 9: Developer A) 7.3.3.10 Commit changes of Program A and Global Variables to the branch (No.8, 9: Developer A) 10 Merge the changes of working branch to the main branch - 7.3.3.10 Commits changes of Program A and Global Variables to the branch (No.8, 9: Developer A) 11 Push the changes in the local repository of Developer A to the repository of Developer A to the remote repository (No.11: Developer A) - 12 - Obtain the latest data from the main branch of Developer B to to the remote repository (No.12: Developer B)			_	
5 Commit the project in which - Configurations and Setup, Data Types, Global Variables, and program POUs are registered (No. 3,4,5: Developer A) and program POUs are registered (No. 3,4,5: Developer A) 6 Push the changes in the local repository to the remote repository to the computer of Developer A and B can computer of Developer B and create the local repository to the computer of Developer B and create the local repository to the computer of Developer B and create the local repository to the computer of Developer B (No. 7: Developer B) 8 Develop Program A Add Global Variables Develop Program A 9 Commit changes of Program A and Global Variables 10 Merge the changes of working branch of the main branch - 11 Push the changes of working branch of the main branch - 11 Push the changes of working branch of the main branch - 12 - - 13 - Obtain the latest data from the main branch of Developer A and B can be merged. 13 - Push the changes in the local repository to the remote repository to the re	-			
Configurations and Setup, Data Types, Global Variables, and program (A) 6 Push the changes in the local repository to the remote repository oncurrently edit the project data. - 7.3.4 Push the changes in the local repository to the remote repository to the remote repository to the remote repository to the remote repository to the remote repository to the computer of Developer B and create the local repository - 7.3.3 Create the working branch of Developer A (No. 7. Developer B) 7 Create Develop Program A Add Global Variables Develop Program B Global Variables - 7.3.3 Commit changes of Program A and Global Variables to the working branch 7.3.3.10 Commit changes of Program B and Global Variables to the working branch (No.8, 9: Developer B) 9 Commit changes of Program A Add Global Variables - 7.3.3.10 Commit changes of Program B and Global Variables to the working branch (No.8, 9: Developer A) 10 Merge the changes of working branch to the main branch *. The project in the local repository created in No, 1 is the main branch *. The project in the local repository created in No, 1 is the main branch - 7.3.3.10 Commit changes of Program B and Global Variables to the branch (No.8, 9: Developer A) 11 Push the changes in the local repository of Developer A to the remote repository (No.11: Developer A) - 7.3.3.10 Lordmits changes of Program B and Global Variables to the main branch (No.12:	5		_	
Types Global Variables, and program A) 6 Push the changes in the local repository to the remote repository to the remote repository to the remote repository (No.6: Developer A) - 7 Create the working branch of Developer A in the local repository - - 7 Create the working branch of Developer A in the local repository - - 8 Developer A in the local repository - - • Create the working branch of Developer A in the local repository - • Create the working branch of Developer A in the local repository - • Create the working branch of Developer B and create the local repository - • Create the working branch of Developer B (No.7: Developer B) 7.3.3.9 Create the working branch of Developer B (No.7: Developer B) 8 Develop Program B Global Variables Develop Program B Global Variables to the working branch (No.8, 9: Developer B) 9 Commit changes of Program A and Global Variables to the working branch (No.8, 9: Developer B) - 10 Merge the changes of working branch to the main branch to the main branch to the main branch (No.10: Developer A) - 11 Push the changes in the local repository of Developer A to the main	5	1 7	-	
POUs are registered - 6 Push the changes in the local repository to the remote repository "After this operation. Developer A and B can concurrently edit the project data. - 7.3.3.4 Push the changes in the local repository to the remote repository (No.6: Developer A) 7 Create the working branch Developer A in the local repository - <t< td=""><td></td><td></td><td></td><td>•</td></t<>				•
6 Push the changes in the local repository to the remote repository (No.6: Developer A) concurrently edit the project data. - 7.3.3.4 Push the changes in the icoal repository (No.6: Developer A) 7 Create the working branch of Developer A in the local repository - - - 7 Create the working branch of Developer A in the local repository - - - 0 Create the working branch of Developer A in the local repository - - - 0 Create the working branch of Developer A in the local repository - - - 0 Create the working branch of Developer B and create the local repository - - - 0 Create the working branch of Developer B - - - - 0 Commit changes of Program A and Global Variables Commit changes of Program B and Global Variables to the working branch of Developer A) -				A)
repository to the remote repository Iocal repository to the remote repository (No.6: Developer A) 7 Create the working branch of Developer A in the local repository • Copy the target project in the remote repository to the computer of Developer B and create the local repository • Copy the target project in the remote repository to the computer of Developer B and create the local repository • Copy the target project in the remote repository (No.7: Developer A) 8 Develop Program A Add Global Variables Develop Program B Change the initial values of Global Variables to the working branch of Developer B) 7.3.3 Commit changes of Program A and Global Variables to the working branch (No.8, 9: Developer A) 9 Commit changes of Program A and Global Variables to the to the to the to the to the remote repository created in No.8, 9: Developer A) Developer A) 10 Merge the changes of working branch of Developer A - 7.3.3.10 Commit changes of Program B and Global Variables to the working branch (No.8, 9: Developer B) 10 Merge the changes of working branch of Developer A - 7.3.3.10 Commits changes of working branch of Developer A) 11 Push the changes in the local repository created in Rober A and B can be merged. - 7.3.3.11 Push the changes in the local repository (No.11: Developer A) 12 - Obtain the latest data from the main branch of Developer B is working branch of Developer B - 7.3.3.12 Merges the changes in the main branc	6			7.2.2.4 Duch the changes in the
 *After this operation. Developer A and B can concently edit the project data. 7 Create the working branch of Developer A in the local repository Copy the target project in the remote repository to the computer of Developer B and colar pository Create the working branch of Developer B and colar pository Create the working branch of Developer B and colar pository Create the working branch of Developer B and colar pository Create the working branch of Developer B and colar pository Create the working branch of Developer B computer (No.7: Developer B) 7.3.3 Create the working branch of Developer B (No.7: Developer B) 7.3.3 Commit changes of Program A and Global Variables Commit changes of Program A and Global Variables to the working branch (No.8, 9: Developer B) Commit changes of Program A and Global Variables to the working branch (No.8, 9: Developer B) Merge the changes of working branch to the main branch of Developer A. The project in the local repository created in No.1 is the main branch of Developer A. Push the changes in the local repository of Developer A. Push the changes in the local repository of Developer A. Push the changes in the local repository of Developer A. Push the changes in the local repository of Developer A. Push the changes in the local repository of Developer A. Push the changes in the local repository of Developer A. Cotal repository of Developer A. Push the changes in the main branch. Push the changes in the main branch of Developer B working branch. Push the changes in the main branch of Developer B to the working branch of Develop	0	5	-	
concurrently edit the project diata. Copy the target project in the remote repository to the computer of Developer B and create the local repository Copy the target project in the remote repository to the computer of Developer B and create the local repository Copy the target project in the remote repository to the computer of Developer B and create the local repository Copy the target project in the remote repository Copy the target project in the remote repository Copy the target project in the computer of Developer B and create the local repository Copy the target project in the computer of Developer B and create the local repository Copy the target project in the computer of Developer B and create the local repository Copy the target project in the computer of Developer B and create the local repository Copy the target project in the computer of Developer B and create the local repository Copy the target project in the Decal Project in the Decal Project in the Developer A (No. 7: Developer B) Commit changes of Program A and Global Variables to the working branch Commit changes of Program A and Global Variables to the working branch (No. 8, 9: Developer A) 9 Commit changes of working branch to the main branch of Developer A (No. 10: Developer A) - 7.3.3.10 Commits changes of Program B and Global Variables to the working branch (No. 8, 9: Developer B) 10 Merge the changes of working branch of Developer A to the remote repository of Developer A to the remote repository of Developer A to the remote repository of Developer A and B can be merged. - 7.3.3.11 Push the changes in the local repository of Developer A)				
7 Create the working branch of Developer A in the local repository · Copy the target project in the remote repository to the computer of Developer B and create the local repository · Copy the target project in the remote repository to the computer of Developer B and create the local repository · Create the working branch · Copy the target project in the remote repository to the computer of Developer B and create the local repository · Create the working branch · Create the working branch · Developer B and create the local repository 8 Develop Program A Add Global Variables Develop Program B Change the initial values of Global Variables to the working branch 7.3.3.7 Commit changes of Program A and Global Variables to the working branch (No.8, 9: Developer A) 9 Commit changes of Program A and Global Variables to the working branch to the main branch - 7.3.3.8 Merges the changes of Program B and Global Variables to the working branch (No.8, 9: Developer B) 10 Merge the changes of working branch to the main branch of Developer A to the remote repository or developer A to the remote repository or developer A to the remote repository or Developer A to the remote repository of Developer B is to the main branch and merge it to Developer B is to the main branch of Developer B is to the main branch of Developer B is				repository (No.6: Developer A)
Developer A in the local repositoryDeveloper B computer of Developer B and create the local repository7.3.3.6 Create the working branch of Developer B A) - Create the working branch8Develop Program A Add Global VariablesDevelop Program B Change the initial values of Global Variables7.3.3.7 Commit changes of Program A and Global Variables9Commit changes of Program A and Global Variables to the working branchCommit changes of Program A and Global Variables10Merge the changes of working branch to the main branch * The project in the local repository or developer A to the main branch of Developer A to the remote repository of Developer A and B can be merged7.3.3.12 Merges the changes in the local repository of Developer B to the main branch (No.12: Developer B's working branch, Then, remerge the data after then, remerge the data after the main branch of Developer B form the local repository to the remote the remote repository (No.13:12-Obtain the latest data from the main branch and merge in the main branch of Developer B form the local repository to the remote repository (No.13:13	7		· Copy the target project in the	
 a bevelop Program A Add Global Variables Commuter of Developer B and create the local repository Create the working branch Create the working branch Create the working branch Create the working branch Bevelop Program A Add Global Variables Global Variables Global Variables to the working branch Merge the changes of Program A and Global Variables to the main branch T.3.3.10 Commit changes of Program A and Global Variables to the working branch (No.8, 9: Developer B) Merge the changes of working branch to the main branch The project in the local repository created in No.1 is the main branch of Developer A. Push the changes in the local repository of Developer A. Push the changes made by Developer B' Obtain the latest data from the main branch of Developer A. Cottain the latest data from the main branch of Developer A. Cottain the latest data from the main branch of Developer A. Cottain the latest data from the main branch of Developer B' Atter this operation, changes made by Developer B's working branch (No.12: Developer B) Cottain the latest data from the main branch of Developer B's working branch (No.12: Developer B) Atter this operation, changes made by Developer B's working branch of Developer B's working branch (No.12: Developer B) Atter this operation, changes made by Developer B's working branch (No.12: Developer B) Cottain the latest data from the main branch of Developer B's working branch (No.12: Developer B) Atter this operation, changes made by Developer B's working branch (No.12: Developer B)		0	., .,	7.3.3.6 Create the working branch
Create the local repository • Create the working branchCreate the working branch • Create the working branchCreate the working branch • Create the working branch of Developer B (No.7: Developer B) 7.3.3 Create the local repository in the Developer B (No.7: Developer B) 7.3.3 Create the working branch of Developer B (No.7: Developer B) 7.3.3 Create the working branch of Developer B (No.7: Developer B) 7.3.3 Create the working branch of Developer B (No.7: Developer B) 7.3.3 Create the working branch of Developer B (No.7: Developer B) 7.3.3 Create the working branch of Developer B (No.7: Developer B) 7.3.3 Create the working branch to the working branch (No.8, 9: Developer A)9Commit changes of Program A and Global Variables to the working branch branchCommit changes in Program B and Global Variables to the working branch7.3.3 Commit changes of Program A and Global Variables to the working branch (No.8, 9: Developer B)10Merge the changes of working branch to the main branch * The project in the local repository created in repository of Developer A11Push the changes in the local repository of Developer A to the remote repository * After this operation, changes made by Developer A and B can be merged12-Obtain the latest data from the main branch and merge it to Developer B's working branch the working branch of Developer C1313-Push the changes in the merging to the main branch branch of Developer B to the remote repository (No.13:13				of Developer A (No.7 Developer
 Create the working branch Create the working branch Create the working branch Create the working branch Bevelop Program A Add Global Variables Commit changes of Program A and Global Variables to the working branch Commit changes in Program B and Global Variables to the working branch to the main branch Merge the changes of working branch to the main branch Merge the changes of working branch to the main branch Tube the changes in the local repository of Developer A to the remote repository After this operation, changes made by Developer A and B can be merged. Obtain the latest data from the main branch and B can be merged. Obtain the latest data from the main branch and B can be merged. Push the changes in the local repository of Developer A to the remote repository Push the changes made by Developer B's working branch, Then, remerge the data after Push the changes in the main branch of Developer B to the working branch of Developer B to the main branch (No.12: Developer B) Obtain the latest is the main branch of Developer B to the remote repository (No.13: 				A)
1 Periodic to the main branch of Developer A and B can be merged. Obtain the latest data from the main branch of Developer A to the remote repository (No.12: Developer A) 12 - Obtain the latest data from the main branch of Developer A to the remote repository (No.12: Developer B) 13 - Push the changes in the local repository (No.13: to the main branch of Developer B) 13 - Push the changes in the local repository (No.13: to the main branch of Developer B)			create the local repository	<Ďeveloper B>
8Develop Program A Add Global VariablesDevelop Program B Change the initial values of Global Variables7.3.3.9 Create the working branch of Developer B (No.7: Developer B)9Commit changes of Program A and Global Variables to the working branchDevelop Program B Global Variables7.3.3.7 Commit changes of Program A and Global Variables9Commit changes of Program A and Global Variables to the working branch to the main branchCommit changes in Program B and Global Variables to the working branch7.3.3.10 Commit changes of Program B and Global Variables to the working branch to the main branch of Developer A.10Merge the changes of working branch to the main branch of Developer A7.3.3.8 Merges the changes of working branch to the main branch (No.8, 9: Developer B)11Push the changes in the local repository of Developer A7.3.3.11 Push the changes in the local repository of Developer A.12-Obtain the latest data from the main branch and merge it to Developer A and B can be merged12-Obtain the latest data from the main branch of Developer B's working branch Then, remerge the data after merging to the main branch (No.12: Developer B)7.3.3.12 Merges the changes in the the working branch (No.12: Developer B)13-Push the changes in the changes in the merging to the main branch of Developer B to the working branch of Developer B from the main branch of Developer B from the merging to the main branch of Developer B to the working branch of Developer B from the main branch of Developer B from the main branch of Developer B from the main b			 Create the working branch 	7.3.3.5 Create the local repository
8 Develop Program A Develop Program B 7.3.3.9 Create the working branch of Developer B (No.7: Developer B) 8 Develop Program A Develop Program B 7.3.3.7 Commit changes of 9 Commit changes of Program A and Global Variables Commit changes in Program B 7.3.3.10 Commits changes of 10 Merge the changes of working branch to the main branch - 7.3.3.8 Merges the changes of 10 Merge the changes of working branch to the main branch - 7.3.3.8 Merges the changes of 11 Push the changes in the local repository created in No.1 is the main branch of Developer A. - 7.3.3.11 Push the changes in the local repository of Developer A. 12 - Obtain the latest data from the main branch and merge it to Developer B's working branch. Then, remerge the data after merging to the main branch of Developer B - 13 - Push the changes in the local repository to the remote repository (No.13:			_	in the Developer B computer
8 Develop Program A Develop Program B 7.3.3.7 Commit changes of 9 Commit changes of Program A and Commit changes in Program B 7.3.3.7 Commit changes of 9 Commit changes of Program A and Commit changes in Program B 7.3.3.7 Commit changes of 9 Commit changes of Program A and Commit changes in Program B 7.3.3.10 Commits changes of 9 Commit changes of working branch Commit changes in Program B 7.3.3.10 Commits changes of 10 Merge the changes of working branch - 7.3.3.8 Merges the changes of 10 Merge the changes in the local repository created in - 7.3.3.11 Push the changes in the 11 Push the changes in the local repository of Developer A. - 7.3.3.11 Push the changes in the 11 Push the changes made by - 7.3.3.11 Push the changes in the - 12 - Obtain the latest data from the 7.3.3.12 Merges the changes in + 12 - Obtain the changes in the main branch - 7.3.3.13 Push the changes in the 13 - - Obtain the changes in the main branch of Developer B's working branch. - - 13 -				(No.7: Developer B)
B) 8 Develop Program A Add Global Variables Develop Program B Change the initial values of Frogram A and Global Variables Global Variables to the working branch Commit changes in Program B Global Variables to the working branch Commit changes in Program B Global Variables to the working branch Commit changes in Program B 10 Merge the changes of working branch * The project in the local repository created in * The project in the local repository created in * The project in the local repository of Developer A. 11 Push the changes in the local repository of Developer A to the remote repository * After this operation, changes made by Developer A and B can be merged. 12 - 13 - 13 -				7.3.3.9 Create the working branch
8 Develop Program A Add Global Variables Develop Program B Change the initial values of Global Variables 7.3.3.7 Commit changes of Program A and Global Variables to the working branch (No.8, 9: Developer A) 9 Commit changes of Program A and Global Variables to the working branch Commit changes in Program B and Global Variables to the working branch 7.3.3.10 Commit changes of Program B and Global Variables to the working branch (No.8, 9: Developer A) 10 Merge the changes of working branch to the main branch * The project in the local repository created in No.1 is the main branch of Developer A. - 11 Push the changes in the local repository of Developer A to the remote repository - 7.3.3.11 Push the changes in the local repository of Developer A to the remote repository of Developer A to the remote repository (No.11: Developer A) 12 - Obtain the latest data from the main branch and merge it to Developer B's working branch. Then, remerge the data after merging to the main branch 7.3.3.12 Merges the changes in the working branch (No.12: Developer B) 13 - Push the changes in the main branch of Developer B from the local repository to the remote remote repository (No.13:				of Developer B (No.7: Developer
Add Global VariablesChange the initial values of Global VariablesProgram A and Global Variables to the working branch (No.8, 9: Developer A)9Commit changes of Program A and Global Variables to the working branchCommit changes in Program B and Global Variables to the working branchProgram A and Global Variables to the working branch to the main branch10Merge the changes of working branch to the main branch-7.3.3.10 Commits changes of Program B and Global Variables to the branch (No.8, 9: Developer B)10Merge the changes of working branch to the main branch * The project in the local repository created in No.1 is the main branch of Developer A11Push the changes in the local repository of Developer A to the remote repository * After this operation, changes made by Developer A and B can be merged12-Obtain the latest data from the main branch and merge it to Developer B's working branch. Then, remerge the data after merging to the main branch7.3.3.12 Merges the changes in the local repository (No.12: Developer B)13-Push the changes in the local repository to the remote repository (No.13:				B)
Global Variablesto the working branch (No.8, 9: Developer A)9Commit changes of Program A and Global Variables to the working branchCommit changes in Program B and Global Variables to the working branchto the working branch (No.8, 9: Developer A)10Merge the changes of working branch to the main branch * The project in the local repository created in No.1 is the main branch of Developer A11Push the changes in the local repository of Developer A to the remote repository-7.3.3.12 Nerges the changes in the local repository of Developer A.12-Obtain the latest data from the main branch and merge it to Developer B's working branch. The, remerge the data after merging to the main branch7.3.3.13 Push the changes in the local repository (No.12: Developer B)13-Push the changes in the main branch of Developer B from the local repository to the remote repository (No.13:	8	Develop Program A	Develop Program B	7.3.3.7 Commit changes of
9 Commit changes of Program A and Global Variables to the working branch Commit changes in Program B and Global Variables to the working branch Developer A) 10 Merge the changes of working branch to the main branch - 7.3.3.8 Merges the changes of working branch to the main branch (No.8, 9: Developer B) 10 Merge the changes of working branch to the main branch - 7.3.3.8 Merges the changes of working branch to the main branch (No.10: Developer A) 11 Push the changes in the local repository of Developer A to the remote repository - 7.3.3.11 Push the changes in the local repository of Developer A to the remote repository (No.11: Developer A) 12 - Obtain the latest data from the main branch and merge it to Developer B's working branch. Then, remerge the data after merging to the main branch 7.3.3.13 Push the changes in the working branch (No.12: Developer B) 13 - Push the changes in the local repository to the remote 7.3.3.13 Push the changes in the main branch of Developer B from the local repository to the remote		Add Global Variables	Change the initial values of	Program A and Global Variables
Global Variables to the working branchand Global Variables to the working branch7.3.3.10 Commits changes of Program B and Global Variables to the branch (No.8, 9: Developer B)10Merge the changes of working branch to the main branch * The project in the local repository created in No,1 is the main branch of Developer A7.3.3.8 Merges the changes of working branch to the main branch (No.10: Developer A)11Push the changes in the local repository of Developer A to the remote repository * After this operation, changes made by Developer A and B can be merged7.3.3.11 Push the changes in the local repository of Developer A to the remote repository (No.11: Developer A)12-Obtain the latest data from the main branch and merge it to Developer B's working branch. Then, remerge the data after merging to the main branch7.3.3.12 Merges the changes in the working branch. Then, remerge the data after merging to the main branch13-Push the changes in the local repository to the remote7.3.3.13 Push the changes in the main branch of Developer B to the remote repository (No.13:			Global Variables	to the working branch (No.8, 9:
Global Variables to the working branchand Global Variables to the working branch7.3.3.10 Commits changes of Program B and Global Variables to the branch (No.8, 9: Developer B)10Merge the changes of working branch to the main branch * The project in the local repository created in No,1 is the main branch of Developer A7.3.3.8 Merges the changes of working branch to the main branch (No.10: Developer A)11Push the changes in the local repository of Developer A to the remote repository * After this operation, changes made by Developer A and B can be merged7.3.3.12 Merges the changes in the local repository of Developer A to the remote repository (No.11: Developer A)12-Obtain the latest data from the main branch and merge it to Developer B's working branch. Then, remerge the data after merging to the main branch7.3.3.13 Push the changes in the main branch of Developer B to the remote repository (No.12: Developer B)13-Push the changes in the nocal repository to the remote7.3.3.13 Push the changes in the main branch of Developer B to the orking branch. Then, remerge the data after merging to the main branch7.3.3.13 Push the changes in the main branch of Developer B to the remote repository (No.13:	9	Commit changes of Program A and	Commit changes in Program B	Developer A)
branchworking branchProgram B and Global Variables to the branch (No.8, 9: Developer B)10Merge the changes of working branch to the main branch-7.3.3.8 Merges the changes of working branch to the main branch of Developer A.11Push the changes in the local repository of Developer A to the remote repository * After this operation, changes made by Developer A and B can be merged7.3.3.11 Push the changes in the local repository of Developer A to the remote repository (No.11: Developer A)12-Obtain the latest data from the main branch and merge it to Developer B's working branch. Then, remerge the data after merging to the main branch7.3.3.12 Merges the changes in the local repository (No.12: Developer B)13-Push the changes in the main branch of Developer B from the local repository to the remote7.3.3.13 Push the changes in the main branch of Developer B to the remote repository (No.13:				7.3.3.10 Commits changes of
10 Merge the changes of working branch to the main branch - 7.3.3.8 Merges the changes of working branch to the main branch of Developer A. 11 Push the changes in the local repository of Developer A to the remote repository - 7.3.3.11 Push the changes in the local repository of Developer A. 11 Push the changes made by Developer A and B can be merged. - 7.3.3.12 Merges the changes in the local repository of Developer A. 12 - Obtain the latest data from the main branch of Developer B's working branch. Then, remerge the data after merging to the main branch (No.12: Developer B) 7.3.3.13 Push the changes in the main branch (No.12: Developer B) 13 - Push the changes in the changes in the local repository to the remote repository to the remote 7.3.3.13 Push the changes in the main branch (No.12: Developer B)		branch	working branch	
10 Merge the changes of working branch to the main branch - 7.3.3.8 Merges the changes of working branch to the main branch of Developer A. 11 Push the changes in the local repository of Developer A. - 7.3.3.11 Push the changes in the local repository of Developer A to the remote repository * After this operation, changes made by Developer A and B can be merged. - 7.3.3.12 Merges the changes in the local to the remote repository of Developer A and B can be merged. 12 - Obtain the latest data from the main branch and merge it to Developer B's working branch. Then, remerge the data after merging to the main branch 7.3.3.13 Push the changes in the main branch (No.12: Developer B) 13 - Push the changes in the changes in the main branch of Developer B from the local repository to the remote repository (No.13:			5	to the branch (No.8, 9:
10 Merge the changes of working branch to the main branch - 7.3.3.8 Merges the changes of working branch to the main branch of Developer A. 11 Push the changes in the local repository of Developer A. - 7.3.3.11 Push the changes in the local repository of Developer A to the remote repository * After this operation, changes made by Developer A and B can be merged. - 7.3.3.12 Merges the changes in the local to the remote repository of Developer A and B can be merged. 12 - Obtain the latest data from the main branch and merge it to Developer B's working branch. Then, remerge the data after merging to the main branch 7.3.3.13 Push the changes in the main branch (No.12: Developer B) 13 - Push the changes in the changes in the main branch of Developer B from the local repository to the remote repository (No.13:				
to the main branch * The project in the local repository created in No,1 is the main branch of Developer A.working branch to the main branch (No.10: Developer A)11Push the changes in the local repository of Developer A to the remote repository * After this operation, changes made by Developer A and B can be merged7.3.3.11 Push the changes in the local repository of Developer A to the remote repository (No.11: Developer A)12-Obtain the latest data from the main branch and merge it to Developer B's working branch. Then, remerge the data after merging to the main branch7.3.3.12 Merges the changes in the working branch of Developer B to the main branch (No.12: Developer B)13-Push the changes in the main branch of Developer B from the local repository to the remote7.3.3.13 Push the changes in the main branch of Developer B to the remote repository (No.13:	10	Merge the changes of working branch	-	
* The project in the local repository created in No,1 is the main branch of Developer A. branch (No.10: Developer A) 11 Push the changes in the local repository of Developer A to the remote repository - * After this operation, changes made by Developer A and B can be merged. - 12 - Obtain the latest data from the main branch and merge it to Developer B's working branch. Then, remerge the data after merging to the main branch 7.3.3.13 Push the changes in the working branch (No.12: Developer B) 13 - Push the changes in the changes in the branch of Developer B from the local repository to the remote 7.3.3.13 Push the changes in the main branch of Developer B to the remote repository (No.13:		v v		
No,1 is the main branch of Developer A. The local repositor of Developer A to the remote repository of Developer A to the remote repository 7.3.3.11 Push the changes in the local repository of Developer A to the remote repository (No.11: Developer A) 12 - Obtain the latest data from the main branch and merge it to Developer B's working branch. Then, remerge the data after merging to the main branch 7.3.3.13 Push the changes in the working branch (No.12: Developer B) 13 - Push the changes in the changes in the branch of Developer B from the local repository to the remote 7.3.3.13 Push the changes in the main branch of Developer B to the remote		* The project in the local repository created in		
repository of Developer A to the remote repository * After this operation, changes made by Developer A and B can be merged.local repository of Developer A to the remote repository (No.11: Developer A)12-Obtain the latest data from the main branch and merge it to Developer B's working branch. Then, remerge the data after merging to the main branch7.3.3.12 Merges the changes in the working branch of Developer B to the main branch (No.12: Developer B)13-Push the changes in the main branch of Developer B from the local repository to the remote7.3.3.13 Push the changes in the main branch of Developer B to the remote repository (No.13:				
remote repository * After this operation, changes made by Developer A and B can be merged.to the remote repository (No.11: Developer A)12-Obtain the latest data from the main branch and merge it to Developer B's working branch. Then, remerge the data after merging to the main branch7.3.3.12 Merges the changes in the working branch of Developer B to the main branch (No.12: Developer B)13-Push the changes in the main branch of Developer B from the local repository to the remote7.3.3.13 Push the changes in the main branch of Developer B to the remote repository (No.13:	11	5	-	
* After this operation, changes made by Developer A and B can be merged.Developer A)12-Obtain the latest data from the main branch and merge it to Developer B's working branch. Then, remerge the data after merging to the main branch7.3.3.12 Merges the changes in the working branch of Developer B to the main branch (No.12: Developer B)13-Push the changes in the main branch of Developer B from the local repository to the remote7.3.3.13 Push the changes in the main branch of Developer B to the remote repository (No.13:				
Developer A and B can be merged. Obtain the latest data from the main branch and merge it to Developer B's working branch. Then, remerge the data after merging to the main branch 7.3.3.12 Merges the changes in the working branch of Developer B to the main branch (No.12: Developer B) 13 - Push the changes in the main branch of Developer B from the local repository to the remote 7.3.3.13 Push the changes in the main branch of Developer B to the remote				
12 - Obtain the latest data from the main branch and merge it to Developer B's working branch. Then, remerge the data after merging to the main branch 7.3.3.12 Merges the changes in the working branch of Developer B's working branch. Then, remerge the data after merging to the main branch 13 - Push the changes in the main branch of Developer B from the local repository to the remote 7.3.3.12 Merges the changes in the working branch of Developer B to the working branch of Developer B to the main branch of Developer B from the local repository to the remote				Developer A)
main branch and merge it to Developer B's working branch. Then, remerge the data after merging to the main branchthe working branch of Developer B to the main branch (No.12: Developer B)13-Push the changes in the main branch of Developer B from the local repository to the remote7.3.3.13 Push the changes in the main branch of Developer B to the remote repository (No.13:	10	Developer A and b can be merged.	Obtain the latest data from the	7 3 3 12 Merces the changes in
Developer B's working branch. Then, remerge the data after merging to the main branchB to the main branch (No.12: Developer B)13-Push the changes in the main branch of Developer B from the local repository to the remote7.3.3.13 Push the changes in the main branch of Developer B to the remote repository (No.13:	12	-	-	
Then, remerge the data after merging to the main branch Developer B) 13 - Push the changes in the main branch 7.3.3.13 Push the changes in the main branch of Developer B from the local repository to the remote			-	•
merging to the main branch 13 - Push the changes in the main branch of Developer B from the local repository to the remote 7.3.3.13 Push the changes in the main branch of Developer B to the remote repository (No.13:				•
13 - Push the changes in the main branch of Developer B from the branch of Developer B from the local repository to the remote repository (No.13: 7.3.3.13 Push the changes in the main branch of Developer B to the remote repository (No.13:				Developer B)
branch of Developer B from the main branch of Developer B to local repository to the remote repository (No.13:	- 40			
local repository to the remote the remote repository (No.13:	13	-	-	•
repository Developer B)				
			repository	Developer B)

