

OMRON

Machine Automation Controller
CJ-series
User Defined CAN Unit

Operation Manual
for NJ-series CPU Unit

CJ1W-CORT21

User Defined
CAN Unit



W517-E2-01

Introduction

Thank you for purchasing a CJ-series CJ1W-CORT21 User Defined CAN Unit. This manual contains information that is necessary to use the CJ-series CJ1W-CORT21 User Defined CAN Unit for an NJ-series CPU Unit. 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 B3503.

Applicable Products

This manual covers the following products.

CJ-series CJ1W-CORT21 User Defined CAN Unit

Relevant Manuals

There are three manuals that provide basic information on the NJ-series CPU Units: the *NJ-series CPU Unit Hardware User's Manual*, the *NJ-series CPU Unit Software User's Manual*, and the *NJ-series Instructions Reference Manual*.

Most operations are performed from the Sysmac Studio Automation Software. Refer to the *Sysmac Studio Version 1 Operation Manual* (Cat. No. W504) for information on the Sysmac Studio.

Other manuals are necessary for specific system configurations and applications.

Read all of the manuals that are relevant to your system configuration and application to make the most of the NJ-series CPU Unit.

	NJ-series User's Manuals								CJ-series Special Unit Operation Manuals for NJ-series CPU Unit
	Basic information			NJ-series CPU Unit Motion Control User's Manual	NJ-series CPU Unit Built-in EtherCAT Port User's Manual	NJ-series Motion Control Instructions Reference Manual	NJ-series CPU Unit Built-in EtherNet/IP Port User's Manual	NJ-series Troubleshooting Manual	
	NJ-series CPU Unit Hardware User's Manual	NJ-series CPU Unit Software User's Manual	NJ-series Instructions Reference Manual						
Introduction to NJ-series Controllers	●								
Setting devices and hardware									
Using motion control				●					
Using EtherCAT					●				
Using EtherNet/IP							●		
Using CJ-series Units									●
Software settings									
Using motion control		●		●					
Using EtherCAT					●				
Using EtherNet/IP							●		
Programming		●	●						
Using motion control				●		●			
Using EtherCAT					●				
Using CJ-series Units									●
Programming error processing								●	
Testing operation and debugging									
Using motion control		●		●					
Using EtherCAT					●				
Using EtherNet/IP							●		
Troubleshooting and managing errors in an NJ-series Controller	△	△		△			△		△
	Use the relevant manuals for references according to any error that occurs.								
Maintenance									
Using EtherCAT	●				●				
Using EtherNet/IP							●		
Using CJ-series Units									●

Manual Configuration

NJ-series CPU Unit Hardware User's Manual (Cat. No. W500)

Section	Description
Section 1 Introduction	This section provides an introduction to the NJ-series Controllers and their features, and gives the NJ-series Controller specifications.
Section 2 System Configuration	This section describes the system configuration used for NJ-series Controllers.
Section 3 Configuration Units	This section describes the parts and functions of the configuration devices in the NJ-series Controller configuration, including the CPU Unit and Configuration Units.
Section 4 Installation and Wiring	This section describes where and how to install the CPU Unit and Configuration Units and how to wire them.
Section 5 Troubleshooting	This section describes the event codes, error confirmation methods, and corrections for errors that can occur.
Section 6 Inspection and Maintenance	This section describes the contents of periodic inspections, the service life of the Battery and Power Supply Units, and replacement methods for the Battery and Power Supply Units.
Appendices	The appendices provide the specifications of the Basic I/O Units, Unit dimensions, load short-circuit protection detection, line disconnection detection, and measures for EMC Directives.

