OMRON

Machine Automation Controller

NJ-series

Database Connection CPU Units

User's Manual

NJ501-1520
NJ501-1420
NJ501-1320
NJ501-4320
NJ101-1020
NJ101-9020

CPU Unit

SYSMAC
always in control

W527-E1-06

Introduction

Thank you for purchasing an NJ-series CPU Unit.
This manual contains information that is necessary to use the Database Connection Service with the NJ-series CPU Unit. Hereinafter the Database Connection Service is called “DB Connection Service”.
Please read this manual and make sure you understand the functionality and performance of the NJ-series CPU Unit before you attempt to use it in a control system.
Keep this manual in a safe place where it will be available for reference during operation.

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.

- NJ-series Database Connection CPU Units
  - NJ501-1520
  - NJ501-1420
  - NJ501-1320
  - NJ501-4320
  - NJ101-1020
  - NJ101-9020
- Sysmac Studio
  - SYSMAC-SE2□□□ (Version 1.14 or higher)

Relevant Manuals

The following table provides the relevant manuals for the NJ-series CPU Units.
Read all of the manuals that are relevant to your system configuration and application before you use the NJ-series CPU Unit.
Most operations are performed from Sysmac Studio Automation Software. Refer to the *Sysmac Studio Version 1 Operation Manual* (Cat. No. W504) for information on Sysmac Studio.

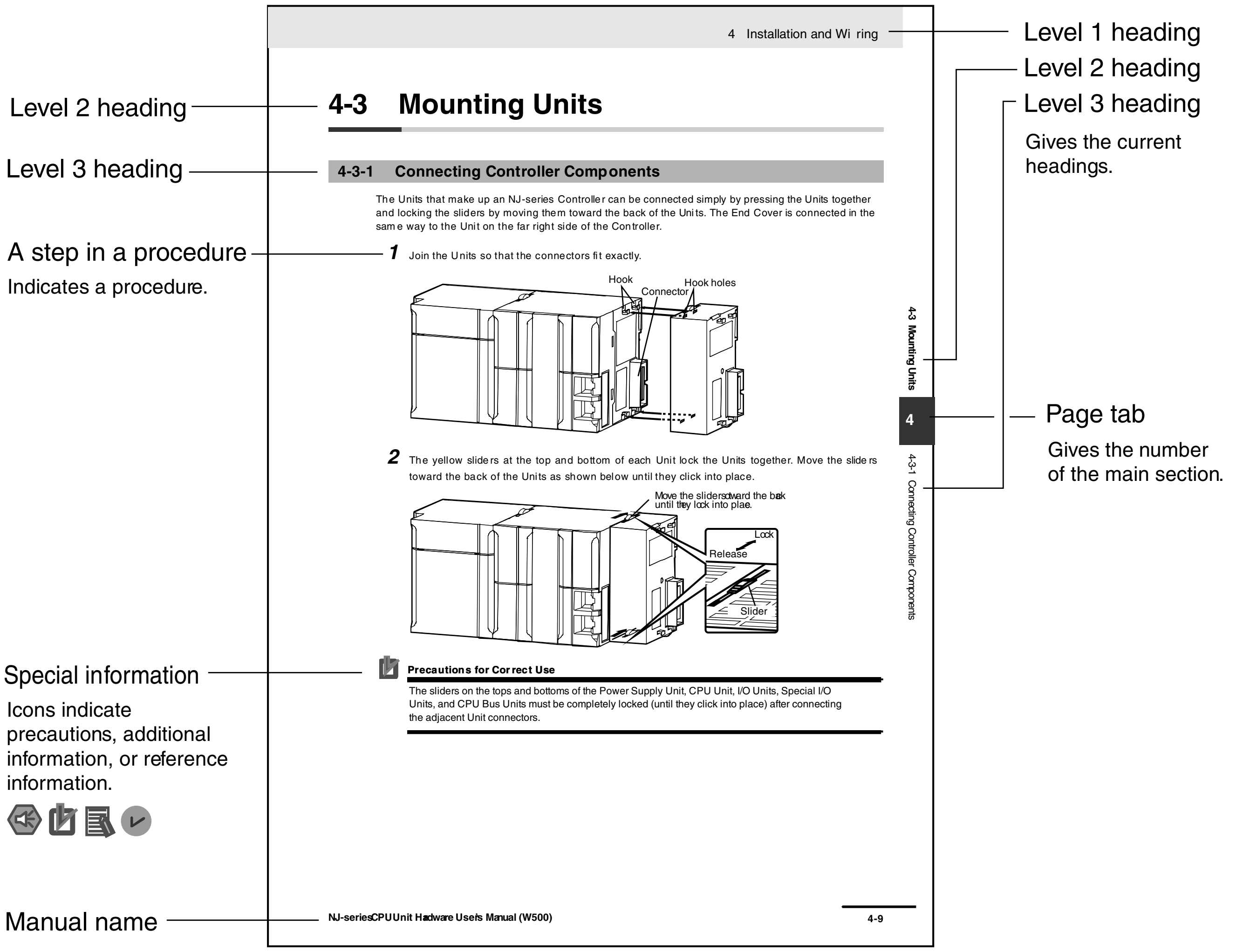
Purpose of use	Manual								
	Basic information								
	NJ-series CPU Unit Hardware User's Manual	NJ/NX-series CPU Unit Software User's Manual	NJ/NX-series Instructions Reference Manual	NJ/NX-series CPU Unit Motion Control User's Manual	NJ/NX-series Motion Control Instructions Reference Manual	NJ/NX-series CPU Unit Built-in EtherCAT Port User's Manual	NJ/NX-series CPU Unit Built-in EtherNet/IP Port User's Manual	NJ-series Database Connection CPU Unit User's Manual	NJ/NX-series Troubleshooting Manual
Introduction to NJ-series Controllers	●								
Setting devices and hardware	●								
Using motion control				●					
Using EtherCAT						●			
Using EtherNet/IP							●		
Using the database connection service								●	
Software settings		●							
Using motion control				●					
Using EtherCAT						●			
Using EtherNet/IP							●		
Using the database connection service								●	
Writing the user program		●	●						
Using motion control				●	●				
Using EtherCAT						●			
Using EtherNet/IP							●		
Using the database connection service								●	
Programming error processing									●
Testing operation and debugging		●							
Using motion control				●					
Using EtherCAT						●			
Using EtherNet/IP							●		
Using the database connection service								●	
Learning about error management and corrections[*1]	▲	▲		▲		▲	▲	▲	●
Maintenance	●								
Using motion control				●					
Using EtherCAT						●			
Using EtherNet/IP							●		

*1 Refer to the *NJ/NX-series Troubleshooting Manual* (Cat. No. W503) for the error management concepts and an overview of the error items. Refer to the manuals that are indicated with triangles for details on errors for the corresponding Units.

Manual Structure

Page Structure

The following page structure is used in this manual.


Level 1 heading
Level 2 heading
Level 3 heading
Gives the current headings.
Level 2 heading
Level 3 heading
A step in a procedure
Indicates a procedure.
Page tab
Gives the number of the main section.
Special information
Icons indicate precautions, additional information, or reference information.
Manual name
Hook
Connector
Hook holes
Move the sliders toward the back until they lock into place.
Lock
Release
Slider
4-3 Mounting Units
4
4-3-1 Connecting Controller Components

4 Installation and Wi ring

4-3 Mounting Units

4-3-1 Connecting Controller Components

The Units that make up an NJ-series Controller can be connected simply by pressing the Units together and locking the sliders by moving them toward the back of the Units. The End Cover is connected in the same way to the Unit on the far right side of the Controller.

1 Join the Units so that the connectors fit exactly.

2 The yellow sliders at the top and bottom of each Unit lock the Units together. Move the sliders toward the back of the Units as shown below until they click into place.

Precautions for Correct Use

The sliders on the tops and bottoms of the Power Supply Unit, CPU Unit, I/O Units, Special I/O Units, and CPU Bus Units must be completely locked (until they click into place) after connecting the adjacent Unit connectors.

NJ-series CPU Unit Hardware User's Manual (W500) 4-9

This illustration is provided only as a sample. It may not literally appear in this manual.

Special Information

Special information in this manual is classified as follows:

Precautions for Safe Use

Precautions on what to do and what not to do to ensure safe usage of the product.

Precautions for Correct Use

Precautions on what to do and what not to do to ensure proper operation and performance.

Additional Information

Additional information to read as required.
This information is provided to increase understanding or make operation easier.

Version Information

Information on differences in specifications and functionality for CPU Units with different unit versions and for different versions of the Sysmac Studio is given.

Note References are provided to more detailed or related information.

Precaution on Terminology

In this manual, “download” refers to transferring data from Sysmac Studio to the physical Controller and “upload” refers to transferring data from the physical Controller to Sysmac Studio.
For Sysmac Studio, synchronization is used to both upload and download data. Here, “synchronize” means to automatically compare the data for Sysmac Studio on the computer with the data in the physical Controller and transfer the data in the direction that is specified by the user.

Sections in this Manual

CONTENTS

Terms and Conditions Agreement

Warranty, Limitations of Liability

Warranties

- Exclusive Warranty

  Omron's exclusive warranty is that the Products will be free from defects in materials and workmanship for a period of twelve months from the date of sale by Omron (or such other period expressed in writing by Omron). Omron disclaims all other warranties, express or implied.

- Limitations

  OMRON MAKES NO WARRANTY OR REPRESENTATION, EXPRESS OR IMPLIED, ABOUT NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE OF THE PRODUCTS. BUYER ACKNOWLEDGES THAT IT ALONE HAS DETERMINED THAT THE PRODUCTS WILL SUITABLY MEET THE REQUIREMENTS OF THEIR INTENDED USE.

  Omron further disclaims all warranties and responsibility of any type for claims or expenses based on infringement by the Products or otherwise of any intellectual property right.

- Buyer Remedy

  Omron's sole obligation hereunder shall be, at Omron's election, to (i) replace (in the form originally shipped with Buyer responsible for labor charges for removal or replacement thereof) the non-complying Product, (ii) repair the non-complying Product, or (iii) repay or credit Buyer an amount equal to the purchase price of the non-complying Product; provided that in no event shall Omron be responsible for warranty, repair, indemnity or any other claims or expenses regarding the Products unless Omron's analysis confirms that the Products were properly handled, stored, installed and maintained and not subject to contamination, abuse, misuse or inappropriate modification. Return of any Products by Buyer must be approved in writing by Omron before shipment. Omron Companies shall not be liable for the suitability or unsuitability or the results from the use of Products in combination with any electrical or electronic components, circuits, system assemblies or any other materials or substances or environments. Any advice, recommendations or information given orally or in writing, are not to be construed as an amendment or addition to the above warranty.

See http://www.omron.com/global/ or contact your Omron representative for published information.

Limitation on Liability; Etc

OMRON COMPANIES SHALL NOT BE LIABLE FOR SPECIAL, INDIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, LOSS OF PROFITS OR PRODUCTION OR COMMERCIAL LOSS IN ANY WAY CONNECTED WITH THE PRODUCTS, WHETHER SUCH CLAIM IS BASED IN CONTRACT, WARRANTY, NEGLIGENCE OR STRICT LIABILITY.

Further, in no event shall liability of Omron Companies exceed the individual price of the Product on which liability is asserted.

Application Considerations

Suitability of Use

Omron Companies shall not be responsible for conformity with any standards, codes or regulations which apply to the combination of the Product in the Buyer's application or use of the Product. At Buyer's request, Omron will provide applicable third party certification documents identifying ratings and limitations of use which apply to the Product. This information by itself is not sufficient for a complete determination of the suitability of the Product in combination with the end product, machine, system, or other application or use. Buyer shall be solely responsible for determining appropriateness of the particular Product with respect to Buyer's application, product or system. Buyer shall take application responsibility in all cases.

NEVER USE THE PRODUCT FOR AN APPLICATION INVOLVING SERIOUS RISK TO LIFE OR PROPERTY WITHOUT ENSURING THAT THE SYSTEM AS A WHOLE HAS BEEN DESIGNED TO ADDRESS THE RISKS, AND THAT THE OMRON PRODUCT(S) IS PROPERLY RATED AND INSTALLED FOR THE INTENDED USE WITHIN THE OVERALL EQUIPMENT OR SYSTEM.

Programmable Products

Omron Companies shall not be responsible for the user's programming of a programmable Product, or any consequence thereof.

Disclaimers

Performance Data

Data presented in Omron Company websites, catalogs and other materials is provided as a guide for the user in determining suitability and does not constitute a warranty. It may represent the result of Omron's test conditions, and the user must correlate it to actual application requirements. Actual performance is subject to the Omron's Warranty and Limitations of Liability.

Change in Specifications

Product specifications and accessories may be changed at any time based on improvements and other reasons. It is our practice to change part numbers when published ratings or features are changed, or when significant construction changes are made. However, some specifications of the Product may be changed without any notice. When in doubt, special part numbers may be assigned to fix or establish key specifications for your application. Please consult with your Omron's representative at any time to confirm actual specifications of purchased Product.

Errors and Omissions

Information presented by Omron Companies has been checked and is believed to be accurate; however, no responsibility is assumed for clerical, typographical or proofreading errors or omissions.

Safety Precautions

Refer to the following manuals for safety precautions.
- NJ-series CPU Unit Hardware User’s Manual (Cat. No. W500)
- NJ/NX-series CPU Unit Software User’s Manual (Cat. No. W501)
- Sysmac Studio Version 1 Operation Manual (Cat. No. W504)

For safety precautions on NJ501-4320, please contact our sales representative and check with the product specification document or other documentation.

Definition of Precautionary Information

The following notation is used in this manual to provide precautions required to ensure safe usage of the NJ-series DB Connection Service function. The safety precautions that are provided are extremely important to safety. Always read and heed the information provided in all safety precautions.
The following notation is used.

Precautions for Safe Use
Indicates precautions on what to do and what not to do to ensure safe usage of the product.

Precautions for Correct Use
Indicates precautions on what to do and what not to do to ensure proper operation and performance.

Precautions for Safe Use

Refer to the following manuals for precautions for safe use.

- NJ-series CPU Unit Hardware User’s Manual (Cat. No. W500)
- NJ/NX-series CPU Unit Software User’s Manual (Cat. No. W501)
- Sysmac Studio Version 1 Operation Manual (Cat. No. W504)

For precautions for safe use on NJ501-4320, please contact our sales representative and check with the product specification document or other documentation.

Precautions for Correct Use

This section describes the precautions for correct use in the DB Connection Service.
Refer to the following manuals for other precautions for correct use.

- NJ-series CPU Unit Hardware User’s Manual (Cat. No. W500)
- NJ/NX-series CPU Unit Software User’s Manual (Cat. No. W501)
- Sysmac Studio Version 1 Operation Manual (Cat. No. W504)

For precautions for correct use on NJ501-4320, please contact our sales representative and check with the product specification document or other documentation.

- When the Spool function is enabled, the DB Connection Service uses the following EM Banks according to the CPU Unit model. If the EM banks are used for processes other than the DB Connection Service, the Spool data in the EM Banks will be overwritten. Do not use the EM Banks that are used by the DB Connection Service for processes other than the DB Connection Service.
  NJ501-□□20: EM Bank No. 9 to 18 (E9_00000 to E18_32767)
  NJ101-□□20: EM Bank No. 1 to 3 (E1_00000 to E3_32767)

Regulations and Standards

Conformance to EC Directives

Applicable Directives

• EMC Directives
• Low Voltage Directive

Concepts

● EMC Directive

OMRON devices that comply with EC Directives also conform to the related EMC standards so that they can be more easily built into other devices or the overall machine. The actual products have been checked for conformity to EMC standards.*
Whether the products conform to the standards in the system used by the customer, however, must be checked by the customer. EMC-related performance of the OMRON devices that comply with EC Directives will vary depending on the configuration, wiring, and other conditions of the equipment or control panel on which the OMRON devices are installed. The customer must, therefore, perform the final check to confirm that devices and the overall machine conform to EMC standards.

* Applicable EMC (Electromagnetic Compatibility) standards are as follows:
EMS (Electromagnetic Susceptibility): EN 61131-2 and EN 61000-6-2
EMI (Electromagnetic Interference): EN 61131-2 and EN 61000-6-4 (Radiated emission: 10-m regulations)

● Low Voltage Directive

Always ensure that devices operating at voltages of 50 to 1,000 VAC and 75 to 1,500 VDC meet the required safety standards. The applicable directive is EN 61131-2.

● Conformance to EC Directives

The NJ-series Controllers comply with EC Directives. To ensure that the machine or device in which the NJ-series Controller is used complies with EC Directives, the Controller must be installed as follows:

- The NJ-series Controller must be installed within a control panel.
- You must use reinforced insulation or double insulation for the DC power supplies connected to DC Power Supply Units and I/O Units.
- NJ-series Controllers that comply with EC Directives also conform to the Common Emission Standard (EN 61000-6-4). Radiated emission characteristics (10-m regulations) may vary depending on the configuration of the control panel used, other devices connected to the control panel, wiring, and other conditions.
  You must therefore confirm that the overall machine or equipment complies with EC Directives.

Conformance to KC Standards

Observe the following precaution if you use NX-series Units in Korea.

A 급 기기 (업무용 방송통신기자재)
이 기기는 업무용(A 급) 전자파적합기기로서 판매자
또는 사용자는 이 점을 주의하시기 바라며, 가정외의
지역에서 사용하는 것을 목적으로 합니다.

Class A Device (Broadcasting Communications Device for Office Use)
This device obtained EMC registration for office use (Class A), and it is intended to be used in places other than homes.
Sellers and/or users need to take note of this.

Conformance to Shipbuilding Standards

Some Database Connection CPU Units comply with shipbuilding standards. If you use a Database Connection CPU Unit that complies with shipbuilding standards and the machinery or system in which you use the Database Connection CPU Unit must also comply with the standards, consult with your OMRON representative. Application conditions are defined according to the installation location. Application may not be possible for some installation locations.

Usage Conditions for NK and LR Shipbuilding Standards

- The NJ-series Controller must be installed within a control panel.
- Gaps in the door to the control panel must be completely filled or covered with gaskets or other material.
- The following noise filter must be connected to the power supply line.

Noise Filter

Manufacturer	Model
Cosel Co., Ltd.	TAH-06-683

Software Licenses and Copyrights

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

By using this product, you will be considered as having accepted the following license conditions. If you do not accept the license conditions, do not use this product.

-MICROSOFT SOFTWARE LICENSE TERMS
REDISTRIBUTION LICENSE FOR MICROSOFT JDBC DRIVER 4.0 FOR SQL SERVER
These license terms are an agreement between Microsoft Corporation (or based on where you live, one of its affiliates) and you. Please read them. They apply to the software named above, which includes the media on which you received it, if any. The terms also apply to any Microsoft

- updates,
- supplements,

- reverse engineer, decompile or disassemble the software, except and only to the extent that applicable law expressly permits, despite this limitation;
- publish the software for others to copy;
- rent, lease or lend the software; or

TRANSFER TO A THIRD PARTY. The first user of the software may transfer it and this agreement directly to a third party. Before the transfer, that party must agree that this agreement applies to the transfer and use of the software. The first user must uninstall the software before transferring it separately from the device. The first user may not retain any copies.

EXPORT RESTRICTIONS. The software is subject to United States export laws and regulations. You must comply with all domestic and international export laws and regulations that apply to the software. These laws include restrictions on destinations, end users and end use. For additional information, see www.microsoft.com/exporting.

SUPPORT SERVICES. Because this software is “as is,” we may not provide support services for it.

ENTIRE AGREEMENT. This agreement, and the terms for supplements, updates, Internet-based services and support services that you use, are the entire agreement for the software and support services.

APPLICABLE LAW.

United States. If you acquired the software in the United States, Washington state law governs the interpretation of this agreement and applies to claims for breach of it, regardless of conflict of laws principles. The laws of the state where you live govern all other claims, including claims under state consumer protection laws, unfair competition laws, and in tort.

Outside the United States. If you acquired the software in any other country, the laws of that country apply.

LEGAL EFFECT. This agreement describes certain legal rights. You may have other rights under the laws of your country. You may also have rights with respect to the party from whom you acquired the software. This agreement does not change your rights under the laws of your country if the laws of your country do not permit it to do so.

DISCLAIMER OF WARRANTY. THE SOFTWARE IS LICENSED “AS-IS.” YOU BEAR THE RISK OF USING IT. MICROSOFT GIVES NO EXPRESS WARRANTIES, GUARANTEES OR CONDITIONS. YOU MAY HAVE ADDITIONAL CONSUMER RIGHTS UNDER YOUR LOCAL LAWS WHICH THIS AGREEMENT CANNOT CHANGE. TO THE EXTENT PERMITTED UNDER YOUR LOCAL LAWS, MICROSOFT EXCLUDES THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT.

LIMITATION ON AND EXCLUSION OF REMEDIES AND DAMAGES. YOU CAN RECOVER FROM MICROSOFT AND ITS SUPPLIERS ONLY DIRECT DAMAGES UP TO U.S. $5.00. YOU CANNOT RECOVER ANY OTHER DAMAGES, INCLUDING CONSEQUENTIAL, LOST PROFITS, SPECIAL, INDIRECT OR INCIDENTAL DAMAGES.

This limitation applies to

- anything related to the software, services, content (including code) on third party Internet sites, or third party programs, and
- claims for breach of contract, breach of warranty, guarantee or condition, strict liability, negligence, or other tort to the extent permitted by applicable law.

It also applies even if Microsoft knew or should have known about the possibility of the damages. The above limitation or exclusion may not apply to you because your country may not allow the exclusion or limitation of incidental, consequential or other damages.

Please note: As this software is distributed in Quebec, Canada, some of the clauses in this agreement are provided below in French.

Remarque : Ce logiciel étant distribué au Québec, Canada, certaines des clauses dans ce contrat sont fournies ci-dessous en français.

EXONÉRATION DE GARANTIE. Le logiciel visé par une licence est offert « tel quel ». Toute utilisation de ce logiciel est à votre seule risque et péril. Microsoft n'accorde aucune autre garantie expresse. Vous pouvez bénéficier de droits additionnels en vertu du droit local sur la protection des consommateurs, que ce contrat ne peut modifier. La ou elles sont permises par le droit locale, les garanties implicites de qualité marchande, d'adéquation à un usage particulier et d'absence de contrefaçon sont exclues.

LIMITATION DES DOMMAGES-INTÉRÊTS ET EXCLUSION DE RESPONSABILITÉ POUR LES DOMMAGES. Vous pouvez obtenir de Microsoft et de ses fournisseurs une indemnisation en cas de dommages directs uniquement à hauteur de 5,00 $ US. Vous ne pouvez prétendre à aucune indemnisation pour les autres dommages, y compris les dommages spéciaux, indirects ou accessoires et pertes de bénéfices.

Cette limitation concerne :

- tout ce qui est relié au logiciel, aux services ou au contenu (y compris le code) figurant sur des sites Internet tiers ou dans des programmes tiers ; et
- les réclamations au titre de violation de contrat ou de garantie, ou au titre de responsabilité stricte, de négligence ou d'une autre faute dans la limite autorisée par la loi en vigueur.

Elle s'applique également, même si Microsoft connaissait ou devrait connaître l'éventualité d'un tel dommage. Si votre pays n'autorise pas l'exclusion ou la limitation de responsabilité pour les dommages indirects, accessoires ou de quelque nature que ce soit, il se peut que la limitation ou l'exclusion ci-dessus ne s'appliquera pas à votre égard.

EFFET JURIDIQUE. Le présent contrat décrit certains droits juridiques. Vous pourriez avoir d'autres droits prévus par les lois de votre pays. Le présent contrat ne modifie pas les droits que vous confèrent les lois de votre pays si celles-ci ne le permettent pas.

-Oracle Technology Network Development and Distribution License Terms

Export Controls on the Programs

affiliates under common control. "You" and "your" refers to the individual or entity that wishes to use the programs from Oracle. "Programs" refers to the software product you wish to download and use and program documentation. "License" refers to your right to use the programs under the terms of this agreement. This agreement is governed by the substantive and procedural laws of California. You and Oracle agree to submit to the exclusive jurisdiction of, and venue in, the courts of San Francisco, San Mateo, or Santa Clara counties in California in any dispute arising out of or relating to this agreement.

We are willing to license the programs to you only upon the condition that you accept all of the terms contained in this agreement. Read the terms carefully and select the "Accept" button at the bottom of the page to confirm your acceptance. If you are not willing to be bound by these terms, select the "Do Not Accept" button and the registration process will not continue.

License Rights
We grant you a nonexclusive, nontransferable limited license to use the programs: (a) for purposes of developing, testing, prototyping and running applications you have developed for your own internal data processing operations; (b) to distribute the programs with applications you have developed to your customers provided that each such licensee agrees to license terms consistent with the terms of this Agreement, you do not charge your end users any additional fees for the use of the programs, and your end users may only use the programs to run your applications for their own business operations; and (c) to use the programs to provide third party demonstrations and training. You are not permitted to use the programs for any purpose other than as permitted under this Agreement. If you want to use the programs for any purpose other than as expressly permitted under this agreement you must contact us, or an Oracle reseller, to obtain the appropriate license. We may audit your use and distribution of the programs. Program documentation is either shipped with the programs, or documentation may accessed online at
http://www.oracle.com/technetwork/indexes/documentation/index.html.

Ownership and Restrictions
We retain all ownership and intellectual property rights in the programs. You may make a sufficient number of copies of the programs for the licensed use and one copy of the programs for backup purposes.

You may not:
- use the programs for any purpose other than as provided above;
- distribute the programs unless accompanied with your applications;
- charge your end users for use of the programs;
- remove or modify any program markings or any notice of our proprietary rights;
- use the programs to provide third party training on the content and/or functionality of the programs, except for training your licensed users;
- assign this agreement or give the programs, program access or an interest in the programs to any individual or entity except as provided under this agreement;
- cause or permit reverse engineering (unless required by law for interoperability), disassembly or decompilation of the programs;
- disclose results of any program benchmark tests without our prior consent.

Program Distribution

We grant you a nonexclusive, nontransferable right to copy and distribute the programs to your end users provided that you do not charge your end users for use of the programs and provided your end users may only use the programs to run your applications for their business operations. Prior to distributing the programs you shall require your end users to execute an agreement binding them to terms consistent with those contained in this section and the sections of this agreement entitled "License Rights," "Ownership and Restrictions," "Export," "Disclaimer of Warranties and Exclusive Remedies," "No Technical Support," "End of Agreement," "Relationship Between the Parties," and "Open Source." You must also include a provision stating that your end users shall have no right to distribute the programs, and a provision specifying us as a third party beneficiary of the agreement. You are responsible for obtaining these agreements with your end users.

You agree to: (a) defend and indemnify us against all claims and damages caused by your distribution of the programs in breach of this agreements and/or failure to include the required contractual provisions in your end user agreement as stated above; (b) keep executed end user agreements and records of end user information including name, address, date of distribution and identity of programs distributed; (c) allow us to inspect your end user agreements and records upon request; and, (d) enforce the terms of your end user agreements so as to effect a timely cure of any end user breach, and to notify us of any breach of the terms.

Export
You agree that U.S. export control laws and other applicable export and import laws govern your use of the programs, including technical data; additional information can be found on Oracle's Global Trade Compliance web site located at http://www.oracle.com/products/export/index.html?content.html. You agree that neither the programs nor any direct product thereof will be exported, directly, or indirectly, in violation of these laws, or will be used for any purpose prohibited by these laws including, without limitation, nuclear, chemical, or biological weapons proliferation.

Disclaimer of Warranty and Exclusive Remedies

THE PROGRAMS ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND. WE FURTHER DISCLAIM ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT.

IN NO EVENT SHALL WE BE LIABLE FOR ANY INDIRECT, INCIDENTAL, SPECIAL, PUNITIVE OR CONSEQUENTIAL DAMAGES, OR DAMAGES FOR LOSS OF PROFITS, REVENUE, DATA OR DATA USE, INCURRED BY YOU OR ANY THIRD PARTY, WHETHER IN AN ACTION IN CONTRACT OR TORT, EVEN IF WE HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. OUR ENTIRE LIABILITY FOR DAMAGES HEREUNDER SHALL IN NO EVENT EXCEED ONE THOUSAND DOLLARS (U.S. $1,000).

No Technical Support
Our technical support organization will not provide technical support, phone support, or updates to you for the programs licensed under this agreement.

Restricted Rights
If you distribute a license to the United States government, the programs, including documentation,

shall be considered commercial computer software and you will place a legend, in addition to applicable copyright notices, on the documentation, and on the media label, substantially similar to the following:

NOTICE OF RESTRICTED RIGHTS

"Programs delivered subject to the DOD FAR Supplement are 'commercial computer software' and use, duplication, and disclosure of the programs, including documentation, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement. Otherwise, programs delivered subject to the Federal Acquisition Regulations are 'restricted computer software' and use, duplication, and disclosure of the programs, including documentation, shall be subject to the restrictions in FAR 52.227-19, Commercial Computer Software-Restricted Rights (June 1987). Oracle America, Inc., 500 Oracle Parkway, Redwood City, CA 94065."

End of Agreement
You may terminate this agreement by destroying all copies of the programs. We have the right to terminate your right to use the programs if you fail to comply with any of the terms of this agreement, in which case you shall destroy all copies of the programs.

Relationship Between the Parties
The relationship between you and us is that of licensee/licensor. Neither party will represent that it has any authority to assume or create any obligation, express or implied, on behalf of the other party, nor to represent the other party as agent, employee, franchisee, or in any other capacity. Nothing in this agreement shall be construed to limit either party's right to independently develop or distribute software that is functionally similar to the other party's products, so long as proprietary information of the other party is not included in such software.

Open Source
"Open Source" software - software available without charge for use, modification and distribution - is often licensed under terms that require the user to make the user's modifications to the Open Source software or any software that the user 'combines' with the Open Source software freely available in source code form. If you use Open Source software in conjunction with the programs, you must ensure that your use does not: (i) create, or purport to create, obligations of us with respect to the Oracle programs; or (ii) grant, or purport to grant, to any third party any rights to or immunities under our intellectual property or proprietary rights in the Oracle programs. For example, you may not develop a software program using an Oracle program and an Open Source program where such use results in a program file(s) that contains code from both the Oracle program and the Open Source program (including without limitation libraries) if the Open Source program is licensed under a license that requires any "modifications" be made freely available. You also may not combine the Oracle program with programs licensed under the GNU General Public License ("GPL") in any manner that could cause, or could be interpreted or asserted to cause, the Oracle program or any modifications thereto to become subject to the terms of the GPL.

Entire Agreement
You agree that this agreement is the complete agreement for the programs and licenses, and this agreement supersedes all prior or contemporaneous agreements or representations. If any term of this agreement is found to be invalid or unenforceable, the remaining provisions will remain effective.

Last updated: 01/24/09

Should you have any questions concerning this License Agreement, or if you desire to contact Oracle for any reason, please write:
Oracle America, Inc.
500 Oracle Parkway,
Redwood City, CA 94065

Oracle may contact you to ask if you had a satisfactory experience installing and using this OTN software download.

The DB Connection Service uses the following libraries protected by GNU GPL and LGPL.
• jtds-1.3.0.jar
• mariadb-java-client-1.1.5.jar
• jaybird-full-2.2.3.jar

Precautions for Use
- Decompilation, disassembly, or other reverse engineering to analyze this software is prohibited.
- Copy or reproduction of all or any part of this software is prohibited.
- Regardless of media or means, redistribution of copies of this software is prohibited in any cases.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.
All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met.　This License explicitly affirms your unlimited permission to run the unmodified Program.　The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work.　This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force.　You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright.　Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.
No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.
You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.
You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.
You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.
"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

d) Limiting the use for publicity purposes of names of licensors or authors of the material; or

e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.
You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License.　If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.
You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance.　However, nothing other than this License grants you permission to propagate or modify any covered work.　These actions infringe copyright if you do not accept this License.　Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.
Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License.　You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations.　If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License.　For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.
A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based.   The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version.   For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement).   To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients.   "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License.   You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or

use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

GNU LESSER GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License").
Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) The modified work must itself be a software library.

b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.

c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.

d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License.
Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work

under the terms of Section 6.
Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b) Use a suitable shared library mechanism for linking with the Library.
A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system.　Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities.　This must be distributed under the terms of the Sections above.

b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License.　Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it.　However, nothing else grants you permission to modify or distribute the Library or its derivative works.　These actions are prohibited by law if you do not accept this License.　Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions.　You may not impose any further restrictions on the recipients' exercise of the rights granted herein.　You are not responsible for enforcing compliance by third parties with this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License.　If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all.　For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR

- a) under this License, provided that you make a good faith effort to ensure that, in the event an Application does not supply the function or data, the facility still operates, and performs whatever part of its purpose remains meaningful, or
- b) under the GNU GPL, with none of the additional permissions of this License applicable to that copy.

3. Object Code Incorporating Material from Library Header Files.
The object code form of an Application may incorporate material from a header file that is part of the Library. You may convey such object code under terms of your choice, provided that, if the incorporated material is not limited to numerical parameters, data structure layouts and accessors, or small macros, inline functions and templates (ten or fewer lines in length), you do both of the following:

- a) Give prominent notice with each copy of the object code that the Library is used in it and that the Library and its use are covered by this License.
- b) Accompany the object code with a copy of the GNU GPL and this license document.

4. Combined Works.
You may convey a Combined Work under terms of your choice that, taken together, effectively do not restrict modification of the portions of the Library contained in the Combined Work and reverse engineering for debugging such modifications, if you also do each of the following:

- a) Give prominent notice with each copy of the Combined Work that the Library is used in it and that the Library and its use are covered by this License.
- b) Accompany the Combined Work with a copy of the GNU GPL and this license document.
- c) For a Combined Work that displays copyright notices during execution, include the copyright notice for the Library among these notices, as well as a reference directing the user to the copies of the GNU GPL and this license document.
- d) Do one of the following:
    - 0) Convey the Minimal Corresponding Source under the terms of this License, and the Corresponding Application Code in a form suitable for, and under terms that permit, the user to recombine or relink the Application with a modified version of the Linked Version to produce a modified Combined Work, in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source.
    - 1) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (a) uses at run time a copy of the Library already present on the user's computer system, and (b) will operate properly with a modified version of the Library that is interface-compatible with the Linked Version.
- e) Provide Installation Information, but only if you would otherwise be required to provide such information under section 6 of the GNU GPL, and only to the extent that such information is necessary to install and execute a modified version of the Combined Work produced by recombining or relinking the Application with a modified version of the Linked Version. (If you use option 4d0, the Installation Information must accompany the Minimal Corresponding Source and Corresponding Application Code. If you use option 4d1, you must provide the Installation Information in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source.)

Versions

Unit Versions

The hardware and software versions are managed in each NJ-series Unit. You can check the versions on the ID information indication on the Unit or with Sysmac Studio.

Version Types

There are two types of versions. One is unit version and the other is DB Connection Service version. These versions are managed independently. Therefore, only one of them may be upgraded.

● Unit Version

This is the version of hardware and software of Units and EtherCAT slaves. The version is upgraded at every specification change in the hardware or software. Therefore, the functionality and performance differ by the versions even in the same model number of Units and EtherCAT slaves.

● DB Connection Service Version

This is the version of DB Connection Service implemented in the Database Connection CPU Units. The version is upgraded at every specification change in the DB Connection Service.

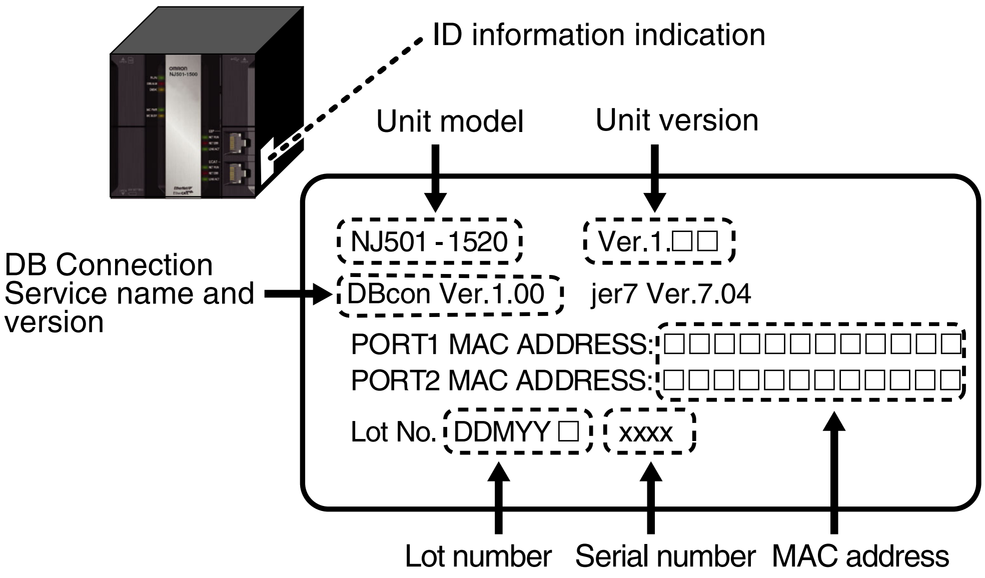
Checking Versions

You can check versions on the ID information indications or with Sysmac Studio.

Checking Unit Versions on ID Information Label

The unit version is given on the ID information indication on the side of the product.

The ID information on an NJ-series NJ501-1520 CPU Unit is shown below.

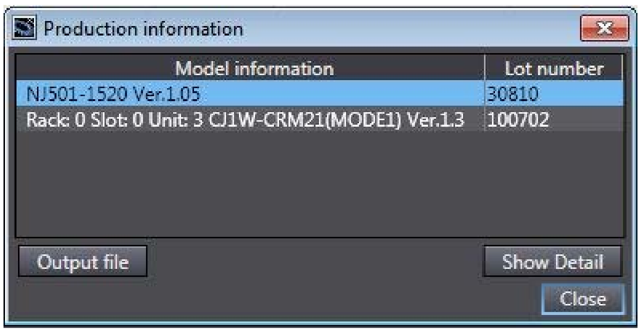

ID information indication
Unit model
Unit version
DB Connection Service name and version
NJ501 - 1520
Ver.1.□□
DBcon Ver.1.00
jer7 Ver.7.04
PORT1 MAC ADDRESS:□□□□□□□□□□□□
PORT2 MAC ADDRESS:□□□□□□□□□□□□
Lot No. DDMYY□ xxxx
Lot number
Serial number
MAC address

Checking Unit Versions with Sysmac Studio

You can use the Unit Production Information on Sysmac Studio while online to check the unit version of the CPU Unit, CJ-series Special I/O Units, and CJ-series CPU Bus Units. The unit versions of CJ-series Basic I/O Units cannot be checked from Sysmac Studio.

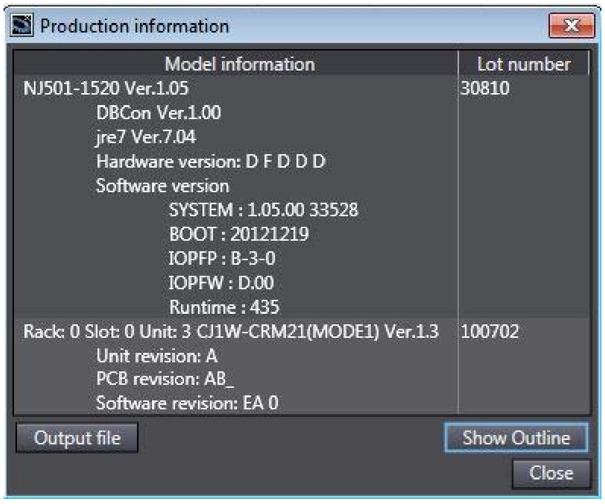
● CPU Unit and CJ-series Units

1. Double-click **CPU/Expansion Racks** under **Configurations and Setup** in the Multiview Explorer. Or, right-click **CPU/Expansion Racks** under **Configurations and Setup** and select ***Edit*** from the menu.
   The Unit Editor is displayed in the Configurations and Setup layer of the Edit Pane.
2. Right-click any open space in the Unit Editor and select ***Production Information*** from the menu.
   The Production Information Dialog Box is displayed.
3. Click the **Show Outline** Button or the **Show Detail** Button on the lower right part of the Production Information Dialog Box.
   The view will change between the production information details and outline.



Model information	Lot number
NJ501-1520 Ver.1.05	30810
Rack: 0 Slot: 0 Unit: 3 CJ1W-CRM21(MODE1) Ver.1.3	100702

Outline View



Model information	Lot number
NJ501-1520 Ver.1.05	30810
DBCon Ver.1.00	
jre7 Ver.7.04	
Hardware version: D F D D D	
Software version	
SYSTEM : 1.05.00 33528	
BOOT : 20121219	
IOPFP : B-3-0	
IOPFW : D.00	
Runtime : 435	
Rack: 0 Slot: 0 Unit: 3 CJ1W-CRM21(MODE1) Ver.1.3	100702
Unit revision: A	
PCB revision: AB_	
Software revision: EA 0	

Detail View

The information that is displayed is different for the Outline View and Detail View. The Detail View displays both the unit versions and DB Connection Service version. The Outline View displays only the unit versions.

Unit Versions and Sysmac Studio Versions

The functions that are supported depend on the unit version of the NJ-series CPU Unit. The version of Sysmac Studio that supports the functions that were added for an upgrade is also required to use those functions. Refer to *B-4 Version Information* for the relationship between the unit versions of the NJ-series Database Connection CPU Units and the Sysmac Studio versions, and for the functions that are supported by each unit version.

Related Manuals

The following manuals are related to this manual. Use these manuals for reference.

Manual name	Cat. No.	Model numbers	Application	Description
NJ-series CPU Unit Hardware User's Manual	W500	NJ501-□□□□ NJ301-□□□□ NJ101-□□□□	Learning the basic specifications of the NJ-series CPU Units, including introductory information, designing, installation, and maintenance. Mainly hardware information is provided.	An introduction to the entire NJ-series system is provided along with the following information on the CPU Unit. • Features and system configuration • Introduction • Part names and functions • General specifications • Installation and wiring • Maintenance and inspection Use this manual together with the *NJ/NX-series CPU Unit Software User's Manual* (Cat. No. W501).
NJ/NX-series CPU Unit Software User's Manual	W501	NJ701-□□□□ NJ501-□□□□ NJ301-□□□□ NJ101-□□□□	Learning how to program and set up an NJ/NX-series CPU Unit. Mainly software information is provided.	The following information is provided on a Controller built with an NJ/NX-series CPU Unit. • CPU Unit operation • CPU Unit features • Initial settings • Programming based on IEC 61131-3 language specifications Use this manual together with the *NJ-series CPU Unit Hardware User's Manual* (Cat. No. W500).
NJ/NX-series Instructions Reference Manual	W502	NJ701-□□□□ NJ501-□□□□ NJ301-□□□□ NJ101-□□□□	Learning detailed specifications on the basic instructions of an NJ/NX-series CPU Unit.	The instructions in the instruction set (IEC 61131-3 specifications) are described. When programming, use this manual together with the *NJ-series CPU Unit Hardware User's Manual* (Cat. No. W500) and *NJ/NX-series CPU Unit Software User's Manual* (Cat. No. W501).
NJ/NX-series CPU Unit Motion Control User's Manual	W507	NJ701-□□□□ NJ501-□□□□ NJ301-□□□□ NJ101-□□□□	Learning about motion control settings and programming concepts.	The settings and operation of the CPU Unit and programming concepts for motion control are described. Use this manual together with the *NJ-series CPU Unit Hardware User's Manual* (Cat. No. W500) and *NJ/NX-series CPU Unit Software User's Manual* (Cat. No. W501).
NJ/NX-series Motion Control Instructions Reference Manual	W508	NJ701-□□□□ NJ501-□□□□ NJ301-□□□□ NJ101-□□□□	Learning about the specifications of the motion control instructions that are provided by OMRON.	The motion control instructions are described. When programming, use this manual together with the *NJ-series CPU Unit Hardware User's Manual* (Cat. No. W500), *NJ/NX-series CPU Unit Software User's Manual* (Cat. No. W501), and *NJ/NX-series CPU Unit Motion Control User's Manual* (Cat. No. W507).
NJ/NX-series CPU Unit Built-in EtherCAT® Port User's Manual	W502	NJ701-□□□□ NJ501-□□□□ NJ301-□□□□ NJ101-□□□□	Using the built-in EtherCAT port on an NJ/NX-series CPU Unit.	Information on the built-in EtherCAT port is provided. This manual provides an introduction and provides information on the configuration, features, and setup. Use this manual together with the *NJ-series CPU Unit Hardware User's Manual* (Cat. No. W500) and *NJ/NX-series CPU Unit Software User's Manual* (Cat. No. W501).
NJ/NX-series CPU Unit Built-in EtherNet/IP™ Port User's Manual	W506	NJ701-□□□□ NJ501-□□□□ NJ301-□□□□ NJ101-□□□□	Using the built-in EtherNet/IP port on an NJ/NX-series CPU Unit.	Information on the built-in EtherNet/IP port is provided. Information is provided on the basic setup, tag data links, and other features. Use this manual together with the *NJ-series CPU Unit Hardware User's Manual* (Cat. No. W500) and *NJ/NX-series CPU Unit Software User's Manual* (Cat. No. W501).
NJ-series Database Connection CPU Units User's Manual	W527	NJ501-□□20 NJ101-□□20	Learning about the functions and application procedures of the NJ-series DB Connection function.	Describes the functions and application procedures of the NJ-series DB Connection function.

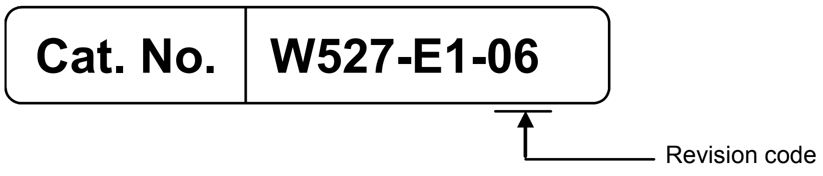
Manual name	Cat. No.	Model numbers	Application	Description
NJ/NX-series Troubleshooting Manual	W503	NJ701-□□□□ NJ501-□□□□ NJ301-□□□□ NJ101-□□□□	Learning about the errors that may be detected in an NJ/NX-series Controller.	Concepts on managing errors that may be detected in an NJ/NX-series Controller and information on individual errors are described. Use this manual together with the *NJ-series CPU Unit Hardware User's Manual* (Cat. No. W500) and *NJ/NX-series CPU Unit Software User's Manual* (Cat. No. W501).
Sysmac Studio Version 1 Operation Manual	W504	SYSMAC -SE2□□□	Learning about the operating procedures and functions of Sysmac Studio.	Describes the operating procedures of Sysmac Studio.

Terminology

Term	Description
Column	One of the information layers of each DB. Refers to the columns of each table.
DB	Refers to a database in a server.
DB Connection	Refers to a virtual communication path established between CPU Unit and DB.
DB Connection function	Used to connect a CPU Unit to a DB. This function operates on a CPU Unit.
DB Connection Instruction	Refers to special instructions for the DB Connection Service.
DB Connection Service	This service provides the DB Connection function to connect a CPU Unit to a DB. In the ID information indication on the side of the CPU Unit and in Sysmac Studio, this service is indicated as “DBCon”.
DB Connection Service shutdown function	Used to shut down the DB Connection Service after automatically saving the Operation Log files into the SD Memory Card.
DB mapping	Means to assign each member of a DB Map Variable to the corresponding column of a table in the connected DB.
DB Map Variable	Refers to a variable that uses a structure data type for DB access as its data type.
Debug Log	One of the Operation Logs. This log is used for recording which SQL statements are executed, and parameters and execution result of each SQL statements.
EM Area	Refers to EM Area of the memory for CJ-series Units. The data in this area are retained even if the power supply to the CPU Unit is cycled (i.e. ON → OFF → ON) or the operating mode of the CPU Unit is changed (i.e. PROGRAM mode ⇔ RUN mode).
Execution Log	One of the Operation Logs. This log is used to record the executions of the DB Connection Service.
Operation Log	Used to trace the operations of the DB Connection function on the CPU Unit. There are three types of Operation Logs; Execution Log, Debug Log, and SQL Execution Failure Log.
Run mode of the DB Connection Service	Used to switch whether to actually access the DB or to normally end the instructions without accessing the DB when DB Connection Instructions are executed.
Spool memory	Refers to the memory area for storing the SQL statements in the Spool function.
Spool function	Used to store some SQL statements for inserting records into the DB or updating the records in the DB that could not be executed due to a network failure.
Spool data	Refers to the SQL statements stored in the Spool memory.
Structure data type for DB access	Refers to structure data type where all or some of the columns of a specified table are registered as structure members.
SQL	Stands for Structured Query Language, which is one of the languages for DB processing such as data read/write.
SQL Execution Failure Log	One of the Operation Logs. This log is used to record execution failure of SQL statements in the DB.
SQL statement	Refers to the statements that show a specific instruction used for DB operations such as data read/write.
Table	One of the information layers of each DB, which contains data.

Revision History

A manual revision code appears as a suffix to the catalog number on the front and back covers of the manual.



Revision code	Date	Revised content
01	April 2013	Original production
02	August 2013	• Added description of the time specified for timeout of DB Connection Instructions. P5-10, A-16, A-21, A-37, and A-41 • Corrected mistakes.
03	February 2014	Added description of the functions supported by the DB Connection Service version 1.01 or higher.
04	July 2014	• Added NJ501-4320. • Corrected mistakes.
05	November 2015	• Added NJ101-□□20. • Corrected mistakes.
06	December 2015	• Added description of the functions supported by the DB Connection Service version 1.02 or higher. • Corrected mistakes.

1 Introduction to the DB Connection Service

This section provides an introduction to the DB Connection Service.

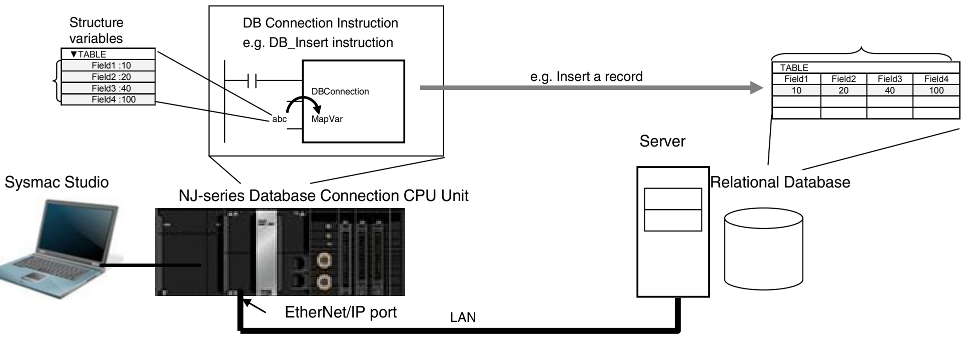
1-1 Overview and Features

This section describes the overview and features of the DB Connection Service.

1-1-1 Overview

The SYSMAC NJ-series Controllers are next-generation machine automation controllers that provide the functionality and high-speed performance that are required for machine control. They provide the safety, reliability, and maintainability that are required of industrial controllers.
The NJ-series Controllers provide the functionality of previous OMRON PLCs, and they also provide the functionality that is required for motion control. Synchronized control of I/O devices on high-speed EtherCAT can be applied to safety devices, vision systems, motion equipment, discrete I/O, and more.
OMRON offers the new Sysmac Series of control devices designed with unified communications specifications and user interface specifications. The NJ-series Machine Automation Controllers are part of the Sysmac Series. You can use them together with EtherCAT slaves, other Sysmac products, and the Sysmac Studio Automation Software to achieve optimum functionality and ease of operation. With a system that is created from Sysmac products, you can connect components and operate the system through unified concepts and usability.

The DB Connection Service is a function to insert, update, retrieve, and delete records to/from a relational database (hereinafter called “DB”) on a server connected to the built-in EtherNet/IP port of an NJ-series CPU Unit by executing special instructions (called “DB Connection Instruction”) on the NJ-series CPU Unit.
The DB Connection Service is available with the NJ-series NJ501-□□20 and NJ101-□□20 CPU Units.



- Oracle Database of Oracle Corporation, SQL Server of Microsoft Corporation, DB2 for Linux, UNIX and Windows of IBM Corporation, MySQL of Oracle Corporation, Firebird of Firebird Foundation Incorporated, and PostgreSQL of PostgreSQL Global Development Group are supported.[*1]
- An NJ501-□□20 CPU Unit can access up to three databases on up to three servers.[*2] It is possible to access more than one database in one or more servers. You can realize flexible operations such as switching the database to access according to the specified data and SQL operations (such as INSERT/SELECT) and connecting to another database in a different server when a database cannot be connected, for example, due to a server problem.

*1 The connectable databases are different between NJ501-1□20/NJ101-□□20 and NJ501-4320. Refer to *1-2-1 DB Connection Service Specifications* for the connectable databases.

*2 An NJ101-□□20 can access only one database.

1-1-2 Features

No Special Unit, Tool, nor Middleware Required

- No special Unit is required for the DB Connection function. You can use the NJ-series CPU Units.
- No special tool is required for the DB Connection function. You can use Sysmac Studio.
- The server does not need any special middleware for connection to the NJ-series CPU Units.

Easy Access to the DB

- The SQL operations such as INSERT and SELECT can be easily executed.
- No special knowledge of SQL statements is required.
- Variables for DB access can be defined just by creating a structure for the table that you want to access.
- You can easily control the execution timing and prepare the write values because the SQL operations can be executed by special instructions.

Recording of Operation Logs

- You can save the execution result logs of special instructions and processing (i.e. internal SQL statements) as a log file into the SD Memory Card mounted in the CPU Unit. Also, you can check the logs using Sysmac Studio or FTP client software.*

* For saving the log files, an SD Memory Card is provided with each Database Connection CPU Unit. The SD Memory Card can be also used for any purposes other than DB Connection functions such as reading from and writing to the files in the SD Memory Card using instructions.

Fail-safe Design against Errors and Power Interruption

- You can spool the data (i.e. internal SQL statements) if the data cannot be sent due to an information exchange error with the DB, and execute the processing when the communications are recovered from the failure.
- You can automatically save the Operation Logs by shutting down the DB Connection Service when turning OFF the power supply to the CPU Unit.

Making a Library of DB Access Function

You can provide and reuse the special instructions as a library file by describing each special instruction as a user-defined function block.

1-2 DB Connection Service Specifications and System

This section describes the specifications and system of the DB Connection Service.

1-2-1 DB Connection Service Specifications

This section describes the specifications of the DB Connection Service. Refer to *B-3 Specifications* for the general specifications, performance specifications, and function specifications of the Database Connection CPU Units.
Refer to *B-4 Version Information* for the information on version upgrades of the DB Connection Service.

Item		Description
CPU Unit model		Special models[*1]. The other functions are same as the NJ501-□□00 or NJ101-□□00 CPU Units. ・NJ501-1520: 64-axis type ・NJ501-1420: 32-axis type ・NJ501-1320: 16-axis type ・NJ501-4320: 16-axis type ・NJ101-1020: 2-axis type ・NJ101-9020: No-axis type
Supported DB		・NJ501-1□20/NJ101-□□20 Microsoft Corporation: SQL Server 2008/2008 R2/2012/2014 Oracle Corporation: Oracle Database 10g /11g/12c International Business Machines Corporation (IBM): DB2 for Linux, UNIX and Windows 9.5/9.7/10.1/10.5 Oracle Corporation: MySQL Community Edition 5.1/5.5/5.6[*2] Firebird Foundation Incorporated: Firebird 2.1/2.5 PostgreSQL Global Development Group: PostgreSQL 9.2/9.3/9.4[*3] ・NJ501-4320 Microsoft Corporation: SQL Server 2008/2008 R2/2012/2014 Oracle Corporation: Oracle Database 10g/11g/12c Oracle Corporation: MySQL Community Edition 5.1/5.5/5.6[*2]
Number of DB Connections (Number of databases that can be connected at the same time)		・NJ501-□□20: 3 connections max.[*4] ・NJ101-□□20: 1 connection max.
Instruction	Supported operations	The following operations can be performed by executing DB Connection Instructions in the NJ-series CPU Units. Inserting records (INSERT), Updating records (UPDATE), Retrieving records (SELECT), and Deleting records (DELETE)
	Number of columns in an INSERT operation	SQL Server: 1,024 columns max Oracle Database: 1,000 columns max. DB2: 1,000 columns max. MySQL: 1,000 columns max. Firebird: 1,000 columns max. PostgreSQL: 1,000 columns max.

Item		Description
	Number of columns in an UPDATE operation	SQL Server: 1,024 columns max. Oracle Database: 1,000 columns max. DB2: 1,000 columns max. MySQL: 1,000 columns max. Firebird: 1,000 columns max. PostgreSQL: 1,000 columns max.
	Number of columns in a SELECT operation	SQL Server: 1,024 columns max. Oracle Database: 1,000 columns max. DB2: 1,000 columns max. MySQL: 1,000 columns max. Firebird: 1,000 columns max. PostgreSQL: 1,000 columns max.
	Number of records in the output of a SELECT operation	65,535 elements max., 4 MB max.
	Number of DB Map Variables for which a mapping can be created	・NJ501-1□20 SQL Server: 60 variables max. Oracle Database: 30 variables max. DB2: 30 variables max. MySQL: 30 variables max. Firebird: 15 variables max. PostgreSQL: 30 variables max. Even if the number of DB Map Variables has not reached the upper limit, the total number of members of structures used as data type of DB Map Variables is 10,000 members max. ・NJ501-4320 SQL Server: 15 variables max. Oracle Database: 15 variables max. MySQL: 15 variables max. Even if the number of DB Map Variables has not reached the upper limit, the total number of members of structures used as data type of DB Map Variables is 10,000 members max. ・NJ101-□□20 SQL Server: 15 variables max. Oracle Database: 15 variables max. DB2: 15 variables max. MySQL: 15 variables max. Firebird: 15 variables max. PostgreSQL: 15 variables max. Even if the number of DB Map Variables has not reached the upper limit, the total number of members of structures used as data type of DB Map Variables is 10,000 members max.
Run mode of the DB Connection Service		Operation Mode or Test Mode ・Operation Mode: When each instruction is executed, the service actually accesses the DB. ・Test Mode: When each instruction is executed, the service ends the instruction normally without accessing the DB actually.
Spool function		Used to store SQL statements when an error occurred and resend the statements when the communications are recovered from the error. [*5] NJ501-□□20: 1 MB NJ101-□□20: 192 KB

Item	Description
Operation Log function	The following three types of logs can be recorded. ・Execution Log: Log for tracing the executions of the DB Connection Service. ・Debug Log: Detailed log for SQL statement executions of the DB Connection Service. ・SQL Execution Failure Log: Log for execution failures of SQL statements in the DB.
DB Connection Service shutdown function	Used to shut down the DB Connection Service after automatically saving the Operation Log files into the SD Memory Card.

*1 The CIP (Common Industrial Protocol) communications using the built-in EtherNet/IP port support the same functions as with the following CPU models. Therefore, when executing the EtherNet/IP tag data link function, please specify the following CPU models on Network Configurator. The following models are also displayed in Sysmac Gateway or CX-Compolet.
- NJ501-1500 for NJ501-1520
- NJ501-1400 for NJ501-1420
- NJ501-1300 for NJ501-1320
- NJ501-4300 for NJ501-4320
- NJ101 for NJ101-□□20

*2 The supported storage engines of the DB are InnoDB and MyISAM.

*3 When you connect the CPU Unit to PostgreSQL, make the following setting to set the locale of the PostgreSQL to C. Otherwise, the error messages are not correctly displayed.
Change the value of lc_messages in the postgresql.comf file stored in the data folder under the installation folder of PostgreSQL and restart the PostgreSQL.
lc_messages = 'C'

*4 When two or more DB Connections are established, the operation cannot be guaranteed if you set different database types for the connections.

*5 Refer to *5-1-9 How to Calculate the Number of SQL Statements that Can be Spooled* for the information.

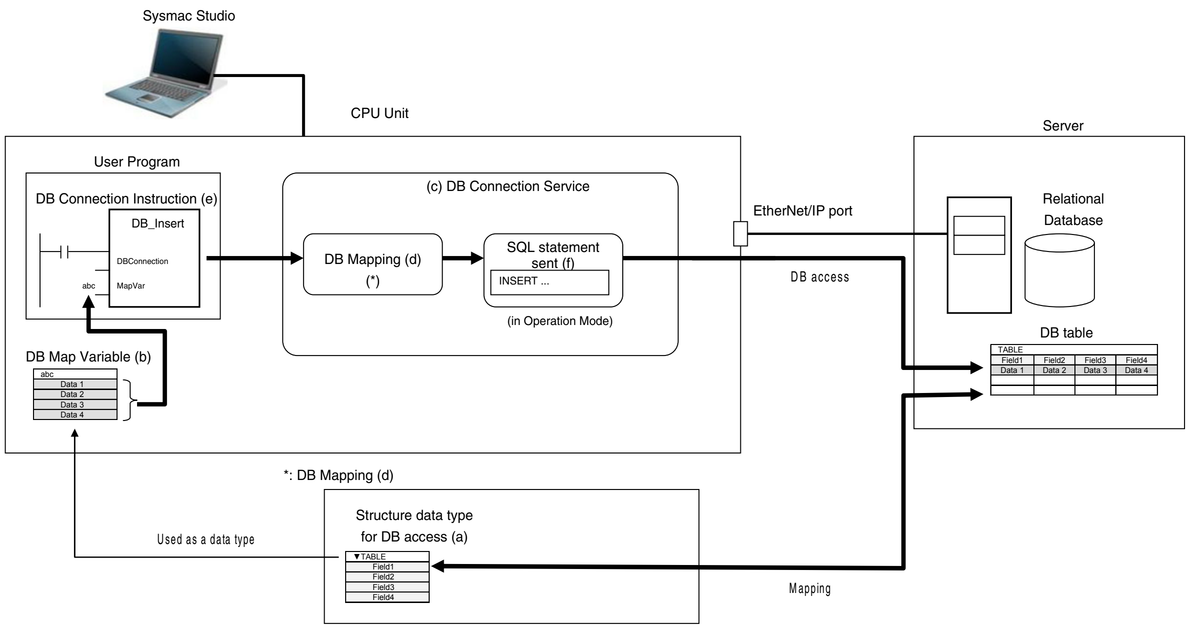
1-2-2 DB Connection System

This section describes the basic and other systems of the DB Connection function.

Refer to *1-3 Operation Flow of the DB Connection Service* for the operation flow.

Basic System

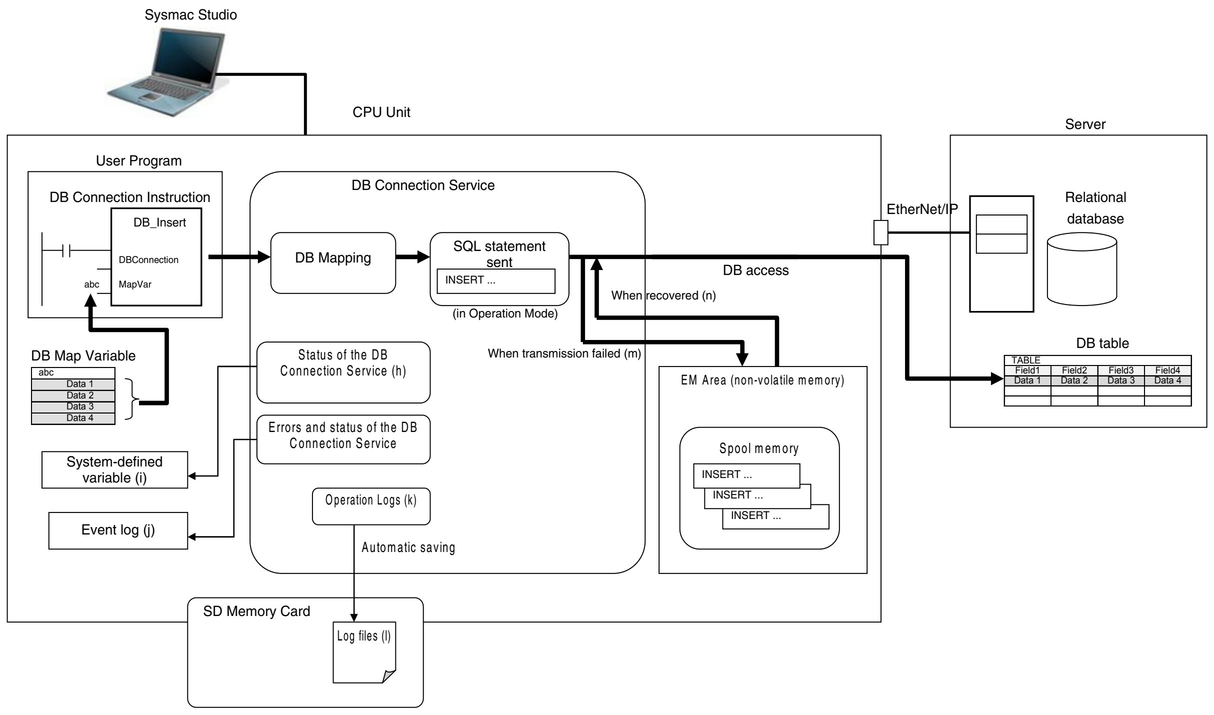
The following figure shows the basic system of the DB Connection function.



Basic System (The numbers show the processing order.)	Reference
1. Create a structure for NJ-series Controller that matches the column names in the DB table. ((a) in the above figure) Section 3-2-2 will help you match the data types between the NJ-series Controllers and database.	Refer to *3-2 Creating a Structure Data Type*.
2. Create a variable called "DB Map Variable" using the structure created in Step 1. ((b) in the above figure)	Refer to *3-3 Creating a DB Map Variable*.
3. Start the DB Connection Service. ((c) in the above figure) Specify the Run mode of the DB Connection Service according to the following conditions. ・ When the DB is connected: Select the Operation Mode. ・ When the DB does not exist or not connected: Select the Test Mode.	Refer to *4-1 Run Mode of the DB Connection Service and Start/Stop Procedures*.
4. Use a DB_Connect instruction to establish a DB Connection. This checks the IP address or name of the server and log on credentials.	Refer to *4-2 Establishing/Closing a DB Connection*.
5.Use a DB_CreateMapping instruction to connect to a table using the DB Map Variable and apply the mapping. (called "DB mapping"). ((d) in the above figure)	Refer to *3-4 Specifying the Table and Apply the Mapping*.
6. Use DB_Insert, DB_Update, and DB_Select instructions to execute the insert, update, and retrieve record processing. ((e) in the above figure) When the DB Connection Service is set to the Operation Mode, the SQL statements are sent. ((f) in the above figure)	Refer to *3-5 Programming and Transfer*.

Other Systems

The following figure shows the other systems of the DB Connection function.



Other Systems	Reference
・You can check the status of the DB Connection Service and each DB Connection ((h) in the above figure) with the DB_GetServiceStatus (Get DB Connection Service Status) instruction, DB_GetConnectionStatus (Get DB Connection Status) instruction, or a system-defined variable ((i) in the above figure).	Refer to *4 Basic Operations and Status Check*.
・Errors and status of the DB Connection Service are stored as an event log. ((j) in the above figure)	Refer to *7 Troubleshooting*.
・The logs of tracing the operations of the DB Connection Service on the CPU Unit (called “Operation Logs”) ((k) in the above figure) are saved as a log file ((l) in the above figure) into the SD Memory Card mounted in the CPU Unit.	Refer to *6 How to Use Operation Logs*.
・When transmission of an SQL statement failed, the SQL statement is automatically saved into the EM Area. ((m) in the above figure) When the communications are recovered, the stored SQL statement is resent automatically or by executing an instruction. ((n) in the above figure)	Refer to *5-1 Spool Function*.

1-3 Operation Flow of the DB Connection Service

This section gives the basic operation flow.

The DB Connection Service is basically used according to the following flow.

STEP 1 Starting Sysmac Studio	Refer to *2-1 Starting Sysmac Studio and Creating a New Project*.

⬇

STEP 2 Creating a New Project	Refer to *2-1 Starting Sysmac Studio and Creating a New Project*.

⬇

STEP 3 Making the DB Connection Settings	Refer to *2-2 DB Connection Settings*.

Make a setting for the entire DB Connection Service and each DB Connection. Also, perform a communications test between Sysmac Studio and the DB as necessary.

1. Setting of the entire DB Connection Service:
Double-click **DB Connection Service Settings** under **Configurations and Setup** - **Host Connection Settings** - **DB Connection** in the Multiview Explorer and set the following in the Service Settings.
Service Start, Execution Log, Debug Log, and SQL Execution Failure Log settings

2. Setting of each DB Connection:
Right-click **DB Connection Settings** under **Configurations and Setup** - **Host Connection Settings** - **DB Connection** in the Multiview Explorer and add up to three DB Connections for NJ501-□□20 or one DB Connection for NJ101-□□20. Then, set the following for each DB Connection.

- Database type
- IP address (IP address of the server)
- Database name (Database name in the server)
- User name, password, etc.
- Spool settings

3. Communications test from Sysmac Studio to the DB (only when necessary):
Double-click a DB Connection under **Configurations and Setup** - **Host Connection Settings** - **DB Connection** - **DB Connection Settings** and click the **Communications Test** Button under the DB Communications Test in the Connection Settings.

STEP 4 Creating a Structure for DB Access	Refer to *3-2 Creating a Structure Data Type*.

Create a structure data type for DB access. The structure members must satisfy the following conditions.

- Member names are the same as corresponding column name of the table to access.
- Members' data types match the data type of corresponding column of the table to access.

STEP 5 Creating a Variable Using above Structure	Refer to *3-3 Creating a DB Map Variable*.

Create a variable called "DB Map Variable" using the structure data type created in STEP 4.

STEP 6 Programming using DB Connection Instructions	Refer to *3-4 Specifying the Table and Apply the Mapping* and *3-5 Programming and Transfer*.

1. Initial Processing

1-1. Write a DB_ControlService (Control DB Connection Service) instruction.
(This instruction is not required if you set the DB Connection Service to auto start in the DB Connection Settings.)

1-2. Write a DB_Connect (Establish DB Connection) instruction.

1-3. Write a DB_CreateMapping (Create DB Map) instruction.
The DB Map Variable is mapped with the columns of the table to access and registered as a variable subject to the record processing.

2. Processing during Operation [*1]

2-1. Write DB_Insert (Insert DB Record), DB_Update (Update DB Record), DB_Select (Retrieve DB Record), and other instructions.

3. End Processing

3-1. Write a DB_Close (Close DB Connection) instruction.

4. Power OFF Processing[*2]

4-1. Write a DB_Shutdown (Shutdown DB Connection Service) instruction.

*1 When you continuously execute DB_Insert (Insert DB Record), DB_Update (Update DB Record), DB_Select (Retrieve DB Record), and other instructions, repeat the *2. Processing during Operation*.

*2 Be sure to execute a DB_Shutdown (Shutdown DB Connection Service) instruction before you turn OFF the power supply to the system. If the power supply is turned OFF without executing a DB_Shutdown (Shutdown DB Connection Service) instruction, the Operation Log file may be corrupted or its contents may be lost.

STEP 7 Transferring a Project to the CPU Unit	Refer to *3-5 Programming and Transfer*.

STEP 8 Starting the DB Connection Service	Refer to *4 Basic Operations and Status Check*.

Use any of the following methods to start the DB Connection Service.

- Automatically start the service when the operating mode of the CPU Unit is changed from PROGRAM mode to RUN mode.
- Right-click **DB Connection Service Settings** under **Configurations and Setup** - **Host Connection Settings** - **DB Connection** in the Multiview Explorer and select ***Online Settings*** from the menu. Then, click the **Start (Test Mode)** or **Start (Operation Mode)** Button.
- Execute a DB_ControlService (Control DB Connection Service) instruction.

Specify the following Run mode when starting the DB Connection Service.

- When the specified DB does not exist in the server or when the DB exists but not connected: Specify the Test Mode.
- When the specified DB is connected: Specify the Operation Mode.

STEP 9 Executing DB Connection Instructions	Refer to *3-5-2 DB Connection Instruction Set* and *Appendix DB Connection Instructions*.

Confirm that the operation status of the DB Connection Service is *Running* with the *_DBC_Status.Run* system-defined variable (Running flag of the DB Connection Service) and then execute the DB Connection Instructions.

STEP 10 Debugging the DB Connection Instructions	Refer to *3-6 Debugging in Design, Startup, and Operation Phases.*

STEP 11 Checking the Status with Sysmac Studio	Refer to *4 Basic Operations and Status Check*.

You can check the status of the entire DB Connection Service and the connection status of each DB Connection.

- Status of the entire DB Connection Service:
  Right-click **DB Connection Service Settings** under **Configurations and Setup** - **Host Connection Settings** - **DB Connection** in the Multiview Explorer and select ***Monitor DB Connection Service*** from the menu. Then, check the status of the entire DB Connection Service on the monitor.
- Connection status of each DB Connection:
  Right-click **DB Connection Settings** under **Configurations and Setup** - **Host Connection Settings** - **DB Connection** in the Multiview Explorer and select ***Connection Monitor Table*** from the menu. You can check the connection status of each DB Connection.

STEP 12 Checking the Operation Logs	Refer to *6 How to Use Operation Logs*.

You can check the following Operation Logs for tracing the operations of the DB Connection Service on the CPU Unit.

● Execution Log

This log is used to trace the executions of the DB Connection Service. Logging is kept while the DB Connection Service is running.

1. Right-click **DB Connection** under **Configurations and Setup** - **Host Connection Settings** and select ***Show Operation Logs*** from the menu and click the Execution Log Tab.

● Debug Log

This log is used for tracing which SQL statements were executed and parameters and execution result of each SQL statement.

1. Right-click **DB Connection** under **Configurations and Setup - Host Connection Settings** and select ***Show Operation Logs*** from the menu and click the Debug Log Tab.

● SQL Execution Failure Log

This log is recorded when an SQL execution failed in the DB.

1. Right-click **DB Connection** under **Configurations and Setup** - **Host Connection Settings** and select ***Show Operation Logs*** from the menu and click the SQL Execution Failure Log Tab.

STEP 13 Checking the Event Log	Refer to *7 Troubleshooting*.

2

DB Connection Settings

This section describes how to make the initial DB Connection settings for using the DB Connection Service.

2-1 Starting Sysmac Studio and Creating a New Project

This section describes how to start Sysmac Studio and create a new project when using the DB Connection function.
Refer to the *Sysmac Studio Version 1 Operation Manual* (Cat. No. W504) for detailed operations.

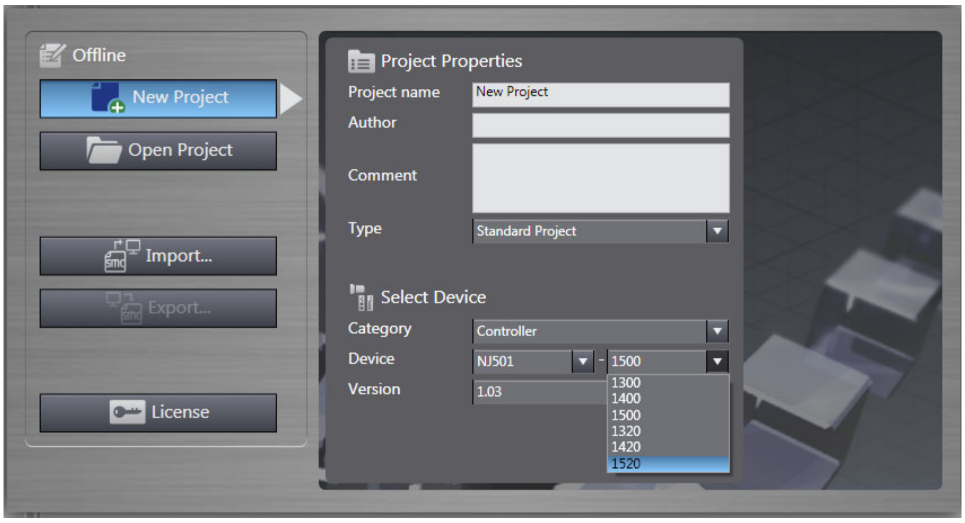
Refer to *B-4 Version Information* for correspondence between CPU Unit and DB Connection Service versions and between CPU Unit and Sysmac Studio versions.

2-1-1 Starting Sysmac Studio

1. Install Sysmac Studio version 1.14 or higher.
2. Start Sysmac Studio.

2-1-2 Creating a New Project

1. Select one of the following devices in the Device Field of the Select Device Area.
   NJ501: 1320,1420, or 1520
   NJ101: 1020 or 9020



Project Properties

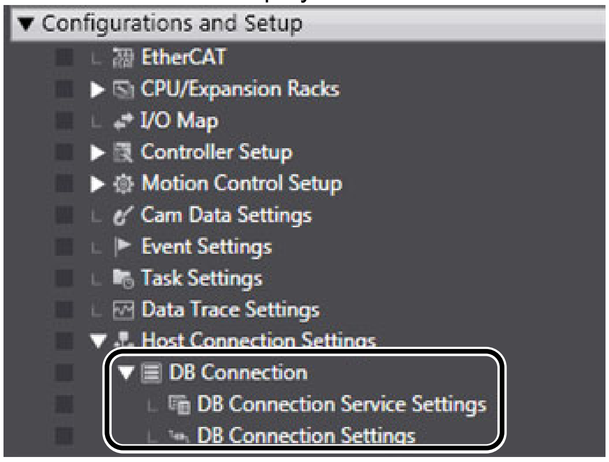
Project name	New Project
Author	
Comment	
Type	Standard Project

Select Device

Category	Controller
Device	NJ501 - 1500
Version	1.03

1300
1400
1500
1320
1420
1520

2. Click the **Create** Button.
   DB Connection is displayed under **Host Connection Settings** in the Multiview Explorer.