-: No task

The following figure illustrates the relationship between the repositories and branches.

The changes in the main branch of the local repository of each developer's computer are exchanged through the remote repository. On the local repository of each developer, a working branch is created and the changes are merged.

The numbers next to \bigcirc in the figure correspond to the task numbers of Developers A and B shown in the above table.



7.3.3 Operating Procedure

This section explains the operating procedure along with the description in 7.3.2.2 Scenario.

7.3.3.1 Register a project in the remote repository (No.1: Developer A)

1. Creating a remote repository

Create a remote repository to share with multiple developers. Here, create a remote repository with the folder name "C: \ Git \ MachineA.git" on Developer A's computer. Refer to *5.6 Creating the shared folder and remote repository* for details.

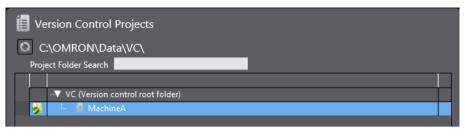
 Registering the local repository and creating and saving the base project Create a new project and register it in the local repository. Here, create a project "MachineA" in the Developer A's computer, commit it to the local repository, and then push it to the remote repository. Refer to *6.3 Preparing for Starting Project Control (Creating a Base Project and Saving It)* for details.

7.3.3.2 Load the project from the remote repository (No.2: Developer A)

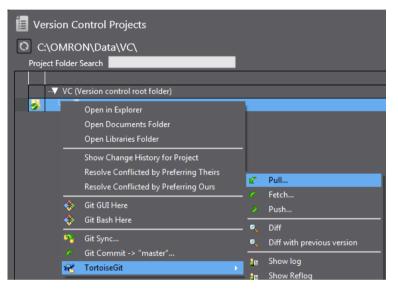
After the project is registered in the remote repository, there is a possibility that other developers are updating the target project of the remote repository. In order to synchronize the local repository in the Developer A's computer with the remote repository which functions as the master, take the following steps.

In this scenario, this operation is unnecessary immediately after the procedure in 7.3.3.1 Register a project in the remote repository (No.1: Developer A), since only Developer A accesses the remote repository.

1. On the Start page of the Sysmac Studio, select **Version Control Explorer** and open the **Version Control Projects** window.



2. Right-click the MachineA and select TortoiseGit - Pull from the pop-up menu.



The "TortoiseGit" Pull dialog box is displayed.

3. Check that "Origin" is selected in **Remote** (the path to the remote repository is set as C:\Git\MachineA.git) and click the **OK** button.

C:\OMRON\Data\VC\Machin	neA - Pull - TortoiseGit
Remote	
Remote:	Origin
O Arbitrary URL:	· · · · · · · · · · · · · · · · · · ·
Remote Branch:	master 🔻
Options	
Squash	No Commit
No Fast Forward	Fast Forward Only
Tags	Default: Reachable
Prune	
AutoLoad Putty Key	Manage Remotes
🔲 Launch Rebase After Fetch	ı
	OK Cancel Help

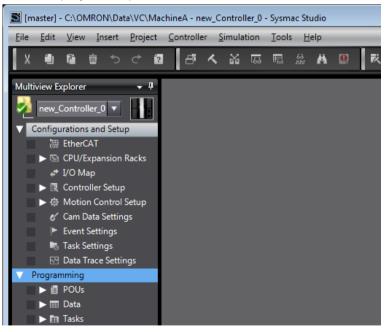
This completes Pull processing.

ೆಸ್ C:\OMRON\Data\VC\MachineA - Git Command Progress - TortoiseGit	- • •
S	
git.exe pullprogress -vno-rebase "Origin" master	
From C:/Git/MachineA * branch master -> FETCH_HEAD = [up to date] master -> Origin/master Already up-to-date.	
Success (156 ms @ 8/31/2017 4:00:54 PM)	
Pulled Diff 🛛 🗸	Abort

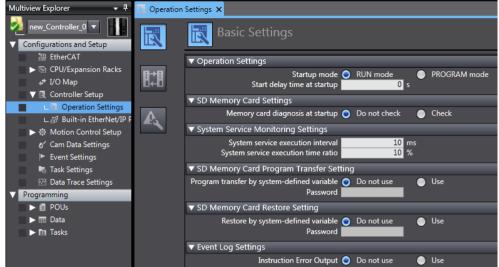
4. Click the **Close** button.

7.3.3.3 Commit the project in which Configurations and Setup, Data Types, Global Variables, and program POUs are registered (No. 3,4,5: Developer A)

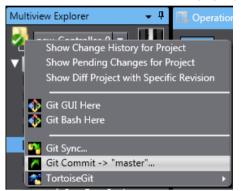
1. Select the project *MachineA* in the **Version Control Explorer**, and then click the **Open** button. The new project is opened.



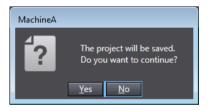
2. Set necessary data for the items in Operation Settings tab.



3. In order to Commit it to the local repository, right-click the folder icon in the Multiview Explorer and select **Git Commit -> "master"** from the pop-up menu.



The confirmation dialog box is displayed.



4. Click the Yes button.

The "TortoiseGit" Commit dialog box is displayed.

양은 C:\OMRON\Data\VC\MachineA - Commit - TortoiseGit			[- • •
	branch			
Message: Changed Configuration and Setup				
Amend Last Commit				1/32
Set author			Add Sig	ned-off-by
Changes made (double-click on file for diff): Check: All None Unversioned Versioned Added D	eleted Mo	dified Files	5ubmodules	
Path	Extension	Status	Lines added	Lines remove
Project/433d135f-2e47-4539-b492-dbdfd3e9f75d.xml	.xml	Modified	6	
🖉 🔮 Project/aa6a82a4-e3d9-45e1-b495-2c8de88e8245.xml	.xml	Modified	1	
Project/f44ff463-0846-4dad-aa4b-2e139ae702c1.oem	.oem	Modified	31	3
٠ (III	1			•
Show Unversioned Files			3 files selec	ted, 3 files total
Do not autoselect submodules			V	iew Patch>>
Show Whole Project				
Message only	Com	mit 🛛 🔻	Cancel	Help

5. Enter comment in the **Message** area and click the **Commit** button. Commit is completed.

양은 C:\OMRON\Data\VC\MachineA - Git Command Progress - TortoiseGit @	
[master 2a00d71] Changed Configuration and Setup 3 files changed, 38 insertions(+), 38 deletions(-)	
Success (78 ms @ 8/31/2017 4:06:49 PM)	
Push V	

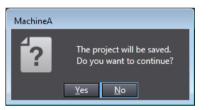
6. Click the **Close** button.

7. Then, enter the Data Types and Global Variables.

Multiview Explorer 🚽 📮	📶 Global Variables 🗙		
💋 new_Controller_0 🔻 🚽	Name	Data Type	Initial Value
Configurations and Setup	P1_Input	BOOL	
EtherCAT	P1_Output	BOOL	
CPU/Expansion Racks	Parameter1	INT	100
	Parameter2	INT	
▼ I/O Map ▼ I Controller Setup	P2_Input	BOOL	
	P2_Output	BOOL	
L □ Operation Settings L □ Automatic Operation Settings L □ Automatic Operation Settings Motion Control Setup C' Cam Data Settings Event Settings Task Settings Data Trace Settings Programming O Data L □ Data L □ Data L □ Data L □ Tasks			

 In order to Commit them to the local repository, right-click the folder icon in the Multiview Explorer and select Git Commit -> "master" from the pop-up menu.

A confirmation dialog box is displayed.



9. Click the Yes button.

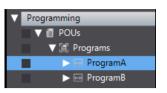
The "TortoiseGit" Commit dialog box is displayed.

$\mathfrak{G}_{\mathfrak{g}^{t,k}}^{j,k}$ C:\OMRON\Data\VC\MachineA - Commit - TortoiseGit				- • •
Commit to: master new	branch			
Add Data types and Global variables				
Amend Last Commit				1/36
Set author			Add Sig	ned-off-by
Changes made (double-click on file for diff): Check: All None Unversioned Versioned Added I	eleted Mo	dified File	s Submodules	;
Path	Extension	Status	Lines added	Lines remove
Project/3da8fac8-47c1-4490-9a9f-69a4a6d2e628.xml	.xml	Modified	6	C
Project/f44ff463-0846-4dad-aa4b-2e139ae702c1.oem	.oem	Modified	1	
•				•
Show Unversioned Files			2 files selec	ted, 2 files total
Do not autoselect submodules			V	iew Patch>>
Show Whole Project	Con	nmit 🛛 🔻	Cancel	Help

- 10. Enter comment in the **Message** area and click the **Commit** button.
- 11. Click the Close button in the Complete Commit dialog box.

12. Next, create the program POUs.

In this example, Developer A develops Program A and Developer B develops Program B. First, Developer A creates the both program POUs in advance.



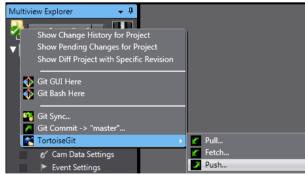
13. Commit the POUs in the same way as data types and global variables were committed.