NJ-series CPU Unit Software User's Manual (Cat. No. W501)

Section	Description
Section 1 Introduction	This section provides an introduction to the NJ-series Controllers and their features, and gives the NJ-series Controller specifications.
Section 2 CPU Unit Operation	This section describes the variables and control systems of the CPU Unit and CPU Unit status.
Section 3 I/O Ports, Slave Configuration, and Unit Configuration	This section describes how to use I/O ports, how to create the slave configuration and unit configuration and how to assign functions.
Section 4 Controller Setup	This section describes the initial settings of the function modules.
Section 5 Designing Tasks	This section describes the task system and types of tasks.
Section 6 Programming	This section describes programming, including the programming languages and the variables and instructions that are used in programming.
Section 7 Simulation, Transferring Projects to the Physical CPU Unit, and Operation	This section describes simulation of Controller operation and how to use the results of simulation.
Section 8 CPU Unit Status	This section describes CPU Unit status.
Section 9 CPU Unit Functions	This section describes the functionality provided by the CPU Unit.
Section 10 Communications Setup	This section describes how to go online with the CPU Unit and how to connect to other devices.
Section 11 Example of Actual Application Procedures	This section describes the procedures that are used to actually operate an NJ-series Controller.
Section 12 Troubleshooting	This section describes the event codes, error confirmation methods, and corrections for errors that can occur.
Appendices	The appendices provide the CPU Unit specifications, task execution times, system-defined variable lists, data attribute lists, CJ-series Unit memory information, CJ-series Unit memory allocation methods, and data type conversion information.

Sysmac Studio Version 1 Operation Manual (Cat. No. W504)

Section	Description
Section 1 Introduction	This section provides an overview and lists the specifications of the Sysmac Studio and describes its features and components.
Section 2 Installation and Uninstallation	This section describes how to install and uninstall the Sysmac Studio.
Section 3 System Design	This section describes the basic concepts for designing an NJ-series System with the Sysmac Studio and the basic operating procedures.
Section 4 Programming	This section describes how to create programs with the Sysmac Studio.
Section 5 Online Connections to a Controller	This section describes how to go online with a Controller.
Section 6 Debugging	This section describes how to debug the programs online on the Controller or debug it offline with the Simulator.
Section 7 Other Functions	This section describes Sysmac Studio functions other than system design functions.
Section 8 Reusing Programming	This section describes how to reuse the programs that you create with the Sysmac Studio.
Section 9 Support Software Provided with the Sysmac Studio	This section describes the Support Software that is provided with the Sysmac Studio.
Section 10 Troubleshooting	This section describes the error messages that are displayed when you check a program on the Sysmac Studio and how to correct those errors.
Appendices	The appendices describe the following: Driver Installation for Direct USB Cable Connection Specifying One of Multiple Ethernet Interface Cards Online Help Simulation Instructions