2-1-3 Setting the Built-in EtherNet/IP Port

1. Right-click **Built-in EtherNet/IP Port Settings** under **Configurations and Setup** - **Controller Setup** in the Multiview Explorer and select ***Edit*** from the menu.

2. Make the TCP/IP, LINK, FTP, NTP, SNMP, SNMP Trap, and FINS settings in the Built-in EtherNet/IP Port Settings Tab Page.

Refer to the *NJ/NX-series CPU Unit Built-in EtherNet/IP Port User's Manual* (Cat. No. W506) for the detailed settings.

When you use the DB Connection Service, the following port numbers are used in the built-in EtherNet/IP port. Do not set them for the other purposes.
Refer to the *NJ/NX-series CPU Unit Built-in EtherNet/IP Port User's Manual* (Cat. No. W506) for the port numbers commonly used in the NJ501-□□□□ and NJ101-□□□□ CPU Units.

Application	UDP	TCP
System-used	---	9800 to 9819

2-1-4 Controller Setup

Use Sysmac Studio to make the operation settings of the Controller.
Refer to the *NJ/NX-series CPU Unit Software User's Manual* (Cat. No. W501) for detailed settings that are not described below.

Operation Settings

1. Right-click **Operation Settings** under **Configurations and Setup** – **Controller Setup** in the Multiview Explorer and select ***Edit*** from the menu.

● Basic Settings

The Basic Settings are functions supported by the CPU Unit, such as the definitions of operations when the power is turned ON or when the operating mode changes.

Category	Item	Description	Value	Default	Update timing	Changes in RUN mode
Operation Settings	Start delay time at startup	Sets the time to perform system services with priority during startup after the power supply is turned ON.*	0 to 10 s	0 s	When downloading to CPU Unit	Not allowed

* The startup time of the DB Connection Service can be reduced with this setting. Set the value to *10* if you give priority to system services. Otherwise, set the value to *0*.
If you set the value to *10*, after the power supply is turned ON, the CPU Unit gives priority to the system services for approximately 10 seconds during startup before the Unit changes the startup state to the normal operation state. The time until the DB Connection Service becomes available (i.e., the *_DBC_Status.Run* system variable changes to True) can be reduced by performing a part of processing of the system services with priority during startup.
If you specify the value between 1 and 10, the time until the CPU Unit changes the state to the normal operation state is increased because the Unit gives priority to the system services for the specified time.

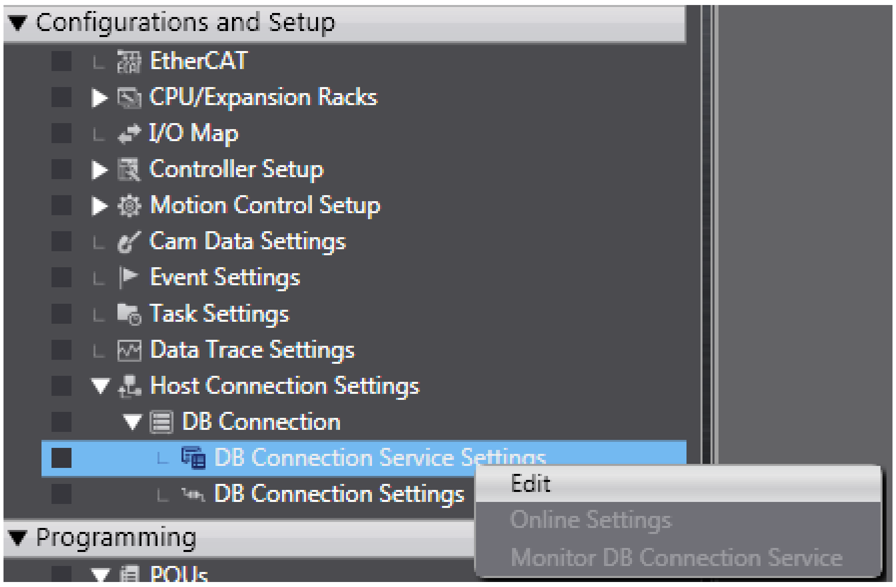
2-2 DB Connection Settings

You need to make the initial DB Connection settings before executing the DB Connection Service. Please make the settings of the entire DB Connection Service and each DB Connection.

This section describes the DB Connection Service settings and DB Connection settings.

2-2-1 DB Connection Service Settings

Right-click **DB Connection Service Settings** under **Configurations and Setup** - **Host Connection Settings** - **DB Connection** in the Multiview Explorer and select ***Edit*** from the menu.

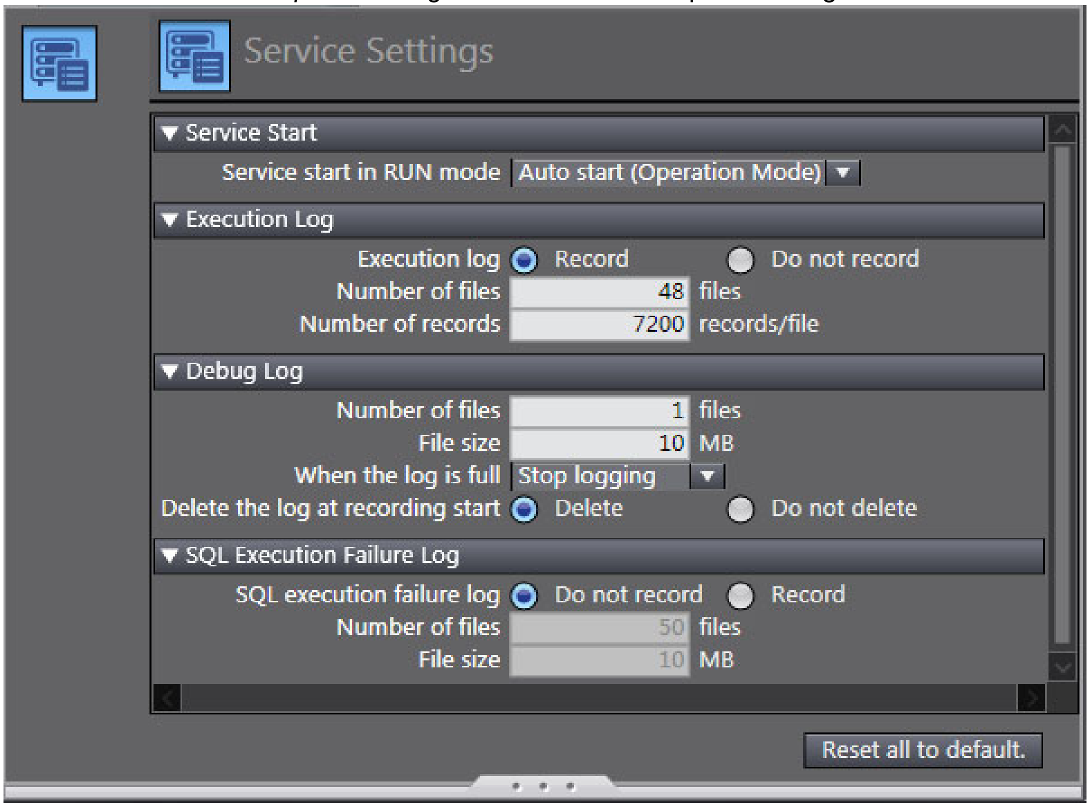


Service Settings

Make a setting for Service Start, Execution Log, Debug Log, and SQL Execution Failure Log in the Service Settings.

Refer to *4-1 Run Mode of DB Connection Service and Start/Stop Procedures* for details on how to start the DB Connection Service.

Refer to *6 How to Use Operation Logs* for details on the Operation Logs.



Service start in RUN mode: Auto start (Operation Mode)

Execution log: [✓ Record] Do not record
Number of files: 48 files
Number of records: 7200 records/file

Number of files: 1 files
File size: 10 MB
When the log is full: Stop logging
Delete the log at recording start: [✓ Delete] Do not delete

SQL execution failure log: [✓ Do not record] Record
Number of files: 50 files
File size: 10 MB

Set the following items.

Category	Item	Description	Value
Service Start	Service start in RUN mode	Set whether to automatically start the DB Connection Service when the operating mode of the CPU Unit is set to RUN mode.	• Auto start (Operation Mode)*1 (Default) • Auto start (Test Mode)*2 • Do not start automatically
Execution Log	Execution log	Set whether to record the Execution Log.	• Record (Default) • Do not record
	Number of files	Set the maximum number of files of the Execution Log. When the maximum number of files is reached, the oldest file is deleted and a new file is created.	2 to 100 files (Default: 48 files)
	Number of records	Set the number of log records that can be contained in each Execution Log file. When the maximum number of records is reached, a new file is created.	100 to 65536 records (Default: 7200 records)
Debug Log	Number of files	Set the maximum number of files of the Debug Log.	1 to 100 files (Default: 1 file)
	File size	Set the maximum file size. When the maximum file size is exceeded or when the number of records exceeds 65,536 records in a file, a new file is created.	1 to 100 MB (Default: 10 MB)

Category	Item	Description	Value
	When the log is full	Set the action to be taken when the log has reached the maximum number of files.	• Continue logging (Delete the oldest file) • Stop logging (Default)
	Delete the log at recording start	Set whether to delete the Debug Log contained in the SD Memory Card when recording is started.	• Delete (Default) • Do not delete
SQL Execution Failure Log	SQL execution failure log	Set whether to record the SQL Execution Failure Log.	• Record • Do not record (Default)
	Number of files	Set the maximum number of files of the SQL Execution Failure Log. When the maximum number of files is reached, the oldest file is deleted and a new file is created.	2 to 100 files (Default: 50 files)
	File size	Set the maximum file size. When the maximum file size is exceeded or when the number of records exceeds 65,536 records in a file, a new file is created.	1 to 100 MB (Default: 10 MB)

*1 When a DB Connection Instruction is executed, the DB Connection Service actually accesses the DB.
*2 When a DB Connection Instruction is executed, the DB Connection Service does not actually access the DB, but the instruction will end normally as if it was executed.

Additional Information

You can calculate the capacity of the Operation Log files that are stored on the SD Memory Card. If the SD Memory Card often runs out of space, please decrease the values of the following settings.

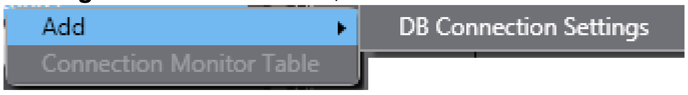
· Execution Log:
  Size of each record (256 bytes) x *Number of records* x *Number of files*
· Debug Log:
  File size x *Number of files*
· SQL Execution Failure Log:
  File size x *Number of files*

2-2-2 DB Connection Settings

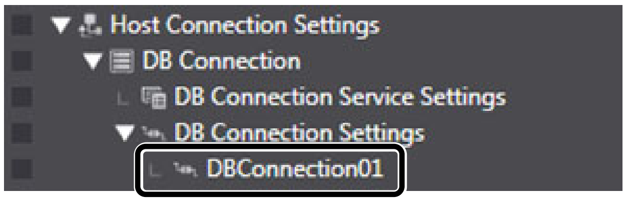
This section describes how to add and rename a DB Connection, and also describes the DB Connection setting procedure and items.

Adding a DB Connection

1. Right-click **DB Connection Settings** under **Configurations and Setup** - **Host Connection Settings** - **DB Connection** in the Multiview Explorer and select ***Add*** - ***DB Connection Settings*** from the menu. Or, select ***DB Connection Settings*** from the Insert Menu.



A DB Connection is added. You can add up to three DB Connections for NJ501-□□20 or up to one DB Connection for NJ101-□□20.

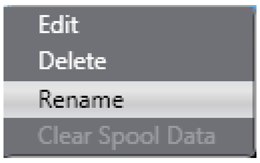


Changing the DB Connection Name

When a DB Connection is created, the following default name is automatically given. "**" is a serial number from 01.

"DBConnection**"

To change the name, right-click the DB Connection in the Multiview Explorer and select ***Rename*** from the menu.



· You can enter single-byte alphanumeric characters and underscores (_).
· Each DB Connection name can be up to 16 bytes.

● Editing or Deleting the DB Connection Settings

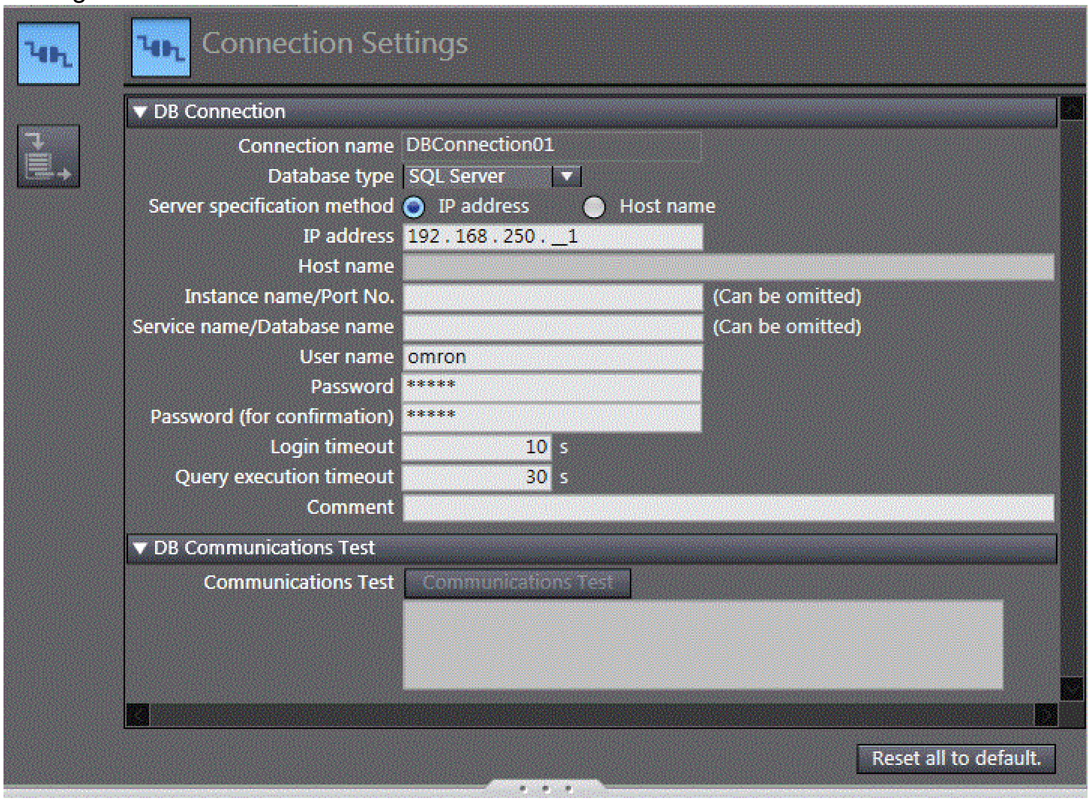
Right-click the DB Connection in the Multiview Explorer and select ***Edit*** or ***Delete*** from the menu.

Connection Settings

This section describes how to make a setting of each DB Connection and how to perform a communications test.

● DB Connection Settings

Double-click each DB Connection that you added and make the settings in the Connection Settings.



Connection name: DBConnection01
Database type: SQL Server
Server specification method: [✓ IP address] Host name
IP address: 192 . 168 . 250 . _1
Host name:
Instance name/Port No.: (Can be omitted)
Service name/Database name: (Can be omitted)
User name: omron
Password: *****
Password (for confirmation): *****
Login timeout: 10 s
Query execution timeout: 30 s
Comment:
Communications Test

Set the following items.

Category	Item	Description	Value
DB Connection	Connection name	The DB Connection name is displayed.	You can change the DB Connection name. To change the name, right-click the DB Connection in the Multiview Explorer and select ***Rename*** from the menu.
	Database type	Set the database type.	• NJ501-1□20 or NJ101-□□20 Oracle SQL Server (Default) DB2 MySQL Firebird PostgreSQL • NJ501-4320 Oracle SQL Server (Default) MySQL
	Server specification method	Select the specification method of the server. Select *IP address* or *Host name*.	• IP address (Default) • Host name
	IP address	Set the IP address of the server.	Default: Blank This setting cannot be omitted when *IP address* is selected for *Server specification*

Category	Item	Description	Value
			method.
	Host name	Set the host name of the server.*	Default: Blank This setting cannot be omitted when *Host name* is selected for *Server specification method*.
	Instance name / Port No.	Set the instance name or port number of the server.	・ Oracle: Port No. (Can be omitted) e.g. 1521 ・ SQL Server: Instance name or Port No. (Can be omitted) e.g. INSTANCE1 or 1433 ・ DB2: Port No. (Cannot be omitted) e.g. 50000 ・ MySQL: Port No. (Can be omitted) e.g. 3306 ・ Firebird: Port No. (Can be omitted) e.g. 3050 ・ PostgreSQL Port No. (Can be omitted) e.g. 5432 Maximum number of characters for instance name: 64 characters Port No.: 1 to 65535 Default: Blank When omitted, the default port number is used. ・Oracle: 1521 ・SQL Server: 1433 ・MySQL: 3306 ・Firebird: 3050 ・PostgreSQL: 5432
	Service name/ Database name	Set the service name or database name in the server.	・Oracle: Service name (Can be omitted) ・SQL Server: Database name (Can be omitted) ・DB2: Database name (Cannot be omitted) ・MySQL: Database name (Cannot be omitted) ・Firebird: Database path (Cannot be omitted) e.g., C:/Firebird/OMRON.FDB ・PostgreSQL: Database name (Cannot be omitted) Maximum number of bytes: 127 bytes When omitted, ・Oracle: Default service ・SQL Server: Default database

Category	Item	Description	Value
	User name	Set the user name for the server.	・DB2: Windows user name of the server ・Other DBs: DB user name of the server Maximum number of characters: 127 characters Default: Blank
	Password	Set the password for the server.	・DB2: Windows password of the server ・Other DBs: DB password of the server Maximum number of characters: 127 characters Default: Blank
	Login timeout	Set the timeout to be applied when connecting to the DB.	1 to 60 seconds Default: 10 seconds
	Query execution timeout	Set the timeout to be applied at the SQL execution.	1 to 600 seconds Default: 30 seconds
	Comment	Enter a comment.	Maximum number of bytes: 1,024 bytes Default: Blank The comment can be omitted.

* When you specify a server by its host name, you need to set *DNS* to *Use* or make the host settings in the Built-in EtherNet/IP Port Settings. Refer to the *NJ/NX-series CPU Unit Built-in EtherNet/IP Port User's Manual* (Cat. No. W506) for details on the settings.

Version Information

When you use NJ501-1□20 or NJ101-□□20, the supported database types depend on the combination of the DB Connection Service version of the CPU Unit and the DB Connection Service version set in the Sysmac Studio project.
For the relationship between the unit version of the CPU Unit and the unit version set in the Sysmac Studio project, refer to *Section B-4-3 Actual Unit Version of CPU Unit and Unit Version Set in the Sysmac Studio Project*.

● Communications Test

You can test the connection to the DB according to the settings made in the Connection Settings* of Sysmac Studio.

* This is not the DB Connection Settings that have been transferred to the Controller.

You can perform the communications test while Sysmac Studio is online with the Controller.

1. Use the Synchronization function to transfer the DB Connection settings from the computer to the Controller.
2. Click the **Communications Test** Button under DB Communications Test.
3. The result of the communications test is displayed in the text box under the **Communications Test** Button.

When the connection to the server failed from any cause, the SQL status, error code, and detailed error message will be displayed.

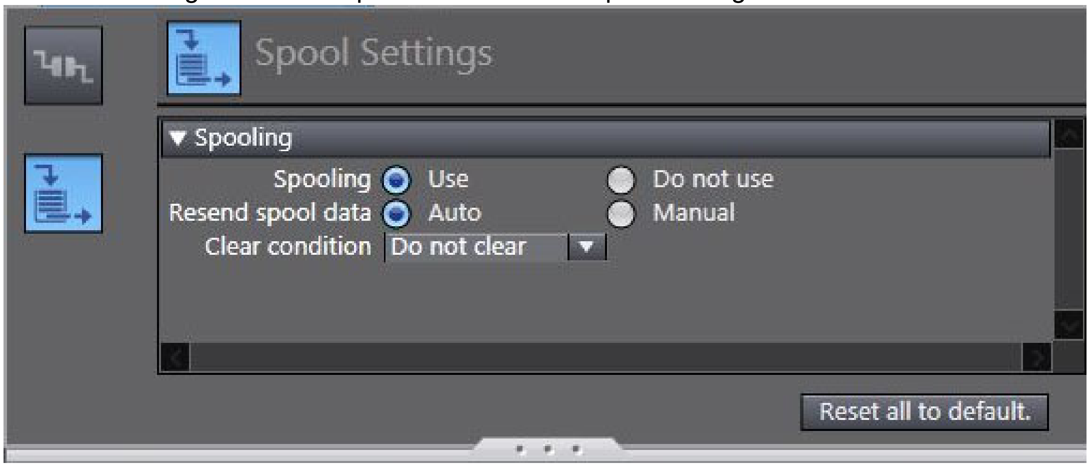
SQL status: Error code defined in the SQL Standards (ISO/IEC 9075).

Error code: Error code specific to the vendor of DB to connect.
When a network failure has occurred, 0 is displayed for error code in some cases. When 0 is displayed, check its SQL status.

Detailed error message: Error message specific to the vendor of DB to connect.

Spool Settings

Make the settings related to Spool function in the Spool Settings.



Spooling [✓ Use] Do not use
Resend spool data [✓ Auto] Manual
Clear condition: Do not clear

Refer to *Section 5-1 Spool Function* for detailed settings.

3 Programming the DB Connection Function

This section describes programming procedure from variable creation to DB access after making the DB Connection settings.

3-1 DB Access Procedure

This section describes a specific programming procedure for using the DB Connection Service. Refer to the NJ/NX-series CPU Unit Software User's Manual (Cat. No. W501) for the general programming procedure.

Use the following procedure to access the DB using DB Connection Instructions after making the DB Connection settings.

After the DB mapping*, you can read from and write to the DB using record processing instructions such as DB_Insert, DB_Update, and DB_Select instructions.

DB mapping*	Create a structure data type for DB access.	Refer to *3-2 Creating a Structure Data Type.*
	↓	
	Create a variable called "DB Map Variable" using the above structure.	Refer to *3-3 Creating a DB Map Variable.*
	↓	
	Establish a DB Connection by executing a DB_Connect (Establish DB Connection) instruction.	Refer to *4-2 Establishing/Closing a DB Connection.*
	↓	
	Create a mapping from the DB Map Variable to a specified table by executing a DB_CreateMapping (Create DB Map) instruction for each SQL type (i.e., INSERT, UPDATE, and SELECT).	Refer to *3-4 Specifying the Table and Applying the Mapping.*
	↓	
DB read/write	Execute the DB_Insert (Insert DB Record), DB_Update (Update DB Record), and DB_Select (Retrieve DB Record) instructions.	Refer to *3-5 Programming Using the DB Connection Instructions.*

* The DB mapping means to assign each member of a structure for DB access to each column of a table. You need to execute the DB mapping for each SQL type (i.e. INSERT, UPDATE, and SELECT).

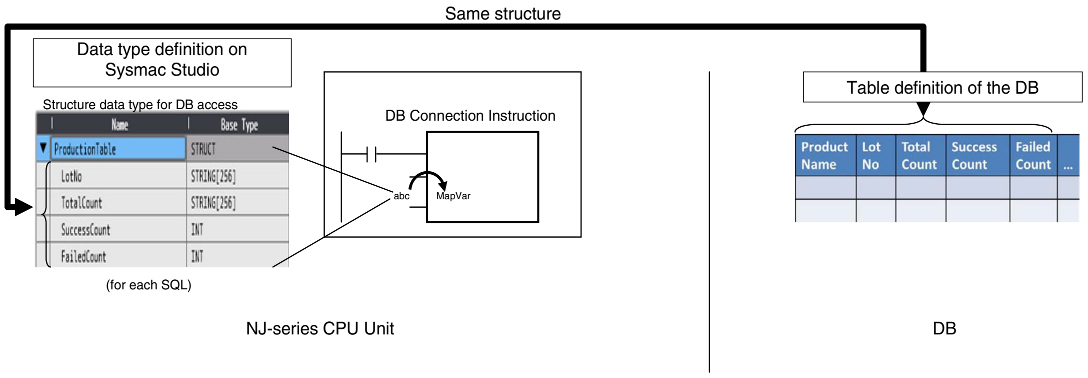
3-2 Creating a Structure Data Type

To access a DB, you need to create a user-defined structure data type according to the table definition of the DB.
This section describes the specifications and creation procedure of the structure data type.

3-2-1 Overview

You create a user-defined structure data type on Sysmac Studio based on the data type of the table to access. Register all or some of the columns of the table as structure members.

Each structure member name and data type must match the corresponding column name and data type of the table.


Same structure
Data type definition on Sysmac Studio
Structure data type for DB access
DB Connection Instruction
abc
MapVar
(for each SQL)
NJ-series CPU Unit
Table definition of the DB
DB

Name	Base Type
▼ ProductionTable	STRUCT
LotNo	STRING[256]
TotalCount	STRING[256]
SuccessCount	INT
FailedCount	INT

Product Name	Lot No	Total Count	Success Count	Failed Count	...

When creating a variable called “DB Map Variable”, you specify the structure as its data type.

3-2-2 Specifications of Structure Data Type for DB Access

Item	Specifications
Structure name	You can specify any name for the structures.
Offset specification for structure members	Specify *NJ* for *Offset Type*.
Structure members	Register all or some of the columns of the table as members.
Structure member name	Define the same name as the corresponding column of the table. The names are case sensitive.
Structure member’s data type	Define a data type that matches the data type of the corresponding column of the table. Refer to *Correspondence of Data Types between NJ-series Controllers and DB* for details. However, you cannot specify the following data types and attribute for structure members. - Derivative data types - Array attribute

Precautions for Correct Use

Restrictions on Table’s Column Names:
You need to specify the same name for structure members to be used in NJ-series Controllers as the column names of the table to access.
There are following restrictions on structure member names in the NJ-series Controllers. Therefore, make the column names satisfy the following conditions.

Item	Description
Usable characters	0 to 9, A to Z, a to z Single-byte *Japanese kana* _ (underscores) Multi-byte characters (e.g., Japanese)
Characters that cannot be used together	- A text string that starts with a number (0 to 9) - A text string that starts with P_ - A text string that starts with an underscore (_) character - A text string that contains more than one underscore (_) character - A text string that ends in an underscore (_) character - Any text string that consists of an identifier and has a prefix or postfix which contains more than one extended empty space character (i.e., multi-byte spaces or any other empty Unicode space characters)

● Correspondence of Data Types between NJ-series Controllers and DB

The correspondence of data types between NJ-series Controllers and DB is given in the following tables.

· Oracle

Data type category	Data type in DB	Data type in NJ-series Controllers
Characters	VARCHAR2	STRING[*1]
	NVARCHAR2	STRING[*1]
	CHAR	STRING[*1]
	NCHAR	STRING[*1]
	LONG	None
	CLOB	None
	NCLOB	None

Numbers[*2]	
	NUMBER(1)
	NUMBER(3)
	NUMBER(5)
	NUMBER(10)
	NUMBER(19)
	NUMBER(3)
	NUMBER(5)
	NUMBER(10)
	NUMBER(20)
		[*3]
		BOOL
		SINT
		INT
		DINT
		LINT
		USINT
		UINT
		UDINT
		ULINT

	NUMBER(19)	TIME[*4]
	BINARY_FLOAT	REAL
	BINARY_DOUBLE	LREAL
	FLOAT	REAL
	INTEGER	DINT
Date	DATE	DATE
	TIMESTAMP	DATE DATE_AND_TIME
	TIMESTAMP WITH TIMEZONE	DATE_AND_TIME

Data type category	Data type in DB	Data type in NJ-series Controllers
	TIMESTAMP WITH LOCAL TIMEZONE	DATE_AND_TIME
	INTERVAL YEAR TO MONTH	None
	INTERVAL DAY TO SECOND	None
Binary	RAW	None
	LONG RAW	None
	BLOB	None
Others	BFILE	None
	ROWID	None
	UROWID	None
	XMLTYPE	None

*1 A NULL character is attached to the end of each text string. Therefore, you need to set the value that is one byte bigger than the number of bytes of the DB's data type for the number of bytes to be used in STRING data.
You need to set an appropriate value for the number of bytes used in the STRING data according to the data type and character code in the DB. In NJ Series, text strings are handled as UTF-8. One byte is used for each single-byte alphanumeric character and multiple bytes are used for each multi-byte character. Three bytes are used for each Japanese character as a guide.

*2 The NUMBER(p[,s]) is expressed in the short form where the number of digits after the decimal point (s) is omitted. When the short form is used, the number of digits after the decimal point (s) is 0. If the number of digits after the decimal point (s) is not omitted and 1 or greater numerical value is set, only the integer portion of the value is applicable.

*3 Digit overflow may occur even in the above data types due to the difference in the valid range.
Example: When the data type in DB is NUMBER(3) and the data type in NJ-series Controllers is USINT:
NUMBER(3)'s range: 0 to 999
USINT's range: 0 to 255

*4 Integer in units of nanoseconds.

· SQL Server

Data type category	Data type in DB	Data type in NJ-series Controllers
Numbers*1	bigint	LINT UDINT
		TIME*2
	bit	BOOL

	decimal(1)
	decimal(3)
	decimal(5)
	decimal(10)
	decimal(19)
	decimal(20)
	decimal(3)
	decimal(5)
	decimal(10)
		*3
		BOOL
		SINT
		INT
		DINT
		LINT
		ULINT
		USINT
		UINT
		UDINT

	decimal(19)	TIME
	int	DINT UINT
	money	LREAL*4
	numeric(1)	*3 BOOL

Data type category	Data type in DB	Data type in NJ-series Controllers

	numeric(3)
	numeric(5)
	numeric(10)
	numeric(19)
	numeric(20)
	numeric(3)
	numeric(5)
	numeric(10)
		SINT
		INT
		DINT
		LINT
		ULINT
		USINT
		UINT
		UDINT

	numeric(19)	TIME
	smallint	INT USINT
	smallmoney	REAL*5
	tinyint	USINT
	float	LREAL
	real	REAL
Date and time	date	DATE
	datetime2	DATE_AND_TIME*6
	datetime	DATE_AND_TIME
	datetimeoffset	DATE_AND_TIME*6
	smalldatetime	DATE_AND_TIME
	time	TIME_OF_DAY*6
String	char	STRING*7
	text	STRING*7
	varchar	STRING*7
	nchar	STRING*7
	ntext	STRING*7
	nvarchar	STRING*7
Binary	binary	None
	image	None
	varbinary	None
Others	cursor	None
	hierarchyid	None
	sql_variant	None
	table	None
	uniqueidentifier	None
	xml	None

*1 The decimal (p[,s]) and numeric (p[,s]) are expressed in the short form where the number of digits after the decimal point (s) is omitted. When the short form is used, the number of digits after the decimal point (s) is 0. If the number of digits after the decimal point (s) is not omitted and 1 or greater numerical value is set, only the integer portion of the value is applicable.

*2 Integer in units of nanoseconds

*3 Digit overflow may occur even in the above data types due to the difference in the valid range.
Example: When the data type in DB is decimal(3) and the data type in NJ-series Controllers is USINT:
decimal(3)'s range: 0 to 999
USINT's range: 0 to 255

*4 The significant figures are 15 digits. When the data is written to the DB by a DB Connection Instruction, a value rounded to four decimal places is written.
Example: When 1.79769 is written to the DB, 1.7977 is written.

*5 The significant figures are 7 digits. When the data is written to the DB by a DB Connection Instruction, a value rounded to four decimal places is written.

Example: When 1.79769 is written to the DB, 1.7977 is written.

*6 The accuracy is milliseconds.

*7 A NULL character is attached to the end of each text string. Therefore, you need to set the value that is one byte bigger than the number of bytes of the DB's data type for the number of bytes to be used in STRING data.
You need to set an appropriate value for the number of bytes used in the STRING data according to the data type and character code in the DB. In NJ Series, text strings are handled as UTF-8. One byte is used for each single-byte alphanumeric character and multiple bytes are used for each multi-byte character. Three bytes are used for each Japanese character as a guide.

・ DB2

Data type category	Data type in DB	Data type in NJ-series Controllers
Numbers	INT	DINT
	INTEGER	DINT
	BIGINT	LINT TIME
	SMALLINT	INT

Fixed-point numbers[*1]	
	DECIMAL(1)
	DECIMAL(3)
	DECIMAL(5)
	DECIMAL(10)
	DECIMAL(20)
	DECIMAL(3)
	DECIMAL(5)
	DECIMAL(10)
	DECIMAL(20)
		[*2]
		BOOL
		SINT
		INT
		DINT
		LINT
		USINT
		UINT
		UDINT
		ULINT

	DECIMAL(20)	TIME
Real numbers	FLOAT	REAL LREAL
	REAL	REAL
	DOUBLE	LREAL
Date and time	DATE	DATE
	TIME	TIME_OF_DAY
	TIMESTAMP	DATE_AND_TIME
String	CHAR	STRING[*3]
	CHARACTER	STRING[*3]
	VARCHAR	STRING[*3]
	CHAR VARYING	STRING[*3]
	CHARACTER VARYING	STRING[*3]
	LONG VARCHAR	STRING[*3]
	CLOB	None
Binary string	BLOB	None
Others	GRAPHIC	None
	VARGRAPHIC	None
	LONG VARGRAPHIC	None
	DBCLOB	None
	DATALINK	None

*1 The DECIMAL(p[,s]) is expressed in the short form where the number of digits after the decimal point (s) is omitted. When the short form is used, the number of digits after the decimal point (s) is 0. If the number of digits after the decimal point (s) is not omitted and 1 or greater numerical value is set, only the integer portion of the value is applicable.

*2 Digit overflow may occur even in the above data types due to the difference in the valid range.

Example: When the data type in DB is DECIMAL(3) and the data type in NJ-series Controllers is USINT:

DECIMAL(3)'s range: 0 to 999

USINT's range: 0 to 255

*3 A NULL character is attached to the end of each text string. Therefore, you need to set the value that is one byte bigger than the number of bytes of the DB's data type for the number of bytes to be used in STRING data.

You need to set an appropriate value for the number of bytes used in the STRING data according to the data type and character code in the DB. In NJ Series, text strings are handled as UTF-8. One byte is used for each single-byte alphanumeric character and multiple bytes are used for each multi-byte character. Three bytes are used for each Japanese character as a guide.

• MySQL

Data type category	Data type in DB	Data type in NJ-series Controllers
Numbers[*1]	BIT	BOOL
	BOOL BOOLEAN	BOOL
	TINYINT	SINT USINT
	SMALLINT	INT UINT
	MEDIUMINT	DINT UDINT
	INT	DINT UDINT
	BIGINT	LINT ULINT TIME

	DECIMAL(1)
	DECIMAL(3)
	DECIMAL(5)
	DECIMAL(10)
	DECIMAL(20)
	DECIMAL(3)
	DECIMAL(5)
	DECIMAL(10)
	DECIMAL(20)
		[*2]
		BOOL
		SINT
		INT
		DINT
		LINT
		USINT
		UINT
		UDINT
		ULINT

	DECIMAL(20)	TIME
	FLOAT	REAL
	DOUBLE	LREAL
Date and time	DATE	DATE
	DATETIME	DATE_AND_TIME
	TIMESTAMP	DATE_AND_TIME
	TIME	TIME_OF_DAY
String	CHAR	STRING[*3]
	VARCHAR	STRING[*3]
	TINYTEXT	STRING[*3]
	TEXT	STRING[*3]
	MEDIUMTEXT	STRING[*3]
	LONGTEXT	STRING[*3]
Binary	BINARY	None
	VARBINARY	None
	TINYBLOB	None

Data type category	Data type in DB	Data type in NJ-series Controllers
	BLOB	None
	MEDIUMBLOB	None
	LONGBLOB	None
Others	ENUM	None
	YEAR	None
	SET	None

*1 The DECIMAL(p[,s]) is expressed in the short form where the number of digits after the decimal point (s) is omitted. When the short form is used, the number of digits after the decimal point (s) is 0. If the number of digits after the decimal point (s) is not omitted and 1 or greater numerical value is set, only the integer portion of the value is applicable.

*2 Digit overflow may occur even in the above data types due to the difference in the valid range.
Example: When the data type in DB is DECIMAL(3) and the data type in NJ-series Controllers is USINT:
DECIMAL(3)'s range: 0 to 999
USINT's range: 0 to 255

*3 A NULL character is attached to the end of each text string. Therefore, you need to set the value that is one byte bigger than the number of bytes of the DB's data type for the number of bytes to be used in STRING data.
You need to set an appropriate value for the number of bytes used in the STRING data according to the data type and character code in the DB. In NJ Series, text strings are handled as UTF-8. One byte is used for each single-byte alphanumeric character and multiple bytes are used for each multi-byte character. Three bytes are used for each Japanese character as a guide.

• Firebird

Data type category	Data type in DB	Data type in NJ-series Controllers
Integers	INTEGER	DINT
	BIGINT	LINT TIME
	SMALLINT	INT

Fixed-point numbers[*1]	
	DECIMAL(1)
	DECIMAL(3)
	DECIMAL(5)
	DECIMAL(10)
	DECIMAL(18)
	DECIMAL(3)
	DECIMAL(5)
	DECIMAL(10)
	DECIMAL(18)
		[*2]
		BOOL
		SINT
		INT
		DINT
		LINT[*3]
		USINT
		UINT
		UDINT
		ULINT[*3]

	NUMERIC(1)
	NUMERIC(3)
	NUMERIC(5)
	NUMERIC(10)
	NUMERIC(18)
	NUMERIC(3)
	NUMERIC(5)
	NUMERIC(10)
	NUMERIC(18)
		[*2]
		BOOL
		SINT
		INT
		DINT
		LINT[*3]
		USINT
		UINT
		UDINT
		ULINT[*3]

Real numbers	FLOAT	REAL
	DOUBLE	LREAL

Data type category	Data type in DB	Data type in NJ-series Controllers
	PRECISION	
Date	DATE	DATE
	TIME	TIME_OF_DAY
	TIMESTAMP	DATE_AND_TIME
String	CHAR	STRING[*4]
	VARCHAR	STRING[*4]
Others	BLOB	None

*1 The DECIMAL(p[,s]) and NUMERIC(p[,s]) are expressed in the short form where the number of digits after the decimal point (s) is omitted. When the short form is used, the number of digits after the decimal point (s) is 0. If the number of digits after the decimal point (s) is not omitted and 1 or greater numerical value is set, only the integer portion of the value is applicable.

2* Digit overflow may occur even in the above data types due to the difference in the valid range.
Example: When the data type in DB is DECIMAL(3) and the data type in NJ-series Controllers is USINT:
DECIMAL(3)'s range: 0 to 999
USINT's range: 0 to 255

*3 The DB can handle up to 18 digits. If an over-18-digit value is written by a DB Connection Instruction, an error will occur.

*4 A NULL character is attached to the end of each text string. Therefore, you need to set the value that is one byte bigger than the number of bytes of the DB's data type for the number of bytes to be used in STRING data.
You need to set an appropriate value for the number of bytes used in the STRING data according to the data type and character code in the DB. In NJ Series, text strings are handled as UTF-8. One byte is used for each single-byte alphanumeric character and multiple bytes are used for each multi-byte character. Three bytes are used for each Japanese character as a guide.

• PostgreSQL

Data type category	Data type in DB	Data type in NJ-series Controllers
Numbers	boolean	BOOL
	smallint	INT
	integer	DINT
	bigint	LINT TIME
	serial	UDINT
	bigserial	ULINT
Fixed-point numbers[*1]	decimal(3) decimal (5) decimal (10) decimal (20) decimal (3) decimal (5) decimal (10) decimal (20)	[*2] SINT INT DINT LINT USINT UINT UDINT ULINT
	numeric (3) numeric (5) numeric (10) numeric (20) numeric (3)	[*2] SINT INT DINT LINT USINT

Data type category	Data type in DB	Data type in NJ-series Controllers
	numeric (5) numeric (10) numeric (20)	UINT UDINT ULINT
Real numbers	real	REAL
	double precision	LREAL
Date and time	timestamp [(p)] [without time zone]	DATE_AND_TIME
	timestamp [(p)] with time zone	DATE_AND_TIME
	date	DATE
	time [(p)] [without time zone]	TIME_OF_DAY
	time [(p)] with time zone	TIME_OF_DAY
String	character(n), char(n)	STRING[*3]
	character varying(n), varchar(n)	STRING[*3]
	Text	STRING[*3]
Others	bit [(n)]	None
	bit varying [(n)]	None
	Box	None
	Bytea	None
	Cidr	None
	Circle	None
	Inet	None
	interval [fields] [(p)]	None
	Line	None
	Lseg	None
	macaddr	None
	money	None
	path	None
	point	None
	polygon	None
	tsquery	None
	tsvector	None
	txid_snapshot	None
	uuid	None
	xml	None

*1 The decimal(p[,s]) and numeric(p[,s]) are expressed in the short form where the number of digits after the decimal point (s) is omitted. When the short form is used, the number of digits after the decimal point (s) is 0. If the number of digits after the decimal point (s) is not omitted and 1 or greater numerical value is set, only the integer portion of the value is applicable.

*2 Digit overflow may occur even in the above data types due to the difference in the valid range.
Example: When the data type in DB is DECIMAL(3) and the data type in NJ-series Controllers is USINT:
DECIMAL(3)’s range: 0 to 999
USINT’s range: 0 to 255

*3 A NULL character is attached to the end of each text string. Therefore, you need to set the value that is one byte bigger than the number of bytes of the DB’s data type for the number of bytes to be used in STRING data.
You need to set an appropriate value for the number of bytes used in the STRING data according to the data type and character code in the DB. In NJ Series, text strings are handled as UTF-8. One byte is used for each single-byte alphanumeric character and multiple bytes are used for each multi-byte character. Three bytes are used for each Japanese character as a guide.

Precautions for Correct Use

- When a data type that is not listed in the above tables is used in the NJ-series Controller, the data may not be converted correctly.
- When reading a value from a database using a DB Connection Instruction, an instruction error (SQL Execution Error) may occur because the data type cannot be converted due to the following reasons.
  The retrieved record contains a column whose value is NULL.
  The combination of data types is not listed in the above tables.

3-2-3 How to Create a Structure Data Type for DB Access

You can use the following procedures for creating a structure data type for accessing a DB.
· Entering the data on the Data Type Editor
· Pasting the data from Microsoft Excel onto the Data Type Editor

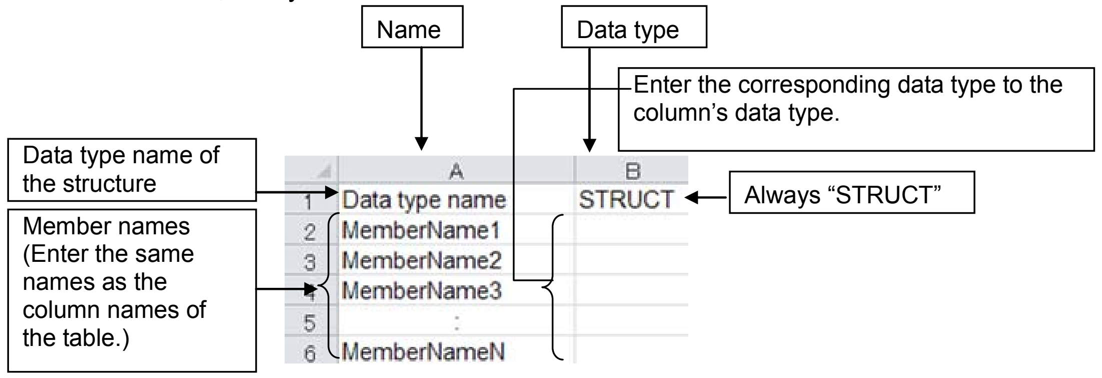
This section gives brief explanation for the operations. Refer to the *Sysmac Studio Version 1 Operation Manual* (Cat. No. W504) for detailed operations.

Entering the Data on the Data Type Editor

1. Double-click **Data Types** under **Programming** - **Data** in the Multiview Explorer.
2. Click the Structures Side Tab of the Data Type Editor.
3. Enter a data type name on the Structure Data Type Editor.
4. Right-click the structure name and select ***Create New Member*** from the menu. Then, enter a name and data type for each member.

Pasting the Data from Microsoft Excel onto the Data Type Editor

1. Use two columns on Microsoft Excel to enter names and data types from the left.
2. In the 1st column, enter the data type name of the structure on the 1st line and each member name from the 2nd line.
   In the 2nd column, always enter “STRUCT” on the 1st line to create a structure.


Name
Data type
Enter the corresponding data type to the column’s data type.
Data type name of the structure
Member names (Enter the same names as the column names of the table.)
Always “STRUCT”

	A	B
1	Data type name	STRUCT
2	MemberName1	
3	MemberName2	
4	MemberName3	
5	:	
6	MemberNameN	

3. Copy the data area in the Name and Data type columns on Microsoft Excel.
4. Paste the data onto the Name and Base Type columns of the Structure Data Type Editor.

Example:

DB's table definition

Data type name

Table name → ProductionTable

Column name

ProductName	VARCHARA2(256)
LotNo	VARCHARA2(256)
TotalCount	NUMBER(5)
SuccessCount	NUMBER(5)
FailedCount	NUMBER(5)

Enter the following on Microsoft Excel

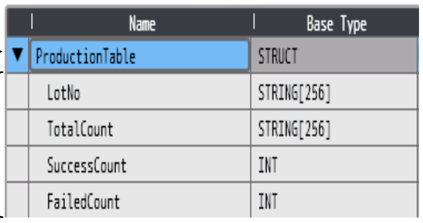
Data type name

Data type name

Member name

ProductionTable	STRUCT
ProductName	STRING[256]
LotNo	STRING[256]
TotalCount	INT
SuccessCount	INT
FailedCount	INT

Copy & paste onto Sysmac Studio



Name	Base Type
▼ ProductionTable	STRUCT
LotNo	STRING[256]
TotalCount	STRING[256]
SuccessCount	INT
FailedCount	INT

Precautions for Correct Use

You cannot paste the data type onto the Structure Data Type Editor in the following cases.

- When a structure member is selected on the editor
- When nothing is selected on the editor

When executing the Paste operation on the Structure Data Type Editor, please select a structure data type, not a member.

Additional Information

You can reuse table definition data of your DB development tool to create a structure data type for DB access.

Use the following procedure.

1. Copy the column name and data type on the table definition data of the DB development tool.

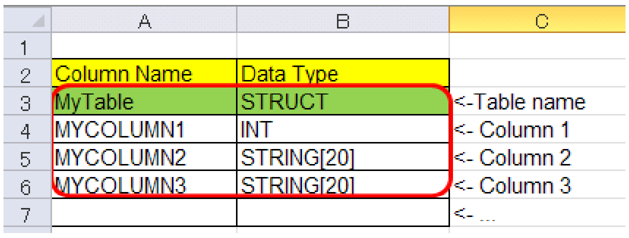
2. Create a Column Name column and a Data Type column on Microsoft Excel or other spreadsheet software.

3. Change the data type of each column to the corresponding data type for variables of NJ-series CPU Units.

4. Insert a line above the data of column names and data types and enter the name of the structure data type.

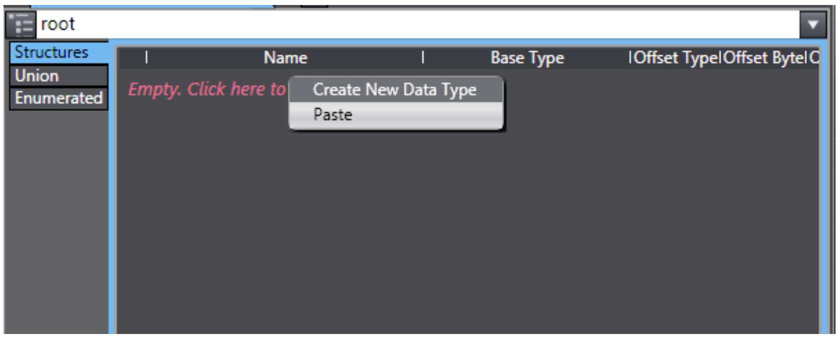
5. Enter “STRUCT” in the Data Type column on the inserted line.

6. Copy the data area under the Column Name and Data Type as shown below.

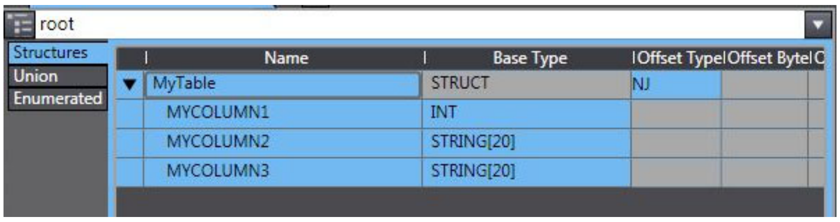


	A	B	C
1			
2	Column Name	Data Type	
3	MyTable	STRUCT	<-Table name
4	MYCOLUMN1	INT	<- Column 1
5	MYCOLUMN2	STRING[20]	<- Column 2
6	MYCOLUMN3	STRING[20]	<- Column 3
7			<- ...

7. Right-click on the Structure Data Type Editor and select ***Paste*** from the menu.



A structure data type is created as shown below.



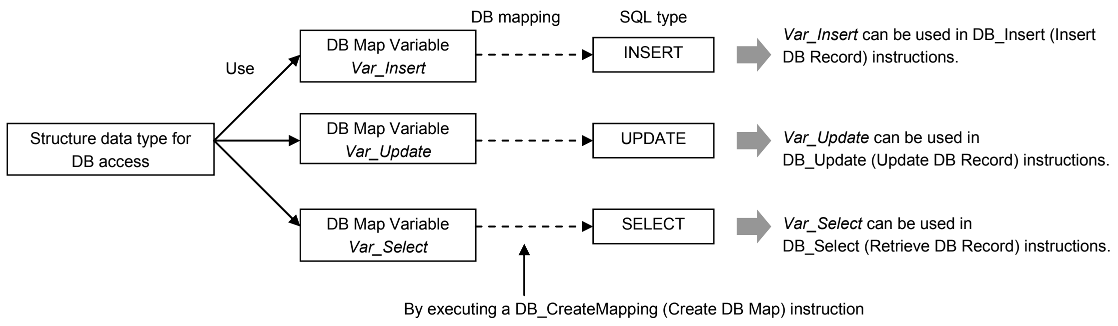
Name	Base Type	Offset Type	Offset Byte
MyTable	STRUCT	NJ	
MYCOLUMN1	INT		
MYCOLUMN2	STRING[20]		
MYCOLUMN3	STRING[20]		

3-3 Creating a DB Map Variable

After creating a user-defined structure data type for DB access, you create a variable using the data type. The variable is called “DB Map Variable”.
This section describes the specifications and creation procedure of DB Map Variables.

3-3-1 DB Map Variables and DB Mapping

Each DB Map Variable uses a structure data type for DB access as its data type.
You create a mapping* for a DB Map Variable to the connected DB for each SQL type (i.e., INSERT, UPDATE, and SELECT) by executing a DB_CreateMapping (Create DB Map) instruction.
After creating the DB mapping, you can execute each record processing for inserting, updating, and retrieving records using the DB Map Variable by executing a DB_Insert (Insert DB Record), DB_Update (Update DB Record), or DB_Select (Retrieve DB Record) instruction.


DB mapping
SQL type
Use
Structure data type for DB access
DB Map Variable Var_Insert
DB Map Variable Var_Update
DB Map Variable Var_Select
INSERT
UPDATE
SELECT
Var_Insert can be used in DB_Insert (Insert DB Record) instructions.
Var_Update can be used in DB_Update (Update DB Record) instructions.
Var_Select can be used in DB_Select (Retrieve DB Record) instructions.
By executing a DB_CreateMapping (Create DB Map) instruction

* The DB mapping means to assign each member of a DB Map Variable to the corresponding column of a table in the connected DB. You need to execute the DB mapping for each record processing for inserting, updating, and retrieving records.

You can map more than one DB Map Variable for a DB Connection.
The following table shows the operation of each record processing (i.e., INSERT, UPDATE, and SELECT) to be performed when you create a structure where not all, but some of the columns are specified as members.

Record processing	Operation
Inserting records (INSERT)	The record values are written to the specified columns of the DB. NULL is entered in the unspecified columns. You need to make a setting for allowing NULL in the DB.
Updating records (UPDATE)	Values are updated only in the specified columns. Values are not changed in the unspecified columns.
Retrieving records (SELECT)	Values are retrieved only from the specified columns. You need to specify only the columns that do not contain NULL.

Precautions for Correct Use

If you retrieve a record that includes a column of NULL value when executing a DB_Select (Retrieve DB Record) instruction, the instruction will result in an instruction error (SQL Execution Error).

Additional Information

When a DB_CreateMapping (Create DB Map) instruction is executed to create a mapping for a DB Map Variable, it is not checked whether the structure members match the table's columns. In this case, the DB_Insert (Insert DB Record), DB_Update (Update DB Record), or DB_Select (Retrieve DB Record) instruction will result in an error.

3-3-2 Registration and Attributes of DB Map Variables

You can specify the following variable types and attributes for DB Map Variables.

Item		Available type/settings	Restrictions
Registration of variables		Global variable Local variable for a program Local variable for a function block	A local variable for a function cannot be specified.[*1]
Attributes	Variable name	Any	Refer to the *NJ/NX-Series CPU Unit Software User's Manual* (Cat. No. W501) for the restrictions on the variable names and other program-related names.
	Data Type	Structure data type for DB access	Refer to *3-2 Creating a Structure Data Type*.
	AT	Any	
	Retain	Any	
	Initial Value	Any	
	Constant	Any	This attribute cannot be specified for SELECT. A compiling error will occur for DB_Select (Retrieve DB Record) instructions.
	Network Publish	Any	
	Edge	This attribute cannot be specified.	
Array specification		Array can be specified for SELECT.	Array cannot be specified for INSERT nor UPDATE. An instruction error will occur for DB_CreateMapping (Create DB Map) instructions. Refer to *3-3-3 Restrictions on DB Map Variables* for details.

*1 The DB Map Variables cannot be used in any function POU because the DB_CreateMapping (Create DB Map) instruction is a function block type of instruction.

Precautions for Correct Use

When a DB Connection Instruction is used in a function block and an in-out variable of the function block is specified as a DB Map Variable, system-defined initial values for the data types are applied to the members of the DB Map Variable when the DB Connection Instruction is executed. Do not specify an in-out variable of a function block as a DB Map Variable.
If you need to use an in-out variable for a DB Connection Instruction, specify an internal variable of the function block as a DB Map Variable and transfer the data between in-out variable and internal variable using a MOVE or other instruction before executing a DB_Insert or DB_Update instruction or after executing a DB_Select instruction.

3-3-3 Restrictions on DB Map Variables

This section describes the restrictions on DB Map Variables.

Array Specification for Data Type of DB Map Variables by SQL Type

Whether you can specify a structure array for DB Map Variables depends on SQL type.
The following table shows the details.

SQL type	Specifying a structure array for DB Map Variable
INSERT	Not possible
UPDATE	
SELECT	Possible

Mapping Cannot be Created for a DB Map Variable

Mapping cannot be created for a DB Map Variable in the following cases. The DB_CreateMapping (Create DB Map) instruction ends in an error.

- When the data type of the DB Map Variable is not a structure
- When a derivative data type is contained in structure members of the DB Map Variable
- When a structure array is specified for a DB Map Variable though INSERT or UPDATE is specified for the SQL type in the instruction.

An Error Occurs when a Record Processing Instruction is Executed

No error is detected when a mapping is created for a DB Map Variable by executing a DB_CreateMapping (Create DB Map) instruction. The DB_Insert (Insert DB Record), DB_Update (Update DB Record), or DB_Select (Retrieve DB Record) instruction will result in an error.

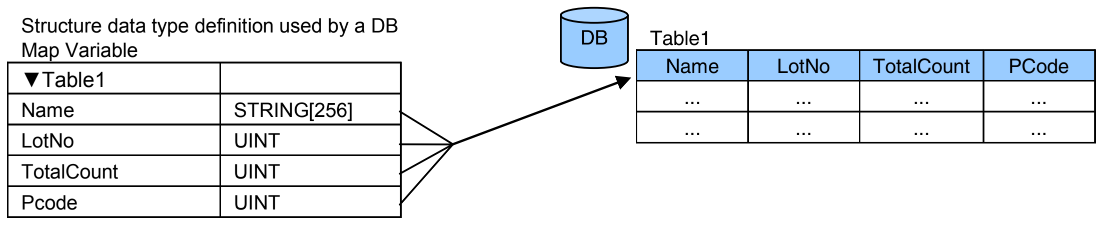
- When the DB cannot be connected
- When the specified table does not exist in the DB
- When a member name of the DB Map Variable does not match a column in the table
- When a member's data type does not match the data type of the corresponding column

3-4 Specifying the Table and Applying the Mapping

You need to create a mapping from a DB Map Variable to the DB for each SQL type (INSERT, UPDATE, and SELECT) before you can execute a record processing instruction (for inserting, updating, or retrieving records).
This section describes how to create and clear the DB mapping and restrictions.

3-4-1 DB Mapping by Executing a Create DB Map Instruction

Execute a DB_CreateMapping (Create DB Map) instruction for mapping a DB Map Variable to the connected DB. Specify the Table Name, DB Map Variable, and SQL Type in the DB_CreateMapping (Create DB Map) instruction.
By doing so, you can map the DB Map Variable to the DB for each SQL type (i.e., INSERT, UPDATE, and SELECT).
Refer to the explanation for DB_CreateMapping (Create DB Map) instruction in Appendix.



▼Table1	
Name	STRING[256]
LotNo	UINT
TotalCount	UINT
Pcode	UINT

Name	LotNo	TotalCount	PCode
...	...	...	...
...	...	...	...

3-4-2 Clearing the Mapping of DB Map Variables

Mapping of DB Map Variables is automatically cleared by the following operations.

- When the DB Connection is closed
- When the DB Connection Service is stopped*
- When the DB Connection Service is shut down
- When another mapping is applied to the DB Map Variable (i.e. mapping to a different table or for a different SQL type)

* Refer to *4-1-3 DB Connection Service is Stopped or Cannot be Started* for details on the stop of the DB Connection Service.

Precautions for Correct Use

Mapping to the DB is automatically cleared when the DB Connection is closed.
Therefore, write the user program so that a DB_Connect (Establish DB Connection) instruction is executed before a DB_CreateMapping (Create DB Map) instruction.

3-4-3 Restrictions on DB Mapping

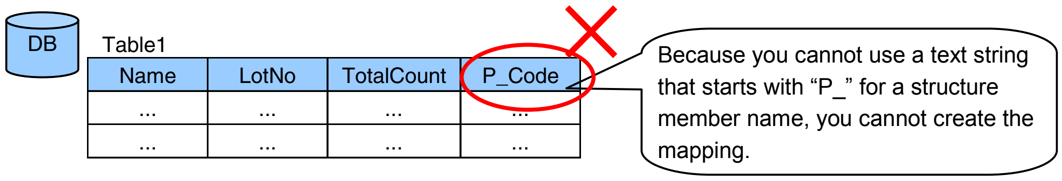
The DB mapping has the following restrictions.

• Restrictions on Table's Column Names:

When a character that cannot be specified for structure member names is used in a column name of the table, you cannot create the mapping. You need to change the column name of the table.

Example:

When a column name is *P_Code*

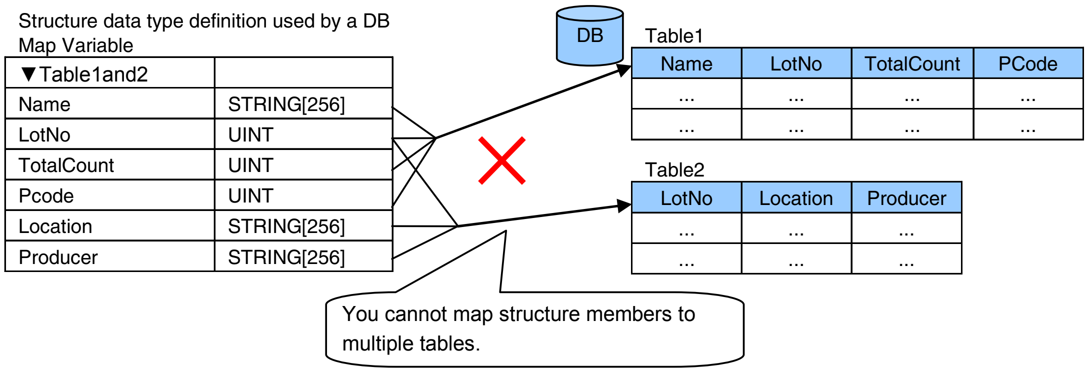

DB
Table1
Name
LotNo
TotalCount
P_Code
Because you cannot use a text string that starts with "P_" for a structure member name, you cannot create the mapping.

Refer to *Precautions for Correct Use: Restrictions on Table's Column Names of 3-2-2 Specifications of Structure Data* Type for DB Access for the characters that cannot be specified for structure member names.

• Restrictions on Mapping to Multiple Tables:

You cannot map the members of a DB Map Variable to columns of different tables.

Example:

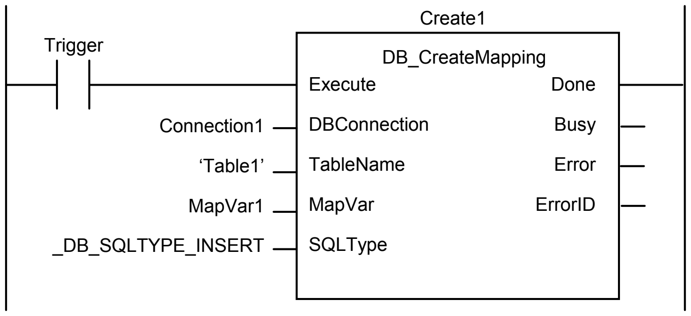


• Restrictions on Mapping to Multiple Tables:

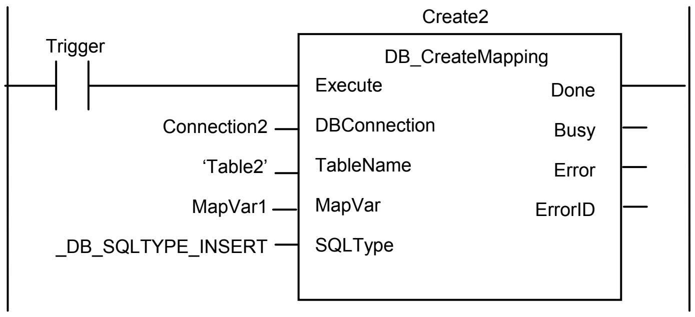
You cannot map a DB Map Variable to two or more tables.

If you execute multiple DB_CreateMapping (Create DB Map) instructions so as to map a single DB Map Variable to two or more tables, the mapping made by the last DB_CreateMapping (Create DB Map) instruction takes effect.

Example:



Members of *MapVar1* variable are mapped with columns of Table1.



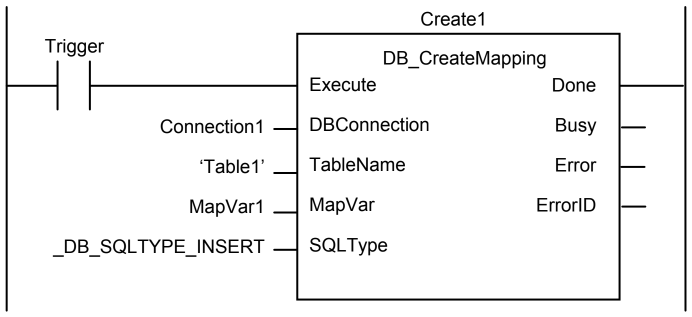
Mapping members of *MapVar1* variable with columns of Table1 of Connection1 is cleared. Members of *MapVar1* variable are mapped with columns of Table2 of Connection2.

• Restrictions on Mapping to Multiple SQL Types

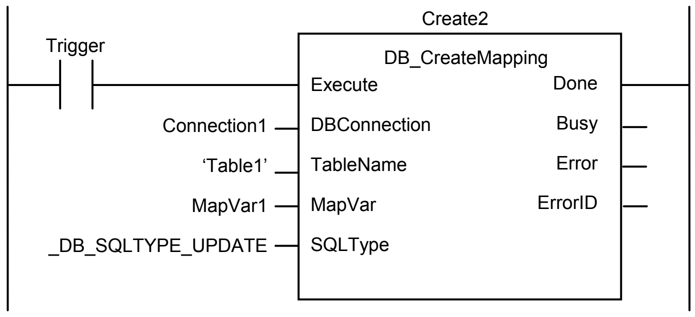
You cannot map a DB Map Variable for two or more SQL types.

If you execute multiple DB_CreateMapping (Create DB Map) instructions so as to map a single DB Map Variable for two or more SQL types, the mapping made by the last DB_CreateMapping (Create DB Map) instruction takes effect.

Example:



Members of *MapVar1* variable are mapped with columns of Table1.



Mapping members of *MapVar1* variable with columns of Table1 for INSERT is cleared. Members of *MapVar1* variable are mapped with columns of Table1 for UPDATE.

• Number of DB Map Variables for which Mapping can be Created

The total number of DB Map Variables for which you can create a mapping in all connections depends on the database type to connect. Refer to *1-2-1 DB Connection Service Specifications* for the number of DB Map Variables supported for each DB. When the upper limit is exceeded, an instruction error (Data Capacity Exceeded) will occur when a DB_CreateMapping (Create DB Map) instruction is executed.

However, even if the number of DB Map Variables has not reached the upper limit, an instruction error (Data Capacity Exceeded) will occur when the following condition is met.

- When the total number of members of structures used as data type of DB Map Variables in all DB Connections exceeds 10,000 members

・Definition of DB Map Variables
When a DB_Insert (Insert DB Record), DB_Update (Update DB Record), DB_Select (Retrieve DB Record) instruction is executed in a POU instance that is different from the POU instance where the DB_CreateMapping (Create DB Map) instruction is executed, the DB Map Variable needs to be a global variable.

3-5 Programming and Transfer

This section describes how to program the DB Connection Service, DB Connection Instruction set, and system-defined variables.
Refer to *Sample Programming* of each DB Connection Instruction given in Appendix for programming examples.

3-5-1 Programming the DB Connection Service

Use the following procedure to program the DB Connection Service.

1. Select a DB Connection Instruction from the DB Connect instruction category of the Toolbox to the right of the program editor of Sysmac Studio.
Write the DB Connection Instructions in the following order.

1. Initial Processing

1-1. Write a DB_ControlService (Control DB Connection Service) instruction when you start the DB Connection Service using the instruction*.

* This instruction is not required if the DB Connection Service is automatically started when the operating mode of the CPU Unit is changed to RUN mode.

1-2. Write a DB_Connect (Establish DB Connection) instruction.

1-3. Write a DB_CreateMapping (Create DB Map) instruction.

2. Processing during Operation[*1]

2-1. Write a DB_Insert (Insert DB Record), DB_Update (Update DB Record), DB_Select (Retrieve DB Record), or other instruction.

3. End Processing

3-1. Write a DB_Close (Close DB Connection) instruction.

4. Power OFF Processing[*2]

4-1. Write a DB_Shutdown (Shutdown DB Connection Service) instruction.

*1 When you continuously execute DB_Insert (Insert DB Record), DB_Update (Update DB Record), DB_Select (Retrieve DB Record), and other instructions, repeat the *2. Processing during Operation*.

*2 Be sure to execute a DB_Shutdown (Shutdown DB Connection Service) instruction before you turn OFF the power supply to the system. If the power supply is turned OFF without executing a DB_Shutdown (Shutdown DB Connection Service) instruction, the Operation Log file may be corrupted or its contents may be lost.

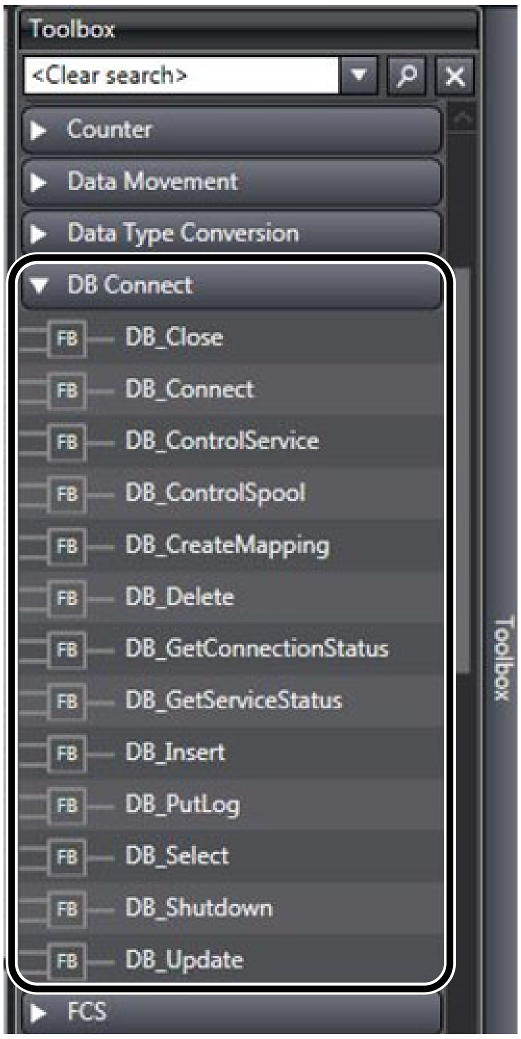
2. Check the status of the DB Connection Service with a system-defined variable.
The status can be *Running in Operation Mode*, *Running in Test Mode*, *Idle*, *Error*, or *Shutdown*.

3. Transfer the DB Connection settings and user program.
Transfer the DB Connection settings and user program to an NJ-series CPU Unit.

4. Cycle the power supply to the Controller.
When you have changed the database type to connect, cycle the power supply to the Controller.

3-5-2 Displaying DB Connection Instructions on Sysmac Studio

The DB Connection Instructions are displayed in the DB Connect instruction category of Toolbox of Sysmac Studio.



Toolbox
<Clear search>
▶ Counter
▶ Data Movement
▶ Data Type Conversion
▼ DB Connect
FB DB_Close
FB DB_Connect
FB DB_ControlService
FB DB_ControlSpool
FB DB_CreateMapping
FB DB_Delete
FB DB_GetConnectionStatus
FB DB_GetServiceStatus
FB DB_Insert
FB DB_PutLog
FB DB_Select
FB DB_Shutdown
FB DB_Update
▶ FCS

3-5-3 DB Connection Instruction Set

The following set of DB Connection Instructions is supported.

Instruction	Name	Function
DB_Connect	Establish DB Connection	Connects to a specified DB.
DB_Close	Close DB Connection	Closes the connection with the DB established by a DB_Connect (Establish DB Connection) instruction.
DB_CreateMapping	Create DB Map	Creates a mapping from a DB Map Variable to a table of a DB.
DB_Insert	Insert DB Record	Inserts values of a DB Map Variable to a table of the connected DB as a record.
DB_Update	Update DB Record	Updates the values of a record of a table with the values of a DB Map Variable.
DB_Select	Retrieve DB Record	Retrieves records from a table to a DB Map Variable.
DB_Delete	Delete DB Record	Deletes the records that match the conditions from a specified table.
DB_ControlService	Control DB Connection Service	Starts/stops the DB Connection Service or starts/finishes recording to the Debug Log.
DB_GetServiceStatus	Get DB Connection Service Status	Gets the current status of the DB Connection Service.
DB_GetConnectionStatus	Get DB Connection Status	Gets the status of a DB Connection.
DB_ControlSpool	Resend/Clear Spool Data	Resends or clears the SQL statements spooled by DB_Insert (Insert DB Record) and DB_Update (Update DB Record) instructions.
DB_PutLog	Record Operation Log	Puts a user-specified record into the Execution Log or Debug Log.
DB_Shutdown	Shutdown DB Connection Service	Shuts down the DB Connection Service.

* Be sure to execute a DB_Shutdown (Shutdown DB Connection Service) instruction before you turn OFF the power supply to the system. If the power supply is turned OFF without executing a DB_Shutdown (Shutdown DB Connection Service) instruction, the Operation Log file may be corrupted or its contents may be lost.

Refer to *Appendix DB Connection Instructions* for details and sample programming of each instruction.

3-5-4 System-defined Variables

You can use the following system-defined variable in the DB Connection Service.

Variable name	Member name	Data type	Meaning	Function	Initial value
_DBC_Status		_sDBC_STATUS	DB Connection Service Status	Shows the operation status of the DB Connection Service. Refer to *4-3-1 Operation Status of the DB Connection Service* for details on the operation status of the DB Connection Service.	
	Run	BOOL	Running flag	TRUE when the DB Connection Service is running in Operation Mode or Test Mode.	FALSE
	Test	BOOL	Test Mode	TRUE when the DB Connection Service is running in Test Mode.	FALSE
	Idle	BOOL	Idle	TRUE when the operation status of the DB Connection Service is *Idle*.	FALSE
	Error	BOOL	Error Stop Flag	TRUE when the operation status of the DB Connection Service is *Error*.	FALSE
	Shutdown	BOOL	Shutdown	TRUE when the operation status of the DB Connection Service is *Shutdown*.	FALSE

3-5-5 Simulation Debugging of DB Connection Instructions

You can perform operation check of the user program using the Simulation function of Sysmac Studio.

The DB Connection Instructions perform the following operations during simulation.

- The DB_Connect, DB_Close, DB_Insert, and other instructions that do not retrieve data will end normally.
- The DB_Select and other instructions that retrieve data will end normally as if there was no applicable data.

3-5-6 Transferring the DB Connection Settings and User Program

You transfer the DB Connection settings and user program to an NJ-series CPU Unit using the Synchronization function of Sysmac Studio.

You can specify the following comparison unit for the DB Connection Service in the Synchronization Window.

Synchronization data name	Level	Number	Detailed comparison	Remarks
Host Connection Settings	2	1	Not supported	
DB Connection	3	1	Not supported	
DB Connection Service Settings	4	1	Not supported	
DB Connection Settings	4	1	Not supported	

The DB Connection settings are reflected when the DB Connection Service is started.

Precautions for Correct Use

If an operation failure or communications error occurs when you execute an operation from Sysmac Studio, retry the operation after performing the following:

- Check the cable connection.
- Check the communications settings.
- Increase the response monitoring time in the Communications Setup.
- Increase the system service execution time ratio.
- Check that the operation status of the DB Connection Service is not *Initializing*, *Error*, or *Shutdown*.

For details of the operation status of the DB Connection Service, refer to *4-3-1 Operation Status of the DB Connection Service*.

When Sysmac Studio cannot go online, refer to the *NJ/NX-series Troubleshooting Manual* (Cat. No. W503).

3-6 Debugging in Design, Startup, and Operation Phases

You can use the following debugging procedures according to the phase and actual device environment.

3-6-1 Design Phase

This section gives the debugging procedure in the design phase.

Actual device environment		Debugging method	
CPU Unit	DB	Check item	Operation
Exist	Not exist, or not connected	Checking the executions of DB Connection Instructions on the physical CPU Unit	・Start the DB Connection Service in Test Mode. ・Execute DB Connection Instructions. Note In Test Mode, SQL statements are not sent actually, but the processing ends as if they were sent normally. ・Check the Operation Logs (i.e., Execution Log and Debug Log).

3-6-2 Startup Phase

This section gives the debugging procedure in the startup phase.

Actual device environment		Debugging method	
CPU Unit	DB	Check item	Operation
Exist	Connected	Connection to the DB	・Start the DB Connection Service in Operation Mode. ・Check the status of the DB Connection Service and each DB Connection from Sysmac Studio.
		Checking the DB read/write and timing	・Execute DB Connection Instructions. ・Check the Operation Logs (i.e., Execution Log, Debug Log, and SQL Execution Failure Log). (including the check of connection to the DB, executions of SQL statements, and responses)

3-6-3 Operation Phase

This section gives the troubleshooting procedure in the operation phase.

Actual device environment		Debugging method	
CPU Unit	DB	Check item	Operation
Exist	Connected	Regular check	• Check the event logs. • Check the Operation Logs (i.e., Execution Log and SQL Execution Failure Log). • Check the status of the DB Connection Service and each DB Connection from Sysmac Studio. • Check the status of the DB Connection Service and each connection using a DB Connection Instruction.

4

Basic Operations and Status Check

This section describes how to start and stop the DB Connection Service, how to establish and close a DB Connection, and how to check the status of the DB Connection Service and each DB Connection.

4-1 Run Mode of DB Connection Service and Start/Stop Procedures

This section describes the Run mode of the DB Connection Service and start/stop procedures.

4-1-1 Run Mode of the DB Connection Service

The DB Connection Service has two Run modes, Operation Mode and Test Mode. You can change the Run mode according to whether to actually access the DB.
This section describes the operations and usage of each Run mode of the DB Connection Service.

Run Mode of the DB Connection Service

You can change the Run mode according to the purpose. In Test Mode, you can test the operations of the DB Connection Service without connecting to the DB. In Operation Mode, you can perform practical operation or trial operation by connecting to the DB.

Run mode	Description	Usage	Environment
Test Mode	・SQL statements are not sent to the DB when DB Connection Instructions are executed. ・DB Connection Instructions end normally. However, the instructions for retrieving from the DB do not output anything to the specified DB Map Variable. ・Spool function is disabled.	Operation check of user program using DB Connection Instructions when the DB is not connected.	When the DB does not exist, or when the DB exists, but not connected
Operation Mode	・SQL statements are sent to the DB when DB Connection Instructions are executed. ・Spool function is enabled.	Practical or trial operation of the system when the DB is connected	When the DB is connected

4-1-2 How to Start/Stop the DB Connection Service

You can use the following three methods to start or stop the DB Connection Service.
・Starting the service automatically when the operating mode of the CPU Unit is changed to RUN mode.
・Starting/stopping the service by online operation from Sysmac Studio.
・Executing a DB_ControlService (Control DB Connection Service) instruction.