C:\OMRON\Data\VC\MachineA - Commit - TortoiseGit			[- • •
Commit to: master	branch			
Message:				
Add and Modify Program POUs				
Amend Last Commit				1/28
Set author date				
Set author			Add Sig	ned-off-by
			, Had big	nearon by
Changes made (double-click on file for diff):			Harbig	incu on by
Changes made (double-click on file for diff): Check: All None Unversioned Versioned Added D	eleted Mo	dified Files		
	eleted Mo Extension	dified Files		Lines rem 🔺
Check: All None Unversioned Versioned Added D			5ubmodules	
Check: All None Unversioned Versioned Added D Path	Extension .xml	Status	Submodules	Lines rem 🔺
Check: All None Unversioned Versioned Added D Path Image: Comparison of the compa	Extension .xml .xml	Status Modified	Submodules	Lines rem 🔺
Check: All None Unversioned Versioned Added D Path Path Project/48629066-2c64-4aff-b04a-61091ae9bdeb.xml Deproject/5882524c-ea35-41fe-8656-e83070666a7a.xml	Extension .xml .xml	Status Modified Added	Submodules	Lines rem 🔺
Check: All None Unversioned Versioned Added D Path Image: Check of the characteristic characteristicharacteristic characteristeristic characteristic char	Extension .xml .xml	Status Modified Added	5ubmodules Lines added	Lines rem
Check: All None Unversioned Versioned Added D Path // @ Project/48629066-2c64-4aff-b04a-61091ae9bdeb.xml Ø @ Project/5882524c-ea35-41fe-8656-e83070666a7a.xml // @ Project/7882524c-ea35-41fe-8656-e83070666a7a.xml Ø @ Project/7e74a07f-d61c-4802-9bea-d6a4830a64df.xml // @ Project/7e74a07f-d61c-4802-9bea-d6a4830a64df.xml	Extension .xml .xml	Status Modified Added	5 Submodules Lines added 1 4 4 7 files selec	Lines rem
Check: All None Unversioned Versioned Added D Path Path Project/48629066-2c64-4aff-b04a-61091ae9bdeb.xml Project/5882524c-ea35-41fe-8656-e83070666a7a.xml Check Check Chec	Extension .xml .xml	Status Modified Added	5 Submodules Lines added 1 4 4 7 files selec	Lines rem
Check: All None Unversioned Versioned Added D Path Path Project/48629066-2c64-4aff-b04a-61091ae9bdeb.xml Project/5882524c-ea35-41fe-8656-e83070666a7a.xml Project/7e74a07f-d61c-4802-9bea-d6a4830a64df.xml Control of the state of the sta	Extension .xml .xml	Status Modified Added Added	5 Submodules Lines added 1 4 4 7 files selec	Lines rem

7.3.3.4 Push the changes in the local repository to the remote repository (No.6: Developer A)

Before Developer B starts working concurrently with Developer A, Developer A pushes his/her working result of common data (Configurations and Setup, Data Types, Global Variables, and program POUs) to the remote repository.

1. Right-click the folder icon in the Multiview Explorer or on the target project in the **Version Control Projects** window, and select **TortoiseGit - Push**.



The "TortoiseGit" Push dialog box is displayed.

2. Make sure that the **Push all branches** check box is selected and **Remote** item is correctly set ("Origin" is set in this example), and then click the **OK** button.

💱 C:\OMRON\Data\VC\MachineA - Push - TortoiseGit
Ref V Push all branches
Local: master
Remote:
Destination
O Arbitrary URL:
Options
Force: May discard known changes unknown changes
Use Thin Pack (For slow network connections)
Include Tags Autoload Putty Key
Set upstream/track remote branch
Always push to the selected remote archive for this local branch
Always push to the selected remote branch for this local branch
Recurse submodule None
OK Cancel Help

This completes Push processing.

🕸 C:\OMRON\Data\VC\MachineA - Git Command Progress - TortoiseGit
Writing objects
git.exe pushallprogress "Origin"
git.exe pushaiiprogress "origin"
Counting objects: 18, done. Delta compression using up to 4 threads. Compressing objects: 100% (18/18), done. Writing objects: 100% (18/18), 3.57 KiB 3.57 MiB/s, done. Total 18 (delta 12), reused 0 (delta 0) To C:/Git/MachineA.git f905f9771dba75 master -> master Success (156 ms @ 8/31/2017 4:13:46 PM)
Create pull request

3. Click the **Close** button.

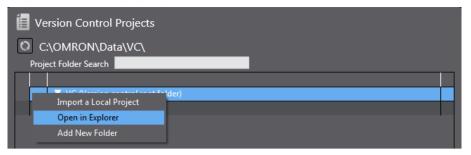
7.3.3.5 Create the local repository in the Developer B computer (No.7: Developer B)

By the operation so far, data that Developer A edited (Configurations and Setup, Data Types, and Global Variables) has been registered in the remote repository.

From this point forward, in order to make it possible for Developer B to work concurrently, obtain the target project from the remote repository and create a local repository in the Developer B's computer.

It is assumed that setup and basic settings of software in the Developer B's computer have been completed.

1. In the Version Control Projects window, right click on VC (Version Control root folder) and select Open in Explorer from the pop-up menu.



In Windows Explorer, the VC folder is opened.



2. Right-click on the arbitrary position in the VC folder and select **Git Clone**.



The TortoiseGit Git Clone dialog box is displayed.

🕸 Git clone - To	rtoiseGit				×		
Clone Existing R	epository						
URL:	□ \\SLRA0383\Git\MachineA.git ▼ Browse ▼						
Directory:	C:\OMRON\Data	NC Machine A	١		Browse		
🔲 Depth	1 🗖 Re	cursive	Clone into B	are Repo	No Checkout		
Branch		0	rigin Name		LFS		
🔽 Load Putty	r Кеу				▼		
From SVN Repos	itory						
From SVN	Repository						
Trunk:	trunk	Tags:	tags	Branch:	branches		
From:	0			Username:			
			ОК	Cancel	Help		

3. Enter the following information in the **Git clone** dialog box.

- The path to the remote repository in URL (In this example, \\SLRA0383\Git\MachineA.git, PC name: SLRA0383, shared name: Git, Repository name: MachineA.git)
- The path to the local repository in **Directory** (In this example, C:\OMRON\Data\VC\MachineA)
- 4. Click the **OK** button.

The Git clone command copied the remote repository to the Developer B's computer. After this, this repository is called "Local repository".

🐝 C:\OMRON\Data\VC\VC\M	achineA - Git Command F	Progress - TortoiseGit	- • •
9 3			
VC/MachineA"	SS -V C. (UIL (PIBLIII)	ICA. STL C. (UNINUM	
Cloning into 'C:\OMRON\ done.	\Data\VC\MachineA'		=
Success (156 ms @ 8/31/	/2017 4:24:28 PM)		•
Show log		Close	Abort

7.3.3.6 Create the working branch of Developer A (No.7 Developer A)

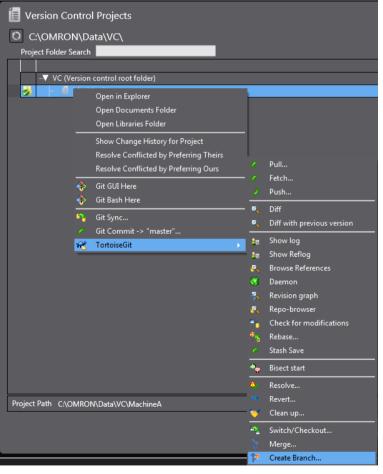
After the project was registered in the remote repository, there is a possibility that other developers are updating the target project in the remote repository. In order to synchronize the local repository in the Developer A's computer with the remote repository which functions as the master, take the following steps.

- 1. On the Start page of the Sysmac Studio, select **Version Control Explorer** and open the **Version Control Projects** window.
- 2. Right-click the MachineA and select **TortoiseGit Pull** from the pop-up menu. The "TortoiseGit" **Pull** dialog box is displayed.

3. Check that "Origin" is selected in **Remote** (the path to the remote repository is set as C:\Git\MachineA.git) and click the **OK** button.

않고 C:\OMRON\Data\VC\Machi	neA - Pull - TortoiseGit 🛛 💽
Remote	
Remote:	Origin
O Arbitrary URL:	•
Remote Branch:	master 💌 📖
Options	
Squash	No Commit
No Fast Forward	Fast Forward Only
Tags	Default: Reachable
Prune	
AutoLoad Putty Key	Manage Remotes
Launch Rebase After Fetc	h
	OK Cancel Help

- 4. Click the **Close** button in the **Complete Pull** dialog box.
- 5. Right-click the folder icon in the Multiview Explorer or on the target project in the **Version Control Projects** window, and select **TortoiseGit Create Branch**.



The "TortoiseGit" Create Branch dialog box is displayed.

6. Enter the branch name (in this example, *ProgramA_work*), and click the **OK** button. For **Base On**, select the latest revision. Select **Head (master).**

🐝 C:\OMRON\Data\\	/C\Machine	A - Create E	ranch -	TortoiseGit	t	×
Name	-					
Branch	ProgramA_	work				
Base On						
HEAD (master)						_
🔘 Branch	master					
🔘 Tag						
Commit					-	
Options						
Track	Force		Switch	n to new bra	anch	
Description						
		ОК		Cancel		Help

The branch is successfully created.

7. To confirm that the branch has been created, right-click the folder icon in the MultiView Explorer or the project in the **Version Control Projects** window and select **TortoiseGit - Show log**.

Version Control Projects	
C:\OMRON\Data\VC\	
Project Folder Search	
···▼ VC (Version control root folder)	
	L
Open in Explorer	
Open Documents Folder	
Open Libraries Folder	
Show Change History for Project	
Resolve Conflicted by Preferring Theirs	
Resolve Conflicted by Preferring Ours	🗹 Pull
	C Fetch
🔅 Git GUI Here	Push
🔅 Git Bash Here	
🐴 Git Sync	Diff with previous version
🦰 Git Commit -> "master"	Show log
😤 TortoiseGit 🕨	🚛 Show Reflog

The "TortoiseGit" Log Messages dialog box is displayed.

laster		From:	8/31/2017	7 🔻 To	8/31/201	7 - ect, M	icoouges, i uuis,	, Authors, Emails, S		thor Email 🔹 🔁
Graph	Actions	Message							Author	Date
		Working t	ee change	s						
•	0 🗗	Program	work n	naster Or	igin/master	Add and M	odify Program	POUs	UserA	8/31/2017 4:48:12
•	0	Add Data	types and	Global varia	bles				UserA	8/31/2017 4:45:37 PM
•	O	Changed	Configurati	on and Set	qu				UserA	8/31/2017 4:44:14 PM
•	4	Register e	mpty proje	ct for Mach	iine A				UserA	8/31/2017 4:41:22 PN
(
Add an	nd Modify			abbale01	f68af8c1					
Add an				ab6a1e01	F68af8c1					
				ab6a1e01	Extension	Status	Lines added	Lines removed		
Path	nd Modify :/48629066-2	Program	POUs	e9bdeb.xm	Extension nl .xml	Modified	Lines added	Lines removed		
Path Project	nd Modify :/48629066-2 :/4d20cbb2-4	Program c64-4aff-bi	POUs)4a-61091a 08d-e731c0	e9bdeb.xm Dfa6cb0.xm	Extension nl .xml nl .xml	Modified Modified	1	1 1		
Path Project Project	nd Modify :/48629066-2 :/4d20cbb2-4 :/69026d5a-4	Program (c64-4aff-b) (44a-4cb6-a (e57-4b88-a	POUs)4a-61091a 08d-e731c(2dd-15cd9	ie9bdeb.xn Jfa6cb0.xm d12b100.xu	Extension 1 .xml 1 .xml 1 .xml 1 .xml	Modified Modified Added	1 1 6	1 1 0		
Path Project Project Project	nd Modify :/48629066-2 :/4d20cbb2-4 :/69026d5a-4 :/7d0f34d8-d	Program (c64-4aff-b) 44a-4cb6-a (e57-4b88-a 328-46be-8	POUs)4a-61091a 08d-e731ct 2dd-15cd9 38b-e9cbba	e9bdeb.xm Jfa6cb0.xm d12b100.xa afa4d49.xm	Extension i .xml .xml i .xml i .xml	Modified Modified Added Modified	1 1 6 2	1 1 0 0		
Path Project Project Project Project	nd Modify (48629066-2 (4420cbb2-4 (69026d5a-4 (860d2f2b-a) (8dd2f2b-a)	Program (c64-4aff-b) 44a-4cb6-a (e57-4b8a-3 328-46be-8 539-4b16-8	POUs)4a-61091a 08d-e731ct 2dd-15cd9 38b-e9cbba ae5-a1bd2	e9bdeb.xm Dfa6cb0.xm d12b100.xa afa4d49.xn 7e5e3cc.xn	Extension i .xml i .xml i .xml i .xml i .xml i .xml	Modified Modified Added Modified Added	1 1 6 2 1	1 1 0		
Path Project Project Project Project	r/48629066-2 /4200b2-4 /69026d5a-4 /780154d8-d /864042f2b-a /86682a4-e	Program (c64-4aff-b) 44a-4cb6-a (e57-4b88- 328-46be-8 339-4b16-8 339-4b16-8 339-451-1	POUs 14a-61091a 08d-e731c0 2dd-15cd9 38b-e9cbba ae5-a1bd2 495-2c8de	ie9bdeb.xn)fa6cb0.xm d 12b 100.xi afa4d49.xn 7e5e3cc.xm 88e8245.xi	Extension 1 .xml 1 .xml 1 .xml 1 .xml 1 .xml 1 .xml 1 .xml	Modified Modified Added Modified Added Modified	1 1 6 2 1 1	1 1 0 0 0 1		
Path Project Project Project Project Project Project	/48629066-2 /4d20cbb2-4 /59026d5a-4 /7d0f34d8-d /aa6a82a4-e /c8a97700-1	Program cc64-4aff-bi 44a-4cb62 e57-4b883 328-46be-2 328-46be-2 328-46be-2 328-46be-2 329-4b16-2 329-4b16-3 329-5	POUs)4a-61091a 08d-e731ct 2dd-15cd9 38b-e9cbba ae5-a1bd2 495-2c8de id3-e2fafe	ie9bdeb.xn Jfa6cb0.xn d12b100.xa afa4d49.xn 7e5e3cc.xn 88e8245.xi 3002e7.oei	Extension i .xml i .xml i .xml i .xml i .xml i .xml ml .xml ml .xml	Modified Modified Added Modified Added Modified Modified	1 1 6 2 1 1 39	1 1 0 0 0 1 8		
Path Project Project Project Project Project Project howing 4 r	/48629066-2 /4d20cbb2-4 /59026d5a-4 /7d0f34d8-d /aa6a82a4-e /c8a97700-1	Program cc64-4aff-bi 44a-4cb62 e57-4b883 328-46be-2 328-46be-2 328-46be-2 328-46be-2 329-4b16-2 329-4b16-3 329-5	POUs)4a-61091a 08d-e731ct 2dd-15cd9 38b-e9cbba ae5-a1bd2 495-2c8de id3-e2fafe	ie9bdeb.xn Jfa6cb0.xn d12b100.xa afa4d49.xn 7e5e3cc.xn 88e8245.xi 3002e7.oei	Extension i .xml i .xml i .xml i .xml i .xml i .xml ml .xml ml .xml	Modified Modified Added Modified Added Modified Modified	1 1 6 2 1 1 39	1 1 0 0 0 1 8) files: modifie	

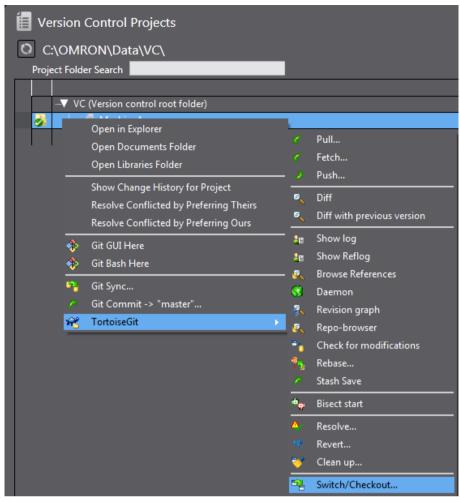
On the top of the upper pane, *ProgramA_work, master*, and *Origin/master* are shown. You can know that the working branch *ProgramA_work* was successfully created.

This line shows that the last revision named as *Add and Modify Program POUs* exists in *ProgramA_work* which is the working branch in the local repository, the main branch *master*, and the *master* in the remote repository (connecting to Origin).

7.3.3.7 Commit changes of Program A and Global Variables to the working branch (No.8, 9: Developer A)

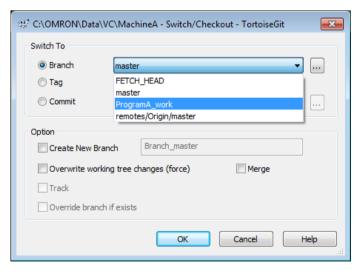
Switch the branch to work on in the Developer A's computer and modify Program A and global variables.

1. Right-click the folder icon in the Multiview Explorer or on the target project in the **Version Control Projects** window, and select **TortoiseGit - Switch/Checkout**.



The TortoiseGit Switch/Checkout dialog box is displayed.

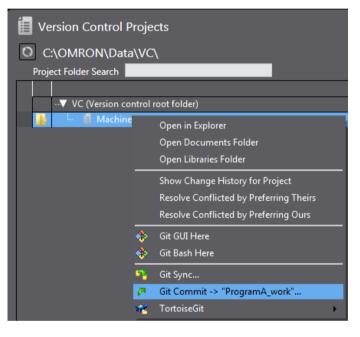
2. Select **Branch** option and *ProgramA_work* which is the working branch of Developer A created in 7.3.3.6 *Create the working branch of Developer A (No.7 Developer A)*, and then click the **OK** button.



The branch is switched.

양* C:\OMRON\Data\VC\MachineA - Git Command Progress - TortoiseGi	it 🗖 🗖 💌
git.exe checkout ProgramA_work	
Switched to branch 'ProgramA_work'	
Success (47 ms @ 8/31/2017 4:57:35 PM)	
Merge ↓▼	Abort

- 3. Click the **Close** button.
- 4. Right-click the folder icon in the Multiview Explorer or on the target project in the **Version Control Projects** window. You can check that the branch name that follows **Git Commit ->** is changed.



5. Develop the ProgramA and modify the settings of the Global Variables. Here, the following edits were made.

Add new Global Variable "P1_Input2".

Global Variables 🗙							
Name	Data Type	Initial Value					
P1_Input	BOOL						
P1_Output	BOOL						
P1_Input2	BOOL						
Parameter1	INT	100					
Parameter2	INT						
P2_Input	BOOL						
P2_Output	BOOL						

Create the following ladder diagram in the ProgramA-Section0.

var Globi	al Variables	🚭 Section0 - ProgramA 🗙	
Varia	bles		
0	P1_in P1_in		P1_Output

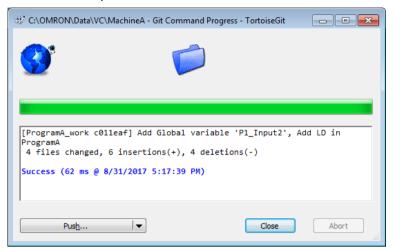
- 6. From the Main menu, select File Save to save the project.
- 7. Select File Close to close the project.
- 8. Right-click the target project in the Version Control Projects window and select Git Commit -> "ProgramA_work" from the pop-up menu.

📕 Version Control Projects								
C:\OMRON\Data\VC\								
Project Folder Search								
···▼ VC (Version control root folder)								
	Open in Explorer							
	Open Documents Folder							
Open Libraries Folder								
Show Change History for Project								
Resolve Conflicted by Preferring Theirs								
	Resolve Conflicted by Preferring Ours							
*	Git GUI Here							
*	Git Bash Here							
-	Git Sync							
P	Git Commit -> "ProgramA_work"							

The "TortoiseGit" Commit dialog box is displayed.

$\otimes_{\mathbb{R}}^{\mathbb{R}^{2}} C: \verb OMRON\Data\VC\MachineA - Commit - TortoiseGit $				- • ×					
Commit to: ProgramA_work new	branch								
Add Global variable <u>'Pl_Input2'</u> , Add LD in	ProgramA								
Amend Last Commit 1/52									
Changes made (double-click on file for diff): Check: All None Unversioned Versioned Added Deleted Modified Files Submodules									
Path	Extension	Status	Lines added	Lines remc					
Project/3da8fac8-47c1-4490-9a9f-69a4a6d2e628.xml	.xml	Modified	1						
V Project/4d20cbb2-444a-4cb6-a08d-e731c0fa6cb0.xml	.xml	Modified	1						
Project/7d0f34d8-d328-46be-838b-e9cbbafa4d49.xml	.xml	Modified	1						
III				•					
Show Unversioned Files	4 files sele	4 files selected, 4 files total							
Do not autoselect submodules				/iew Patch>>					
Show Whole Project									
Message only	Cor	nmit 🔷 🔻	Cancel	Help					

9. Enter comment in the **Message** area and click the **Commit** button. Commit is completed.

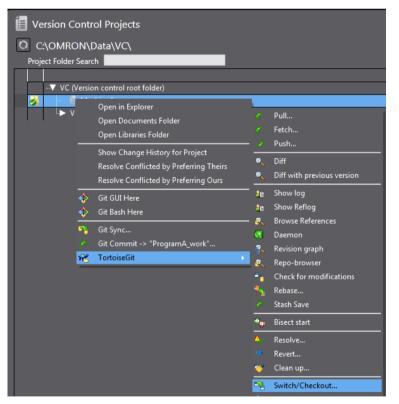


10. Click the **Close** button.

7.3.3.8 Merges the changes of working branch to the main branch (No.10: Developer A)

Switch the branch you work on to the main branch and then merge the changes in the working branch to the main branch.