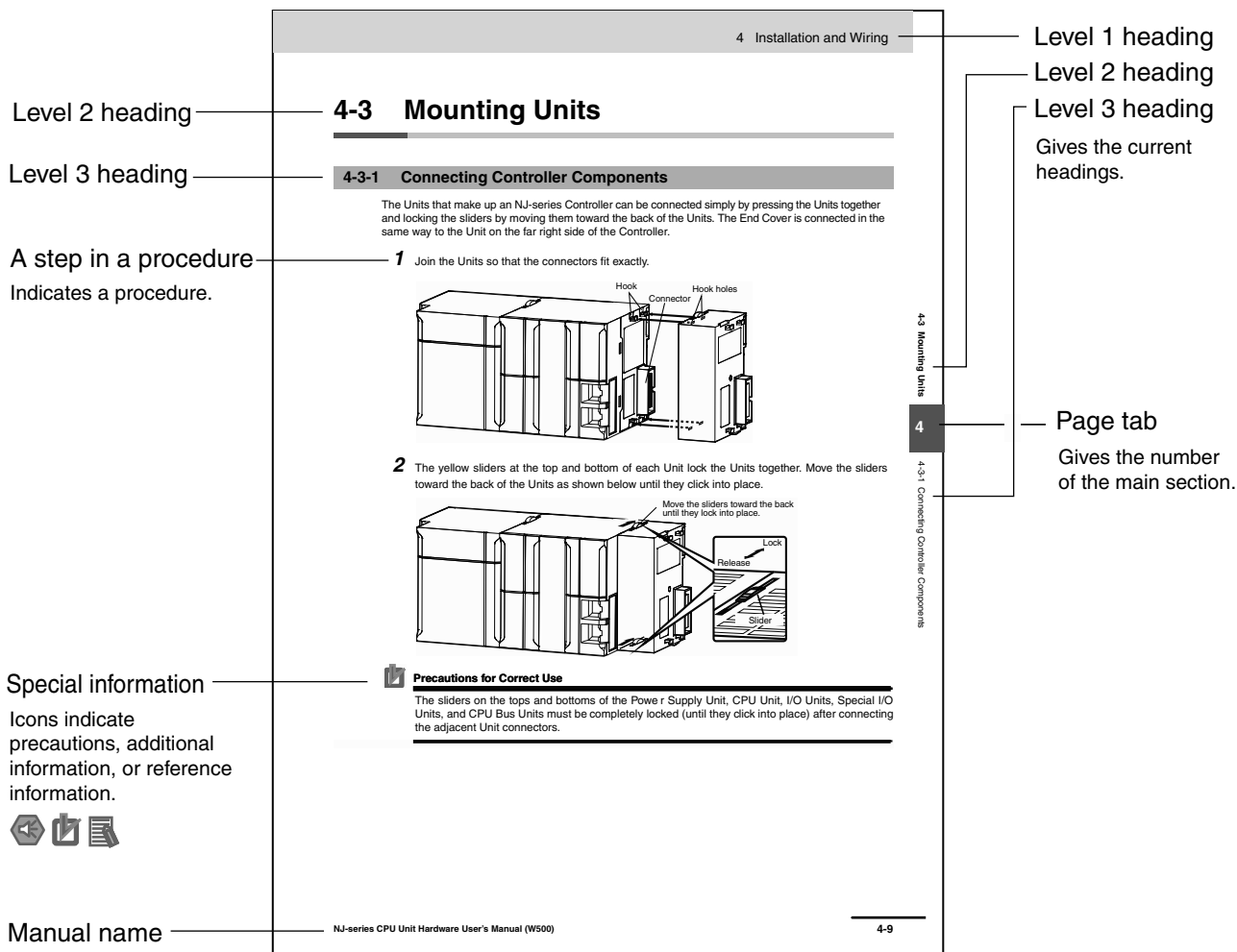
CJ-series User Defined CAN Unit Operation Manual for NJ-series CPU Unit (Cat. No. W517) (This Manual)

Section	Description
Section 1 Features and System Configuration	This section provides an introduction to the User Defined CAN Units and their features. It also describes the operating procedure and the specifications of the User Defined CAN Units.
Section 2 Nomenclature and Installation	This section describes the nomenclature, functionality and installation of the User Defined CAN Unit.
Section 3 Data Exchange with the CPU Unit	This section describes the data exchange between the CPU Unit and User Defined CAN Unit and the definitions of the device variables for CJ-series Unit.
Section 4 Message Communications	This section describes the message communications of the User Defined CAN Unit.
Section 5 Communications Timing	This section describes the communications timing of the User Defined CAN Unit and the performances of the remote I/O communications and message communications.
Section 6 Troubleshooting and Maintenance	This section describes the troubleshooting procedure, event logs and maintenance procedure for the User Defined CAN Unit.
Appendices	---

Manual Structure

Page Structure

The following page structure is used in this manual.



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.