Please note that the Run mode of the DB Connection Service cannot be changed while the service is running. To change the Run mode, you need to stop the DB Connection Service, and then start the service again.

Starting the Service Automatically when Operating Mode of the CPU Unit is Changed to RUN Mode

Double-click **DB Connection Service Settings** under **Configurations and Setup** - **Host Connection Settings** - **DB Connection** in the Multiview Explorer. Then, set *Service start in Run mode* to *Auto start (Operation Mode)* or *Auto start (Test Mode)* in the Service Settings. (Default: *Auto start (Operation Mode*))

When the operating mode of the CPU Unit is changed from PROGRAM mode to RUN mode, the DB Connection Service is automatically started.

Precautions for Correct Use

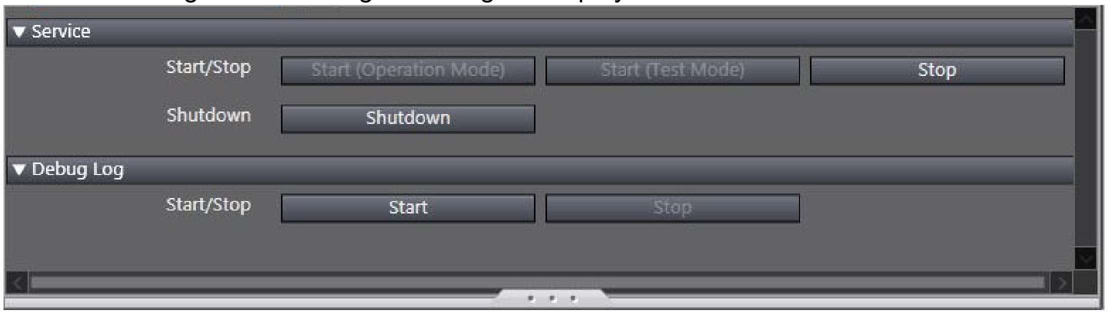
Even if you set *Auto Start* for the DB Connection Service, you cannot execute the DB Connection Instructions until the startup processing of the DB Connection Service is completed. An Instruction Execution Error will occur.
Therefore, write the user program so that the DB Connection Instructions are executed after confirming the status of the DB Connection Service is *Running* with the *_DBC_Status.Run* system-defined variable (Running flag of the DB Connection Service Status).

User program example:

```
IF _DBC_Status.Run = FALSE THEN
    RETURN;    (* Abort the processing because the DB Connection Service is not running *)
END_IF;
(* Execution of DB Connection Instructions *)
(Omitted after this)
```

Starting/Stopping the Service by Online Operation from Sysmac Studio

1. Right-click **DB Connection Service Settings** under **Configurations and Setup** - **Host Connection Settings** - **DB Connection** in the Multiview Explorer and select ***Online Settings*** from the menu while online with an NJ-series CPU Unit.
   The following Online Settings Tab Page is displayed.

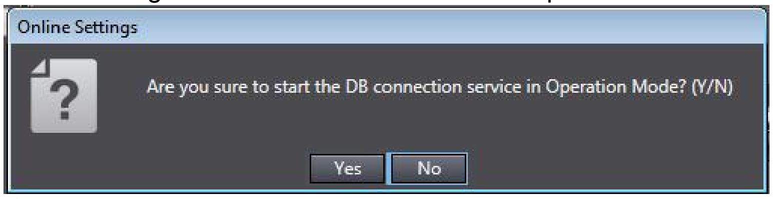


You can start or stop the DB Connection Service by clicking a button.

Category	Item	Button	Operation
Service	Start/Stop	Start (Operation Mode)	The DB Connection Service is started in Operation Mode.
		Start (Test Mode)	The DB Connection Service is started in Test Mode.
		Stop	The DB Connection Service is stopped.

2. To start the DB Connection Service:
Click the **Start (Operation Mode)** or **Start (Test Mode)** Button.
To stop the DB Connection Service:
Click the **Stop** Button.

A confirmation message is displayed. The following is an example dialog box to be displayed when starting the DB Connection Service in Operation Mode.



Are you sure to start the DB connection service in Operation Mode? (Y/N)

3. Click the **Yes** Button.

Note You can start or stop the DB Connection Service regardless of the operating mode of the CPU Unit.

Additional Information

You can shut down the DB Connection Service by clicking the **Shutdown** Button. Refer to *5-2 DB Connection Service Shutdown Function* for details.

Executing a DB_ControlService (Control DB Connection Service) Instruction

Specify one of the following commands in the *Cmd* input variable of the DB_ControlService (Control DB Connection Service) instruction.
- Start the service in Operation Mode
- Start the service in Test Mode
- Stop the service

Refer to *Appendix DB Connection Instructions* for details of the DB_ControlService (Control DB Connection Service) instruction.

4-1-3 DB Connection Service is Stopped or Cannot be Started

In the following conditions, the DB Connection Service cannot be started or the service is stopped.

● DB Connection Service cannot be Started

The DB Connection Service cannot be started in the following cases.
- When the DB Connection Service settings are invalid
- When the operation status of the DB Connection Service is *Initializing*.
- When the operation status of the DB Connection Service is *Shutdown*.

● DB Connection Service is Stopped

The DB Connection Service is stopped in the following cases.

- When the DB Connection Service is stopped by a DB_ControlService (Control DB Connection Service) instruction or Sysmac Studio.
- When the operating mode of the CPU Unit is changed to PROGRAM mode.
- When the Synchronization (download) operation is executed (regardless of whether the DB Connection settings are transferred)
- When the Clear All Memory operation is executed
- When the Restore Controller operation is executed from Sysmac Studio
- When a major fault level Controller error has occurred
- When the DB Connection Service is shut down

Additional Information

- If you stop the DB Connection Service when it is waiting for a response from the DB after sending an SQL statement, the DB Connection Service is stopped after it receives the response from the DB or a communications error is detected.
- If a DB Connection has been established when the DB Connection Service is stopped, the DB Connection is closed.

4-1-4 Changing the Run Mode of the DB Connection Service

You cannot change the Run mode of the DB Connection Service between Operation Mode and Test Mode while the service is running.

To change the Run mode, stop the DB Connection Service and then start the service again.

4-2 Establishing/Closing a DB Connection

After starting the DB Connection Service, you establish or close a DB Connection using an instruction as shown below.

● Establishing a DB Connection

Use a DB_Connect (Establish DB Connection) instruction to establish a DB Connection with a specified name.

Precautions for Correct Use

Mapping to the DB is automatically cleared when the DB Connection is closed. Therefore, write the user program so that a DB_Connect (Establish DB Connection) instruction is executed before a DB_CreateMapping (Create DB Map) instruction is executed.

● Closing a DB Connection

Specify the DB Connection name given in the DB_Connect (Establish DB Connection) instruction in a DB_Close (Close DB Connection) instruction and execute the instruction.

Refer to *Appendix DB Connection Instructions* for details of each instruction.

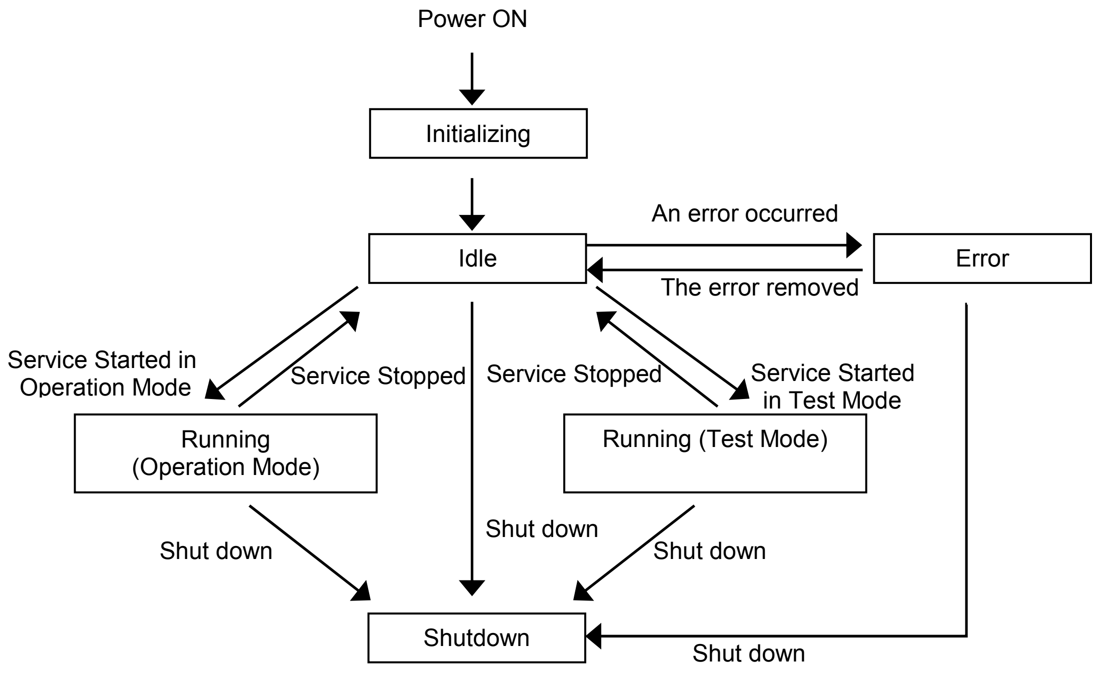
4-3 Checking the Status of DB Connection Service and each DB Connection

This section describes how to check the following status.

- DB Connection Service
- Each DB Connection

4-3-1 Operation Status of the DB Connection Service

This section describes the operation status of the DB Connection Service.



The DB Connection Service has six operation statuses, *Initializing*, *Idle*, *Running (Operation Mode), Running (Test Mode)*, *Error*, *Shutdown*.

After the power supply to the CPU Unit is turned ON, the DB Connection Service enters the *Initializing* status. When the initialization processing is completed, the service enters the *Idle* status. If the DB Connection Service settings are invalid in the *Idle* status, the service enters the *Error* status. When the error is removed, the service returns to the *Idle* status.

When the DB Connection Service is started, the service enters the *Running (Operation Mode)* or *Running (Test Mode)* status according to the Run mode of the DB Connection Service.

When the DB Connection Service is stopped in the *Running (Operation Mode)* or *Running (Test Mode)* status, the service enters the *Idle* status.

When the DB Connection Service shutdown function is executed, the service enters the *Shutdown* status.

The following table gives the details of each status.

Status	Description	Remarks
Initializing	The DB Connection Service was started but has not entered the *Idle* status after the power supply to the CPU Unit was turned ON.	The DB Connection Service cannot be started.

Status	Description	Remarks
Idle	The DB Connection Service is not running without having any error.	The DB Connection settings can be changed. The DB Connection Instructions cannot be executed.
Running (Operating Mode)	The DB Connection Service is running in Operation Mode.	The DB Connection settings cannot be changed. The DB Connection Instructions can be executed.
Running (Test Mode)	The DB Connection Service is running in Test Mode.	The DB Connection settings cannot be changed. The DB Connection Instructions can be executed (, but SQL statements are not sent to the DB).
Error	The DB Connection Service cannot run due to an error.	The status changes to *Error* in the following case. ・ When the DB Connection Service settings are invalid.
Shutdown	The DB Connection Service is already shut down.	The status changes to *Shutdown* when the DB Connection Service is shut down by an instruction or Sysmac Studio operation. After the shutdown processing of the DB Connection Service is completed, you can safely turn OFF the power supply to the CPU Unit. You cannot start the DB Connection Service again until you execute the Reset Controller operation or cycle the power supply to the CPU Unit.

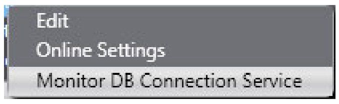
4-3-2 Checking the Status of the DB Connection Service

You can use the following methods to check the status of the DB Connection Service.

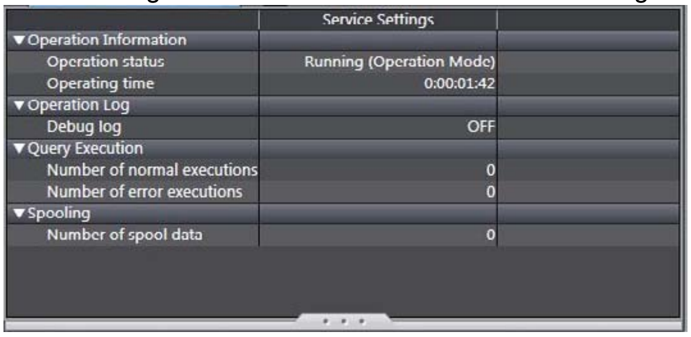
- DB Connection Service Monitor of Sysmac Studio
- DB_GetServiceStatus (Get DB Connection Service Status) instruction
- System-defined variable

Checking the Status with DB Connection Service Monitor of Sysmac Studio

Right-click **DB Connection Service Settings** under **Configurations and Setup** - **Host Connection Settings** - **DB Connection** in the Multiview Explorer and select ***Monitor DB Connection Service*** from the menu while online with an NJ-series CPU Unit.



The following DB Connection Service Monitor Tab Page is displayed.



	Service Settings	
▼Operation Information		
Operation status	Running (Operation Mode)	
Operating time	0:00:01:42	
▼Operation Log		
Debug log	OFF	
▼Query Execution		
Number of normal executions	0	
Number of error executions	0	
▼Spooling		
Number of spool data	0	

You can check the following in the monitor unless the operation status of the DB Connection Service is *Initializing* or *Shutdown*.

Category	Item	Description	Values
Operation Information	Operation status	Operation status of the DB Connection Service.	- Running (Operation Mode) - Running (Test Mode) - Idle - Error Refer to *4-3-1 Operation Status of the DB Connection Service.*
	Operating time	Time elapsed since the DB Connection Service was started.	Duration (Unit: d:h:m:s)
Operation Log	Debug log	ON while the Debug Log is recorded.*	ON/OFF
Query Execution	Number of normal executions	Total number of times in all connections when an SQL statement is normally executed. Including the number of times when a spooled SQL statement is resent. This value is cleared when the DB Connection Service is started.	Number of normal executions
	Number of error executions	Total number of times in all connections when an SQL statement execution failed. This is the number of times when an SQL statement is not spooled, but discarded. The number of times when a statement is spooled is not included. This value is cleared when the DB Connection Service is started.	Number of error executions
Spooling	Number of spool data	Number of spooled SQL statements in all connections.	Number of Spool data

* The *Debug log* flag remains ON even if recording to the log is stopped in the following cases.
- When the *When the log is full* parameter is set to *Stop logging* in the Service Settings, and the maximum number of files is reached
- When the SD Memory Card capacity is insufficient
- When writing to the SD Memory Card failed

Checking the Status using a Get DB Connection Service Status Instruction

You can check the following operation information of the DB Connection Service using a DB_GetServiceStatus (Get DB Connection Service Status) instruction.

Information	Description
Debug Log flag	TRUE while the Debug Log is recorded.*
Operating time	Time elapsed since the DB Connection Service was started. When the DB Connection Service is stopped, the time from start to stop is retained. This value is cleared the next time the DB Connection Service is started.
Number of normal executions	Total number of times in all connections when an SQL statement is normally executed. Including the number of times when a spooled SQL statement is resent. This value is cleared when the DB Connection Service is started.
Number of error executions	Total number of times in all connections when an SQL statement execution failed. This value is cleared when the DB Connection Service is started.
Number of Spool data	Number of spooled SQL statements in all connections.

* The *Debug log* flag remains TRUE even if recording to the log is stopped in the following cases.
- When the *When the log is full* parameter is set to *Stop logging* in the Service Settings, and the maximum number of files is reached
- When the SD Memory Card capacity is insufficient
- When writing to the SD Memory Card failed

Checking the Status with a System-defined Variable

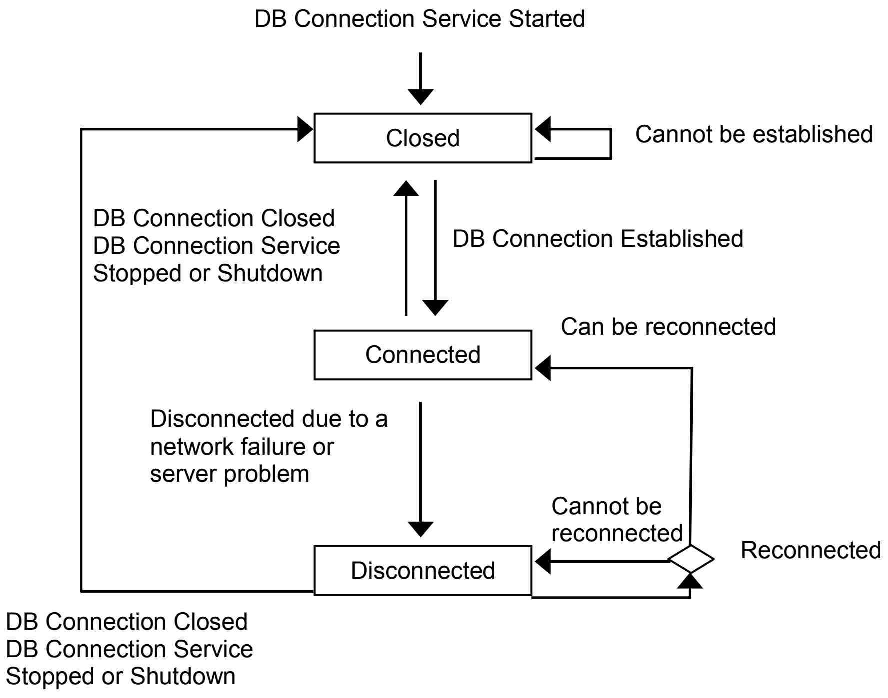
You can check the operation status of the DB Connection Service with the *_DBC_Status* system-defined variable.

Use this variable when checking the status of the DB Connection Service from the user program or checking the shutdown of the DB Connection Service from an HMI.

_DBC_Status system-defined variable		Status					
Member	Meaning	Initializing	Running (Operation Mode)	Running (Test Mode)	Idle	Error	Shut down
Run	Running flag	FALSE	TRUE	TRUE	FALSE	FALSE	FALSE
Test	Test mode	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE
Idle	Idle	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE
Error	Error stop flag	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE
Shutdown	Shutdown	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE

4-3-3 Connection Status of each DB Connection

This section describes the connection status of each DB Connection.



Each DB Connection has three statuses, *Closed*, *Connected*, and *Disconnected*.
After the DB Connection Service is started, each DB Connection enters the *Closed* status.
When the DB Connection is established in the *Closed* status, the DB Connection enters the *Connecte*d status. If the DB Connection cannot be established, it remains in the *Closed* status.
When a network failure or server problem occurs in the *Connected* status, the DB Connection enters the *Disconnected* status.
The DB Connection tries reconnection periodically in the *Disconnected* status. The DB Connection enters the *Connected* status if the DB can be reconnected and remains in the *Disconnected* status if the DB cannot be reconnected.
The following table gives the details of each status.

Status	Description	Remarks
Closed	The DB is not connected.	
Connected	The DB is connected.	You can execute SQL statements such as INSERT and SELECT using instructions.
Disconnected	The DB was disconnected due to a network failure, server's problem, or other causes.	If the DB Connection enters this status during instruction execution, the SQL statement is spooled. Reconnection is attempted periodically.

4-3-4 Checking the Status of each DB Connection

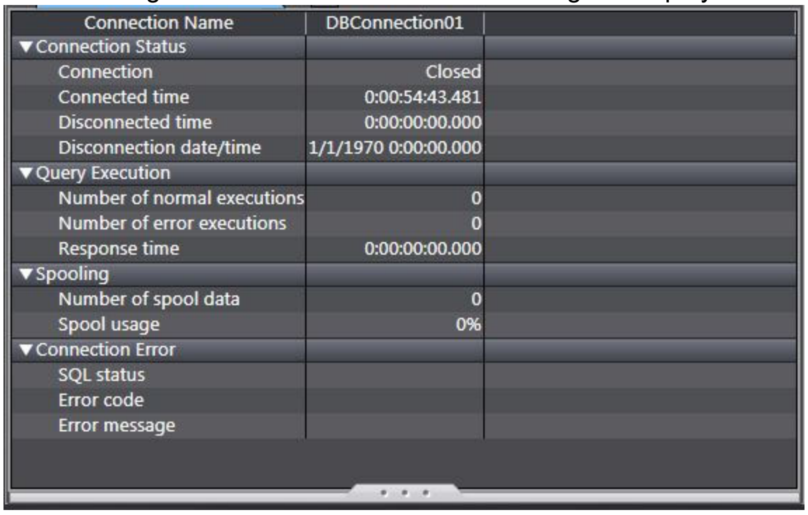
You can use the following methods to check the status of each DB Connection.

- Connection Monitor Table of Sysmac Studio
- DB_GetConnectionStatus (Get DB Connection Status) instruction

Checking the Status with Connection Monitor Table of Sysmac Studio

Right-click **DB Connection Settings** under **Configurations and Setup** - **Host Connection Settings** - **DB Connection** in the Multiview Explorer and select ***Connection Monitor Table*** from the menu while online with an NJ-series CPU Unit.

The following Connection Monitor Table Tab Page is displayed.



Connection Name	DBConnection01
▼Connection Status	
Connection	Closed
Connected time	0:00:54:43.481
Disconnected time	0:00:00:00.000
Disconnection date/time	1/1/1970 0:00:00.000
▼Query Execution	
Number of normal executions	0
Number of error executions	0
Response time	0:00:00:00.000
▼Spooling	
Number of spool data	0
Spool usage	0%
▼Connection Error	
SQL status	
Error code	
Error message	

You can monitor the following of each DB Connection unless the operation status of the DB Connection Service is *Idle* or *Shutdown*.

Category	Item	Description	Values
Connection Status	Connection	Status of the DB Connection.	- Closed - Connected - Disconnected Refer to *4-3-3 Connection Status of each DB Connection*.
	Connected time	Total time when the DB is connected. This value is cleared when *Connection* changes from *Closed* to *Connected*.	Duration (Unit: d:h:m:s.ms)
	Disconnected time	Total time when the DB is disconnected due to an error. This value is cleared when *Connection* changes from *Closed* to *Connected*.	Duration (Unit: d:h:m:s.ms)
	Disconnection date/time	Date and time when the DB is disconnected due to a network failure, server's problem, or other causes.[*1] This value is cleared when the DB Connection Service is started.	Date and time
Query Execution	Number of normal executions	Number of times when an SQL statement is normally executed. Including the number of times when a spooled SQL statement is resent. This value is cleared when the DB Connection Service is started.	Number of normal executions

Category	Item	Description	Values
	Number of error executions	Number of times when an SQL statement execution failed. This is the number of times when an SQL statement is not spooled, but discarded. The number of times when a statement is spooled is not included. This value is cleared when the DB Connection Service is started.	Number of error executions
	Response time	Time elapsed since the CPU Unit sent the SQL statement until the CPU Unit received its SQL execution result in the latest execution of SQL statement[*2]. The response time is stored only when normal response is returned from the DB. If a DB Connection Instruction Execution Timeout has occurred, the response time is not stored when the execution of the instruction is completed (i.e. when the *Error* output variable changes from FALSE to TRUE). The response time is stored when a normal response is returned from the DB after the DB Connection Instruction Execution Timeout occurred. This value is cleared when the DB Connection Service is started.	Duration (Unit: d:h:m:s.ms)
Spooling	Number of spool data	Number of SQL statements stored in the Spool memory.	Number of spool data
	Spool usage	Use rate of the Spool memory for each DB Connection.	Spool usage in percentage (%)
Connection Error	SQL status	Error code defined in SQL Standards (ISO/IEC 9075) to be shown when a network failure or an SQL Execution Error occurred.[*3] The value of the latest error in the connection is stored. This value is cleared when the DB Connection Service is started.	---
	Error code	Error code that is specific to DB vendor to be shown when a network failure or an SQL Execution Error occurred. [*3] When a network error has occurred, 0 is displayed for error code in some cases. When 0 is displayed, check its SQL status. The code of the latest error in the connection is stored. This value is cleared when the DB Connection Service is started.	---
	Error message	Error message that is specific to DB vendor to be shown when a network failure or an SQL Execution Error occurred. [*3] The message of the latest error in the connection is stored. This value is cleared when the DB Connection Service is started.	---

*1 The date and time information follows the time zone set when the power supply to the Controller is turned ON. After you change the time zone, cycle the power supply.

*2 Execution of SQL statement refers to the execution of DB_Insert (Insert DB Record), DB_Update (Update DB Record), DB_Select (Retrieve DB Record), or DB_Delete (Delete DB Record) instruction, or resending of Spool data (automatically or manually by executing a DB_ControlSpool instruction).

*3 The value may differ by unit version of the CPU Unit.
The value of connection error to SQL Server was changed in the unit version 1.08 of the CPU Units.

Checking the Status using a Get DB Connection Status Instruction

You can check the connection status and information of each DB Connection using a DB_GetConnectionStatus (Get DB Connection Status) instruction.

Information		Description
Connection status of the DB Connection		Connection status (Closed, Connected, or Disconnected) of the DB Connection.
Connection information of the DB Connection	Connected time	Total time when the DB is connected. This value is cleared when the status changes from *Closed* to *Connected*.
	Disconnected time	Total time when the DB is disconnected. This value is cleared when the status changes from *Closed* to *Connected*.
	Number of normal executions	Number of times when an SQL statement is normally executed. Including the number of times when a spooled SQL statement is resent. This value is cleared when the DB Connection Service is started.
	Number of error executions	Number of times when an SQL statement execution failed. This is the number of times when an SQL statement is not spooled, but discarded. The number of times when a statement is spooled is not included. This value is cleared when the DB Connection Service is started.
	Number of Spool data	Number of SQL statements stored in the Spool memory. This value returns to 0 when the Spool data is cleared.
	Spool usage	Use rate of the Spool memory for the DB Connection in percentage (%). This value returns to 0 when the Spool data is cleared.
	Disconnection date/time	Date and time when the DB is disconnected due to a network failure, server's problem, or other causes.[*1] This value is cleared when the DB Connection Service is started.
	SQL status	Error code defined in SQL Standards (ISO/IEC 9075) to be shown when a network failure or an SQL Execution Error occurred.[*2] This value is cleared when the DB Connection Service is started.
	Error code	Error code that is specific to DB vendor to be shown when a network failure or an SQL Execution Error occurred. [*2] When a network error has occurred, 0 is displayed for error code in some cases. When 0 is displayed, check its SQL status. This value is cleared when the DB Connection Service is started.
	Error message	Error message that is specific to DB vendor to be shown when a network failure or an SQL Execution Error occurred. [*2] This value is cleared when the DB Connection Service is started.

*1 The date and time information follows the time zone set when the power supply to the Controller is turned ON. After you change the time zone, cycle the power supply.

*2 The value may differ by unit version of the CPU Unit.
The value of connection error to SQL Server was changed in the unit version 1.08 of the CPU Units.

5 Other Functions

This section describes other functions of the DB Connection Service.

5-1 Spool Function

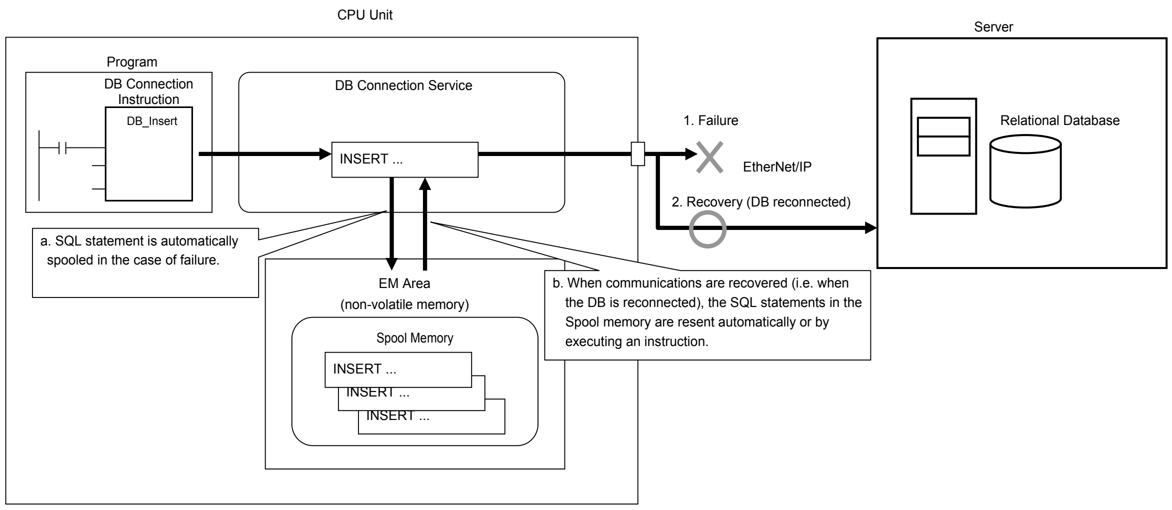
This section describes spooling of unsent SQL statements in the DB Connection Service.

5-1-1 Overview

When a failure occurred in information exchange between DB Connection Service and DB, the unsent SQL statements are stored in a memory area and resent when the problem is solved.
You can set whether to enable or disable the Spool function for each DB Connection.

5-1-2 Spooling System

The following figure shows the spooling system.


CPU Unit
Program
DB Connection Instruction
DB_Insert
DB Connection Service
INSERT ...
a. SQL statement is automatically spooled in the case of failure.
EM Area
(non-volatile memory)
Spool Memory
INSERT ...
INSERT ...
INSERT ...
b. When communications are recovered (i.e. when the DB is reconnected), the SQL statements in the Spool memory are resent automatically or by executing an instruction.
1. Failure
EtherNet/IP
2. Recovery (DB reconnected)
Server
Relational Database

a. When a failure occurred in information exchange between DB Connection Service and DB, the unsent SQL statements are automatically stored in the Spool memory (EM Area).
b. When communications are recovered from the failure and the DB is reconnected, the SQL statements in the Spool memory are resent automatically or by executing an instruction.

5-1-3 Applicable Instructions and Spooling Execution Conditions

Applicable Instructions

The following two instructions are applicable to this function.
- DB_Insert (Insert DB Record) instruction
- DB_Update (Update DB Record) instruction

Precautions for Correct Use

Only the processing for inserting or updating records is spooled. For the other processing, you need to execute the instruction again.

Spooling Execution Conditions

SQL statements are spooled in the following cases.

- When an applicable instruction is executed, the SQL statement cannot be sent due to a network failure.
- When an applicable instruction is executed, the response from the DB cannot be received due to a network failure.
- When an applicable instruction is executed, the DB is stopped due to a server's problem or other causes.
- When an applicable instruction is executed, one or more SQL statements are already stored in the Spool memory.
- When an applicable instruction is executed, a DB Connection Instruction Execution Timeout occurs.

Precautions for Correct Use

- The following error codes are applicable to the spooling execution conditions when the instructions end in an error. When the instructions end in an error with other error codes, the SQL statement is not stored in the Spool memory.
  3011 hex: DB Connection Disconnected Error Status
  3012 hex: DB Connection Instruction Execution Timeout
  3014 hex: Data Already Spooled
  3016 hex: DB in Process
- If an instruction error (SQL Execution Error) occurs, the transmitted SQL statement itself can be the cause of the SQL Execution Error. Therefore, the SQL statement is not stored in the Spool memory because the SQL Execution Error may occur again when the SQL statement is resent.
- Even if a response cannot be received from the DB, the transmitted SQL statement may have been processed in the DB.

5-1-4 Memory Area Used by the Spool Function

The following memory area is used by the Spool function.

Memory area	Description	
EM Area	The unsent SQL statements are stored in the following EM Area. NJ501-□□20: 16 EM banks from No. 9 hex to 18 hex. NJ101-□□20: 3 EM banks from No. 1 hex to 3 hex.	･Total capacity of Spool memory: NJ501-□□20: 1 MB max. NJ101-□□20: 192 KB max. ･Spool capacity for each DB Connection: Total capacity is equally divided by DB Connections for which the Spool function is enabled.

You can prevent losing the Spool data even if a power interruption occurred in the CPU Unit because the EM Area is non-volatile memory.

Precautions for Correct Use

･When the Spool function is enabled, the DB Connection Service uses EM Banks. Please design the system so that the EM Banks used by the DB Connection Service are not used for the following purposes because the Spool data is corrupted if used.
  ･ AT specification of user-defined variables
  ･ I/O memory address specification of tags for tag data link
  ･ Access by communications commands
  ･ Access from HMI
  ･ Specification of Expansion Area words allocated to Special Units for CJ-series Special Units

･The data values in the EM Area are retained by a battery.
If the battery is not mounted or weak, the CPU Unit detects a Battery-backup Memory Check Error. In that case, the Spool data is cleared.

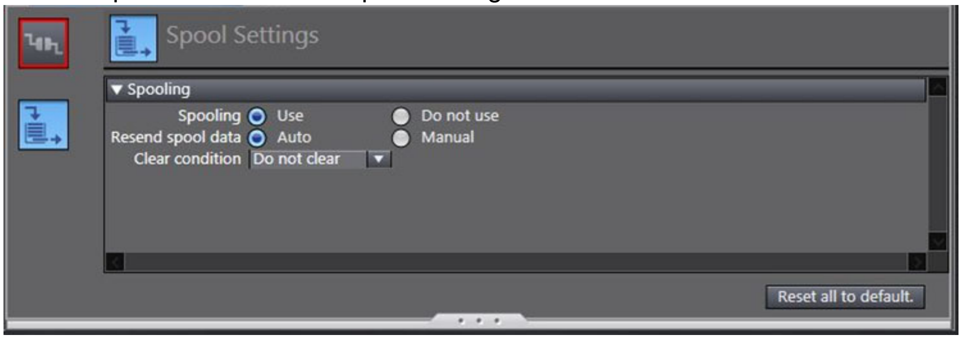
･In the DB Connection settings, the default setting of *Spooling* is *Use*.
If you do not use the Spool function, be sure to set *Spooling* to *Do not use* in the Spool Settings of the DB Connection settings and then download the DB Connection settings when you add a DB Connection.
If you download the DB Connection settings while *Spooling* is set to *Use*, the values stored in the EM banks used by the DB Connection Service will be overwritten by the initialization processing of the Spool function.

･If you select *DM, EM and Holding Memory used for CJ-series Units* for the memory type when backing up or restoring variables or memory on Sysmac Studio, the spool data will be also backed up or restored. If you don't need the spool data after executing a restore operation, clear the SQL statements from the Spool memory. Refer to *5-1-7 Clearing the SQL Statements from the Spool Memory* for the procedure.

5-1-5 Spool Function Settings

Right-click a DB Connection name under **Configurations and Setup** - **Host Connection Settings** - **DB Connection** - **DB Connection Settings** in the Multiview Explorer and select ***Edit*** from the menu.

Set the Spool function in the Spool Settings.



Spool Settings

Spooling

Spooling [✓ Use] Do not use
Resend spool data [✓ Auto] Manual
Clear condition: Do not clear

Set the following items for the Spool function.

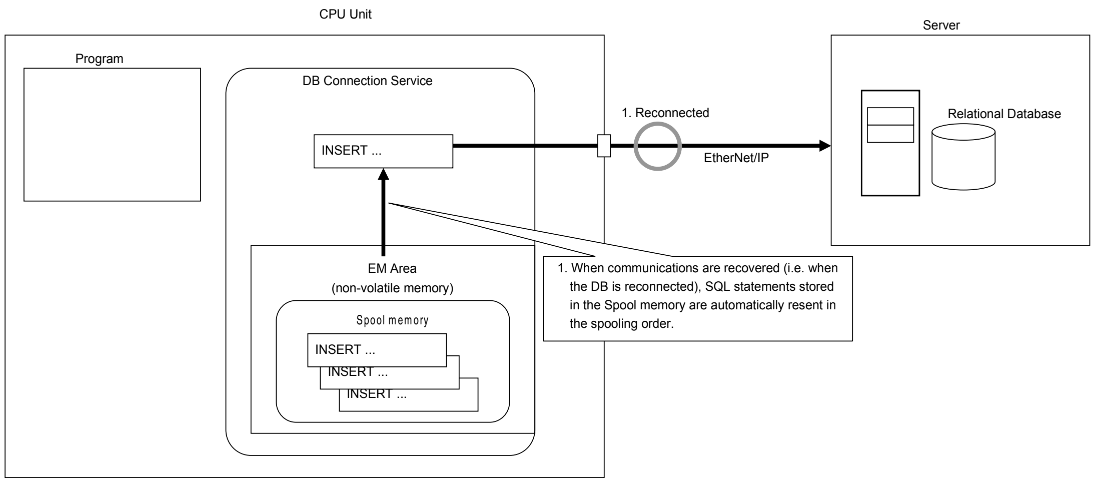
Item	Description	Values
Spooling	Set whether to use the spool function.	• Use (Default) • Do not use
Resend spool data	Set this item when you select *Use* for *Spooling*. Set whether to resend the SQL statements stored in the Spool memory automatically or manually.	• Auto (Default) • Manual
Clear condition	Set this item when you select *Auto* for *Resend spool data*. Set the condition for clearing the SQL statements from the Spool memory.	• Do not clear (Default) • At power ON • When DB connection service started • When DB connection established

5-1-6 How to Resend the SQL Statements Stored in the Spool Memory

You can resend the SQL statements stored in the Spool memory automatically or manually, which can be selected in the *Resend Spool Data* of the Spool Settings.

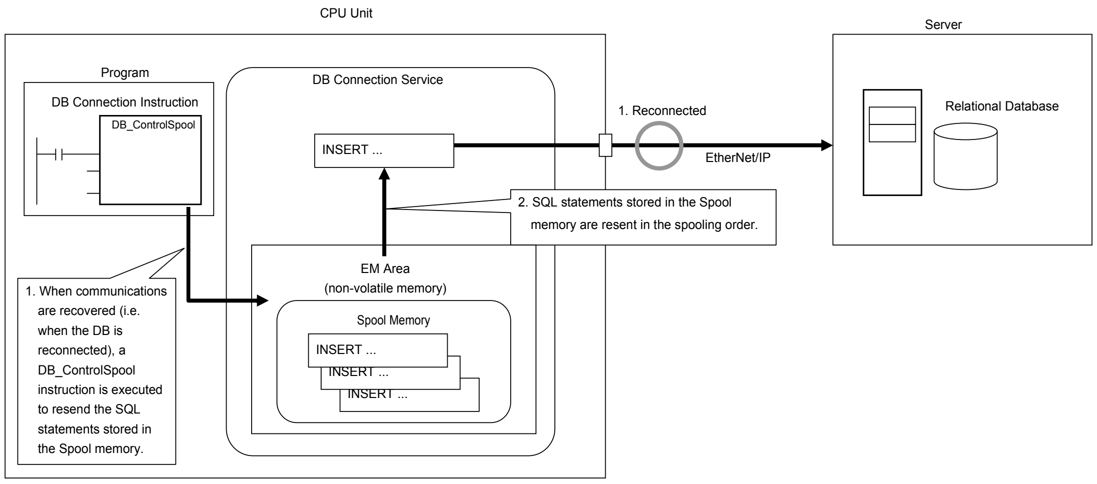
Auto Resend

The SQL statements stored in the Spool memory are automatically resent when the DB is reconnected.



Manual Resend

The SQL statements stored in the Spool memory are resent when a DB_ControlSpool (Resend/Clear Spool Data) instruction is executed.
All of the SQL statements stored in the Spool memory are sent in the spooling order by one execution of the DB_ControlSpool (Resend/Clear Spool Data) instruction.


CPU Unit
Server
Program
DB Connection Instruction
DB_ControlSpool
DB Connection Service
1. Reconnected
Relational Database
INSERT ...
EtherNet/IP
2. SQL statements stored in the Spool memory are resent in the spooling order.
1. When communications are recovered (i.e. when the DB is reconnected), a DB_ControlSpool instruction is executed to resend the SQL statements stored in the Spool memory.
EM Area
(non-volatile memory)
Spool Memory
INSERT ...
INSERT ...
INSERT ...

If a Failure Occurred in Information Exchange with the DB when Resending the SQL Statements

If a failure occurred again when the SQL statements stored in the Spool memory are resent, the unsent SQL statements are kept in the Spool memory. The SQL statements are resent again by auto resend or manual resend. The resend order is not changed.

5-1-7 Clearing the SQL Statements from the Spool Memory

The SQL statements are cleared from the Spool memory in the following cases.
- When the specified clear condition is met.
- When a DB_ControlSpool (Resend/Clear Spool Data) instruction is executed
- When the Clear Spool Data operation is executed from Sysmac Studio
- When the automatic clear condition is met

When the Specified Clear Condition is Met

When *Auto* is selected for *Resend Spool Data* in the Spool Settings, you can set the condition for clearing the SQL statements from the Spool memory for each DB Connection in *Clear condition* under *DB Connection Settings* - *Spool Settings* on Sysmac Studio.
Select from the following options.

Clear condition	Description
Do not clear (Default)	The SQL statements stored in the Spool memory are not cleared.
At power ON	The SQL statements are cleared from the Spool memory when the power supply to the CPU Unit is turned ON.
When DB connection service started	The SQL statements are cleared from the Spool memory when the DB Connection Service is started.
When DB connection established	The SQL statements are cleared from the Spool memory when the DB Connection is established (i.e. when the status changes from *Closed* to *Connected*). If you select this option, the SQL statements are cleared from the Spool memory without being resent.

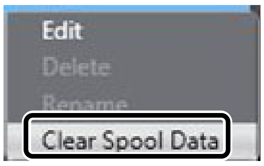
When a DB_ControlSpool (Resend/Clear Spool Data) Instruction is Executed

You can clear the SQL statements from the Spool memory by executing a DB_ControlSpool (Resend/Clear Spool Data) instruction.

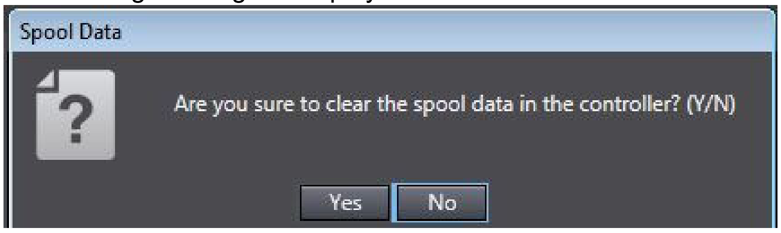
When the Clear Spool Data operation is executed from Sysmac Studio

You can clear the SQL statements from the Spool memory by the following operation from Sysmac Studio.

1. Right-click a DB Connection in the Multiview Explorer and select ***Clear Spool Data*** from the menu while online with an NJ-series CPU Unit.



The following message is displayed.



Are you sure to clear the spool data in the controller? (Y/N)

2. Click the **Yes** Button.

When the automatic clear condition is met

The SQL statements are automatically cleared from the Spool memory regardless of the *Resend spool data* setting in the following cases.

- When you change the DB Connection settings and execute the Synchronization (download) operation on Sysmac Studio.
- When you execute the Clear All Memory operation
- When a Battery-backup Memory Check Error occurred
- When you execute the Restore operation of the SD Memory Card backup function or Sysmac Studio Controller backup function.
- When you restore the memory using the Restore Variables/Memory function of Sysmac Studio

5-1-8 Relationship with the DB Connection Instructions

This section describes the operations of DB Connection Instructions to be performed when one or more SQL statements are already stored in the Spool memory and the impacts to the spooling operations to be performed when an Instruction Execution Timeout occurred for a DB Connection Instruction.

Executing DB Connection Instructions when SQL Statements are Already Stored in the Spool Memory

This section describes the operation to be performed when each DB Connection Instruction is executed for a DB Connection that already has one or more SQL statements in the Spool memory.

Instruction	Operation
DB_Insert (Insert DB Record)	The SQL statement (INSERT) is spooled.* The instruction ends in an error. (Error = TRUE, SendStatus = _DBC_SEND_SPOOLED) Refer to *Appendix DB Connection Instructions* for *ErrorID* of the instruction execution error.
DB_Update (Update DB Record)	The SQL statement (UPDATE) is spooled.* The instruction ends in an error. (Error = TRUE, SendStatus = _DBC_SEND_SPOOLED) Refer to *Appendix DB Connection Instructions* for *ErrorID* of the instruction execution error.
DB_Select (Retrieve DB Record)	The SQL statement (SELECT) is not sent to the DB. An instruction execution error occurs. (Error = TRUE) Refer to *Appendix DB Connection Instructions* for *ErrorID* of the instruction execution error.
DB_Delete (Delete DB Record)	The SQL statement (DELETE) is not sent to the DB. An instruction execution error occurs. (Error = TRUE) Refer to *Appendix DB Connection Instructions* for *ErrorID* of the instruction execution error.

* If the remaining Spool memory area is not enough when the SQL statement is spooled, the SQL statements will be discarded without being stored in the Spool memory.

Instruction	Operation
DB_Insert (Insert DB Record)	The SQL statement (INSERT) is not sent to the DB. An instruction execution error occurs. (Error = TRUE, SendStatus=_DBC_SEND_SENDING) Refer to *Appendix DB Connection Instructions* for *ErrorID* of the instruction execution error.
DB_Update (Update DB Record)	The SQL statement (UPDATE) is not sent to the DB. An instruction execution error occurs. (Error = TRUE, SendStatus=_DBC_SEND_SENDING) Refer to *Appendix DB Connection Instructions* for *ErrorID* of the instruction execution error.

Operations of Instructions and DB Connection Service in the Case of DB Connection Instruction Execution Timeout

When a DB Connection Instruction Execution Timeout occurs, the transmitted SQL statement is stored in the Spool memory. The DB Connection Service waits for a response from the DB for the time set in the *Query execution timeout* parameter plus 10 seconds* after the DB Connection Instruction is executed.
When a response is returned from the DB, the SQL statement stored in the Spool memory is deleted. If no response has been returned from the DB when the time set in the *Query execution timeout* parameter plus 10 seconds* has elapsed, the DB Connection is changed to the *Disconnected* status.

If a DB_Insert (Insert DB Record), DB_Update (Update DB Record), DB_Select (Retrieve DB Record), or DB_Delete (Delete DB Record) instruction is executed while the DB Connection Service is waiting for a response from the DB, an error (DB in Process) occurs for the instruction.

● DB_Insert (Insert DB Record) or DB_Update (Update DB Record) Instruction

If the Spool function is enabled, the SQL statement to send is spooled.
Regardless of the *Resend spool data* setting, the spooled SQL statement is sent after the response to the previous DB Connection Instruction is returned.

● DB_Select (Retrieve DB Record) or DB_Delete (Delete DB Record) Instruction

To execute the DB_Select (Retrieve DB Record) or DB_Delete (Delete DB Record) instruction after the response to the previous DB Connection Instruction is returned, write the user program so that the execution of the DB_Select (Retrieve DB Record) or DB_Delete (Delete DB Record) instruction is retried until it is normally completed.

Precautions for Correct Use

If the time set in the *Query execution timeout* parameter has elapsed after execution of a DB Connection Instruction, a cancel request of the applicable SQL operation is sent to the DB. The details of the SQL operation cancel processing are given below.

(1) When the cancel processing is completed within 10 seconds*:
- The instruction will be terminated due to an error (SQL Execution Error).

(2) When the cancel processing is not completed within 10 seconds*:
- A communications timeout will occur. When the communications timeout has occurred, the instruction will be terminated due to an error (DB Connection Disconnected Error Status) and the DB Connection is changed to the *Disconnected* status.
- In the case of DB_Insert (Insert DB Record) or DB_Update (Update DB Record) instruction, the SQL statement is stored in the Spool memory.
- If resending of Spool data and disconnection of DB Connection occur repeatedly, increase the time set in the *Query execution timeout* parameter or review the SQL operation to make an adjustment so that the communications timeout does not occur. Refer to *5-4 Timeout Monitoring Functions* for timeout monitoring.

* The time differs by the DB type and DB status.

5-1-9 How to Estimate the Number of SQL Statements that Can be Spooled

The number of SQL statements that can be spooled depends on the user program.
This section describes how to estimate the number of SQL statements that can be spooled.

● Calculation of the Number of Bytes of each SQL Statement

You can calculate the number of bytes of each SQL statement as shown below.
You can check the contents of SQL statements with the Debug Log.
Refer to *6-3 Debug Log* for the information on the Debug Log.

Instruction	SQL statement	Calculating formula of the number of bytes of each SQL statement*
DB_Insert (Insert DB Record)	insert into <TableName> (<ColumnName1>, <ColumnName2>, <ColumnName3>..., <ColumnNameN>) values(<Value1>, <Value2>, <Value3>..., <ValueN>)	50 + (Number of bytes of <TableName>) + (Number of bytes of <ColumnName1>) + (2 + Number of bytes of <ColumnName2>) + (2 + Number of bytes of <ColumnName3>) ... +(2 + Number of bytes of <ColumnNameN>) + (Number of bytes of <Value1>) + (2 + Number of bytes of<Value2>) + (2 + Number of bytes of <Value3>) ... +(2+ Number of bytes of <ValueN>)
DB_Update (Update DB Record)	update <TableName> set <ColumnName1>=<Value1>, <ColumnName2>=<Value2>..., <ColumnNameN>=<ValueN> where <RetrievalCondition>	45 + (Number of bytes of <TableName>) + (3 + Number of bytes of <ColumnName1> + Number of bytes of <Value1>) + (5 + Number of bytes of <ColumnName2> + Number of bytes of <Value2>) + (5 + Number of bytes of <ColumnName3> + Number of bytes of <Value3>) ... + (5 + Number of bytes of <ColumnNameN> + Number of bytes of <ValueN>) + (Number of bytes of <RetrievalCondition>)

* Text strings of SQL statements are handled as UTF-8. One byte is used for each single-byte alphanumeric character and multiple bytes are used for each multi-byte character. Three bytes are used for each Japanese character as a guide.

● Calculation of the Number of SQL Statements that Can be Spooled

You can estimate the number of SQL statements that can be spooled using the following formulae.

Number of SQL statements that can be spooled =
Spool capacity per DB Connection (bytes) ÷ *Number of bytes of each SQL statement*

Spool capacity per DB connection (bytes) =
Capacity of the entire Spool memory (1,048,576 bytes for NJ501-□□20 or 196,608 bytes for NJ101-□□20) ÷ *Number of DB Connections for which the Spool function is enabled*

5-2 DB Connection Service Shutdown Function

This section describes the shutdown function of the DB Connection Service to prevent losing the Operation Log data.
Refer to *4-3-1 Operation Status of the DB Connection Service* for the information on the operation status of the DB Connection Service.

5-2-1 Overview

The DB Connection Service shutdown function (hereinafter called “shutdown function”) is used to shut down the DB Connection Service after saving the Operation Log files into the SD Memory Card.
Execute the shutdown function before turning OFF the power supply to the CPU Unit. You can prevent losing the Operation Log data by executing the shutdown function.

Precautions for Correct Use

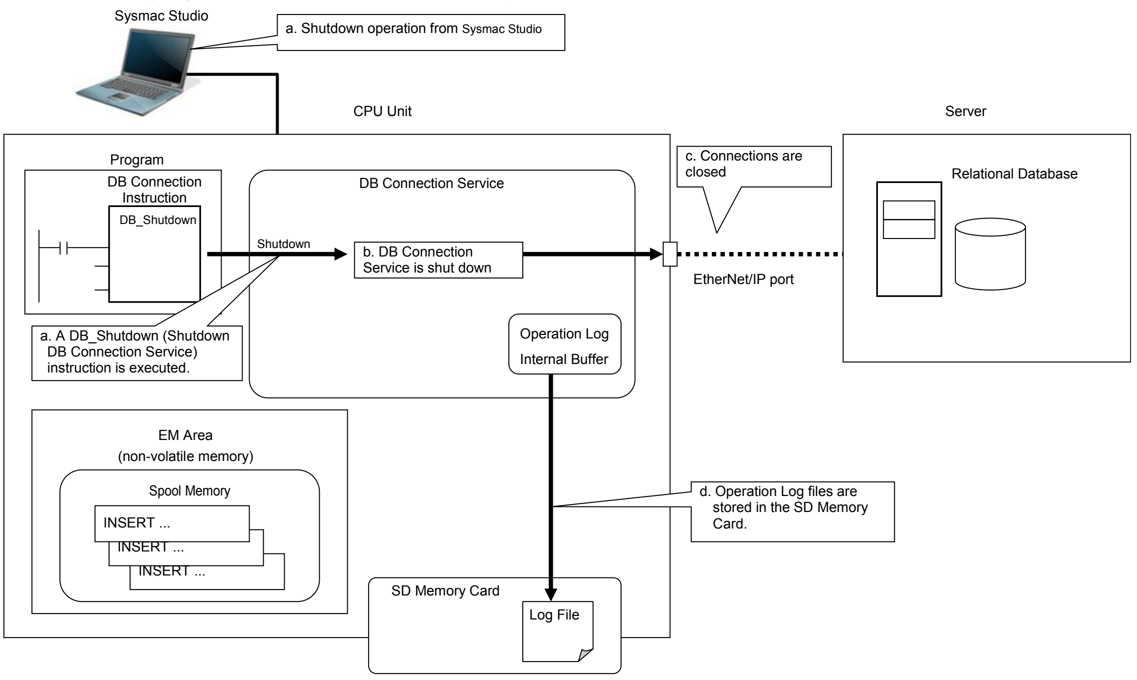
If the power supply to the CPU Unit is turned OFF without executing the shutdown function while the DB Connection Service is running, the contents of the Operation Logs cannot be guaranteed. The Operation Log files may be corrupted or the data may be lost.

Additional Information

We recommended that you take countermeasures against power interruption such as installation of uninterruptible power supply system to prevent data loss by unexpected power interruption.

5-2-2 Shutdown System

The following figure shows the shutdown system.


Sysmac Studio
a. Shutdown operation from Sysmac Studio
CPU Unit
Server
Program
DB Connection Instruction
DB_Shutdown
DB Connection Service
Shutdown
b. DB Connection Service is shut down
c. Connections are closed
EtherNet/IP port
Relational Database
a. A DB_Shutdown (Shutdown DB Connection Service) instruction is executed.
Operation Log Internal Buffer
EM Area (non-volatile memory)
Spool Memory
INSERT ...
INSERT ...
INSERT ...
d. Operation Log files are stored in the SD Memory Card.
SD Memory Card
Log File

a. The DB Connection Service is shut down by a Sysmac Studio operation or by executing a DB_Shutdown (Shutdown DB Connection Service) instruction.
b. The DB Connection Service is shut down.
c. The DB Connections are closed.
d. The Operation Log files (Execution Log files, Debug Log files, and SQL Execution Failure Log files) are stored in the SD Memory Card.

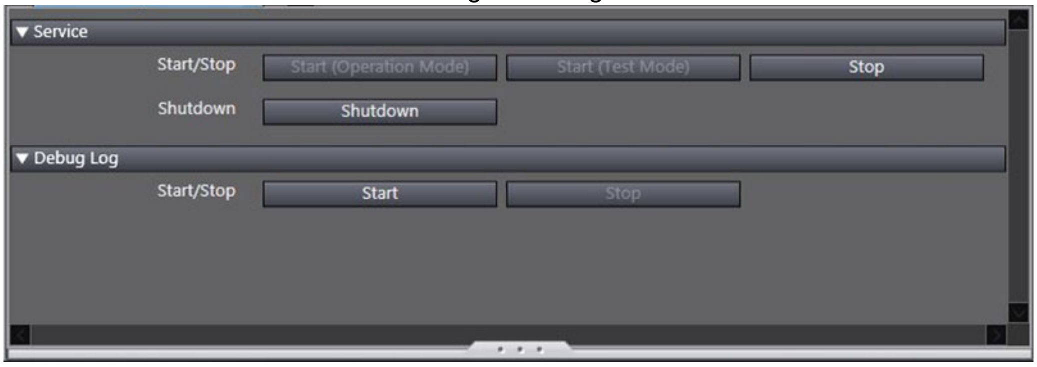
5-2-3 How to Execute the Shutdown Function

You can use the following procedure to execute the shutdown function.

- Sysmac Studio operation
- Instruction execution

Sysmac Studio Operation

Right-click **DB Connection Service Settings** under **Configurations and Setup** - **Host Connection Settings** - **DB Connection** in the Multiview Explorer and select ***Online Settings*** from the menu while online with an NJ-series CPU Unit. Then, click the **Shutdown** Button under *Service* - *Shutdown* in the Online Settings Tab Page.



Additional Information

When you execute the Reset Controller operation on Sysmac Studio, the shutdown function is automatically executed before resetting the Controller.

Instruction Execution

Execute a DB_Shutdown (Shutdown DB Connection Service) instruction.

5-2-4 How to Check the Shutdown of the DB Connection Service

Confirm that the DB Connection Service has been shut down by the following methods before turning OFF the power supply to the CPU Unit.

- Checking with a system-defined variable
  Confirm that *_DBC_Status.Shutdown* system-defined variable (Shutdown flag of the DB Connection Service Status) is TRUE.
- Checking by executing an instruction
  Confirm that the *Done* output variable of the DB_Shutdown (Shutdown DB Connection Service) instruction is TRUE.

5-3 How to Prevent Losing SQL Statements at Power Interruption

This section describes how to write the user program so as not to lose the SQL statements at power interruption.

5-3-1 Overview

You can prevent losing the SQL statements to send and the SQL statements stored in the Spool memory even if a power interruption occurred during execution of a record processing instruction (such as DB_Insert and DB_Update instructions) by using the Spool function in combination with the user program.

5-3-2 Procedures

Use the following procedures.

Checking the Progress of the DB Connection Instruction

The progress of the DB Connection Instructions is output to the *SendStatus* output variable as enumeration data. Use this data to create the user program.

Output variable	Meaning	Data type	Description
SendStatus	Send Status	_eDBC_SEND_STATUS	_DBC_SEND_INIT(0): Initial status _DBC_SEND_UNSENT(1): SQL statement unsent _DBC_SEND_SENDING(2): Sending SQL statement _DBC_SEND_SPOOLED(3): SQL statement spooled _DBC_SEND_COMPLETE(4): SQL statement transmission completed

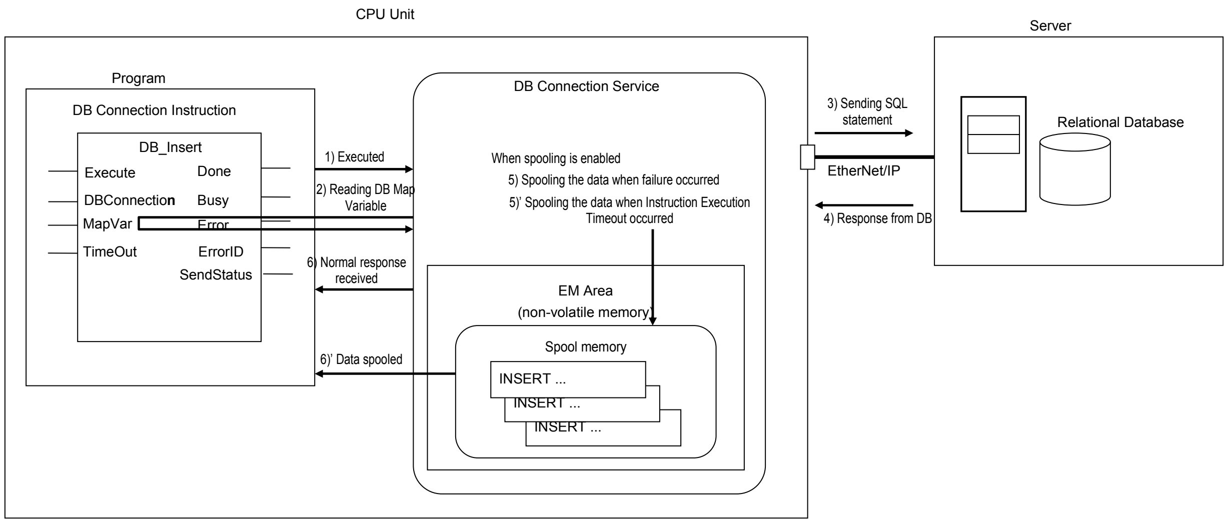
Variable Settings

- Set the Retain attribute of the input parameter (DB Map Variable) of the *MapVar* input variable to *Retained*.
- Set the Retain attribute of the output parameter of the *Busy* output variable to *Retained*.
- Set the Retain attribute of the output parameter of the *SendStatus* output variable to *Retained*.

Necessary Actions against Power Interruption

You need to take an action against power interruption according to when power interruption occurs.

This section describes the necessary actions using the following figure.



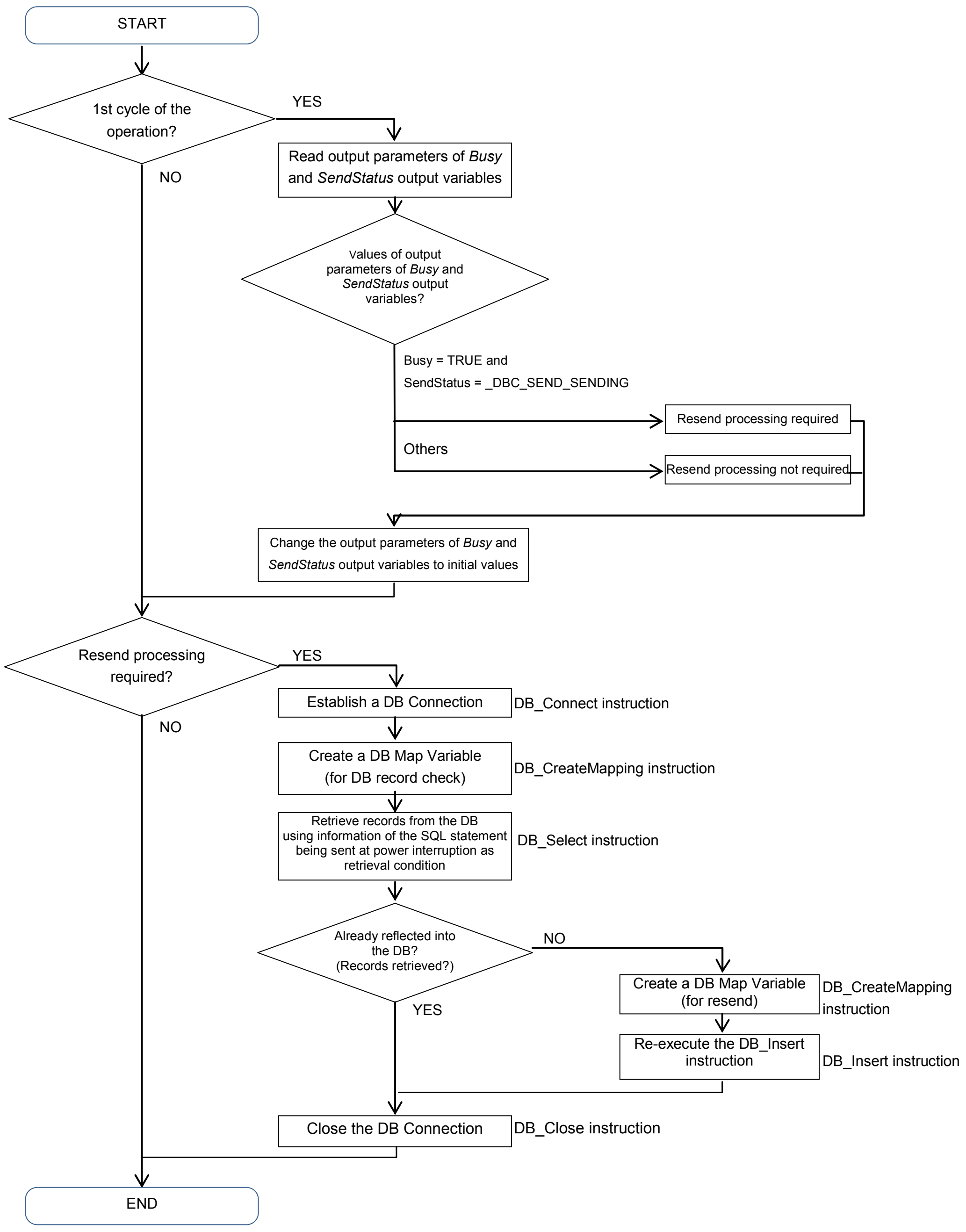
The numbers in the following table are corresponding to the numbers in the above figure.

Power interruption timing during execution of a DB Connection Instruction		Value of *SendStatus* output variable	Action
1) Executed (When instruction execution is started)	Until the DB Connection Service reads the present value of the DB Map Variable after *Execute* of the DB Connection Instruction changed from FALSE to TRUE	_DBC_SEND_SENDING: Sending SQL statement	Resend by user program
2) Reading DB Map Variable	Until the DB Connection Service sends the SQL statement to the DB after the service started reading the present value of the DB Map Variable		
3) Sending SQL statement	Until the transmission is completed since immediately before the DB Connection Service sends the SQL statement to the DB		
4) Response from DB	Until the response from DB is received after the SQL statement was sent to DB		
5) Spooling the data when failure occurred	While the SQL statement is being spooled because a failure has occurred (when spooling is enabled)		
5)' Spooling the data when Instruction Execution Timeout occurred	While the SQL statement is being spooled because an Instruction Execution Timeout has occurred. (when spooling is enabled)		
6) Normal response received	After normal response is received from the DB	_DBC_SEND_COMPLETE: SQL statement transmission completed	Action not required
6)' Data spooled	After the SQL statement is spooled (when spooling is enabled)	_DBC_SEND_SPOOLED: SQL statement spooled	Resend by Spool function (auto resend or manual resend)

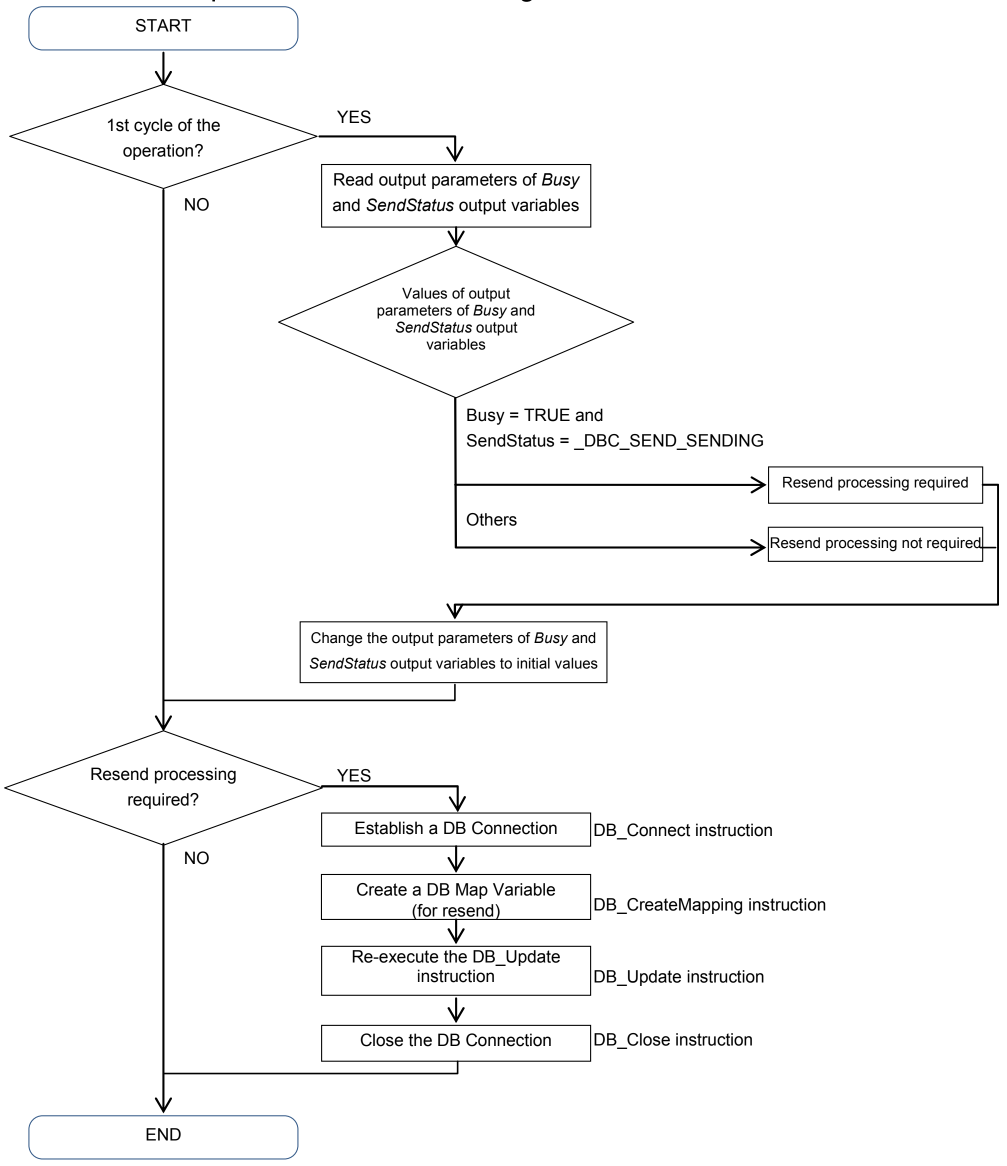
Resend Flow by User Program

Write the user program to re-execute the instruction that is being executed at the time of power interruption. The resend flow differs by whether a DB_Insert or DB_Update instruction is being executed at the time of power interruption.

● When a DB_Insert instruction is being executed


START
1st cycle of the operation?
YES
NO
Read output parameters of *Busy* and *SendStatus* output variables
Values of output parameters of *Busy* and *SendStatus* output variables?
Busy = TRUE and SendStatus = _DBC_SEND_SENDING
Resend processing required
Others
Resend processing not required
Change the output parameters of *Busy* and *SendStatus* output variables to initial values
Resend processing required?
YES
NO
Establish a DB Connection
DB_Connect instruction
Create a DB Map Variable (for DB record check)
DB_CreateMapping instruction
Retrieve records from the DB using information of the SQL statement being sent at power interruption as retrieval condition
DB_Select instruction
Already reflected into the DB? (Records retrieved?)
NO
YES
Create a DB Map Variable (for resend)
DB_CreateMapping instruction
Re-execute the DB_Insert instruction
DB_Insert instruction
Close the DB Connection
DB_Close instruction
END

- When a DB_Update instruction is being executed


START
1st cycle of the operation?
YES
NO
Read output parameters of *Busy* and *SendStatus* output variables
Values of output parameters of *Busy* and *SendStatus* output variables
Busy = TRUE and SendStatus = _DBC_SEND_SENDING
Resend processing required
Others
Resend processing not required
Change the output parameters of *Busy* and *SendStatus* output variables to initial values
Resend processing required?
YES
NO
Establish a DB Connection
DB_Connect instruction
Create a DB Map Variable (for resend)
DB_CreateMapping instruction
Re-execute the DB_Update instruction
DB_Update instruction
Close the DB Connection
DB_Close instruction
END

Precautions for Correct Use

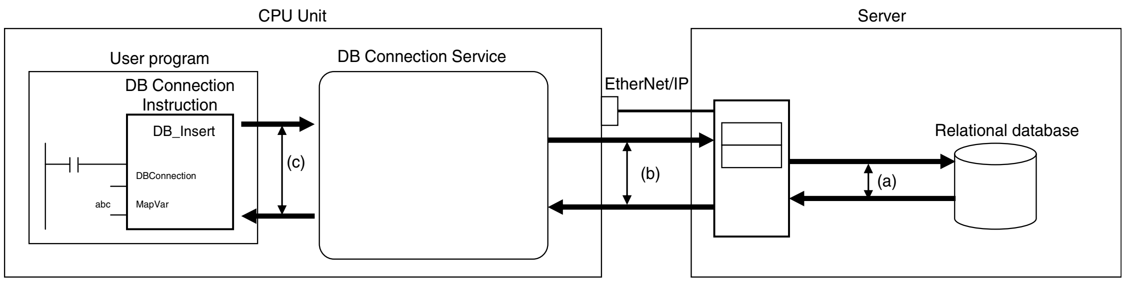
- The value of the *SendStatus* output variable is overwritten when the value of the *Execute* input variable is evaluated regardless of the value of the *Execute* input variable. Therefore, write the user program so that the value of the *SendStatus* output variable is read before evaluating the value of the *Execute* input variable of the DB Connection Instruction in the first cycle of the operation.
- The DB Connection Instruction is not executed if the *Execute* input variable is already TRUE at the operation start. You need to change the *Execute* input variable to FALSE to execute the instruction.

5-4 Timeout Monitoring Functions

This section describes timeout monitoring for the DB Connection Service.

5-4-1 Timeout Monitoring Functions

The following figure shows the types of timeouts that can be monitored.



Function name	Setting range	Description	Reference
Login timeout	1 to 60 seconds Default: 10 seconds	Time until the DB Connection Service detects a login failure due to a communications failure between DB Connection Service and DB or server's problem	Refer to *2-2-2 DB Connection Settings*.
Query execution timeout ((a) in the above figure)	1 to 600 seconds Default: 30 seconds	Time until the DB Connection Service detects an error when the DB takes time for query execution. You can cancel the SQL operation when the DB takes longer than expected for query execution.	Refer to *2-2-2 DB Connection Settings*.
Communications timeout ((b) in the above figure)	Time specified for *Query execution timeout* plus 10 seconds*	Time until the DB Connection Service detects an error due to a communications failure between DB Connection Service and DB	---
Instruction execution timeout ((c) in the above figure)	Not monitored, or 0.05 to180 seconds Default: Not monitored	Time until the DB Connection Service detects an error when a DB_Insert, DB_Update, DB_Select or DB_Delete instruction takes time due to a communications failure between DB Connection Service and DB or server's problem or heavy load. You can use this when you do not want to extend the takt time (i.e., lower the equipment performance).	Refer to *Appendix DB Connection Instructions*.
Keep Alive monitoring time	1 to 65535 seconds Default: 300 seconds	This function is used to check whether the server is normally connected. When you set this Keep Alive monitoring time, a communications failure can be detected even while the DB Connection Service is waiting for a response from the server because the DB is executing a query.	Refer to the *NJ/NX-series CPU Unit Built-in EtherNet/IP Port User's Manual* (Cat. No. W506).

* The time to detect a communications timeout differs by the DB type and DB status.

5-4-2 Login Timeout

The login timeout is monitored in the following cases.

- When connecting to a DB using a DB_Connect (Establish DB Connection) instruction
- When reconnecting to a DB while a DB Connection is in the *Disconnected* status

The following table shows the operation to be performed when a login timeout has occurred.

When the timeout occurred	DB Connection status after the timeout occurred	Instruction execution result
When executing a DB_Connect instruction	Closed	ErrorID = 3005 hex (DB Connection Failed)
When reconnecting to a DB	Disconnected	---

5-4-3 Query Execution Timeout

The query execution timeout is monitored in the following cases.

- When sending an SQL statement to a DB using a DB_Insert (Insert DB Record), DB_Update (Update DB Record), DB_Select (Retrieve DB Record), or DB_Delete (Delete DB Record) instruction
- When resending an SQL statement stored in the Spool memory

The following table shows the operation to be performed when a query execution timeout has occurred.

When the timeout occurred	DB Connection status after the timeout occurred	Instruction execution result

When executing a DB_Insert or DB_Update instruction	Connected	ErrorID = 300B hex (SQL Execution Error)
		SendStatus = _DBC_SEND_COMPLETE
		The SQL statement is not stored in the Spool memory.*

When executing a DB_Select or DB_Delete instruction	Connected	ErrorID = 300B hex (SQL Execution Error)
When resending Spool data	Connected	The SQL statement is not stored in the Spool memory again.*

* If an instruction error (SQL Execution Error) occurs, the transmitted SQL statement itself can be the cause of the SQL Execution Error. Therefore, the SQL statement is not stored in the Spool memory because the SQL Execution Error may occur again when the SQL statement is resent.

5-4-4 Communications Timeout

The communications timeout is monitored in the following cases.

- When sending an SQL statement to a DB using a DB_Insert (Insert DB Record), DB_Update (Update DB Record), DB_Select (Retrieve DB Record), or DB_Delete (Delete DB Record) instruction
- When resending an SQL statement stored in the Spool memory

The following table shows the operation to be performed when a communications timeout has occurred.

When the timeout occurred	DB Connection status after the timeout occurred	Spool function	Instruction execution result

When executing a DB_Insert or DB_Update instruction	Disconnected	Enabled	ErrorID = 3011 hex (DB Connection Disconnected Error Status)
			SendStatus = _DBC_SEND_SPOOLED
			The SQL statement is stored in the Spool memory.

		Disabled	ErrorID = 3011 hex (DB Connection Disconnected Error Status)
			SendStatus = _DBC_SEND_SENDING

When executing a DB_Select or DB_Delete instruction	Disconnected	---	ErrorID = 3011Hex (DB Connection Disconnected Error Status)
			SendStatus = _DBC_SEND_SENDING

When resending Spool data	Disconnected	Enabled	The SQL statement is stored in the Spool memory again.

5-4-5 Instruction Execution Timeout

Refer to *5-1-8 Relationship with DB Connection Instructions* for details on the instruction execution timeout.

5-4-6 Keep Alive Monitoring Time

Whether the server is normally connected is monitored while the DB Connection is in the *Connected* status.
When the connection to the server cannot be confirmed for the time set in the *Keep Alive monitoring time* parameter plus 12 seconds due to a communications failure or server's problem, the DB Connection is closed.
The DB Connection is changed to the *Disconnected* status, when Spool data is resent or a DB_Insert (Insert DB Record), DB_Update (Update DB Record), DB_Select (Retrieve DB Record), or DB_Delete (Delete DB Record) instruction is executed after the DB Connection is closed.

The keep-alive function operates as shown below in the DB Connection Service.

- Regardless of the Keep Alive setting, the function is always used.
- Regardless of the Linger option setting, the option is always specified.