1. Right-click the target project in the Version Control Projects window and select TortoiseGit - Switch/Checkout from the pop-up menu.



The "TortoiseGit" Switch/Checkout dialog box is displayed.

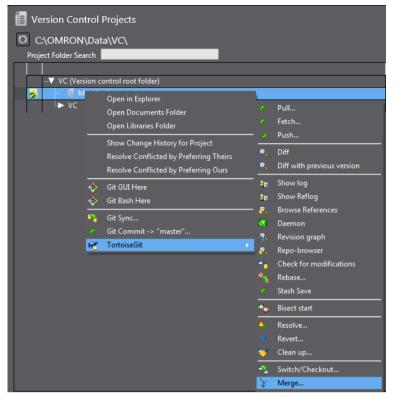
2. Select **Branch** option, then *master* (main branch), and click the **OK** button.

🕸 C:\OMRON\Data	S# ² C:\OMRON\Data\VC\MachineA - Switch/Checkout - TortoiseGit							
Switch To								
e Branch	ProgramA_work 🔹							
🔘 Tag	FETCH_HEAD							
Commit	master ProgramA_work remotes/Origin/master							
Option								
Create New B	ranch Branch_ProgramA_work							
Overwrite wor	king tree changes (force)							
Track								
Override bran	ch if exists							
	OK Cancel Help							

The branch is switched.

C:\OMRON\Data\VC\Ma	chineA - Git Command Pro	gress - TortoiseGit	
9			
git.exe checkout mas	ton		
Switched to branch '			
Success (93 ms @ 8/3			
Merge	▼	Close	Abort

- 3. Click the **Close** button.
- 4. To merge the changes to the main branch, right-click the target project in the **Version Control Projects** window and select **TortoiseGit Merge** from the pop-up menu.



The "TortoiseGit" Merge dialog box is displayed.

5. Select **Branch** option, then the working branch *ProgramA_work* to merge to the main branch *master*, and then click the **OK** button.

🕸 C:\OMRON\Data\	VC\MachineA - Mer	ge - TortoiseGit		×
From				
Ø Branch	ProgramA_work			
🔘 Tag	FETCH_HEAD			
Commit	ProgramA_work remotes/Origin/mas	ter		
Option				
Squash		Messages	20	
🔲 No Fast Forwa	ard	E Fast Forward C	Dnly	
No Commit				
Strategy	•		·	
Merge Message				
<auto genera<="" th=""><td>ted by Git></td><td></td><td></td><td></td></auto>	ted by Git>			
		ОК	Cancel H	lelp

This completes Merge processing.

👾 C:\OMRON\Data\VC\MachineA - Git Command Progress - TortoiseGit
S 📁
<pre>git.exe merge ProgramA_work Updating 88dcbb5c011eaf Fast-forward Project/3da8fac8-47c1-4490-9a9f-69a4a6d2e628.xml 1 + Project/4d20cbb2-444a-4cb6-a08d-e731c0fa6cb0.xml 2 +- Project/7d0f34d8-d328-46be-838b-e9cbbafa4d49.xml 1 + Project/c8a97700-1232-4fc9-bdd3-e2fafe3002e7.oem 6 +++ 4 files changed, 6 insertions(+), 4 deletions(-) Success (94 ms @ 8/31/2017 5:21:27 PM)</pre>
Close Abort ai

- 6. Click the **Close** button.
- 7. Right-click the folder icon in the MultiView Explorer or the project in the **Version Control Projects** window and select **TortoiseGit Show log.**

The "TortoiseGit" Log Messages dialog box is displayed.

ter		From:	8/31/201	.7 🔻	To:	8/31/2017	→ gct _j	Messages, Paths	, Authors, Emails,	SHA-1, R Author Email 🔻	·] 🔶
aph	Actions	Message							Author	Date	
		Working t	ree change	es							
	ø	Program	A_work	master	Add Gl	obal varia	able 'P1_I	nput2', Add L	D in UserA	8/31/2017 5:17:39	
	🚯 🖡	Origin/ma	ster Add a	and Modi	ify Progra	m POUs			UserA	8/31/2017 4:48:12 PM	
	O	Add Data	types and	Global v	/ariables				UserA	8/31/2017 4:45:37 PM	
	o a	-	Configurat						UserA	8/31/2017 4:44:14 PM	
	40	Register (empty proj	ect for №	lachine A				UserA	8/31/2017 4:41:22 PM	
dd G]	lobal vari	iable 'P	1_Input:	2', Ad	ld LD ir	n Progra	mA				
dd GJ	lobal vari	iable 'P	1_Input;	2', Ad	ld LD ir	n Progra	mA				
idd G3	lobal vari	iable 'P	1_Input	2', Ad			mA Status	Lines added	Lines removed		
h Project	:/3da8fac8-4:	7c1-4490-9	a9f-69a4a	6d2e628	Ex 3.xml .xr	tension	Status Modified	1	Lines removed		
h Project	:/3da8fac8-4: :/4d20cbb2-4	7c1-4490-9 44a-4cb6-a	a9f-69a4a 08d-e7310	6d2e628	Ex 3.xml .xr 0.xml .xr	tension ml ml	Status Modified Modified	1	0		
th Project Project	:/3da8fac8-4: :/4d20cbb2-4 :/7d0f34d8-d	7c1-4490-9 44a-4cb6-z 328-46be-8	a9f-69a4a 08d-e7310 138b-e9cbb	6d2e628 :0fa6cb0 pafa4d49	3.xml .xr 3.xml .xr 9.xml .xr	tension ml ml ml	Status Modified Modified Modified	1 1 1	0 1 0		
th Project Project	:/3da8fac8-4: :/4d20cbb2-4	7c1-4490-9 44a-4cb6-z 328-46be-8	a9f-69a4a 08d-e7310 138b-e9cbb	6d2e628 :0fa6cb0 pafa4d49	3.xml .xr 3.xml .xr 9.xml .xr	tension ml ml ml	Status Modified Modified	1	0		
h Project Project Project	//da8fac8-4; //d20cbb2-4 //d0f34d8-d //c8a97700-1 revision(s), fr	7c1-4490-9 44a-4cb6-a 328-46be-8 232-4fc9-b	a9f-69a4a 108d-e731c 138b-e9cbb dd3-e2fafe	6d2e628 20fa6cb0 2afa4d49 23002e7	Ex 3.xml .xr 3.xml .xr 9.xml .xr 7.0em .06	itension ml ml ml em	Status Modified Modified Modified	1 1 3	0 1 3) files: modified = 0 added = 0	delete
h Project Project Project	/3da8fac8-4 /4d20cbb2-4 /7d0f34d8-d /c8a97700-1 revision(s), fr	7c1-4490-9 44a-4cb6-a 328-46be-8 232-4fc9-b	a9f-69a4a 108d-e731c 138b-e9cbb dd3-e2fafe	6d2e628 20fa6cb0 2afa4d49 23002e7	Ex 3.xml .xr 3.xml .xr 9.xml .xr 7.0em .06	itension ml ml ml em	Status Modified Modified Modified	1 1 3	0 1 3	-) files: modified = 0 added = 0	delete

On the top of the upper pane, *ProgramA_work* and *master* are shown, and in the middle pane *Add Global variable 'P1_Input2* and *Add LD in ProgramA* are indicated. You can know that it is the latest revision of the

working branch *ProgramA_work*, and the main branch *master* is the same as the working branch *ProgramA_work*.

7.3.3.9 Create the working branch of Developer B (No.7: Developer B)

After registering the local repository in the Developer B's computer, there is a possibility that other developers are updating the target project in the remote repository. In order to synchronize the local repository in the Developer B's computer with the remote repository which functions as the master, take the following steps.

- 1. On the Start page of the Sysmac Studio, select **Version Control Explorer** and open the **Version Control Projects** window.
- 2. Right-click the *MachineA* and select **TortoiseGit Pull** from the pop-up menu. The "TortoiseGit" **Pull** dialog box is displayed.
- 3. Check that **Origin** is selected in **Remote** (the path to the remote repository is set as <u>\\SLRA0383\Git\MachineA.git</u> in this explanation) and click the **OK** button.

Remote	MachineA - Pull - TortoiseGit 📃
<u> <u> </u> </u>	Origin
O Arbitrary URL:	
Remote <u>B</u> ranch:	master 👻
Options	
Squash	No Commit
No <u>F</u> ast Forward	Fast Forward Only
Tags	Default: Reachable
Prune	
AutoLoad Putty Key	Manage Remotes
🔲 Launch Rebase After Fe	etch
	OK Cancel Help

- 4. Click the Close button in the Complete Pull dialog box.
- Right-click the folder icon in the Multiview Explorer or on the target project in the Version Control Projects window, and select TortoiseGit - Create Branch. The "TortoiseGit" Create Branch dialog box is displayed.
- 6. Enter the branch name (in this example, *ProgramB_work*), and click the **OK** button. For **Base On**, select the latest revision. Select **Head (master)**.

💒 C:\OMRON\Data\	VC\VC\MachineA - Create Branch - TortoiseGit 🧮	x
Name Branch	ProgramB_work	
Base On		
Ø Branch	master 🔻	
Tag		
Commit	· · · · · · · · · · · · · · · · · · ·	
Options		
Track	Force Switch to new branch	
Description		
	, , ,	
	OK Cancel Help	

The branch is successfully created.

7. Right-click the folder icon in the MultiView Explorer or the project in the Version Control Projects window and select TortoiseGit - Show log.

aster		From:	8/31/2017	▼ To:	8/31/201	7 🔻 🔭, Au	uthor Author Em	ail 🔻 👍 📢
Graph	Actions	Message			Author	Date		
		Working t	ree changes					
•	0 🗗	Program	B_work m	aster origi	n/ma UserA	8/31/	2017 4:48:12)
•	ü	Add Data	types and G	lobal variabl	es UserA	8/31/2	017 4:45:37 PM	
•	O	Changed	Configuration	n and Setup	UserA	8/31/2	017 4:44:14 PM	
•	4	Register (empty projec	t for Machin	e A UserA	8/31/2	017 4:41:22 PM	
Add ar	d Modify	Program	POUs					
Add ar	nd Modify	Program	POUs					
	nd Modify	Program	POUs		Extension	Status	Lines added	Lines removed 4
Path	nd Modify			9bdeb.xml	Extension .xml	Status Modified	Lines added	Lines removed
Path Project		c64-4aff-b	04a-61091ae					
Path Project	;/48629066-2	- 1c64-4aff-b 44a-4cb6-a	04a-61091ae 08d-e731c01	fa6cb0.xml	.xml .xml	Modified	1	1
Path Project	:/48629066-2 :/4d20cbb2-4	2 2c64-4aff-b 44a-4cb6-a ie57-4b88-a	04a-61091ae 08d-e731c0 a2dd-15cd9d	fa6cb0.xml 12b100.xml	.xml .xml .xml	Modified Modified	1	1
Path Project	:/48629066-2 :/4d20cbb2-4 :/69026d5a-4	- 1c64-4aff-b 44a-4cb6-a ie57-4b88-a 328-46be-8	04a-61091ae 08d-e731c01 a2dd-15cd9d 38b-e9cbba1	fa6cb0.xml 12b100.xml fa4d49.xml	.xml .xml .xml	Modified Modified Added	1 1 6	1 1 0
Path Project Project Project Project	:/48629066-2 :/4d20cbb2-4 :/69026d5a-4 :/7d0f34d8-d:	c64-4aff-b 44a-4cb6-a e57-4b88-i 328-46be-8 b39-4b16-8	04a-61091ae 08d-e731c01 a2dd-15cd9d 138b-e9cbba1 ae5-a1bd270	fa6cb0.xml 12b100.xml fa4d49.xml e5e3cc.xml	.xml .xml .xml .xml .xml	Modified Modified Added Modified	1 1 6 2	1 1 0 0
Path Project Project Project Project	:/48629066-2 :/4d20cbb2-4 :/69026d5a-4 :/7d0f34d8-d: :/8ddd2f2b-al	c64-4aff-b 44a-4cb6-a e57-4b88-i 328-46be-8 b39-4b16-8	04a-61091ae 08d-e731c01 a2dd-15cd9d 138b-e9cbba1 ae5-a1bd270	fa6cb0.xml 12b100.xml fa4d49.xml e5e3cc.xml	.xml .xml .xml .xml .xml .xml	Modified Modified Added Modified Added	1 1 6 2 1	1 1 0 0
Path Project Project Project Project	:/48629066-2 :/4d20cbb2-4 ;/69026d5a-4 ;/7d0f34d8-d :/8ddd2f2b-al :/aa6a82a4-e	cc64-4aff-b 44a-4cb6-a e57-4b88-d 328-46be-8 338-46be-8 339-4b16-8 339-45e1-b	04a-61091ae 08d-e731c0t a2dd-15cd9d 38b-e9cbbat 38b-e9cbbat 3e5-a1bd27 o495-2c8de8	Fa6cb0.xml 12b100.xml Fa4d49.xml e5e3cc.xml 8e8245.xml	.xml .xml .xml .xml .xml .xml	Modified Modified Added Modified Added Modified	1 1 6 2 1 1	1 1 0 0
Path Project Project Project Project Project	:/48629066-2 :/4d20cbb2-4 ;/69026d5a-4 ;/7d0f34d8-d :/8ddd2f2b-al :/aa6a82a4-e	cc64-4aff-b 44a-4cb6-a e57-4b88-d 328-46be-8 338-46be-8 339-4b16-8 339-45e1-b	04a-61091ae 08d-e731c0t a2dd-15cd9d 38b-e9cbbat 38b-e9cbbat 3e5-a1bd27 o495-2c8de8	Fa6cb0.xml 12b100.xml Fa4d49.xml e5e3cc.xml 8e8245.xml	.xml .xml .xml .xml .xml t ddbb5 - 1 revis	Modified Modified Added Modified Added Modified	1 1 6 2 1 1	1 1 1 0 0 0 0 0 1
Path Project Project Project Project Project	:/48629066-2 :/4d20cbb2-4 :/69026d5a-4 :/7d0f34d8-d. :/8ddd2f2b-al :/aa6a82a4-e revision(s), fr nole Project	cc64-4aff-b 44a-4cb6-a e57-4b88-d 328-46be-8 338-46be-8 339-4b16-8 339-45e1-b	04a-61091ae 08d-e731c0t a2dd-15cd9d 38b-e9cbbat 38b-e9cbbat 3e5-a1bd27 o495-2c8de8	Fa6cb0.xml 12b100.xml Fa4d49.xml e5e3cc.xml 8e8245.xml	.xml .xml .xml .xml .xml t ddbb5 - 1 revis	Modified Modified Added Modified Added Modified	1 1 6 2 1 1	

The "TortoiseGit" Log Messages dialog box is displayed.

On the top of the upper pane, *ProgramB_work, master, origin/master* and *origin/HEAD* are shown. You can know that the working branch *ProgramB_work* was successfully created.

This line shows that the changes named as *Add and Modify Program POUs* applied in the latest revision by Developer A exists in ProgramB_work which is the working branch of the Developer B's local repository, the main branch *master*, and the *master* in the remote repository (connecting to Origin).

7.3.3.10 Commits changes of Program B and Global Variables to the branch (No.8, 9: Developer B)

Switch the branch to work on in the Developer B's computer and modify Program B and Global Variables.

- Right-click the folder icon in the Multiview Explorer or on the target project in the Version Control Projects window, and select TortoiseGit - Switch/Checkout. The "TortoiseGit" Switch/Checkout dialog box is displayed.
- 2. Select **Branch** option and *ProgramB_work* which is the working branch of Developer B created in 7.3.3.9 *Create the working branch of Developer B (No.7: Developer B)*, and then click the **OK** button.

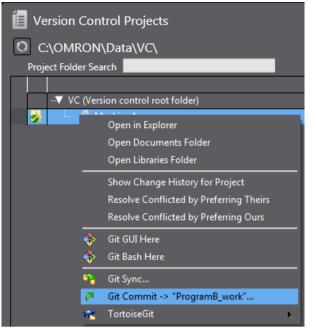
C:\OMRON\Data\V	C\MachineA - Switch/Checkout - TortoiseGit	×
Switch To		
Branch	master	
🔘 Tag	FETCH_HEAD	
Commit	master ProgramB_work	
	remotes/origin/HEAD	
Option	remotes/origin/master	
Create New Bran	ch Branch_master	
🔲 Overwrite workin	g tree changes (force)	
Track		
Override branch	if exists	
	OK Cancel	Help

The branch is switched.

ి: సార్టి C:\OMRON\Data\VC\MachineA - Git Command Progres	ss - TortoiseGit 📃 🗖 💌
S	
git.exe checkout ProgramB_work	
Switched to branch 'ProgramB_work'	
Success (78 ms @ 9/1/2017 9:49:45 AM)	
	Close Abort

- 3. Click the **Close** button.
- 4. Right-click the folder icon in the Multiview Explorer or on the target project in the **Version Control Projects** window.

You can check that the branch name that follows **Git Commit ->** is changed.



5. Develop the ProgramB and change the Global Variables. Here, the following edits are made.

Change the initial value of Global Variable Parameter1 to 200.

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

Create the following ladder diagram in the ProgramB-Section0.

0	P2_Input		MOVE	P2_Output
		EN	ENO	O
	Parameter1-	In	Out	- Parameter2

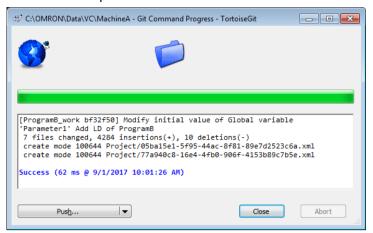
- 6. From the Main menu, select File Save to save the project.
- 7. Select File Close to close the project.
- 8. Right-click the target project in the **Version Control Projects** window and select **Git Commit ->** "**ProgramB_work**" from the pop-up menu.

Version (Control Projects						
C:\OMR	ON\Data\VC\						
Project Folde	r Search						
V VC							
j 🚺 🛄 🗐							
	Open in Explorer						
	Version Control Projects C:\OMRON\Data\VC\ Project Folder Search VC (Version control root folder) VC (Version control root folder) VC (Version control root folder) MachineA Open in Explorer Open Documents Folder Open Libraries Folder Show Change History for Project Resolve Conflicted by Preferring Theirs Resolve Conflicted by Preferring Ours Git GUI Here Git Bash Here Git Sync Git Commit -> "ProgramB_work"						
	MachineA Open in Explorer Open Documents Folder Open Libraries Folder Show Change History for Project Resolve Conflicted by Preferring Theirs Resolve Conflicted by Preferring Ours Git GUI Here						
	MachineA Open in Explorer Open Documents Folder Open Libraries Folder Show Change History for Project Resolve Conflicted by Preferring Theirs Resolve Conflicted by Preferring Ours Git GUI Here						
	Resolve Conflicted by Preferring Theirs						
	Project Folder Search						
	Project Folder Search Image: VC (Version control root folder) Image: MachineA Open in Explorer Open Documents Folder Open Libraries Folder Show Change History for Project Resolve Conflicted by Preferring Theirs Resolve Conflicted by Preferring Ours Image: Ima						
	🚸 Git Bash Here						
	😚 Git Sync						
C:\OMRON\Data\VC\ Project Folder Search VC (Version control root folder) VC (Version control root folder) MachineA Open in Explorer Open Documents Folder Open Libraries Folder Show Change History for Project Resolve Conflicted by Preferring Theirs Resolve Conflicted by Preferring Ours Git GUI Here Git Bash Here Git Sync Git Commit -> "ProgramB_work"							
	📸 TortoiseGit 🔸						

The "TortoiseGit" **Commit** dialog box is displayed.