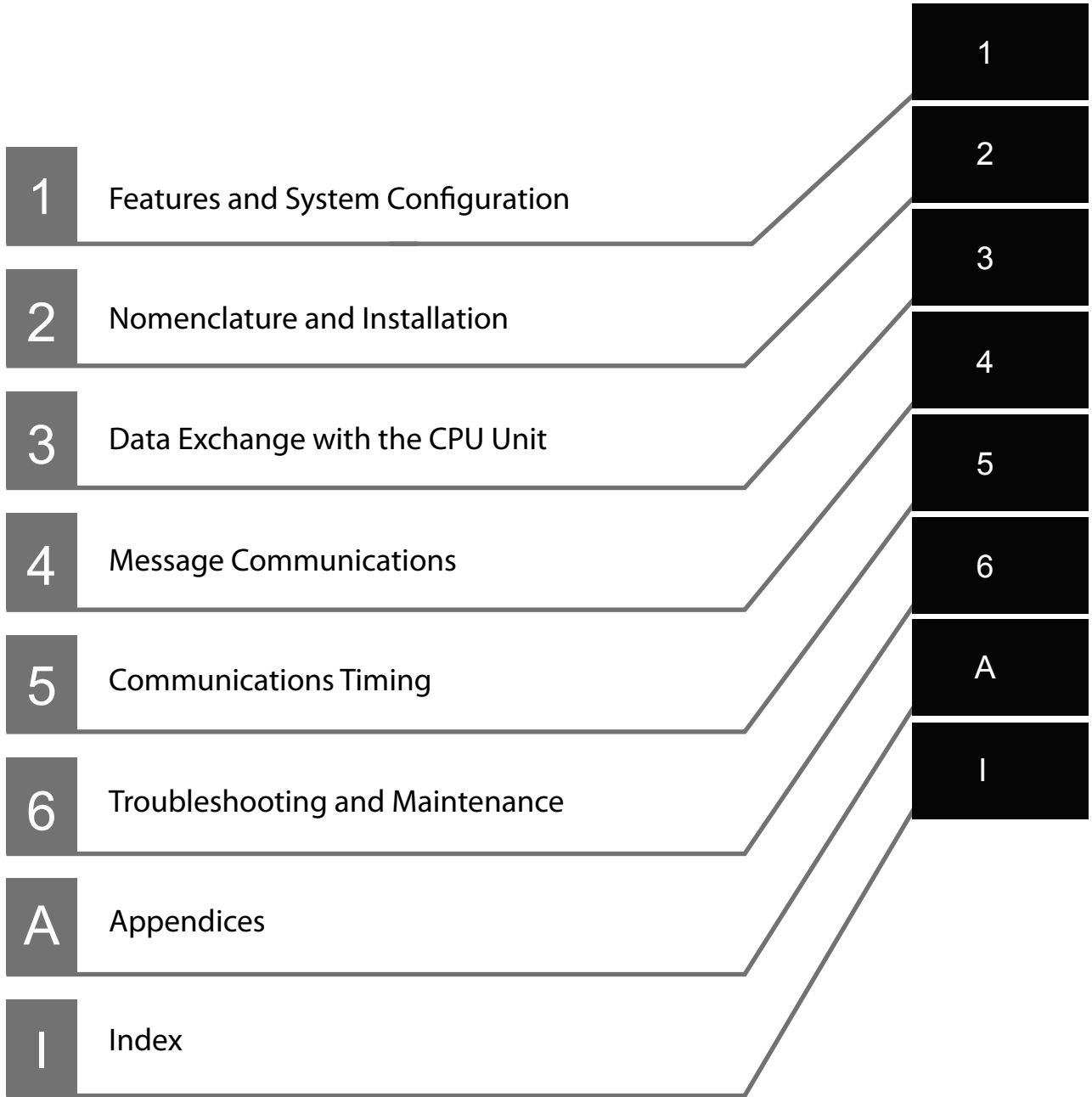
Additional Information

Additional information to read as required.

This information is provided to increase understanding or make operation easier.

Note References are provided to more detailed or related information.

Sections in this Manual



CONTENTS

Introduction.....	1
Relevant Manuals	2
Manual Configuration.....	3
Manual Structure	5
Sections in this Manual.....	6
CONTENTS.....	7
Read and Understand this Manual.....	10
Safety Precautions	13
Precautions for Safe Use	18
Precautions for Correct Use	24
Regulations and Standards	27
Unit Versions.....	29
Related Manuals	31
Revision History	32