The operation to be performed after the DB Connection is closed by the keep-alive monitoring function is the same as the communications timeout. Refer to *5-4-4 Communications Timeout* for the operation.

Precautions for Correct Use

- The Keep Alive monitoring time is a common setting to the built-in EtherNet/IP port.
  When you set the Keep Alive monitoring time, confirm that the operations of the following functions in the built-in EtherNet/IP port are not affected before changing the value.
  Socket service, FTP server function, communications with Sysmac Studio, FINS/TCP

5-5 Other Functions

This section describes the other DB Connection functions related to the backup/restore function of the NJ-series Controllers and verification of operation authority from Sysmac Studio.

5-5-1 Backup/Restore Function in the DB Connection Service

The backup function is used to back up the setting data in an NJ-series Controller into an SD Memory Card or a computer. And the restore function is used to restore the data from an SD Memory Card or a computer to the Controller.
This section describes the Controller's backup/restore function related to the DB Connection Service.

The following table shows whether each data can be backed up and restored by the function.

Data	Backup / Restore function	Available operations	Remarks
DB Connection settings	Supported	Backup / Restore*	Data group in the backup function is *User program and settings*.
Event log		Backup only	Data group in the backup function is *Event log*.
Operation Logs	Not supported	---	Refer to the Additional Information below.
Spool data			The Spool data is cleared by the Restore operation.

* The Restore operation cannot be performed in the following cases.
When any of the following is applicable, the DB Connection settings cannot be restored. The Restore Operation Failed to Start event is registered into the event log when the Restore operation is executed.

- You attempt to restore the data from a CPU Unit other than NJ501-1□20 to an NJ501-1□20 CPU Unit.
- You attempt to restore the data from an NJ501-1□20 CPU Unit to a CPU Unit other than NJ501-1□20.
- You attempt to restore the data from a CPU Unit other than NJ501-4320 to an NJ501-4320 CPU Unit.
- You attempt to restore the data from an NJ501-4320 CPU Unit to a CPU Unit other than NJ501-4320.
- You attempt to restore the data from a CPU Unit other than NJ101-1020 to an NJ101-1020 CPU Unit.
- You attempt to restore the data from an NJ101-1020 CPU Unit to a CPU Unit other than NJ101-1020.
- You attempt to restore the data from a CPU Unit other than NJ101-9020 to an NJ101-9020 CPU Unit.
- You attempt to restore the data from an NJ101-9020 CPU Unit to a CPU Unit other than NJ101-9020.
- The Unit version of the restore-destination CPU Unit is earlier than the Unit version of the backup-source CPU Unit.

The restore operation can be performed between the NJ501-1□20 CPU Units even if the model number (i.e., the number of axes) is different.

Additional Information

The Operation Logs cannot be backed up nor restored by the Backup/Restore operation. If you want to keep the Operation Log data after replacement of the CPU Unit, insert the used SD Memory Card to the restore-destination CPU Unit after completion of the Restore operation.

5-5-2 Operation Authority Verification in the DB Connection Service

This function is used to restrict the online operations that can be performed on the CPU Unit from Sysmac Studio according to the operation rights.

This section describes the operation authority verification function related to the DB Connection Service.

Refer to the *NJ/NX-series CPU Unit Software User's Manual* (Cat. No. W501) and the *Sysmac Studio Version 1 Operation Manual* (Cat. No. W504) for details of the operation authority verification function.

The functions, authorities, and operation restrictions that require verification in the DB Connection Service are given below.

OP: Operation possible

VR: Verification required for each operation

NP: Operation not possible

Monitoring status	Administrator	Designer	Maintainer	Operator	Observer
DB Connection Service Monitor	OP	OP	OP	OP	OP
Connection Monitor Table	OP	OP	OP	OP	OP

Controller operations	Administrator	Designer	Maintainer	Operator	Observer
Displaying the Operation Logs	OP	OP	OP	OP	NP
Clearing the Operation Logs	OP	OP	OP	NP	NP
Starting/stopping the DB Connection Service	OP	OP	NP	NP	NP
Shutting down the DB Connection Service	OP	OP	NP	NP	NP
Starting/stopping the Debug Log	OP	OP	VR	NP	NP
Clearing the Spool data	OP	OP	NP	NP	NP

DB connection test	Administrator	Designer	Maintainer	Operator	Observer
Communications test	OP	OP	OP	NP	NP

6

How to Use Operation Logs

This section describes how to use the Operation Logs for tracing the operations of the DB Connection Service.

6-1 Operation Logs

Operation Logs are used to trace the operations of the DB Connection Service on the CPU Unit. The logs are saved on the SD Memory Card mounted in the CPU Unit.

The following three types of Operation Logs are provided.

Operation Log type	Description
Execution Log	Used to record the executions of the DB Connection Service in order to check the execution records of the DB Connection function.
Debug Log	Used to record the contents and results of SQL executions and user-specified logs for debugging.
SQL Execution Failure Log	Used to record the transmitted SQL statements and error information in order to check the information on execution failure of SQL statements in the DB.

6-2 Execution Log

This section describes the Execution Log used to trace the executions of the DB Connection Service.

6-2-1 Overview

You can check the start/stop of the DB Connection Service, connection/disconnection with the DB, and success/failure of SQL statement executions with the Execution Log. Thus, you can check whether the expected DB Connection Service processing is executed.

You can record this log by setting *Execution log* to *Record* in the DB Connection Service Settings of Sysmac Studio. You can also record a specified log as Execution Log by executing a DB_PutLog (Record Operation Log) instruction.

When you record this log, the Execution Log file is constantly saved on the SD Memory Card mounted in the CPU Unit while the DB Connection Service is running.

The Execution Log is temporarily recorded in the internal buffer (volatile memory) of the CPU Unit and then saved on the SD Memory Card. While the SD Memory Card is being replaced, the Execution Log is kept in the internal buffer (volatile memory) of the CPU Unit. When you insert an SD Memory Card, the Execution Log temporarily stored in the internal buffer is automatically saved on the SD Memory Card. Refer to *6-5-3 Operation Log Operations in Replacing the SD Memory Card* for details.

You can check the contents of this log in the Execution Log Tab Page of the Operation Log Window in Sysmac Studio.

Precautions for Correct Use

When you use the Execution Log, be sure to insert an SD Memory Card into the CPU Unit. The Execution Log is temporarily recorded in the internal buffer of the CPU Unit and then saved on the SD Memory Card. If no SD Memory Card is mounted at power OFF or shutdown processing of the CPU Unit, the Execution Log recorded in the internal buffer will be lost.

6-2-2 Application Procedure

Use the Execution Log according to the following procedure.

Step	Reference
1. Set the Execution Log.	Refer to *6-2-3 Setting the Execution Log*.
↓	
2. Check the Execution Log.	Refer to *6-6 Checking the Operation Logs*.

6-2-3 Setting the Execution Log

Double-click **DB Connection Service Settings** under **Configurations and Setup** - **Host Connection Settings** - **DB Connection** in the Multiview Explorer. Then, set the following in the Service Setting.

Item	Description	Values
Execution log	Set whether to record the Execution Log.	- Record (Default) - Do not record
Number of files	Set the maximum number of files of the Execution Log. When the maximum number of files is reached, the oldest file is deleted and a new file is created.	2 to 100 files (Default: 48)
Number of records	Set the number of log records that can be contained in each Execution Log file. When the maximum number of records is reached, a new file is created.	100 to 65536 records (Default: 7200)

You can record a specified log as Execution Log using a DB_PutLog (Record Operation Log) instruction. The logs recorded by a DB_PutLog (Record Operation Log) instruction are called “user-specified log”.
To record a user-specified log, set *Log Type* to *Execution Log* and specify the log code, log name, and log message in a DB_PutLog (Record Operation Log) instruction and execute the instruction. Refer to *Appendix DB Connection Instructions* for details of the DB_PutLog (Record Operation Log) instruction.

6-2-4 Checking the Execution Log

Refer to *6-6 Checking the Operation Logs* for how to check the Execution Log.

6-2-5 Execution Log File Specifications

This section describes the specifications of Execution Log files.
- Each Execution Log file is composed of multiple records.
- Each record is expressed in one line.
- The maximum number of records to be contained in each Execution Log file is set in Sysmac Studio.
- The size of each record is 256 bytes max.
- The following table shows the file name and type.

File name	File type
DB_ExecutionLog.log	Latest log file of the log
DB_ExecutionLog_[year_month_date_hours_minutes_seconds_milliseconds].log* Example: DB_ExecutionLog_20120724220915040.log	Previous log files
DB_ExecutionLog.fjc	Log control file

* The system time of the CPU Unit is used for the time information included in the file name.

- The files are stored in the following directory (of the SD Memory Card).
  - Log files:
    /packages/DB_Connection/ExecutionLog/
  - Log control file:
    /packages/DB_Connection/System/

- The following is the format of records.
  Each record is expressed in one line and composed of multiple parameters. The parameters are separated from each other by a tab.

```
[Serial number]<tab>[Date]<tab>[Time]<tab>[Millisecond]<tab>[Category]<tab>[Log code]<tab>[Log
name]<tab>[Result]<tab>[DB Connection name]<tab>[Serial ID]<tab>[Details]<CR><LF>
```

Parameter	Size	Description
Serial number	1 to 5 bytes	0 to 65535 When exceeding 65535, this value returns to 0. The serial number is given across multiple files. (Even if a new file is created, the serial number is not reset to 0.)
Date	10 bytes (Fixed)	Displays year, month, and date when the log was recorded.*1 YYYY-MM-DD Example: 2012-07-23
Time	8 bytes (Fixed)	Displays hours, minutes, and seconds when the log was recorded. *1 hh:mm:ss Example: 15:33:45
Millisecond	3 bytes (Fixed)	Displays 3-digit decimal integer (000 to 999) that shows millisecond of the time when the log was recorded*1 Example: 10 ms: 010 623 ms: 623
Category	16 bytes max. (Variable)	Displays the category.*2
Log code	4 bytes (Fixed)	Displays a 4-digit decimal code that is a unique identification code in the category.*3
Log name	32 bytes max. (Variable)	Displays a name that shows the contents of the log.*4
Result	6 bytes (Fixed)	Displays a 4-digit hexadecimal code that shows the execution result. (e.g., 0x1234) 0x0000: Succeeded Other than 0x0000: Failed (Same code as *ErrorID* of DB Connection Instruction)
DB Connection name	16 bytes max. (Variable)	Displays a DB Connection name (single-byte alphanumeric characters) * When the category is DB Connection Service or User-specified Log, nothing is displayed.
Serial ID	10 bytes max. (Variable)	ID code given at each execution of DB_Insert, DB_Update, DB_Select, or DB_Delete instruction. Decimal code consisting of 10 digits max. Possible range: 0 to 2147483647 When this value exceeds 2147483647 or when the power supply to the CPU Unit is turned ON, the value returns to 0. * When the category is DB Connection Service, DB Connection, or User-specified Log, nothing is displayed.

Parameter	Size	Description
Details	Variable	Displays the details of the Execution Log. The contents differ according to the category. In the Details parameter, information items are separated from each other by a tab. Category: DB Connection Service None Category: DB Connection [SQL status]<tab>[DB error code]<tab>[Error message] SQL status: The SQLSTATE value defined in the SQL Standards (ISO/IEC 9075) is displayed. DB error code: Error code that is specific to DB vendor of the device to connect. When a network error has occurred, 0 is displayed for DB error code in some cases. When 0 is displayed, check its SQL status. Error message: The error message is displayed from the first character within the record size (i.e., 256 bytes). Category: SQL [Table name]<tab>[DB Map Variable name]<tab>[DB response time]<tab>[DB error code] Table name and DB Map Variable name: A maximum of 60 bytes from the beginning are displayed. DB Map Variable name: Variable name specified in the *MapVar* input variable (The POU instance name is not displayed. Nothing is displayed for DELETE.) DB response time: An integer value in milliseconds is displayed. DB error code: Error code that is specific to DB vendor of the device to connect. When a network error has occurred, 0 is displayed for DB error code in some cases. When 0 is displayed, check the Result parameter. Category: SQL Resend [DB response time]<tab>[DB error code] DB response time: An integer value in milliseconds is displayed. DB error code: Error code that is specific to DB vendor of the device to connect. When a network error has occurred, 0 is displayed for DB error code in some cases. When 0 is displayed, check the Result parameter. Category: User-specified Log "[Log message]" Displays the text string specified in the *LogMsg* input variable of the DB_PutLog instruction. (128 bytes max.)
Tab separation	10 bytes in total	
CR+LF	2 bytes	

*1 The date and time information follows the time zone set when the power supply to the Controller is turned ON. After you change the time zone, cycle the power supply.

*2 Category

Category	Characters displayed in the log
DB Connection Service	DB_SERVICE
DB Connection	DB_CONNECTION
SQL	SQL
SQL Resend	SQL_RESEND
User-specified Log	USER

*3 Code

Category	Code (decimal)	Operation	Log recording timing
DB Connection Service	0001	DB Connection Service Started	When the start processing of the DB Connection Service is completed (succeeded/failed)
	0002	DB Connection Service Stopped	When the stop processing of the DB Connection Service is completed (succeeded/failed)
	0003	DB Connection Service Shutdown	When the shutdown processing of the DB Connection Service is completed (succeeded/failed)
DB Connection	0001	DB Connection Established	When the establishment processing of a DB Connection is completed (succeeded/failed) after the establishment is commanded from Sysmac Studio or the applicable instruction.
	0002	DB Connection Closed	When the close processing of a DB Connection is completed (succeeded/failed) after the close is commanded from Sysmac Studio or the applicable instruction.
	0003	DB Connection Disconnected	When disconnection from the DB is detected.
	0004	DB Connection Reestablished	When the DB Connection status changes from *Disconnected* to *Connected*.
SQL	0001	INSERT	When a response (succeeded/failed) is returned to INSERT that is issued from DB Connection Service to DB after execution of a DB_Insert (Insert DB Record) instruction.
	0002	UPDATE	When a response (succeeded/failed) is returned to UPDATE that is issued from DB Connection Service to DB after execution of a DB_Update (Update DB Record) instruction.
	0003	SELECT	When a response (succeeded/failed) is returned to SELECT that is issued from DB Connection Service to DB after execution of a DB_Select (Retrieve DB Record) instruction.
	0004	DELETE	When a response (succeeded/failed) is returned to DELETE that is issued from DB Connection Service to DB after execution of a DB_Delete (Delete DB Record) instruction.
SQL Resend	0001	INSERT	When a response (succeeded/failed) is returned to INSERT after resending the INSERT statement stored in the Spool memory.
	0002	UPDATE	When a response (succeeded/failed) is returned to UPDATE after resending the UPDATE statement stored in the Spool memory.
User-specified Log	0000 to 9999 (specified by the user)	DB_PutLog Instruction Executed	When a DB_PutLog (Record Operation Log) instruction is executed

*4 Log Name

Category	Operation	Log name
DB Connection Service	DB Connection Service Started	Start
	DB Connection Service Stopped	Stop
	DB Connection Service Shutdown	Shutdown
DB Connection	DB Connection Established	Connect
	DB Connection Closed	Close
	DB Connection Disconnected	Disconnect
	DB Connection Reestablished	Reconnect
SQL	INSERT	INSERT
	UPDATE	UPDATE
	SELECT	SELECT
	DELETE	DELETE
SQL Resend	INSERT	INSERT
	UPDATE	UPDATE
User-specified Log	DB_PutLog Instruction Executed	Text string specified in the *LogName* input variable of the DB_PutLog instruction.

●Record examples:

・DB Connection Service Started:

```
1   2012-07-24   21:29:45   267   DB_SERVICE 0001  Start  0x0000
```

・INSERT (Failed):

```
1   2012-07-24   21:29:45   267   SQL  0001  INSERT 0x1234  DBConnection1   45 TableX   VarY  100   17026
```

・User-specified Log:

```
1   2012-07-24   21:29:45   267   USER    9876  LineA1   0x0000          "ProductionStarted"
```

・Log file example:

```
0   2012-07-24 08:29:45  267  DB_SERVICE         0001 Start     0x0000
1   2012-07-24 08:31:52  002  DB_CONNECTION      0001 Connect   0x0000 MyDatabase1
2   2012-07-24 08:31:53  959  DB_CONNECTION      0001 Connect   0x0000 MyDatabase2
3   2012-07-24 09:00:00  052  USER               0001 LineA1    0x0000                  "ProductionStarted"
4   2012-07-24 09:00:00  150  SQL                0001 INSERT    0x0000 MyDatabase1  0   TABLE_Production   Production           100  0
5   2012-07-24 09:10:00  150  SQL                0001 INSERT    0x0000 MyDatabase1  1   TABLE_Production   Production           100  0
6   2012-07-24 09:20:00  151  SQL                0001 INSERT    0x0000 MyDatabase1  2   TABLE_Production   Production           100  0
7   2012-07-24 09:30:00  150  SQL                0001 INSERT    0x0000 MyDatabase1  3   TABLE_Production   Production           100  0
8   2012-07-24 09:55:23  422  USER               0002 LIneA1    0x0000                  "ProductionFinished"
9   2012-07-24 10:15:00  549  SQL                0003 SELECT    0x0000 MyDatabase2  4   TABLE_MPS          ProductionSchedule   200  0
```

Precautions for Correct Use

Do not delete the latest log file (DB_ExecutionLog.log) and the log control file (DB_ExecutionLog.fjc) from the SD Memory Card. If they are deleted, the log files are not saved correctly, for example, the Execution Log data are lost.

6-3 Debug Log

This section describes the Debug Log used for debugging the DB Connection Service.

6-3-1 Overview

You can check which SQL statement is executed, parameters of each SQL statement, and execution results with the Debug Log.

You can record this log by clicking the **Start** Button for Debug Log in the Online Settings Tab Page of Sysmac Studio. You can also record a specified log as Debug Log by executing a DB_PutLog (Record Operation Log) instruction.

This log is saved as Debug Log files on the SD Memory Card mounted in the CPU Unit. When no SD Memory Card is mounted in the CPU Unit, you cannot record the Debug Log.

You can check the contents of this log in the Debug Log Tab Page of the Operation Log Window in Sysmac Studio.

Additional Information

The Debug Log is used to check the parameters and execution results of the SQL statements executed using the DB Connection Instructions. When the Spool data is resent, it is not recorded to the Debug Log. To check the time and execution results of SQL statements resent from the Spool memory, check the Execution Log record with the same serial ID. To check the parameters of the SQL statements in that case, check the log record at the time when the applicable SQL statement is spooled in the Debug Log.

6-3-2 Application Procedure

Use the Debug Log according to the following procedure.

Step	Reference
1. Set the Debug Log.	Refer to *6-3-3 Setting the Debug Log*.
↓	
2. Start recording to the Debug Log.	Refer to *6-3-4 Starting recording to Debug Log*.
↓	
3. Check the Debug Log.	Refer to *6-6 Checking the Operation Logs*.

6-3-3 Setting the Debug Log

Double-click **DB Connection Service Settings** under **Configurations and Setup** - **Host Connection Settings** - **DB Connection** in the Multiview Explorer. Then, set the following in the Service Setting.

Item	Description	Values
Number of files	Set the maximum number of files of the Debug Log.	1 to 100 files (Default: 1)
File size	Set the maximum file size. When the maximum file size is exceeded or when the number of records exceeds 65,536 records in a file, a new file is created.	1 to 100 MB (Default: 10 MB)
When the log is full	Set the action to be taken when the Debug Log has reached the maximum number of files.	- Stop logging (Default) - Continue logging (Delete the oldest file)
Delete the log at recording start	Set whether to delete the Debug Log contained in the SD Memory Card when recording is started.	- Delete (Default) - Do not delete

You can record a specified log as Debug Log using a DB_PutLog (Record Operation Log) instruction. The logs recorded by a DB_PutLog (Record Operation Log) instruction are called “user-specified log”.

To record the user-specified log, set *Log Type* to *Debug Log* and specify the log code, log name, and log message in a DB_PutLog (Record Operation Log) instruction and execute the instruction. Refer to *Appendix DB Connection Instructions* for details of the DB_PutLog (Record Operation Log) instruction.

6-3-4 Starting Recording to Debug Log

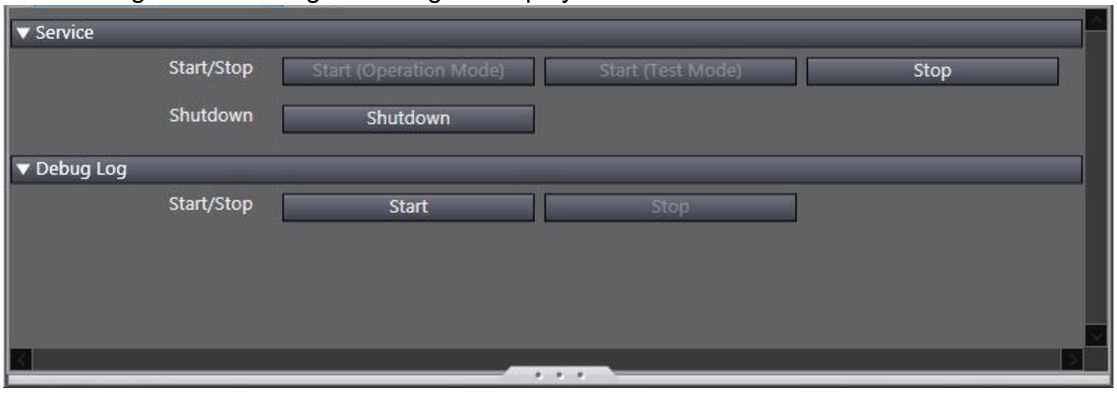
You can start recording to the Debug Log by the following methods.

- Online operation from Sysmac Studio
- Executing a DB_ControlService (Control DB Connection Service) instruction

Start by Online Operation from Sysmac Studio

1. Right-click **DB Connection Service Settings** under **Configurations and Setup** - **Host Connection Settings** - **DB Connection** in the Multiview Explorer and select ***Online Settings*** from the menu.

The following Online Settings Tab Page is displayed.

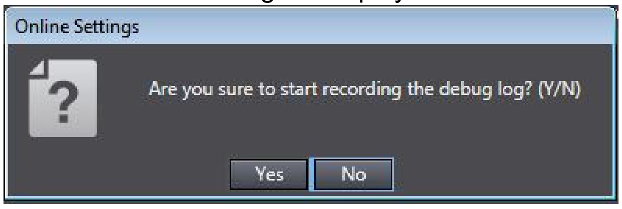


You can start and stop recording to the Debug Log by clicking the following buttons.

Category	Item	Button	Operation
Debug Log	Start/Stop	Start	Recording to the Debug Log is started.
		Stop	Recording to the Debug Log is stopped.

2. Click the **Start** Button.

A confirmation message is displayed.


Online Settings

Are you sure to start recording the debug log? (Y/N)

3. Click the Yes Button.

Start by executing a DB_ControlService Instruction

Specify *Start recording to Debug Log* in the *Cmd* input variable of the DB_ControlService (Control DB Connection Service) instruction and execute the instruction. Refer to *Appendix DB Connection Instructions* for details of the instruction.

6-3-5 Stopping Recording to Debug Log

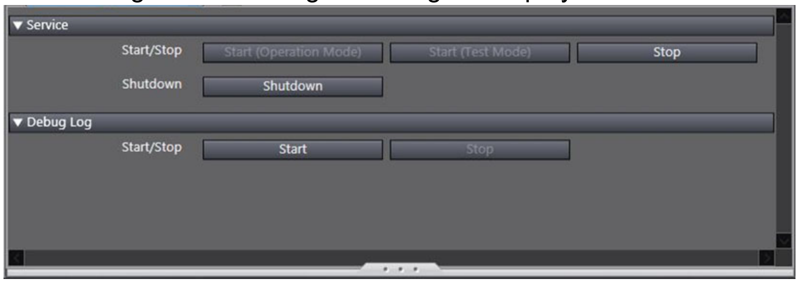
You can stop recording to the Debug Log by the following methods.

- Online operation from Sysmac Studio
- Executing a DB_ControlService (Control DB Connection Service) instruction
- Automatically stopped when a specified condition is met

Stop by Online Operation from Sysmac Studio

1. Right-click **DB Connection Service Settings** under **Configurations and Setup** - **Host Connection Settings** - **DB Connection** in the Multiview Explorer and select ***Online Settings*** from the menu.

The following Online Settings Tab Page is displayed.

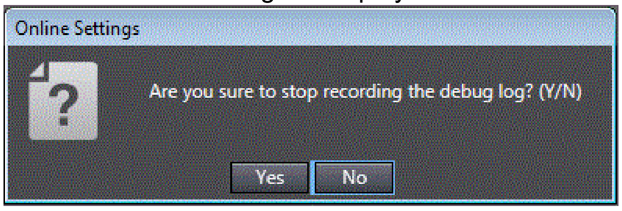


You can start and stop recording to the Debug Log by clicking the following buttons.

Category	Item	Button	Operation
Debug Log	Start/Stop	Start	Recording to the Debug Log is started.
		Stop	Recording to the Debug Log is stopped.

2. Click the **Stop** Button.

A confirmation message is displayed.



Online Settings

Are you sure to stop recording the debug log? (Y/N)

3. Click the **Yes** Button.

Stop by executing a DB_ControlService Instruction

Specify *Finish recording to Debug Log* in the *Cmd* input variable of the DB_ControlService (Control DB Connection Service) instruction and execute the instruction. Refer to *Appendix DB Connection Instructions* for details of the instruction.

Automatically Stopped when a Condition is Met

The recording to Debug Log is automatically stopped in the following conditions.

- When the SD Memory Card power supply switch is pressed
- When the Synchronization (download) operation is executed on Sysmac Studio
- When the Clear All Memory operation is executed
- When the Restore operation of the SD Memory Card backup function or Sysmac Studio Controller backup function is executed

6-3-6 Checking the Debug Log

Refer to *6-6 Checking the Operation Logs* for how to check the Debug Log.

6-3-7 Debug Log File Specifications

This section describes the specifications of Debug Log files.

- Each Debug Log file is composed of multiple records.
- The maximum size of each Debug Log file is set in Sysmac Studio.
- The size of each record is 58 KB max.
- The following table shows the file name and type.

File name	File type
DB_DebugLog.log	Latest log file of the log
DB_DebugLog_[year_month_date_hours_minutes_seconds_milliseconds].log* Example: DB_DebugLog_20120724220915040.log	Previous log files
DB_DebugLog.fjc	Log control file

* The system time of the CPU Unit is used for the time information included in the file name.

- The files are stored in the following directory (of the SD Memory Card).
  - Log files:
    /packages/DB_Connection/DebugLog/
  - Log control file:
    /packages/DB_Connection/System/

- The record format is shown below.

Each record is expressed in one line and composed of multiple parameters. The parameters are separated from each other by a tab.

```
[Serial number]<tab>[Date]<tab>[Time]<tab>[Millisecond]<tab>[Category]<tab>[Log code]<tab>[Log name]<tab>
[Result]<tab>[DB Connection name]<tab>[Serial ID]<tab>[Details]<CR><LF>
```

Parameter	Size	Description
Serial number	1 to 5 bytes	0 to 65535 When exceeding 65535, this value returns to 0. The serial number is given across multiple files. (Even if a new file is created, the serial number is not reset to 0.)
Date	10 bytes (Fixed)	Displays year, month, and date when the log was recorded. [*1] YYYY-MM-DD Example: 2012-07-23
Time	8 bytes (Fixed)	Displays hours, minutes, and seconds when the log was recorded. [*1] hh:mm:ss Example: 15:33:45
Millisecond	3 bytes (Fixed)	Displays 3-digit decimal integer (000 to 999) that shows millisecond of the time when the log was recorded. [*1] Example: 10 ms: 010 623 ms: 623
Category	16 bytes max. (Variable)	Displays the category.[*2]
Log code	4 bytes (Fixed)	Displays a 4-digit decimal code that is a unique identification code in the category.[*3]
Log name	32 bytes max. (Variable)	Displays a name that shows the contents of the log.[*4]

Parameter	Size	Description
Result	6 bytes (Fixed)	Displays a 4-digit hexadecimal code that shows the execution result. (e.g., 0x1234) 0x0000: Succeeded Other than 0x0000: Failed (Same code as *ErrorID* of DB Connection Instruction)
DB Connection name	16 bytes max. (Variable)	Displays a DB Connection name (single-byte alphanumeric characters) * When the category is DB Connection Service or User-specified Log, nothing is displayed.
Serial ID	10 bytes max. (Variable)	ID code given at each execution of DB_Insert, DB_Update, DB_Select, or DB_Delete instruction. (Displays the same ID as the serial ID displayed for the SQL category records in the Execution Log) Decimal code consisting of 10 digits max. Possible range: 0 to 2147483647 When this value exceeds 2147483647 or when the power supply to the CPU Unit is turned ON, the value returns to 0. * When the category is DB Connection Service, DB Connection, or User-specified Log, nothing is displayed.
Details	Variable	Displays the details of the Debug Log. The contents differ according to the category. In the Details parameter, information items are separated from each other by a tab. Category: DB Connection [DB type]<tab>[Connection text string]<tab>[User name]<tab>[DB error code]<tab>[Error message] DB error code: Error code that is specific to DB vendor of the device to connect. When a network error has occurred, 0 is displayed for DB error code in some cases. When 0 is displayed, check the Result parameter. Category: SQL [Table name]<tab>[DB Map Variable name]<tab>[SQL statement] DB Map Variable name: The POU instance name is not displayed. Category: SQL Execution Result [Table name]<tab>[DB Map Variable name]<tab>[DB response time]<tab>[DB error code]<tab>[Error message] DB Map Variable name: The POU instance name is not displayed. DB response time: An integer value in milliseconds is displayed. DB error code: Error code that is specific to DB vendor of the device to connect. When a network error has occurred, 0 is displayed for DB error code in some cases. When 0 is displayed, check the Result parameter. Category: User-specified Log "[Log message]" Displays the text string specified in the *LogMsg* input variable of the DB_PutLog instruction. (128 bytes max.)
Tab separation	10 bytes in total	
CR+LF	2 bytes	

*1 The date and time information follows the time zone set when the power supply to the Controller is turned ON. After you change the time zone, cycle the power supply.

*2 Category

Category	Characters displayed in the log
DB Connection	DB_CONNECTION
SQL	SQL
SQL Execution Result	SQL_RESULT
User-specified Log	USER

*3 Code

Category	Code (decimal)	Operation	Log recording timing
DB Connection	0001	DB Connection Established	When the establishment processing of a DB Connection is completed (succeeded/failed) after the establishment is commanded from the applicable instruction.
SQL	0001	INSERT	• Before the DB Connection Service sends an SQL statement after a DB_Insert (Insert DB Record) instruction is executed • When an SQL statement is stored in the Spool memory
	0002	UPDATE	• Before the DB Connection Service sends an SQL statement after a DB_Update (Update DB Record) instruction is executed • When an SQL statement is stored in the Spool memory
	0003	SELECT	Before the DB Connection Service sends an SQL statement after a DB_Select (Retrieve DB Record) instruction is executed.
	0004	DELETE	Before the DB Connection Service sends an SQL statement after a DB_Delete (Delete DB Record) instruction is executed.
SQL Execution Result	0001	INSERT	When a response (succeeded/failed) is returned to the INSERT issued from DB Connection Service to DB.
	0002	UPDATE	When a response (succeeded/failed) is returned to the UPDATE issued from DB Connection Service to DB.
	0003	SELECT	When a response (succeeded/failed) is returned to the SELECT issued from DB Connection Service to DB.
	0004	DELETE	When a response (succeeded/failed) is returned to the DELETE issued from DB Connection Service to DB.
User-specified Log	0000 to 9999 (specified by the user)	DB_PutLog Instruction Executed	When a DB_PutLog (Record Operation Log) instruction is executed

*4 Log Name

Category	Operation	Log name
DB Connection	DB Connection Established	Connect
SQL	INSERT	INSERT
	UPDATE	UPDATE
	SELECT	SELECT
	DELETE	DELETE
SQL Execution Result	INSERT	INSERT
	UPDATE	UPDATE
	SELECT	SELECT
	DELETE	DELETE
User-specified Log	DB_PutLog Instruction Executed	Text string specified in the *LogName* input variable of the DB_PutLog instruction.

Log file example:

```
1   2012-07-24   09:00:00   150   SQL          0001   INSERT    0x0000   MyDatabase1   45   TABLE_Production   Production
    INSERT INTO TABLE_Production(Column1) VALUES('1000')"
2   2012-07-24   09:00:00   200   SQL_RESULT   0001   INSERT    0x300B   MyDatabase1   46   17072  ORA-17072: Inserted value
    too large for column
```

Precautions for Correct Use

Do not delete the latest log file (DB_DebugLog.log) and the log control file (DB_DebugLog.fjc) from the SD Memory Card. If they are deleted, the log files are not saved correctly, for example, the Debug Log data are lost.

6-4 SQL Execution Failure Log

This section describes the SQL Execution Failure Log used to trace the execution failures of the DB Connection Service due to a DB-caused factor.

6-4-1 Overview

You can check the SQL statements and error information when transmission of an SQL statement failed due to a problem* of the DB itself.

* For example,

- Because the column names of the table have been changed, they do not match the column names of an SQL statement sent from the DB Connection Service.
- A value to insert is outside the valid range of the data type of the column.

You can record this log by setting *SQL execution failure log* to *Record* in the DB Connection Service Setting of Sysmac Studio.

This log is saved as SQL Execution Failure Log files on the SD Memory Card mounted in the CPU Unit. When no SD Memory Card is mounted in the CPU Unit, you cannot record the SQL Execution Failure Log.

You can check the contents of this log in the SQL Execution Failure Log Tab Page of the Operation Log Window in Sysmac Studio.

6-4-2 Application Procedure

Use the SQL Execution Failure Log according to the following procedure.

Step	Reference
1. Set the SQL Execution Failure Log.	Refer to *6-4-3 Setting the SQL Execution Failure Log*.
↓	
2. Check the SQL Execution Failure Log.	Refer to *6-6 Checking the Operation Logs*.

6-4-3 Setting the SQL Execution Failure Log

Double-click **DB Connection Service Settings** under **Configurations and Setup** - **Host Connection Settings** - **DB Connection** in the Multiview Explorer. Then, set the following in the Service Setting.

Item	Description	Values
SQL execution failure log	Set whether to record the SQL Execution Failure Log.	• Record • Do not record (Default)
Number of files	Set the maximum number of files of the SQL Execution Failure Log. When the maximum number of files is reached, the oldest file is deleted and a new file is created.	2 to 100 files (Default: 50)
File size	Set the maximum file size. When the maximum file size is exceeded or when the number of records exceeds 65,536 records in a file, a new file is created.	1 to 100 MB (Default: 10 MB)

6-4-4 Checking the SQL Execution Failure Log

Refer to *6-6 Checking the Operation Logs* for how to check the SQL Execution Failure Log.

6-4-5 SQL Execution Failure Log File Specifications

This section describes the specifications of SQL Execution Failure Log files.

• Each SQL Execution Failure Log file is composed of multiple records.
• Each record is expressed in one line.
• The maximum size of each SQL Execution Failure Log file is set on Sysmac Studio.
• The size of each record is 58 KB max.
• The following table shows the file name and type.

File name	File type
DB_SQLFailedLog.log	Latest log file of the log
DB_SQLFailedLog_[year_month_date_hours_minutes_seconds_milliseconds].log* Example: DB_SQLFailedLog_20120724220915040.log	Previous log files
DB_SQLFailedLog.fjc	Log control file

* The system time of the CPU Unit is used for the time information included in the file name.

• The files are stored in the following directory (of the SD Memory Card).
- Log files:
/packages/DB_Connection/SQLFailedLog/
- Log control file:
/packages/DB_Connection/System/

• The following is the format of records.
Each record is expressed in one line and composed of multiple parameters. The parameters are separated from each other by a tab.

```
[Serial number]<tab>[Date]<tab>[Time]<tab>[Millisecond]<tab>[Category]<tab>[Log code]<tab>[Log
name]<tab>[Result]<tab>[DB Connection name]<tab>[Serial ID]<tab>[Details]<CR><LF>
```

Parameter	Size	Description
Serial number	1 to 5 bytes	0 to 65535 When exceeding 65535, this value returns to 0. The serial number is given across multiple files. (Even if a new file is created, the serial number is not reset to 0.)
Date	10 bytes (Fixed)	Displays year, month, and date when the log was recorded. [*1] YYYY-MM-DD Example: 2012-07-23
Time	8 bytes (Fixed)	Displays hours, minutes, and seconds when the log was recorded. [*1] hh:mm:ss Example: 15:33:45
Millisecond	3 bytes (Fixed)	Displays 3-digit decimal integer (000 to 999) that shows millisecond of the time when the log was recorded. [*1] Example: 10 ms: 010 623 ms: 623
Category	16 bytes max. (Variable)	Displays the category.[*2]
Log code	4 bytes (Fixed)	Displays a 4-digit decimal code that is a unique identification code in the category.[*3]
Log name	32 bytes max. (Variable)	Displays a name that shows the contents of the log.[*4]

Parameter	Size	Description
Result	6 bytes (Fixed)	Displays a 4-digit hexadecimal code that shows the execution result. (e.g., 0x1234) 0x0000: Succeeded Other than 0x0000: Failed (Same code as *ErrorID* of DB Connection Instruction)
DB Connection name	16 bytes max. (Variable)	Displays a DB Connection name (single-byte alphanumeric characters)
Serial ID	10 bytes max. (Variable)	ID code given at each execution of DB_Insert, DB_Update, DB_Select, or DB_Delete instruction. (The same ID as Serial ID displayed in the SQL or SQL Resend record of Execution Log is displayed.)
Details	Variable	Displays the details of the SQL Execution Failure Log. The contents differ according to the category. In the Details parameter, information items are separated from each other by a tab. Category: SQL Execution Failed [Table name]<tab>[DB Map Variable name]<tab>[DB error code]<tab>[Error message]<tab>[SQL statement] DB Map Variable name: The POU instance name is not displayed. DB error code: Error code that is specific to DB vendor of the device to connect. When a network error has occurred, 0 is displayed for DB error code in some cases. When 0 is displayed, check the Result parameter. Category: Spooled [Table name]<tab>[DB Map Variable name]<tab>[SQL statement] DB Map Variable name: The POU instance name is not displayed. Category: Status Error [Table name]<tab>[DB Map Variable name]<tab>[SQL statement] DB Map Variable name: The POU instance name is not displayed.
Tab separation	10 bytes in total	
CR+LF	2 bytes	

*1 The date and time information follows the time zone set when the power supply to the Controller is turned ON. After you change the time zone, cycle the power supply.

*2 Category

Category	Characters displayed in the log
SQL Execution Failed	SQL_FAIL
Spooled	SPOOL
Status Error	STATUS_ERROR

*3 Code

Category	Code (decimal)	Operation	Log recording timing
SQL Execution Failed	0001	INSERT	When execution of an SQL statement issued from DB Connection Service to DB failed due to a DB-caused factor.
	0002	UPDATE	
	0003	SELECT	
	0004	DELETE	
Spooled	0001	INSERT	When an SQL statement is stored in the Spool memory because a failure occurred in information exchange between DB Connection Service and DB.
	0002	UPDATE	

Category	Code (decimal)	Operation	Log recording timing
Status Error	0001	INSERT	・When the DB Connection Service detected an error and could not send an SQL statement. ・When a failure occurred in information exchange between DB Connection Service and DB (when spooling is disabled) ・When an SQL statement cannot be stored in the Spool memory because the Spool capacity is insufficient as a failure occurred in information exchange between DB Connection Service and DB
	0002	UPDATE	
	0003	SELECT	・When the DB Connection Service detected an error and could not send an SQL statement. ・When a failure occurred in information exchange between DB Connection Service and DB. ・When an SQL statement cannot be executed because one or more SQL statements are stored in the Spool memory.
	0004	DELETE	

*4 Log Name

Category	Operation	Log name
SQL Execution Failed	INSERT	INSERT
	UPDATE	UPDATE
	SELECT	SELECT
	DELETE	DELETE
Spooled	INSERT	INSERT
	UPDATE	UPDATE
Status Error	INSERT	INSERT
	UPDATE	UPDATE
	SELECT	SELECT
	DELETE	DELETE

Log file example:

```
1  2012-07-24  09:00:00  200  SQL_FAIL      0001  INSERT  0x300B  MyDatabase1  0   17072  ORA-17072: Inserted value too large for column
   INSERT INTO TABLE_Production(Column1) VALUES(‘1000’)
2  2012-07-24  09:01:13  550  SPOOL         0001  INSERT  0x3012  MyDatabase1  15  INSERT INTO TABLE_Production(Column2) VALUES(‘200’)
3  2012-07-24  09:01:14  050  SPOOL         0001  INSERT  0x3014  MyDatabase1  18  INSERT INTO TABLE_Production(Column2) VALUES(‘300’)
4  2012-07-24  09:01:14  550  STATUS_ERROR0001  INSERT  0x300C  MyDatabase1  19  INSERT INTO TABLE_Production(Column2) VALUES(‘400’)
```

Precautions for Correct Use

Do not delete the latest log file (DB_SQLFailedLog.log) and the log control file (DB_SQLFailedLog.fjc) from the SD Memory Card. If they are deleted, the log files are not saved correctly, for example, the SQL Execution Failure Log data are lost.

6-5 SD Memory Card Operations

In the DB Connection Service, the SD Memory Card mounted in the CPU Unit is used for the Operation Log function.
The Execution Log files, Debug Log files, and SQL Execution Failure Log files are stored in the SD Memory Card.
This section describes how to save the log files on the SD Memory Card and precautions for replacing the SD Memory Card.

Refer to the *NJ/NX-series CPU Unit Software User's Manual (W501)* for details of the SD Memory Card functions.

6-5-1 Saving Operation Log Files on SD Memory Card

Each Operation Log file is stored in the SD Memory Card in the following conditions.

Operation Logs	Operation to use the function	Conditions for saving log files on SD Memory Card
Execution Log	Set *Execution log* to *Record* in the DB Connection Service Settings of Sysmac Studio.	Constantly saved while the DB Connection Service is running.[*1]
Debug Log	Right-click **DB Connection Service Settings** in the Multiview Explorer on Sysmac Studio and select ***Online Settings*** from the menu. Then, click the **Start** Button for Debug Log in the Online Settings Tab Page. Or Execute a DB_ControlService (Control DB Connection Service) instruction to start recording to the Debug Log.	Constantly saved while the Debug Log is recorded.
SQL Execution Failure Log	Set *SQL execution failure log* to *Record* in the DB Connection Service Settings of Sysmac Studio.	Saved when transmission of an SQL statement failed due to a DB-caused factor.[*2]

*1 If the power supply to the CPU Unit is turned ON while no SD Memory Card is mounted in the CPU Unit, an Execution Log Save Failed Error is registered into the event log when the Execution Log is saved. Recording to the Execution Log is started when an SD Memory Card is inserted into the CPU Unit.

*2 If the power supply to the CPU Unit is turned ON while no SD Memory Card is mounted in the CPU Unit, an SQL Execution Failure Log Save Failed Error is registered into the event log when the SQL Execution Failure Log is saved. Recording to the SQL Execution Failure Log is started when an SD Memory Card is inserted into the CPU Unit.

6-5-2 Directory Used for DB Connection Service

The DB Connection Service uses the directory under *packages/DB_Connection* in the SD Memory Card.

packages/DB_Connection/System:	Contains log control files.
packages/DB_Connection/ExecutionLog:	Contains Execution Log files.
packages/DB_Connection/DebugLog:	Contains Debug Log files.
packages/DB_Connection/SQLFailedLog:	Contains SQL Execution Failure Log files.

6-5-3 Operation Log Operations in Replacing the SD Memory Card

This section describes operations of each Operation Log when the SD Memory Card is replaced while the DB Connection Service is running.

Operation Log function	SD Memory Card Replacing Status		
	When the SD Memory Card power supply switch is pressed	When no SD Memory Card is mounted	When an SD Memory Card is inserted
Execution Log	Continued If Execution Log is contained in the internal buffer of the CPU Unit, it is recorded into the SD Memory Card.	Temporarily recorded into the internal buffer of the CPU Unit.	The log that is temporarily recorded in the internal buffer is automatically recorded to the SD Memory Card.
Debug Log	Stopped. If Debug Log is contained in the internal buffer of the CPU Unit, it is recorded into the SD Memory Card.	Debug Log is not recorded.	Recording to the Debug Log is still stopped. Recording is started by an online operation from Sysmac Studio or by executing a DB_ControlService (Control DB Connection Service) instruction.
SQL Execution Failure Log	Stopped. If SQL Execution Failure Log is contained in the internal buffer of the CPU Unit, it is recorded into the SD Memory Card.	SQL Execution Failure Log is not recorded.	Recording to the SQL Execution Failure Log is automatically started.

Precautions for Correct Use.

Please note the following for replacing the SD Memory Card.

- Use a formatted SD Memory Card when replacing the SD Memory Card.
- When you replace the SD Memory Card while recording the Execution Log, press the SD Memory Card power supply switch and insert a new SD Memory Card within five minutes after the SD PWR indicator is turned OFF. If it takes more than five minutes, Execution Log recorded in the internal buffer may be lost.
  If the internal buffer space becomes full before inserting the SD Memory Card, an Execution Log Save Failed Error is registered into the event log.

6-5-4 Replacement Timing of SD Memory Card

● How to Know the Replacement Timing of the SD Memory Card

You can know the replacement timing of the SD Memory Card by the SD Memory Card Life Exceeded Event or the SD Memory Card Life Warning Flag (_*Card1Deteriorated* system-defined variable).

6-6 Checking the Operation Logs

This section describes how to check the Operation Logs stored on the SD Memory Card mounted in the CPU Unit.

6-6-1 How to Check the Operation Logs

You can use the following methods to check the Operation Logs (i.e., Execution Log, Debug Log, and SQL Execution Failure Log).

- Checking the log on the Operation Log Window in Sysmac Studio
- Checking the log with the SD Memory Card
- Checking the log by transferring data using FTP client software

Precautions for Correct Use

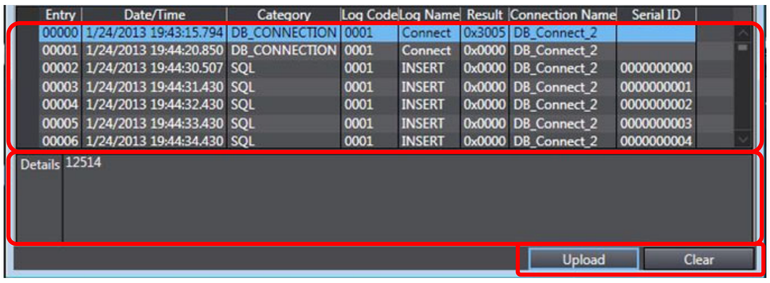
Each Operation Log file is encoded by the UTF-8 character code.

6-6-2 Checking the Log on the Operation Log Window in Sysmac Studio

You can check the Operation Logs (i.e., Execution Log, Debug Log, and SQL Execution Failure Log) stored in the SD Memory Card on the Operation Log Window in Sysmac Studio while online with the CPU Unit.

1. Right-click **DB Connection** under **Configurations and Setup** - **Host Connection Settings** in the Multiview Explorer and select ***Show Operation Logs*** from the menu while online with the CPU Unit.
The Execution Log, Debug Log, and SQL Execution Failure Log are displayed in the different tab pages.

2. Click the Execution Log Tab, Debug Log Tab, or SQL Execution Failure Log Tab.



Entry	Date/Time	Category	Log Code	Log Name	Result	Connection Name	Serial ID
00000	1/24/2013 19:43:15.794	DB_CONNECTION	0001	Connect	0x3005	DB_Connect_2	
00001	1/24/2013 19:44:20.850	DB_CONNECTION	0001	Connect	0x0000	DB_Connect_2	
00002	1/24/2013 19:44:30.507	SQL	0001	INSERT	0x0000	DB_Connect_2	0000000000
00003	1/24/2013 19:44:31.430	SQL	0001	INSERT	0x0000	DB_Connect_2	0000000001
00004	1/24/2013 19:44:32.430	SQL	0001	INSERT	0x0000	DB_Connect_2	0000000002
00005	1/24/2013 19:44:33.430	SQL	0001	INSERT	0x0000	DB_Connect_2	0000000003
00006	1/24/2013 19:44:34.430	SQL	0001	INSERT	0x0000	DB_Connect_2	0000000004

Details 12514

The following information is displayed.

- List view

Item	Description
Entry	Displays a serial number.
Date/Time	Displays a date and time.
Category	Displays a category.
Log Code	Displays a log code.
Log Name	Displays a log name.
Result	Displays results.
Connection Name	Displays a DB Connection name.
Serial ID	Displays a serial ID.

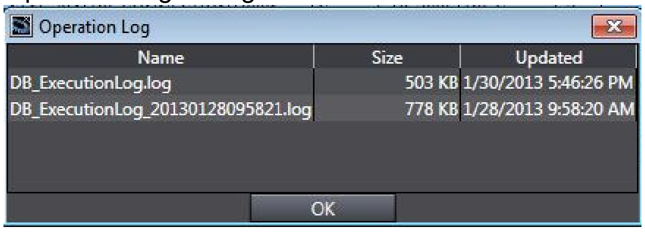
• Detailed information

The Details parameter of the log is displayed.

• Buttons

Upload Button:

The log files are uploaded from the Controller. A list of log files is displayed in the following Operation Log Dialog Box.



Operation Log

Name	Size	Updated
DB_ExecutionLog.log	503 KB	1/30/2013 5:46:26 PM
DB_ExecutionLog_20130128095821.log	778 KB	1/28/2013 9:58:20 AM

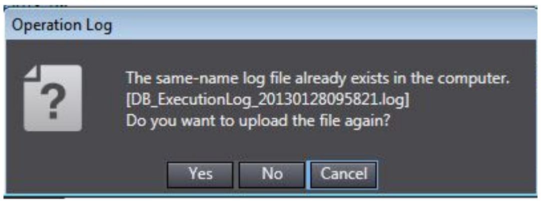
Select a log file to display and click the **OK** Button. The log file is uploaded.

• Execution Log Tab Page: Execution Log is uploaded from the Controller.

• Debug Log Tab Page: Debug Log is uploaded from the Controller.

• SQL Execution Failure Log Tab Page: SQL Execution Failure Log is uploaded from the Controller.

Note 1 If the same-name log file exists in the computer, the following message is displayed.



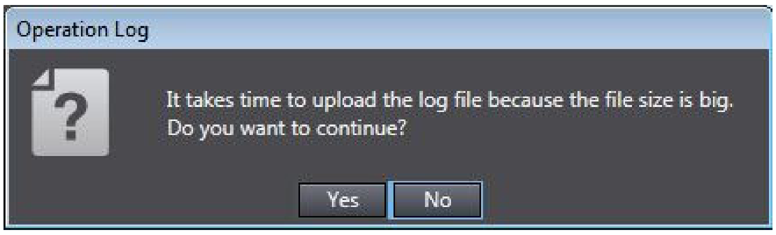
Operation Log

The same-name log file already exists in the computer.
[DB_ExecutionLog_20130128095821.log]
Do you want to upload the file again?

Click a button.

Yes:	The specified file is uploaded from the Controller and displayed.
No:	The specified file is not uploaded from the Controller and the contents of the file that already exists in the computer are displayed.
Cancel:	The file list is displayed again.

Note 2 If the selected log file is bigger than 10 MB, the following message is displayed.



Operation Log

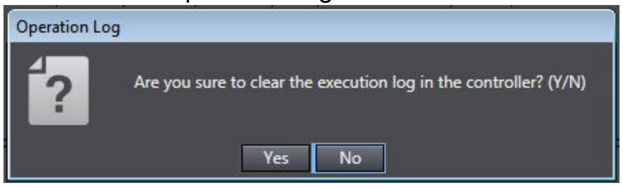
It takes time to upload the log file because the file size is big.
Do you want to continue?

Click a button.

Yes:	The specified file is uploaded from the Controller and displayed.
No:	The file list is displayed again.

Clear Button:

The selected Operation Log is cleared in the Controller. A confirmation message is displayed.



Operation Log

Are you sure to clear the execution log in the controller? (Y/N)

When you click the **Yes** Button, the selected log is cleared.

- Execution Log Tab Page: Execution Log is cleared in the Controller.
- Debug Log Tab Page: Debug Log is cleared in the Controller.
- SQL Execution Failure Log Tab Page: SQL Execution Failure Log is cleared in the Controller.

6-6-3 Checking the Log with the SD Memory Card

Remove the SD Memory Card from the CPU Unit and insert it into a computer. Then, check the contents of the logs on Microsoft Excel or a text editor.

6-6-4 Checking the Log by Transfer using FTP Client Software

You can transfer the log files using the FTP Server function via the Ethernet network and check the contents on Microsoft Excel or a text editor.
Use the following procedure.
You use the FTP Server function of the built-in EtherNet/IP port.

1. Double-click **Built-in EtherNet/IP Port Settings** under **Configurations and Setup** - **Controller Setup** in the Multiview Explorer and set *FTP server* to *Use* in the FTP Settings.
2. Log into the CPU Unit using the FTP client software.
3. Transfer Operation Log files.
   You can transfer more than one log file by using a wildcard in the Mget command.
   Example: mget DB_ExecutionLog_*.log
4. Disconnect the FTP client software from the CPU Unit.
5. Open the transferred Operation Log files on Microsoft Excel or a text editor to check the contents.

7

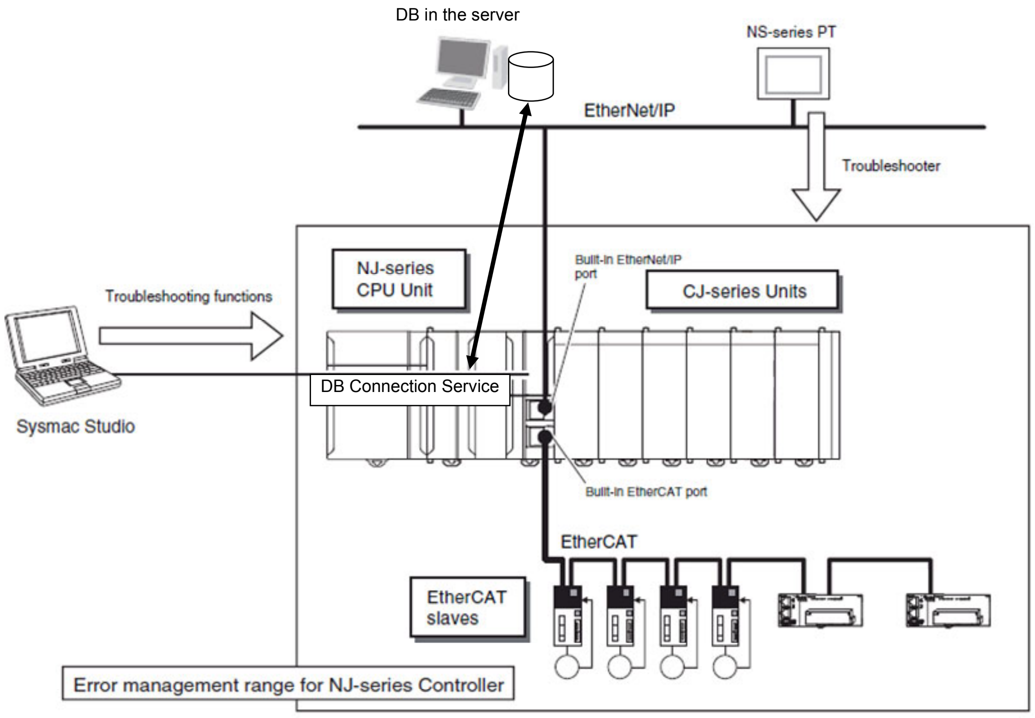
Troubleshooting

This section describes the error confirmation methods and corrections for errors that can occur in the DB Connection Service.

7-1 Overview of Errors

You manage all of the errors that occur on the NJ-series Controller as events. The same methods are used for all events. This allows you to see what errors have occurred and find corrections for them with the same methods for the entire range of errors that is managed (i.e., CPU Unit, EtherCAT slaves,* and CJ-series Units).

* Only Sysmac devices are supported.



You can use the troubleshooting functions of Sysmac Studio or the Troubleshooter on an NS-series PT to quickly check for errors that have occurred and find corrections for them.

This manual describes the errors that originate in the DB Connection Service. Refer to the *NJ/NX-series Troubleshooting Manual* (Cat. No. W503) for specific corrections when errors occur and for troubleshooting information on the entire NJ-series Controller. For information on errors that occur when DB Connection Instructions are executed, refer to *Appendix DB Connection Instructions*.

7-1-1 How to Check for Errors

You can check to see if an error has occurred with the following methods.

Checking method	What you can check
Checking the indicators	CPU Unit operating status
Troubleshooter of Sysmac Studio	You can check for current Controller errors, a log of past Controller errors, error sources, error causes, corrections, and error log of CJ-series Special Units.*1
Checking with the Troubleshooter of an NS-series PT*2	You can check for current Controller errors, a log of past Controller errors, error sources, causes, and corrections.
Checking with instructions that read function module error status	You can check the highest-level status and highest-level event code in the current Controller errors.
Checking with system-defined variables	You can check the current Controller error status for each function module.

*1 Detailed information such as error causes and corrections are not displayed.

*2 To perform troubleshooting from an NS-series PT, connect the PT to the built-in EtherNet/IP port on the CPU Unit.

This section describes the above checking methods.

Checking the Indicators

You can use the PWR indicator on the Power Supply Unit and the RUN and ERROR indicators on the CPU Unit to determine the event level for an error. The following table shows the relationship between the Controller's indicators and the event level.

Indicator			CPU Unit operating status	Error confirmation with Sysmac Studio or an NS-series PT
PWR	RUN	ERROR		
Not lit	Not lit	Not lit	Power Supply Error	Not possible: Refer to the *NJ/NX-series Troubleshooting Manual* (Cat. No. W503).
Lit	Not lit	Not lit	CPU Unit Reset*1	
Lit	Flashing	Lit	Incorrect Power Supply Unit Connected	
Lit	Not lit	Lit	CPU Unit Watchdog Timer Error*2	
Lit	Not lit	Lit	Major fault level*2	Possible: Connect Sysmac Studio or an NS-series PT and check the cause of and correction for the error in the troubleshooting functions of Sysmac Studio or the Troubleshooter of the NS-series PT.
Lit	Lit	Flashing	Partial fault level	
Lit	Lit	Flashing	Minor fault level	
Lit	Lit	Not lit	Observation	
Lit	Lit	Not lit	Normal operation in RUN mode	---
Lit	Not lit	Not lit	Normal operation in PROGRAM mode*1	---
Lit	Flashing	Not lit	Normal operation in startup state	---

*1 If you can go online with the CPU Unit from Sysmac Studio with a direct USB connection, the CPU Unit is in PROGRAM mode. If you cannot go online, the CPU Unit is being reset.*3

*2 If you can go online with the CPU Unit from Sysmac Studio with a direct USB connection, a major fault level error has occurred. If you cannot go online, a watchdog timer error has occurred in the CPU Unit.*3

*3 If you cannot go online with the CPU Unit from Sysmac Studio, it is also possible that the USB cable is faulty or that the network type on Sysmac Studio is not set for a direct USB connection. Refer to the *NJ/NX-series Troubleshooting Manual* (Cat. No. W503) if you cannot go online with the CPU Unit.

Checking with the Troubleshooting Function of Sysmac Studio

When an error occurs, you can connect Sysmac Studio online to the Controller to check current Controller errors and the log of past Controller errors. You can also check the cause of the error and corrections.
Refer to the *NJ/NX-series Troubleshooting Manual* (Cat. No. W503) for the procedures to check for errors with Sysmac Studio.

Checking with the Troubleshooter of an NS-series PT

If you can connect communications between an NS-series PT and the Controller when an error occurs, you can check for current Controller errors and the log of past Controller errors. You can also check the cause of the error and corrections.

Refer to the *NJ/NX-series Troubleshooting Manual* (Cat. No. W503) for the procedures to check for errors with an NS-series PT.

Checking with Instructions That Read Error Status

You can use instructions in the user program to check the error status of each function module. The following table gives the instruction that is used to get error information for the DB Connection Service.

Instruction	Name	Function
GetPLCError	Get PLC Error Status	The GetPLCError instruction gets the highest level status (partial fault or minor fault) and highest level event code of the current Controller errors in the PLC Function Module.

For details on the instructions that get error status, refer to the *NJ/NX-series Instructions Reference Manual* (Cat. No. W502).

Checking with System-defined Variables

You can use the error status variables and status variables in the system-defined variables to check for errors that have occurred in the DB Connection Service.

● Error Status Variables

You can check for errors in each function module of the NJ-series Controller with error status variables. The following variables show the error status of the PLC Function Module.

Variable name	Data type	Meaning	Function
_PLC_ErrSta	WORD	PLC Function Module Error Status	Gets the collective error status of all error status for the PLC Function Module.

● Status Variables

Variable name		Data type	Meaning	Function
_DBC_Status		_sDBC_STATUS	DB Connection Service Status	Shows the status of the DB Connection Service.
	Run	BOOL	Running Flag	TRUE while the DB Connection Service is running. FALSE while the DB Connection Service is not running.
	Test	BOOL	Test Mode	TRUE while the DB Connection Service is running in Test Mode. FALSE while the DB Connection Service is not running in Test Mode.
	Idle	BOOL	Idle	TRUE while the DB Connection Service is idle. FALSE while the DB Connection Service is not idle.
	Error	BOOL	Error Flag	TRUE when the DB Connection Service has an error. FALSE when the DB Connection Service has no error.
	Shutdown	BOOL	Shutdown	TRUE when the DB Connection Service has been shut down. FALSE when the DB Connection Service has not been shut down.

7-1-2 Errors Related to the DB Connection Service

Classifications

There are the following two sources of errors in the DB Connection Service.

Classification	Event source	Source details	Log category		
			System log	Access log	User-defined event log
DB Connection Service	PLC Function Module	DB Connection Service	Yes	No	No
DB Connection Instruction	PLC Function Module	Instruction	Yes	No	No

Event Levels

This section describes the operation of the DB Connection Service for each event level.

Event level of the error	Operation
Major fault	All NJ-series Controller control operations stop for errors in this event level.
Partial fault	All control operations for one of the function modules in the NJ-series Controller stop for errors in this event level. If a partial fault level error occurs in the DB Connection Service, all functions of the DB Connection Service stop.
Minor fault	Some of the control operations for one of the function modules in the NJ-series Controller stop for errors in this event level.
Observation	Errors in the observation level do not affect NJ-series Controller control operations. Observations are reported in order to prevent them from developing into errors at the minor fault level or higher.
Information	Events that are classified as information provide information that do not indicate errors.

DB Connection Service Errors by Source

The following tables list the errors in each event level that can occur for each source.

DB Connection Service Errors

Level	Error name
Major fault	None
Partial fault	None
Minor fault	- Spool Memory Corrupted - Execution Log Save Failed - SQL Execution Failure Log Save Failed - DB Connection Setting Error - DB Connection Disconnected Error
Observation	None
Information	- DB Connection Service Started - DB Connection Service Stopped - DB Connection Service Shutdown

DB Connection Instruction Errors

Level	Error name
Major fault	None
Partial fault	None
Minor fault	None
Observation	・DB Connection Service Not Started ・DB Connection Service Run Mode Change Failed ・DB Connection Service Shutdown or Shutting Down ・Invalid DB Connection Name ・DB Connection Rejected ・DB Connection Failed ・DB Connection Already Established ・Too Many DB Connections ・Invalid DB Connection ・Invalid DB Map Variable ・Unregistered DB Map Variable ・SQL Execution Error ・Spool Capacity Exceeded ・Invalid Extraction Condition ・Log Code Out of Range ・DB Connection Disconnected Error Status ・DB Connection Instruction Execution Timeout ・DB Connection Service Error Stop ・Data Already Spooled ・DB Connection Service Initializing ・DB in Process ・Operation Log Disabled
Information	None

7-2 Troubleshooting

This section describes the errors that can occur in the DB Connection Service and the corrections for them.

7-2-1 Error Table

The errors (i.e., events) that can occur in the DB Connection Service and DB Connection Instructions are given on the following pages.
The following abbreviations and symbols are used in the event level column.

Abbreviation	Name
Maj	Major fault level
Prt	Partial fault level
Min	Minor fault level
Obs	Observation
Info	Information

Symbol	Meaning
S	Event levels that are defined by the system.
U	Event levels that can be changed by the user.*

* This symbol appears only for events for which the user can change the event level.

Refer to the *NJ/NX-series Troubleshooting Manual* (Cat. No. W503) for all NJ-series event codes.

Errors Related to DB Connection Service

Event code	Event name	Meaning	Assumed cause	Level					Reference
				Maj	Prt	Min	Obs	Info	
14D0 0000 hex	Spool Memory Corrupted	The Spool memory is corrupted.	• The user application made an invalid writing to the Spool memory.			S			7-16
14D2 0000 hex	Execution Log Save Failed	Failed to save the Execution Log to the SD Memory Card.	• An SD Memory Card is not inserted. • The SD Memory Card is not the correct type of card. • The format of the SD Memory Card is not correct. • The SD Memory Card is write-protected. • The capacity of the SD Memory Card is insufficient. • The SD Memory Card is damaged.			S	U		7-17
14D3 0000 hex	SQL Execution Failure Log Save Failed	Failed to save the SQL Execution Failure Log to the SD Memory Card.	• An SD Memory Card is not inserted. • The SD Memory Card is not the correct type of card. • The format of the SD Memory Card is not correct. • The SD Memory Card is write-protected. • The capacity of the SD Memory Card is insufficient. • The SD Memory Card is damaged.			S	U		7-18
3530 0000 hex	DB Connection Setting Error	The DB Connection settings are not correct.	• The power supply to the Controller was interrupted during a download of the DB Connection settings. • The DB Connection settings are not correct because the power supply to the Controller was interrupted during a Clear All Memory operation. • The DB Connection settings are not correct because the power supply to the Controller was interrupted during a Restore operation. • Non-volatile memory failed.			S			7-19
8510 0000 hex	DB Connection Disconnected Error	The DB Connection was disconnected due to an error.	• The power supply to the server is OFF. • The DB is stopped in the server. • The Ethernet cable connector is disconnected. • The Ethernet cable is broken. • Noise			S			7-20

Event code	Event name	Meaning	Assumed cause	Level					Reference
				Maj	Prt	Min	Obs	Info	
95300000hex	DB Connection Service Started	The DB Connection Service was started.	• The DB Connection Service was successfully started.					S	7-21
95310000hex	DB Connection Service Stopped	The DB Connection Service was stopped.	• The DB Connection Service was stopped.					S	7-21
95320000hex	DB Connection Service Shutdown	The DB Connection Service was shut down.	• The DB Connection Service was shut down.					S	7-22

Errors Related to DB Connection Instructions

Errors are given as event codes that use the error code as the lower four digits. For descriptions of an error code, refer to the description of the corresponding event code. For example, if the error code for the instruction is 16#3000, refer to the description for event code 5401 3000 hex.

Event code	Event name	Meaning	Assumed cause	Level					Reference
				Maj	Prt	Min	Obs	Info	
54013000hex	DB Connection Service Not Started	The DB Connection Service has not been started.	• A command to start the DB Connection Service was not given before the execution of relevant instruction. • A command to stop the DB Connection Service was given before the execution of relevant instruction.				S		7-23
5401 3001 hex	DB Connection Service Run Mode Change Failed	Failed to change the Run mode of the DB Connection Service.	• Run mode change to Test Mode was executed by the relevant instruction while running in Operation Mode. • Run mode change to Operation Mode was executed by the relevant instruction while running in Test Mode. • Start of the DB Connection Service was commanded while the DB Connection Service was being stopped. • Shutdown of the DB Connection Service was commanded while the DB Connection Service was being stopped.				S		7-24
54013002hex	DB Connection Service Shutdown or Shutting Down	The DB Connection Service is already shut down or being shut down.	• The relevant instruction was executed after the DB Connection Service was shut down. • The relevant instruction was executed while the shutdown processing of the DB Connection Service was in progress.				S		7-25
5401 3003 hex	Invalid DB Connection Name	The specified DB Connection Name is not set in any DB Connection settings.	• The DB Connection Name specified in the *DBConnectionName* input variable of the relevant instruction is wrong. • The DB Connection Name set in the DB Connection settings is wrong.				S		7-25
5401 3004 hex	DB Connection Rejected	The DB rejected the connection.	• The user name or password set in the DB Connection settings is wrong.				S		7-26

Event code	Event name	Meaning	Assumed cause	Level					Reference
				Maj	Prt	Min	Obs	Info	
54013005hex	DB Connection Failed	Failed to connect to the DB.	• A server does not exist for the specified IP address or the specified host name. • The power supply to the server is OFF. • The DB is stopped in the server. • The Ethernet cable connector is disconnected. • The Ethernet cable is broken.				S		7-26
5401 3006 hex	DB Connection Already Established	A same-name DB Connection is already established.	• The relevant instruction was executed when a same-name DB Connection was already established.				S		7-27
5401 3007 hex	Too Many DB Connections	The number of DB Connections that can be established at the same time is exceeded.	• The relevant instruction was executed when the maximum number of DB Connections that can be established at the same time were already established.				S		7-27
5401 3008 hex	Invalid DB Connection	The specified DB Connection is not correct, or the DB Connection is already closed.	• The DB Connection specified in the DBConnection input variable of the relevant instruction is wrong. • The DB Connection specified in the DBConnection input variable of the relevant instruction is closed.				S		7-28
5401 3009 hex	Invalid DB Map Variable	The specified DB Map Variable is not correct.	• A structure variable that contains a derivative data type of member was specified as a DB Map Variable. • A non-structure variable was specified as a DB Map Variable. • A structure array variable was specified as a DB Map Variable for INSERT or UPDATE.				S		7-29
5401 300A hex	Unregistered DB Map Variable	The specified DB Map Variable has not been registered.	• The DB Map Variable has not been created by a DB_CreateMapping instruction. • A variable that is not registered as a DB Map Variable was specified in *MapVar*. • The DB Connection specified in the relevant instruction is different from the one specified at the execution of DB_CreateMapping instruction.				S		7-30

Event code	Event name	Meaning	Assumed cause	Level					Reference
				Maj	Prt	Min	Obs	Info	
5401300B hex	SQL Execution Error	The executed SQL statement resulted in an error.	• There is no column with the same name as a structure member of the DB Map Variable. • The table specified in the DB_CreateMapping instruction does not exist in the DB. • One or more structure member values of the DB Map Variable cannot be converted to the corresponding column's data type. • One or more column values cannot be converted to the corresponding structure member's data type of the DB Map Variable. • One or more structure member values of the DB Map Variable exceed the valid range of the corresponding column's data type. • The column specified in the extraction condition does not exist in the DB's records. (DB_Select instruction, DB_Update instruction, DB_Delete instruction) • The extraction condition has a syntax error. (DB_Select instruction, DB_Update instruction, DB_Delete instruction) • The column specified in the sort condition does not exist in the DB's records. (DB_Select instruction) • The sort condition has a syntax error. (DB_Select instruction) • The user does not have the access rights to the table.				S		7-31
5401300C hex	Spool Capacity Exceeded	The SQL statement could not be stored in the Spool memory because its maximum capacity was exceeded.	• The DB connection failure has been continuing due to network failure or other factors. • The resend processing of the SQL statements stored in the Spool memory has not been executed (when the *Resend spool data* parameter is set to *Manual*).				S		7-33
5401300E hex	Invalid Extraction Condition	The entered extraction condition is invalid.	• A text string that consists of a NULL (16#00) character only was specified in the *Where* input variable.				S		7-34

Event code	Event name	Meaning	Assumed cause	Level					Reference
				Maj	Prt	Min	Obs	Info	
54013010 hex	Log Code Out of Range	The value of the entered log code is outside the valid range.	• A value outside the valid range from 0 to 9999 was specified.				S		7-35
54013011 hex	DB Connection Disconnected Error Status	The instruction could not be executed because the DB Connection had been disconnected due to an error.	• The power supply to the server is OFF. • The DB is stopped in the server. • The Ethernet cable connector is disconnected. • The Ethernet cable is broken. • Noise				S		7-35
54013012 hex	DB Connection Instruction Execution Timeout	The instruction was not completed within the time specified for timeout.	• The power supply to the server is OFF. • The Ethernet cable connector is disconnected. • The Ethernet cable is broken. • The server's processing time is long.				S		7-36
54013013 hex	DB Connection Service Error Stop	The instruction could not be executed because the DB Connection Service was stopped due to an error.	• The DB Connection settings are corrupted.				S		7-36
54013014 hex	Data Already Spooled	One or more SQL statements are already stored in the Spool memory.	• A DB_Insert or DB_Update instruction was executed when one or more SQL statements were already stored in the Spool memory. • A DB_Select or DB_Delete instruction was executed when one or more SQL statements were already stored in the Spool memory.				S		7-37
54013015 hex	DB Connection Service Initializing	The instruction could not be executed because the initialization processing of the DB Connection Service is in progress.	• The relevant instruction was executed during the initialization processing of the DB Connection Service.				S		7-37
54013016 hex	DB in Process	The instruction could not be executed because the DB is under processing in the server.	• Though a DB Connection Instruction Execution Timeout occurred for the previous instruction, the relevant instruction was executed before completion of the DB's processing in the server.				S		7-38

Event code	Event name	Meaning	Assumed cause	Level					Reference
				Maj	Prt	Min	Obs	Info	
54013017 hex	Operation Log Disabled	The log could not be recorded because the specified Operation Log is disabled.	• Though Execution Log was specified in the *LogType* input variable, the Execution Log is disabled. • Though Debug Log was specified in the *LogType* input variable, recording to the Debug Log is stopped.				S		7-38

7-2-2 Error Descriptions

Controller Error Descriptions

The items that are used to describe individual errors (events) are described in the following copy of an error table.

Event name	Gives the name of the error.			Event code	Gives the code of the error.	
Meaning	Gives a short description of the error.					
Source	Gives the source of the error.		Source details	Gives details on the source of the error.	Detection timing	Tells when the error is detected.
Error attributes	Level	Tells the level of influence on control.*1	Recovery	Gives the recovery method.*2	Log category	Tells which log the error is saved in.*3
Effects	User program	Tells what will happen to execution of the user program.*4	Operation	Provides special information on the operation that results from the error.		
System-defined variables	Variable		Data type		Name	
	Lists the variable names, data types, and meanings for system-defined variables that provide direct error notification, that are directly affected by the error, or that contain settings that cause the error.					
Cause and correction	Assumed cause		Correction		Prevention	
	Lists the possible causes, corrections, and preventive measures for the error.					
Attached information	This is the attached information that is displayed by Sysmac Studio or an NS-series PT.					
Precautions/ Remarks	Provides precautions, restrictions, and supplemental information. If the user can set the event level, the event levels that can be set, the recovery method, operational information, and other information is also provided.					

*1 One of the following:
Major fault: Major fault level
Partial fault: Partial fault level
Minor fault: Minor fault level
Observation
Information

*2 One of the following:
Automatic recovery: Normal status is restored automatically when the cause of the error is removed.
Error reset: Normal status is restored when the error is reset after the cause of the error is removed.
Cycle the power supply: Normal status is restored when the power supply to the Controller is turned OFF and then back ON after the cause of the error is removed.
Controller reset: Normal status is restored when the Controller is reset after the cause of the error is removed.
Depends on cause: The recovery method depends on the cause of the error.

*3 One of the following:
System: System event log
Access: Access event log

*4 One of the following:
Continues: Execution of the user program will continue.
Stops: Execution of the user program stops.
Starts: Execution of the user program starts.

Errors Related to DB Connection Service

Event name	Spool Memory Corrupted			Event code	14D00000hex		
Meaning	The Spool memory is corrupted.						
Source	PLC Function Module		Source details	DB Connection Service	Detection timing	When the DB Connection Service is started	
Error attribute	Level	Minor fault	Recovery	Error reset	Log category	System	
Effects	User program	Continues.	Operation	Not affected.			
System-defined variables	Variable		Data type		Name		
	None		---		---		
Cause and correction	Assumed cause		Correction		Prevention		
	The user application made an invalid writing to the Spool memory.		Check for writing from the user application to the Spool memory area. Correct the user application, and then execute the Clear Spool Data operation.		Do not write to the Spool memory area from the user application.		
Attached information	None						
Precautions/ Remarks	None						

Event name	Execution Log Save Failed			Event code	14D20000hex	
Meaning	Failed to save the Execution Log to the SD Memory Card.					
Source	PLC Function Module		Source details	DB Connection Service	Detection timing	Continuously
Error attributes	Level	Minor fault	Recovery	Error reset	Log category	System
Effects	User program	Continues.	Operation	Not affected		
System-defined variables	Variable		Data type		Name	
	None		---		---	
Cause and correction	Assumed cause		Correction		Prevention	
	An SD Memory Card is not inserted.		Insert an SD Memory Card.		Insert an SD Memory Card.	
	The SD Memory Card is not the correct type of card.		Replace the SD Memory Card with an SD or SDHC card.		Use an SD or SDHC card.	
	The format of the SD Memory Card is not correct.		Format the SD Memory Card with Sysmac Studio.		Use a formatted SD Memory Card. Also, do not remove the SD Memory Card or turn OFF the power supply while the SD BUSY indicator is lit.	
	The SD Memory Card is write-protected.		Remove write protection from the SD Memory Card.		Make sure that the SD Memory Card is not write-protected.	
	The capacity of the SD Memory Card is insufficient.		Replace the SD Memory Card for one with sufficient available space.		Use an SD Memory Card that has sufficient available space.	
	The SD Memory Card is damaged.		If none of the above causes applies, replace the SD Memory Card.		Do not remove the SD Memory Card or turn OFF the power supply while the SD BUSY indicator is lit. Do not remove the SD Memory Card while the SD PWR indicator is lit. Replace the SD Memory Card periodically according to the write life of the SD Memory Card.	
Attached information	Attached information 1: Error Details 0001 hex: An SD Memory Card is not inserted. 0002 hex: The SD Memory Card is damaged, the format of the SD Memory Card is not correct, or the SD Memory Card is not the correct type of card. 0003 hex: The SD Memory Card is write-protected. 0005 hex: The capacity of the SD Memory Card is insufficient. 0302 hex: The SD Memory Card is damaged or failed to save a file to the SD Memory Card due to other factors.					
Precautions/ Remarks	You can change the error level to the observation.					