ೆಗೆ C:\OMRON\Data\VC\MachineA - Commit - TortoiseGit	
Commit to: ProgramB_work new branch Message:	
Modify initial value of Global variable <u>'Paramete</u> ProgramB	nl'Add LD of ⊋
Amend Last Commit	1/72
Set author	Add Signed-off-by
Changes made (double-click on file for diff): Check: All None Unversioned Versioned Added Deleted	Modified Files Submodule
Path Exter	nsion Status Line
Project/05ba15e1-5f95-44ac-8f81-89e7d2523c6a.xml .xml	Added
Project/2167f849-e4ea-49d3-9e5a-47e7e29ae9d1.oem .oem	Modified
Project/2h0c86ee-h9hh-43e7-af8c-7a8a3hc22dh0 vml vml III	Modified
Show Unversioned Files	7 files selected, 7 files total
Do not autoselect submodules	View Patch>>
	view Patch>>
Show Whole Project	Cancel Help

9. Enter comment in the **Message** area and click the **Commit** button. Commit is completed.



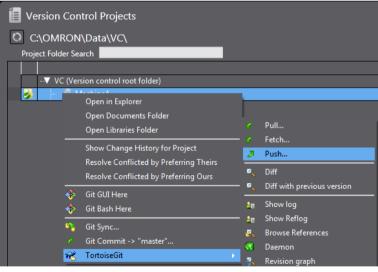
10. Click the Close button.

7.3.3.11 Push the changes in the local repository of Developer A to the remote repository (No.11: Developer A)

In order to merge the changes made by Developer A and Developer B, first push the changes in the Developer A's local repository to the remote repository, and then apply the changes in the remote repository to the Developer B's local repository. Then, merge the changes of Developer B.

This section explains how to push changes in Developer A's local repository to the remote repository.

1. Right-click the target project in the **Version Control Projects** window and select **TortoiseGit - Push** from the pop-up menu.



The "TortoiseGit" **Push** dialog box is displayed.

2. Make sure that the **Push all branches** check box is selected and **Remote** item under **Destination** is correctly set ("Origin" is set in this example), and then click the **OK** button.

👾 C:\OMRON\Data\VC\MachineA - Push - TortoiseGit
Ref Ø Push all branches
Local: master
Remote:
Destination
O Arbitrary URL:
Options
Force: May discard known changes unknown changes
Indude Tags
📝 Autoload Putty Key
Set upstream/track remote branch
Always push to the selected remote archive for this local branch
Always push to the selected remote branch for this local branch
Recurse submodule None
OK Cancel Help

This completes Push processing.

🚓 C:\OMRON\Data\VC\MachineA - Git Command Progress - TortoiseGit		- • ×
Writing objects		
git.exe pushallprogress "Origin"		
Counting objects: 7, done. Delta compression using up to 4 threads. Compressing objects: 100% (7/7), done. Writing objects: 100% (7/7), 835 bytes 835.00 KiB/s, done. Total 7 (delta 5), reused 0 (delta 0) To C:/Git/MachineA.git 2baf9dcf502463 master -> master * [new branch] ProgramA_work -> ProgramA_work Success (187 ms @ 9/1/2017 10:06:12 AM)		
Create pull request	Close	Abort
	Ciose	Abort

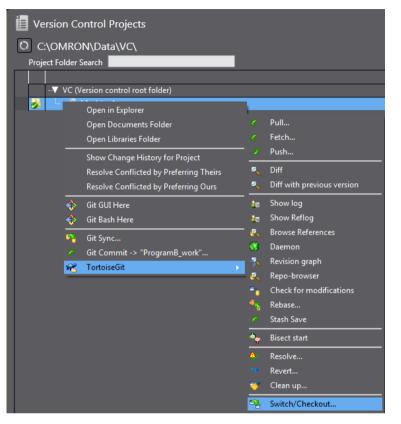
3. Click the **Close** button.

7.3.3.12 Merges the changes in the working branch of Developer B to the main branch (No.12: Developer B)

Switch the branch you work on to the main branch and then acquire the changes made by Developer A from the remote repository. After that, switch to the working branch of Developer B and merge the changes made by Developer A in the main branch to the working branch of Developer B. Since the changes made by Developer A conflict with the changes of Developer B, merge processing fails. In this case, resolve the conflicts and complete merging.

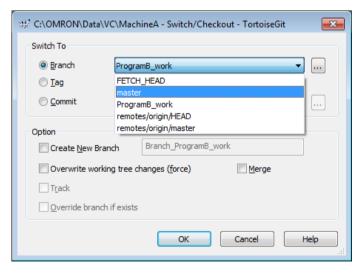
After merging, switch the branch you work on to the main branch and then merge the data after resolving conflicts to the main branch. This completes the merge processing.

1. Right-click the target project in the **Version Control Projects** window and select **TortoiseGit - Switch/Checkout** from the pop-up menu.



The "TortoiseGit" Switch/Checkout dialog box is displayed.

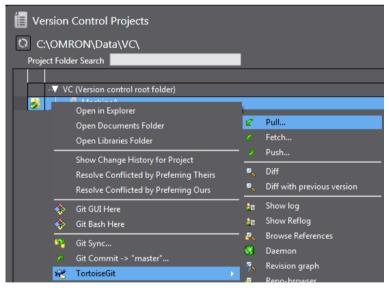
2. Select **Branch** option, then *master* (main branch), and click the **OK** button.



The branch is switched.

🔆 C:\OMRON\Data\VC\MachineA - Git Command Progress - TortoiseGit		- • •
S 📁		
git.exe checkout master		
Your branch is up-to-date with 'Origin/master'. Switched to branch 'master'		
Success (93 ms @ 9/1/2017 10:12:27 AM)		
Merge ▼	Close	Abort

- 3. Click the **Close** button.
- 4. Since the remote repository includes the changes made by Developer A, take the following steps to synchronize the remote repository and the local repository of the Developer B's computer. On the Start page of the Sysmac Studio, select Version Control Explorer and open the Version Control Projects window.
- 5. Right-click the *MachineA* and select **TortoiseGit Pull** from the pop-up menu.



The "TortoiseGit" Pull dialog box is displayed.

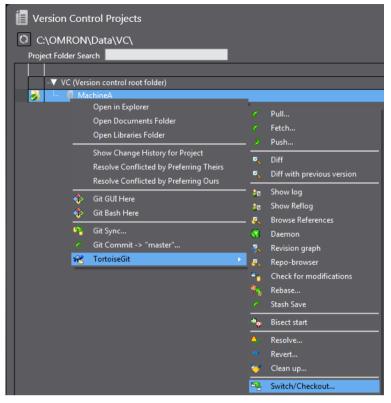
6. Check that *Origin* is selected in **Remote** (the path to the remote repository is set as \\SLRA0383\Git\MachineA.git in this explanation) and click the **OK** button.

综 C:\OMRON\Data\VC\Machi	neA - Pull - TortoiseGit 🧾
Remote	
Remote:	Origin
O Arbitrary URL:	
Remote Branch:	master 💌 📖
Options	
Squash	No Commit
No Fast Forward	Fast Forward Only
Tags	Default: Reachable
Prune	
V AutoLoad Putty Key	Manage Remotes
Launch Rebase After Fetc	h
	OK Cancel Help

This completes Pull processing.

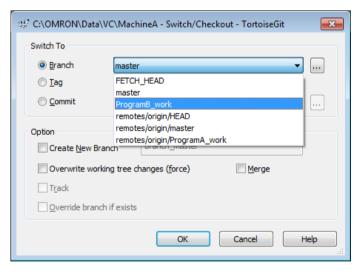
، C:\OMRON\Data\VC\MachineA - Git Command Progress - TortoiseGit	
remote: Compressing objects	
<pre>* [new branch] master -> Origin/master * [new branch] ProgramA_work -> Origin/ProgramA_work Updating 2baf9dc.,f502463 Fast-forward</pre>	*
Project/2167f849-e4ea-49d3-9e5a-47e7e29ae9d1.oem 6 +++	
Project/3da8fac8-47c1-4490-9a9f-69a4a6d2e628.xml 1 + Project/4d20cbb2-444a-4cb6-a08d-e731c0fa6cb0.xml 2 +-	
Project/7d0f34d8-d328-46be-838b-e9cbbafa4d49.xm1 1 +	E
4 files changed, 6 insertions(+), 4 deletions(-)	
Success (234 ms @ 9/1/2017 10:16:27 AM)	-
·	
Pulled Diff Close	Abort

- 7. Click the **Close** button.
- 8. Next, switch to the *ProgramB_work* in order to merge the changes made by Developer A in the main branch *master* to the working branch of Developer B *ProgramB_work*. Right-click the target project in the **Version Control Projects** window and select **TortoiseGit Switch/Checkout** from the pop-up menu.



The "TortoiseGit" Switch/Checkout dialog box is displayed.

9. Select **Branch** option and *ProgramB_work* which is the working branch of Developer B, and then click the **OK** button.



The branch is switched.

$\mathfrak{S}^{\mathfrak{p}^2}_{\mathfrak{P}}$ C:¥OMRON¥Data¥VC¥MachineA - Git Comma	nd Progress - To 🗖 🗖 💌
🔮 📁	
git.exe checkout ProgramB_work	
Switched to branch 'ProgramB_work'	
Success (218 ms @ 2017/09/08 16:05:45)	
1	
Merge	Close Abort

- 10. Click the **Close** button.
- 11. Right-click the target project in the **Version Control Projects** window and select **TortoiseGit Merge** from the pop-up menu. The "TortoiseGit" **Merge** dialog box is displayed.
- 12. Select **Branch** option, then the main branch *master* to merge it to the working branch *ProgramB_work*, and click the **OK** button.

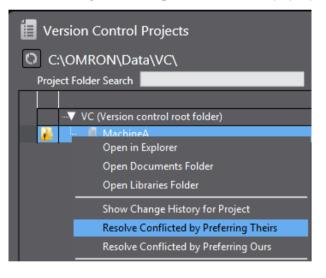
🕸 C:\OMRON\Da	ta\VC\MachineA - Merge - TortoiseGit	×
From		
● Branch○ Tag○ Commit	FETCH_HEAD master remotes/origin/HEAD remotes/origin/master	
Option Squash	remotes/origin/ProgramA_work	
No Fast For		
Strategy (• • • • • • • • • • • • • • • • • • •	
Merge Message	rated by Git>	
	OK Cancel Hel	lp

Merge fails.

ap C:\OMRON\Data\VC\MachineA - Git Command Progress - TortoiseGit	
git.exe merge master	
Auto-merging Project/3da8fac8-47c1-4490-9a9f-69a4a6d2e628.xml CONFLICT (content): Merge conflict in Project/3da8fac8-47c1-4490-9a9f- 69a4a6d2e8.xml Auto-merging Project/2167f849-e4ea-49d3-9e5a-47e7e29ae9d1.eem CONFLICT (content): Merge conflict in Project/2167f849-e4ea-49d3-9e5a- 47e7e29ae9d1.eem Automatic merge failed; fix conflicts and then commit the result. git did not exit cleanly (exit code 1) (78 ms @ 9/1/2017 10:26:12 AM)	
Resolve I	Abort

This is because Developer A and Developer B made changes in the same global variables and a conflict occurred.

- 13. Click the **Close** button.
- 14. Right-click the target project in the Version Control Projects window and select TortoiseGit Resolve Conflicted by Preferring Theirs from the pop-up menu.



The menu **Resolve Conflicted by Preferring Theirs** forcibly merges the data with a conflict caused by changes made in the same files. This command prioritizes the changes in merge target data specified in the **From** section in the "TortoiseGit" **Merge** dialog box, assuming that changes in the merge source data are invalid. In this example, changes in the *master* are given priority in merge processing.

In this explanation, a conflict occurred in the Global Variables. If you select **Resolve Conflicted by Preferring Theirs** menu, merge result is shown on the **Sysmac Diff** dialog box.

et:bf32f50	(HEAD -> ProgramB_work)	Target: Update Date and	Source: Update Date and So	urce:Working project	Detailed Comp
C)ata Trace Settings	8/31/2017 12:29:40 PM	8/31/2017 12:29:40 PM	Data Trace Settings	
	Comparison ID for Configuration	9/1/2017 9:58:06 AM	9/1/2017 9:23:08 AM	Comparison ID for Configuration	
3	D equipment model	8/31/2017 12:29:40 PM	8/31/2017 12:29:40 PM	3D equipment model	
		8/31/2017 12:29:40 PM	8/31/2017 12:29:40 PM		
🛛 🗸 🗸 Prog	gramming			Programming	
- V P				-▼ POUs	
-7	Programs			Programs	
		9/1/2017 9:23:08 AM	9/1/2017 9:23:08 AM		
	Variables	9/1/2017 9:23:08 AM	9/1/2017 9:36:11 AM	Variables	
	Section0	9/1/2017 9:23:08 AM	9/1/2017 9:36:11 AM	Section0	
		9/1/2017 9:23:08 AM	9/1/2017 9:23:08 AM	ProgramB	
		9/1/2017 9:58:06 AM	9/1/2017 9:58:06 AM		
	Section0	9/1/2017 9:58:06 AM	9/1/2017 9:58:06 AM	Section0	
	Function Blocks			- Function Blocks	
L V D)ata				
-	Data Types	8/31/2017 12:29:40 PM	8/31/2017 12:29:40 PM	Data Types	
	Global Variables	9/1/2017 9:58:06 AM	9/1/2017 9:36:11 AM	Global Variables	
				Select	this to overwrite with left Cano

The purple bar is shown on the left end for the data in which a conflict occurs. The red bar is for the data which were merged without conflict though there were differences.

15. Click the ... button of the Global Variables with conflicts.

			-																
Sysm	ac Diff																		X
														€	e,	¹⁰ 0	1	t	F
	Target: new_C	Controller_0) (Global Varia	ables)					:	Source: new_0	Controller_) (Global Variat	oles)						
			Initial Value			Network Publish	Comment			Name	Data Type	Initial Value			ant I	Vetwork	Publish	Cor	nm
0	P1_Input	BOOL		FALSE	FALSE	Do not publish		0		P1_Input	BOOL		FALSE				publish		
								1		P1_Input2	BOOL		FALSE				publish		
2	P1_Output	BOOL			FALSE	Do not publish		2		P1_Output	BOOL		FALSE				publish		
	P2_Input	BOOL			FALSE	Do not publish		3		P2_Input	BOOL		FALSE				publish		
	P2_Output	BOOL	200		FALSE	Do not publish		4		P2_Output	BOOL	100	FALSE				publish		
1 5	Parameter1 Parameter2	INT	200		FALSE	Do not publish Do not publish		5 6	1	Parameter1 Parameter2		100	FALSE FALSE				publish publish		
	Parameterz	101		TALSE	TAESE	DO HOT PUBLISH		0		Parameterz	TIME		TALSE	TALS	-	Donot	Publish		
						v→Cor	y Selected i	from Let	ft to	Right 🛟 Co	opy All Diffe	erence from Lef	t to Righ	t 🔊) Und	lo Selec	ted 🚓	Und	o /
egend:	Different	Left side or	nly Right s	ide only	Copied									Back	Rec	ompare	e Appl	y (llo

The **Detailed Comparison** dialog box is displayed.

The data of the working branch *ProgramB_work* (changes of Developer B) before merging is displayed on the left pane and the right pane shows the data being edited to which data was merged preferring the main branch (changes of Developer A).

16. Here, you need to leave the value of the variable *Parameter1* on the fourth line. For that, select the fourth line.

📓 Sys	smac Diff																	• ×
1													D	Ð	Q	100	↑ ↓	Ð
	Target: ne	w Controller	_0 (Global Varia	hler)			_	_		Source: new_	Controller	0 (Global Var	iabler)					
	Name				Constant	Network Publish	Commont					Initial Value		n ICor	octanti	Notwork	Dublicht	Commont
0	P1 Input				FALSE	Do not publish		0	1	P1_Input	BOOL			E FA		Do not		comment
, v	T_mput	DOOL		TALSE	TALUE	Do not publish		1	11	P1_Input2	BOOL	_		E FA		Do not		
1	P1_Outp	ut BOOL		FALSE	FALSE	Do not publish		2	11	P1 Output	BOOL			E FA		Do not		
2	P2_Input				FALSE	Do not publish		3	1.	P2_Input	BOOL			E FA		Do not		
3	P2 Outp				FALSE	Do not publish		4		P2 Output	BOOL			E FA		Do not		
4	Paramet		200		FALSE	Do not publish		5			INT	100		E FA		Do not		
5	Paramet				FALSE	Do not publish		6		Parameter2				E FA		Do not		
				_	_		_	_	_		_		_	_	_	_	_	
				_	_									_				- 14
						>+ Cop	by Selected	from Le	eft to	Right 🛟 Co	py All Diff	erence from l	eft to Rig	ht			ted 🚓 L	Indo All
Leger	nd: Different	Left side	only Right si		Copied									Bac	k Re	compare	Apply	Close
L						1												

17. Click the **Copy Selected from Left to Right** button. The value of variable *Parameter1* is set to the data being edited on the right pane.

	Sysma	c Diff																	• ×
] (a, Q,	102	↑ ↓	- E
		Target: new (Controller	0 (Global Varial	oles)						Source: new_(Controller	0 (Global Vari	ables)					i
						Constant	Network Publish	Comment					Initial Value		in IC	Constant	Network	Publish	Comment
	0	P1_Input	BOOL		FALSE	FALSE	Do not publish		0	Г	P1_Input	BOOL		FAL	SE	FALSE	Do not		
									1	iT.	P1_Input2	BOOL		FAL	SE I	FALSE	Do not		
	1	P1 Output	BOOL		FALSE	FALSE	Do not publish		2	11	P1_Output	BOOL		FAL	SE I	FALSE	Do not	publish	
	2	P2_Input	BOOL		FALSE	FALSE	Do not publish		3		P2_Input	BOOL		FAL	SE	FALSE	Do not		
	3	P2_Output	BOOL		FALSE	FALSE	Do not publish		4	1	P2_Output	BOOL		FAL	SE	FALSE	Do not		
	4	Parameter1	INT	200	FALSE	FALSE	Do not publish		5	Т	Parameter1	INT	200	FAL	SE I	FALSE	Do not	publish	
	5	Parameter2	INT		FALSE	FALSE	Do not publish		6	1	Parameter2	INT		FAL	SE I	FALSE		publish	
—																			
	<pre>climite</pre>																		
															_	_			
							~→ Cop	by Selected f	rom Le	ft to	Right 🛟 Co	py All Diffe	erence from L	eft to Ri	ght	င်္ဘာ Ur	ndo Selec	ted 🚓 l	Indo All
ŀ.,	.egend:		Left side a	nly Right sig		Copied												Arrela	
	egenu.		cert side d			Copied									В	Back R	ecompare	e Apply	Close

18. Click the **Apply** button, and then click the **Back** button. The **Sysmac Diff** dialog box appears again.

Sysmac Diff				
arget:bf32f50 (HEAD -> ProgramB_work)	Target: Update Date and	•	Source:Working project	Detailed Comp
 Data Trace Settings 	8/31/2017 12:29:40 PM	8/31/2017 12:29:40 PM	 Data Trace Settings 	
	n 9/1/2017 9:58:06 AM	9/1/2017 9:23:08 AM		n
	8/31/2017 12:29:40 PM	8/31/2017 12:29:40 PM		
	8/31/2017 12:29:40 PM	8/31/2017 12:29:40 PM		
V Programming				
V POUs			▼ POUs	
→ Programs			-▼ Programs	
-▼ ProgramA	9/1/2017 9:23:08 AM	9/1/2017 9:23:08 AM	✓ ProgramA	
Variables	9/1/2017 9:23:08 AM	9/1/2017 9:36:11 AM	Variables	
Section0	9/1/2017 9:23:08 AM	9/1/2017 9:36:11 AM	Section0	
-▼ ProgramB	9/1/2017 9:23:08 AM	9/1/2017 9:23:08 AM	▼ ProgramB	
	9/1/2017 9:58:06 AM	9/1/2017 9:58:06 AM		
Section0	9/1/2017 9:58:06 AM	9/1/2017 9:58:06 AM	Section0	
- Function Blocks			Function Blocks	
🗸 Data			-▼ Data	
- Data Types	8/31/2017 12:29:40 PM	8/31/2017 12:29:40 PM	Data Types	
Global Variables	9/1/2017 9:58:06 AM	9/1/2017 9:36:11 AM	Global Variables	
			Select	t this to overwrite with left Cance
gend: Different Left side only Right s	ide only Conflict Resolved	d Overwriting targets with le		Overwrite Close

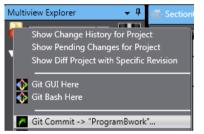
19. Make sure there is no other conflict (with purple bar) and click the **Close** button.