Section 1 Features and System Configuration

1-1	User Defined CAN Unit Features	1-2
1-2	Overview of CAN Bus	1-5
1-2-1	CAN Communication Protocol	1-6
1-2-2	Physical CAN Connection	1-10
1-2-3	Principles of Data Exchange	1-11
1-2-4	Principles of Non-Destructive Bitwise Arbitration	1-11
1-2-5	Efficiency of Bus Allocation	1-12
1-2-6	Message Frame Formats	1-14
1-2-7	Detecting and Signaling Errors	1-15
1-2-8	Data Reliability of the CAN Protocol	1-17
1-2-9	Extended Format CAN Message	1-18
1-2-10	Implementations of the CAN Protocol	1-19
1-2-11	Configuring a CAN Network	1-19
1-3	Specifications	1-20
1-3-1	User Defined CAN Unit	1-20
1-4	Basic Operating Procedures	1-22
1-4-1	Network Installation Procedure	1-22
1-4-2	User Defined CAN Unit Startup Procedure	1-23

Section 2 Nomenclature and Installation

2-1	Nomenclature and Installation	2-2
2-1-1	Nomenclature and Functions	2-2
2-1-2	Switch Settings	2-5
2-2	Installing the User Defined CAN Unit	2-8
2-2-1	System Configuration Precautions	2-8
2-2-2	Mounting	2-8
2-2-3	Handling Precautions	2-9
2-2-4	External Dimensions	2-9
2-2-5	Unit States	2-10

Section 3 Data Exchange with the CPU Unit

3-1	Data Exchange with the CPU Unit	3-2
3-1-1	Data Flow	3-2
3-1-2	Accessing From the User Program	3-5
3-2	Device Variable for CJ-series Unit	3-8
3-2-1	Enable CAN Communications	3-8
3-2-2	Status Communication	3-9
3-2-3	Number of Delayed Messages	3-12
3-2-4	Number of Received Messages Waiting to be Processed	3-12

Section 4 Message Communications

4-1	Overview	4-2
4-1-1	Outline of Message Communications	4-2
4-2	Unit Configuration and Control	4-4
4-2-1	Configure Memory Areas (2902)	4-4
4-2-2	Configure 11-Bit ID Output Message Buffer (2903)	4-10
4-2-3	Configure 29-Bit ID Output Message Buffer (2904)	4-12
4-2-4	Configure 11-Bit ID Input Message Buffer (2905)	4-14
4-2-5	Configure 29-Bit ID Input Message Buffer (2906)	4-16
4-2-6	Direct Transmit of an 11-bit ID CAN Message (2907)	4-17
4-2-7	Direct Transmit of an 29-bit ID CAN Message (2908)	4-19
4-2-8	Setting the CAN Bit Rate and Sample Point (2909)	4-21
4-2-9	Error Log Read (2102)	4-22
4-2-10	Error Log Clear (2103)	4-23

Section 5 Communications Timing

5-1	Performance	5-2
5-1-1	I/O Refresh Time	5-2
5-1-2	Output Message Evaluation Time	5-2
5-1-3	Input Message Processing Time	5-4
5-1-4	CAN Interface	5-5
5-1-5	I/O Response Time	5-5
5-1-6	Transmission of CAN Messages	5-8
5-1-7	Reception of CAN Messages	5-9

Section 6 Troubleshooting and Maintenance

6-1	Overview	6-2
6-1-1	Troubleshooting the User Defined CAN Unit	6-2
6-2	Troubleshooting with the User Defined CAN Unit Indicators	6-4
6-2-1	Run LED Indicator	6-4
6-2-2	ERR LED Indicator	6-5
6-2-3	Two 7-segment Display	6-5
6-2-4	Two Dot Indicators	6-6
6-3	Error Log Functions	6-7
6-3-1	Error Log Table	6-7
6-3-2	Error Codes and Detail Codes	6-8
6-3-3	Status Information	6-8
6-4	Troubleshooting	6-10
6-4-1	CPU Unit's ERR/ALM Indicator Lit or Flashing	6-10
6-5	Event Logs	6-11
6-5-1	Overview of the Event Logs	6-11
6-5-2	Error Table	6-12
6-5-3	Error Descriptions	6-12
6-6	Maintenance and Replacement	6-16
6-6-1	Cleaning	6-16
6-6-2	Inspection	6-16
6-6-3	Replacing Faulty Units	6-17

Appendices

A-1	Differences in Available Functions Depending on the CPU Unit (NJ/CJ-series) to be Connected	A-2
A-1-1	Differences in Available Functions	A-2
A-1-2	Differences in Accessing from User Program	A-2
A-2	User Program Example	A-4

Index

Read and Understand this Manual

Please read and understand this manual before using the product. Please consult your OMRON representative if you have any questions or comments.