Event name	SQL Execution Failure Log Save Failed			Event code	14D30000hex	
Meaning	Failed to save the SQL Execution Failure Log to the SD Memory Card.					
Source	PLC Function Module		Source details	DB Connection Service	Detection timing	Continuously
Error attributes	Level	Minor fault	Recovery	Error reset	Log category	System
Effects	User program	Continues.	Operation	Not affected.		
System-defined variables	Variable		Data type		Name	
	None		---		---	
Cause and correction	Assumed cause		Correction		Prevention	
	An SD Memory Card is not inserted.		Insert an SD Memory Card.		Insert an SD Memory Card.	
	The SD Memory Card is not the correct type of card.		Replace the SD Memory Card with an SD or SDHC card.		Use an SD or SDHC card.	
	The format of the SD Memory Card is not correct.		Format the SD Memory Card with Sysmac Studio.		Use a formatted SD Memory Card. Also, do not remove the SD Memory Card or turn OFF the power supply while the SD BUSY indicator is lit.	
	The SD Memory Card is write-protected.		Remove write protection from the SD Memory Card.		Make sure that the SD Memory Card is not write-protected.	
	The capacity of the SD Memory Card is insufficient.		Replace the SD Memory Card for one with sufficient available space.		Use an SD Memory Card that has sufficient available space.	
	The SD Memory Card is damaged.		If none of the above causes applies, replace the SD Memory Card.		Do not remove the SD Memory Card or turn OFF the power supply while the SD BUSY indicator is lit. Do not remove the SD Memory Card while the SD PWR indicator is lit. Replace the SD Memory Card periodically according to the write life of the SD Memory Card.	
Attached information	Attached information 1: Error Details 0001 hex: An SD Memory Card is not inserted. 0002 hex: The SD Memory Card is damaged, the format of the SD Memory Card is not correct, or the SD Memory Card is not the correct type of card. 0003 hex: The SD Memory Card is write-protected. 0005 hex: The capacity of the SD Memory Card is insufficient. 0302 hex: The SD Memory Card is damaged or failed to save a file to the SD Memory Card due to other factors.					
Precautions/ Remarks	You can change the error level to the observation.					

Event name	DB Connection Setting Error			Event code	35300000hex	
Meaning	The DB Connection settings are not correct.					
Source	PLC Function Module		Source details	DB Connection Service	Detection timing	At download, power ON, or Controller reset
Error attributes	Level	Minor fault	Recovery	Automatic recovery	Log category	System
Effects	User program	Continues.	Operation	The DB Connection Service cannot be started. The operation status of the DB Connection Service is changed to Error Stop.		

System-defined variables	Variable	Data type	Name
	_DBC_Status	_sDBC_STATUS	DB Connection Service Status
Cause and correction	Assumed cause	Correction	Prevention
	The power supply to the Controller was interrupted during a download of the DB Connection settings.	Transfer the DB Connection settings again from Sysmac Studio.	Do not turn OFF the power supply to the Controller during a download of the user program or the Controller Configurations and Setup.
	The DB Connection settings are not correct because the power supply to the Controller was interrupted during a Clear All Memory operation.		Do not interrupt the power supply to the Controller during a Clear All Memory operation.
	The DB Connection settings are not correct because the power supply to the Controller was interrupted during a Restore operation.		Do not interrupt the power supply to the Controller during a Restore operation.
	Non-volatile memory failed.	If the error persists even after you make the above correction, replace the CPU Unit.	None
Attached information	None		
Precautions/ Remarks	None		

Event name	DB Connection Disconnected Error			Event code	85100000hex	
Meaning	The DB Connection was disconnected due to an error.					
Source	PLC Function Module		Source details	DB Connection Service	Detection timing	When a DB Connection Instruction is executed, or when Spool data is resent
Error attributes	Level	Minor fault	Recovery	Automatic recovery	Log category	System
Effects	User program	Continues.	Operation	Not affected.		
System-defined variables	Variable		Data type		Name	
	_DBC_Status		_sDBC_STATUS		DB Connection Service Status	
Cause and correction	Assumed cause		Correction		Prevention	
	The power supply to the server is OFF.		Check the server status and start it properly.		Check the server status and start it properly.	
	The DB is stopped in the server.					
	The Ethernet cable connector is disconnected.		Reconnect the connector and make sure it is mated correctly.		Connect the connector securely.	
	The Ethernet cable is broken.		Replace the Ethernet cable.		None	
	Noise		Implement noise countermeasures if there is excessive noise.		Implement noise countermeasures if there is excessive noise.	
Attached information	Attached information 1: DB Connection Name					
Precautions/ Remarks	None					

Event name	DB Connection Service Started			Event code	9530 0000 hex	
Meaning	The DB Connection Service was started.					
Source	PLC Function Module		Source details	DB Connection Service	Detection timing	When the DB Connection Service is started
Error attributes	Level	Information	Recovery	---	Log category	System
Effects	User program	Continues.	Operation	Not affected.		
System-defined variables	Variable		Data type		Name	
	_DBC_Status		_sDBC_STATUS		DB Connection Service Status	
Cause and correction	Assumed cause		Correction		Prevention	
	The DB Connection Service was successfully started.		---		---	
Attached information	Attached information 1: Start reason 01 hex: Execution of a DB_ControlService instruction or operation from Sysmac Studio 02 hex: Controller's operating mode change (from PROGRAM to RUN mode)					
Precautions/ Remarks	None					

Event name	DB Connection Service Stopped			Event code	9531 0000 hex	
Meaning	The DB Connection Service was stopped.					
Source	PLC Function Module		Source details	DB Connection Service	Detection timing	When the DB Connection Service is stopped
Error attributes	Level	Information	Recovery	---	Log category	System
Effects	User program	Continues.	Operation	Not affected.		
System-defined variables	Variable		Data type		Name	
	_DBC_Status		_sDBC_STATUS		DB Connection Service Status	
Cause and correction	Assumed cause		Correction		Prevention	
	The DB Connection Service was stopped.		---		---	
Attached information	Attached information 1: Stop reason 01 hex: Execution of a DB_ControlService instruction or operation from Sysmac Studio 02 hex: Controller's operating mode change (from RUN to PROGRAM mode) 03 hex: Execution of Synchronization (download), Clear All Memory, or Restore operation 04 hex: A major fault level Controller error					
Precautions/ Remarks	None					

Event name	DB Connection Service Shutdown			Event code	9532 0000 hex	
Meaning	The DB Connection Service was shut down.					
Source	PLC Function Module		Source details	DB Connection Service	Detection timing	When the DB Connection Service is shut down.
Error attributes	Level	Information	Recovery	---	Log category	System
Effects	User program	Continues.	Operation	Not affected.		
System-defined variables	Variable		Data type		Name	
	_DBC_Status		_sDBC_STATUS		DB Connection Service Status	
Cause and correction	Assumed cause		Correction		Prevention	
	The DB Connection service was shut down.		---		---	
Attached information	Attached information 1: Shutdown reason 01 hex: Execution of a DB_Shutdown instruction or operation from Sysmac Studio					
Precautions/ Remarks	None					

Errors Related to DB Connection Instructions

Event name	DB Connection Service Not Started			Event code	54013000 hex	
Meaning	The DB Connection Service has not been started.					
Source	PLC Function Module		Source details	Instruction	Detection timing	At instruction execution
Error attributes	Level	Observation	Recovery	---	Log category	System
Effects	User program	Continues.	Operation	The relevant instruction will end according to specifications.		
System-defined variables	Variable		Data type		Name	
	None		---		---	
Cause and correction	Assumed cause		Correction		Prevention	
	A command to start the DB Connection Service was not given before the execution of relevant instruction.		Start the DB Connection Service. Or, correct the user program so that the relevant instruction is executed while the DB Connection Service is running.		Write the user program so that the relevant instruction is executed while the DB Connection Service is running.	
	A command to stop the DB Connection Service was given before the execution of relevant instruction.					
Attached information	Attached information 1: Error Location Attached information 2: Error Location Detail, Rung Number. For a program section, the rung number from the start of the section is given. For ST, the line number is given. Attached information 3: Instruction Name and Instruction Instance Name Where Error Occurred. If there is more than one instruction, all of them are given. If the instruction cannot be identified, nothing is given. Attached information 4: Expansion Error Code (*ErrorIDEx*)					
Precautions/ Remarks	None					

Event name	DB Connection Service Run Mode Change Failed			Event code	5401 3001 hex	
Meaning	Failed to change the Run mode of the DB Connection Service.					
Source	PLC Function Module		Source details	Instruction	Detection timing	At instruction execution
Error attributes	Level	Observation	Recovery	---	Log category	System
Effects	User program	Continues.	Operation	The relevant instruction will end according to specifications.		
System-defined variables	Variable		Data type		Name	
	None		---		---	
Cause and correction	Assumed cause		Correction		Prevention	
	Run mode change to Test Mode was executed by the relevant instruction while running in Operation Mode.		Stop the DB Connection Service, and then execute the relevant instruction. Or, correct the user program so that the relevant instruction is executed when the operation status of the DB Connection Service is *Idle*.		Write the user program so that the relevant instruction is executed when the operation status of the DB Connection Service is *Idle*.	
	Run mode change to Operation Mode was executed by the relevant instruction while running in Test Mode.					
	Start of the DB Connection Service was commanded while the DB Connection Service was being stopped.		Execute the relevant instruction later.		While a DB_Insert, DB_Update, DB_Select, or DB_Delete instruction is being executed, the DB Connection Service becomes stopping status If stop of the DB Connection Service is commanded. Stop the DB Connection Service after completion of the DB_Insert, DB_Update, DB_Select, or DB_Delete instruction.	
	Shutdown of the DB Connection Service was commanded while the DB Connection Service was being stopped.					
Attached information	Attached information 1: Error Location Attached information 2: Error Location Detail, Rung Number. For a program section, the rung number from the start of the section is given. For ST, the line number is given. Attached information 3: Instruction Name and Instruction Instance Name Where Error Occurred. If there is more than one instruction, all of them are given. If the instruction cannot be identified, nothing is given. Attached information 4: Expansion Error Code (*ErrorIDEx*)					
Precautions/ Remarks	None					

Event name	DB Connection Service Shutdown or Shutting Down			Event code	54013002 hex	
Meaning	The DB Connection Service is already shut down or being shut down.					
Source	PLC Function Module		Source details	Instruction	Detection timing	At instruction execution
Error attributes	Level	Observation	Recovery	---	Log category	System
Effects	User program	Continues.	Operation	The relevant instruction will end according to specifications.		
System-defined variables	Variable		Data type		Name	
	None		---		---	
Cause and correction	Assumed cause		Correction		Prevention	
	The relevant instruction was executed after the DB Connection Service was shut down.		Cycle the power supply to the Controller, start the DB Connection Service, and then execute the relevant instruction.		Write the user program so that the relevant instruction is not executed after the execution of DB_Shutdown instruction. Or, write the user program so that the relevant instruction is not executed after shutdown is commanded from Sysmac Studio.	
	The relevant instruction was executed while the shutdown processing of the DB Connection Service was in progress.					
Attached information	Attached information 1: Error Location Attached information 2: Error Location Detail, Rung Number. For a program section, the rung number from the start of the section is given. For ST, the line number is given. Attached information 3: Instruction Name and Instruction Instance Name Where Error Occurred. If there is more than one instruction, all of them are given. If the instruction cannot be identified, nothing is given. Attached information 4: Expansion Error Code (*ErrorIDEx*)					
Precautions/ Remarks	None					

Event name	Invalid DB Connection Name			Event code	54013003 hex	
Meaning	The specified DB Connection Name is not set in any DB Connection settings.					
Source	PLC Function Module		Source details	Instruction	Detection timing	At instruction execution
Error attributes	Level	Observation	Recovery	---	Log category	System
Effects	User program	Continues.	Operation	The relevant instruction will end according to specifications.		
System-defined variables	Variable		Data type		Name	
	None		---		---	
Cause and correction	Assumed cause		Correction		Prevention	
	The DB Connection Name specified in the *DBConnectionName* input variable of the relevant instruction is wrong.		Specify a correct DB Connection Name in the *DBConnectionName* input variable of the relevant instruction.		Confirm that a DB Connection Name is correctly specified in the *DBConnectionName* input variable of the relevant instruction.	
	The DB Connection Name set in the DB Connection settings is wrong.		Specify a correct DB Connection Name in the DB Connection settings.		Confirm that a DB Connection Name is correctly set in the DB Connection Settings.	
Attached information	Attached information 1: Error Location Attached information 2: Error Location Detail, Rung Number. For a program section, the rung number from the start of the section is given. For ST, the line number is given. Attached information 3: Instruction Name and Instruction Instance Name Where Error Occurred. If there is more than one instruction, all of them are given. If the instruction cannot be identified, nothing is given. Attached information 4: Expansion Error Code (*ErrorIDEx*)					
Precautions/ Remarks	None					

Event name	DB Connection Rejected			Event code	5401 3004 hex	
Meaning	The DB rejected the connection.					
Source	PLC Function Module		Source details	Instruction	Detection timing	At instruction execution
Error attributes	Level	Observation	Recovery	---	Log category	System
Effects	User program	Continues.	Operation	The relevant instruction will end according to specifications.		
System-defined variables	Variable		Data type		Name	
	None		---		---	
Cause and correction	Assumed cause		Correction		Prevention	
	The user name or password set in the DB Connection settings is wrong.		Enter the correct user name and password in the DB Connection settings.		Enter the correct user name and password in the DB Connection settings.	
Attached information	Attached information 1: Error Location Attached information 2: Error Location Detail, Rung Number. For a program section, the rung number from the start of the section is given. For ST, the line number is given. Attached information 3: Instruction Name and Instruction Instance Name Where Error Occurred. If there is more than one instruction, all of them are given. If the instruction cannot be identified, nothing is given. Attached information 4: Expansion Error Code (*ErrorIDEx*)					
Precautions/ Remarks	None					

Event name	DB Connection Failed			Event code	5401 3005 hex	
Meaning	Failed to connect to the DB.					
Source	PLC Function Module		Source details	Instruction	Detection timing	At instruction execution
Error attributes	Level	Observation	Recovery	---	Log category	System
Effects	User program	Continues.	Operation	The relevant instruction will end according to specifications.		
System-defined variables	Variable		Data type		Name	
	None		---		---	
Cause and correction	Assumed cause		Correction		Prevention	
	A server does not exist for the specified IP address or the specified host name.		Enter the correct IP address or host name in the DB Connection settings.		Enter the correct IP address or host name in the DB Connection settings.	
	The power supply to the server is OFF.		Check the server status and start it properly.		Check the server status and start it properly.	
	The DB is stopped in the server.					
	The Ethernet cable connector is disconnected.		Reconnect the connector and make sure it is mated correctly.		Connect the connector securely.	
	The Ethernet cable is broken.		Replace the Ethernet cable.		None	
Attached information	Attached information 1: Error Location Attached information 2: Error Location Detail, Rung Number. For a program section, the rung number from the start of the section is given. For ST, the line number is given. Attached information 3: Instruction Name and Instruction Instance Name Where Error Occurred. If there is more than one instruction, all of them are given. If the instruction cannot be identified, nothing is given. Attached information 4: Expansion Error Code (*ErrorIDEx*)					
Precautions/ Remarks	None					

Event name	DB Connection Already Established			Event code	5401 3006 hex	
Meaning	A same-name DB Connection is already established.					
Source	PLC Function Module		Source details	Instruction	Detection timing	At instruction execution
Error attributes	Level	Observation	Recovery	---	Log category	System
Effects	User program	Continues.	Operation	The relevant instruction will end according to specifications.		
System-defined variables	Variable		Data type		Name	
	None		---		---	
Cause and correction	Assumed cause		Correction		Prevention	
	The relevant instruction was executed when a same-name DB Connection was already established.		Correct the user program so that the relevant instruction is executed when the DB Connection is closed.		Write the user program so that the relevant instruction is executed when the DB Connection is closed.	
Attached information	Attached information 1: Error Location Attached information 2: Error Location Detail, Rung Number. For a program section, the rung number from the start of the section is given. For ST, the line number is given. Attached information 3: Instruction Name and Instruction Instance Name Where Error Occurred. If there is more than one instruction, all of them are given. If the instruction cannot be identified, nothing is given. Attached information 4: Expansion Error Code (*ErrorIDEx*)					
Precautions/ Remarks	None					

Event name	Too Many DB Connections			Event code	5401 3007 hex	
Meaning	The number of DB Connections that can be established at the same time is exceeded.					
Source	PLC Function Module		Source details	Instruction	Detection timing	At instruction execution
Error attributes	Level	Observation	Recovery	---	Log category	System
Effects	User program	Continues.	Operation	The relevant instruction will end according to specifications.		
System-defined variables	Variable		Data type		Name	
	None		---		---	
Cause and correction	Assumed cause		Correction		Prevention	
	The relevant instruction was executed when the maximum number of DB Connections that can be established at the same time were already established.		Correct the user program so that the number of established DB Connections does not exceed the maximum number of DB Connections that can be established at the same time.		Write the user program so that the number of established DB Connections does not exceed the maximum number of DB Connections that can be established at the same time.	
Attached information	Attached information 1: Error Location Attached information 2: Error Location Detail, Rung Number. For a program section, the rung number from the start of the section is given. For ST, the line number is given. Attached information 3: Instruction Name and Instruction Instance Name Where Error Occurred. If there is more than one instruction, all of them are given. If the instruction cannot be identified, nothing is given. Attached information 4: Expansion Error Code (*ErrorIDEx*)					
Precautions/ Remarks	None					

Event name	Invalid DB Connection			Event code	54013008 hex	
Meaning	The specified DB Connection is not correct, or the DB Connection is already closed.					
Source	PLC Function Module		Source details	Instruction	Detection timing	At instruction execution
Error attributes	Level	Observation	Recovery	---	Log category	System
Effects	User program	Continues.	Operation	The relevant instruction will end according to specifications.		
System-defined variables	Variable		Data type		Name	
	None		---		---	
Cause and correction	Assumed cause		Correction		Prevention	
	The DB Connection specified in the *DBConnection* input variable of the relevant instruction is wrong.		Specify a correct DB Connection in the *DBConnection* input variable of the relevant instruction.		Confirm that a correct DB Connection is specified in the *DBConnection* input variable of the relevant instruction.	
	The DB Connection specified in the *DBConnection* input variable of the relevant instruction is closed.		Correct the user program so that the relevant instruction is executed after the DB Connection is established by a DB_Connect instruction.		Write the user program so that the relevant instruction is executed after the DB Connection is established by a DB_Connect instruction.	
Attached information	Attached information 1: Error Location Attached information 2: Error Location Detail, Rung Number. For a program section, the rung number from the start of the section is given. For ST, the line number is given. Attached information 3: Instruction Name and Instruction Instance Name Where Error Occurred. If there is more than one instruction, all of them are given. If the instruction cannot be identified, nothing is given. Attached information 4: Expansion Error Code (*ErrorIDEx*)					
Precautions/ Remarks	None					

Event name	Invalid DB Map Variable			Event code	54013009 hex	
Meaning	The specified DB Map Variable is not correct.					
Source	PLC Function Module		Source details	Instruction	Detection timing	At instruction execution
Error attributes	Level	Observation	Recovery	---	Log category	System
Effects	User program	Continues.	Operation	The relevant instruction will end according to specifications.		
System-defined variables	Variable		Data type		Name	
	None		---		---	
Cause and correction	Assumed cause		Correction		Prevention	
	A structure variable that contains a derivative data type of member was specified as a DB Map Variable.		Specify a basic data type for the members of the structure data used in the DB Map Variable.		Confirm the data type of the variables to be specified as a DB Map Variable when writing the user program.	
	A non-structure variable was specified as a DB Map Variable.		Specify a structure variable for the DB Map Variable.			
	A structure array variable was specified as a DB Map Variable for INSERT or UPDATE.		Specify a structure variable for the DB Map Variable for INSERT or UPDATE.			
Attached information	Attached information 1: Error Location Attached information 2: Error Location Detail, Rung Number. For a program section, the rung number from the start of the section is given. For ST, the line number is given. Attached information 3: Instruction Name and Instruction Instance Name Where Error Occurred. If there is more than one instruction, all of them are given. If the instruction cannot be identified, nothing is given. Attached information 4: Expansion Error Code (*ErrorIDEx*)					
Precautions/ Remarks	None					

Event name	Unregistered DB Map Variable			Event code	5401300A hex	
Meaning	The specified DB Map Variable has not been registered.					
Source	PLC Function Module		Source details	Instruction	Detection timing	At instruction execution
Error attributes	Level	Observation	Recovery	---	Log category	System
Effects	User program	Continues.	Operation	The relevant instruction will end according to specifications.		
System-defined variables	Variable		Data type		Name	
	None		---		---	
Cause and correction	Assumed cause		Correction		Prevention	
	The DB Map Variable has not been created by a DB_CreateMapping instruction.		Correct the user program so that the relevant instruction is executed after the DB Map Variable is created by a DB_CreateMapping instruction.		Write the user program so that the relevant instruction is executed after the DB Map Variable is created by a DB_CreateMapping instruction.	
	A variable that is not registered as a DB Map Variable was specified in *MapVar*.		Check the input parameters of the relevant instruction and correct the user program.		In the input parameters of the relevant instruction, specify the DB Connection specified in the DB_CreateMapping instruction and the DB Map Variable created by the DB_CreateMapping instruction.	
	The DB Connection specified in the relevant instruction is different from the one specified at the execution of DB_CreateMapping instruction.					
Attached information	Attached information 1: Error Location Attached information 2: Error Location Detail, Rung Number. For a program section, the rung number from the start of the section is given. For ST, the line number is given. Attached information 3: Instruction Name and Instruction Instance Name Where Error Occurred. If there is more than one instruction, all of them are given. If the instruction cannot be identified, nothing is given. Attached information 4: Expansion Error Code (*ErrorIDEx*)					
Precautions/ Remarks	None					

Event name	SQL Execution Error			Event code	5401300B hex	
Meaning	The executed SQL statement resulted in an error.					
Source	PLC Function Module		Source details	Instruction	Detection timing	At instruction execution
Error attributes	Level	Observation	Recovery	---	Log category	System
Effects	User program	Continues.	Operation	The relevant instruction will end according to specifications.		
System-defined variables	Variable		Data type		Name	
	None		---		---	
Cause and correction	Assumed cause		Correction		Prevention	
	There is no column with the same name as a structure member of the DB Map Variable.		Check whether the column names match the structure member names of the DB Map Variable.		Confirm that the column names match the structure member names of the DB Map Variable.	
	The table specified in the DB_CreateMapping instruction does not exist in the DB.		Check whether the table name specified in the DB_CreateMapping instruction is correct.		Confirm that the table name specified in the DB_CreateMapping instruction is correct.	
	One or more structure member values of the DB Map Variable cannot be converted to the corresponding column's data type.		Check whether the data types of the structure members of the DB Map Variable can be converted to the corresponding column's data type.		Confirm that the data types of the structure members of the DB Map Variable can be converted to the corresponding column's data type.	
	One or more column values cannot be converted to the corresponding structure member's data type of the DB Map Variable.		Check whether the data types of the columns can be converted to the corresponding structure member's data type of the DB Map Variable. Or, confirm that the values of the mapped columns are not NULL.		Confirm that the data types of the columns can be converted to the corresponding structure member's data type of the DB Map Variable. Or, define the structure members so as not to map a column whose value can be NULL.	
	One or more structure member values of the DB Map Variable exceed the valid range of the corresponding column's data type.		Check the structure member values of the DB Map Variable.		Write the user program so that the structure member values of the DB Map Variable are within the valid range of the corresponding column's data type.	
	The column specified in the extraction condition does not exist in the DB's records. (DB_Select instruction, DB_Update instruction, DB_Delete instruction)		Check whether the column name specified in the extraction condition is correct. Or, check whether the syntax of the extraction condition is correct.		Confirm that the column name specified in the extraction condition is correct. Or, confirm that the syntax of the extraction condition is correct.	
	The extraction condition has a syntax error. (DB_Select instruction, DB_Update instruction, DB_Delete instruction)					
	The column specified in the sort condition does not exist in the DB's records. (DB_Select instruction)		Check whether the column name specified in the sort condition is correct. Or, check whether the syntax of the sort condition is correct.		Confirm that the column name specified in the sort condition is correct. Or, confirm that the syntax of the sort condition is correct.	
	The sort condition has a syntax error. (DB_Select instruction)					
	The user does not have the access rights to the table.		Check the access rights to the table.		Confirm the access rights to the table.	

Attached information	Attached information 1: Error Location Attached information 2: Error Location Detail, Rung Number. For a program section, the rung number from the start of the section is given. For ST, the line number is given. Attached information 3: Instruction Name and Instruction Instance Name Where Error Occurred. If there is more than one instruction, all of them are given. If the instruction cannot be identified, nothing is given. Attached information 4: Expansion Error Code (*ErrorIDEx*)
Precautions/ Remarks	None

Event name	Spool Capacity Exceeded			Event code	5401300C hex	
Meaning	The SQL statement could not be stored in the Spool memory because its maximum capacity was exceeded.					
Source	PLC Function Module		Source details	Instruction	Detection timing	At instruction execution
Error attributes	Level	Observation	Recovery	---	Log category	System
Effects	User program	Continues.	Operation	The relevant instruction will end according to specifications.		
System-defined variables	Variable		Data type		Name	
	None		---		---	
Cause and correction	Assumed cause		Correction		Prevention	
	The DB connection failure has been continuing due to network failure or other factors.		Recover from the network failure.		Control from the user program like below. Check the Spool memory usage using a DB_GetConnectionStatus instruction, and when the Spool memory usage has exceeded a certain value, do not execute the DB_Insert nor DB_Update instructions. Or, check the DB Connection status using a DB_GetConnectionStatus instruction, and when the status has changed to *Connected*, resend the SQL statements stored in the Spool memory using a DB_ControlSpool instruction.	
	The resend processing of the SQL statements stored in the Spool memory has not been executed (when the *Resend spool data* parameter is set to *Manual*).		Resend the SQL statements stored in the Spool memory using a DB_ControlSpool instruction after establishing the DB Connection again.		Check the DB Connection status using a DB_GetConnectionStatus instruction, and when the status has changed to *Connected*, resend the SQL statements stored in the Spool memory using a DB_ControlSpool instruction.	
Attached information	Attached information 1: Error Location Attached information 2: Error Location Detail, Rung Number. For a program section, the rung number from the start of the section is given. For ST, the line number is given. Attached information 3: Instruction Name and Instruction Instance Name Where Error Occurred. If there is more than one instruction, all of them are given. If the instruction cannot be identified, nothing is given. Attached information 4: Expansion Error Code (*ErrorIDEx*)					
Precautions/ Remarks	None					

Event name	Invalid Extraction Condition		Event code	5401300E hex	
Meaning	The entered extraction condition is invalid.				
Source	PLC Function Module	Source details	Instruction	Detection timing	At instruction execution
Error attributes	Level: Observation	Recovery	---	Log category	System
Effects	User program: Continues.	Operation	The relevant instruction will end according to specifications.		
System-defined variables	Variable	Data type		Name	
	None	---		---	
Cause and correction	Assumed cause	Correction		Prevention	
	A text string that consists of a NULL (16#00) character only was specified in the *Where* input variable.	Enter a text string that specifies the extraction condition in the *Where* input variable.		Enter a text string that specifies the extraction condition in the *Where* input variable.	
Attached information	Attached information 1: Error Location Attached information 2: Error Location Detail, Rung Number. For a program section, the rung number from the start of the section is given. For ST, the line number is given. Attached information 3: Instruction Name and Instruction Instance Name Where Error Occurred. If there is more than one instruction, all of them are given. If the instruction cannot be identified, nothing is given. Attached information 4: Expansion Error Code (*ErrorIDEx*)				
Precautions/ Remarks	None				

Event name	Log Code Out of Range			Event code	5401 3010 hex	
Meaning	The value of the entered log code is outside the valid range.					
Source	PLC Function Module		Source details	Instruction	Detection timing	At instruction execution
Error attributes	Level	Observation	Recovery	---	Log category	System
Effects	User program	Continues.	Operation	The relevant instruction will end according to specifications.		
System-defined variables	Variable		Data type		Name	
	None		---		---	
Cause and correction	Assumed cause		Correction		Prevention	
	A value outside the valid range from 0 to 9999 was specified.		Correct the user program so that the log code is within the valid range from 0 to 9999.		Write the user program so that the log code is within the valid range from 0 to 9999.	
Attached information	Attached information 1: Error Location Attached information 2: Error Location Detail, Rung Number. For a program section, the rung number from the start of the section is given. For ST, the line number is given. Attached information 3: Instruction Name and Instruction Instance Name Where Error Occurred. If there is more than one instruction, all of them are given. If the instruction cannot be identified, nothing is given. Attached information 4: Expansion Error Code (*ErrorIDEx*)					
Precautions/ Remarks	None					

Event name	DB Connection Disconnected Error Status			Event code	5401 3011 hex	
Meaning	The instruction could not be executed because the DB Connection had been disconnected due to an error.					
Source	PLC Function Module		Source details	Instruction	Detection timing	At instruction execution
Error attributes	Level	Observation	Recovery	---	Log category	System
Effects	User program	Continues.	Operation	The relevant instruction will end according to specifications.		
System-defined variables	Variable		Data type		Name	
	None		---		---	
Cause and correction	Assumed cause		Correction		Prevention	
	The power supply to the server is OFF.		Check the server status and start it properly.		Check the server status and start it properly.	
	The DB is stopped in the server.					
	The Ethernet cable connector is disconnected.		Reconnect the connector and make sure it is mated correctly.		Connect the connector securely.	
	The Ethernet cable is broken.		Replace the Ethernet cable.		None	
	Noise		Implement noise countermeasures if there is excessive noise.		Implement noise countermeasures if there is excessive noise.	
Attached information	Attached information 1: Error Location Attached information 2: Error Location Detail, Rung Number. For a program section, the rung number from the start of the section is given. For ST, the line number is given. Attached information 3: Instruction Name and Instruction Instance Name Where Error Occurred. If there is more than one instruction, all of them are given. If the instruction cannot be identified, nothing is given. Attached information 4: Expansion Error Code (*ErrorIDEx*)					
Precautions/ Remarks	None					

Event name	DB Connection Instruction Execution Timeout			Event code	54013012 hex	
Meaning	The instruction was not completed within the time specified for timeout.					
Source	PLC Function Module		Source details	Instruction	Detection timing	At instruction execution
Error attributes	Level	Observation	Recovery	---	Log category	System
Effects	User program	Continues.	Operation	The relevant instruction will end according to specifications.		
System-defined variables	Variable		Data type		Name	
	None		---		---	
Cause and correction	Assumed cause		Correction		Prevention	
	The power supply to the server is OFF.		Check the server status and start it properly.		Check the server status and start it properly.	
	The Ethernet cable connector is disconnected.		Reconnect the connector and make sure it is mated correctly.		Connect the connector securely.	
	The Ethernet cable is broken.		Replace the Ethernet cable.		None	
	The server's processing time is long.		Check the server's response time in the Debug Log and change the timeout parameter to an appropriate value.		Check the server's response time in the Debug Log and specify an appropriate value in the timeout parameter.	
Attached information	Attached information 1: Error Location Attached information 2: Error Location Detail, Rung Number. For a program section, the rung number from the start of the section is given. For ST, the line number is given. Attached information 3: Instruction Name and Instruction Instance Name Where Error Occurred. If there is more than one instruction, all of them are given. If the instruction cannot be identified, nothing is given. Attached information 4: Expansion Error Code (*ErrorIDEx*)					
Precautions/ Remarks	None					

Event name	DB Connection Service Error Stop			Event code	54013013 hex	
Meaning	The instruction could not be executed because the DB Connection Service was stopped due to an error.					
Source	PLC Function Module		Source details	Instruction	Detection timing	At instruction execution
Error attributes	Level	Observation	Recovery	---	Log category	System
Effects	User program	Continues.	Operation	The relevant instruction will end according to specifications.		
System-defined variables	Variable		Data type		Name	
	None		---		---	
Cause and correction	Assumed cause		Correction		Prevention	
	The DB Connection settings are corrupted.		Transfer the DB Connection settings again using the synchronization function of Sysmac Studio.		Do not interrupt the power supply to the Controller during a download of the DB Connection settings.	
Attached information	Attached information 1: Error Location Attached information 2: Error Location Detail, Rung Number. For a program section, the rung number from the start of the section is given. For ST, the line number is given. Attached information 3: Instruction Name and Instruction Instance Name Where Error Occurred. If there is more than one instruction, all of them are given. If the instruction cannot be identified, nothing is given. Attached information 4: Expansion Error Code (*ErrorIDEx*)					
Precautions/ Remarks	None					

Event name	Data Already Spooled			Event code	5401 3014 hex	
Meaning	One or more SQL statements are already stored in the Spool memory.					
Source	PLC Function Module		Source details	Instruction	Detection timing	At instruction execution
Error attributes	Level	Observation	Recovery	---	Log category	System
Effects	User program	Continues.	Operation	The relevant instruction will end according to specifications.		
System-defined variables	Variable		Data type		Name	
	None		---		---	
Cause and correction	Assumed cause		Correction		Prevention	
	A DB_Insert or DB_Update instruction was executed when one or more SQL statements were already stored in the Spool memory.		None		None	
	A DB_Select or DB_Delete instruction was executed when one or more SQL statements were already stored in the Spool memory.		Execute the instruction again after the resend processing of the SQL statements stored in the Spool memory is completed.		Execute the relevant instruction when no SQL statements are stored in the Spool memory.	
Attached information	Attached information 1: Error Location Attached information 2: Error Location Detail, Rung Number. For a program section, the rung number from the start of the section is given. For ST, the line number is given. Attached information 3: Instruction Name and Instruction Instance Name Where Error Occurred. If there is more than one instruction, all of them are given. If the instruction cannot be identified, nothing is given. Attached information 4: Expansion Error Code (*ErrorIDEx*)					
Precautions/ Remarks	None					

Event name	DB Connection Service Initializing			Event code	5401 3015 hex	
Meaning	The instruction could not be executed because the initialization processing of the DB Connection Service is in progress.					
Source	PLC Function Module		Source details	Instruction	Detection timing	At instruction execution
Error attributes	Level	Observation	Recovery	---	Log category	System
Effects	User program	Continues.	Operation	The relevant instruction will end according to specifications.		
System-defined variables	Variable		Data type		Name	
	_DBC_Status		_sDBC_STATUS		DB Connection Service Status	
Cause and correction	Assumed cause		Correction		Prevention	
	The relevant instruction was executed during the initialization processing of the DB Connection Service.		Execute the relevant instruction after the operation status of the DB Connection Service changes to *Running* or *Idle*.		Execute the relevant instruction after confirming the operation status of the DB Connection Service with the *_DBC_Status* system-defined variable.	
Attached information	Attached information 1: Error Location Attached information 2: Error Location Detail, Rung Number. For a program section, the rung number from the start of the section is given. For ST, the line number is given. Attached information 3: Instruction Name and Instruction Instance Name Where Error Occurred. If there is more than one instruction, all of them are given. If the instruction cannot be identified, nothing is given. Attached information 4: Expansion Error Code (*ErrorIDEx*)					
Precautions/ Remarks	None					

Event name	DB in Process			Event code	54013016 hex	
Meaning	The instruction could not be executed because the DB is under processing in the server.					
Source	PLC Function Module		Source details	Instruction	Detection timing	At instruction execution
Error attributes	Level	Observation	Recovery	---	Log category	System
Effects	User program	Continues.	Operation	The relevant instruction will end according to specifications.		
System-defined variables	Variable		Data type		Name	
	None		---		---	
Cause and correction	Assumed cause		Correction		Prevention	
	Though a DB Connection Instruction Execution Timeout occurred for the previous instruction, the relevant instruction was executed before completion of the DB's processing in the server.		Re-execute the relevant instruction from the user program. However, if you execute a DB_Insert or DB_Update instruction and the spool function is enabled, you do not have to re-execute the relevant instruction because the SQL statement will be stored in the Spool memory.		Estimate the processing time of the DB in the server and adjust the execution timing of the DB Connection Instruction to an appropriate frequency.	
Attached information	Attached information 1: Error Location Attached information 2: Error Location Detail, Rung Number. For a program section, the rung number from the start of the section is given. For ST, the line number is given. Attached information 3: Instruction Name and Instruction Instance Name Where Error Occurred. If there is more than one instruction, all of them are given. If the instruction cannot be identified, nothing is given. Attached information 4: Expansion Error Code (*ErrorIDEx*)					
Precautions/ Remarks	None					

Event name	Operation Log Disabled			Event code	54013017 hex	
Meaning	The log could not be recorded because the specified Operation Log is disabled.					
Source	PLC Function Module		Source details	Instruction	Detection timing	At instruction execution
Error attributes	Level	Observation	Recovery	---	Log category	System
Effects	User program	Continues.	Operation	The relevant instruction will end according to specifications.		
System-defined variables	Variable		Data type		Name	
	None		---		---	
Cause and correction	Assumed cause		Correction		Prevention	
	Though Execution Log was specified in the *LogType* input variable, the Execution Log is disabled.		Enable the Execution Log in the DB Connection Service settings.		Execute the instruction when the Execution Log is enabled.	
	Though Debug Log was specified in the *LogType* input variable, recording to the Debug Log is stopped.		Start recording to the Debug Log using a DB_ControlService instruction. Or, start recording to the Debug Log from Sysmac Studio.		Execute the instruction after the recording to the Debug Log is started.	
Attached information	Attached information 1: Error Location Attached information 2: Error Location Detail, Rung Number. For a program section, the rung number from the start of the section is given. For ST, the line number is given. Attached information 3: Instruction Name and Instruction Instance Name Where Error Occurred. If there is more than one instruction, all of them are given. If the instruction cannot be identified, nothing is given. Attached information 4: Expansion Error Code (*ErrorIDEx*)					
Precautions/ Remarks	None					

A

Appendix A DB Connection Instructions

A-1 DB Connection Instructions and Variables

A-1-1 DB Connection Instruction Set

This section gives a list of DB Connection Instructions.

Instruction	Name	Page
DB_Connect	Establish DB Connection	A-6
DB_Close	Close DB Connection	A-9
DB_CreateMapping	Create DB Map	A-11
DB_Insert	Insert DB Record	A-14
DB_Update	Update DB Record	A-18
DB_Select	Retrieve DB Record	A-34
DB_Delete	Delete DB Record	A-39
DB_ControlService	Control DB Connection Service	A-54
DB_GetServiceStatus	Get DB Connection Service Status	A-60
DB_GetConnectionStatus	Get DB Connection Status	A-65
DB_ControlSpool	Resend/Clear Spool Data	A-71
DB_PutLog	Record Operation Log	A-78
DB_Shutdown	Shutdown DB Connection Service	A-84

A-1-2 Variables Used in the DB Connection Instructions

This section describes the details of the variables used in the DB Connection Instructions.

Common Input and Output Variables Used in the DB Connection Instructions

● DBConnection

Input variable	Meaning	Data type	Description
DBConnection	DB Connection	DWORD	DB Connection output from a DB_Connect instruction. The instructions are executed for a specified DB Connection.

● ServiceStatus

Output variable		Meaning	Data type	Description
	Member			
ServiceStatus		DB Connection Service Status	_sDBC_SERVICE_STATUS	Structure to show the status of the DB Connection Service.
	Status	Service Status	_eDBC_STATUS	Enumeration data type to show the service status _DBC_STATUS_IDLE(0): Idle _DBC_STATUS_RUNNING(1): Running in Operation Mode _DBC_STATUS_TEST(2): Running in Test Mode
	DebugLog	Debug Log Flag	BOOL	TRUE while the Debug Log is recorded. FALSE while recording to the Debug Log is stopped.
	OperatingTime	Operating Time	TIME	Time elapsed since the service was started.
	ExecCnt	Number of Normal Executions	DINT	Total number of times in all connections when an SQL statement was normally executed.
	FailedCnt	Number of Error Executions	DINT	Total number of times in all connections when an SQL statement execution failed.
	SpoolDataCnt	Number of Spool Data	DINT	Number of SQL statements stored in the Spool memory in all connections.

● ConnectionStatus

Output variable		Meaning	Data type	Description
	Member			
ConnectionStatus		DB Connection Status	_sDBC_CONNECTION _STATUS	Structure to show the status of a DB Connection.
	Status	Connection Status	_eDBC_CONNECTION _STATUS	Enumeration data type to show the status of a DB Connection _DBC_CONNECTION_STATUS_CLOSED(0): Closed _DBC_CONNECTION_STATUS_CONNECTED(1): Connected _DBC_CONNECTION_STATUS_DISCONNECTED(2): Disconnected (Disconnected due to a network failure while the DB is connected.)
	ConnectedTime	Connected Time	TIME	Total time when the DB is connected.
	DisconnectedTime	Disconnected Time	TIME	Total time when the DB is disconnected due to an error.
	ExecCnt	Number of Normal Executions	DINT	Number of times when an SQL statement was executed normally in the DB Connection.

Output variable		Meaning	Data type	Description
	Member			
	FailedCnt	Number of Error Executions	DINT	Number of times when an SQL statement execution failed in the DB Connection.
	DBRespTime	DB Response Time	TIME	Time since an SQL statement is sent from the CPU Unit until the SQL execution result is returned from the CPU Unit when an SQL statement is executed. This is stored only when a normal response is returned from the DB. If an instruction execution timeout occurred, the DB Response Time is not stored when the instruction execution is completed (i.e. when the *Error* output variable changes from FALSE to TRUE). (The previous DB Response Time is held.) The new DB Response Time is stored when a normal response is returned from the DB after the instruction execution timeout.
	SpoolDataCnt	Number of Spool Data	INT	Number of SQL statements stored in the Spool memory for the DB Connection.
	SpoolUsageRate	Spool Usage	SINT	Use rate of the Spool memory for the DB Connection. The unit is percentage (%).
	ErrorDateTime	Disconnection Date/Time	DATE_AND_TIME	Date and time the last time the connection was disconnected due to an error.
	SQLSTATE	SQL Status	STRING(8)	Error code*2 defined in SQL Standards (ISO/IEC 9075) for disconnection*1
	ErrorCode	Error Code	DINT	Error code*2 for disconnection*1, which is specific to DB vendor
	ErrorMsg	Error Message	STRING(128)	Error message*2 for disconnection*1, which is specific to DB vendor

*1 When a network failure or an SQL Execution Error occurred
*2 The value may differ by unit version of the CPU Unit.
The value of connection error to SQL Server was changed in the unit version 1.08 of the CPU Units.

● SendStatus

Output variable	Meaning	Data type	Description
SendStatus	Send Status	_eDBC_SEND_STATUS	Enumeration data type that shows transmission status of the SQL statement to DB _DBC_SEND_INIT(0): Initial status _DBC_SEND_UNSENT(1): SQL statement unsent _DBC_SEND_SENDING(2): Sending SQL statement _DBC_SEND_SPOOLED(3): SQL statement spooled _DBC_SEND_COMPLETE(4): SQL statement transmission completed

Common Variables Used in NJ-series Instructions

Input variable	Meaning	Data type	Description
Execute	Execute	BOOL	The instruction is executed when *Execute* changes to TRUE.

Output variable	Meaning	Data type	Description
Done	Done	BOOL	Shows whether the instruction is normally completed. TRUE: Normally completed FALSE: ・ Terminated due to an error ・ Being executed ・ Execution conditions not satisfied
Busy	Executing	BOOL	Shows whether the instruction is being executed. TRUE: Being executed FALSE: Not being executed
Error	Error	BOOL	Shows whether the instruction is terminated due to an error. TRUE: Terminated due to an error FALSE: ・Normally ended ・Being executed ・Execution conditions not satisfied
ErrorID	Error Code	WORD	Contains the error code when the instruction is terminated due to an error. WORD#16#0 indicates normal execution.

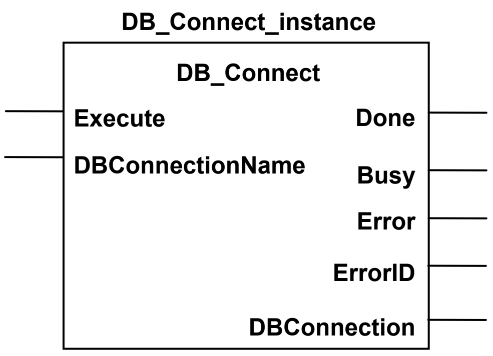
System-defined Variables Related to DB Connection Service

Name	Meaning	Data type	Description
_DBC_Status	DB Connection Service Status	_sDBC_ STATUS	System-defined variable that shows the status of the DB Connection Service.

Refer to *3-5-4 System-defined Variables* for details of the system-defined variables.

DB_Connect (Establish DB Connection)

The DB_Connect instruction connects to a specified DB.

Instruction	Name	FB/FUN	Graphic expression	ST expression
DB_Connect	Establish DB Connection	FB	 DB_Connect_instance DB_Connect Execute DBConnectionName Done Busy Error ErrorID DBConnection	DB_Connect_instance(Execute, DBConnectionName, Done, Busy, Error, ErrorID, DBConnection);

Note The *DB_Connect_instance* is an instance of DB_Connect instruction, which is declared as a variable.

Variables

Input Variables

Name	Meaning	Data type	Valid range	Unit	Default	Description
Execute	Execute	BOOL	TRUE or FALSE	---	FALSE	Specify the execution condition.
DBConnectionName	DB Connection Name	STRING	17 bytes max. (including the final NULL character)	---	‘’	Specify a DB Connection name set on Sysmac Studio.

Output Variables

Name	Meaning	Data type	Valid range	Unit	Description
Done	Done	BOOL	TRUE or FALSE	---	TRUE when the instruction is normally completed.
Busy	Executing	BOOL	TRUE or FALSE	---	TRUE when the instruction is being executed.
Error	Error	BOOL	TRUE or FALSE	---	TRUE when the instruction is terminated due to an error.
ErrorID	Error Code	WORD	16#0000 to 16#FFFF	---	Contains the error code when an error occurs.
DBConnection	DB Connection	DWORD	16#00000000 to 16#FFFFFFFF	---	Outputs a DB Connection. Specify this DB Connection in DB_CreateMapping, DB_Insert, DB_Update, DB_Select, DB_Delete, and DB_Close instructions.

Related System-defined Variables

Name	Meaning	Data type	Description
_EIP_EtnOnlineSta	Online	BOOL	Status of the communications function of the built-in EtherNet/IP port. TRUE: Can be used. FALSE: Cannot be used.

Related Error Codes

Error code	Meaning	Description
0406 hex	Illegal Data Position Specified	The *DBConnectionName* input variable is a text string consisting of NULL characters (16#00) only.
0410 hex	Text String Format Error	・ A space character is included in the text string specified for the *DBConnectionName* input variable. ・ The *DBConnectionName* input variable does not end in NULL.
041D hex	Too Many Instructions Executed at the Same Time	More than 32 DB Connection Instructions were executed at the same time.
3000 hex	DB Connection Service not Started	The instruction was executed when the DB Connection Service was not running.
3002 hex	DB Connection Service Shutdown or Shutting Down	The instruction was executed after the DB Connection Service was shut down or while the DB Connection Service was being shut down.
3003 hex	Invalid DB Connection Name	The DB Connection name specified in the *DBConnectionName* input variable is not set in any DB Connection Settings.
3004 hex	DB Connection Rejected	The DB set in the DB Connection Settings rejected the connection.
3005 hex	DB Connection Failed	・ The DB Connection Service cannot communicate with the DB due to a network failure or other factors. ・ The address set in the DB Connection Settings is wrong.
3006 hex	DB Connection Already Established	A same-name DB Connection is already established.
3007 hex	Too Many DB Connections	The number of DB Connections that can be established at the same time is exceeded.
3008 hex	Invalid DB Connection	The instruction was executed for the same connection at the same time.
3013 hex	DB Connection Service Error Stop	The instruction was executed while the DB Connection Service was stopped due to an error.
3015 hex	DB Connection Service Initializing	The instruction was executed while the initialization processing of the DB Connection Service was in progress.

Function

This instruction is used to connect to the DB specified in the *DBConnectionName* input variable.

The DB Connection name is set in the DB Connection Settings on Sysmac Studio.

When this instruction is normally completed (i.e. when the *Done* output variable changes to TRUE), a DB Connection is established and a value is output to the *DBConnection* output variable. This value is used to specify a DB Connection in some instructions described below.

Precautions for Correct Use

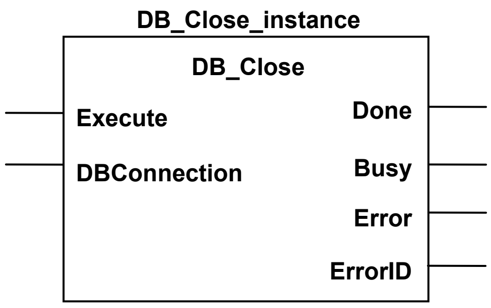
- Execution of this instruction is continued until processing is completed even if the value of *Execute* changes to FALSE or the execution time exceeds the task period. The value of *Done* changes to TRUE when processing is completed. Use this to confirm normal completion of processing.
- Refer to *Using this Section* of the *NJ/NX-series Instructions Reference Manual* (Cat. No. W502) for a timing chart for *Execute, Done, Busy*, and *Error*.
- This instruction cannot be used on an event task. A compiling error will occur.
- This instruction can be used only for the built-in EtherNet/IP port of an NJ-series CPU Unit. It is impossible to connect to a DB via an EtherNet/IP Unit connected to an NJ-series CPU Unit.
- The DB Connection created by this instruction is closed in the following cases.
  - When a DB_Close or DB_Shutdown instruction is executed.
  - When the operating mode of the Controller is changed from RUN mode to PROGRAM mode.
  - When the DB Connection Service is stopped.
- The number of DB Connections that can be established at the same time is up to three for NJ501-□□20 and only one for NJ101-□□20.
- When the DB Connection Service was started in Test Mode, this instruction is completed normally without connecting to the DB actually.
- When a same-name DB Connection is already established, the already-established DB Connection is output to the *DBConnection* output variable.
- An error occurs for this instruction in the following cases. *Error* will be TRUE.
  - When the instruction was executed when the DB Connection Service was not running
  - When the instruction was executed while the initialization processing of the DB Connection Service was in progress
  - When the instruction was executed while the DB Connection Service was stopped due to an error
  - When the instruction was executed after the DB Connection Service was shut down or while the DB Connection Service was being shut down
  - When the DB Connection name specified in the *DBConnectionName* input variable is not set in any DB Connection Settings
  - When the *DBConnectionName* input variable is a text string consisting of NULL characters (16#00) only
  - When a space character is included in the text string specified for the *DBConnectionName* input variable
  - When the *DBConnectionName* input variable does not end in NULL
  - When the connection could not be established because the address set in the DB Connection Settings was wrong
  - When the DB set in the DB Connection Settings rejected the connection
  - When the DB Connection Service cannot communicate with the DB due to a network failure or other causes
  - When the instruction was executed for the same connection at the same time
  - When a same-name DB Connection is already established
  - When the maximum number of connections that can be established at the same time is exceeded
  - When more than 32 DB Connection Instructions were executed at the same time

Sample Programming

Refer to the sample programming that is provided for the DB_Update instruction.

DB_Close (Close DB Connection)

The DB_Close instruction closes the connection with the DB established by a DB_Connect (Establish DB Connection) instruction.

Instruction	Name	FB/FUN	Graphic expression	ST expression
DB_Close	Close DB Connection	FB	 DB_Close_instance DB_Close Execute DBConnection Done Busy Error ErrorID	DB_Close_instance (Execute, DBConnection, Done, Busy, Error, ErrorID);

Note The *DB_Close_instance* is an instance of DB_Close instruction, which is declared as a variable.

Variables

Input Variables

Name	Meaning	Data type	Valid range	Unit	Default	Description
Execute	Execute	BOOL	TRUE or FALSE	---	FALSE	Specify the execution condition.
DBConnection	DB Connection	DWORD	16#00000000 to 16#FFFFFFFF	---	---	Specify the DB Connection established by a DB_Connect instruction.

Output Variables

Name	Meaning	Data type	Valid range	Unit	Description
Done	Done	BOOL	TRUE or FALSE	---	TRUE when the instruction is normally completed.
Busy	Executing	BOOL	TRUE or FALSE	---	TRUE when the instruction is being executed.
Error	Error	BOOL	TRUE or FALSE	---	TRUE when the instruction is terminated due to an error.
ErrorID	Error Code	WORD	16#0000 to 16#FFFF	---	Contains the error code when an error occurs.

Related System-defined Variables

Name	Meaning	Data type	Description
_EIP_EtnOnlineSta	Online	BOOL	Status of the communications function of the built-in EtherNet/IP port. TRUE: Can be used. FALSE: Cannot be used.

Related Error Codes

Error code	Meaning	Description
041D hex	Too Many Instructions Executed at the Same Time	More than 32 DB Connection Instructions were executed at the same time.
3000 hex	DB Connection Service not Started	The instruction was executed when the DB Connection Service was not running.
3002 hex	DB Connection Service Shutdown or Shutting Down	The instruction was executed after the DB Connection Service was shut down or while the DB Connection Service was being shut down.
3008 hex	Invalid DB Connection	The value of the *DBConnection* input variable is invalid or the specified DB Connection is already closed.
3013 hex	DB Connection Service Error Stop	The instruction was executed while the DB Connection Service was stopped due to an error.
3015 hex	DB Connection Service Initializing	The instruction was executed while the initialization processing of the DB Connection Service was in progress.

Function

This instruction is used to close the DB Connection specified in the *DBConnection* input variable.

Precautions for Correct Use

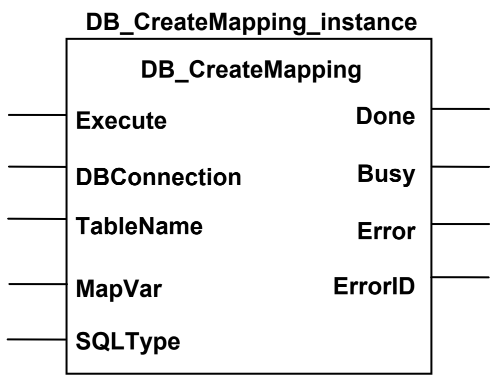
- Execution of this instruction is continued until processing is completed even if the value of *Execute* changes to FALSE or the execution time exceeds the task period. The value of *Done* changes to TRUE when processing is completed. Use this to confirm normal completion of processing.
- Refer to *Using this Section* of the *NJ/NX-series Instructions Reference Manual* (Cat. No. W502) for a timing chart for *Execute, Done, Busy*, and *Error*.
- This instruction cannot be used on an event task. A compiling error will occur.
- When the DB Connection Service was started in Test Mode, this instruction is completed normally without connecting to the DB actually.
- An error occurs for this instruction in the following cases. *Error* will be TRUE.
  - When the instruction was executed when the DB Connection Service was not running
  - When the instruction was executed while the initialization processing of the DB Connection Service was in progress
  - When the instruction was executed while the DB Connection Service was stopped due to an error
  - When the instruction was executed after the DB Connection Service was shut down or while the DB Connection Service was being shut down
  - When the value of the *DBConnection* input variable is invalid or the specified DB Connection is already closed
  - When more than 32 DB Connection Instructions were executed at the same time

Sample Programming

Refer to the sample programming that is provided for the DB_Update instruction.

DB_CreateMapping (Create DB Map)

The DB_CreateMapping instruction creates a mapping from a DB Map Variable to a table of a DB.

Instruction	Name	FB/FUN	Graphic expression	ST expression
DB_CreateMapping	Create DB Map	FB	 DB_CreateMapping_instance DB_CreateMapping Execute DBConnection TableName MapVar SQLType Done Busy Error ErrorID	DB_CreateMapping_instance (Execute, DBConnection, TableName, MapVar, SQLType, Done, Busy, Error, ErrorID);

Note The *DB_CreateMapping_instance* is an instance of DB_CreateMapping instruction, which is declared as a variable.

Variables

Input Variables

Name	Meaning	Data type	Valid range	Unit	Default	Description
Execute	Execute	BOOL	TRUE or FALSE	---	FALSE	Specify the execution condition.
DBConnection	DB Connection	DWORD	16#00000000 to 16#FFFFFFFF	---	16#0000 0000	Specify the DB connection established by a DB_Connect instruction.
TableName	Table Name	STRING	Depends on the data type.*	---	''	Specify a table name in the DB.
MapVar	DB Map Variable	Structure, Structure array (entire array)	Depends on the data type.	---	---	Specify a structure variable defined for accessing the DB.
SQLType	SQL Type	_eDBC_SQLTYPE	_DBC_SQLTYPE_INSERT(1): INSERT _DBC_SQLTYPE_UPDATE(2): UPDATE _DBC_SQLTYPE_SELECT(3): SELECT	---	0	Specify a type of SQL command for the variable to map.

* When the database is case sensitive, specify the table name as shown below.
When connecting to MySQL, enclose the table name in single-byte backquotes
Example: \`TableName1\`
When connecting to other databases, enclose the table name in single-byte double quotes.
Example: “TableName1”

Output Variables

Name	Meaning	Data type	Valid range	Unit	Description
Done	Done	BOOL	TRUE or FALSE	---	TRUE when the instruction is normally completed.
Busy	Executing	BOOL	TRUE or FALSE	---	TRUE when the instruction is being executed.

Error	Error	BOOL	TRUE or FALSE	---	TRUE when the instruction is terminated due to an error.
ErrorID	Error Code	WORD	16#0000 to 16#FFFF	---	Contains the error code when an error occurs.

Related System-defined Variables

Name	Meaning	Data type	Description
_EIP_EtnOnlineSta	Online	BOOL	Status of the communications function of the built-in EtherNet/IP port. TRUE: Can be used. FALSE: Cannot be used.

Related Error Codes

Error code	Meaning	Description
0400 hex	Input Value Out of Range	A value that is not defined as an enumerator was specified in the *SQLType* input variable.
0406 hex	Illegal Data Position Specified	The *TableName* input variable is a text string consisting of NULL characters (16#00) only.
0410 hex	Text String Format Error	A space character is included in the text string specified for the *TableName* input variable.
041B hex	Data Capacity Exceeded	The maximum number of DB Map Variables for which a mapping can be created is exceeded.
041D hex	Too Many Instructions Executed at the Same Time	More than 32 DB Connection Instructions were executed at the same time.
3000 hex	DB Connection Service not Started	The instruction was executed when the DB Connection Service was not running.
3002 hex	DB Connection Service Shutdown or Shutting Down	The instruction was executed after the DB Connection Service was shut down or while the DB Connection Service was being shut down.
3008 hex	Invalid DB Connection	The value of the *DBConnection* input variable is invalid or the specified DB Connection is already closed.
3009 hex	Invalid DB Map Variable	- The data type of the variable specified in the *MapVar* input variable is not a structure. - A derivative data type is included as a member of the structure variable specified in the *MapVar* input variable. - The DB Map Variable specified in the *MapVar* input variable is a structure array though INSERT or UPDATE is specified for the SQL Type.
300B hex	SQL Execution Error	The executed SQL statement resulted in an error in the DB.
3011 hex	DB Connection Disconnected Error Status	The DB Connection Service cannot communicate with the DB due to a network failure or other causes.
3013 hex	DB Connection Service Error Stop	The instruction was executed while the DB Connection Service was stopped due to an error.
3015 hex	DB Connection Service Initializing	The instruction was executed while the initialization processing of the DB Connection Service was in progress.

Function

This instruction is used to map the table specified in the *TableName* input variable with a DB Map Variable specified in the *MapVar* input variable.
You need to execute this instruction before executing a DB_Insert, DB_Update, or DB_Select instruction.
Specify the type of SQL command for the variable to map in the *SQLType* input variable. For example, specify *_DBC_SQLTYPE_INSERT* to insert the values of the DB Map Variable to the table using a DB_Insert instruction.

Precautions for Correct Use

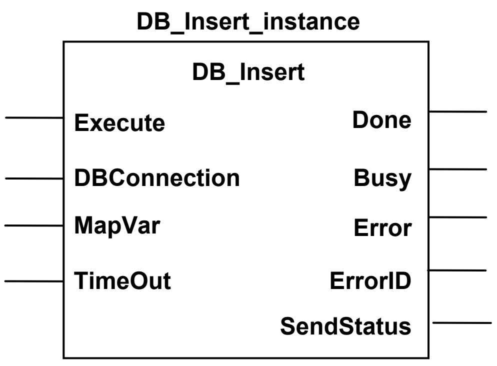
- Execution of this instruction is continued until processing is completed even if the value of *Execute* changes to FALSE or the execution time exceeds the task period. The value of *Done* changes to TRUE when processing is completed. Use this to confirm normal completion of processing.
- Refer to *Using this Section* of the *NJ/NX-series Instructions Reference Manual* (Cat. No. W502) for a timing chart for *Execute, Done, Busy,* and *Error*.
- This instruction cannot be used on an event task. A compiling error will occur.
- When the DB Connection Service was started in Test Mode, this instruction is completed normally without connecting to the DB actually.
- Refer to *1-2-1 DB Connection Service Specifications* for the number of DB Map Variables for which you can create a mapping. However, even if the number of DB Map Variables has not reached the upper limit, an instruction error (Data Capacity Exceeded) will occur when any of the following conditions is met.
    - When the total number of members of structures used as data type of DB Map Variables in all DB Connections exceeds 10,000 members
- An error occurs for this instruction in the following cases. *Error* will be TRUE.
    - When the instruction was executed when the DB Connection Service was not running
    - When the instruction was executed while the initialization processing of the DB Connection Service was in progress
    - When the instruction was executed while the DB Connection Service was stopped due to an error
    - When the instruction was executed after the DB Connection Service was shut down or while the DB Connection Service was being shut down
    - When the value of the *DBConnection* input variable is invalid or the specified DB Connection is already closed
    - When the *TableName* input variable is a text string consisting of NULL characters (16#00) only
    - When a space character is included in the text string specified for the *TableName* input variable.
    - When the data type of the variable specified in the *MapVar* input variable is not a structure
    - When a derivative data type is included as a member of the structure variable specified in the *MapVar* input variable
    - When the DB Map Variable specified in the *MapVar* input variable is a structure array though INSERT or UPDATE is specified for the SQL Type
    - When a value that is not defined as an enumerator was specified in the *SQLType* input variable
    - When the executed SQL statement resulted in an error in the DB
    - When the DB Connection Service cannot communicate with the DB due to a network failure or other causes
    - When the maximum number of DB Map Variables for which a mapping can be created is exceeded
    - When more than 32 DB Connection Instructions were executed at the same time

Sample Programming

Refer to the sample programming that is provided for the DB_Update instruction.

DB_Insert (Insert DB Record)

The DB_Insert instruction inserts values of a DB Map Variable to a table of the connected DB as a record.

Instruction	Name	FB/FUN	Graphic expression	ST expression
DB_Insert	Insert DB Record	FB	 DB_Insert_instance DB_Insert Execute Done DBConnection Busy MapVar Error TimeOut ErrorID SendStatus	DB_Insert_instance (Execute, DBConnection, MapVar, TimeOut, Done, Busy, Error, ErrorID, SendStatus);

Note The *DB_Insert_instance* is an instance of DB_Insert instruction, which is declared as a variable.

Variables

Input Variables

Name	Meaning	Data type	Valid range	Unit	Default	Description
Execute	Execute	BOOL	TRUE or FALSE	---	FALSE	Specify the execution condition.
DBConnection	DB Connection	DWORD	16#00000000 to 16#FFFFFFFF	---	16#0000 0000	Specify the DB Connection established by a DB_Connect instruction.
MapVar	DB Map Variable	Structure	Depends on the data type.	---	---	Specify the DB Map Variable mapped by a DB_CreateMapping instruction.
TimeOut	Timeout	TIME	T#0s, T#0.05s to T#180s	---	T#0s	Specify the time to detect timeout. When T#0s is specified, timeout is not monitored.

Output Variables

Name	Meaning	Data type	Valid range	Unit	Description
Done	Done	BOOL	TRUE or FALSE	---	TRUE when the instruction is normally completed.
Busy	Executing	BOOL	TRUE or FALSE	---	TRUE when the instruction is being executed.
Error	Error	BOOL	TRUE or FALSE	---	TRUE when the instruction is terminated due to an error.
ErrorID	Error Code	WORD	16#0000 to 16#FFFF	---	Contains the error code when an error occurs.
SendStatus	Send Status	_eDBC_SEND_STATUS	Depends on the data type.	---	Outputs the progress of transmission of the SQL statement.

Related System-defined Variables

Name	Meaning	Data type	Description
_EIP_EtnOnlineSta	Online	BOOL	Status of the communications function of the built-in EtherNet/IP port. TRUE: Can be used. FALSE: Cannot be used.

Related Error Codes

Error code	Meaning	Description
0400 hex	Input Value Out of Range	The value of the *TimeOut* input variable is outside the valid range.
041D hex	Too Many Instructions Executed at the Same Time	More than 32 DB Connection Instructions were executed at the same time.
3000 hex	DB Connection Service not Started	The instruction was executed when the DB Connection Service was not running.
3002 hex	DB Connection Service Shutdown or Shutting Down	The instruction was executed after the DB Connection Service was shut down or while the DB Connection Service was being shut down.
3008 hex	Invalid DB Connection	The value of the *DBConnection* input variable is invalid or the specified DB Connection is already closed.
300A hex	DB Map Variable Unregistered	The variable specified in the *MapVar* input variable has not been mapped by a DB_CreateMapping instruction.
300B hex	SQL Execution Error	The executed SQL statement resulted in an error in the DB. The combination of data types is not listed in the table of data type correspondence between NJ-series Controllers and database and the data type cannot be converted.
300C hex	Spool Capacity Exceeded	The SQL statement cannot be stored in the Spool memory because its capacity is exceeded.
3011 hex	DB Connection Disconnected Error Status	The DB Connection Service cannot communicate with the DB due to a network failure or other causes.
3012 hex	DB Connection Instruction Execution Timeout	The instruction was not completed within the time specified in the *TimeOut* input variable.
3013 hex	DB Connection Service Error Stop	The instruction was executed while the DB Connection Service was stopped due to an error.
3014 hex	Data Already Spooled	The SQL statement was spooled because one or more SQL statements are already stored in the Spool memory.
3015 hex	DB Connection Service Initializing	The instruction was executed while the initialization processing of the DB Connection Service was in progress.
3016 hex	DB in Process	The instruction was executed before completion of the DB's processing for the DB Connection Instruction Execution Timeout that occurred for the previous DB_Insert, DB_Update, DB_Select, or DB_Delete instruction.

Function

This instruction is used to insert the values of the DB Map Variable specified in the *MapVar* input variable to the table mapped by a DB_CreateMapping instruction as a record.

When the Spool function is enabled, the SQL statement is stored in the Spool memory in the following cases. In these cases, *_DBC_SEND_SPOOLED* is set in the *SendStatus* output variable and the instruction is terminated due to an error.

- When the values cannot be inserted to the DB due to a network failure or other causes (DB Connection Disconnected Error Status)
- When the values cannot be inserted to the DB within the time specified in the *TimeOut* input variable (DB Connection Instruction Execution Timeout)
- When one or more SQL statements are already stored in the Spool memory (Data Already Spooled)

If an instruction error (SQL Execution Error) occurs when the Spool function is enabled, the transmitted SQL statement itself can be the cause of the SQL Execution Error. Therefore, the SQL statement is not stored in the Spool memory because the SQL Execution Error may occur again when the SQL statement is resent.
When the Spool capacity for each DB Connection is exceeded by spooling the SQL statement, this instruction is terminated due to an error (Spool Capacity Exceeded).

Precautions for Correct Use

- Execution of this instruction is continued until processing is completed even if the value of *Execute* changes to FALSE or the execution time exceeds the task period. The value of *Done* changes to TRUE when processing is completed. Use this to confirm normal completion of processing.
- Refer to *Using this Section* of the *NJ/NX-series Instructions Reference Manual* (Cat. No. W502) for a timing chart for *Execute, Done, Busy*, and *Error*.
- This instruction cannot be used on an event task. A compiling error will occur.
- If the values cannot be registered to the DB, for example, because the SQL statement is invalid, this instruction is terminated due to an error without storing the SQL statement into the Spool memory.
- When the DB Connection Service was started in Test Mode, this instruction is completed normally without executing the INSERT operation for the DB actually.
- When the error code is 300B hex (SQL Execution Error), you can get the detailed information of the SQL Execution Error by executing a DB_GetConnectionStatus instruction.
- The measurement error of timeout is +50 ms for a 100-column record when the percentage of task execution time is 50% as a guide. However, the measurement error varies according to the percentage of task execution time and the number of columns.
- When two or more DB Connection Instructions are executed for a DB Connection at the same time, the DB Connection Service executes the instructions one by one. The measurement of timeout for the second and later instructions is started when the instruction is executed by the DB Connection Service, not when the *Execute* input variable is changed to TRUE. Therefore, the time from when the *Execute* input variable is changed to TRUE to when the timeout occurs for the instruction is longer than the time set for the timeout.
- If a value of a DB Map Variable is changed before the DB Connection Instruction is actually executed, the new value may be used when the DB Connection Instruction is executed. When changing a value of a DB Map Variable, write the user program so that the value is changed after confirming completion of the DB Connection Instruction.
- An error occurs for this instruction in the following cases. *Error* will be TRUE.
  - When the instruction was executed when the DB Connection Service was not running
  - When the instruction was executed while the initialization processing of the DB Connection Service

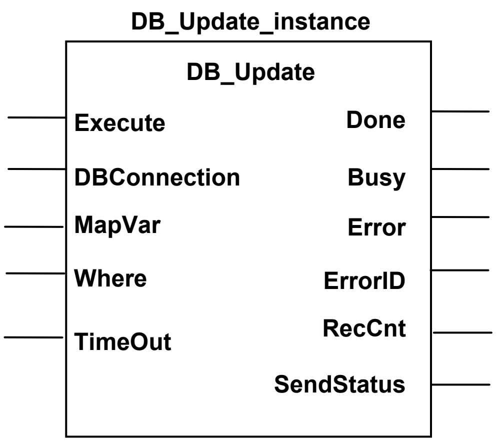
was in progress
- When the instruction was executed while the DB Connection Service was stopped due to an error
- When the instruction was executed after the DB Connection Service was shut down or while the DB Connection Service was being shut down
- When the value of the *DBConnection* input variable is invalid or the specified DB Connection is already closed
- When the variable specified in the *MapVar* input variable has not been mapped by a DB_CreateMapping instruction.
- When the value of the *Timeout* input variable is outside the valid range
- When the executed SQL statement resulted in an error in the DB
- When the combination of data types is not listed in the table of data type correspondence between NJ-series Controllers and database and the data type cannot be converted
- When the DB Connection Service cannot communicate with the DB due to a network failure or other causes
- When one or more SQL statements are already stored in the Spool memory
- When the SQL statement cannot be spooled because the Spool capacity is exceeded
- When the instruction was not completed within the time specified in the *TimeOut* input variable
- When the instruction was executed before completion of the DB's processing for the DB Connection Instruction Execution Timeout that occurred for the previous DB_Insert, DB_Update, DB_Select, or DB_Delete instruction
- When more than 32 DB Connection Instructions were executed at the same time

Sample Programming

Refer to the sample programming that is provided for the DB_Update instruction.

DB_Update (Update DB Record)

The DB_Update (Update DB Record) instruction updates the values of a record of a table with the values of a DB Map Variable.

Instruction	Name	FB/FUN	Graphic expression	ST expression
DB_Update	Update DB Record	FB	 DB_Update_instance DB_Update Execute DBConnection MapVar Where TimeOut Done Busy Error ErrorID RecCnt SendStatus	DB_Update_instance (Execute, DBConnection, MapVar, Where, TimeOut, Done, Busy, Error, ErrorID, RecCnt, SendStatus);

Note The *DB_Update_instance* is an instance of DB_Update instruction, which is declared as a variable.

Variables

Input Variables

Name	Meaning	Data type	Valid range	Unit	Default	Description
Execute	Execute	BOOL	TRUE or FALSE	---	FALSE	Specify the execution condition.
DBConnection	DB Connection	DWORD	16#00000000 to 16#FFFFFFFF	---	16#0000 0000	Specify the DB Connection established by a DB_Connect instruction.
MapVar	DB Map Variable	Structure	Depends on the data type.	---	---	Specify the DB Map Variable mapped by a DB_CreateMapping instruction.
Where	Retrieval Conditions	STRING	1,986 bytes max. (including the final NULL character)*	---	''	Specify a text string that expresses retrieval conditions (WHERE clause). ('WHERE' is not included.)
TimeOut	Timeout	TIME	T#0s, T#0.05s to T#180s	---	T#0s	Specify the time to detect timeout. When T#0s is specified, timeout is not monitored.

* When the database is case sensitive, specify the column name as shown below.
When connecting to MySQL, enclose the column name in single-byte backquotes
Example: \`ColumnA\`
When connecting to other databases, enclose the column name in single-byte double quotes.
Example: “ColumnA”

Output Variables

Name	Meaning	Data type	Valid range	Unit	Description
Done	Done	BOOL	TRUE or FALSE	---	TRUE when the instruction is normally completed.
Busy	Executing	BOOL	TRUE or FALSE	---	TRUE when the instruction is being executed.
Error	Error	BOOL	TRUE or FALSE	---	TRUE when the instruction is terminated due to an error.
ErrorID	Error Code	WORD	16#0000 to 16#FFFF	---	Contains the error code when an error occurs.
RecCnt	Number of Records	DINT	0 to 2147483647	---	Contains the number of records that were updated.
SendStatus	Send Status	_eDBC_SEND_STATUS	Depends on the data type.	---	Outputs the progress of transmission of the SQL statement.

Related System-defined Variables

Name	Meaning	Data type	Description
_EIP_EtnOnlineSta	Online	BOOL	Status of the communications function of the built-in EtherNet/IP port. TRUE: Can be used. FALSE: Cannot be used.

Related Error Codes

Error code	Meaning	Description
0400 hex	Input Value Out of Range	The value of the *Timeout* input variable is outside the valid range.
041D hex	Too Many Instructions Executed at the Same Time	More than 32 DB Connection Instructions were executed at the same time.
3000 hex	DB Connection Service not Started	The instruction was executed when the DB Connection Service was not running.
3002 hex	DB Connection Service Shutdown or Shutting Down	The instruction was executed after the DB Connection Service was shut down or while the DB Connection Service was being shut down.
3008 hex	Invalid DB Connection	The value of the *DBConnection* input variable is invalid or the specified DB Connection is already closed.
300A hex	DB Map Variable Unregistered	The variable specified in the *MapVar* input variable has not been mapped by a DB_CreateMapping instruction.
300B hex	SQL Execution Error	The executed SQL statement resulted in an error in the DB. The combination of data types is not listed in the table of data type correspondence between NJ-series Controllers and database and the data type cannot be converted.
300C hex	Spool Capacity Exceeded	The SQL statement cannot be stored in the Spool memory because its capacity is exceeded.
300E hex	Invalid Retrieval Conditions	The *Where* input variable is a text string consisting of NULL characters (16#00) only.
3011 hex	DB Connection Disconnected Error Status	The DB Connection Service cannot communicate with the DB due to a network failure or other causes.
3012 hex	DB Connection Instruction Execution Timeout	The instruction was not completed within the time specified in the *TimeOut* input variable.
3013 hex	DB Connection Service Error Stop	The instruction was executed while the DB Connection Service was stopped due to an error.
3014 hex	Data Already Spooled	The SQL statement was spooled because one or more SQL statements are already stored in the Spool memory.
3015 hex	DB Connection Service Initializing	The instruction was executed while the initialization processing of the DB Connection Service was in progress.

3016 hex	DB in Process	The instruction was executed before completion of the DB's processing for the DB Connection Instruction Execution Timeout that occurred for the previous DB_Insert, DB_Update, DB_Select, or DB_Delete instruction.

Function

This instruction is used to update the values of the records retrieved from the table mapped by a DB_CreateMapping instruction according to the retrieval conditions specified in the *Where* input variable (WHERE clause) with the values of a DB Map Variable specified in the *MapVar* input variable.

The records to update are retrieved according to the retrieval conditions specified in the *Where* input variable (WHERE clause). The *Where* input variable is expressed as a text string.
The text string in the *Where* input variable cannot consist of NULL characters (16#00) only. In that case, the instruction is terminated due to an error.

When using single quotes in the WHERE clause, use the escape character ($').
Refer to the *NJ/NX-series CPU Unit Software User's Manual* (Cat. No. W501) for the escape character.
Refer to the manual of the database for the format of the WHERE clause.

Specify the retrieval conditions by the following values in the *Where* input variable.