When the project is opened, you can find, in the working branch *ProgramB_work*, the variable *P1_Input2* which was registered by Developer A.

Multiview Explorer 🗸 🖣	😑 Secti	on0 - ProgramA 🗙	
🚺 new_Controller_0 🔻 📲	Varia	bles	
▼ Configurations and Setup	0	P1_Input P1_Input2	P1_Output

The conflict was resolved and changes made by Developer A and B were successfully merged.

20. Next, commit the changes (merge result) to the local repository. Right-click the folder icon in the Multiview Explorer and select **Git Commit -> "ProgramB_work"** from the pop-up menu.



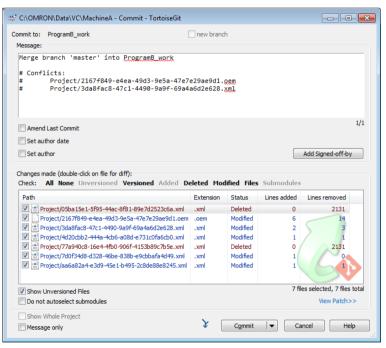
When the project is not yet saved, a confirmation dialog box is displayed to ask if you need to save the project. Click the **Yes** button.

The following dialog box is displayed with the "TortoiseGit" Commit dialog box.

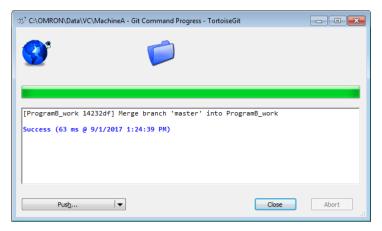


This dialog box is displayed in order to call attention for the conflict occurred when merging the branches.

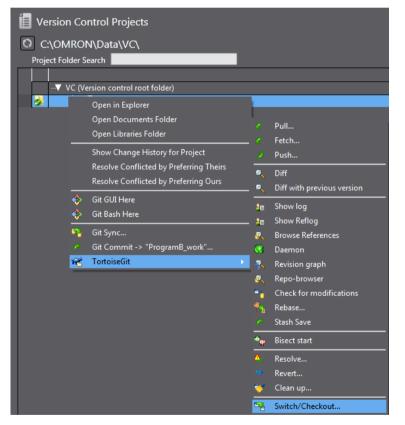
- 21. Click the **OK** button.
- 22. The **Message** area of the "TortoiseGit"Commit dialog box automatically shows comments that conflicts occurred when merging the branches. If necessary, add or change comments in the **Message** area and click the **Commit** button.



This completes Commit processing.

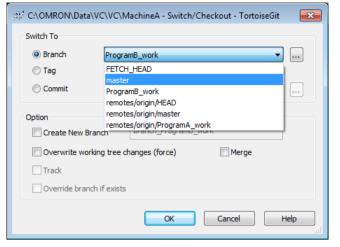


- 23. Click the **Close** button.
- 24. Then, switch to the main branch and apply it the data in the branch *ProgramB_work* after merging. Right-click the target project in the **Version Control Projects** window and select **TortoiseGit -Switch/Checkout** from the pop-up menu.



The "TortoiseGit" Switch/Checkout dialog box is displayed.

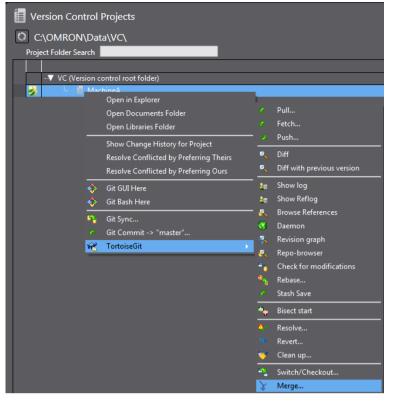
25. Select **Branch** option, then *master* (main branch), and click the **OK** button.



The branch is switched.

$\mathbb{S}_{2^n}^{k^*}$ C:\OMRON\Data\VC\MachineA - Git Command Progress - TortoiseGit	
S 📁	
git.exe checkout master	
Your branch is up-to-date with 'Origin/master'. Switched to branch 'master'	
Success (78 ms @ 9/1/2017 1:31:05 PM)	
Merge	Abort .:i

- 26. Click the **Close** button.
- 27. Right-click the target project in the **Version Control Projects** window and select **TortoiseGit Merge** from the pop-up menu.



The "TortoiseGit" Merge dialog box is displayed.

28. Select **Branch** option, then the working branch *ProgramB_work* to merge it to the main branch *master,* and click the **OK** button.

🔆 C:\OMRON\Dat	a\VC\MachineA - Merge - TortoiseGit	×
From		
Branch	remotes/origin/master	ן ר
🔘 Tag	FETCH_HEAD	_
Commit	ProgramB_work remotes/origin/HEAD	ן ר
Option	remotes/origin/master	_
Squash	remotes/origin/ProgramA_work	
No Fast For	ward Fast Forward Only	
No Commit		
Strategy	▼	
Merge Message		
<auto gener<="" th=""><td>ated by Git></td><td></td></auto>	ated by Git>	
	OK Cancel Help	
		_

A dialog box is displayed to indicate the completion of the merging.

ట్లి C:\OMRON\Data\VC\MachineA - Git Command Progress - TortoiseGit	- • •
git.exe merge ProgramB_work	_
Updating f50246314232df Fast-forward Project/2167f849-e4ea-49d3-9e5a-47e7e29ae9d1.oem 6 +++ Project/2b0c86ee-b9bb-43e7-af8c-7a8a3bc22db0.xml 2 +- Project/3da67ac8-47c1-4490-9a9f-69a4a6d2e628.xml 2 +- Project/8c280bcd-6469-4a6b-b412-9eb0ea117f9a.xml 2 ++ 4 files changed, 7 insertions(+), 5 deletions(-) Success (78 ms @ 9/1/2017 1:36:19 PM)	
Remove branch Close	Abort

The data in the main branch is overwritten with the merged data after resolving the conflicts caused by changes of Developer A and B.

- 29. Click the **Close** button.
- 30. Right-click the folder icon in the Multiview Explorer and select **TortoiseGit Show log** from the pop-up menu. The "TortoiseGit" **Log Messages** dialog box is displayed.

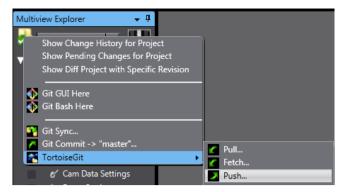
Graph Actio			5/ 1/2017		ter by Subject, M	lessages, Pat	ths, Author	s, Emails, SHA-1, Refname, Note	es Author Email	
	ns Message						Author	Date		
	Working tree ch	anges								
h 🙆	🖌 ProgramB_wo	rk master Merg	e branch 'n	naster' into	ProgramB_wo	nk	UserB	9/1/2017 1:24:39 PM		
Merge branch	Modify initial val Add and Modify Add Data types Changed Config	and Global variable uration and Setup project for Machine 16dce4a6e97cdd	e 'Parameter: s A 4ba03			nput2', A	UserA UserA UserA UserA UserA UserA	9/1/2017 9:38:04 AM 9/1/2017 10:01:26 AM 9/1/2017 9:23:13 AM 9/1/2017 9:22:13 AM 9/1/2017 9:22:53 AM 9/1/2017 9:20:53 AM 9/1/2017 9:19:47 AM		
	/ <u>2167f849</u> -e4ea- /3da8fac8-47c1-									
Project	/ <u>2167f849</u> -e4ea- / <u>3da8fac8</u> -47c1-				Lines added	Lines remo	wed			
Project Project	/ <u>3da8fac8</u> -47c1-		4a6d2e628	.xml	Lines added	Lines remo	wed			
Project Project ath Diff with parent	/ <u>3da8fac8</u> -47c1-	4490-9a9f- <u>69a</u>	4a6d2e628	.xml	Lines added		wed			
Project Project ath Diff with parent	/ <u>3da8fac8</u> -47c1-	4490-9a9f- <u>69a</u> 9e7d2523c6a.xml	Extension	.xml Status						
Project Project	/ <u>3da8fac8</u> -47c1- 1: bf32f50 :1-5f95-44ac-8f81-89	4490-9a9f- <u>69a</u> 0e7d2523c6a.xml 7e7e29ae9d1.oem	Extension	.xml Status Deleted	0		131			
Project Project Path Diff with parent Project/05ba15e Project/2167f84	/ <u>3da8fac8</u> -47c1- l: bf32f50 :1-5f95-44ac-8f81-89 9-e4ea-49d3-9e5a-4	4490-9a9f- <u>69a</u> 9e7d2523c6a.xml 7e7e29ae9d1.oem Ia4a6d2e628.xml	Extension .xml .oem	. xm1 Status Deleted Modified	0		131 14			
Project Project ath Diff with parent Project/05ba15e Project/2167f84 Project/3da8fac Project/3da8fac	/ <u>3da8fac8</u> -47c1- .: bf32f50 .: 1-5f95-44ac-8f81-89 9-e4ea-49d3-9e5a-4 8-47c1-4490-9a9f-69	4490-9a9f- <u>69a</u> e7d2523c6a.xml 7e7e29ae9d1.oem la4a6d2e628.xml 731c0fa6cb0.xml	Extension .xml .oem .xml	. xml Status Deleted Modified Modified	0	2	131 14 3			

On the upper pane, you can know that the edits made by Developer B on the working branch *PrgramB_work* have been merged to the main branch *master*. Click the **OK** button to close the dialog box.

7.3.3.13 Push the changes in the main branch of Developer B to the remote repository (No.13: Developer B)

Apply (Push) the changes merged in the Developer B's computer into the remote repository.

1. Right-click the folder icon in the Multiview Explorer or on the target project in the **Version Control Projects** window, and select **TortoiseGit - Push**.

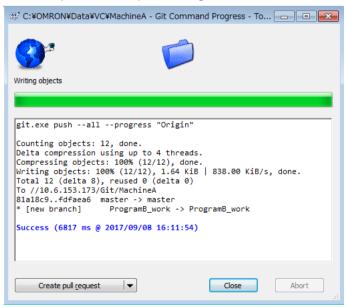


The "TortoiseGit" Push dialog box is displayed.

2. Make sure that the **Push all branches** check box is selected and **Remote** item under **Destination** is correctly set ("Origin" is set in this example), and then click the **OK** button.

🐺 C:¥OMRON¥Da	ata¥VC¥MachineA - Push - TortoiseGit	×
Ref	res	
Local:	master 🔹]
Remote:	master 💌 🗔	
Destination		
Remote:	Origin Manage	
O Arbitrary URL:]
Options		
Force: May discare		
Use Thin Pack	(For slow network connections)	
Autoload Putty	/ Kev	
	track remote branch	
Always push to	o the selected remote archive for this local branch	
Always push to	o the selected remote branch for this local branch	
Recurse submodul	le None 🔹	
	OK Cancel Help	

This completes Push processing.



3. Click the **Close** button.

4. Right-click the folder icon in the Multiview Explorer and select **TortoiseGit - Show log** from the pop-up menu. The "TortoiseGit" **Log Messages** dialog box is displayed.

		From: 2017/09/08 - To:		••		es, Paths, Authors, Email) (
Graph	Actions	Message						Author	Date	-
		Working tree changes								
h.	e e	origin/ProgramA work Ad					ork Merge	UserB	2017/0 2017/09	
11	8	Modify initial value of Glob						UserA	2017/09	
IJ	õ.	Add and Modify Program P		arameter 1 A	du LD of Program	ind indi		UserA	2017/09	
Ţ.	à l	Add Data types and Globa						UserA	2017/09	L
↓	ŏ	Changed Configuration an						UserA	2017/09	
1		Denieter empty project for	r Machine A					HearA	2017/00	
Merge Confli	branch 'mast	ea09d98ce742bbbe814447 eer' into ProgramB_wor <u>beb0</u> -c435-4904-baca- <u>b2</u> ;	k	<u>5</u> .xml						
Merge Confli P	branch 'mast	ter' into ProgramB_wor	k	5.xm1 Status	Lines added	Lines removed				
Merge Confli P	branch 'mast	:er' into ProgramB_wor <u>leb0</u> -c435-4904-baca- <u>b2</u> ;	k a7c1e93bd		Lines added	Lines removed				[
Merge Confli P Path Diff wit	branch 'mast icts: 'roject/ <u>09b60</u> h parent 1: 56	:er' into ProgramB_wor <u>leb0</u> -c435-4904-baca- <u>b2</u> ;	k a7c1e93bd Extension		Lines added	Lines removed				[
Merge Confli Path Diff wit	branch 'mast icts: 'roject/ <u>09b60</u> h parent 1: 56	ter' into ProgramB_wor l <u>eb0</u> -c435-4904-baca- <u>b2</u> aedb9	k a7c1e93bd Extension	Status						
Merge Confli Path Diff wit	branch 'mast ccts: 'roject/ <u>09b60</u> h parent 1: 56 :/09b60eb0-c435- :/3b40f31f-ecd3	ter' into ProgramB_wor heb0-c435-4904-baca-b2 aedb9 4904-baca-b2a7c1e93bd5.xml kS9a-bad6-80ec3b01dc1b.xml	k a7c1e93bd Extension .xml	Status Modified						:
Merge Confli Path Diff wit Project Project	branch 'mast icts: 'roject/ <u>09b60</u> h parent 1: 56 /09b60eb0-c435 /3b40f31f-ecd3 /590d1f57-c402-	ter' into ProgramB_wor <u>leb8</u> -c435-4904-baca- <u>b2</u> aedb9 4904-baca-b2a7c1e93bd5.xml k3a-bad6-80ec3b01dc1b.xml	k a7c1e93bd Extension .xml .xml .xml	Status Modified Modified	1					[
Merge Confli P Path Diff wit Project Project Project	branch 'mast icts: 'roject/ <u>09b60</u> h parent 1: 56 /09b60eb0-c435- /09b0157-c402- /590d157-c402-	ter' into ProgramB_wor teb0-c435-4904-baca-b2 aedb9 4904-baca-b2a7c1e93bd5.xml K9a-bad6-80ec3b01dc1b.xml 4007-add0-2e83423776f6.xml 486c-b7e6-9827f2cea971.cem	k a7c1e93bd Extension .xml .xml .xml .xml .oem	Status Modified Modified Modified	1 1 3 3	0 1 0 3				_ [
Merge Confli P Path Diff wit Project Project Project	branch 'mast icts: 'roject/ <u>09b60</u> h parent 1: 56 /09b60eb0-c435- /09b0157-c402- /590d157-c402-	ter' into ProgramB_wor leb0-c435-4904-baca-b22 aedb9 490+baca-b2a7c1e93bd5.vml k9a-bad6-b0ec3b01dc1b.vml 400-bad0-2e83423776f6.vml	k a7c1e93bd Extension .xml .xml .xml .xml .oem	Status Modified Modified Modified	1 1 3 3	0 1 0 3	: modified = 0	added = 0 c	deleted = 0 replace	_ [

On the upper pane, you can find that the edits made by Developer B on the working branch *PrgramB_work* are taken into the main branch *master* and the *master* of the remote repository (origin) also synchronizes. Click the **OK** button to close the dialog box.

7.4 Derived Development of Sysmac Studio Projects

This section describes the operating procedure to promote derived development of Sysmac Studio projects on an example.

This example uses a configuration in which the remote repository is shared using the shared folder.

7.4.1 An example of derived development

In this example, Machine B is developed and derived from Machine A. While the Machine B is being developed, defects were found and corrected in the common program of Machine A and Machine B. The correction is applied to the project of Machine B to complete the development.

7.4.1.1 About the Project

It is assumed that, based on the project of Machine A, one developer develops a project of derivative Machine B. ProjectA: the project of Machine A. On the base of this project, the project of the derived Machine B is developed. ProjectB: the project of Machine B derived from Machine A project. It has different configurations and setup. This project utilizes the same program as that of ProjectA.

7.4.1.2 Scenario

Create and manage branches *master*, *ProductA*, *and ProductB*; *master* is for common program management, *ProductA* is for Machine A, and *ProductB* is for Machine B. The common program is modified in the *master* and the changes are applied to the *ProductA* and *ProductB*.

- 1. In the development of Machine A, create a new project *ProjectA* and incorporate it into the Machine A as Ver.1.0. Register this *ProjectA* to the *master* as the master project of derivation development.
- 2. In the development of Machine B, create *ProjectB* on the base of *ProjectA* Ver.1.0 and change the settings in Configurations and Setup.
- 3. Defects are found in the program of Machine A and the program is modified to fix them. Modify the program in the *master* project and apply the corrections to the Machine A as Ver.1.1.
- 4. In the same way, apply the corrections to the *ProjectB* which is being developed and incorporate it to the Machine B as its Ver.1.0.

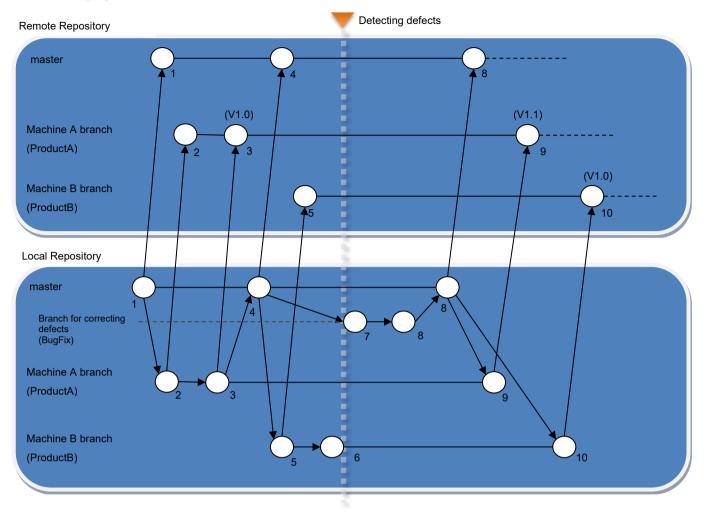
No.	Machine A development	Machine B development	Correction of defects	Reference for the tasks
1	Register a project in the remote repository	-	-	7.4.3.1 Register a project in the remote repository (No.1: Machine A development)
2	Create a branch for Machine A in the local repository and push it to the remote repository * The project of Machine A is developed in this branch.	-	-	7.4.3.2 Create a branch for Machine A in the local repository and push it to the remote repository (No.2: Machine A development)
3	Create the program of Machine A, commit it, and push it to the remoter repository * The tag as Ver.1.0 is applied to this project and the project is incorporated in the Machine A.	-	-	7.4.3.3 Create the program of Machine A, commit it, and push it to the remoter repository (No.3: Machine A development)

7.4.2 Operation flow on the version control system

4	Merge the changes in the Machine A to the <i>master</i> in the local repository and push it to the remote repository * Register the project of the Machine A in the <i>master</i> as the master project of subsequent derivation development.		-	7.4.3.4 Merge the changes in the Machine A into the <i>master</i> in the local repository and push it to the remote repository (No.4: Machine A development)
5	-	Create a branch for Machine B in the local repository and push it to the remote repository * The project of Machine B is developed in this branch.	-	7.4.3.5 Create a branch for Machine B in the local repository and push it to the remote repository (No.5: Machine B development) *The Project of Machine B is developed in this branch.
6	-	Modify the settings in Configurations and Setup of the Machine B and commit them * Right after this, defects are found in the common program and they are corrected.	-	7.4.3.6 Modify the settings in Configurations and Setup of the Machine B and commit them (No.6: Machine B development) *Right after this, defects are found in the common program and they are corrected.
7	-	-	From the <i>master</i> in the local repository, create a branch for correcting defects.	7.4.3.7 From the <i>master</i> in the local repository, create a branch for correcting defects (No.7: Correction of defects)
8	-	-	Commit correction of the defects, merge them to the <i>master</i> and push it to the remote repository	7.4.3.8 Commit correction of the defects, merge it to the <i>master</i> and push it to the remote repository (No.8: Correcting defects)
9	Merge correction of defects into the branch of the Machine A and push it to the remote repository * Apply a tag as Ver.1.1 to this project and incorporate it into the Machine A.	-	-	7.4.3.9 Merge correction of defects into the branch of the Machine A and push it to the remote repository (No.9: Machine A development) *Apply a tag as Ver.1.1 to this project and incorporate it into the Machine A.
10	-	Merge the correction of defects to the branch of Machine B and push it to the remote repository * Apply a tag as Ver.1.0 to this project and incorporate it into the Machine B.	-	7.4.3.9 Merge correction of defects into the branch of the Machine A and push it to the remote repository (No.9: Machine A development) *Apply a tag as Ver.1.0 to this project and incorporate it into the Machine B.