Warranty and Limitations of Liability

WARRANTY

OMRON's exclusive warranty is that the products are free from defects in materials and workmanship for a period of one year (or other period if specified) from date of sale by OMRON.

OMRON MAKES NO WARRANTY OR REPRESENTATION, EXPRESS OR IMPLIED, REGARDING NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR PARTICULAR PURPOSE OF THE PRODUCTS. ANY BUYER OR USER ACKNOWLEDGES THAT THE BUYER OR USER ALONE HAS DETERMINED THAT THE PRODUCTS WILL SUITABLY MEET THE REQUIREMENTS OF THEIR INTENDED USE. OMRON DISCLAIMS ALL OTHER WARRANTIES, EXPRESS OR IMPLIED.

LIMITATIONS OF LIABILITY

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

In no event shall the responsibility of OMRON for any act exceed the individual price of the product on which liability is asserted.

IN NO EVENT SHALL OMRON BE RESPONSIBLE FOR WARRANTY, REPAIR, OR OTHER CLAIMS 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 OR REPAIR.

Application Considerations

SUITABILITY FOR USE

OMRON shall not be responsible for conformity with any standards, codes, or regulations that apply to the combination of products in the customer's application or use of the products.

At the customer's request, OMRON will provide applicable third party certification documents identifying ratings and limitations of use that apply to the products. This information by itself is not sufficient for a complete determination of the suitability of the products in combination with the end product, machine, system, or other application or use.

The following are some examples of applications for which particular attention must be given. This is not intended to be an exhaustive list of all possible uses of the products, nor is it intended to imply that the uses listed may be suitable for the products:

- Outdoor use, uses involving potential chemical contamination or electrical interference, or conditions or uses not described in this manual.
- Nuclear energy control systems, combustion systems, railroad systems, aviation systems, medical equipment, amusement machines, vehicles, safety equipment, and installations subject to separate industry or government regulations.
- Systems, machines, and equipment that could present a risk to life or property.

Please know and observe all prohibitions of use applicable to the products.

NEVER USE THE PRODUCTS 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 PRODUCTS ARE PROPERLY RATED AND INSTALLED FOR THE INTENDED USE WITHIN THE OVERALL EQUIPMENT OR SYSTEM.

PROGRAMMABLE PRODUCTS

OMRON shall not be responsible for the user's programming of a programmable product, or any consequence thereof.

Disclaimers

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 model numbers when published ratings or features are changed, or when significant construction changes are made. However, some specifications of the products may be changed without any notice. When in doubt, special model numbers may be assigned to fix or establish key specifications for your application on your request. Please consult with your OMRON representative at any time to confirm actual specifications of purchased products.

DIMENSIONS AND WEIGHTS

Dimensions and weights are nominal and are not to be used for manufacturing purposes, even when tolerances are shown.

PERFORMANCE DATA

Performance data given in this manual 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 users must correlate it to actual application requirements. Actual performance is subject to the OMRON Warranty and Limitations of Liability.

ERRORS AND OMISSIONS


The information in this manual has been carefully checked and is believed to be accurate; however, no responsibility is assumed for clerical, typographical, or proofreading errors, or omissions.


Safety Precautions

Definition of Precautionary Information

The following notation is used in this manual to provide precautions required to ensure safe usage of an NJ-series Controller. 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.

 WARNING	Indicates a potentially hazardous situation which, if not avoided, could result in death or serious injury. Additionally, there may be severe property damage.
--	--

 Caution	Indicates a potentially hazardous situation which, if not avoided, may result in minor or moderate injury, or property damage.
--	--



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.