Example 1: Update the values of the records where the value of a specific column is equal to or greater than the specified value.
Update the values of records where the value of "ColumnA" (unsigned integer) is 1234 or greater.
'"ColumnA" >= 1234'
SQL statement to create: UPDATE TableProduct SET "ColumnA" =<value>, "ColumnB" =<value> Where "ColumnA" >= 1234

Example 2: Update the values of the records where the value of a specific column starts with the specified text string.
Update the values of records where the value of "ColumnB" (text string) starts with 'ABC'.
'"ColumnB" LIKE $'ABC%$''
SQL statement to create: UPDATE TableProduct SET "ColumnA" =<value>, "ColumnB" =<value> Where "ColumnB" LIKE 'ABC%'

Example 3: Update the values of the records where the value of a specific column is equal to or greater than the value of the specified variable.
Update the values of records where the value of "ColumnA" (unsigned integer) is equal to or greater than the specified variable.
Specified value: UINTVar := 1234;
Input parameter in the WHERE clause:
WhereCond_Update := CONCAT('$"ColumnA$" >= ', UINT_TO_STRING(UINTVar));
SQL statement to create:
UPDATE TableProduct SET "ColumnA" =<Value>, "ColumnB" =<Value> Where "ColumnA" >= 1234

When the Spool function is enabled, the SQL statement is stored in the Spool memory in the following cases. In these cases, *_DBC_SEND_SPOOLED* is set in the *SendStatus* output variable and the instruction is terminated due to an error.

- When the DB records cannot be updated due to a network failure or other causes (DB Connection Disconnected Error Status)
- When the DB records cannot be updated within the time specified in the *TimeOut* input variable (DB Connection Instruction Execution Timeout)

If an instruction error (SQL Execution Error) occurs when the Spool function is enabled, the transmitted SQL statement itself can be the cause of the SQL Execution Error, for example, due to a retrieval condition setting error. Therefore, the SQL statement is not stored in the Spool memory because the SQL Execution Error may occur again when the SQL statement is resent.

When the Spool capacity for each DB Connection is exceeded by spooling the SQL statement, this instruction is terminated due to an error (Spool Capacity Exceeded).

Precautions for Correct Use

- Execution of this instruction is continued until processing is completed even if the value of *Execute* changes to FALSE or the execution time exceeds the task period. The value of *Done* changes to TRUE when processing is completed. Use this to confirm normal completion of processing.
- Refer to *Using this Section* of the *NJ/NX-series Instructions Reference Manual* (Cat. No. W502) for a timing chart for *Execute, Done, Busy,* and *Error*.
- This instruction cannot be used on an event task. A compiling error will occur.
- This instruction cannot be executed without specifying the retrieval conditions.
- If the values cannot be registered to the DB, for example, because the SQL statement is invalid, this instruction is terminated due to an error without storing the SQL statement into the Spool memory.
- When the DB Connection Service was started in Test Mode, this instruction is completed normally without executing the UPDATE operation for the DB actually.
- When the error code is 300B hex (SQL Execution Error), you can get the detailed information of the SQL Execution Error by executing a DB_GetConnectionStatus instruction
- The measurement error of timeout is +50 ms for a 100-column record when the percentage of task execution time is 50% as a guide. However, the measurement error varies according to the percentage of task execution time and the number of columns.
- When two or more DB Connection Instructions are executed for a DB Connection at the same time, the DB Connection Service executes the instructions one by one. The measurement of timeout for the second and later instructions is started when the instruction is executed by the DB Connection Service, not when the *Execute* input variable is changed to TRUE. Therefore, the time from when the *Execute* input variable is changed to TRUE to when the timeout occurs for the instruction is longer than the time set for the timeout.
- If a value of a DB Map Variable is changed before the DB Connection Instruction is actually executed, the new value may be used when the DB Connection Instruction is executed. When changing a value of a DB Map Variable, write the user program so that the value is changed after confirming completion of the DB Connection Instruction.
- An error occurs for this instruction in the following cases. *Error* will be TRUE.
    - When the instruction was executed when the DB Connection Service was not running
    - When the instruction was executed while the initialization processing of the DB Connection Service was in progress
    - When the instruction was executed while the DB Connection Service was stopped due to an error
    - When the instruction was executed after the DB Connection Service was shut down or while the DB Connection Service was being shut down
    - When the value of the *DBConnection* input variable is invalid or the specified DB Connection is already closed
    - When the variable specified in the *MapVar* input variable has not been mapped by a DB_CreateMapping instruction
    - When the *Where* input variable is a text string consisting of NULL characters (16#00) only
    - When the value of the *Timeout* input variable is outside the valid range

- When the executed SQL statement resulted in an error in the DB
- When the combination of data types is not listed in the table of data type correspondence between NJ-series Controllers and database and the data type cannot be converted
- When the DB Connection Service cannot communicate with the DB due to a network failure or other causes
- When the SQL statement cannot be spooled because the Spool capacity is exceeded
- When one or more SQL statements are already stored in the Spool memory
- When the instruction was not completed within the time specified in the *TimeOut* input variable
- When the instruction was executed before completion of the DB's processing for the DB Connection Instruction Execution Timeout that occurred for the previous DB_Insert, DB_Update, DB_Select, or DB_Delete instruction
- When more than 32 DB Connection Instructions were executed at the same time

Sample Programming

This section gives sample programming for the following operations.

- Insert production data into a specified DB when the trigger variable changes to TRUE.
- Update production data in a specified DB when the trigger variable changes to TRUE.

DB Connection Settings and Data Type Definition

The minimum settings necessary for the sample programming are shown below.

● DB Connection Settings

DB Connection name: MyDatabase1

● Structure Data Type Definition

Name		Data type
PRODUCTION_INSERT		STRUCT
	Name	STRING[256]
	LotNo	STRING[32]
	Status	STRING[8]
	ProductionDate	DATE

Name		Data type
PRODUCTION_UPDATE		STRUCT
	Status	STRING[8]
	FinishTime	DATE_AND_TIME

Ladder Diagram

● Main Variables

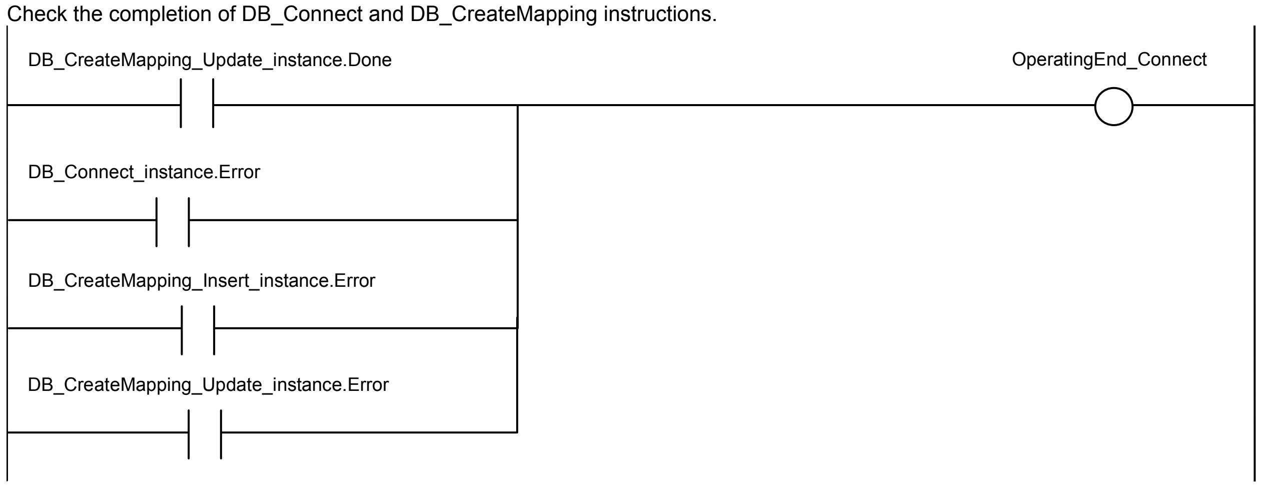
Name	Data type	Initial value	Comment
_DBC_Status	_sDBC_STATUS	---	System-defined variable that shows the status of the DB Connection Service
DB_Connect_instance	DB_Connect	---	Instance of DB_Connect instruction
MyDB1	DWORD	---	This variable is assigned to the *DBConnection* output variable from *DB_Connect_instance.*
Trigger_Connect	BOOL	FALSE	Variable used as a trigger for establishing a DB Connection

Name	Data type	Initial value	Comment
RS_Connect_instance	RS	---	Instance of RS instruction
Operating_Connect	BOOL	FALSE	The DB_Connect instruction is executed when this variable is TRUE.
OperatingEnd_Connect	BOOL	FALSE	This variable changes to TRUE when the DB_Connect instruction is completed.
DB_CreateMapping_Insert_instance	DB_CreateMapping	---	Instance of DB_CreateMapping instruction
MapVar_Insert	PRODUCTION_INSERT		This variable is assigned to the *MapVar* input variable to *DB_CreateMapping_Insert_instance*.
DB_Insert_instance	DB_Insert	---	Instance of DB_Insert instruction
Name	STRING[256]	‘WORK001’	Production information: Product name
LotNo	UINT	1234	Production information: Lot number
Trigger_Insert	BOOL	FALSE	Variable used as a trigger for inserting DB records
RS_Insert_instance	RS	---	Instance of RS instruction
Operating_Insert	BOOL	FALSE	The DB_Insert instruction is executed when this variable is TRUE.
OperatingEnd_Insert	BOOL	FALSE	This variable changes to TRUE when the DB_Insert instruction is completed.
DB_CreateMapping_Update_instance	DB_CreateMapping	---	Instance of DB_CreateMapping instruction
MapVar_Update	PRODUCTION_UPDATE		This variable is assigned to the *MapVar* input variable to *DB_CreateMapping_Update_instance*.
WhereCond	STRING[256]	---	This variable is assigned to the *Where* input variable to *DB_CreateMapping_Update_instance*.
DB_Update_instance	DB_Update	---	Instance of DB_Update instruction
Trigger_Update	BOOL	FALSE	Variable used as a trigger for updating DB records
RS_Update_instance	RS	---	Instance of RS instruction
Operating_Update	BOOL	FALSE	The DB_Update instruction is executed when this variable is TRUE.
OperatingEnd_Update	BOOL	FALSE	This variable changes to TRUE when the DB_Update instruction is completed.
DB_Close_instance	DB_Close	---	Instance of DB_Close instruction
Trigger_Close	BOOL	FALSE	Variable used as a trigger for closing the DB Connection
RS_Close_instance	RS	---	Instance of RS instruction
Operating_Close	BOOL	FALSE	The DB_Close instruction is executed when this variable is TRUE.
OperatingEnd_Close	BOOL	FALSE	This variable changes to TRUE when the DB_Close instruction is completed.

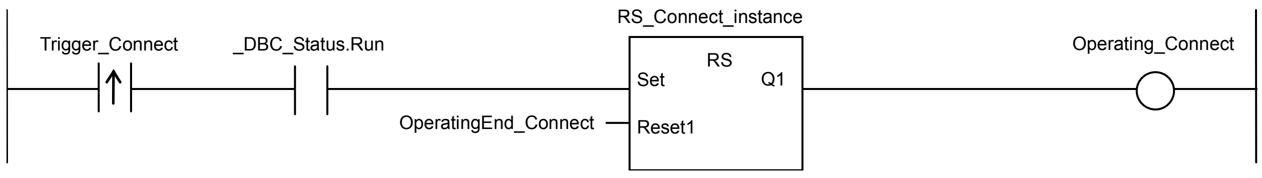
● Sample Programming

- Establish a DB Connection named MyDatabase1 and map a table with a variable.

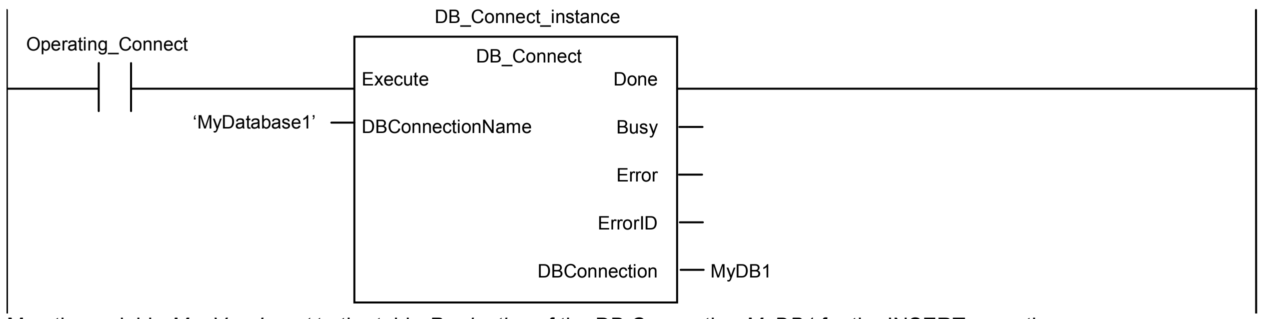
Check the completion of DB_Connect and DB_CreateMapping instructions.


DB_CreateMapping_Update_instance.Done
OperatingEnd_Connect
DB_Connect_instance.Error
DB_CreateMapping_Insert_instance.Error
DB_CreateMapping_Update_instance.Error

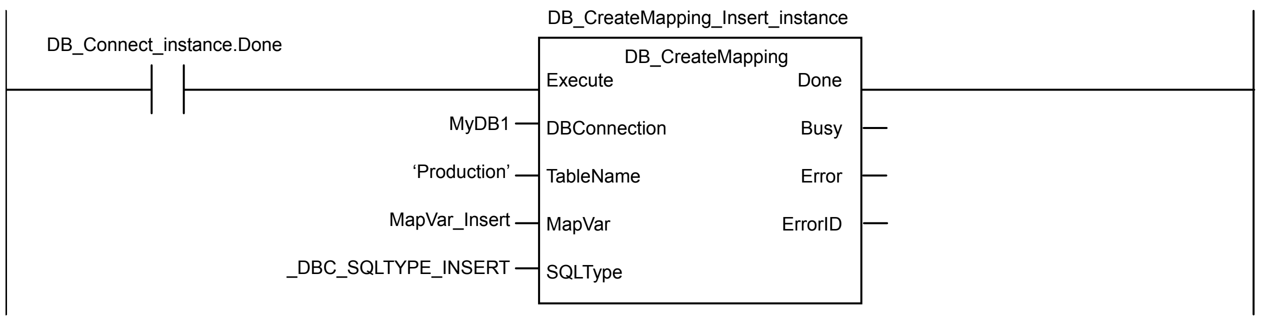
Accept the trigger for establishing the DB Connection.



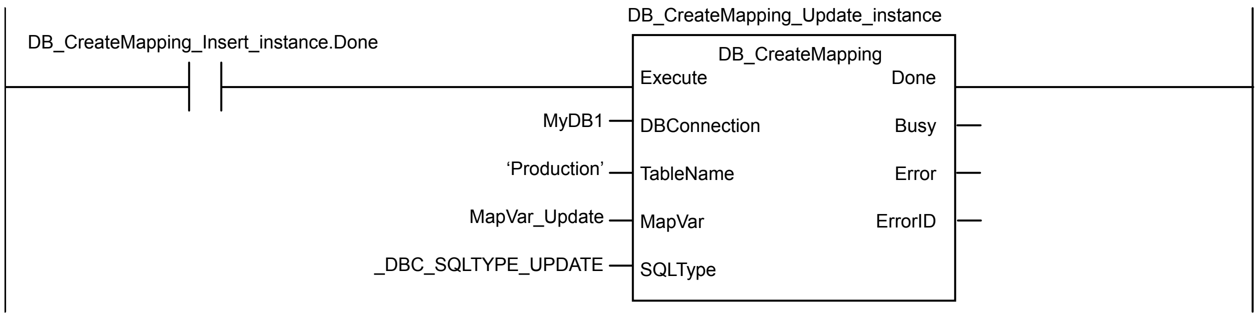
Establish the DB Connection named MyDatabase1.



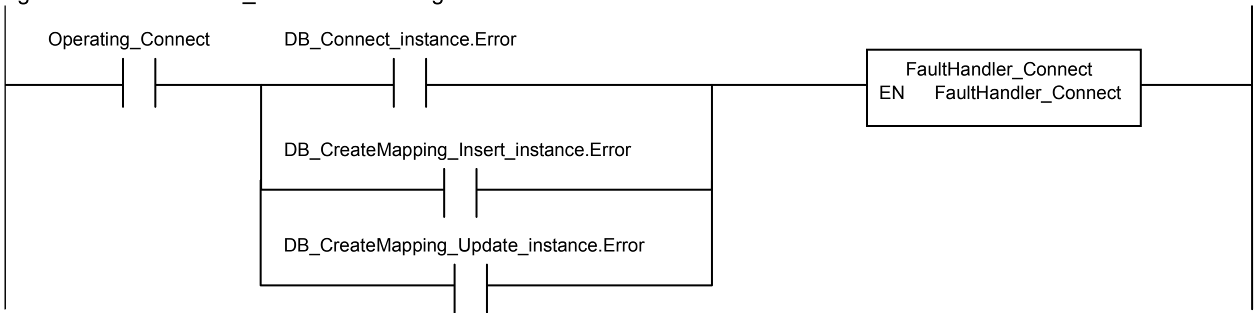
Map the variable *MapVar_Insert* to the table *Production* of the DB Connection *MyDB1* for the INSERT operation.



Map the variable *MapVar_Update* to the table *Production* of the DB Connection *MyDB1* for the UPDATE operation.

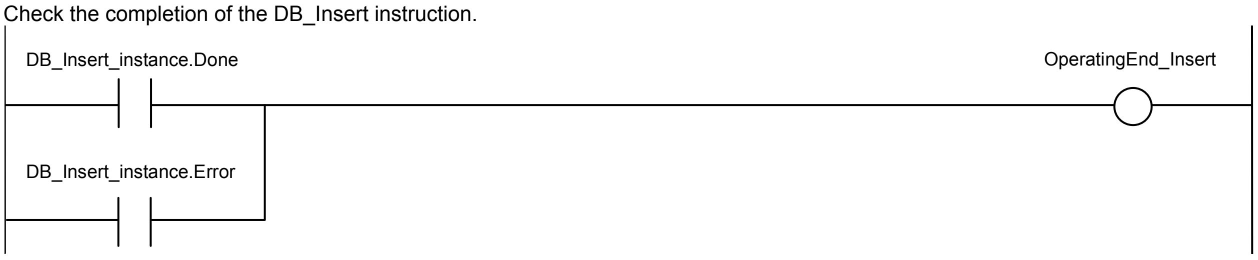


When the instruction is terminated due to an error, execute the error handler for the device (FaultHandler_Connect).
Program the FaultHandler_Connect according to the device.

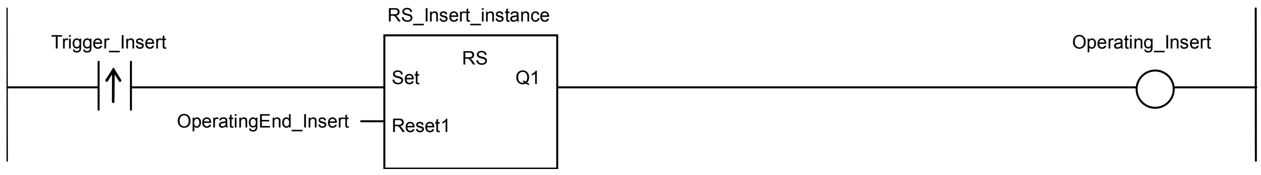

Operating_Connect
DB_Connect_instance.Error
FaultHandler_Connect
EN FaultHandler_Connect
DB_CreateMapping_Insert_instance.Error
DB_CreateMapping_Update_instance.Error

- Insert production data to the DB Connection *MyDB1* when the variable *Trigger_Insert* changes to TRUE.

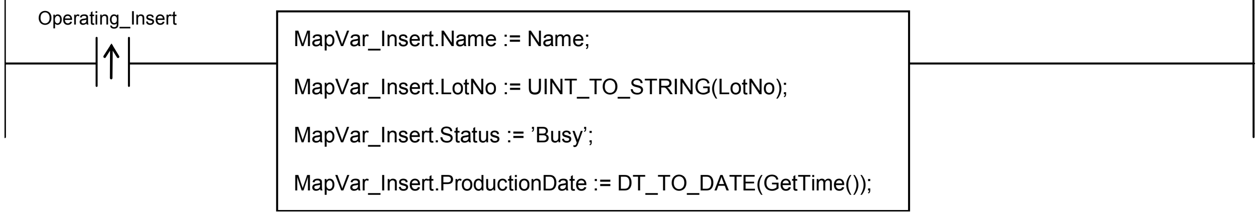
Check the completion of the DB_Insert instruction.



Accept the trigger for inserting DB records.



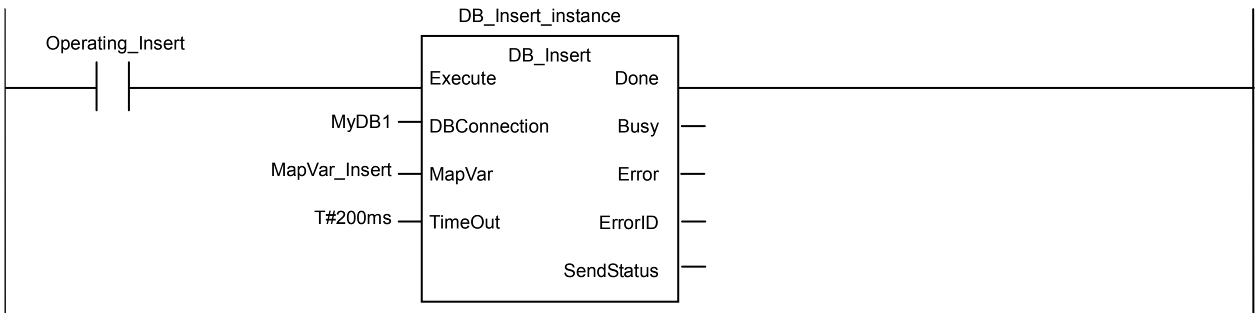
Create production data to insert.



```
MapVar_Insert.Name := Name;
MapVar_Insert.LotNo := UINT_TO_STRING(LotNo);
MapVar_Insert.Status := 'Busy';
MapVar_Insert.ProductionDate := DT_TO_DATE(GetTime());
```

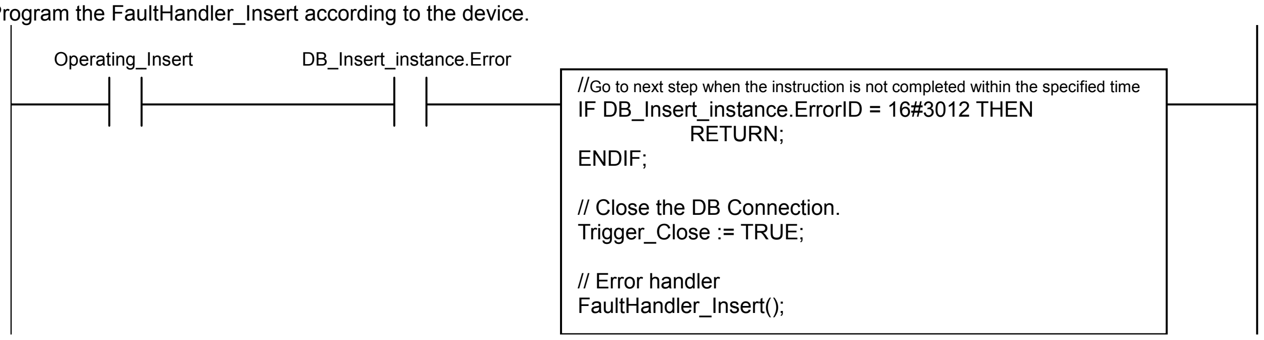
Insert production data to the DB Connection *MyDB1*.

Set the timeout for instruction execution to 200 ms.



When the instruction is terminated due to an error, execute the error handler for the device (FaultHandler_Insert).

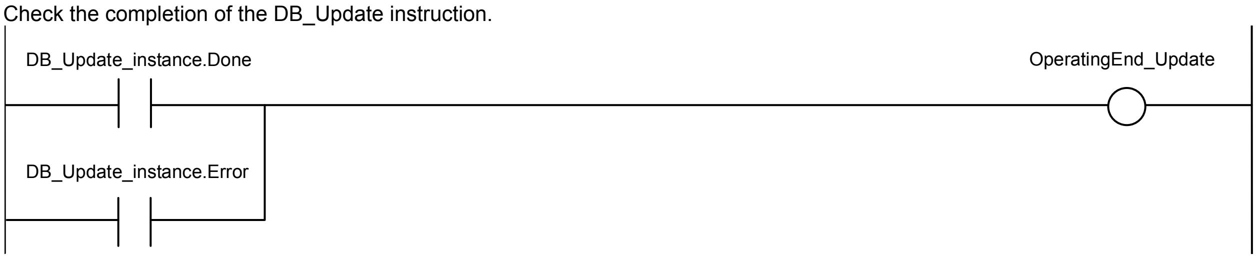
Program the FaultHandler_Insert according to the device.



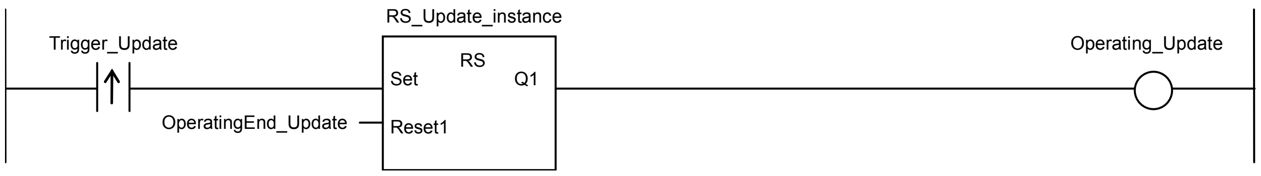
```
//Go to next step when the instruction is not completed within the specified time
IF DB_Insert_instance.ErrorID = 16#3012 THEN
            RETURN;
ENDIF;

// Close the DB Connection.
Trigger_Close := TRUE;

// Error handler
FaultHandler_Insert();
```

- Update the records in the DB Connection *MyDB1* when the variable *Trigger_Update* changes to TRUE.

Check the completion of the DB_Update instruction.

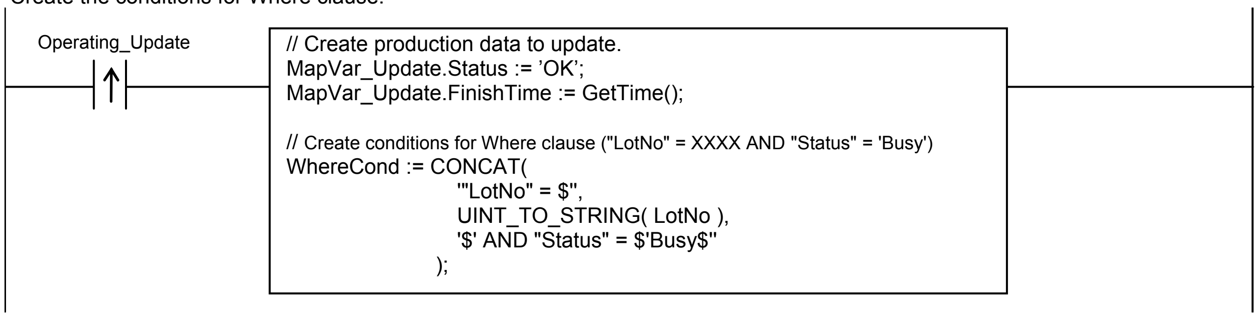


Accept the trigger for updating DB records.



Create production data to update.

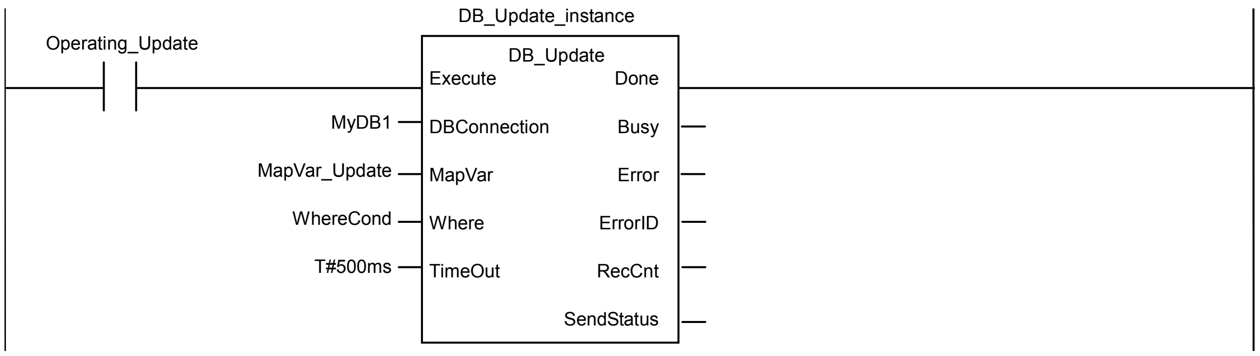
Create the conditions for Where clause.



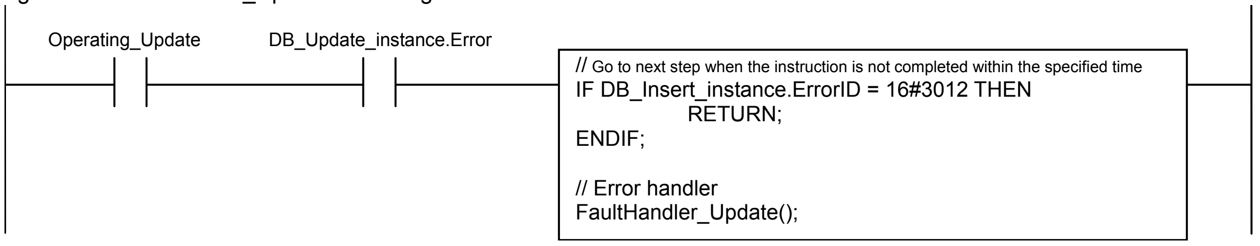
```
// Create production data to update.
MapVar_Update.Status := 'OK';
MapVar_Update.FinishTime := GetTime();

// Create conditions for Where clause ("LotNo" = XXXX AND "Status" = 'Busy')
WhereCond := CONCAT(
                    '"LotNo" = $'',
                    UINT_TO_STRING( LotNo ),
                    '$' AND "Status" = $'Busy$''
                );
```

Update production data in the DB Connection *MyDB1*.

Set the timeout for instruction execution to 500 ms.



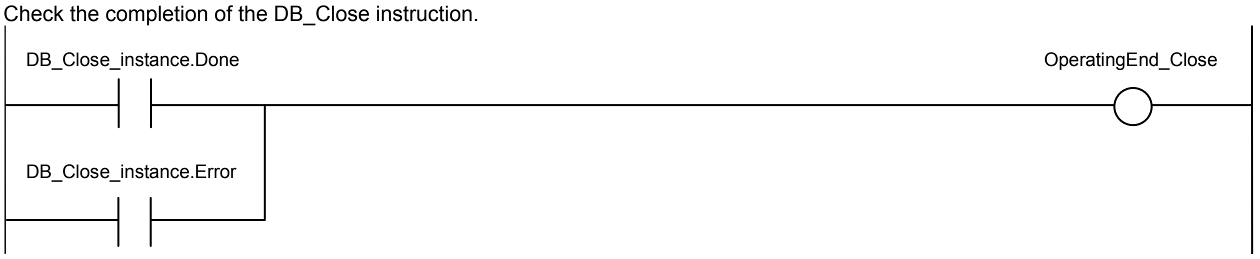
When the instruction is terminated due to an error, execute the error handler for the device (FaultHandler_Update). Program the FaultHandler_Update according to the device.



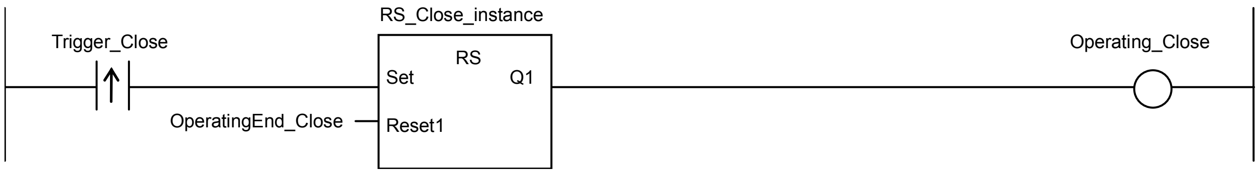
```
// Go to next step when the instruction is not completed within the specified time
IF DB_Insert_instance.ErrorID = 16#3012 THEN
        RETURN;
ENDIF;

// Error handler
FaultHandler_Update();
```

Close the DB Connection *MyDB1*.

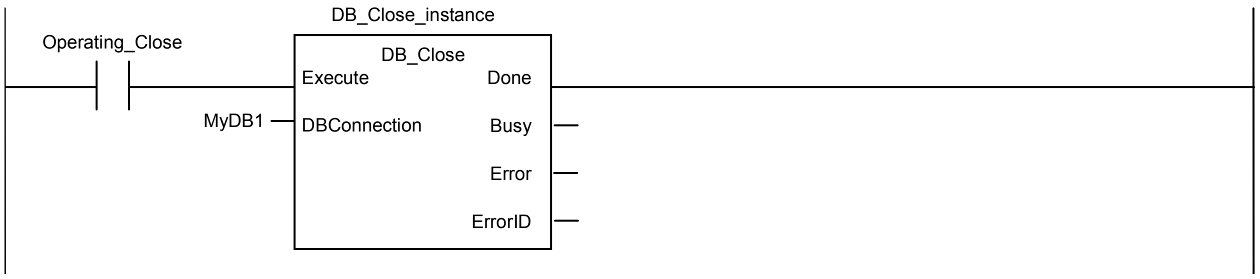
Check the completion of the DB_Close instruction.



Accept the trigger for closing the DB Connection.

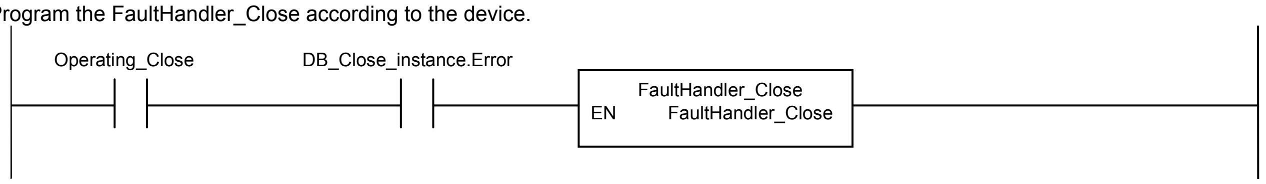


Close the DB Connection *MyDB1*.



When the instruction is terminated due to an error, execute the error handler for the device (FaultHandler_Close).

Program the FaultHandler_Close according to the device.



Structured Text (ST)

● Main Variables

Name	Data type	Initial value	Comment
_DBC_Status	_sDBC_STATUS	---	System-defined variable that shows the status of the DB Connection Service
DB_Connect_instance	DB_Connect	---	Instance of DB_Connect instruction
MyDB1	DWORD	---	This variable is assigned to the *DBConnection* output variable from *DB_Connect_instance.*
Trigger_Connect	BOOL	FALSE	Variable used as a trigger for establishing a DB Connection
LastTrigger_Connect	BOOL	FALSE	Variable to retain the trigger status of the previous execution
Operating_Connect	BOOL	FALSE	The DB_Connect instruction is executed when this variable is TRUE.
OperatingStart_Connect	BOOL	FALSE	The start processing for establishing the DB Connection is executed when this variable is TRUE.
DB_CreateMapping_Insert_instance	DB_CreateMapping	---	Instance of DB_CreateMapping instruction
MapVar_Insert	PRODUCTION_INSERT	---	This variable is assigned to the *MapVar* input variable to *DB_CreateMapping_Insert_instance*.
DB_Insert_instance	DB_Insert	---	Instance of DB_Insert instruction
Name	STRING[256]	‘WORK001’	Production information: Product name
LotNo	UINT	1234	Production information: Lot number
Trigger_Insert	BOOL	FALSE	Variable used as a trigger for inserting DB records
LastTrigger_Insert	BOOL	FALSE	Variable to retain the trigger status of the previous execution
Operating_Insert	BOOL	FALSE	The DB_Insert instruction is executed when this variable is TRUE.
OperatingStart_Insert	BOOL	FALSE	The start processing for inserting DB records is executed when this variable is TRUE.
DB_CreateMapping_Update_instance	DB_CreateMapping	---	Instance of DB_CreateMapping instruction
MapVar_Update	PRODUCTION_UPDATE	---	This variable is assigned to the *MapVar* input variable to *DB_CreateMapping_Update_instance*.
DB_Update_instance	DB_Update	---	Instance of DB_Update instruction
WhereCond	STRING[256]	---	This variable is assigned to the *Where* input variable to *DB_CreateMapping_Update_instance*.
Trigger_Update	BOOL	FALSE	Variable used as a trigger for updating DB records
LastTrigger_Update	BOOL	FALSE	Variable to retain the trigger status of the previous execution
Operating_Update	BOOL	FALSE	The DB_Update instruction is executed when this variable is TRUE.
OperatingStart_Update	BOOL	FALSE	The start processing for updating DB records is executed when this variable is TRUE.

Name	Data type	Initial value	Comment
DB_Close_instance	DB_Close	---	Instance of DB_Close instruction
Trigger_Close	BOOL	FALSE	Variable used as a trigger for closing the DB Connection
LastTrigger_Close	BOOL	FALSE	Variable to retain the trigger status of the previous execution
Operating_Close	BOOL	FALSE	The DB_Close instruction is executed when this variable is TRUE.
OperatingStart_Close	BOOL	FALSE	The start processing for closing the DB Connection is executed when this variable is TRUE.
Stage	INT	---	Variable that shows the status of the DB Connection

● Sample Programming

```
// - Establish a DB Connection named MyDatabase1 and map a table with a variable.

// Start the sequence when the variable Trigger_Connect changes to TRUE.
   IF ( (Trigger_Connect=TRUE)
     AND (LastTrigger_Connect=FALSE)
     AND (_DBC_Status.Run=TRUE) ) THEN
       OperatingStart_Connect := TRUE;
      Operating_Connect := TRUE;
    END_IF;
    LastTrigger_Connect:=Trigger_Connect;

   // Sequence start processing
    IF (OperatingStart_Connect=TRUE) THEN
    // Initialize the instances of the applicable DB Connection Instructions.
       DB_Connect_instance( Execute:=FALSE );

       DB_CreateMapping_Insert_instance(
          Execute      := FALSE,
          MapVar       := MapVar_Insert,
      SQLType          := _DBC_SQLTYPE_INSERT
    );

    DB_CreateMapping_Update_instance(
       Execute      := FALSE,
       MapVar       := MapVar_Update,
       SQLType   := _DBC_SQLTYPE_UPDATE
       );

       Stage := INT#1;
       OperatingStart_Connect := FALSE;
    END_IF;

    // Establish the DB Connection named MyDatabese1
// Map the variable MapVar_Insert to the table Production of the DB Connection MyDB1 for the INSERT operation.
// Map the variable MapVar_Update to the table Production of the DB Connection MyDB1 for the UPDATE operation.
IF (Operating_Connect=TRUE) THEN
  CASE Stage OF
  1 : // Establish the DB Connection
    DB_Connect_instance(
      Execute               := TRUE,
      DBConnectionName      := 'MyDatabase1',
      DBConnection          => MyDB1
    );

    IF (DB_Connect_instance.Done=TRUE) THEN
      Stage := INT#2; // Normal end
    END_IF;
    IF (DB_Connect_instance.Error=TRUE) THEN
      Stage := INT#99; // Error
    END_IF;

  2 : // Map the DB table with the variable
    DB_CreateMapping_Insert_instance(
      Execute           := TRUE,
      DBConnection      := MyDB1,
      TableName         := 'Production',
      MapVar            := MapVar_Insert,
      SQLType           := _DBC_SQLTYPE_INSERT
    );

    DB_CreateMapping_Update_instance(
      Execute           := TRUE,
      DBConnection      := MyDB1,
      TableName         := 'Production',
      MapVar            := MapVar_Update,
      SQLType           := _DBC_SQLTYPE_UPDATE
    );

    IF ( (DB_CreateMapping_Insert_instance.Done=TRUE)
      AND (DB_CreateMapping_Update_instance.Done=TRUE) ) THEN
        Operating_Connect:=FALSE; // Normal end
    END_IF;
    IF ( (DB_CreateMapping_Insert_instance.Error=TRUE)
      OR (DB_CreateMapping_Update_instance.Error = TRUE) ) THEN
        Stage := INT#99; // Error
    END_IF;

  99 :
    // Execute the error handler.
    // Program the error hander (FaultHandler_Connect) according to the device.
    FaultHandler_Connect();
```

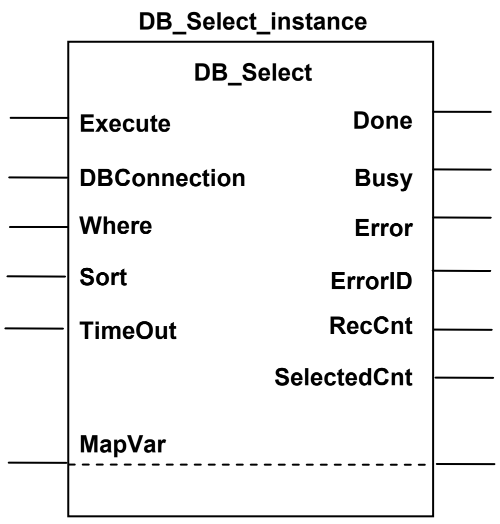
```
        Operating_Connect := FALSE;
    END_CASE;
END_IF;



// -----------------------------------------------------------------------------------------------------------------------
// - Insert production data to DB Connection MyDB1 when the variable Trigger_Insert changes to TRUE.

// Start the sequence when the variable Trigger_Insert changes to TRUE.
IF ( (Trigger_Insert=TRUE) AND (LastTrigger_Insert=FALSE) ) THEN
    OperatingStart_Insert := TRUE;
    Operating_Insert := TRUE;
END_IF;
LastTrigger_Insert := Trigger_Insert;

// Sequence start processing
IF (OperatingStart_Insert=TRUE) THEN
    // Initialize the instance of the applicable DB Connection Instruction.
    DB_Insert_instance( Execute:=FALSE, MapVar:=MapVar_Insert );

    // Create production data to insert.
    MapVar_Insert.Name             := Name;
    MapVar_Insert.LotNo            := UINT_TO_STRING(LotNo);
    MapVar_Insert.Status           := 'Busy';
    MapVar_Insert.ProductionDate   := DT_TO_DATE(GetTime( ));

    OperatingStart_Insert := FALSE;
END_IF;

// Insert production data to the DB Connection MyDB1. Set the timeout for instruction execution to 200 ms.
IF (Operating_Insert=TRUE) THEN
    // Insert records
    DB_Insert_instance(
        Execute         := TRUE,
        DBConnection    := MyDB1,
        MapVar          := MapVar_Insert,
        TimeOut         := T#200ms
    );

IF (DB_Insert_instance.Done=TRUE) THEN
    Operating_Insert:=FALSE; // Normal end
END_IF;
IF (DB_Insert_instance.Error=TRUE) THEN
    // Go to the next step when the instruction is not completed within the specified time.
    IF (DB_Insert_instance.ErrorID = 16#3012) THEN
        Operating_Insert:=FALSE; // Normal end
    ELSE
        // Execute the error handler.
```

```
            // Program the error handler (FaultHandler_Insert) according to the device.
            FaultHandler_Insert();
            Operating_Insert := FALSE;
        END_IF;
    END_IF;
END_IF;
    // ------------------------------------------------------------------------------------------------------------------------
    // - Update the records in the DB Connection MyDB1 when the variable Trigger_Update changes to TRUE.

    // Start the sequence when the variable Trigger_Update changes to TRUE.
    IF ( (Trigger_Update=TRUE) AND (LastTrigger_Update=FALSE) ) THEN
        OperatingStart_Update := TRUE;
        Operating_Update := TRUE;
    END_IF;
    LastTrigger_Update := Trigger_Update;

    // Sequence start processing
    IF (OperatingStart_Update=TRUE) THEN
        // Initialize the instance of the applicable DB Connection Instruction.
        DB_Update_instance( Execute:=FALSE, MapVar:=MapVar_Update );

        // Create production data to update.
        MapVar_Update.Status := 'OK';
        MapVar_Update.FinishTime := GetTime();

        // Create the conditions for Where clause. ("LotNo" = XXXX AND "Status" = 'Busy')
        WhereCond := CONCAT(
                                '"LotNo" = $'',
                                UINT_TO_STRING( LotNo ),
                                '$' AND "Status" = $'Busy$''
                            );

        OperatingStart_Update := FALSE;
    END_IF;

    // Update production data in the DB Connection MyDB1. Set the timeout for instruction execution to 200 ms.
    IF (Operating_Update=TRUE) THEN
        // Update records
        DB_Update_instance(
            Execute         := TRUE,
            DBConnection    := MyDB1,
            MapVar          := MapVar_Update,
            Where           := WhereCond,
            TimeOut         := T#200ms );

    IF (DB_Update_instance.Done=TRUE) THEN
        Operating_Update:=FALSE; // Normal end
    END_IF;
```

```
        IF (DB_Update_instance.Error=TRUE) THEN
            // Go to the next step when the instruction is not completed within the specified time.
            IF (DB_Update_instance.ErrorID = 16#3012) THEN
                Operating_Update:=FALSE; // Normal end
            ELSE
                // Execute the error handler.
                // Implement the error handler (FaultHandler_Update) according to the device.
                FaultHandler_Update();
                Operating_Update := FALSE;
            END_IF;
        END_IF;
    END_IF;

    // ------------------------------------------------------------------------------------------------------------------------
    // - Close the DB Connection "MyDB1".

    // Start the sequence when the variable Trigger_Close changes to TRUE.
    IF ( (Trigger_Close=TRUE) AND (LastTrigger_Close=FALSE) ) THEN
        OperatingStart_Close := TRUE;
        Operating_Close := TRUE;
    END_IF;
    LastTrigger_Close := Trigger_Close;

    // Sequence start processing
    IF (OperatingStart_Close=TRUE) THEN
        // Initialize the instance of the applicable DB Connection Instruction.
        DB_Close_instance( Execute:=FALSE );

        OperatingStart_Close := FALSE;
    END_IF;

    // Close the DB Connection "MyDB1".
    IF (Operating_Close=TRUE) THEN
        // Close the DB Connection.
        DB_Close_instance( Execute:=TRUE, DBConnection:=MyDB1 );

        IF (DB_Close_instance.Done=TRUE) THEN
            Operating_Close := FALSE; // Normal end
        END_IF;
        IF (DB_Close_instance.Error=TRUE) THEN
            // Execute the error handler.
            // Program the error handler (FaultHandler_Close) according to the device.
            FaultHandler_Close();
            Operating_Close := FALSE;
        END_IF;
    END_IF;
```

DB_Select (Retrieve DB Record)

The DB_Select instruction retrieves records from a table to a DB Map Variable.

Instruction	Name	FB/FUN	Graphic expression	ST expression
DB_Select	Retrieve DB Record	FB	 DB_Select_instance DB_Select Execute DBConnection Where Sort TimeOut MapVar Done Busy Error ErrorID RecCnt SelectedCnt	DB_Select_instance (Execute, DBConnection, Where, Sort, TimeOut, MapVar, Done, Busy, Error, ErrorID, RecCnt, SelectedCnt);

Note The *DB_Select_instance* is an instance of DB_Select instruction, which is declared as a variable.

Variables

Input Variables

Name	Meaning	Data type	Valid range	Unit	Default	Description
Execute	Execute	BOOL	TRUE or FALSE	---	FALSE	Specify the execution condition.
DBConnection	DB Connection	DWORD	16#00000000 to 16#FFFFFFFF	---	16#0000 0000	Specify the DB Connection established by a DB_Connect instruction.
Where	Retrieval Conditions	STRING	1,986 bytes max. (including the final NULL character)*	---	''	Specify a text string that expresses retrieval conditions (WHERE clause). (‘WHERE’ is not included.)
Sort	Sort Conditions	STRING	1,986 bytes max. (including the final NULL character)*	---	''	Specify a text string that expresses sort conditions (ORDER BY clause). (‘ORDER BY’ is not included.)
TimeOut	Timeout	TIME	T#0s, T#0.05s to T#180s	---	T#0s	Specify the time to detect timeout. When T#0s is specified, timeout is not monitored.

* When the database is case sensitive, specify the column name as shown below.
When connecting to MySQL, enclose the column name in single-byte backquotes
Example: \`ColumnA\`
When connecting to other databases, enclose the column name in single-byte double quotes.
Example: “ColumnA”

In-out Variables

Name	Meaning	Data type	Valid range	Unit	Description
MapVar	DB Map Variable	Structure, Structure array (entire array)	Depends on the data type.	---	Specify the DB Map Variable mapped by a DB_CreateMapping instruction.

Output Variables

Name	Meaning	Data type	Valid range	Unit	Description
Done	Done	BOOL	TRUE or FALSE	---	TRUE when the instruction is normally completed.
Busy	Executing	BOOL	TRUE or FALSE	---	TRUE when the instruction is being executed.
Error	Error	BOOL	TRUE or FALSE	---	TRUE when the instruction is terminated due to an error.
ErrorID	Error Code	WORD	16#0000 to 16#FFFF	---	Contains the error code when an error occurs.
RecCnt	Number of Records	DINT	0 to 65535	---	Contains the number of records that were retrieved to the DB Map Variable.
SelectedCnt	Number of Retrieved Records	DINT	0 to 2147483647	---	Total number of records retrieved according to the retrieval conditions.

Related System-defined Variables

Name	Meaning	Data type	Description
_EIP_EtnOnlineSta	Online	BOOL	Status of the communications function of the built-in EtherNet/IP port. TRUE: Can be used. FALSE: Cannot be used.

Related Error Codes

Error code	Meaning	Description
0400 hex	Input Value Out of Range	The value of the *TimeOut* input variable is outside the valid range.
041D hex	Too Many Instructions Executed at the Same Time	More than 32 DB Connection Instructions were executed at the same time.
3000 hex	DB Connection Service not Started	The instruction was executed when the DB Connection Service was not running.
3002 hex	DB Connection Service Shutdown or Shutting Down	The instruction was executed after the DB Connection Service was shut down or while the DB Connection Service was being shut down.
3008 hex	Invalid DB Connection	The value of the *DBConnection* input variable is invalid or the specified DB Connection is already closed.
300A hex	DB Map Variable Unregistered	The variable specified in the *MapVar* in-out variable has not been mapped by a DB_CreateMapping instruction.
300B hex	SQL Execution Error	The executed SQL statement resulted in an error in the DB. The retrieved record contains a column whose value is NULL. The combination of data types is not listed in the table of data type correspondence between NJ-series Controllers and database and the data type cannot be converted.
300E hex	Invalid Retrieval Conditions	The *Where* input variable is a text string consisting of NULL characters (16#00) only.

3011 hex	DB Connection Disconnected Error Status	The DB Connection Service cannot communicate with the DB due to a network failure or other causes.
3012 hex	DB Connection Instruction Execution Timeout	The instruction was not completed within the time specified in the *TimeOut* input variable.
3013 hex	DB Connection Service Error Stop	The instruction was executed while the DB Connection Service was stopped due to an error.
3014 hex	Data Already Spooled	This instruction cannot be executed because one or more SQL statements are already stored in the Spool memory.
3015 hex	DB Connection Service Initializing	The instruction was executed while the initialization processing of the DB Connection Service was in progress.
3016 hex	DB in Process	The instruction was executed before completion of the DB's processing for the DB Connection Instruction Execution Timeout that occurred for the previous DB_Insert, DB_Update, DB_Select, or DB_Delete instruction.

Function

This instruction is used to retrieve records from a table mapped by a DB_CreateMapping instruction into the DB Map Variable specified in the *MapVar* in-out variable.
Define the DB Map Variable as an array when you want to retrieve more than one record.
The number of records retrieved to the DB Map Variable is output to the *RecCnt* output variable. The number of records retrieved according to the retrieval conditions is output to the *SelectedCnt* output variable.

The relationship between the number of array elements in the DB Map Variable and the number of records in the *RecCnt* and *SelectedCnt* output variables is described below.

[When the number of array elements of the DB Map Variable is equal to or smaller than (≤) the number of retrieved records]
The records up to the maximum number of elements in the DB Map Variable are output.
For example, in the case where 30 records are retrieved for the DB Map Variable with 10 array elements, the records from *MapVar[0]* to *MapVar[9]* are retrieved.
The value of *RecCnt* will be 10 and the value of *SelectedCnt* will be 30 in this case.

[When the number of array elements of the DB Map Variable is bigger than (>) the number of retrieved records]
The records up to the number of elements of the retrieved records are output. For the later elements, the records are not retrieved, but the previous values are retained.
For example, in the case where 3 records are retrieved for the DB Map Variable with 10 array elements, the records from *MapVar[0]* to *MapVar[2]* are retrieved. The values of *MapVar[3]* to *MapVar[9]* do not change.
The value of *RecCnt* will be 3 and the value of *SelectedCnt* will be also 3 in this case.

The records are retrieved according to the retrieval conditions specified in the *Where* input variable (WHERE clause). The *Where* input variable is expressed as a text string.
The text string in the *Where* input variable cannot consist of NULL characters (16#00) only. In that case, the instruction is terminated due to an error.

Specify the sort conditions in the *Sort* input variable (ORDER BY clause) to sort out the retrieved records. The *Sort* input variable is expressed as a text string.
When the sort conditions are specified, the records are contained in the DB Map Variable in the order specified by the sort conditions.
When the sort conditions are not specified, the output order to the DB Map Variable depends on the specifications of the DB type to connect.

When using single quotes in the WHERE and SORT clauses, use the escape character ($’).
Refer to the *NJ/NX-series CPU Unit Software User’s Manual* (Cat. No. W501) for the escape character.
Refer to the manual of the database for the format of the WHERE and SORT clauses.
Specify the retrieval conditions by the following values in the *Where* input variable.

Example 1: Retrieve the values of the records where the value of a specific column is equal to or greater than the specified value.

Retrieve the values of records where the value of “ColumnA” (unsigned integer) is 1234 or greater.
'“ColumnA” >= 1234'
SQL statement to create: SELECT FROM TableProduct Where “ColumnA” = 1234

Example 2: Retrieve the records where the values of specific two columns are within the specified range.

Retrieve the records where the value of “ColumnA” (unsigned integer) is bigger than 1000 and the value of “ColumnB” (unsigned integer) is smaller than 2000.
'“ColumnA” > 1000 AND “ColumnB” < 2000'
SQL statement to create: SELECT FROM TableProduct Where “ColumnA” > 1000 AND “ColumnB” < 2000

Example 3: Retrieve the values of the records where the value of a specific column is equal to or greater than the value of the specified variable.

Retrieve the values of records where the value of “ColumnA” (unsigned integer) is equal to or greater than the specified variable.
Specified value: UINTVar := 1234;
Input parameter in the WHERE clause:
WhereCond_Select := CONCAT(‘$”ColumnA$” >= ‘, UINT_TO_STRING(UINTVar));
SQL statement to create: SELECT FROM TableProduct Where “ColumnA” >= 1234

Specify the sort conditions in the *Sort* input variable by the following values.

Example: Retrieve the records sorted by the values of two columns.

Sort the values of “ColumnA” in ascending order and values of “ColumnB” in descending order.
'“ColumnA” ASC, “ColumnB” DESC'
SQL statement to create: SELECT FROM TableProduct ORDER BY “ColumnA” ASC, “ColumnB” DESC

Precautions for Correct Use

- Execution of this instruction is continued until processing is completed even if the value of *Execute* changes to FALSE or the execution time exceeds the task period. The value of *Done* changes to TRUE when processing is completed. Use this to confirm normal completion of processing.
- Refer to *Using this Section* of the *NJ/NX-series Instructions Reference Manual* (Cat. No. W502) for a timing chart for *Execute, Done, Busy,* and *Error*.
- This instruction cannot be used on an event task. A compiling error will occur.
- This instruction cannot be executed without specifying the retrieval conditions.
- When no record is retrieved as the execution result of this instruction, the values of the *RecCnt* and *SelectedCnt* output variables are both 0 and the instruction is normally completed.
- Even if the number of array elements of the DB Map Variable does not match the number of retrieved records as the execution result of this instruction, the instruction is also normally completed.
- When the DB Connection Service was started in Test Mode, this instruction is normally ended without executing the SELECT operation for the DB actually. No values are stored in the DB Map Variable specified in the *MapVar* in-out variable and 0 is output to both the *RecCnt* and *SelectedCnt* output variables.
- Even if the specified number of bytes in STRING data is shorter than the table data, this instruction is normally ended.
  Example: When 12 characters are contained in a table column and data type of the corresponding member of

the DB Map Variable is STRING[11], this instruction can retrieve only up to 11 characters, but will be normally ended.

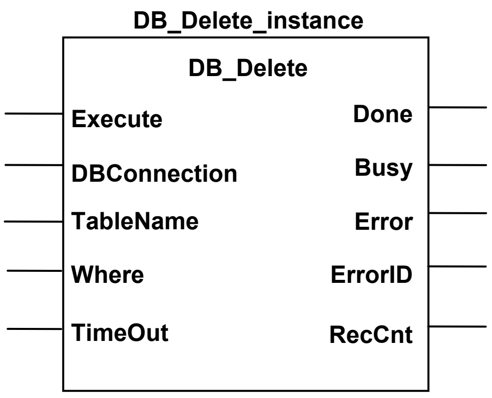
- When the error code is 300B hex (SQL Execution Error), you can get the detailed information of the SQL Execution Error by executing a DB_GetConnectionStatus instruction.
- The measurement error of timeout is +50 ms for a 100-column record when the percentage of task execution time is 50% as a guide. However, the measurement error varies according to the percentage of task execution time and the number of columns.
- When two or more DB Connection Instructions are executed for a DB Connection at the same time, the DB Connection Service executes the instructions one by one. The measurement of timeout for the second and later instructions is started when the instruction is executed by the DB Connection Service, not when the *Execute* input variable is changed to TRUE. Therefore, the time from when the *Execute* input variable is changed to TRUE to when the timeout occurs for the instruction is longer than the time set for the timeout.
- An error occurs for this instruction in the following cases. *Error* will be TRUE.
  - When the instruction was executed when the DB Connection Service was not running
  - When the instruction was executed while the initialization processing of the DB Connection Service was in progress
  - When the instruction was executed while the DB Connection Service was stopped due to an error
  - When the instruction was executed after the DB Connection Service was shut down or while the DB Connection Service was being shut down
  - When the value of the *DBConnection* input variable is invalid or the specified DB Connection is already closed
  - When the value of the *Timeout* input variable is outside the valid range
  - When the variable specified in the *MapVar* in-out variable has not been mapped by a DB_CreateMapping instruction.
  - When the executed SQL statement resulted in an error in the DB
  - When the data types cannot be converted between NJ-series Controllers and database
  - When the DB Connection Service cannot communicate with the DB due to a network failure or other causes
  - When one or more SQL statements are already stored in the Spool memory
  - When the instruction was not completed within the time specified in the *TimeOut* input variable
  - When the instruction was executed before completion of the DB's processing for the DB Connection Instruction Execution Timeout that occurred for the previous DB_Insert, DB_Update, DB_Select, or DB_Delete instruction
  - When more than 32 DB Connection Instructions were executed at the same time

Sample Programming

Refer to the sample programming that is provided for the DB_Delete instruction.

DB_Delete (Delete DB Record)

The DB_Delete instruction deletes the records that match the conditions from a specified table.

Instruction	Name	FB/FUN	Graphic expression	ST expression
DB_Delete	Delete DB Record	FB		DB_Delete_instance (Execute, DBConnection, TableName, Where, TimeOut, Done, Busy, Error, ErrorID, RecCnt);

Note The *DB_Delete_instance* is an instance of DB_Delete instruction, which is declared as a variable.

Variables

Input Variables

Name	Meaning	Data type	Valid range	Unit	Default	Description
Execute	Execute	BOOL	TRUE or FALSE	---	FALSE	Specify the execution condition.
DBConnection	DB Connection	DWORD	16#00000000 to 16#FFFFFFFF	---	16#0000 0000	Specify the DB Connection established by a DB_Connect instruction.
TableName	Table Name	STRING	Depends on the data type.	---	''	Specify a table name in the DB.
Where	Retrieval Conditions	STRING	1,986 bytes max. (including the final NULL character)*	---	''	Specify a text string that expresses retrieval conditions (WHERE clause). (‘WHERE’ is not included.)
TimeOut	Timeout	TIME	T#0s, T#0.05s to T#180s	---	T#0s	Specify the time to detect timeout. When T#0s is specified, timeout is not monitored.

* When the database is case sensitive, specify the column name as shown below.
When connecting to MySQL, enclose the column name in single-byte backquotes
Example: `ColumnA`
When connecting to other databases, enclose the column name in single-byte double quotes.
Example: “ColumnA”

Output Variables

Name	Meaning	Data type	Valid range	Unit	Description
Done	Done	BOOL	TRUE or FALSE	---	TRUE when the instruction is normally completed.
Busy	Executing	BOOL	TRUE or FALSE	---	TRUE when the instruction is being executed.
Error	Error	BOOL	TRUE or FALSE	---	TRUE when the instruction is terminated due to an error.
ErrorID	Error Code	WORD	16#0000 to 16#FFFF	---	Contains the error code when an error occurs.
RecCnt	Number of Records	DINT	0 to 2147483647	---	Contains the number of records that were deleted.

Related System-defined Variables

Name	Meaning	Data type	Description
_EIP_EtnOnlineSta	Online	BOOL	Status of the communications function of the built-in EtherNet/IP port. TRUE: Can be used. FALSE: Cannot be used.

Related Error Codes

Error code	Meaning	Description
0400 hex	Input Value Out of Range	The value of the *TimeOut* input variable is outside the valid range.
0406 hex	Illegal Data Position Specified	The *TableName* input variable is a text string consisting of NULL characters (16#00) only.
0410 hex	Text String Format Error	A space character is included in the text string specified for the *TableName* input variable.
041D hex	Too Many Instructions Executed at the Same Time	More than 32 DB Connection Instructions were executed at the same time.
3000 hex	DB Connection Service not Started	The instruction was executed when the DB Connection Service was not running.
3002 hex	DB Connection Service Shutdown or Shutting Down	The instruction was executed after the DB Connection Service was shut down or while the DB Connection Service was being shut down.
3008 hex	Invalid DB Connection	The value of the *DBConnection* input variable is invalid or the specified DB Connection is already closed.
300B hex	SQL Execution Error	The executed SQL statement resulted in an error in the DB.
300E hex	Invalid Retrieval Conditions	The *Where* input variable is a text string consisting of NULL characters (16#00) only.
3011 hex	DB Connection Disconnected Error Status	The DB Connection Service cannot communicate with the DB due to a network failure or other causes.
3012 hex	DB Connection Instruction Execution Timeout	The instruction was not completed within the time specified in the *TimeOut* input variable.
3013 hex	DB Connection Service Error Stop	The instruction was executed while the DB Connection Service was stopped due to an error.
3014 hex	Data Already Spooled	This instruction cannot be executed because one or more SQL statements are already stored in the Spool memory.
3015 hex	DB Connection Service Initializing	The instruction was executed while the initialization processing of the DB Connection Service was in progress.
3016 hex	DB in Process	The instruction was executed before completion of the DB's processing for the DB Connection Instruction Execution Timeout that occurred for the previous DB_Insert, DB_Update, DB_Select, or DB_Delete instruction.

Function

This instruction is used to delete the records that match the conditions specified in the *Where* input variable from the table specified in the *TableName* input variable.

The records to delete are retrieved according to the retrieval conditions specified in the *Where* input variable (WHERE clause). The *Where* input variable is expressed as a text string.
The text string in the *Where* input variable cannot consist of Null characters (16#00) only. In that case, the instruction is terminated due to an error.

When using single quotes in the WHERE clause, use the escape character ($').
Refer to the *NJ/NX-series CPU Unit Software User's Manual* (Cat. No. W501) for the escape character.
Refer to the manual of the database for the format of the WHERE clause.

Specify the retrieval conditions in the *Where* input variable by the following values.
Example: Delete the records where either of the values of the specified two columns is equal to the specified value.
Delete the records where the value of "ColumnA" (unsigned integer) is equal to 1000 or the value of "ColumnB" (unsigned integer) is equal to 2000
'"ColumnA" = 1000 OR "ColumnB" = 2000'
SQL statement to create: DELETE FROM TableProduct Where "ColumnA" = 1000 OR "ColumnB" = 2000

Precautions for Correct Use

- Execution of this instruction is continued until processing is completed even if the value of *Execute* changes to FALSE or the execution time exceeds the task period. The value of *Done* changes to TRUE when processing is completed. Use this to confirm normal completion of processing.
- Refer to *Using this Section* of the *NJ/NX-series Instructions Reference Manual* (Cat. No. W502) for a timing chart for *Execute, Done, Busy,* and *Error*.
- This instruction cannot be used on an event task. A compiling error will occur.
- This instruction cannot be executed without specifying the retrieval conditions.
- When the DB Connection Service was started in Test Mode, this instruction is normally ended without executing the DELETE operation for the DB actually.
- When the error code is 300B hex (SQL Execution Error), you can get the detailed information of the SQL Execution Error by executing a DB_GetConnectionStatus instruction.
- The measurement error of timeout is +50 ms for a 100-column record when the percentage of task execution time is 50% as a guide. However, the measurement error varies according to the percentage of task execution time and the number of columns.
- When two or more DB Connection Instructions are executed for a DB Connection at the same time, the DB Connection Service executes the instructions one by one. The measurement of timeout for the second and later instructions is started when the instruction is executed by the DB Connection Service, not when the *Execute* input variable is changed to TRUE. Therefore, the time from when the *Execute* input variable is changed to TRUE to when the timeout occurs for the instruction is longer than the time set for the timeout.
- An error occurs for this instruction in the following cases. *Error* will be TRUE.
  - When the instruction was executed when the DB Connection Service was not running
  - When the instruction was executed while the initialization processing of the DB Connection Service was in progress
  - When the instruction was executed while the DB Connection Service was stopped due to an error

- When the instruction was executed after the DB Connection Service was shut down or while the DB Connection Service was being shut down
- When the value of the *DBConnection* input variable is invalid or the specified DB Connection is already closed
- When the *TableName* input variable is a text string consisting of NULL characters (16#00) only
- When a space character is included in the text string specified for the *TableName* input variable.
- When the *Where* input variable is a text string consisting of NULL characters (16#00) only
- When the value of the *Timeout* input variable is outside the valid range
- When a value that is over T#180s was specified in the *TimeOut* input variable
- When the executed SQL statement resulted in an error in the DB
- When the DB Connection Service cannot communicate with the DB due to a network failure or other causes
- When one or more SQL statements are already stored in the Spool memory
- When the instruction was not completed within the time specified in the *TimeOut* input variable
- When the instruction was executed before completion of the DB's processing for the DB Connection Instruction Execution Timeout that occurred for the previous DB_Insert, DB_Update, DB_Select, or DB_Delete instruction
- When more than 32 DB Connection Instructions were executed at the same time

Sample Programming

This section gives sample programming of the following operations for Oracle database.

- Retrieve production data for the specified lot number from a DB table when the trigger variable changes to TRUE.
- Delete the records other than the latest one if more than one record was retrieved.

DB Connection Settings and Data Type Definition

The minimum settings necessary for the sample programming are shown below.

● DB Connection Settings

DB Connection name: MyDatabase1

● Structure Data Type Definition

Name		Data type
PRODUCTION_SELECT		STRUCT
	Name	STRING[256]
	LotNo	STRING[32]
	Status	STRING[8]
	ProductionDate	DATE
	FinishTime	DATE_AND_TIME

Ladder Diagram

● Main Variables

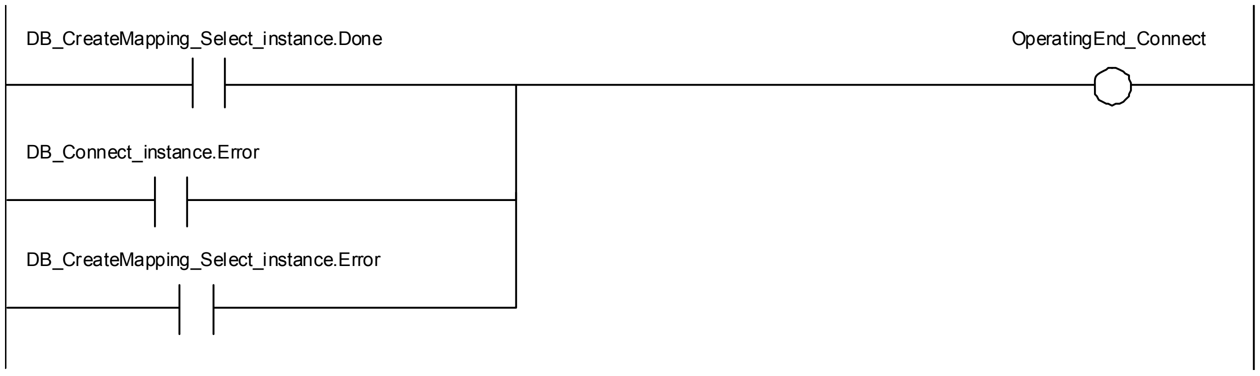
Name	Data type	Initial value	Comment
_DBC_Status	_sDBC_STATUS	---	System-defined variable that shows the status of the DB Connection Service
DB_Connect_instance	DB_Connect	---	Instance of DB_Connect instruction

Name	Data type	Initial value	Comment
MyDB1	DWORD	---	This variable is assigned to the *DBConnection* output variable from *DB_Connect_instance*.
LotNo	UINT	1234	Variable to specify the lot number for retrieving/deleting DB records
Trigger_Connect	BOOL	FALSE	Variable used as a trigger for establishing a DB Connection
RS_Connect_instance	RS	---	Instance of RS instruction
Operating_Connect	BOOL	FALSE	The DB_Connect instruction is executed when this variable is TRUE.
OperatingEnd_Connect	BOOL	FALSE	This variable changes to TRUE when the DB_Connect instruction is completed.
DB_CreateMapping_Select_instance	DB_CreateMapping	---	Instance of DB_CreateMapping instruction
MapVar_Select	ARRAY[0..9] OF PRODUCTION_SELECT		This variable is assigned to the *MapVar* input variable to *DB_CreateMapping_Select_instance*.
WhereCond_Select	STRING[256]	---	This variable is assigned to the *Where* input variable to *DB_Select_instance*.
SortCond_Select	STRING[256]	---	This variable is assigned to the *Sort* input variable to *DB_Select_instance*.
DB_Select_instance	DB_Select	---	Instance of DB_Select instruction
Trigger_Select	BOOL	FALSE	Variable used as a trigger for retrieving DB records
RS_Select_instance	RS	---	Instance of RS instruction
Operating_Select	BOOL	FALSE	The DB_Select instruction is executed when this variable is TRUE.
OperatingEnd_Select	BOOL	FALSE	This variable changes to TRUE when the DB_Select instruction is completed.
WhereCond_Delete	STRING[256]	---	This variable is assigned to the *Where* input variable to *DB_Delete_instance*.
Request_Delete	BOOL	FALSE	The DB_Delete instruction is executed when this variable is TRUE.
DB_Delete_instance	DB_Delete	---	Instance of DB_Delete instruction
RS_Delete_instance	RS	---	Instance of RS instruction
Operating_Delete	BOOL	FALSE	The DB_Delete instruction is executed when this variable is TRUE.
OperatingEnd_Delete	BOOL	FALSE	This variable changes to TRUE when the DB_Delete instruction is completed.
DB_Close_instance	DB_Close	---	Instance of DB_Close instruction
Trigger_Close	BOOL	FALSE	Variable used as a trigger for closing the DB Connection
RS_Close_instance	RS	---	Instance of RS instruction
Operating_Close	BOOL	FALSE	The DB_Close instruction is executed when this variable is TRUE.
OperatingEnd_Close	BOOL	FALSE	This variable changes to TRUE when the DB_Close instruction is completed.

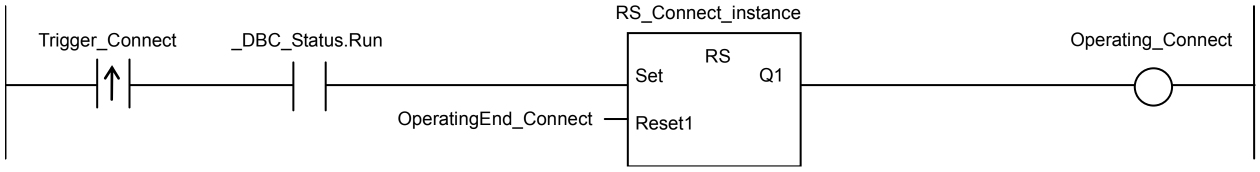
● Sample Programming

- Establish a DB Connection named MyDatabase1 and map a table with a variable.

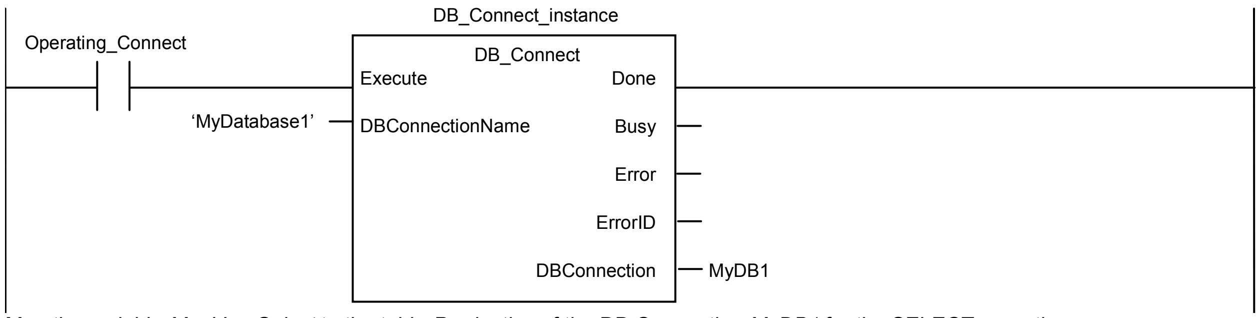
Check the completion of DB_Connect and DB_CreateMapping instructions.


DB_CreateMapping_Select_instance.Done
OperatingEnd_Connect
DB_Connect_instance.Error
DB_CreateMapping_Select_instance.Error

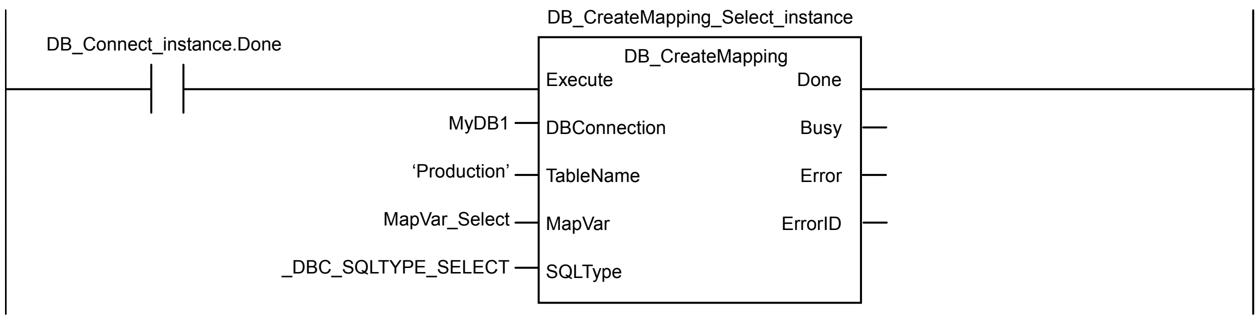
Accept the trigger for establishing the DB Connection.



Establish the DB Connection named MyDatabase1.

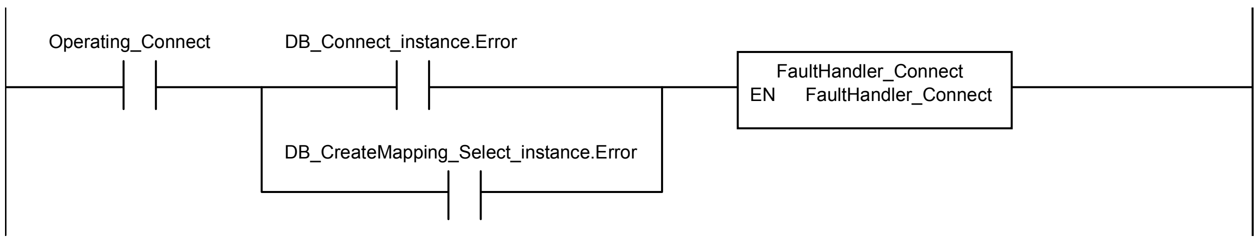


Map the variable *MapVar_Select* to the table *Production* of the DB Connection *MyDB1* for the SELECT operation.



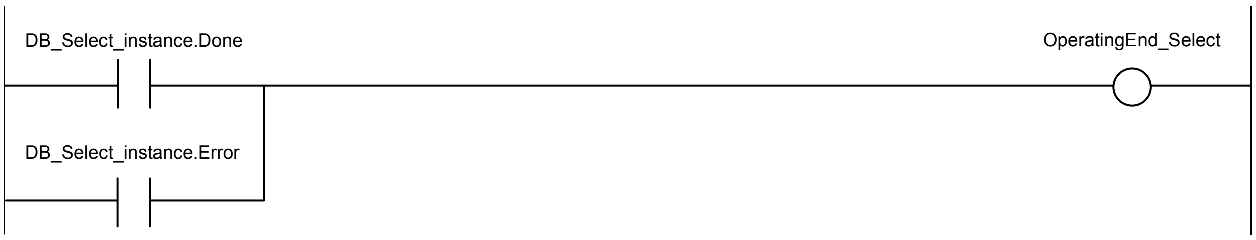
When the instruction is terminated due to an error, execute the error handler for the device (FaultHandler_Connect).

Program the FaultHandler_Connect according to the device.