-: No task

The following figure illustrates the relationship between the repositories and branches.



7.4.3 Operating Procedure

Operating procedure is explained along with the steps in 7.4.1.2 Scenario.

7.4.3.1 Register a project in the remote repository (No.1: Machine A development)

1. Creating a remote repository

Create a remote repository to manage the deliverables. Here, create a remote repository with the folder name *C*: \ *Git* \ *MachineA.git* in the Developer's computer. Refer to *5.6 Creating the shared folder and remote repository* for details.

 Registering a local repository and creating and saving the base project Create a new project and register it in the local repository. Here, create a project *MachineA*, commit it to the local repository, and then push it to the remote repository. Refer to *6.3 Preparing for Starting Project Control (Creating a Base Project and Saving It)* for details.

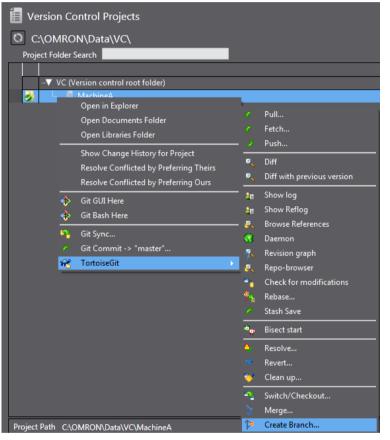
7.4.3.2 Create a branch for Machine A in the local repository and push it to the remote repository (No.2: Machine A development)

1. Synchronizing the remote repository and local repository

After the project is registered in the remote repository, there is a possibility that other developers are updating the target project in the remote repository. Configure the settings to synchronize the remote repository to be the master and the local repository in the Developer A's computer. Refer to 7.3.3.2 Load the project from the remote repository (No.2: Developer A) for details.

In this scenario, this operation is unnecessary immediately after the procedure described in 7.4.3.1 Register a project in the remote repository (No.1: Machine A development) since only the Developer A accesses the remote repository.

2. Right-click the folder icon in the Multiview Explorer or on the target project in the **Version Control Projects** window, and select **TortoiseGit - Create Branch**.



The "TortoiseGit" Create Branch dialog box is displayed.

3. Enter the branch name (in this example, *ProductA*), and click the **OK** button. For **Base On**, select the latest revision. Select **Head (master)**.

🕸 C:\OMRON\Dat	a\VC\MachineA - Create Branch - TortoiseGit
Name Branch	ProductA
Base On HEAD (master	r)
O Branch	master 💌
🔘 Tag	
Commit	
Options	
Track	Force Switch to new branch
Description	
	OK Cancel Help

The branch is successfully created.

4. Push the created branch *ProductA* from the local repository to the remote repository. For the procedure to push the project data to the remote repository, refer to *6.4.3 Registering the Changes to the Remote Repository* (*Push*).

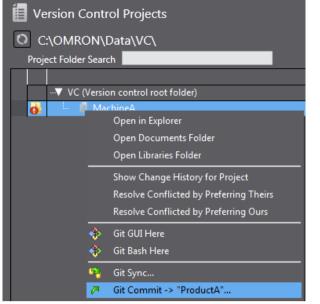
7.4.3.3 Create the program of Machine A, commit it, and push it to the remoter repository (No.3: Machine A development)

Switch the working branch on the developer's computer and register Global Variables and create programs. After creation, consider this as a finished data and apply the tag as "Version 1.0". Then, push this to the remote repository.

- 1. Switch to the branch for Machine A *ProductA* to edit it. For the procedure to switch to the branch to edit, refer to 7.3.3.7 *Commit changes of Program A and Global Variables to the working branch (No.8, 9: Developer A)*.
- 2. Create the program for the product A. The figure below illustrates an example of the program.

🗧 Sectio	n0 - Program0 🗙							
Variab	oles							
Name	space - Using							
Internals	Name	Data Type	Initial Value	I AT	Retain	Constant	Comment	1
Externals	P1_Input	BOOL						
	P1_Output	BOOL						
	P1_Input2	BOOL						
0	P1_Input P1_Input2						P1_Output	

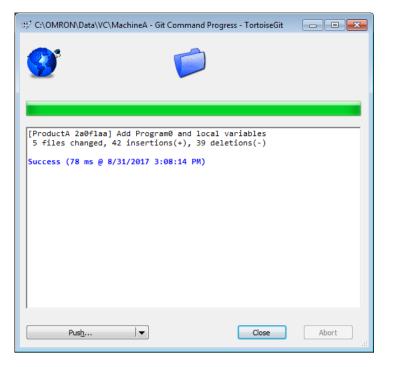
- 3. From the Main menu, select File Save to save the project.
- 4. Select **File Close** to close the project.
- 5. Right-click the target project in the **Version Control Projects** window and select **Git Commit -> "ProductA"** from the pop-up menu.



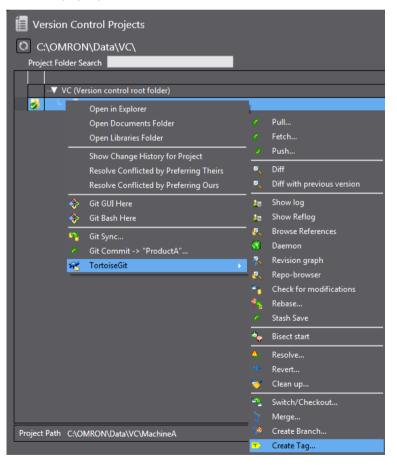
The "TortoiseGit" Commit dialog box is displayed.

		branch			
Message:					
Add RC	ogram0 and local variables				
-					1/
Ameno	d Last Commit				
📃 Set au	uthor date				
Set au				Add Sig	ned-off-by
				Add Sig	ned-off-by
Set au				Add Sig	ned-off-by
Set au	uthor made (double-click on file for diff):	eleted Mod i	ified File		
Set au	uthor made (double-click on file for diff):	eleted Modi Extension	i fied File Status		
Set au Changes n Check: Path	uthor made (double-click on file for diff):			s Submodules	;
Changes n Check: Path	uthor made (double-dick on file for diff): All None Unversioned Versioned Added D	Extension	Status	s Submodules	Lines rem
Changes r Check: Path	uthor made (double-dick on file for diff): All None Unversioned Versioned Added D Project/433d135f-2e47-4539-b492-dbdfd3e9f75d.xml	Extension .xml	Status Modified	s Submodules Lines added 5	Lines rem
Changes r Check: Path	uthor made (double-dick on file for diff): All None Unversioned Versioned Added D Project/433d135f-2e47-4539-b492-dbdfd3e9f75d.xml Project/4d20cbb2-444a-4cb6-a08d-e731c0fa6cb0.xml	Extension .xml .xml	Status Modified Modified	 Submodules Lines added 5 1 	Lines rem 2
Set au Changes n Check: Path ♥ ♥ ₽ ♥ ♥ ₽	uthor made (double-dick on file for diff): All None Unversioned Versioned Added D Project/433d135f-2e47-4539-b492-dbdfd3e9f75d.xml Project/4320cbb2-444a-4cb6-a08d-e731c0fa6cb0.xml Project/536b893b-4f01-4c3f-b48f-8a7cd2a42605.oem	Extension .xml .xml	Status Modified Modified	s Submodules Lines added 5 1 32	Lines rem 2
Set au Changes n Check: Path Ø @ P Ø @ P Ø P Ø P	withor made (double-click on file for diff): All None Universioned Versioned Added D project/433d135f-2e47-4539-b492-dbdfd3e9f75d.xml project/4320cbb2-444a-4cb6-a08d-e731c0fa6cb0.xml project/536b893b-4f01-4c3f-b48f-8a7cd2a42605.oem III	Extension .xml .xml	Status Modified Modified	s Submodules Lines added 5 32 5 files select	Lines rem 2
Set au Changes n Check: Path V • P V • P	withor made (double-click on file for diff): All None Unversioned Versioned Added D Project/433d135f-2e47-4539-b492-dbdfd3e9f75d.xml Project/4420cbb2-444a-4cb6-a08d-e731c0fa6cb0.xml Project/536b893b-4f01-4c3f-b48f-8a7cd2a42605.com III Unversioned Files	Extension .xml .xml	Status Modified Modified	s Submodules Lines added 5 32 5 files select	Lines rem

6. Enter comment in the **Message** area and click the **Commit** button. Commit is completed.



- 7. Click the **Close** button.
- 8. Right-click the target project in the **Version Control Projects** window and select **TortoiseGit Create Tag** from the pop-up menu.



The "TortoiseGit" Create Tag dialog box is displayed.

9. Enter the version No. or other text string to identify the tag in the **Tag** text box and click the **OK** button. In this example, enter *V1.0.*

Tag	V1.0		
Base On			
) <u>H</u> EAD (Produ	ictA)		
© <u>B</u> ranch	ProductA		▼
© <u>T</u> ag			-
Commit			•
Options			
Trac <u>k</u>	Eorce	Sign	
<u>M</u> essage			

10. Right-click the folder icon in the MultiView Explorer or the project in the **Version Control Projects** window and select **TortoiseGit - Show log.**

The "TortoiseGit" Log Messages dialog box is displayed.

11. You can see that the tag name is displayed in the Working tree changes field.

oductA		From:	8/31/201	17 -	To:	8/31/2017	7 - issag	es, Paths, Autho	rs, Emails, SH Au		🔁 🛃
Graph	Actions	Message					Autho	r Date			
		Working t	ree change	es							
•	ø	Product/	V1.0	Add	Program) and local	lv User	A 8/31/20)17 3:08:14)	
•	4	master 0	rigin/Produ	uctA	Origin/ma	ister Regist	er UserA	8/31/201	7 2:58:45 PM		
Add Pr	ogram0 ar	nd local	variab.	les							
	ogram0 ar	nd local	Variab.	les							
Path						Extension	Status	Lines added	Lines removed		
P ath ■ Project	- /433d135f-2	e47-4539-b	492-dbdfd	13e9f7		.xml	Modified	5	5		
P ath ■ Project	/433d135f-2/	e47-4539-b 44a-4cb6-a	492-dbdfd 08d-e731c	13e9f) c0fa6o	zb0.xml	.xml .xml	Modified Modified	5	5		
Path Project Project	/433d135f-2/ /4d20db2-4 /536b893b-4	e47-4539-b 44a-4cb6-a ₩01 -4c3f-b +	492-dbdfd 08d-e731c 18f-8a7cd ?	13e9f) c0fa6o 2a426	cb0.xml i05.oem	.xml .xml	Modified	5	5		
Path Project Project Project	/433d135f-2/	e47-4539-b 44a-4cb6-a if01-4c3f-b · 328-46be-8	492-dbdfd 08d-e731c 18f-8a7cd 38b-e9cbb	13e9f7 c0fa6o 2 a426 pafa4o	cb0.xml i 05.oem d49.xml	.xml .xml .oem .xml	Modified Modified Modified	5 1 32	5 1 32		
Path Project Project Project Project	/433d135f-2 /4d20cbb2-4 /536b893b -4 /537d0 54d8 /aa6a82a4-e	e47-4539-b 44a-4cb6-a f01-4c3f-b 328-46be- 328-45be-3 3d9-45e1-b	492-dbdfd 08d-e731c 18f-8a7cd 38b-e9cbb 1495-2c8dd	13e9f; c0fa6c 2a426 aafa4c e88e8	cb0.xml i0 5.oem d49.xml 8245.xml	.xml .xml .oem .xml .xml	Modified Modified Modified Modified	5 1 32 3 1	5 1 32) files: mod	ified = 0 added

12. Push the changes (entry of a program) in the branch of Machine A from the local repository to the remote repository. Refer to *6.4.3 Registering the Changes to the Remote Repository (Push)* for details of the procedure.

In the "TortoiseGit" Push dialog box, select the check box for Include Tags under Options, and then click the

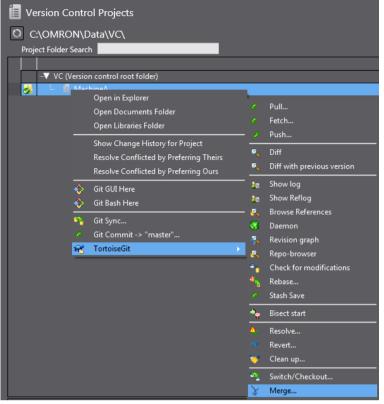
OK button.

🕸 C:\OMRON\Da	ata\VC\MachineA - Push - TortoiseGit	×
Ref		
✓ Push all bran	ches	
Local:	ProductA 👻 🕨]
<u>R</u> emote:	▼]
Destination		
• Remote:	Origin Manage]
C Arbitrary UR	.:]
Options		
Force: May disca		
	k (For slow network connections)	
Include Tags Autoload Put		
	n/track remote branch	
	to the selected remote archive for this local branch	
Always push	to the selected remote branch for this local branch	
Recurse submod	Jule None 🗸	
	OK Cancel Help	

7.4.3.4 Merge the changes in the Machine A into the *master* in the local repository and push it to the remote repository (No.4: Machine A development)

Switch to the *master* branch and then merge the changes in the the Machine A branch *ProductA* into the *master* branch.

- 1. Switch to the *master* branch to edit it. For the procedure to switch to the branch to edit, refer to 7.3.3.7Commit changes of Program A and Global Variables to the working branch (No.8, 9: Developer A).
- 2. To merge the changes to the *master* branch, right-click the target project in the **Version Control Projects** window and select **TortoiseGit Merge** from the pop-up menu.



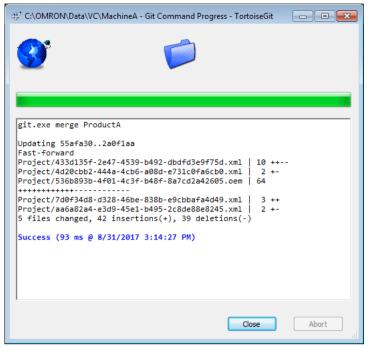
The "TortoiseGit" **Merge** dialog box is displayed.

3. Select Branch option, then the working branch *ProductA*, and click the OK button.

From				
<u> Branch</u>	ProductA			•
) <u>T</u> ag	V1.0			-
Ommit				·
Option		_		
Squash		Messages	20	
No <u>F</u> ast For	ward	E Fast Forward	O <u>n</u> ly	
No Commit				
Strategy	•		•	
Merge <u>M</u> essage				
<auto gener<="" td=""><td>ated by Git></td><td></td><td></td><td></td></auto>	ated by Git>			

In this scenario, since the contents of the *ProductA* branch and *Tag V1.0* are the same, you can also select **Tag** option and *V1.0*.

Merging is completed.



- 4. Click the **Close** button.
- 5. Push the changes (entry of program) in the branch of Machine A from the local repository to the remote repository. For the procedure to push the project data to the remote repository, refer to *6.4.3 Registering the Changes to the Remote Repository (Push)*.

7.4.3.5 Create a branch for Machine B in the local repository and push it to the remote repository (No.5: Machine B development)

Here, create a branch *ProductB* from the *master*, and then push the changes to the remote repository. For details on the series of procedure, refer to 7.4.3.2 *Create a branch for Machine A in the local repository and push it to the remote repository (No.2: Machine A development)*.

In the "TortoiseGit" Create Branch dialog box, configure the settings as follows.

Branch	ProductB	
Base On		
HEAD (mast)	r)	
© <u>B</u> ranch	master	▼
© <u>T</u> ag	V1.0	-
© <u>C</u> ommit		•
Options		
Trac <u>k</u>	Eorce Switch to new	w branch
Description		

Create the *ProductB* branch in the local repository and then push this change to the remote repository. After Push operation, the "TortoiseGit" **Log Messages** dialog box is displayed as shown below.

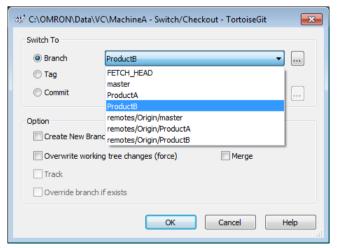
naster		From:	8/31/2017 👻 To	8/31/201	7 👻 issag	es, Paths, Autho	rs, Emails, SF Aut	thor Email	<u> </u>
Graph	Actions	Message							Author
		_	ree changes						
•	ø	Product/	A ProductB master	Origin/Pro	ductA Orig	jin/Product B	Origin/master	V1.0 Add.	UserA
•	÷	Register e	empty project for Mach	ne A					UserA
٠									
	rogram0 ar								
Path				Extension	Status	Lines added	Lines removed		
Project	:/433d135f-2		1492-dbdfd3e9f75d.xm	.xml	Status Modified	Lines added	Lines removed 5		
Project	:/433d135f-2/	44a-4cb6-a	08d-e731c0fa6cb0.xm	.xml .xml	Modified Modified	5	5		
Project	:/433d135f-2 :/4d20cbb2-4 : /536b893b -4	44a-4cb6-a f01-4c3f-b	08d-e731c0fa6cb0.xm 48f-8a7cd2a42605.oer	.xml .xml .oem	Modified Modified Modified	5 1 32	5 1 32		
Project Project Project	:/433d135f-2 :/4d20cbb2-4 :/ 536b893b -4 :/7d0f34d8-d	44a-4cb6-a # f01-4c3f-b 328-46be-8	08d-e731c0fa6cb0.xm 48f-8a7cd2a42605.oe r 138b-e9cbbafa4d49.xm	.xml .xml .oem .xml	Modified Modified Modified Modified	5 1 32 3	5 1 32 0		
Project Project Project	:/433d135f-2 :/4d20cbb2-4 :/ 536b893b -4 :/7d0f34d8-d	44a-4cb6-a # f01-4c3f-b 328-46be-8	08d-e731c0fa6cb0.xm 48f-8a7cd2a42605.oer	.xml .xml .oem .xml	Modified Modified Modified	5 1 32	5 1 32		
Project Project Project Project Project	;/433d135f-2; /420cbb2-4 / 536b893b -4 / 536b893b -4 /736 f 34d8- ;/aa6a82a4-e	44a-4cb6-a f f01-4c3f-b 328-46be-8 3d9-45e1-{	08d-e731c0fa6cb0.xm 48f-8a7cd2a42605.oe r 138b-e9cbbafa4d49.xm	.xml .xml .cem .xml .xml al .xml	Modified Modified Modified Modified	5 1 32 3 1	5 1 32 0 1) files: modif	ied = 0 added

You can see that, in both of the remote and local repositories, *master*, *ProductA* and *ProductB* branches are synchronized.

7.4.3.6 Modify the settings in Configurations and Setup of the Machine B and commit them (No.6: Machine B development)

Create the project for the Machine B and commit it. In this scenario, it is assumed that Machine B project is developed on the base of Machine A project and they utilize the same program, but their Configurations and Setup differ.

 Switch to the branch of Machine B to edit it. For the procedure to switch to the branch to edit, refer to 7.3.3.7 Commit changes of Program A and Global Variables to the working branch (No.8, 9: Developer A). In the "TortoiseGit" Switch/Checkout dialog box, select Branch option and ProductB.