Symbols



The circle and slash symbol indicates operations that you must not do. The specific operation is shown in the circle and explained in text. This example indicates prohibiting disassembly.



The triangle symbol indicates precautions (including warnings). The specific operation is shown in the triangle and explained in text. This example indicates a precaution for electric shock.



The triangle symbol indicates precautions (including warnings). The specific operation is shown in the triangle and explained in text. This example indicates a general precaution.

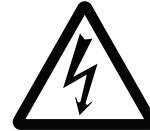


The filled circle symbol indicates operations that you must do. The specific operation is shown in the circle and explained in text. This example shows a general precaution for something that you must do.

WARNING

During Power Supply

Do not touch any of the terminals or terminal blocks while the power is being supplied. Doing so may result in electric shock.



Do not attempt to take any Unit apart. In particular, high-voltage parts are present in the Power Supply Unit while power is supplied or immediately after power is turned OFF. Touching any of these parts may result in electric shock. There are sharp parts inside the Unit that may cause injury.



Fail-safe Measures

Provide safety measures in external circuits to ensure safety in the system if an abnormality occurs due to malfunction of the CPU Unit, other Units, or slaves or due to other external factors affecting operation. Not doing so may result in serious accidents due to incorrect operation.



Emergency stop circuits, interlock circuits, limit circuits, and similar safety measures must be provided in external control circuits.



The Controller outputs may remain ON or OFF due to deposition or burning of the output relays or destruction of the output transistors. As a countermeasure for such problems, external safety measures must be provided to ensure safe operation of the system.



The CPU Unit will turn OFF all outputs from Basic Output Units in the following cases.

- If an error occurs in the power supply
- If the power supply connection becomes faulty
- If a CPU watchdog timer error or CPU reset occurs
- If a major fault level Controller error occurs
- While the CPU Unit is on standby until RUN mode is entered after the power is turned ON



External safety measures must be provided to ensure safe operation of the system even if the outputs turn OFF.

If external power supplies for slaves or other devices are overloaded or short-circuited, the voltage will drop, outputs will turn OFF, and the system may be unable to read inputs. Provide external safety measures in controls with monitoring of external power supply voltage as required so that the system operates safely in such a case.



WARNING

Fail-safe Measures

Unintended outputs may occur when an error occurs in variable memory or in memory used for CJ-series Units. As a countermeasure for such problems, external safety measures must be provided to ensure safe operation of the system.



Provide measures in the communications system and user program to ensure safety in the overall system even if errors or malfunctions occur in data link communications or remote I/O communications.



If there is interference in remote I/O communications or if a major fault level error occurs, output status will depend on the products that are used. Confirm the operation that will occur when there is interference in communications or a major fault level error, and implement safety measures. Correctly set all of the EtherCAT slaves.



The NJ-series Controller continues normal operation for a certain period of time when a momentary power interruption occurs. This means that the NJ-series Controller may receive incorrect signals from external devices that are also affected by the power interruption. Accordingly, take suitable actions, such as external fail-safe measures and interlock conditions, to monitor the power supply voltage of the external device as required.



You must take fail-safe measures to ensure safety in the event of incorrect, missing, or abnormal signals caused by broken signal lines, momentary power interruptions, or other causes. Not doing so may result in serious accidents due to incorrect operation.



Voltage and Current Inputs

Make sure that the voltages and currents that are input to the Units and slaves are within the specified ranges. Inputting voltages or currents that are outside of the specified ranges may cause accidents or fire.



Downloading

Always confirm safety at the destination before you transfer a user program, configuration data, setup data, device variables, or values in memory used for CJ-series Units from the Sysmac Studio. The devices or machines may perform unexpected operation regardless of the operating mode of the CPU Unit.



Caution

Application

Do not touch any Unit when power is being supplied or immediately after the power supply is turned OFF. Doing so may result in burn injury.



Wiring

Be sure that all terminal screws and cable connector screws are tightened to the torque specified in the relevant manuals. The loose screws may result in fire or malfunction.