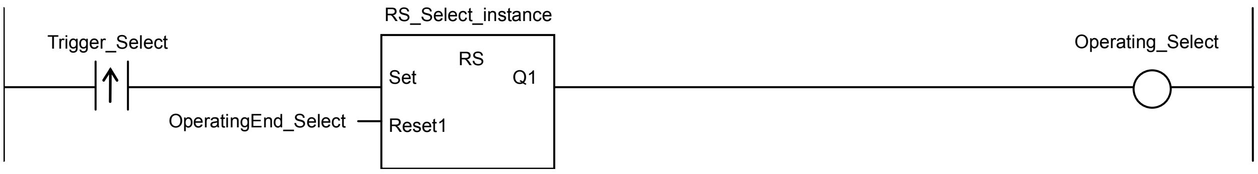

Operating_Connect
DB_Connect_instance.Error
FaultHandler_Connect
EN
FaultHandler_Connect
DB_CreateMapping_Select_instance.Error

- Retrieve records for the specified lot number from the DB Connection *MyDB1* when the variable *Trigger_Select* changes to TRUE.

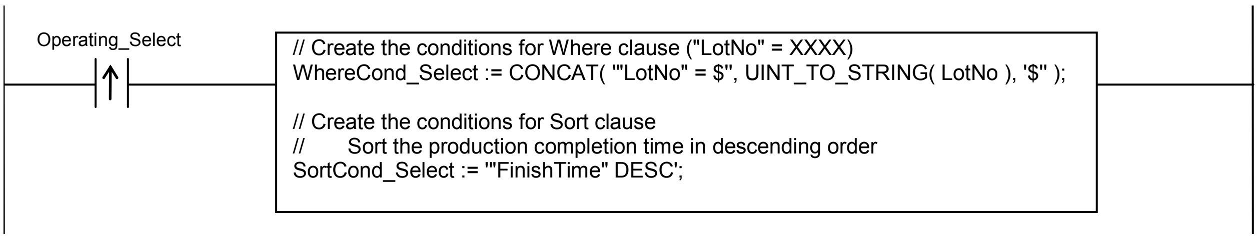
Check the completion of the DB_Select instruction.



Accept the trigger for retrieving DB records.



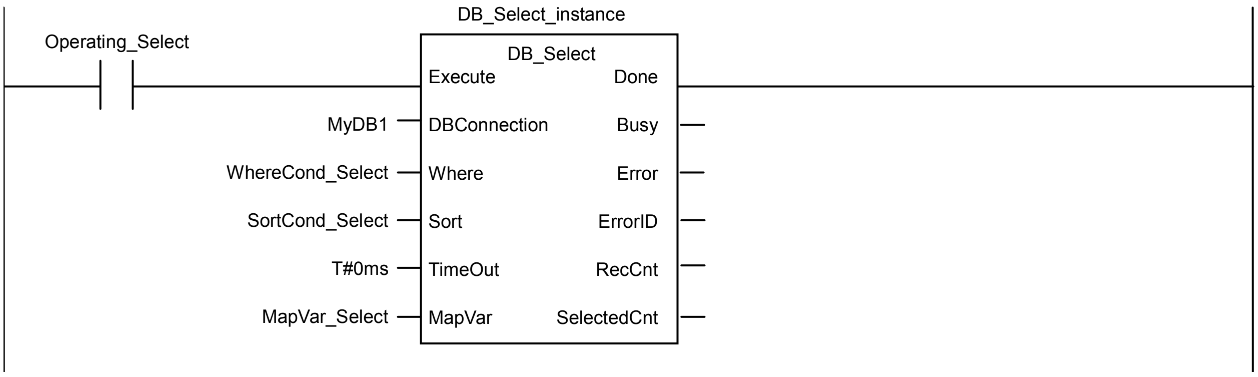
Create the conditions for the Where and Sort clauses.



```
// Create the conditions for Where clause ("LotNo" = XXXX)
WhereCond_Select := CONCAT( '"LotNo" = $'', UINT_TO_STRING( LotNo ), '$'' );

// Create the conditions for Sort clause
//      Sort the production completion time in descending order
SortCond_Select := '"FinishTime" DESC';
```

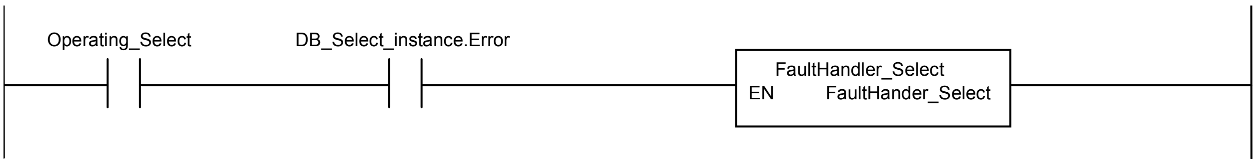
Retrieve the records from the DB Connection *MyDB1*.

Timeout is not monitored for the instruction execution.

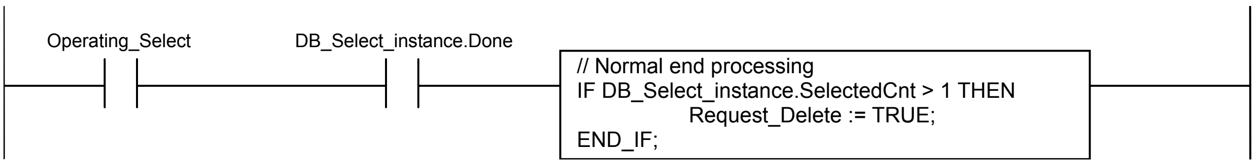


When the instruction is terminated due to an error, execute the error handler for the device (FaultHandler_Select).

Program the FaultHandler_Select according to the device.



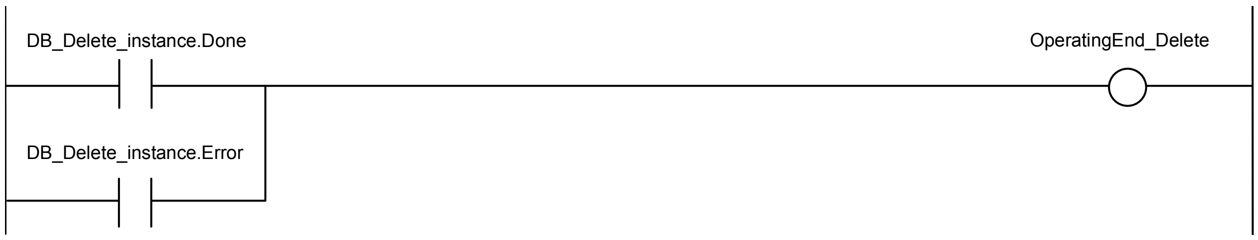
If two or more records were retrieved, delete the records other than the latest one.



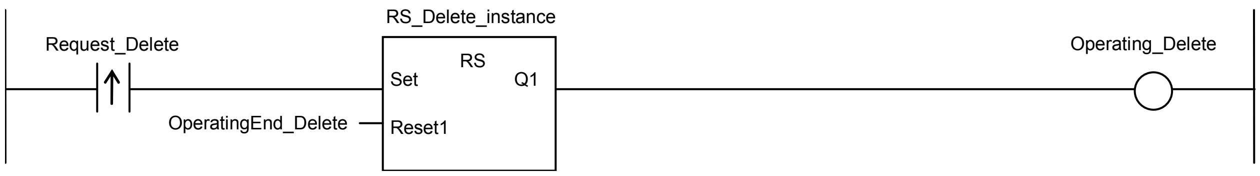
```
// Normal end processing
IF DB_Select_instance.SelectedCnt > 1 THEN
            Request_Delete := TRUE;
END_IF;
```

- Delete the records other than the latest one from the DB table

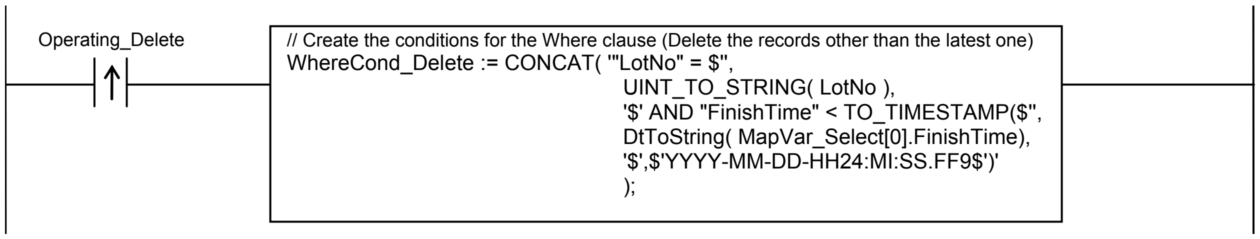
Check the completion of the DB_Delete instruction.



Accept the trigger for deleting DB records.

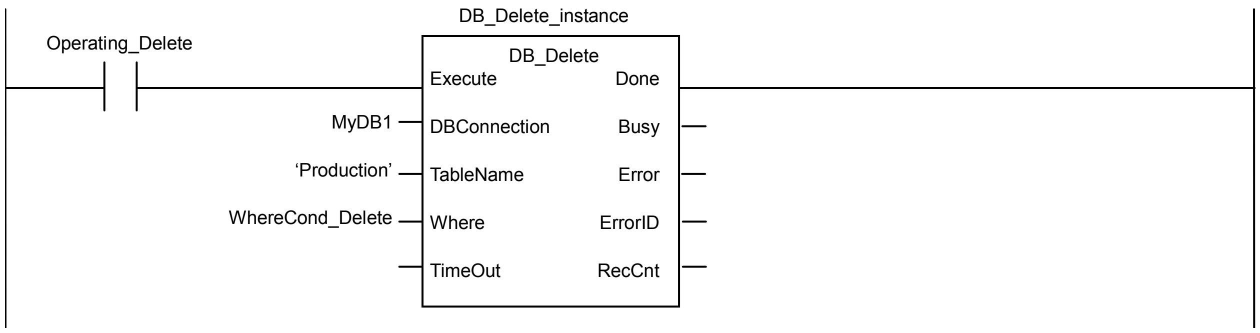


Create the conditions for the Where clause.

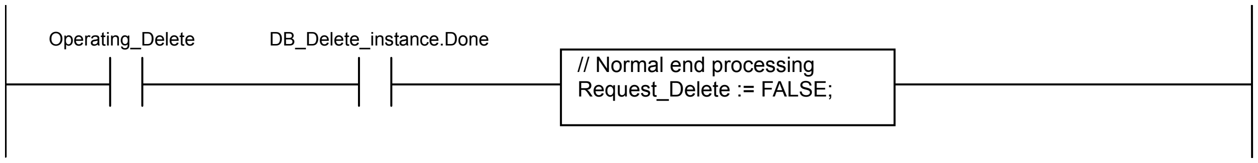


```
// Create the conditions for the Where clause (Delete the records other than the latest one)
WhereCond_Delete := CONCAT( '"LotNo" = $'',
                            UINT_TO_STRING( LotNo ),
                            '$' AND "FinishTime" < TO_TIMESTAMP($'',
                            DtToString( MapVar_Select[0].FinishTime),
                            '$',$'YYYY-MM-DD-HH24:MI:SS.FF9$')'
                            );
```

Delete records from the table *Production* of the DB Connection *MyDB1*. Timeout is not monitored for the instruction execution.

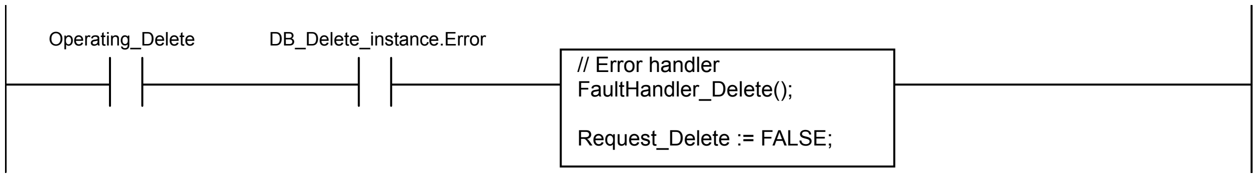


Execute the normal end processing.



```
// Normal end processing
Request_Delete := FALSE;
```

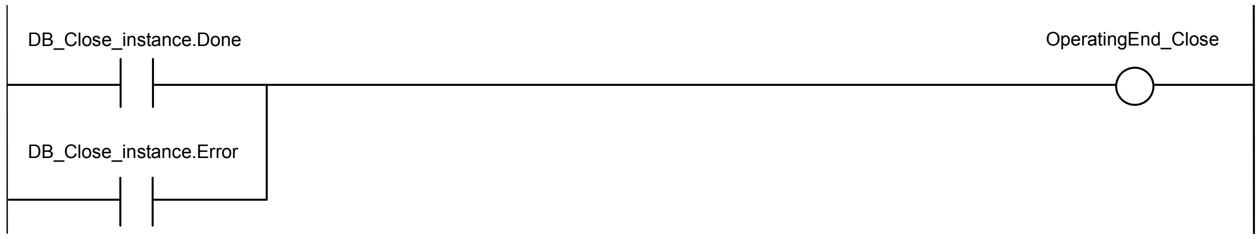
When the instruction is terminated due to an error, execute the error handler for the device (FaultHandler_Delete).
Program the FaultHandler_Delete for the device.



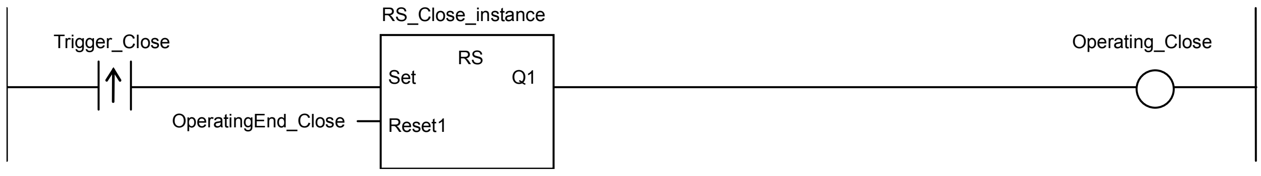
```
// Error handler
FaultHandler_Delete();

Request_Delete := FALSE;
```

- Close the DB Connection *MyDB1*.

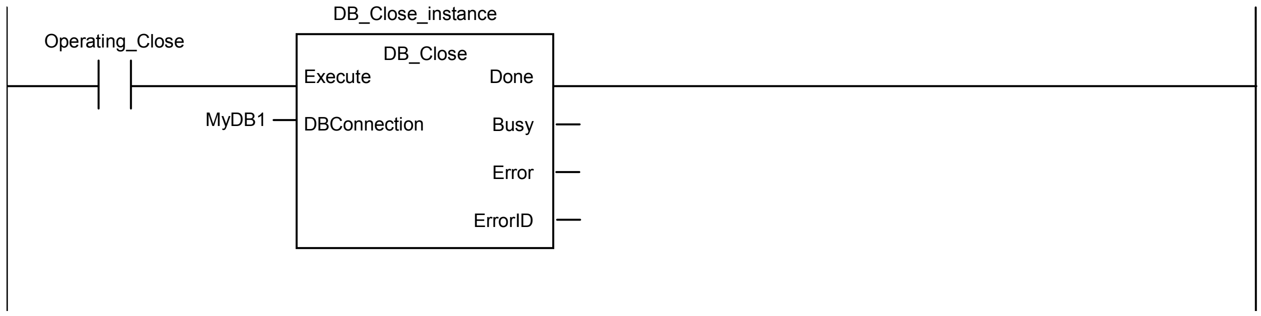
Check the completion of the DB_Close instruction.



Accept the trigger for closing the DB Connection.

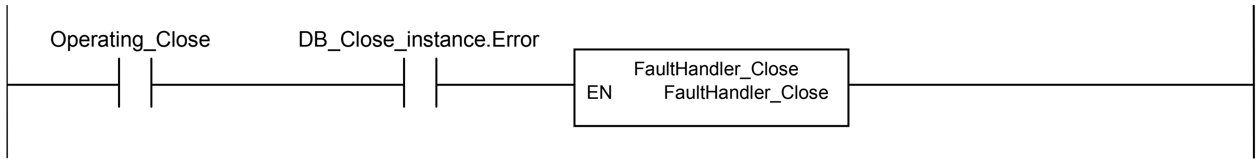


Close the DB Connection *MyDB1*.



When the instruction is terminated due to an error, execute the error handler for the device (FaultHandler_Close).

Program the FaultHandler_Close according to the device.



Structured Text (ST)

● Main Variables

Name	Data type	Initial value	Comment
_DBC_Status	_sDBC_STATUS	---	System-defined variable that shows the status of the DB Connection Service
DB_Connect_instance	DB_Connect	---	Instance of DB_Connect instruction
MyDB1	DWORD	---	Variable that is assigned to the *DBConnection* output variable from *DB_Connect_instance*
LotNo	UINT	1234	Variable to specify the lot number for retrieving/deleting DB records
Trigger_Connect	BOOL	FALSE	Variable used as a trigger for establishing a DB Connection
LastTrigger_Connect	BOOL	FALSE	Variable to retain the trigger status of the previous execution
Operating_Connect	BOOL	FALSE	The DB_Connect instruction is executed when this variable is TRUE.
OperatingStart_Connect	BOOL	FALSE	The start processing for establishing the DB Connection is executed when this variable is

Name	Data type	Initial value	Comment
			TRUE.
DB_CreateMapping_Select_instance	DB_CreateMapping	---	Instance of DB_CreateMapping instruction
MapVar_Select	ARRAY[0..99] OF PRODUCTION_SELECT	---	This variable is assigned to the *MapVar* input variable to *DB_CreateMapping_Select_instance*.
DB_Select_instance	DB_Select	---	Instance of DB_Select instruction
Trigger_Select	BOOL	FALSE	Variable used as a trigger for retrieving DB records
LastTrigger_Select	BOOL	FALSE	Variable to retain the trigger status of the previous execution
Operating_Select	BOOL	FALSE	The DB_Select instruction is executed when this variable is TRUE.
OperatingStart_Select	BOOL	FALSE	The start processing for retrieving DB records is executed when this variable is TRUE.
WhereCond_Select	STRING[256]	---	This variable is assigned to the *Where* input variable to *DB_Select_instance*.
SortCond_Select	STRING[256]	---	This variable is assigned to the *Sort* input variable to *DB_Select_instance*.
DB_Delete_instance	DB_Delete	---	Instance of DB_Delete instruction
WhereCond_Delete	STRING[256]	---	This variable is assigned to the *Where* input variable to *DB_Delete_instance*.
Request_Delete	BOOL	FALSE	The DB_Delete instruction is executed when this variable is TRUE.
LastRequest_Delete	BOOL	FALSE	Variable to retain the request status of the previous execution
Operating_Delete	BOOL	FALSE	The DB_Delete instruction is executed when this variable is TRUE.
OperatingStart_Delete	BOOL	FALSE	The start processing for deleting DB records is executed when this variable is TRUE.
DB_Close_instance	DB_Close	---	Instance of DB_Close instruction
Trigger_Close	BOOL	FALSE	Variable used as a trigger for closing the DB Connection
LastTrigger_Close	BOOL	FALSE	Variable to retain the trigger status of the previous execution
Operating_Close	BOOL	FALSE	The DB_Close instruction is executed when this variable is TRUE.
OperatingStart_Close	BOOL	FALSE	The start processing for closing the DB Connection is executed when this variable is TRUE.
Stage	INT	---	Variable that shows the status of the DB Connection

● Sample Programming

```
// ------------------------------------------------------------------------------------------------------------------------
// - Establish a DB Connection named MyDatabase1 and map a table with a variable.

// Start the sequence when the variable Trigger_Connect changes to TRUE.
IF ( (Trigger_Connect=TRUE)
    AND (LastTrigger_Connect=FALSE)
    AND (_DBC_Status.Run=TRUE) ) THEN
        OperatingStart_Connect := TRUE;
        Operating_Connect := TRUE;
END_IF;
LastTrigger_Connect:=Trigger_Connect;

// Sequence start processing
IF (OperatingStart_Connect=TRUE) THEN
    // Initialize the instances of the applicable DB Connection Instructions.
    DB_Connect_instance( Execute:=FALSE );
    DB_CreateMapping_Select_instance(
        Execute     := FALSE,
        MapVar      := MapVar_Select,
        SQLType     := _DBC_SQLTYPE_SELECT
    );

Stage := 1;
OperatingStart_Connect := FALSE;
END_IF;

// Establish the DB Connection named MyDatabese1.
// Map the variable MapVar_Select to the table Production of the DB Connection MyDB1 for the SELECT operation.
IF (Operating_Connect=TRUE) THEN
    CASE Stage OF
    1 : // Establish the DB Connection
        DB_Connect_instance(
            Execute                  := TRUE,
            DBConnectionName         := 'MyDatabase1',
            DBConnection             => MyDB1
        );

    IF (DB_Connect_instance.Done=TRUE) THEN
        Stage := INT#2; // Normal end
    END_IF;
    IF (DB_Connect_instance.Error=TRUE) THEN
        Stage := INT#99; // Error
    END_IF;

2 : // Map the DB table with the variable
```

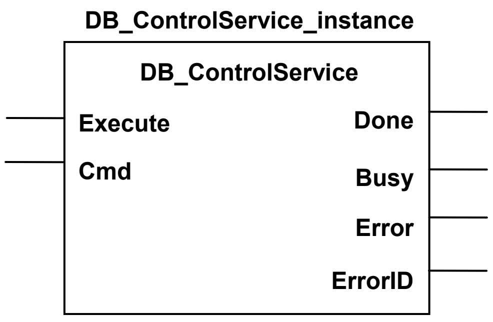
```
        DB_CreateMapping_Select_instance(
            Execute             := TRUE,
            DBConnection        := MyDB1,
            TableName           := 'Production',
            MapVar              := MapVar_Select,
            SQLType             := _DBC_SQLTYPE_SELECT
        );

        IF (DB_CreateMapping_Select_instance.Done=TRUE) THEN
            Operating_Connect:=FALSE; // Normal end
        END_IF;
        IF (DB_CreateMapping_Select_instance.Error=TRUE) THEN
            Stage := INT#99; // Error
        END_IF;

    99 :
        // Execute the error handler.
        // Program the error handler (FaultHandler_Connect) according to the device.
        FaultHandler_Connect();
        Operating_Connect := FALSE;
    END_CASE;
END_IF;

// ------------------------------------------------------------------------------------------------------------------------
// - Retrieve the records for the specified lot number from the DB Connection MyDB1.

// Start the sequence when the variable Trigger_Select changes to TRUE.
IF ( (Trigger_Select=TRUE) AND (LastTrigger_Select=FALSE) ) THEN
    OperatingStart_Select := TRUE;
    Operating_Select := TRUE;
END_IF;
LastTrigger_Select := Trigger_Select;

// Sequence start processing
IF (OperatingStart_Select=TRUE) THEN
    // Initialize the instance of the applicable DB Connection Instruction.
    DB_Select_instance( Execute:=FALSE, MapVar:=MapVar_Select );

    // Create the conditions for the Where clause ("LotNo" = XXXX).
    WhereCond_Select := CONCAT( '"LotNo" = $'', UINT_TO_STRING( LotNo ), '$'' );

    // Create the conditions for the Sort clause.
    // Sort the production completion time in descending order.
    SortCond_Select := '"FinishTime" DESC';

        OperatingStart_Select := FALSE;
    END_IF;
    // Retrieve the records from the DB Connection MyDB1. Timeout is not monitored for the instruction execution.
    IF (Operating_Select=TRUE) THEN
        // Retrieve records.
        DB_Select_instance(
            Execute             := TRUE,
            DBConnection        := MyDB1,
            Where               := WhereCond_Select,
            Sort                := SortCond_Select,
            MapVar              := MapVar_Select
        );

    IF (DB_Select_instance.Done=TRUE) THEN
        // If two or more records were retrieved, delete the older records.
        IF (DB_Select_instance.SelectedCnt > 1) THEN
            Request_Delete := TRUE;
        END_IF;
        Operating_Select:=FALSE; // Normal end
    END_IF;
    IF (DB_Select_instance.Error=TRUE) THEN
        // Error handler.
        // Program the error handler (FaultHandler_Select) according to the device.
        FaultHandler_Select();

        Operating_Select := FALSE;
    END_IF;
END_IF;

// -------------------------------------------------------------------------------------------------------------------
// - Delete the records other than the latest one from the DB table.

// Start the sequence when the variable Trigger_Delete changes to TRUE.
IF ( (Request_Delete=TRUE) AND (LastRequest_Delete=FALSE) ) THEN
    OperatingStart_Delete := TRUE;
    Operating_Delete := TRUE;
END_IF;
LastRequest_Delete := Request_Delete;

// Sequence start processing
IF (OperatingStart_Delete=TRUE) THEN
    // Initialize the instance of the applicable DB Connection Instruction.
    DB_Delete_instance( Execute:=FALSE );

    // Create the conditions for the Where clause (delete the records other than the latest one).
    WhereCond_Delete := CONCAT( '"LotNo" = $'',
                                UINT_TO_STRING( LotNo ),
                                '$' AND "FinishTime" < TO_TIMESTAMP($'',
                                DtToString( MapVar_Select[0].FinishTime),
                                '$',$'YYYY-MM-DD-HH24:MI:SS.FF9$')'
```

```
                                    );
    OperatingStart_Delete := FALSE;
END_IF;

// Delete records from the table Production of the DB Connection MyDB1. Timeout is not monitored for the instruction execution.
IF (Operating_Delete=TRUE) THEN
    // Delete the records.
    DB_Delete_instance(
        Execute         := TRUE,
        DBConnection    := MyDB1,
        TableName       := 'Production',
        Where           := WhereCond_Delete
    );

    IF (DB_Delete_instance.Done=TRUE) THEN
        Operating_Delete :=FALSE; // Normal end
        Request_Delete :=FALSE;
    END_IF;
    IF (DB_Delete_instance.Error=TRUE) THEN
        // Execute the error handler.
        // Program the error handler (FaultHandler_Delete) for the device.
        FaultHandler_Delete();

        Operating_Delete := FALSE;
        Request_Delete :=FALSE;
    END_IF;
END_IF;

// ------------------------------------------------------------------------------------------------------------------------
// - Close the DB Connection MyDB1.

// Start the sequence when the variable Trigger_Close changes to TRUE.
IF ( (Trigger_Close=TRUE) AND (LastTrigger_Close=FALSE) ) THEN
    OperatingStart_Close := TRUE;
    Operating_Close := TRUE;
END_IF;
LastTrigger_Close := Trigger_Close;

// Sequence start processing
IF (OperatingStart_Close=TRUE) THEN
    // Initialize the instance of the applicable DB Connection Instruction.
    DB_Close_instance( Execute:=FALSE );

    OperatingStart_Close := FALSE;
END_IF;

// Close the DB Connection MyDB1.
IF (Operating_Close=TRUE) THEN
```

```
    // Close the DB Connection.
    DB_Close_instance( Execute:=TRUE, DBConnection:=MyDB1 );

    IF (DB_Close_instance.Done=TRUE) THEN
        Operating_Close := FALSE; // Normal end
    END_IF;
    IF (DB_Close_instance.Error=TRUE) THEN
        // Error handler
        // Program the error handler (FaultHandler_Close) according to the device.
        FaultHandler_Close();

        Operating_Close := FALSE;
    END_IF;
END_IF;
```

DB_ControlService (Control DB Connection Service)

The DB_ControlService instruction starts/stops the DB Connection Service or starts/finishes recording to the Debug Log.

Instruction	Name	FB/FUN	Graphic expression	ST expression
DB_ControlService	Control DB Connection Service	FB	 DB_ControlService_instance DB_ControlService Execute Cmd Done Busy Error ErrorID	DB_ControlService_instance (Execute, Cmd, Done, Busy, Error, ErrorID);

Note The *DB_ControlService_instance* is an instance of DB_ControlService instruction, which is declared as a variable.

Variables

Input Variables

Name	Meaning	Data type	Valid range	Unit	Default	Description
Execute	Execute	BOOL	TRUE or FALSE	---	FALSE	Specify the execution condition.
Cmd	Command	_eDBC_CMD	_DBC_CMD_START(1): Start the service in Operation Mode _DBC_CMD_START_TEST(2): Start the service in Test Mode _DBC_CMD_STOP(3): Stop the service _DBC_CMD_DEBUGLOG_ON(4): Start recording to Debug Log _DBC_CMD_DEBUGLOG_OFF(5): Finish recording to Debug Log		0	Specify the command to execute

Output Variables

Name	Meaning	Data type	Valid range	Unit	Description
Done	Done	BOOL	TRUE or FALSE	---	TRUE when the instruction is normally completed.
Busy	Executing	BOOL	TRUE or FALSE	---	TRUE when the instruction is being executed.
Error	Error	BOOL	TRUE or FALSE	---	TRUE when the instruction is terminated due to an error.
ErrorID	Error Code	WORD	16#0000 to 16#FFFF	---	Contains the error code when an error occurs.

Related System-defined Variables

System-defined variables	Name	Data type	Valid range	Unit	Description
_DBC_Status.Idle	DB Connection Service Idle Status	BOOL	TRUE or FALSE	---	TRUE when the operation status of the DB Connection Service is Idle. Otherwise, FALSE.
_DBC_Status.Run	DB Connection Service Running Status	BOOL	TRUE or FALSE	---	TRUE when the DB Connection Service is started in Operation Mode or Test Mode. FALSE when the DB Connection Service is stopped.
_DBC_Status.Test	DB Connection Service Test Mode Status	BOOL	TRUE or FALSE	---	TRUE when the DB Connection Service is started in Test Mode. FALSE when the DB Connection Service is stopped.
_DBC_Status.Shutdown	DB Connection Service Shutdown Status	BOOL	TRUE or FALSE	---	TRUE when the operation status of the DB Connection Service is shutdown. Otherwise, FALSE.

Related Error Codes

Error code	Meaning	Description
0400 hex	Input Value Out of Range	A value that is not defined as an enumerator was specified in the *Cmd* input variable.
041D hex	Too Many Instructions Executed at the Same Time	More than 32 DB Connection Instructions were executed at the same time.
1400 hex	SD Memory Card Access Failure	This instruction was executed with *_DBC_CMD_DEBUGLOG_ON* selected in the *Cmd* input variable when the SD Memory Card was not available
1401 hex	SD Memory Card Write-protected	This instruction was executed with *_DBC_CMD_DEBUGLOG_ON* selected in the *Cmd* input variable when the SD Memory Card was write-protected.
3001 hex	DB Connection Service Run Mode Change Failed	- This instruction was executed with *_DBC_CMD_START_TEST* selected in the *Cmd* input variable while the service was running in Operation Mode. - This instruction was executed with *_DBC_CMD_START* selected in the *Cmd* input variable while the service was running in Test Mode. - Start of the DB Connection Service was commanded while the DB Connection Service was being stopped.
3002 hex	DB Connection Service Shutdown or Shutting Down	The instruction was executed after the DB Connection Service was shut down or while the DB Connection Service was being shut down.
3013 hex	DB Connection Service Error Stop	The instruction was executed while the DB Connection Service was stopped due to an error.
3015 hex	DB Connection Service Initializing	The instruction was executed while the initialization processing of the DB Connection Service was in progress.

Function

This instruction is used to start and stop the DB Connection Service, and start and finish recording to the Debug Log.

When the DB can be connected, start the DB Connection Service in Operation Mode.
When there is no DB, for example, in the course of development, start the DB Connection Service in Test Mode. In this case, the following instructions are normally completed without accessing the DB and executing the SQL statement actually.

- DB_Connect instruction,
- DB_CreateMapping instruction,
- DB_Insert instruction
- DB_Update instruction
- DB_Select instruction
- DB_Delete instruction

When the DB Connection Service is stopped, the established connections are all closed.

When recording to the debug log is started, the detailed log for each execution of DB Connection Instructions (such as transmitted SQL statements) is output to the Debug Log file in the SD Memory Card.

Precautions for Correct Use

- Execution of this instruction is continued until processing is completed even if the value of *Execute* changes to FALSE or the execution time exceeds the task period. The value of *Done* changes to TRUE when processing is completed. Use this to confirm normal completion of processing.
- Refer to *Using this Section* of the *NJ/NX-series Instructions Reference Manual* (Cat. No. W502) for a timing chart for *Execute, Done, Busy,* and *Error*.
- This instruction cannot be used on an event task. A compiling error will occur.
- When starting the DB Connection Service, confirm that the value of *_DBC_Status.Idle* is TRUE and then execute this instruction. If this instruction is executed while the DB Connection Service is being initialized, an error (DB Connection Connection Service Initializing) will occur.
- It is impossible to change the DB Connection Service from Operation Mode to Test Mode and vice versa while the DB Connection Service is running. Stop the service before changing the Run mode.
- The recording status of the Debug Log (i.e. whether or not to record the Debug Log) is held after the DB Connection Service is stopped and started again.
- Besides this instruction, recording to the Debug Log is stopped in the following cases.
    - When a DB_Shutdown instruction is executed
    - When the power supply to the CPU Unit is turned OFF
    - When the SD Memory Card is taken out
- An error occurs for this instruction in the following cases. *Error* will be TRUE.
    - When the instruction was executed while the initialization processing of the DB Connection Service was in progress
    - When the instruction was executed while the DB Connection Service was stopped due to an error
    - When the instruction was executed after the DB Connection Service was shut down or while the DB Connection Service was being shut down
    - When this instruction was executed with *_DBC_CMD_START_TEST* selected in the *Cmd* input variable while the service was running in Operation Mode
    - When this instruction was executed with *_DBC_CMD_START* selected in the *Cmd* input variable while the service was running in Test Mode

- When start of the DB Connection Service was commanded while the DB Connection Service was being stopped.
- When this instruction was executed with *_DBC_CMD_DEBUGLOG_ON* selected in the *Cmd* input variable when the SD Memory Card was not available or write-protected
- When a value that is not defined as an enumerator was specified in the *Cmd* input varaible
- When more than 32 DB Connection Instructions were executed at the same time

Sample Programming

This section gives sample programming for starting recording to the Debug Log when the trigger variable changes to TRUE and finishing the recording when another trigger variable changes to FALSE.

Ladder Diagram

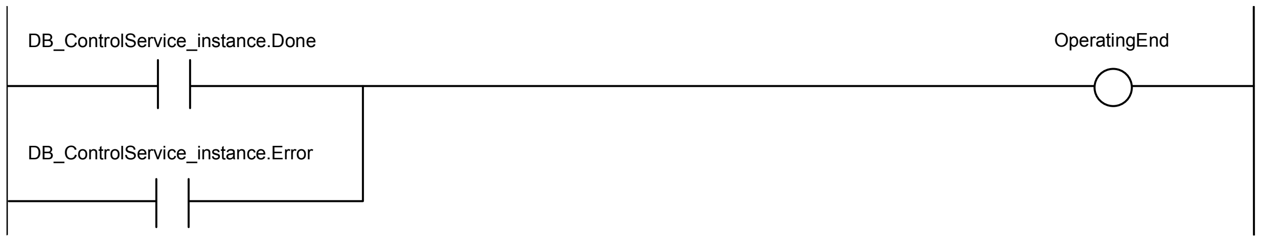
● Main Variables

Name	Data type	Initial value	Comment
DB_ControlService_instance	DB_ControlService	---	Instance of DB_ControlService instruction
LogOn	BOOL	FALSE	Variable used as a trigger for controlling the Debug Log
Operating	BOOL	FALSE	The DB_ControlService instruction is executed when this variable is TRUE.
OperatingEnd	BOOL	FALSE	This variable changes to TRUE when the DB_ControlService instruction is completed.
RS_instance	RS	---	Instance of RS instruction
MyCmd	_eDBC_CMD	---	This variable is assigned to the *Cmd* input variable to *DB_ControlService_instance*.
ControlService_OK	BOOL	FALSE	This variable changes to TRUE when the DB_ControlService instruction is completed normally.

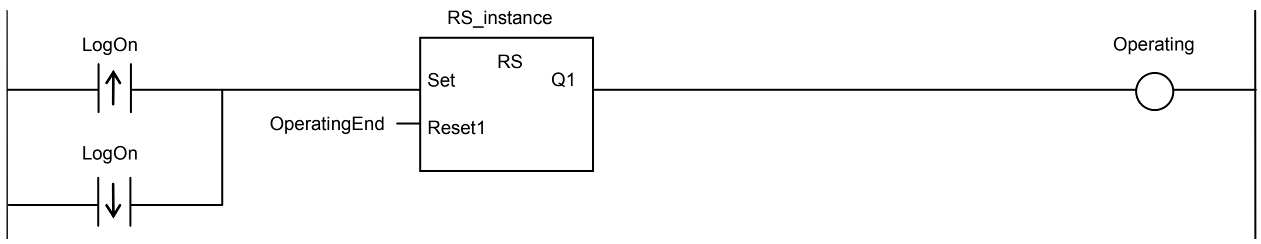
● Sample Programming

- Start recording to the Debug Log when the variable *LogOn* changes to TRUE and finish the recording when the variable *LogOn* changes to FALSE.

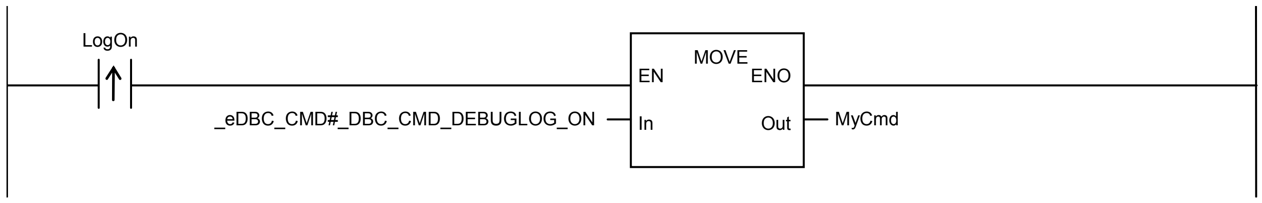
Check the completion of DB_ControlService instruction.


DB_ControlService_instance.Done
DB_ControlService_instance.Error
OperatingEnd

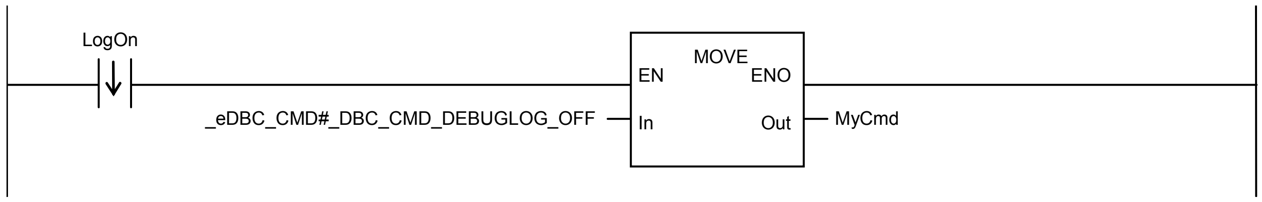
Accept the trigger for controlling the Debug Log.



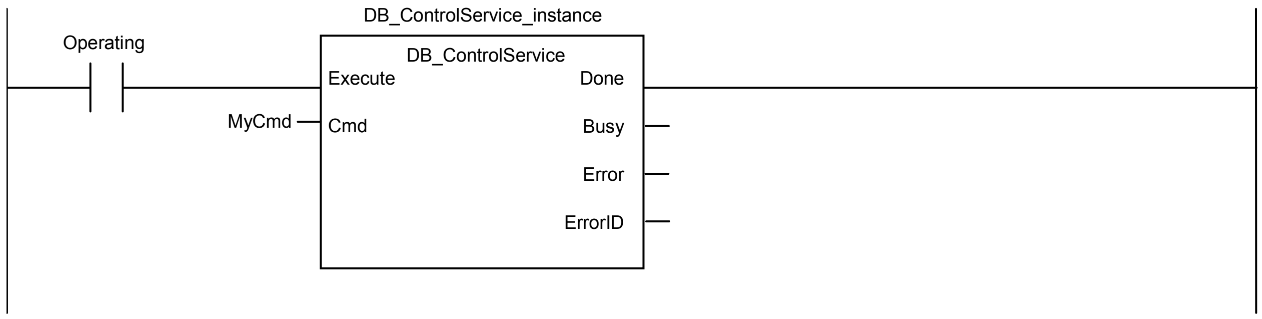
Start recording to the Debug Log.



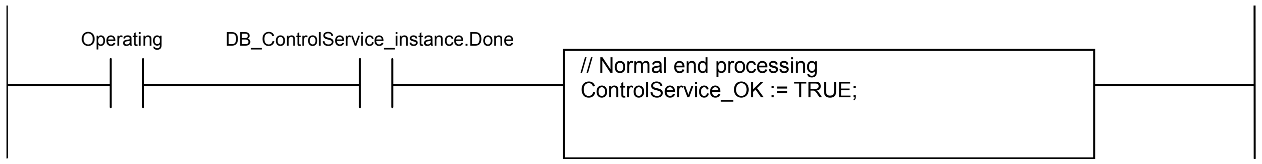
Finish recording to the Debug Log.



Command to start/finish recording to the Debug Log.

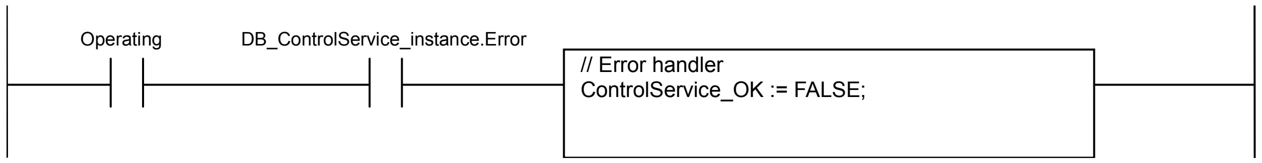


When the instruction is normally completed, change the variable *ControlService_OK* to TRUE.



```
// Normal end processing
ControlService_OK := TRUE;
```

When the instruction is terminated due to an error, change the variable *ControlService_OK* to FALSE.



```
// Error handler
ControlService_OK := FALSE;
```

Structured Text (ST)

● Main Variables

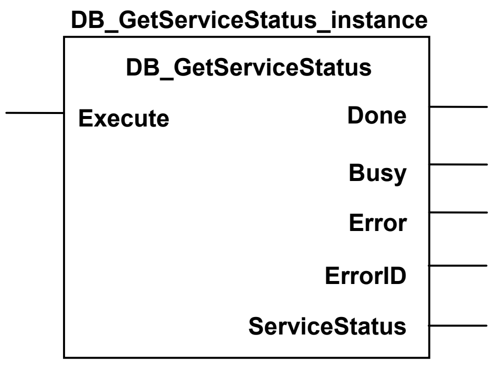
Name	Data type	Initial value	Comment
DB_ControlService_instance	DB_ControlService	---	Instance of DB_ControlService instruction
LogOn	BOOL	FALSE	Variable used as a trigger for controlling the Debug Log
LastTrigger	BOOL	FALSE	Variable to retain the trigger status of the previous execution
Operating	BOOL	FALSE	The DB_ControlService instruction is executed when this variable is TRUE.
OperatingStart	BOOL	FALSE	The initialization processing is executed when this variable is TRUE.
MyCmd	_eDBC_CMD	---	This variable is assigned to the *Cmd* input variable to *DB_ControlService_instance*.

● Sample Programming

```
(* ------------------------------------------------------------------------------------------------------------------------
     - Start recording to the Debug Log when the variable LogOn changes to TRUE.
       Finish the recording when the variable LogOn changes to FALSE.
   ------------------------------------------------------------------------------------------------------------------------ *)

// Start the sequence when the variable LogOn changes to TRUE.
IF ( (LogOn=TRUE) AND (LastTrigger=FALSE) ) THEN
   OperatingStart := TRUE;
   Operating := TRUE;
   MyCmd := _DBC_CMD_DEBUGLOG_ON;      // Start recording to the Debug Log.
ELSIF ( (LogOn=FALSE) AND (LastTrigger=TRUE) ) THEN
   OperatingStart := TRUE;
   Operating := TRUE;
   MyCmd := _DBC_CMD_DEBUGLOG_OFF; // Finish recording to the Debug Log.
END_IF;
LastTrigger := LogOn;

// Sequence start processing
IF (OperatingStart=TRUE) THEN
   // Initialize the instruction instance.
   DB_ControlService_instance( Execute:=FALSE );
   OperatingStart := FALSE;
END_IF;

// Command to start or finish recording to the Debug Log.
IF (Operating=TRUE) THEN
   // Start or finish recording to the Debug Log.
   DB_ControlService_instance(
      Execute := TRUE,
      Cmd     := MyCmd
   );

   IF (DB_ControlService_instance.Done=TRUE) THEN
      // Normal end processing
      Operating := FALSE;
   END_IF;
   IF (DB_ControlService_instance.Error=TRUE) THEN
      // Error handler.
      Operating := FALSE;
   END_IF;
END_IF;
```

DB_GetServiceStatus (Get DB Connection Service Status)

The DB_GetServiceStatus instruction gets the current status of the DB Connection Service.

Instruction	Name	FB/FUN	Graphic expression	ST expression
DB_GetService Status	Get DB Connection Service Status	FB	 DB_GetServiceStatus_instance DB_GetServiceStatus Execute Done Busy Error ErrorID ServiceStatus	DB_GetServiceStatus_instance (Execute, Done, Busy, Error, ErrorID, ServiceStatus);

Note The *DB_GetServiceStatus_instance* is an instance of DB_GetServiceStatus instruction, which is declared as a variable.

Variables

Input Variables

Name	Meaning	Data type	Valid range	Unit	Default	Description
Execute	Execute	BOOL	TRUE or FALSE	---	FALSE	Specify the execution condition.

Output Variables

Name	Meaning	Data type	Valid range	Unit	Description
Done	Done	BOOL	TRUE or FALSE	---	TRUE when the instruction is normally completed.
Busy	Executing	BOOL	TRUE or FALSE	---	TRUE when the instruction is being executed.
Error	Error	BOOL	TRUE or FALSE	---	TRUE when the instruction is terminated due to an error.
ErrorID	Error Code	WORD	16#0000 to 16#FFFF	---	Contains the error code when an error occurs.
ServiceStatus	DB Connection Service Status	_sDBC_SERVICE_STATUS	Depends on the data type.		Shows the status of the DB Connection Service.

Related Error Codes

Error code	Meaning	Description
041D hex	Too Many Instructions Executed at the Same Time	More than 32 DB Connection Instructions were executed at the same time.
3002 hex	DB Connection Service Shutdown or Shutting Down	The instruction was executed after the DB Connection Service was shut down or while the DB Connection Service was being shut down.
3013 hex	DB Connection Service Error Stop	The instruction was executed while the DB Connection Service was stopped due to an error.
3015 hex	DB Connection Service Initializing	The instruction was executed while the initialization processing of the DB Connection Service was in progress.

Function

This instruction is used to get the current status of the DB Connection Service. The current status is output to the *ServiceStatus* output variable.
Refer to the ■ *ServiceStatus* of *A-1-2 Variables Used in the DB Connection Instructions* for the status.

Precautions for Correct Use

- Execution of this instruction is continued until processing is completed even if the value of *Execute* changes to FALSE or the execution time exceeds the task period. The value of *Done* changes to TRUE when processing is completed. Use this to confirm normal completion of processing.
- Refer to *Using this Section* of the *NJ/NX-series Instructions Reference Manual* (Cat. No. W502) for a timing chart for *Execute, Done, Busy,* and *Error*.
- This instruction cannot be used on an event task. A compiling error will occur.
- An error occurs for this instruction in the following cases. *Error* will be TRUE.
    - When the instruction was executed while the initialization processing of the DB Connection Service was in progress
    - When the instruction was executed while the DB Connection Service was stopped due to an error
    - When the instruction was executed after the DB Connection Service was shut down or while the DB Connection Service was being shut down
    - When more than 32 DB Connection Instructions were executed at the same time

Sample Programming

This section gives sample programming for the following operations.

- Get the status of the DB Connection Service when the trigger variable changes to TRUE.
- Change the value of the *Warning* variable to TRUE if the number of error executions is 100 or greater.

Ladder Diagram

● Main Variables

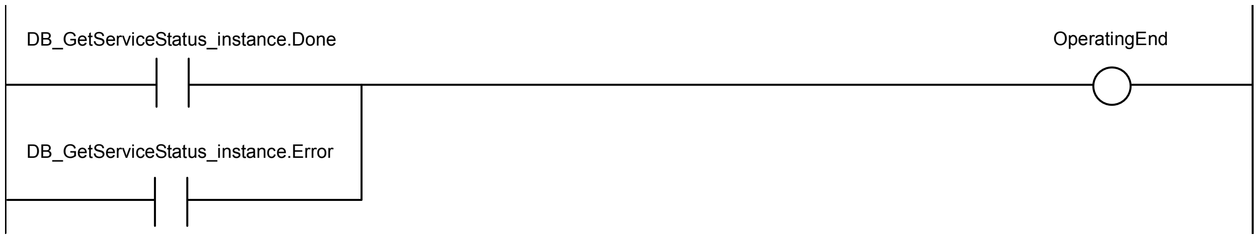
Name	Data type	Initial value	Comment
DB_GetServiceStatus_instance	DB_GetServiceStatus	---	Instance of DB_GetServiceStatus instruction
Trigger	BOOL	FALSE	Variable used as a trigger for getting the status of the DB Connection Service
Operating	BOOL	FALSE	The DB_GetServiceStatus instruction is executed when this variable is TRUE.
OperatingEnd	BOOL	FALSE	This variable changes to TRUE when the DB_GetServiceStatus instruction is completed.

Name	Data type	Initial value	Comment
RS_instance	RS	---	Instance of RS instruction
MyStatus	_sDBC_SERVICE_STATUS	---	This variable is assigned to the *ServiceStatus* input variable to *DB_GetServiceStatus_instance*.
Warning	BOOL	FALSE	This variable changes to TRUE when the number of error executions is 100 or greater.
GetServiceStatus_OK	BOOL	FALSE	This variable changes to TRUE when the DB_GetServiceStatus instruction is completed normally.

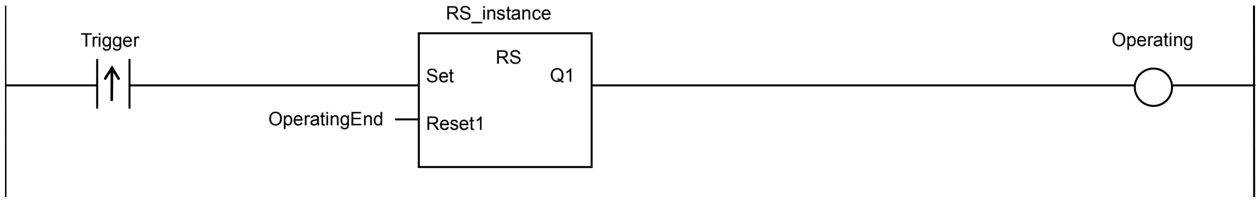
● Sample Programming

- Change the value of the variable *Warning* to TRUE when the number of error executions is 100 or greater.

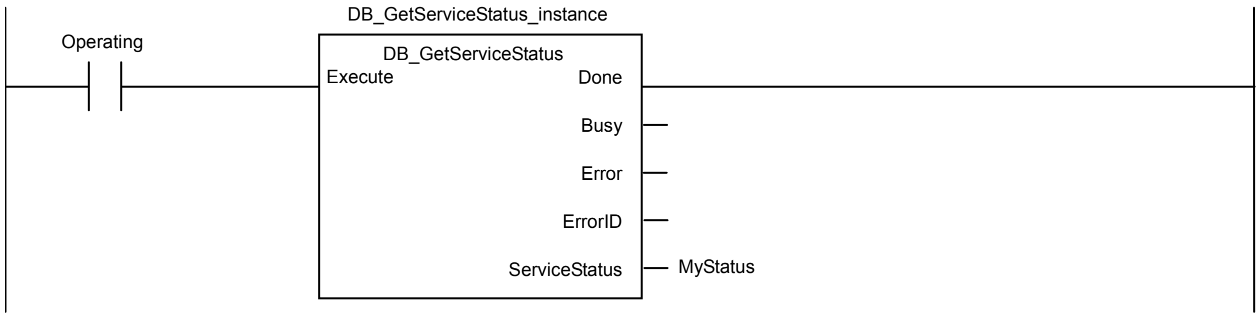
Check the completion of the DB_GetServiceStatus instruction.


DB_GetServiceStatus_instance.Done
OperatingEnd
DB_GetServiceStatus_instance.Error

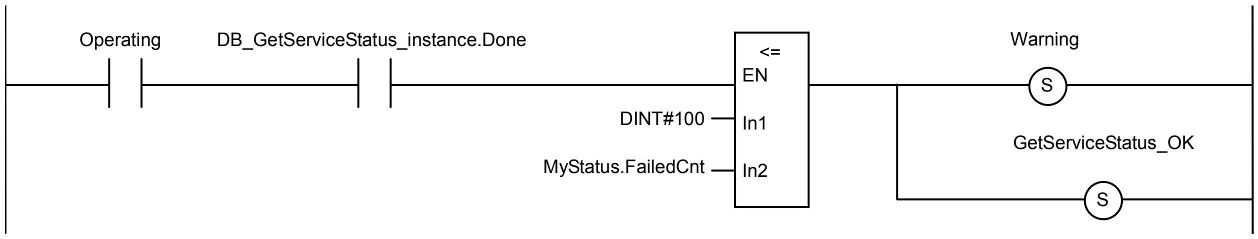
Accept the trigger.



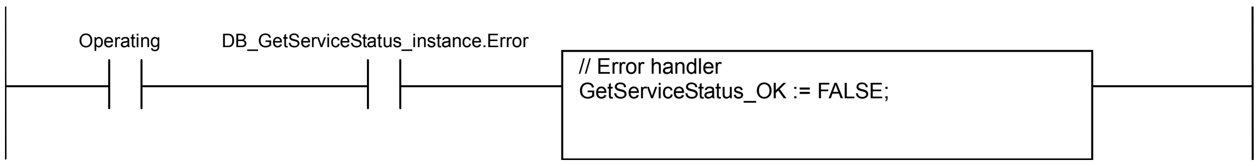
Get the status of the DB Connection Service.



When the instruction is normally completed, change the variable *Warning* to TRUE if the number of error executions is 100 or greater.



When the instruction is terminated due to an error, change the variable *Warning* to FALSE.



```
// Error handler
GetServiceStatus_OK := FALSE;
```

Structured Text (ST)

● Main Variables

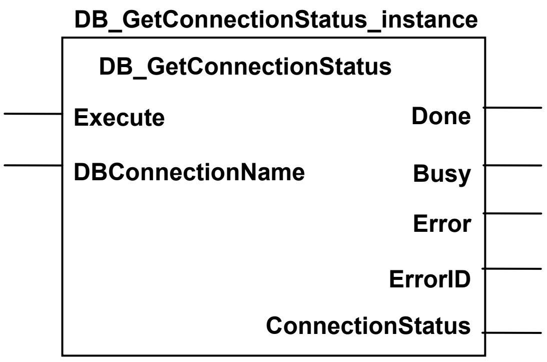
Name	Data type	Initial value	Comment
DB_GetServiceStatus_instance	DB_GetServiceStatus	---	Instance of DB_GetServiceStatus instruction
Trigger	BOOL	FALSE	Variable used as a trigger for getting the status of the DB Connection Service
LastTrigger	BOOL	FALSE	Variable to retain the trigger status of the previous execution
Operating	BOOL	FALSE	The DB_GetServiceStatus instruction is executed when this variable is TRUE.
OperatingStart	BOOL	FALSE	The initialization processing is executed when this variable is TRUE.
MyStatus	_sDBC_SERVICE_STATUS	---	This variable is assigned to the *ServiceStatus* input variable to *DB_GetServiceStatus_instance*.
Warning	BOOL	FALSE	This variable changes to TRUE when the number of error executions is 100 or greater.

● Sample Programming

```
(* ----------------------------------------------------------------------------------------------------------------------
    - Change the value of the variable Warning to TRUE when the number of SQL execution failures in all connections is 100 or greater.
   ---------------------------------------------------------------------------------------------------------------------- *)

// Start the sequence when the variable Trigger changes to TRUE.
IF ( (Trigger=TRUE) AND (LastTrigger=FALSE) ) THEN
   OperatingStart := TRUE;
   Operating := TRUE;
END_IF;
LastTrigger := Trigger;

// Sequence start processing
IF (OperatingStart=TRUE) THEN
   // Initialize the instruction instance.
   DB_GetServiceStatus_instance( Execute:=FALSE );
   OperatingStart := FALSE;
END_IF;

IF (Operating=TRUE) THEN
   // Get the status of the DB Connection Service.
   DB_GetServiceStatus_instance(
      Execute          := TRUE,
      ServiceStatus    => MyStatus
   );

   IF (DB_GetServiceStatus_instance.Done=TRUE) THEN
      // Normal end processing
      // Change the variable Warning to TRUE when the number of error executions is 100 or greater.
      IF (MyStatus.FailedCnt >= DINT#100) THEN
         Warning := TRUE;
      END_IF;
      Operating := FALSE;
   END_IF;
   IF (DB_GetServiceStatus_instance.Error=TRUE) THEN
      // Error handler
      Operating := FALSE;
   END_IF;
END_IF;
```

DB_GetConnectionStatus (Get DB Connection Status)

The DB_GetConnectionStatus instruction gets the status of a DB Connection.

Instruction	Name	FB/FUN	Graphic expression	ST expression
DB_GetConnection Status	Get DB Connection Status	FB	 DB_GetConnectionStatus_instance DB_GetConnectionStatus Execute DBConnectionName Done Busy Error ErrorID ConnectionStatus	DB_GetConnectionStatus_instance (Execute, DBConnectionName, Done, Busy, Error, ErrorID, ConnectionStatus);

Note The *DB_GetConnectionStatus_instance* is an instance of DB_GetConnectionStatus instruction, which is declared as a variable.

Variables

Input Variables

Name	Meaning	Data type	Valid range	Unit	Default	Description
Execute	Execute	BOOL	TRUE or FALSE	---	FALSE	Specify the execution condition.
DBConnectionName	DB Connection Name	STRING	17 bytes max. (including the final NULL character)	---	‘’	Specify a DB Connection name set on Sysmac Studio.

Output Variables

Name	Meaning	Data type	Valid range	Unit	Description
Done	Done	BOOL	TRUE or FALSE	---	TRUE when the instruction is normally completed.
Busy	Executing	BOOL	TRUE or FALSE	---	TRUE when the instruction is being executed.
Error	Error	BOOL	TRUE or FALSE	---	TRUE when the instruction is terminated due to an error.
ErrorID	Error Code	WORD	16#0000 to 16#FFFF	---	Contains the error code when an error occurs.
ConnectionStatus	Connection Status	_sDBC_CONNECTION _STATUS	Depends on the data type		Shows the status of the connection specified in the *DBConnectionName* input variable.

Related System-defined Variables

Name	Meaning	Data type	Description
_EIP_EtnOnlineSta	Online	BOOL	Status of the communications function of the built-in EtherNet/IP port. TRUE: Can be used. FALSE: Cannot be used.

Related Error Codes

Error code	Meaning	Description
0406 hex	Illegal Data Position Specified	The *DBConnectionName* input variable is a text string consisting of NULL characters (16#00) only.
0410 hex	Text String Format Error	• A space character is included in the text string specified for the *DBConnectionName* input variable. • The *DBConnectionName* input variable does not end in NULL.
041D hex	Too Many Instructions Executed at the Same Time	More than 32 DB Connection Instructions were executed at the same time.
3000 hex	DB Connection Service not Started	The instruction was executed when the DB Connection Service was not running.
3002 hex	DB Connection Service Shutdown or Shutting Down	The instruction was executed after the DB Connection Service was shut down or while the DB Connection Service was being shut down.
3003 hex	Invalid DB Connection Name	The DB Connection name specified in the *DBConnectionName* input variable is not set in any DB Connection Settings.
3013 hex	DB Connection Service Error Stop	The instruction was executed while the DB Connection Service was stopped due to an error.
3015 hex	DB Connection Service Initializing	The instruction was executed while the initialization processing of the DB Connection Service was in progress.

Function

This instruction is used to get the status of the DB Connection specified in the *DBConnection* input variable. The current status is output to the *ConnectionStatus* output variable.
Refer to the ■ *ServiceStatus* of *A-1-2 Variables Used in the DB Connection Instructions* for the status.
Refer to *Section B-2-3 How to Measure DB Response Time* for the measurement of the DB response time.

Precautions for Correct Use

- Execution of this instruction is continued until processing is completed even if the value of *Execute* changes to FALSE or the execution time exceeds the task period. The value of *Done* changes to TRUE when processing is completed. Use this to confirm normal completion of processing.
- Refer to *Using this Section* of the *NJ/NX-series Instructions Reference Manual* (Cat. No. W502) for a timing chart for *Execute, Done, Busy,* and *Error*.
- This instruction cannot be used on an event task. A compiling error will occur.
- If you execute this instruction before completion of a DB_Connect instruction and confirm that the connection status of the DB Connection is *Connected,* an instruction error (Invalid DB Connection) may occur when you execute the next DB Connection Instruction. When you use the *DBConnection* output variable from the DB_Connect instruction, confirm that the *Done* output variable of the DB_Connect instruction is TRUE or the value of the *DBConnection* output variable is not 16#00000000 before executing the DB Connection Instruction.
- An error occurs for this instruction in the following cases. *Error* will be TRUE.
    - When the instruction was executed when the DB Connection Service was not running
    - When the instruction was executed while the initialization processing of the DB Connection Service was in progress

- When the instruction was executed while the DB Connection Service was stopped due to an error
- When the instruction was executed after the DB Connection Service was shut down or while the DB Connection Service was being shut down
- When the DB Connection name specified in the *DBConnectionName* input variable is not set in any DB Connection Settings
- When the *DBConnectionName* input variable is a text string consisting of NULL characters (16#00) only
- When a space character is included in the text string specified for the *DBConnectionName* input variable
- When the *DBConnectionName* input variable does not end in NULL.
- When more than 32 DB Connection Instructions were executed at the same time

Sample Programming

This section gives sample programming for the following operations.

- Get the status of the DB Connection when the trigger variable changes to TRUE.
- Change the value of the *Warning* variable to TRUE when the spool usage has exceeded 80%.

Ladder Diagram

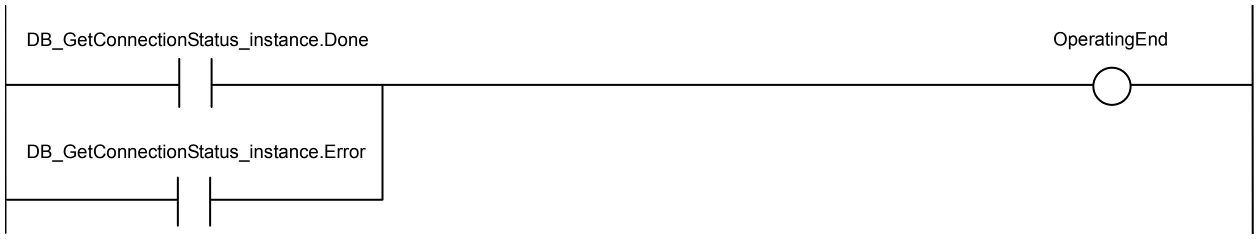
● Main Variables

Name	Data type	Initial value	Comment
DB_GetConnectionStatus_instance	DB_GetConnectionStatus	---	Instance of DB_GetConnectionStatus instruction
Trigger	BOOL	FALSE	Variable used as a trigger for getting the status of the DB Connection
Operating	BOOL	FALSE	The DB_GetConnectionStatus instruction is executed when this variable is TRUE.
OperatingEnd	BOOL	FALSE	This variable changes to TRUE when the DB_GetConnectionStatus instruction is completed.
RS_instance	RS	---	Instance of RS instruction
MyStatus	_sDBC_CONNECTION_STATUS	---	This variable is assigned to the *ConnectionStatus* output variable from *DB_GetConnectionStatus_instance*.
Warning	BOOL	FALSE	This variable changes to TRUE when the Spool usage has exceeded 80%.
GetConnectionStatus_OK	BOOL	FALSE	This variable changes to TRUE when the DB_GetConnectionStatus instruction is completed normally.

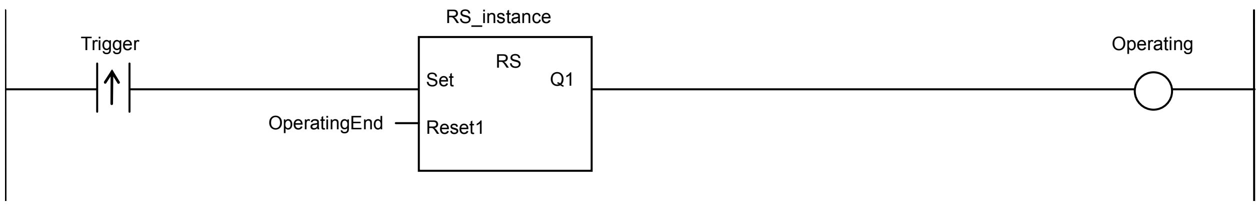
● Sample Programming

- Change the variable *Warning* to TRUE when the Spool usage of the DB Connection named MyDatabase1 has exceeded 80%.

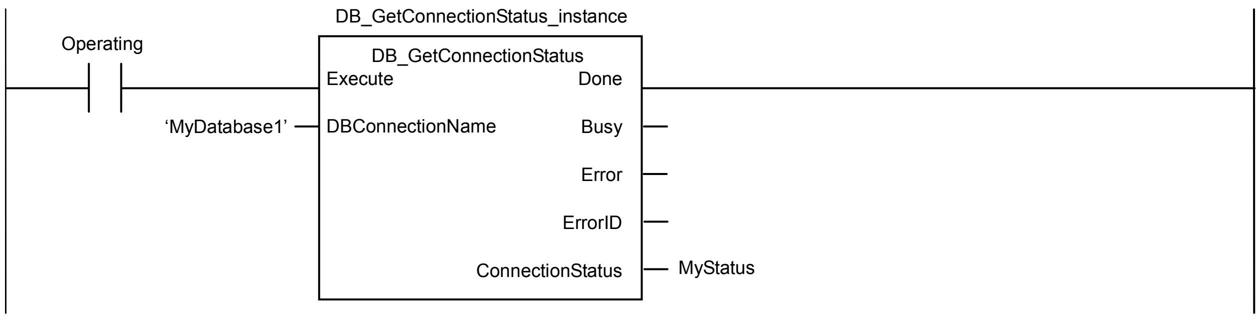
Check the completion of the DB_GetConnectionStatus instruction.


DB_GetConnectionStatus_instance.Done
OperatingEnd
DB_GetConnectionStatus_instance.Error

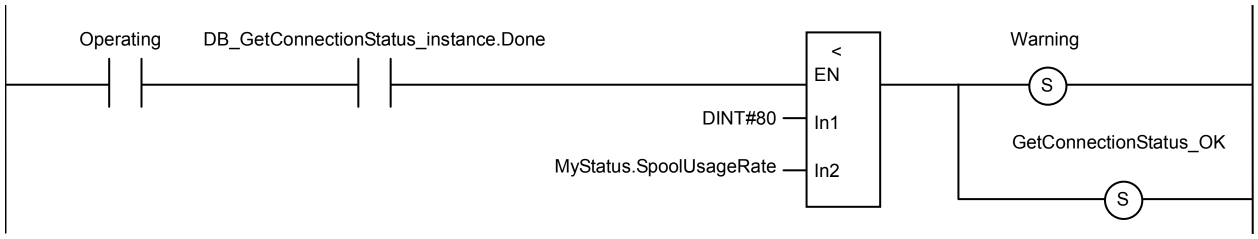
Accept the trigger.



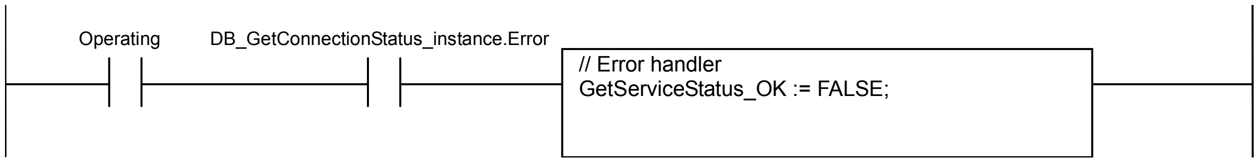
Get the status of the DB Connection.



When the instruction is normally completed, change the value of the variable *Warning* to TRUE if the Spool usage has exceeded 80%.



When the instruction is terminated due to an error, change the variable *Warning* to FALSE.



```
// Error handler
GetServiceStatus_OK := FALSE;
```

Structured Text (ST)

● Main Variables

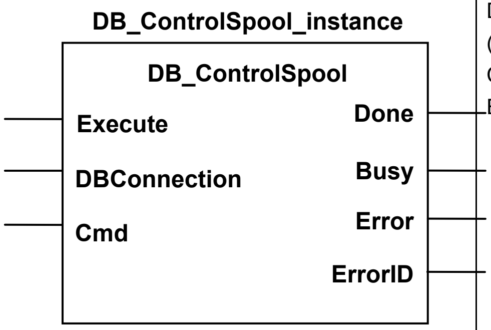
Name	Data type	Initial value	Comment
DB_GetConnectionStatus_instance	DB_GetConnectionStatus	---	Instance of DB_GetConnectionStatus instruction
Trigger	BOOL	FALSE	Variable used as a trigger for getting the status of the DB Connection
LastTrigger	BOOL	FALSE	Variable to retain the trigger status of the previous execution
Operating	BOOL	FALSE	The DB_GetConnectionStatus instruction is executed when this variable is TRUE.
OperatingStart	BOOL	FALSE	The initialization processing is executed when this variable is TRUE.
MyStatus	_sDBC_CONNECTION_STATUS	---	This variable is assigned to the *ConnectionStatus* output variable from *DB_GetConnectionStatus_instance*.
Warning	BOOL	FALSE	This variable changes to TRUE when the Spool usage has exceeded 80%.

● Sample Programming

```
(* ----------------------------------------------------------------------------------------------------------------------
      - Change the variable Warning to TRUE when the Spool usage of the DB Connection named MyDababase1 has exceeded 80%.
      ---------------------------------------------------------------------------------------------------------------------- *)
// Start the sequence when the variable Trigger changes to TRUE.
IF ( (Trigger=TRUE) AND (LastTrigger=FALSE) ) THEN
    OperatingStart := TRUE;
    Operating := TRUE;
END_IF;
LastTrigger := Trigger;

// Sequence start processing
IF (OperatingStart=TRUE) THEN
    // Initialize the instruction instance.
    DB_GetConnectionStatus_instance( Execute:=FALSE );
    OperatingStart := FALSE;
END_IF;

IF (Operating=TRUE) THEN
    // Get the status of the DB Connection.
    DB_GetConnectionStatus_instance(
        Execute               := TRUE,
        DBConnectionName      := 'MyDatabase1',
        ConnectionStatus      => MyStatus
    );

    IF (DB_GetConnectionStatus_instance.Done=TRUE) THEN
        // Normal end processing
        // Change the variable Warning to TRUE when the Spool usage has exceeded 80%.
        IF (MyStatus.SpoolUsageRate > SINT#80) THEN
            Warning := TRUE;
        END_IF;
        Operating := FALSE;
    END_IF;
    IF (DB_GetConnectionStatus_instance.Error=TRUE) THEN
        // Error handler
        Operating := FALSE;
    END_IF;
END_IF;
```

DB_ControlSpool (Resend/Clear Spool Data)

The DB_ControlSpool instruction resends or clears the SQL statements spooled by DB_Insert (Insert DB Record) and DB_Update (Update DB Record) instructions.

Instruction	Name	FB/FUN	Graphic expression	ST expression
DB_ControlSpool	Resend/Clear Spool Data	FB	 DB_ControlSpool_instance DB_ControlSpool Execute DBConnection Cmd Done Busy Error ErrorID	DB_ControlSpool_instance (Execute, DBConnection, Cmd, Done, Busy, Error, ErrorID);

Note The *DB_ControlSpool_instance* is an instance of DB_ControlSpool instruction, which is declared as a variable.

Variables

Input Variables

Name	Meaning	Data type	Valid range	Unit	Default	Description
Execute	Execute	BOOL	TRUE or FALSE	---	FALSE	Specify the execution condition.
DBConnection	DB connection	DWORD	16#00000000 to 16#FFFFFFFF	---	---	Specify the DB connection established by a DB_Connect instruction.
Cmd	Command	_eDBC_SPOOL _CMD	_DBC_SPOOL_CLEAR(1): Clear _DBC_SPOOL_RESEND(2): Resend		0	Specify the command to execute

Output Variables

Name	Meaning	Data type	Valid range	Unit	Description
Done	Done	BOOL	TRUE or FALSE	---	TRUE when the instruction is normally completed.
Busy	Executing	BOOL	TRUE or FALSE	---	TRUE when the instruction is being executed.
Error	Error	BOOL	TRUE or FALSE	---	TRUE when the instruction is terminated due to an error.
ErrorID	Error Code	WORD	16#0000 to 16#FFFF	---	Contains the error code when an error occurs.

Related System-defined Variables

Name	Meaning	Data type	Description
_EIP_EtnOnlineSta	Online	BOOL	Status of the communications function of the built-in EtherNet/IP port. TRUE: Can be used. FALSE: Cannot be used.

Related Error Codes

Error code	Meaning	Description
0400 hex	Input Value Out of Range	A value that is not defined as an enumerator was specified in the *Cmd* input variable.
041D hex	Too Many Instructions Executed at the Same Time	More than 32 DB Connection Instructions were executed at the same time.
3000 hex	DB Connection Service not Started	The Resend Spool Data operation was executed by this instruction when the DB Connection Service was not running.
3002 hex	DB Connection Service Shutdown or Shutting Down	The instruction was executed after the DB Connection Service was shut down or while the DB Connection Service was being shut down.
3008 hex	Invalid DB Connection	The value of the *DBConnection* input variable is invalid or the specified DB Connection is already closed.
300B hex	SQL Execution Error	The executed SQL statement resulted in an error in the DB.
3011 hex	DB Connection Disconnected Error Status	The DB Connection Service cannot communicate with the DB due to a network failure or other causes.
3013 hex	DB Connection Service Error Stop	The instruction was executed while the DB Connection Service was stopped due to an error.
3015 hex	DB Connection Service Initializing	The instruction was executed while the initialization processing of the DB Connection Service was in progress.

Function

This instruction is used to resend or clear the SQL statements stored in the Spool memory for the DB Connection specified in the *DBConnection* input variable.
When you select manual resend for Spool data, the SQL statements stored in the Spool memory are resent by executing this instruction.

Precautions for Correct Use

- Execution of this instruction is continued until processing is completed even if the value of *Execute* changes to FALSE or the execution time exceeds the task period. The value of *Done* changes to TRUE when processing is completed. Use this to confirm normal completion of processing.
- When you execute this instruction to resend the Spool data, this instruction just starts the Spool data resending processing. When the value of the *Done* output variable changes to TRUE, the resending processing of the SQL statements stored in the Spool memory has not been completed. Confirm the completion of resending processing by reading the number of Spool data using the DB_GetConnectionStatus instruction.
- When the Spool function is not enabled, this instruction will be completed normally without executing the resend or clear processing of the SQL statements stored in the Spool memory.
- The Clear Spool Data operation can be executed even when the DB Connection Service is not running.
- Refer to *Using this Section* of the *NJ/NX-series Instructions Reference Manual* (Cat. No. W502) for a timing chart for *Execute, Done, Busy,* and *Error*.
- This instruction cannot be used on an event task. A compiling error will occur.

- An error occurs for this instruction in the following cases. *Error* will be TRUE.
  - When the Resend Spool Data operation was executed by this instruction when the DB Connection Service was not running
  - When the instruction was executed while the initialization processing of the DB Connection Service was in progress
  - When the instruction was executed while the DB Connection Service was stopped due to an error
  - When the instruction was executed after the DB Connection Service was shut down or while the DB Connection Service was being shut down
  - When the value of the *DBConnection* input variable is invalid or the specified DB Connection is already closed
  - When a value that is not defined as an enumerator was specified in the *Cmd* input variable
  - When the executed SQL statement resulted in an error in the DB
  - When the DB Connection Service cannot communicate with the DB due to a network failure or other causes
  - When more than 32 DB Connection Instructions were executed at the same time

Sample Programming

This section gives sample programming for resending the SQL statements stored in the Spool memory if the status of the DB Connection is *Connected* when the trigger variable changes to TRUE

Ladder Diagram

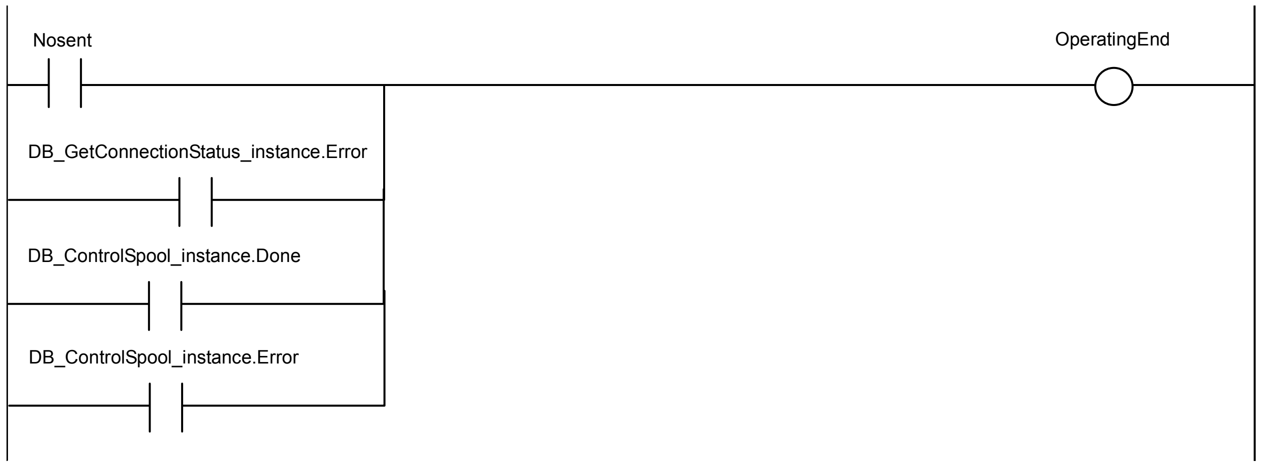
● Main Variables

Name	Data type	Initial value	Comment
DB_GetConnectionStatus_instance	DB_GetConnectionStatus	---	Instance of DB_GetConnectionStatus instruction
DB_ControlSpool_instance	DB_ControlSpool	---	Instance of DB_ControlSpool instruction
Trigger	BOOL	FALSE	Variable used as a trigger for resending the Spool data
Operating	BOOL	FALSE	When this variable is TRUE, the resending processing of Spool data is executed if necessary.
OperatingEnd	BOOL	FALSE	This variable changes to TRUE when the resending processing of Spool data is completed.
RS_instance	RS	---	Instance of RS instruction
MyStatus	_sDBC_CONNECTION_STATUS	---	This variable is assigned to the *ConnectionStatus* output variable from *DB_GetConnectionStatus_instance*.
Resend	BOOL	FALSE	This variable changes to TRUE when the status of the DB Connection is *Connected*.
Nosent	BOOL	FALSE	This variable changes to TRUE when the status of the DB Connection is not *Connected*.
ControlSpool_OK	BOOL	FALSE	This variable changes to TRUE when the DB_ControlSpool instruction is completed normally.