2. Modify the settings in Configurations and Setup for the Machine B. For example, add an EtherCAT slave.

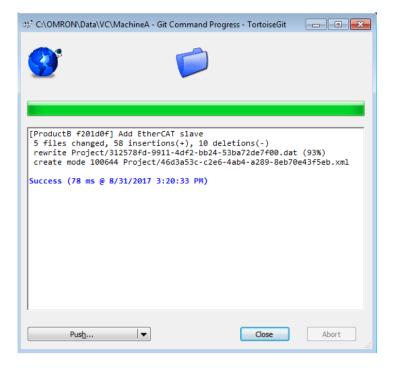
- 3. From the Main menu, select File Save to save the project.
- 4. Select File Close to close the project.
- 5. Right-click the target project in the **Version Control Projects** window and select **Git Commit -> "ProductB"** from the pop-up menu.

🔳 Version Conti	ol Projects
	Data\VC\
Project Folder Sear	ch in the second se
▼ VC (Versio	on control root folder)
i	Open in Explorer
	Open Documents Folder
	Open Libraries Folder
	Show Change History for Project
	Resolve Conflicted by Preferring Theirs
	Resolve Conflicted by Preferring Ours
*	Git GUI Here
*	Git Bash Here
2	Git Sync
P	Git Commit -> "ProductB"

The "TortoiseGit" Commit dialog box is displayed.

Commit to: ProductB Inew branch Message: Add EtherCAT slave	
Add EtherCAT slave	
Amend Last Commit Set author date	1/19
Set author Add Signed-off-	-by
Changes made (double-dick on file for diff): Check: All None Unversioned Versioned Added Deleted Modified Files Submodules	
Path Extension Status Lines added Lines r	
Project/312578fd-9911-4df2-bb24-53ba72de7f00.dat .dat Modified -	=
☑ Project/46d3a53c-c2e6-4ab4-a289-8eb70e43f5eb.xml .xml Added 28	
Image: Project/536b893b-4f01-4c3f-b48f-8a7cd2a42605.oem .oem Modified 26	-
•	- F
Show Unversioned Files 5 files selected, 5 files	les total
Do not autoselect submodules View Patch	h>>
Show Whole Project	
	Help

6. Enter comment in the **Message** area and click the **Commit** button. Commit is completed.

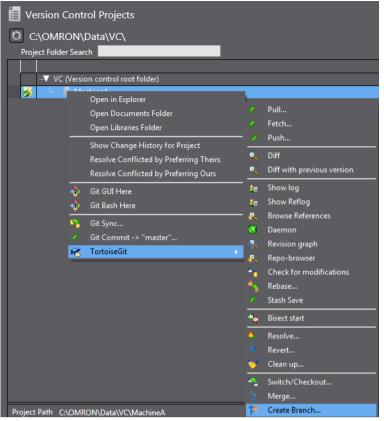


7. Click the **Close** button.

7.4.3.7 From the *master* in the local repository, create a branch for correcting defects (No.7: Correction of defects)

In order to apply correction of defects found in the Machine A to the Machine B, fix defects in the *BugFix* branch that is branched from the *master*. First create the *BugFix* branch.

- 1. Switch to the *master* branch to edit it. For the procedure to switch to the branch to edit, refer to 7.3.3.7 Commit changes of Program A and Global Variables to the working branch (No.8, 9: Developer A).
- 2. Right-click the folder icon in the Multiview Explorer or on the target project in the **Version Control Projects** window, and select **TortoiseGit Create Branch**.



The "TortoiseGit" Create Branch dialog box is displayed.

3. Enter the branch name (in this example, enter *BugFix*), and click the **OK** button. For **Base On**, select the latest revision. Select **Head (master)**.

	BugFix	
Base On		
HEAD (masterna)	er)	
© <u>B</u> ranch	master	· · · · · · · · · · · · · · · · · · ·
© <u>T</u> ag	V1.0	
© <u>C</u> ommit		▼
Options		
Trac <u>k</u>	Eorce	Switch to new branch
Description		

The branch is successfully created.

7.4.3.8 Commit correction of the defects, merge it to the *master* and push it to the remote repository (No.8: Correcting defects)

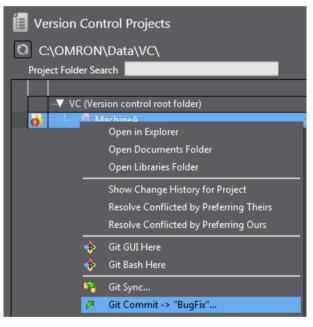
Switch to the *BugFix* branch and modify the program. After that, merge the changes to correct defects to the *master*, and push it to the remote repository.

1. Switch to the *BugFix* branch to fix the defects. For the procedure to switch to the branch to edit, refer to 7.3.3.7 *Commit changes of Program A and Global Variables to the working branch (No.8, 9: Developer A).*

2. Modify the program to fix the defects. The figure below illustrates an example of the program.

	n0 - Program0 🗙							
Variabl	Variables							
Names	space - Using							
Internals	Name	Data Type	Initial Value	I AT	Retain	Constant	Comment	1
Externals	P1_Input	BOOL						
	P1_Output	BOOL						
	P1_Input2	BOOL						
	P2_Input	BOOL						
0	P1_Input						P1_Output	
F								
	P1_Input2							
F								
	P2_Input							
F								

- 3. From the Main menu, select File Save to save the project.
- 4. Select File Close to close the project.
- 5. Right-click the target project in the **Version Control Projects** window and select **Git Commit -> "BugFix"** from the pop-up menu.



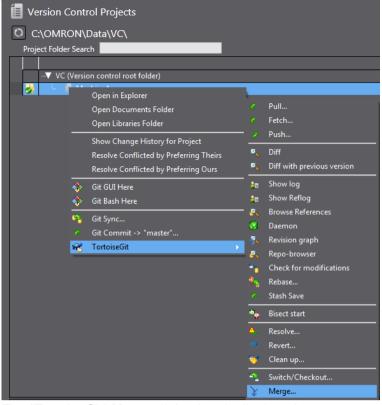
The "TortoiseGit" Commit dialog box is displayed.

🕸 C:\OMR	ON\Data\VC\MachineA - Commit - TortoiseGit				
Commit to: Message:	BugFix new	branch			
Add cor	ntact P2_Input				
	Last Commit				1/21
	thor date				
📃 Set au	thor			Add Sig	ned-off-by
-	nade (double-click on file for diff): All None Unversioned Versioned Added D	eleted Mc	dified Files	5 Submodules	5
Path		Extension	Status	Lines added	Lines removed
	roject/4d20cbb2-444a-4cb6-a08d-e731c0fa6cb0.xml		Modified	1	1
	roject/536b893b-4f01-4c3f-b48f-8a7cd2a42605.oem		Modified	2	
🗹 🖀 Pi	roject/7d0f34d8-d328-46be-838b-e9cbbafa4d49.xml	.xml	Modified	1	C C
•	III				•
V Show I	Unversioned Files			3 files selec	ted, 3 files total
📃 Do not	autoselect submodules			V	iew Patch>>
Show \	Whole Project				
Messa	ge only	Cor	nmit 🛛 🔻	Cancel	Help

6. Enter comment in the **Message** area and click the **Commit** button. Commit is completed.

$\mathbb{S}_{\mathbb{R}^{d}}^{d^{2}}$ C:\OMRON\Data\VC\MachineA - Git Command Progress - TortoiseGit	- • •
[BugFix 83e4637] Add contact P2_Input 3 files changed, 4 insertions(+), 3 deletions(-)	
Success (63 ms @ 8/31/2017 3:28:39 PM)	
Pus <u>h</u> I	Abort

- 7. Click the **Close** button.
- 8. Switch to the *master* branch to edit it. For the procedure to switch to the branch to edit, refer to 7.3.3.7 *Commit* changes of *Program A* and *Global Variables to the working branch (No.8, 9: Developer A)*.
- 9. To merge the changes in the *BugFix* branch to the *master* branch, right-click the target project in the **Version Control Projects** window and select **TortoiseGit Merge** from the pop-up menu.

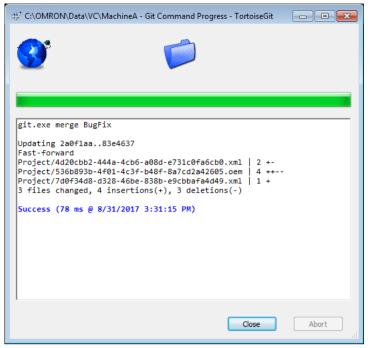


The "TortoiseGit" Merge dialog box is displayed.

10. Select **Branch** option, then the *BugFix* branch, and click the **OK** button.

C:\OMRON\Da	ta\VC\MachineA - Merg	je - TortoiseGit		-
From				
Branch				•
🔘 Tag	BugFix			
Commit	FETCH_HEAD ProductA			
Option	ProductB remotes/Origin/mast remotes/Origin/Prod	uctA		
No Fast For	warc remotes/Origin/Prod			
Strategy	•		·	
Merge Message				
	ated by Git>			
		ОК	Cancel	Help

11. Merging is completed.

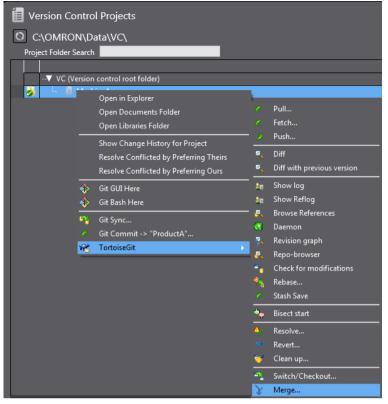


- 12. Click the **Close** button.
- 13. Push the changes in the branch to correct defects and *master* branch from the local repository to the remote repository. For the procedure to push to the remote repository, refer to *6.4.3 Registering the Changes to the Remote Repository (Push)*.

7.4.3.9 Merge correction of defects into the branch of the Machine A and push it to the remote repository (No.9: Machine A development)

Merge the changes to fix defects of the Machine A merged into the *master* to the branch of Machine A. After merging, consider this as a finished data after correcting defects and apply the tag as "Version 1.1". Then, push this to the remote repository.

- 1. Switch to the branch for Machine A *ProductA* to edit it. For the procedure to switch to the branch to edit, refer to 7.3.3.7Commit changes of Program A and Global Variables to the working branch (No.8, 9: Developer A).
- 2. To merge the changes to the *ProductA* branch, right-click the target project in the **Version Control Projects** window and select **TortoiseGit Merge** from the pop-up menu.

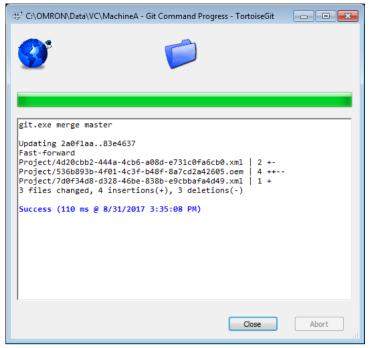


The "TortoiseGit" Merge dialog box is displayed.

3. Select **Branch** option, then *master* of the master branch, and click the **OK** button.

From		
Branch	master	▼
🔘 Tag	V1.0	
Commit		· · · · · · · · · · · · · · · · · · ·
Option		_
C Squash		Messages 20
No Fast For	ward	Fast Forward Only
No Commit		
Strategy	-	
Merge Message		
<auto gener<="" td=""><td>ated by Git></td><td></td></auto>	ated by Git>	

Merging is completed.



- 4. Click the **Close** button.
- 5. Right-click the target project in the Version Control Projects window and select TortoiseGit Create Tag from the pop-up menu.

The "TortoiseGit" Create Tag dialog box is displayed.

6. Enter the version No. or other text string to identify the tag in the **Tag** text box and click the **OK** button. In this example, enter *V1.1*.

Tag	ProductB_V1.0		
Base On			
HEAD (Produ	ictB)		
🔘 Branch	ProductB		
🔘 Tag	V1.0		-
Commit			-
Options			
Track	Force	Sign	
Message			

7. Push the changes in the branch of Machine A (applying defect correction) from the local repository to the remote repository. Refer to *6.4.3 Registering the Changes to the Remote Repository (Push)* for details of the procedure.

In the "TortoiseGit" **Push** dialog box, select the check box for **Include Tags** under **Options**, and then click the **OK** button.

Ref
Local: ProductA V
Remote:
Destination
O Arbitrary URL:
Options Force: May discard known changes unknown changes Use Thin Pack (For slow network connections) I lack (For slow network connections) I lack (For slow network connections) I dual Tags I dual t

7.4.3.10 Merge the correction of defects to the branch of Machine B and push it to the remote repository (No.10: Machine A development)

Merge modifications to correct defects in Machine A merged into the *master* to the branch of the Machine B. After merging, consider this as a finished program of the Machine B and apply the tag as "Version 1.0". Then, push this to the remote repository.

- 1. Switch to the *ProductB* branch of the Machine B to edit it. For the procedure to switch to the branch to edit, refer to 7.3.3.7 *Commit changes of Program A and Global Variables to the working branch (No.8, 9: Developer A)*.
- To merge the changes to the *ProductB* branch, right-click the target project in the Version Control Projects window and select TortoiseGit - Merge from the pop-up menu. The "TortoiseGit" Merge dialog box is displayed.

3. Select **Branch** option, then *master* of the master branch, and click the **OK** button.

얇? C:\OMRON\Dat	ta\VC\MachineA - Merge - TortoiseGit	x
From		
Isranch	master 💌 🗔	
🔘 Tag	V1.0	
Commit	· · · · · · · · · · · · · · · · · · ·	
Option		
📃 Squash	Messages 20	
No Fast For	ward 📃 Fast Forward Only	
No Commit		
Strategy	• •	
Merge Message		
<auto gener<="" th=""><td>ated by Git></td><td></td></auto>	ated by Git>	
	OK Cancel Help	

Merging is completed.

🕸 C:\OMRON\Data\VC\M	achineA - Git Command Progress - TortoiseGit	
git.exe merge master	r.	
Merge made by the 'r Project/4d20cbb2-444 Project/536b893b-4f0 Project/7d0f34d8-d32	4a-4cb6-a08d-e731c0fa6cb0.xml 2 +- 91-4c3f-b48f-8a7cd2a42605.oem 4 ++- 28-46be-838b-e9cbbafa4d49.xml 1 + insertions(+), 3 deletions(-)	
	Close	Abort

4. Click the **Close** button.

<Precaution for correct use>

In the example of this document, conflicts do not occur when merging the changes. If the same portion of data in the Machine A and B are modified, a conflict occurs and merge processing fails. In that case, in order to maintain the changes for Machine B in Configurations and Setup, merge the changes preferring changes of Product B branch. For the procedure to resolve conflicts, refer to 7.3.3.12 Merges the changes in the working branch of Developer B to the main branch (No.12: Developer B).

5. If you open the project, as shown below, you can confirm that the program has been changed while maintaining the Configurations and Setup that was modified for the Machine B.

Multiview Explorer 🔹 🖣	着 Section0 - Program0 🗙	
🤌 new_Controller_0 🔻 📊	Variables	
Configurations and Setup	0 P1_Input P1_Output	4
▼ The EtherCAT	P1 Input2	
L Node1 : NX-ECC201 ► S CPU/Expansion Racks		
at I/O Map	P2_input	
🕨 🕨 🔍 Controller Setup		
Motion Control Setup		
🖋 Cam Data Settings		
Event Settings		
🍋 Task Settings		
☑ Data Trace Settings		

6. Right-click the target project in the Version Control Projects window and select TortoiseGit - Create Tag from the pop-up menu.

The "TortoiseGit" Create Tag dialog box is displayed.

7. Enter the version No. or other text string to identify the tag in the **Tag** text box and click the **OK** button. In this example, enter *ProductB_V1.0.*

Tag	ProductB_V1.0		
-			
Base On			
HEAD (Produ	uctB)		
Branch	ProductB		–
🔘 Tag	V1.0		-
Commit			•
Options			
Track	Force	Sign	
Message			

 Push the changes (changing Configurations and Setup and applying defect correction) in the branch of Machine B from the local repository to the remote repository. Refer to 6.4.3 Registering the Changes to the Remote Repository (Push) for details of the procedure.

In the "TortoiseGit" **Push** dialog box, select the check box for **Include Tags** under **Options**, and then click the **OK** button.

👾 C:\OMRON\Data	a\VC\MachineA - Push - TortoiseGit	— ———————————————————————————————————
Ref	es	
Local:	ProductB	•
Remote:		·
Destination		
Remote:	Origin 🔹 Ma	nage
O Arbitrary URL:		-
Options		
Force: May discard	d known changes unknown change (For slow network connections)	s
Include Tags		
Autoload Putty	Key	
	rack remote branch	
	the selected remote archive for this local branch	
Always push to Recurse submodul	the selected remote branch for this local branch	
Recurse submodul	e None 🔻	
	OK Cancel	Help

9. After Push operation, the "TortoiseGit" Log Messages dialog box is displayed as shown below.

Actions	Message Working tree changes Product B Origin/Pro BugFix Product A maste						
i i i i i i i i i i i i i i i i i i i	ProductB Origin/Pro						Author
i i i i i i i i i i i i i i i i i i i							
ě.	BugFix ProductA maste						UserA
		r Origin/Bug	Fix Origin/F	ProductA Origin	/master V1.1 Ad	d contact P2_Input	UserA
	Add EtherCAT slave						UserA
0	V1.0 Add Program0 and		S				UserA UserA
		111					
		Extension	Status	Lines added	Lines removed		
28f0e7-7756-4	lf1d-8f37-bf4df61f39f2.oem	.oem	Modified	26	8		
5dd7d0-0cbc-4	3b9-85ca-5d14005a84cc.xml	.xml	Modified	3	1		
					-		
	4f45-8c0a-279c25152cd3.xml		Modified	1	1		
ec2785-9635-4	4b10-92de-dd5a9fffb321.dat	.dat	Modified	1	1		
ec2785-9635-4		.dat		1 - 28	1 - 0		
ec2785-9635-4 f80f74-7474-4	4b10-92de-dd5a9fffb321.dat	.dat .xml	Modified Added			10(-) files: modified =	4 added =
	4980c95e86 rCAT slave 28f0e7-7756-4	4980c95e8668375f37040629c36ffe4	## 4980c95e8668375f37040629c36ffe4621207 rCAT slave 28f0e7-7756-4f1d-8f37-bf4df61f39f2.oem cem	III III 4980c95e8668375f37040629c36ffe4621207 Image: Comparison of the comparison	III III 4980c95e8668375f37040629c36ffe4621207 rCAT slave Extension Status Lines added 28f0e7-7756-4f1d-8f37-bf4df61f39f2.oem oem Modified 26	III 4980c95e8668375f37040629c36ffe4621207 rCAT slave 2870e7-7756-4f1d-8f37-bf4df61f39f2.cem .oem Modified 26 8	4980c95e8668375f37040629c36ffe4621207 rCAT slave Extension Status Lines added Lines removed

In the **Message** field, you can know that changes made in the *BugFix* branch were applied to the *master* branch and *ProductA* branch, and contents in the *master* branch (reflection of changes in *BugFix* branch) is applied to the *ProductB* branch.

OMRON Corporation Industrial Automation Company Kyoto, JAPAN

Contact: www.ia.omron.com

Regional Headquarters OMRON EUROPE B.V. Wegalaan 67-69, 2132 JD Hoofddorp The Netherlands Tel: (31)2356-81-300/Fax: (31)2356-81-388

OMRON ASIA PACIFIC PTE. LTD. No. 438A Alexandra Road # 05-05/08 (Lobby 2), Alexandra Technopark, Singapore 119967 Tel: (65) 6835-3011/Fax: (65) 6835-2711

OMRON ELECTRONICS LLC 2895 Greenspoint Parkway, Suite 200 Hoffman Estates, IL 60169 U.S.A. Tel: (1) 847-843-7900/Fax: (1) 847-843-7787

OMRON (CHINA) CO., LTD. Room 2211, Bank of China Tower, 200 Yin Cheng Zhong Road, PuDong New Area, Shanghai, 200120, China Tel: (86) 21-5037-2222/Fax: (86) 21-5037-2200

Authorized Distributor:

© OMRON Corporation 2017-2020 All Rights Reserved. In the interest of product improvement, specifications are subject to change without notice.

Cat. No. P125-E1-03