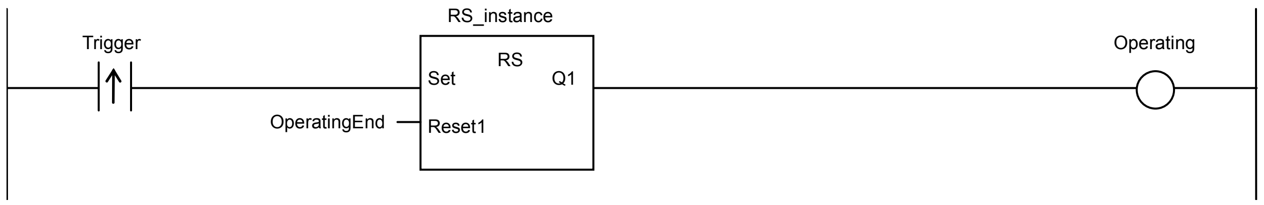
● Sample Programming

- Resend the SQL statements stored in the Spool memory when the status of the DB Connection is *Connected*.

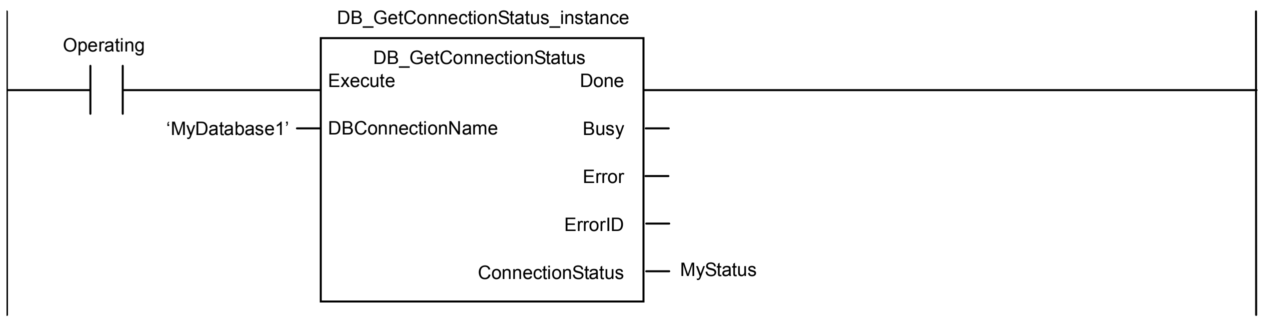
Check the completion of the instruction.


Nosent
OperatingEnd
DB_GetConnectionStatus_instance.Error
DB_ControlSpool_instance.Done
DB_ControlSpool_instance.Error

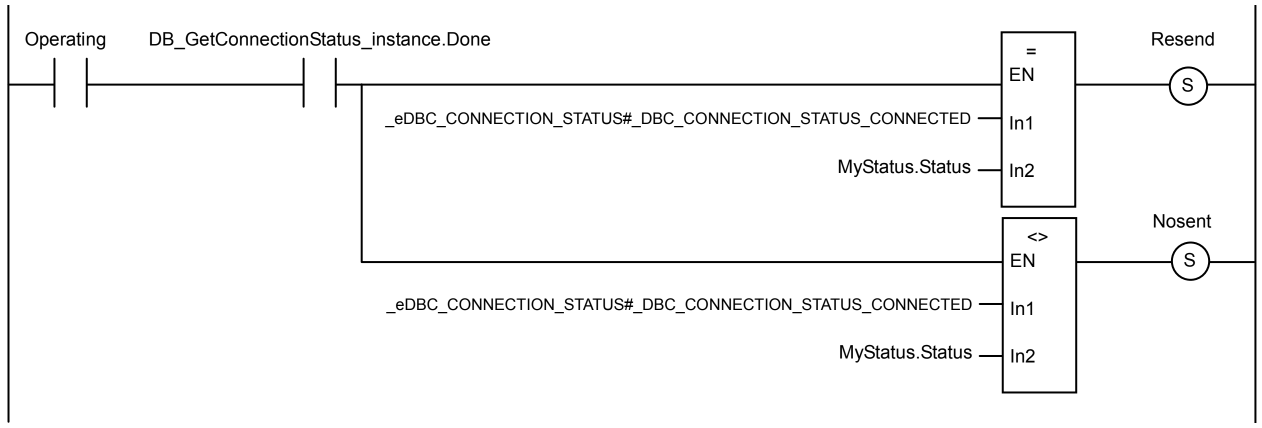
Accept the trigger.



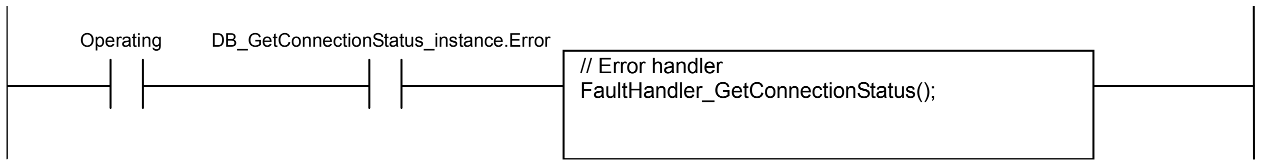
Get the status of the DB Connection.



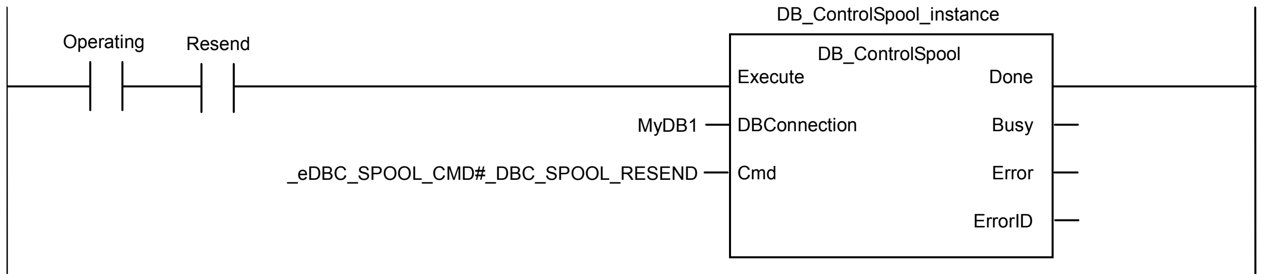
When the instruction is normally completed, change the Resend variable to TRUE if the status of the DB Connection is *Connected*.


Operating
DB_GetConnectionStatus_instance.Done
=
EN
Resend
S
_eDBC_CONNECTION_STATUS#_DBC_CONNECTION_STATUS_CONNECTED
In1
MyStatus.Status
In2
<>
EN
Nosent
S
_eDBC_CONNECTION_STATUS#_DBC_CONNECTION_STATUS_CONNECTED
In1
MyStatus.Status
In2

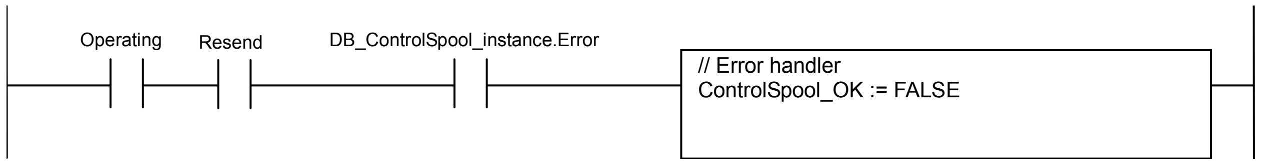
When the instruction is terminated due to an error, execute the error handler for the device (FaultHandler_GetConnectionStatus). Program the FaultHandler_GetConnectionStatus according to the device.


Operating
DB_GetConnectionStatus_instance.Error
// Error handler
FaultHandler_GetConnectionStatus();

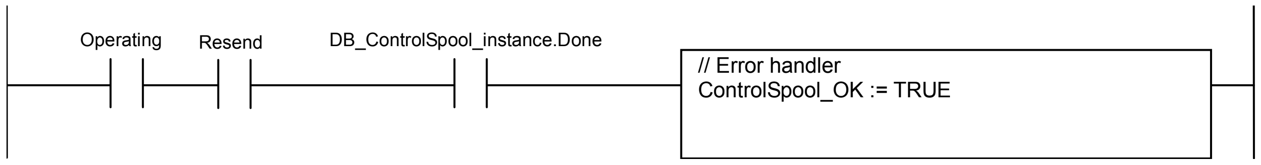
Resend the Spool data.



When the instruction is terminated due to an error, change the variable *ControlSpool_OK* to FALSE.



When the instruction is normally completed, change the variable *ControlSpool_OK* to TRUE.



Structured Text (ST)

● Main Variables

Name	Data type	Initial value	Comment
DB_GetConnectionStatus_instance	DB_GetConnectionStatus	---	Instance of DB_GetConnectionStatus instruction
DB_ControlSpool_instance	DB_ControlSpool	---	Instance of DB_ControlSpool instruction
Trigger	BOOL	FALSE	Variable used as a trigger for resending the Spool data
LastTrigger	BOOL	FALSE	Variable to retain the trigger status of the previous execution
Operating	BOOL	FALSE	When this variable is TRUE, the resending processing of Spool data is executed if necessary.
OperatingStart	BOOL	FALSE	The initialization processing is executed when this variable is TRUE.
Resend	BOOL	FALSE	This variable changes to TRUE when the status of the DB Connection is *Connected*.
MyStatus	_sDBC_CONNECTION_STATUS	---	This variable is assigned to the *ConnectionStatus* output variable from *DB_GetConnectionStatus_instance*.
MyDB1	DWORD	---	This variable is assigned to the *DBConnection* input variable to *DB_ControlSpool_instance*.

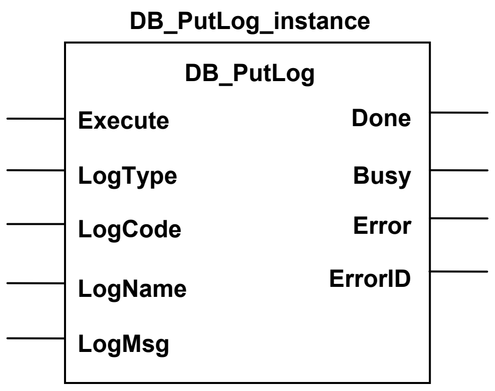
● Sample Programming

```
(* ------------------------------------------------------------------------------------------------------------------------------------------------
  - Resend the SQL statements stored in the Spool memory when the status of the DB Connection is Connected.
   ------------------------------------------------------------------------------------------------------------------------------------------- *)

// Start the sequence when the Trigger variable changes to TRUE.
IF ( (Trigger=TRUE) AND (LastTrigger=FALSE) ) THEN
    OperatingStart := TRUE;
    Operating := TRUE;
END_IF;
LastTrigger := Trigger;

// Sequence start processing
IF (OperatingStart=TRUE) THEN
    // Initialize the instruction instance.
    DB_GetConnectionStatus_instance( Execute:=FALSE );
    DB_ControlSpool_instance( Execute:=FALSE );
    OperatingStart := FALSE;
END_IF;

IF (Operating=TRUE) THEN
```

```
    // Get the status of the DB Connection.
    DB_GetConnectionStatus_instance(
        Execute                := TRUE,
        DBConnectionName       := 'MyDatabase1',
        ConnectionStatus       => MyStatus
    );

    IF (DB_GetConnectionStatus_instance.Done=TRUE) THEN
        // Normal end processing
        // Change the variable Resend to TRUE when the status of the DB Connection is Connected.
        IF (MyStatus.Status = _DBC_CONNECTION_STATUS_CONNECTED) THEN
            Resend := TRUE;
        ELSE
            Resend := FALSE;
            Operating := FALSE;
        END_IF;
    END_IF;
    IF (DB_GetConnectionStatus_instance.Error=TRUE) THEN
        // Error handler
        Operating := FALSE;
    END_IF;
END_IF;

IF ( (Operating=TRUE) AND (Resend=TRUE) ) THEN
    // Resend the Spool data.
    DB_ControlSpool_instance(
        Execute          := TRUE,
        DBConnection     := MyDB1,
        Cmd              := _DBC_SPOOL_RESEND
    );

    IF (DB_ControlSpool_instance.Done=TRUE) THEN
        // Normal end processing
        Resend := FALSE;
        Operating := FALSE;
    END_IF;
    IF (DB_ControlSpool_instance.Error=TRUE) THEN
        // Error handler
        Resend := FALSE;
        Operating := FALSE;
    END_IF;
END_IF;
```

DB_PutLog (Record Operation Log)

The DB_PutLog instruction puts a user-specified record into the Execution Log or Debug Log.

Instruction	Name	FB/FUN	Graphic expression	ST expression
DB_PutLog	Record Operation Log	FB	 DB_PutLog_instance DB_PutLog Execute LogType LogCode LogName LogMsg Done Busy Error ErrorID	DB_PutLog_instance (Execute, LogType, LogCode, LogName, LogMsg, Done, Busy, Error, ErrorID);

Note The *DB_PutLog_instance* is an instance of DB_PutLog instruction, which is declared as a variable.

Variables

Input Variables

Name	Meaning	Data type	Valid range	Unit	Default	Description
Execute	Execute	BOOL	TRUE or FALSE	---	FALSE	Specify the execution condition.
LogType	Log Type	_eDBC_LOGTYPE	_DBC_LOGTYPE_EXECUTION(1): Execution Log _DBC_LOGTYPE_DEBUG(2): Debug Log		0	Specify the type of log to output
LogCode	Log Code	INT	0 to 9999	---	0	Specify the code to record in the log.
LogName	Log Name	STRING	33 bytes max. (including the final NULL character)	---	''	Specify the name to record in the log.
LogMsg	Log Message	STRING	129 bytes max. (including the final NULL character)	---	''	Specify the message to record in the log.

Output Variables

Name	Meaning	Data type	Valid range	Unit	Description
Done	Done	BOOL	TRUE or FALSE	---	TRUE when the instruction is normally completed.
Busy	Executing	BOOL	TRUE or FALSE	---	TRUE when the instruction is being executed.
Error	Error	BOOL	TRUE or FALSE	---	TRUE when the instruction is terminated due to an error.
ErrorID	Error Code	WORD	16#0000 to 16#FFFF	---	Contains the error code when an error occurs.

Related Error Codes

Error code	Meaning	Description
0400 hex	Input Value Out of Range	A value that is not defined as an enumerator was specified in the *LogType* input variable.
041D hex	Too Many Instructions Executed at the Same Time	More than 32 DB Connection Instructions were executed at the same time.
1400 hex	SD Memory Card Access Failure	The SD Memory Card is not available.
1401 hex	SD Memory Card Write-protected	The SD Memory Card is write-protected.
3002 hex	DB Connection Service Shutdown or Shutting Down	The instruction was executed after the DB Connection Service was shut down or while the DB Connection Service was being shut down.
3010 hex	Log Code Out of Range	The value of the *LogCode* input variable is outside the valid range.
3013 hex	DB Connection Service Error Stop	The instruction was executed while the DB Connection Service was stopped due to an error.
3015 hex	DB Connection Service Initializing	The instruction was executed while the initialization processing of the DB Connection Service was in progress.
3017 hex	Operation Log Disabled	The log cannot be recorded because the specified Operation Log is disabled.

Function

This instruction is used to put a user-specified record into the Execution Log or Debug Log.
Specify whether to record in the Execution Log or Debug Log in the *LogType* input variable.
You can record any log code and log message into an Operation Log by specifying the *LogCode* and *LogMsg* input variables in the user program.

The log record format is shown below.
[Serial number]<tab>[Time]<tab>[Category]<tab>[Code]<tab>[Log name]<tab>[Result]<tab>[Details]<CR><LF>

[Serial number]:	A serial number from 0 to 65535. The value returns to 0 after 65535.
[Time]:	Time when the instruction is executed.
[Category]:	Always “USER“
[Code]:	Value of log code specified in the *LogCode* input variable Nothing is output for a text string consisting of NULL characters (16#00) only.
[Log name]:	Text string of log name specified in the *LogName* input variable Nothing is output for a text string consisting of NULL characters (16#00) only.
[Result]:	Always “0x0000“
[Details]:	Text string of log message specified in the *LogMsg* input variable

Precautions for Correct Use

- Execution of this instruction is continued until processing is completed even if the value of *Execute* changes to FALSE or the execution time exceeds the task period. The value of *Done* changes to TRUE when processing is completed. Use this to confirm normal completion of processing.
- Refer to *Using this Section* of the *NJ/NX-series Instructions Reference Manual* (Cat. No. W502) for a timing chart for *Execute, Done, Busy,* and *Error.*
- This instruction cannot be used on an event task. A compiling error will occur.
- When this instruction is executed during replacement of the SD Memory Card, the following operations are performed.

When the Execution Log is specified:
The log is recorded to the internal buffer of the CPU Unit and the instruction is completed normally.
When an SD Memory Card is inserted into the CPU Unit, the log records stored in the internal buffer are saved into the SD Memory Card.
When the Debug Log is specified:
The Debug Log cannot be recorded. The instruction is terminated due to an error (Operation Log Disabled).

・ An error occurs for this instruction in the following cases. *Error* will be TRUE.
  - When the instruction was executed while the initialization processing of the DB Connection Service was in progress
  - When the instruction was executed while the DB Connection Service was stopped due to an error
  - When the instruction was executed after the DB Connection Service was shut down or while the DB Connection Service was being shut down
  - When a value that is not defined as an enumerator was specified in the *LogType* input variable.
  - When the value of the *LogCode* input variable is outside the valid range
  - When the SD Memory Card is not available or write-protected
  - When the log cannot be recorded because the specified Operation Log is disabled
  - When more than 32 DB Connection Instructions were executed at the same time

Sample Programming

This section gives sample programming for putting the following log record into the Execution Log when the trigger variable changes to TRUE.

- Log code: 100
- Log name: Production Order
- Log message: ‘Production Start, RecipeCode=12345678’

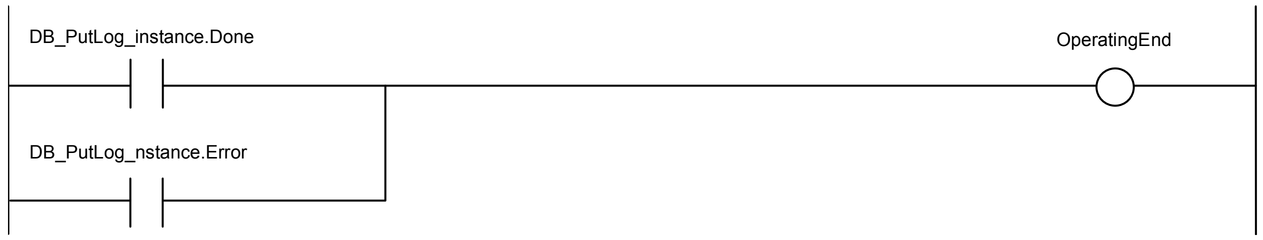
Ladder Diagram

● Main Variables

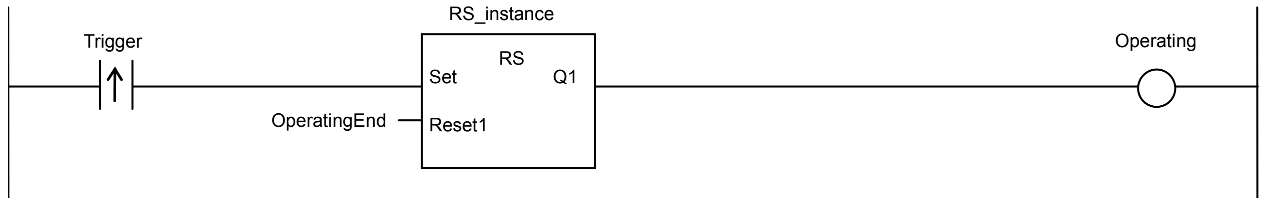
Name	Data type	Initial value	Comment
DB_PutLog_instance	DB_PutLog	---	Instance of DB_PutLog instruction
Trigger	BOOL	FALSE	Variable used as a trigger for recording the user-specified log
Operating	BOOL	FALSE	When this variable is TRUE, recording of the user-specified log is executed.
OperatingEnd	BOOL	FALSE	This variable changes to TRUE when recording of the user-specified log is completed.
RS_instance	RS	---	Instance of RS instruction
RecipeCode	UDINT	1234678	Recipe code used in the log message.
Msg	STRING[256]	‘’	Log message to record
PutLog_OK	BOOL	FALSE	This variable changes to TRUE when the DB_PutLog instruction is completed normally.

● Sample Programming

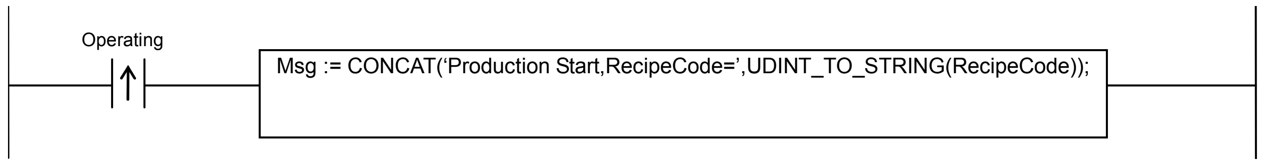
-Record the log code *100*, log name *Production Order*, and log message *Production Start, RecipeCode=12345678* into the Execution Log.
Check the completion of the DB_PutLog instruction.



Accept the trigger.

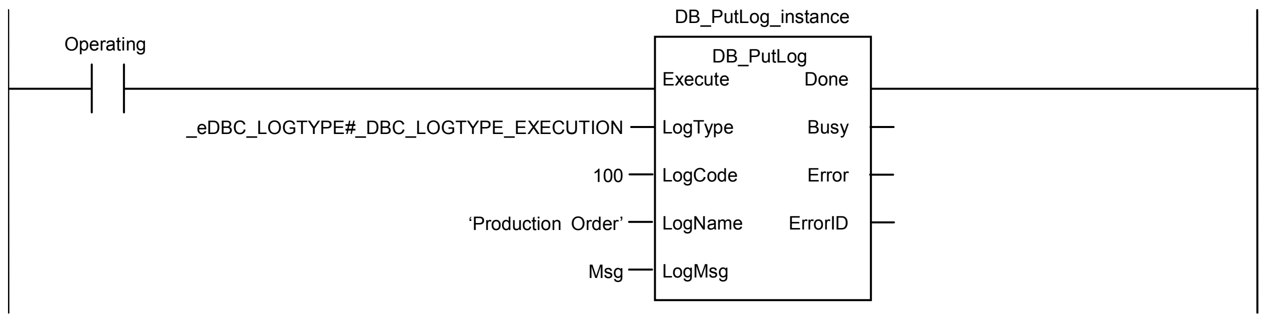


Create the log message.

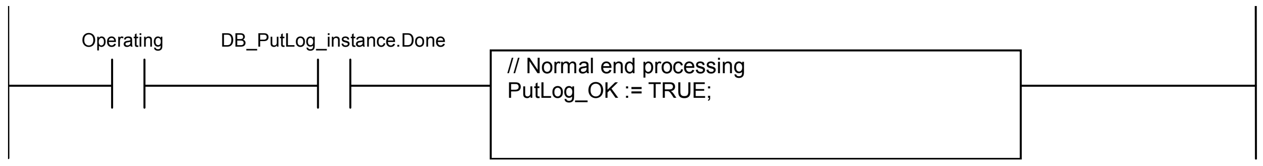


```
Msg := CONCAT('Production Start,RecipeCode=',UDINT_TO_STRING(RecipeCode));
```

Record the log message into the Execution Log.

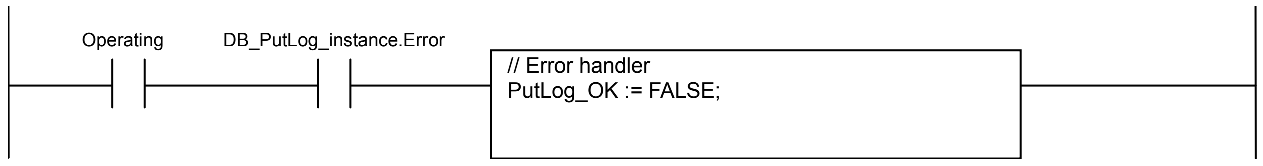


When the instruction is normally completed, change the variable *PutLog_OK* to TRUE.



```
// Normal end processing
PutLog_OK := TRUE;
```

When the instruction is terminated due to an error, change the variable *PutLog_OK* to FALSE.



```
// Error handler
PutLog_OK := FALSE;
```

Structured Text (ST)

● Main Variables

Name	Data type	Initial value	Comment
DB_PutLog_instance	DB_PutLog	---	Instance of DB_PutLog instruction
Trigger	BOOL	FALSE	Variable used as a trigger for recording the user-specified log
LastTrigger	BOOL	FALSE	Variable to retain the trigger status of the previous execution
Operating	BOOL	FALSE	When this variable is TRUE, recording of the user-specified log is executed.
OperatingStart	BOOL	FALSE	The initialization processing is executed when this variable is TRUE.
RecipeCode	UDINT	1234678	Recipe code used in the log message.
Msg	STRING[256]	''	Log message to record

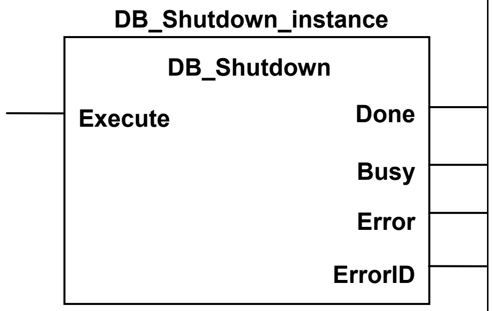
● Sample Programming

```
(* ------------------------------------------------------------------------------------------------------------------------------------------------------------
   - Record the log code 100, log name Production Order, and log message Production Start, RecipeCode=12345678 into the Execution Log.
   ------------------------------------------------------------------------------------------------------------------------------------------------------------ *)

// Start the sequence when the variable Trigger changes to TRUE.
IF ( (Trigger=TRUE) AND (LastTrigger=FALSE) ) THEN
   OperatingStart := TRUE;
   Operating := TRUE;
END_IF;
LastTrigger := Trigger;

// Sequence start processing
IF (OperatingStart=TRUE) THEN
   // Initialize the instruction instance.
   DB_PutLog_instance( Execute:=FALSE );
   // Create the log message.
   Msg := CONCAT('Production Start,RecipeCode=',UDINT_TO_STRING(RecipeCode));

   OperatingStart := FALSE;
END_IF;

IF (Operating=TRUE) THEN
   // Record the log message into the Execution Log.
   DB_PutLog_instance(
      Execute     := TRUE,
      LogType     := _DBC_LOGTYPE_EXECUTION,
      LogCode     := 100,
      LogName     := 'Production Order',
      LogMsg      := Msg );

   IF (DB_PutLog_instance.Done=TRUE) THEN
      // Normal end processing
      Operating := FALSE;
```

```
    END_IF;
    IF (DB_PutLog_instance.Error=TRUE) THEN
        // Error handler
        Operating := FALSE;
    END_IF;
END_IF;
```

DB_Shutdown (Shutdown DB Connection Service)

The DB_Shutdown instruction shuts down the DB Connection Service so as to prevent losing the Operation Log data.

Instruction	Name	FB/FUN	Graphic expression	ST expression
DB_Shutdown	Shutdown DB Connection Service	FB	 DB_Shutdown_instance DB_Shutdown Execute Done Busy Error ErrorID	DB_Shutdown_instance (Execute, Done, Busy, Error, ErrorID);

Note The *DB_Shutdown_instance* is an instance of DB_Shutdown instruction, which is declared as a variable.

Variables

Input Variables

Name	Meaning	Data type	Valid range	Unit	Default	Description
Execute	Execute	BOOL	TRUE or FALSE	---	FALSE	Specify the execution condition.

Output Variables

Name	Meaning	Data type	Valid range	Unit	Description
Done	Done	BOOL	TRUE or FALSE	---	TRUE when the instruction is normally completed.
Busy	Executing	BOOL	TRUE or FALSE	---	TRUE when the instruction is being executed.
Error	Error	BOOL	TRUE or FALSE	---	TRUE when the instruction is terminated due to an error.
ErrorID	Error Code	WORD	16#0000 to 16#FFFF	---	Contains the error code when an error occurs.

Related System-defined Variables

System-defined variables	Name	Data type	Valid range	Description
_DBC_Status.Run	DB Connection Service Running Status	BOOL	TRUE or FALSE	This variable changes to FALSE when this instruction is executed.
_DBC_Status.Test	DB Connection Service Test Mode Status	BOOL	TRUE or FALSE	This variable changes to FALSE when this instruction is executed.
_DBC_Status.Shutdown	DB Connection Service Shutdown Status	BOOL	TRUE or FALSE	This variable changes to TRUE when this instruction is executed.

Related Error Codes

Error code	Meaning	Description
041D hex	Too Many Instructions Executed at the Same Time	More than 32 DB Connection Instructions were executed at the same time.
3001 hex	DB Connection Service Run Mode Change Failed	The instruction was executed while the stopping processing of the DB Connection Service was in progress.
3002 hex	DB Connection Service Shutdown or Shutting Down	The instruction was executed after the DB Connection Service was shut down or while the DB Connection Service was being shut down.
3015 hex	DB Connection Service Initializing	The instruction was executed while the initialization processing of the DB Connection Service was in progress.

Function

This instruction is used to shut down the DB Connection Service.
Be sure to execute this instruction before turning OFF the power supply to the CPU Unit to prevent data loss of Operation Logs.

Precautions for Correct Use

- Execution of this instruction is continued until processing is completed even if the value of *Execute* changes to FALSE or the execution time exceeds the task period. The value of *Done* changes to TRUE when processing is completed. Use this to confirm normal completion of processing.
- Refer to *Using this Section* of the *NJ/NX-series Instructions Reference Manual* (Cat. No. W502) for a timing chart for *Execute, Done, Busy,* and *Error*.
- This instruction cannot be used on an event task. A compiling error will occur.
- The DB Connection Instructions cannot be executed during and after execution of this instruction. When a DB Connection Instruction is executed, it will be terminated due to an error.
- Be sure to execute this instruction before you turn OFF the power supply to the CPU Unit. If the power supply is turned OFF without executing this instruction, the Operation Log file may be corrupted or its contents may be lost.
- An error occurs for this instruction in the following cases. *Error* will be TRUE.
    - When the instruction was executed while the initialization processing of the DB Connection Service was in progress
    - When the instruction was executed while the stopping processing of the DB Connection Service was in progress
    - When the instruction was executed after the DB Connection Service was shut down or while the DB Connection Service was being shut down
    - When more than 32 DB Connection Instructions were executed at the same time

Sample Programming

This section gives sample programming for shutting down the DB Connection Service when the trigger variable changes to TRUE.

Ladder Diagram

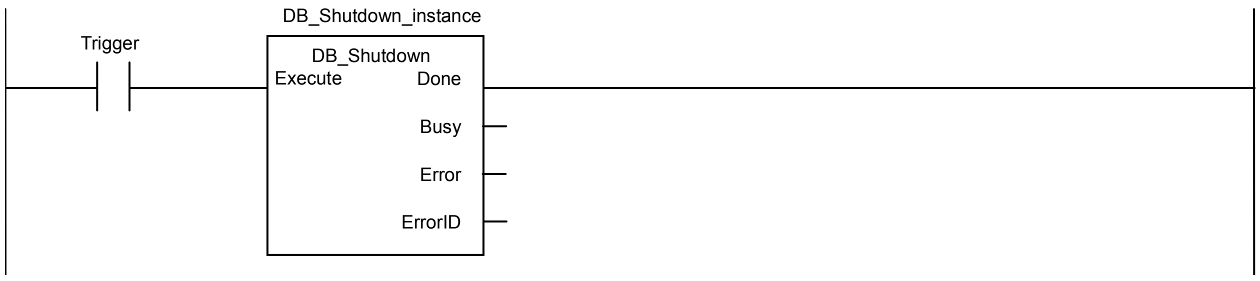
● Main Variables

Name	Data type	Initial value	Comment
DB_Shutdown_instance	DB_Shutdown	---	Instance of DB_Shutdown instruction
Trigger	BOOL	FALSE	Variable used as a trigger for shutting down the DB Connection Service
Shutdown_OK	BOOL	FALSE	This variable changes to TRUE when the DB_Shutdown instruction is completed normally.

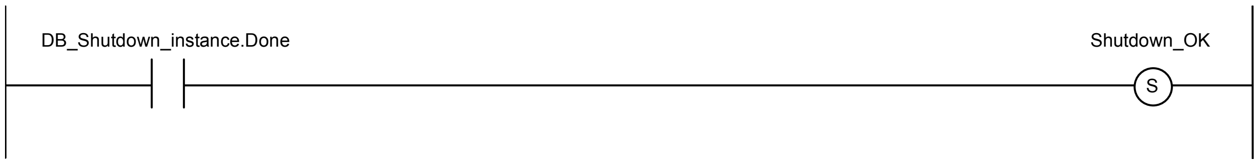
● Sample Programming

- Shut down the DB Connection Service.

Shut down the DB Connection Service.



When the instruction is normally completed, change the variable *Shutdown_OK* to TRUE.



Structured Text (ST)

● Main Variables

Name	Data type	Initial value	Comment
DB_Shutdown_instance	DB_Shutdown	---	Instance of DB_Shutdown instruction
Trigger	BOOL	FALSE	Variable used as a trigger for shutting down the DB Connection Service
LastTrigger	BOOL	FALSE	Variable to retain the trigger status of the previous execution
Operating	BOOL	FALSE	Shutting down the DB Connection Service is executed when this variable is TRUE.
OperatingStart	BOOL	FALSE	The initialization processing is executed when this variable is TRUE.
ShutdownOK	BOOL	FALSE	This variable changes to TRUE when the DB_Shutdown instruction is completed normally.

● Sample Programming

```
(* ------------------------------------------------------------------------------------------------------------------------
   ♦ Shut down the DB Connection Service.
   ------------------------------------------------------------------------------------------------------------------------ *)

// Start the sequence when the variable Trigger changes to TRUE.
IF ( (Trigger=TRUE) AND (LastTrigger=FALSE) ) THEN
   OperatingStart := TRUE;
   Operating := TRUE;
END_IF;
LastTrigger := Trigger;

// Sequence start processing
IF (OperatingStart=TRUE) THEN
   // Initialize the instruction instance.
   DB_Shutdown_instance( Execute:=FALSE );

   OperatingStart := FALSE;
END_IF;

IF (Operating=TRUE) THEN
   // Shut down the DB Connection Service.
   DB_Shutdown_instance( Execute:=TRUE );

   IF (DB_Shutdown_instance.Done=TRUE) THEN
      // Normal end processing
      ShutdownOK := TRUE;
      Operating := FALSE;
   END_IF;
   IF (DB_Shutdown_instance.Error=TRUE) THEN
      // Error handler
      Operating := FALSE;
   END_IF;
END_IF;
```

Appendix B

B-1 Task Design Procedure

This section describes the task design procedure for using the DB Connection function.

Refer to the *NJ/NX-series CPU Unit Software User's Manual* (Cat. No. W501) for task and system service operation specifications of the NJ-series Controllers.

B-1-1 Startup Time of DB Connection Service

The time required to get the DB Connection Service ready for operation (i.e. until the *_DBC_Status.Run* system-defined variable changes to True) after turning ON the power supply to the CPU Unit (hereinafter called "startup time") depends on the database type to connect and the percentage of task execution time.
The following table shows the reference values for some combinations.
Please design your system in reference to these values.

• NJ501-1520

DB type	Percentage of task execution time*	Reference value for startup time of the DB Connection Service (Average)
Oracle	50%	58.43 s
	80%	124.95 s
SQL Server	50%	54.02 s
	80%	120.95 s
DB2	50%	56.26 s
	80%	128.49 s
MySQL	50%	57.41 s
	80%	131.33 s
Firebird	50%	56.65 s
	80%	129.07 s
PostgreSQL	50%	59.06 s
	80%	124.26 s

• NJ101-1020

DB type	Percentage of task execution time*	Reference value for startup time of the DB Connection Service (Average)
Oracle	50%	75.59 s
	60%	89.31 s
SQL Server	50%	56.36 s
	60%	67.17 s
DB2	50%	61.90 s
	60%	73.35 s
MySQL	50%	54.46 s
	60%	66.83 s
Firebird	50%	57.61 s
	60%	70.98 s
PostgreSQL	50%	63.61 s
	60%	76.63 s

* Percentage of task execution time on the Task Execution Time Monitor of Sysmac Studio

The following table shows the measurement conditions and items.

Measurement conditions		Description
Item	Subitem	
CPU Unit	Task composition	Primary periodic task only Task period: 1 ms
System configuration	Basic configuration	• No EtherCAT network • No CJ-series Units • USB connection with Sysmac Studio
	Network configuration	• No connection with other controllers • No connection with HMI

Precautions for Correct Use

The DB Connection Service is executed as a system service.
Therefore, the execution time of each processing may require time if the startup processing of the DB Connection Service and other system service processing are executed at the same time.

B-1-2 Reference Values for Execution Time of DB Connection Instructions

The DB Connection Instructions are function block type of instructions that are executed over multiple task periods.
The following table gives the reference values for execution time of each DB Connection Instruction. Refer to *B-2-1 Restrictions to Execution Time of DB Connection Instructions* for the factors that fluctuate execution time of DB Connection Instructions.

Conditions
DB_Insert: When executing an INSERT operation for 100-column record
DB_Select: When searching for one record from 100,000 records and retrieving 100-column data*
* The primary key is specified for the retrieval condition.

• NJ501-1520

DB type	Percentage of task execution time	Instruction	Reference value for instruction execution time	
			Average	Maximum
Oracle Database 11g	50%	DB_Insert	16.2 ms	65 ms
		DB_Select	37.1 ms	75 ms
	80%	DB_Insert	49.2 ms	184 ms
		DB_Select	101.6 ms	272 ms
SQL Server 2012	50%	DB_Insert	16.1 ms	57 ms
		DB_Select	23.8 ms	98 ms
	80%	DB_Insert	45.5 ms	112 ms
		DB_Select	72.5 ms	236 ms
DB2 10.5	50%	DB_Insert	27.5 ms	115 ms
		DB_Select	37.1 ms	80 ms
	80%	DB_Insert	69.4 ms	176 ms
		DB_Select	99.5 ms	352 ms
MySQL 5.6 Storage engine: InnoDB	50%	DB_Insert	40.3 ms	273 ms
		DB_Select	32.0 ms	41 ms
	80%	DB_Insert	65.0 ms	315 ms
		DB_Select	69.4 ms	164 ms
Firebird 2.5	50%	DB_Insert	23.8 ms	156 ms
		DB_Select	71.7 ms	153 ms
	80%	DB_Insert	52.8 ms	139 ms
		DB_Select	118.4 ms	234 ms

PostgreSQL 9.4	50%	DB_Insert	17.0 ms	78 ms
		DB_Select	30.9 ms	83 ms
	80%	DB_Insert	48.3 ms	175 ms
		DB_Select	89.1 ms	250 ms

• NJ101-1020

DB type	Percentage of task execution time	Instruction	Reference value for instruction execution time	
			Average	Maximum
Oracle Database 11g	50%	DB_Insert	27.8 ms	311 ms
		DB_Select	42.0 ms	311 ms
	60%	DB_Insert	39.0 ms	342 ms
		DB_Select	62.4 ms	369 ms
SQL Server 2012	50%	DB_Insert	26.7 ms	287 ms
		DB_Select	36.2 ms	626 ms
	60%	DB_Insert	37.5 ms	621 ms
		DB_Select	52.1 ms	456 ms
DB2 10.5	50%	DB_Insert	39.8 ms	544 ms
		DB_Select	59.0 ms	467 ms
	60%	DB_Insert	52.3 ms	397 ms
		DB_Select	81.0 ms	655 ms
MySQL 5.6 Storage engine: InnoDB	50%	DB_Insert	44.2 ms	365 ms
		DB_Select	36.0 ms	599 ms
	60%	DB_Insert	54.6 ms	834 ms
		DB_Select	52.4 ms	450 ms
Firebird 2.5	50%	DB_Insert	34.0 ms	314 ms
		DB_Select	78.4 ms	403 ms
	60%	DB_Insert	45.4 ms	388 ms
		DB_Select	101.0 ms	472 ms
PostgreSQL 9.4	50%	DB_Insert	28.7 ms	306 ms
		DB_Select	45.0 ms	291 ms
	60%	DB_Insert	41.3 ms	471 ms
		DB_Select	66.0 ms	433 ms

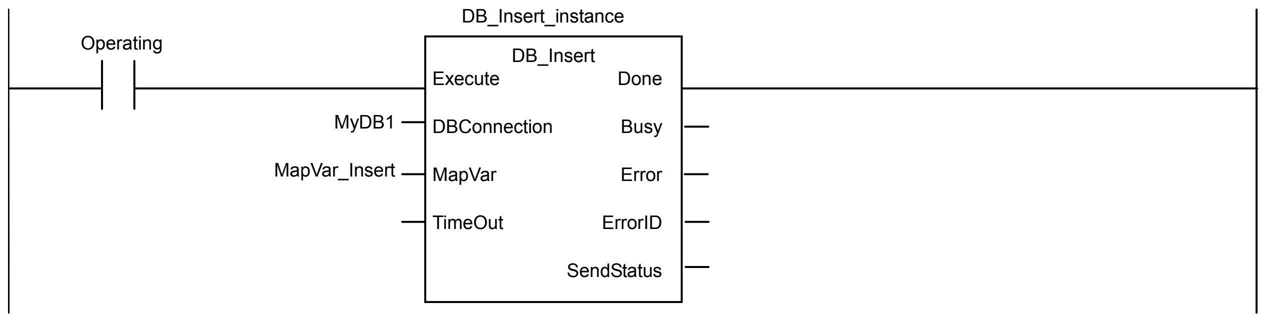
The following table shows the measurement conditions and items.

Measurement conditions		Description
Item	Subitem	
CPU Unit	Task composition	Primary periodic task only Task period: 1 ms DB Map Variable: An internal variable of a program
Server	Computer	CPU: Intel Xeon(R) CPU E31220 @ 3.10 GHz 3.09 GHz Memory: 8.00 GB
	Operating system	Windows Server 2008 Standard SP2 64 bits
	DB type	Oracle Database Express Edition 11g 11.2.0 SQL Server 2012 DB2 for Linux, UNIX and Windows 10.5 MySQL Community Edition 5.6 Firebird 2.5 PostgreSQL 9.4
SQL statement to execute	Record composition	INT: 40 columns REAL: 40 columns STRING[16]: 16 columns DATE_AND_TIME: 4 columns
Operation Logs	Execution Log	Recorded
	Debug Log	Stopped
	SQL Execution Failure Log	Not recorded

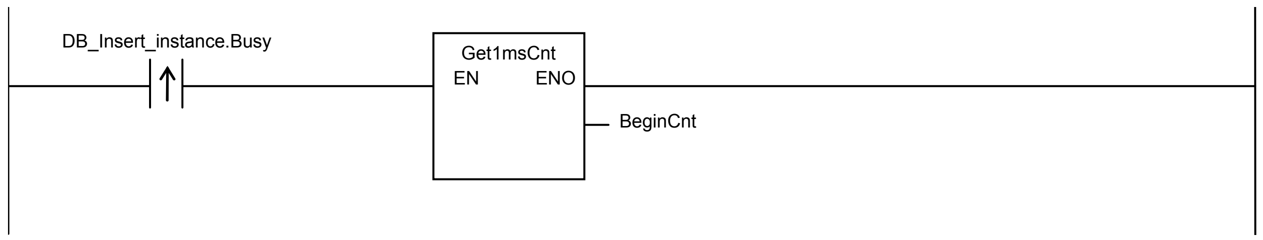
B-1-3 How to Measure Execution Time of DB Connection Instructions

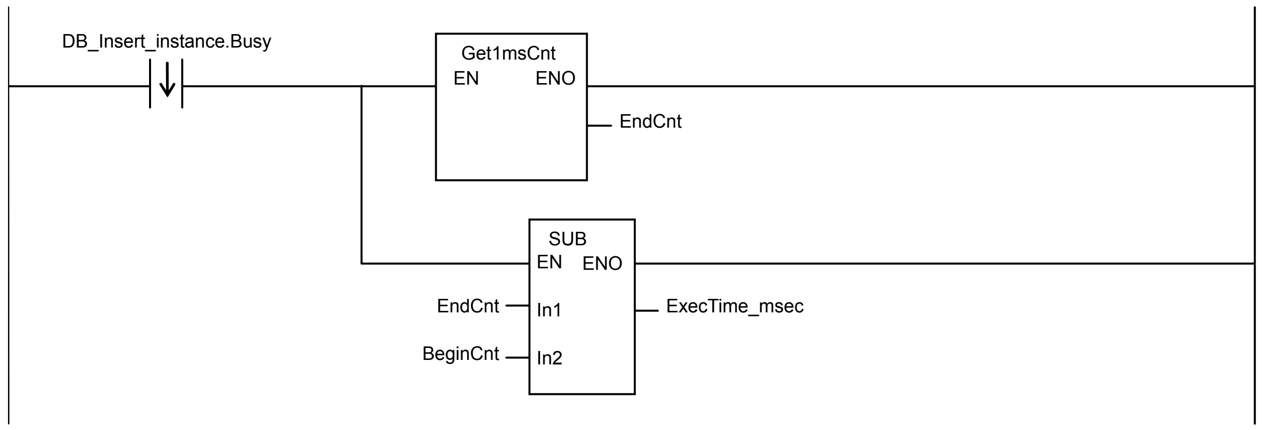
The execution time of DB Connection Instructions can be measured by a Get1msCnt instruction. The instruction calculates the value of free-running counter of the cycle from when the *Busy* output variable changes to TRUE to when the variable changes to FALSE.

・Example for measuring execution time of a DB_Insert instruction
Insert a record to the DB Connection *MyDB1*.



Measure execution time of the DB_Insert instruction and output the result to the *ExecTime_msec* output variable of the SUB instruction.





B-1-4 Guideline for System Service Execution Time Ratio

The DB Connection Service is executed as a system service.
When a DB Connection Instruction is executed by a user program, the DB Connection Service executes the processing as a system service. If sufficient execution time cannot be allocated to the system services, the DB Connection Instruction may take long execution time. Or, other processing executed in the system services may take long execution time.
To execute the DB Connection Instructions according to the performance specifications, design the task so that the system service execution time ratio (CPU usage) meets the following.

CPU Unit model	Guideline for system service execution time ratio
NJ501-□□20	20% or greater
NJ101-□□20	40% or greater

Precautions for Safe Use

The above system service execution time ratio (CPU usage) is just a guideline.
The appropriate value of system service execution time ratio (CPU usage) depends on the usage of other services executed as a system service.
Before starting actual operation, you must test performance under all foreseeable conditions on the actual system and make sure that the DB Connection Instructions are executed within the appropriate execution time.

Precautions for Correct Use

- If the system service execution time ratio is reduced, operation failures or communications errors may occur when each operation is executed from Sysmac Studio. If an operation failure or communications error occurs when you execute an operation from Sysmac Studio, retry the operation after performing the following:
  - Check the cable connection.
  - Check the communications settings.
  - Increase the response monitoring time in the Communications Setup.
  - Check that the operation status of the DB Connection Service is not *Initializing*, *Error*, or *Shutdown*.
    For details of the operation status of the DB Connection Service, refer to *4-3-1 Operation Status of the DB Connection Service*.
- When Sysmac Studio cannot go online, refer to the *NJ/NX-series Troubleshooting Manual* (Cat. No. W503) for details.
- If the time set for system service monitoring cannot be secured for system services, an Insufficient System Service Time Error will occur. The error is a major fault level Controller error. When the error has occurred, user program execution stops. To secure enough time for system services and task execution, set the minimum value that can satisfy the response performance of the system service processing for system service monitoring. The system service monitoring setting is just for monitoring whether or not the specified time can be secured for system service execution. It does not guarantee that system services are executed for the specified time.
- The system service execution time is affected by task execution time and tag data links. Refer to the *NJ/NX-series CPU Unit Software User's Manual* (Cat. No. W501) for details of task specifications, tag data link service, and system services.

B-1-5 Checking the System Service Execution Time Ratio

When you design the tasks, confirm that sufficient execution time can be allocated to system services by the following methods.

● Desktop Calculations

This is an example for a project that consists of one primary periodic task.
Refer to the *NJ/NX-series CPU Unit Software User’s Manual* (Cat. No. W501) to make a rough estimate of the average task execution time on paper.

Design the task using the following as a guideline:

- NJ501-□□20
  Average task execution time < task period x 0.8
- NJ101-□□20
  Average task execution time < task period x 0.6

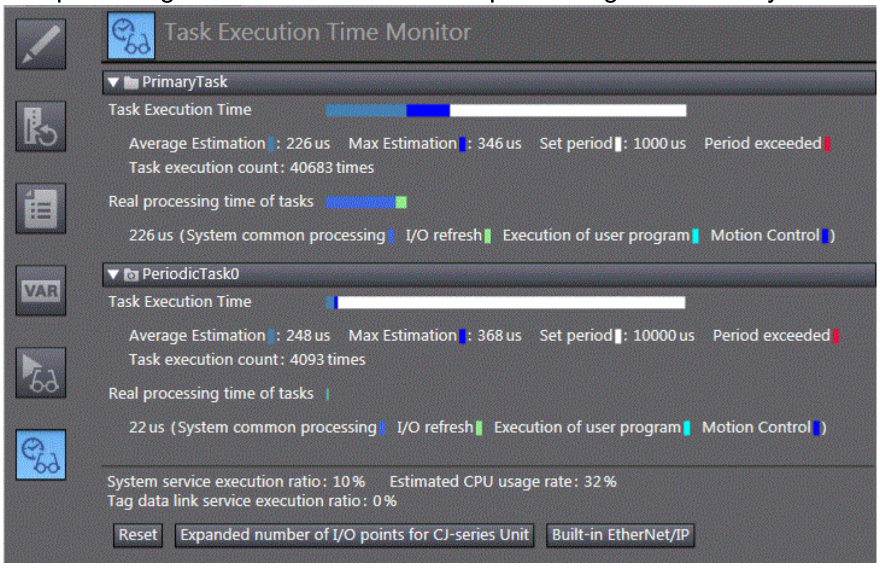
● Estimating with the Simulator on Sysmac Studio

Check the value of *Estimated CPU usage rate* with the Task Execution Time Monitor of the Simulator on Sysmac Studio.
Refer to the *NJ/NX-series CPU Unit Software User’s Manual* (Cat. No. W501) for the procedure to check the operation in the Simulator.

Design the task using the following as a guideline:

- NJ501-□□20
  Estimated CPU usage rate - System service execution time ratio < 80%
- NJ101-□□20
  Estimated CPU usage rate - System service execution time ratio < 60%

The estimated CPU usage rate shows the percentage of the total of the following times in the task period:
Estimated maximum value of the task processing time + Tag data link service execution ratio + Required system service processing time for system service monitoring.
The value found by subtracting the system service execution ratio from the estimated CPU usage rate is the percentage for the execution time of processing other than system services.


Task Execution Time Monitor

▼ PrimaryTask
Task Execution Time
Average Estimation : 226 us Max Estimation : 346 us Set period : 1000 us Period exceeded
Task execution count : 40683 times
Real processing time of tasks
226 us (System common processing I/O refresh Execution of user program Motion Control)

▼ PeriodicTask0
Task Execution Time
Average Estimation : 248 us Max Estimation : 368 us Set period : 10000 us Period exceeded
Task execution count : 4093 times
Real processing time of tasks
22 us (System common processing I/O refresh Execution of user program Motion Control)

System service execution ratio : 10 % Estimated CPU usage rate : 32 %
Tag data link service execution ratio : 0 %

● Calculating Times on the Physical Controller

When the project consists of one primary periodic task, check the average task execution time using the Task Execution Time Monitor function of Sysmac Studio while online with the physical Controller. Design the task using the following as a guideline:

- NJ501-□□20
  Average task execution time < Task period x 0.8
- NJ101-□□20
  Average task execution time < Task period x 0.6

When the project consists of multiple tasks, test performance under all foreseeable conditions on the actual system and make sure that the DB Connection Instructions are executed within the appropriate execution time before starting actual operation.

B-2 Execution Time of DB Connection Instructions

This section describes execution time of DB Connection Instructions.

B-2-1 Restrictions to Execution Time of DB Connection Instructions

Execution time of DB Connection Instructions varies according to the following factors.

- Status of the NJ-series CPU Unit
- DB type
- Processing capability and load status of the server that contains the DB
- DB response time
- Contents of the SQL statement to execute
- Number of retrieved records in the execution of DB_Select instruction

Due to the above factors, execution time of a DB Connection Instruction may exceed the reference value given in *B-1-2 Reference Values for Execution Time of DB Connection Instructions*.
The following table lists the phenomena that we confirmed under our measurement environment and their countermeasures.

No.	Phenomena
1	After the power supply to the CPU Unit was turned ON, execution time of the first DB Connection Instruction (i.e. DB_Insert, DB_Update, DB_Select, or DB_Delete instruction) got longer.
2	After execution of a DB_CreateMapping instruction, execution time of the first DB_Insert instruction got longer.
3	When communications or SD Memory Card processing was executed in the CPU Unit, execution time of a DB Connection Instruction got longer.
4	Execution time of DB Connection Instructions is steadily long.
5	Depending on the DB’s status, execution time of a DB Connection Instruction (i.e., DB_Insert, DB_Update, DB_Select, or DB_Delete instruction) got longer.

Refer to *B-1-2 Reference Values for Execution Time of DB Connection Instructions* for the measurement conditions and items.

Phenomenon 1: After the Power Supply to the CPU Unit was Turned ON, Execution Time of the First DB Connection Instruction (i.e. DB_Insert, DB_Update, DB_Select, or DB_Delete instruction) Got Longer

- Possible causes

The following can be the causes:

1. For the first DB Connection Instruction (i.e. DB_Insert, DB_Update, DB_Select, or DB_Delete instruction) that is executed after the power supply to the CPU Unit is turned ON, the CPU Unit may require longer processing time than usual.
2. For the first DB_Insert instruction that is executed after execution of a DB_CreateMapping instruction, the DB may require longer processing time than usual.

The following table gives the reference values for execution time of the first DB Connection Instruction after the power supply to the CPU Unit is turned ON.

• NJ501-□□20

DB type	Instruction	Reference value for instruction execution time	Measurement condition
Oracle Database 11g	DB_Insert	124 ms	When executing an INSERT operation for 100-column record
	DB_Select	175 ms	When searching for one record from 100,000 records and retrieving 100-column data*
SQL Server 2012	DB_Insert	136 ms	When executing an INSERT operation for 100-column record
	DB_Select	130 ms	When searching for one record from 100,000 records and retrieving 100-column data*
DB2 10.5	DB_Insert	315 ms	When executing an INSERT operation for 100-column record
	DB_Select	839 ms	When searching for one record from 100,000 records and retrieving 100-column data*
MySQL 5.6 Storage engine: InnoDB	DB_Insert	62 ms	When executing an INSERT operation for 100-column record
	DB_Select	38 ms	When searching for one record from 100,000 records and retrieving 100-column data*
Firebird 2.5	DB_Insert	35 ms	When executing an INSERT operation for 100-column record
	DB_Select	175 ms	When searching for one record from 100,000 records and retrieving 100-column data*
PostgreSQL 9.4	DB_Insert	87 ms	When executing an INSERT operation for 100-column record
	DB_Select	111 ms	When searching for one record from 100,000 records and retrieving 100-column data*

• NJ101-□□20

DB type	Instruction	Reference value for instruction execution time	Measurement condition
Oracle Database 11g	DB_Insert	219 ms	When executing an INSERT operation for 100-column record
	DB_Select	406 ms	When searching for one record from 100,000 records and retrieving 100-column data*
SQL Server 2012	DB_Insert	213 ms	When executing an INSERT operation for 100-column record
	DB_Select	248 ms	When searching for one record from 100,000 records and retrieving 100-column data*
DB2 10.5	DB_Insert	373 ms	When executing an INSERT operation for 100-column record
	DB_Select	395 ms	When searching for one record from 100,000 records and retrieving 100-column data*
MySQL 5.6 Storage engine: InnoDB	DB_Insert	219 ms	When executing an INSERT operation for 100-column record
	DB_Select	245 ms	When searching for one record from 100,000 records and retrieving 100-column data*
Firebird 2.5	DB_Insert	162 ms	When executing an INSERT operation for 100-column record
	DB_Select	450 ms	When searching for one record from 100,000 records and retrieving 100-column data*
PostgreSQL 9.4	DB_Insert	277 ms	When executing an INSERT operation for 100-column record
	DB_Select	379 ms	When searching for one record from 100,000 records and retrieving 100-column data*

Percentage of task execution time: 50%

* The primary key is specified for the retrieval condition.

・Countermeasures

Measure the execution time of each DB Connection Instruction in reference to *B-1-3 How to Measure Execution Time of DB Connection Instructions*. If the execution time of a DB Connection Instruction exceeds the acceptable range of the equipment, take the following actions.

1. Set a timeout for the DB Connection Instruction. Refer to B-2-4 Ensuring Equipment Performance (Takt Time) by Monitoring the Instruction Execution Timeout for details.
2. Execute a dummy DB_Insert instruction once after executing the DB_CreateMapping instruction as a preparation for starting the actual operation.

Phenomenon 2: After Execution of a DB_CreateMapping Instruction, Execution Time of the First DB_Insert Instruction Got Longer

• Possible causes

The following can be the causes:

1. For the first DB_Insert instruction that is executed after execution of a DB_CreateMapping instruction, the DB may require longer processing time than usual.

The following table gives the reference values for execution time of the first DB_Insert instruction that is executed after execution of a DB_CreateMapping instruction.

• NJ501-□□20

DB type	Instruction	Reference value for instruction execution time	Measurement condition
Oracle Database 11g	DB_Insert	29.9 ms	When executing an INSERT operation for 100-column record
SQL Server 2012	DB_Insert	17.5 ms	When executing an INSERT operation for 100-column record
DB2 10.5	DB_Insert	26.4 ms	When executing an INSERT operation for 100-column record
MySQL 5.6 Storage engine: InnoDB	DB_Insert	41.7 ms	When executing an INSERT operation for 100-column record
Firebird 2.5	DB_Insert	22.5 ms	When executing an INSERT operation for 100-column record
PostgreSQL 9.4	DB_Insert	14.1 ms	When executing an INSERT operation for 100-column record

• NJ101-□□20

DB type	Instruction	Reference value for instruction execution time	Measurement condition
Oracle Database 11g	DB_Insert	28.2 ms	When executing an INSERT operation for 100-column record
SQL Server 2012	DB_Insert	35.6 ms	When executing an INSERT operation for 100-column record
DB2 10.5	DB_Insert	52.7 ms	When executing an INSERT operation for 100-column record
MySQL 5.6 Storage engine: InnoDB	DB_Insert	59.3 ms	When executing an INSERT operation for 100-column record
Firebird 2.5	DB_Insert	32.6 ms	When executing an INSERT operation for 100-column record
PostgreSQL 9.4	DB_Insert	32.1 ms	When executing an INSERT operation for 100-column record

Percentage of task execution time: 50%

• Countermeasures

1. Measure the execution time of the DB Connection Instruction in reference to *B-1-3 How to Measure Execution Time of DB Connection Instructions*. If the execution time of the DB Connection Instruction exceeds the acceptable range of the equipment, take the following actions.
   • Execute a dummy DB_Insert instruction once after executing the DB_CreateMapping instruction as a preparation for starting the actual operation.

Phenomenon 3: When Communications or SD Memory Card Processing was Executed in the CPU Unit, Execution Time of a DB Connection Instruction Got Longer

・Possible causes

The following can be the causes:

1. The sufficient processing time may not be allocated to the DB Connection Service that is executed as a system service due to execution of communications or SD Memory Card processing.

・ Countermeasures

1. Reconsider the task design so that the sufficient execution time can be allocated to the system services in reference to *B-1-4 Guideline for System Service Execution Time Ratio*.

Phenomenon 4: Execution Time of DB Connection Instructions is Steadily Long

・Possible causes

The following can be the causes:

1. The sufficient execution time may not be allocated to the system services.

・Countermeasures

1. Reconsider the task design so that the sufficient execution time can be allocated to the system services in reference to *B-1-4 Guideline for System Service Execution Time Ratio*.

Phenomenon 5: Depending on the DB's Status, Execution Time of a DB Connection Instruction (i.e., DB_Insert, DB_Update, DB_Select, or DB_Delete Instruction Got Longer.

・Possible causes

The following can be the causes:

1. Load on the server was temporarily increased.
2. The specified table contains many records.
3. The data clear operation was executed for the specified table.
4. The specified table was temporarily locked.

・Countermeasures

Measure the processing time in the DB in reference to *B-2-3 How to Measure DB Response Time*. Identify the cause based on the timing when the processing time got longer in the DB and take a countermeasure in the server.

B-2-2 Impact of Operation Log Recording on Execution Time of DB Connection Instructions

When the Operation Logs are recorded, execution time of DB Connection Instructions (i.e. DB_Insert, DB_Update, DB_Select, and DB_Delete instructions) gets longer.
The following table gives the reference values for increased execution time of DB Connection Instructions while the Operation Logs are recorded.
Confirm that the equipment will not be adversely affected before starting recording to the Operation Logs.

・ NJ501-□□20

Log type	Instruction	Reference value for increase in instruction execution time	Measurement condition
Execution Log	DB_Insert	+1.4 ms	When executing an INSERT operation for 100-column record
Debug Log	DB_Insert	+3.3 ms	When executing an INSERT operation for 100-column record

・ NJ101-□□20

Log type	Instruction	Reference value for increase in instruction execution time	Measurement condition
Execution Log	DB_Insert	+2.0 ms	When executing an INSERT operation for 100-column record
Debug Log	DB_Insert	+7.6 ms	When executing an INSERT operation for 100-column record

Percentage of task execution time: 50%

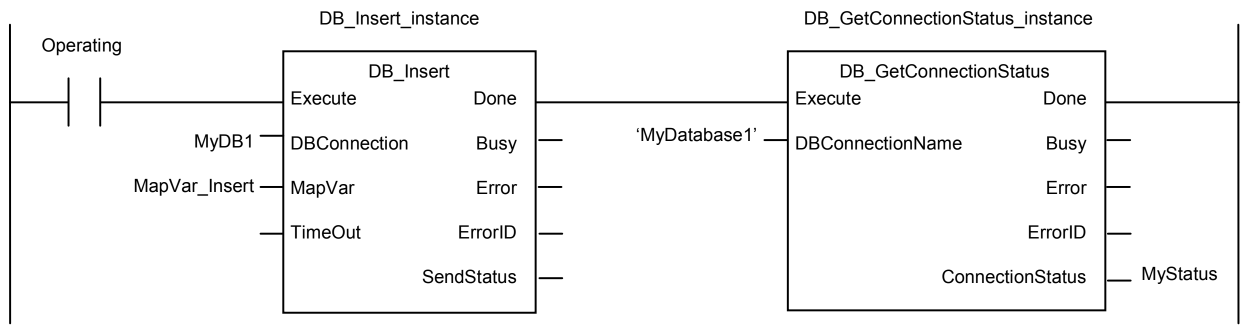
B-2-3 How to Measure DB Response Time

The DB response time refers to the time since an SQL statement is sent from the CPU Unit until the SQL execution result is returned from the DB. You can find the DB response time by executing a DB_GetConnectionStatus instruction after executing an instruction that sends an SQL statement.

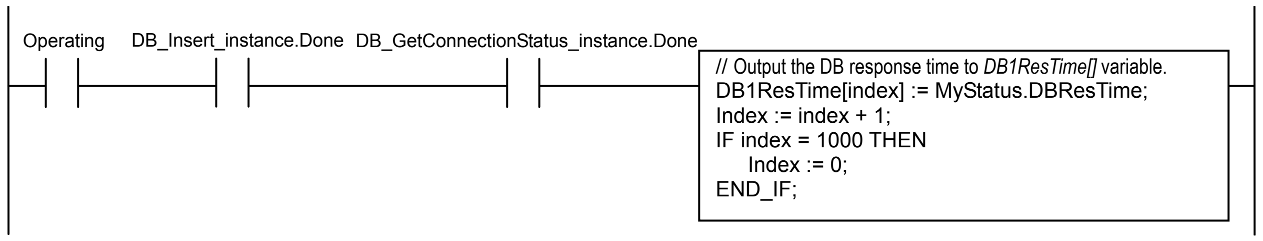
An example user program is given below.

・ Measurement example of DB response time for a DB_Insert instruction

Find the DB response time for a DB_Insert instruction.



Normal end processing



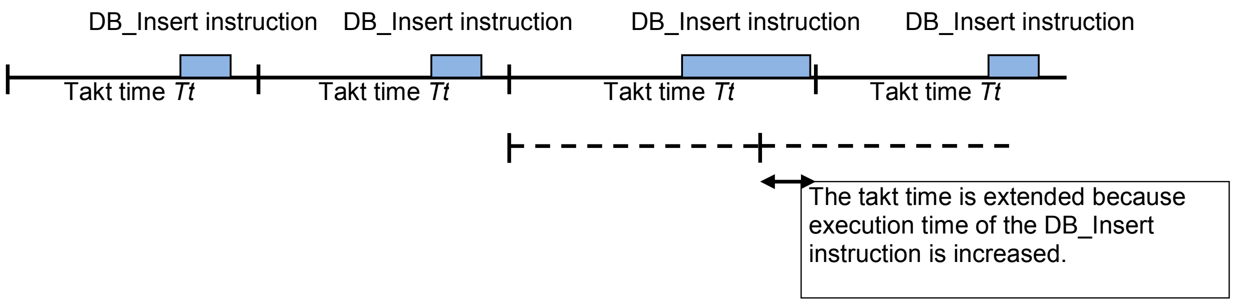
```
// Output the DB response time to DB1ResTime[] variable.
DB1ResTime[index] := MyStatus.DBResTime;
Index := index + 1;
IF index = 1000 THEN
    Index := 0;
END_IF;
```

You can also check the DB response time with the Execution Log or Debug Log.

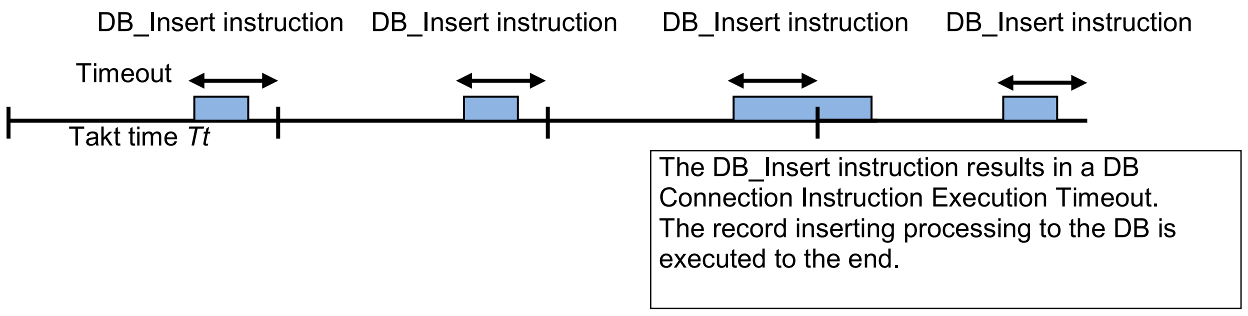
B-2-4 Ensuring Equipment Performance (Takt Time) by Monitoring Instruction Execution Timeout

If you do not want to lower the equipment performance (or extend the takt time) when the execution time of DB Connection Instruction is increased, set a timeout for the instructions.
You can specify a timeout in the *TimeOut* input variable to the DB_Insert, DB_Update, DB_Select, and DB_Delete instructions.
For the timeout of instructions, specify the maximum time that can be used for DB access in the takt time.
If you set a timeout for a DB_Insert instruction for the equipment where production data is stored into the DB using the DB_Insert instruction at the end of the takt time, for example, a DB Connection Instruction Execution Timeout will occur for the DB_Insert instruction when the record inserting processing to the DB is not completed in the takt time. In this case, the record inserting processing to the DB is executed to the end.
You can continue the operation without lowering the equipment performance (or extending the takt time) by specifying a timeout for the instruction even if execution time of DB Connection Instructions is temporarily increased.

・ When timeout is not specified



・ When timeout is specified


DB_Insert instruction
DB_Insert instruction
DB_Insert instruction
DB_Insert instruction
Timeout
Takt time *Tt*
The DB_Insert instruction results in a DB Connection Instruction Execution Timeout. The record inserting processing to the DB is executed to the end.

Precautions for Correct Use

- When a DB Connection Instruction Execution Timeout occurred for a DB_Select instruction, the values of the retrieved record are not stored in the *MapVar* in-out variable.
- When a DB Connection Instruction Execution Timeout occurs repeatedly, reconsider the task design and the server environment that contains the DB.

B-3 Specifications

This section gives the specifications of the Database Connection CPU Units.

B-3-1 General Specifications

Refer to the following manual.

• NJ/NX-series CPU Unit Software User's Manual (Cat. No. W501)

B-3-2 Performance Specifications

Refer to the following manual.

• NJ/NX-series CPU Unit Software User's Manual (Cat. No. W501)

B-3-3 Function Specifications

Refer to the following manual.

• NJ/NX-series CPU Unit Software User's Manual (Cat. No. W501)

• Common Specifications to NJ-series CPU Units

Item			NJ501-1□20		NJ501-4320	NJ101-□□20
			Version 1.07 or earlier	Version 1.08 or later		
Debugging	Data tracing	Maximum number of simultaneous data traces	4	2*	2*	2*

* If the trace number is set to 2 or greater when executing a data trace related instruction, an error (Illegal Data Position Specified) will occur for the instruction. *ENO* of the instruction will become FALSE.

• DB Connection Service Functionality

Refer to *1-2-1 DB Connection Service Specifications* for detailed specifications of the DB Connection Service.

Item		NJ501-1□20	NJ501-4320	NJ101-□□20
Maximum number of DB Connections		3	3	1
Supported DB*	Oracle	Supported	Supported	Supported
	SQL Server	Supported	Supported	Supported
	DB2	Supported	Not supported	Supported
	MySQL	Supported	Supported	Supported
	Firebird	Supported	Not supported	Supported
	PostgreSQL	Supported	Not supported	Supported
Number of DB Map Variables for which a mapping can be connected*1 *	Oracle	30	15	15
	SQL Server	60	15	15
	DB2	30	N/A	15
	MySQL	30	15	15
	Firebird	15	N/A	15
	PostgreSQL	30	N/A	15
Spool function	Spool capacity	1 MB	1 MB	192 KB

*1 Even if the number of DB Map Variables has not reached the upper limit, the total number of members of structures used as data type of DB Map Variables is 10,000 members max.

Note: The items marked with * (asterisk) were added or changed by version upgrades. Refer to *B-4 Version Information* for the version upgrades.

B-4 Version Information

This section describes the relationship between the unit versions of CPU Units and the Sysmac Studio versions, and the DB Connection functions that were added or changed for each unit version of the CPU Units.

B-4-1 Unit Versions and Corresponding DB Connection Service Versions

The following table gives the relationship between unit versions of CPU Units and the DB Connection Service versions.

Unit version of CPU Unit	DB Connection Service version
1.10 or later*	1.02
1.10*	1.01
1.09	
1.08	
1.07 or earlier	1.00

* The CPU Units with unit version 1.10 come with DB Connection Service version 1.01 or 1.02. The version can be checked with the Production Information Dialog Box of Sysmac Studio while online. Refer to *Versions* (P. 50) for how to check the versions of the CPU Units and DB Connection Service.

B-4-2 DB Connection Functions That Were Added or Changed for Each Unit Version

This section gives the DB Connection functions that were added or changed for version upgrades of CPU Units.

● Additions and Changes to Function Specifications

The following table gives the unit version of the CPU Units and the Sysmac Studio version for each addition or change to the function specifications.
Refer to the following manual for other function specifications.
• NJ/NX-series CPU Unit Software User's Manual (Cat. No. W501)

Function		Addition/ change	Unit version	Sysmac Studio version	Reference
DB Connection settings	Database type	Change	1.08	1.09	2-2-2 DB Connection Settings
		Change	1.10	1.14	
DB Connection status	SQL status*	Change	1.08	---	4-3-4 Checking the Status of each DB Connection
	Error code*				
	Error message*				

* Error information in the SQL Server connection was changed.

B-4-3 Actual Unit Version of CPU Unit and Unit Version Set in the Sysmac Studio Project

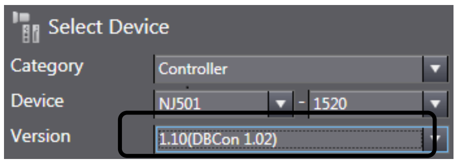
The following table gives the relationship between the unit versions of CPU Units and the corresponding Sysmac Studio versions.

Unit Versions and Corresponding Sysmac Studio Versions

The following table gives the relationship between the unit versions of CPU Units, the DB Connection Service versions, and the Sysmac Studio versions that can set the unit versions.

Unit version of CPU Unit	DB Connection Service version	Sysmac Studio version that can set the unit version
1.10	1.02	1.14 or higher[*1, *2]
	1.01	1.13 or higher[*1, *2]
1.09		1.10 or higher
1.08		1.09 or higher
1.07	1.00	1.08 or higher
1.05		1.06 or higher

*1 When you set a unit version in Sysmac Studio version 1.14 or higher, a unit version and DB Connection Service version are displayed in the Version box because more than one DB connection version exists for the same unit version of the CPU Unit. For example, when you want to create a project for a CPU Unit with unit version 1.10 with the DB Connection Service version 1.02, select *1.10 (DBCon 1.02)* in the Version box.


Select Device

Category	Controller
Device	NJ501 - 1520
Version	1.10(DBCon 1.02)

*2 Sysmac Studio version 1.14 or higher is required to use NJ101-□□20 Database Connection CPU Units. NJ101-□□20 cannot be used with Sysmac Studio version 1.13 or lower.

Relationship between Actual Unit Version of CPU Unit and Unit Version Set in the Sysmac Studio Project

The following table shows the differences in the specifications by the combination of actual DB Connection Service version of CPU Unit and DB Connection Service version set in the Sysmac Studio project when using an NJ501-1□20 CPU Unit.

● Supported Database Type

DB Connection Service version of the CPU Unit	DB Connection Service version set in the Sysmac Studio project		
	1.00	1.01	1.02
1.02	Oracle SQL Server	Oracle SQL Server DB2 MySQL Firebird	Oracle SQL Server DB2 MySQL Firebird PostgreSQL
1.01	Oracle SQL Server	Oracle SQL Server DB2 MySQL Firebird	Transfer is not posible.
1.00	Oracle SQL Server	Transfer is not posible.	Transfer is not posible.

● Number of DB Map Variables for which a Mapping can be Created

DB Connection Service version of the CPU Unit	DB Connection Service version set in the Sysmac Studio project		
	1.00	1.01	1.02
1.02	15 variables max.	SQL Server: 60 variables max. Oracle: 30 variables max. DB2: 30 variables max. MySQL: 30 variables max. Firebird: 15 variables max.	SQL Server: 60 variables max. Oracle: 30 variables max. DB2: 30 variables max. MySQL: 30 variables max. Firebird: 15 variables max. PostgreSQL: 30 variables max.
1.01	15 variables max.	SQL Server: 60 variables max. Oracle: 30 variables max. DB2: 30 variables max. MySQL: 30 variables max. Firebird: 15 variables max.	Transfer is not posible.
1.00	15 variables max.	Transfer is not posible.	Transfer is not posible.

The following table shows the differences in the specifications by the combination of actual DB Connection Service version of CPU Unit and DB Connection Service version set in the Sysmac Studio project when using an NJ101-□□20 CPU Unit.

● Supported Database Type

DB Connection Service version of the CPU Unit	DB Connection Service version set in the Sysmac Studio project	
	1.01	1.02
1.02	Oracle SQL Server DB2 MySQL Firebird	Oracle SQL Server DB2 MySQL Firebird PostgreSQL
1.01	Oracle SQL Server DB2 MySQL Firebird	Transfer is not posible.

I

Index

Index

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 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 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 (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:

Cat. No. W527-E1-06 1215