Online Editing

Execute online editing only after confirming that no adverse effects will be caused by deviations in the timing of I/O. If you perform online editing, the task execution time may exceed the task period, I/O may not be refreshed with external devices, input signals may not be read, and output timing may change.



Precautions for Safe Use

Disassembly and Dropping

- Do not attempt to disassemble, repair, or modify any Units. Doing so may result in malfunction or fire.
- Do not drop any Unit or subject it to abnormal vibration or shock. Doing so may result in Unit malfunction or burning.

Mounting

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

Installation

- Always connect to a ground of 100 Ω or less when installing the Units. A ground of 100 Ω or less must be installed when shorting the GR and LG terminals on the Power Supply Unit.

Wiring

- Follow the instructions in this manual to correctly perform wiring. Double-check all wiring and switch settings before turning ON the power supply.
- Use crimp terminals for wiring. Do not connect bare stranded wires directly to terminals.
- Do not pull on the cables or bend the cables beyond their natural limit. Do not place heavy objects on top of the cables or other wiring lines. Doing so may break the cables.
- Mount terminal blocks and connectors only after checking the mounting location carefully.
- Be sure that the terminal blocks, expansion cables, and other items with locking devices are properly locked into place.
- Always remove any dustproof labels that are on the top of the Units when they are shipped before you turn ON the power supply. If the labels are not removed, heat will accumulate and malfunctions may occur.
- Before you connect a computer to the CPU Unit, disconnect the power supply plug of the computer from the AC outlet. Also, if the computer has an FG terminal, make the connections so that the FG terminal has the same electrical potential as the FG (GR) terminal on the Power Supply Unit. A difference in electric potential between the computer and Controller may cause failure or malfunction.
- If the external power supply to an Output Unit or slave has polarity, connect it with the correct polarity. If the polarity is reversed, current may flow in the reverse direction and damage the connected devices regardless of the operation of the Controller.

Power Supply Design

- Do not exceed the rated supply capacity of the Power Supply Units in the NJ-series Controller. The rated supply capacities are given in the *NJ-series CPU Unit Hardware User's Manual* (Cat. No. W500).
If the capacity is exceeded, operation may stop, malfunctions may occur, or data may not be backed up normally for power interruptions.
Use NJ-series Power Supply Units for both the NJ-series CPU Rack and Expansion Racks.
Operation is not possible if a CJ-series Power Supply Unit is used with an NJ-series CPU Unit or an NJ-series Power Supply Unit is used with a CJ-series CPU Unit.

- Do not apply voltages or connect loads to the Output Units or slaves in excess of the maximum ratings.
- Surge current occurs when the power supply is turned ON. When selecting fuses or breakers for external circuits, consider the above precaution and allow sufficient margin in shut-off performance. Refer to the relevant manuals for surge current specifications. Refer to the *NJ-series CPU Unit Hardware User's Manual* (Cat. No. W500) for surge current specifications.
- If the full dielectric strength voltage is applied or turned OFF using the switch on the tester, the generated impulse voltage may damage the Power Supply Unit. Use the adjustment on the tester to gradually increase and decrease the voltage.
- Apply the voltage between the Power Supply Unit's L1 or L2 terminal and the GR terminal when testing insulation and dielectric strength. You do not have to disconnect the LG and GR terminals to perform these tests.
- Do not supply AC power from an inverter or other device with a square-wave output. Internal temperature rise may result in smoking or burning. Always input a sinusoidal wave with the frequency that is given in the *NJ-series CPU Unit Hardware User's Manual* (Cat. No. W500).
- Install external breakers and take other safety measures against short-circuiting in external wiring.

Turning ON the Power Supply

- It takes up to approximately 10 to 20 s to enter RUN mode after the power is turned ON. During that time, outputs will be OFF or will be the values specified in the Unit or slave settings, and external communications cannot be performed. Use the RUN output on the Power Supply Unit, for example, to implement fail-safe circuits so that external devices do not operate incorrectly.
- Configure the external circuits so that the power supply to the control system turns ON only after the power supply to the Controller has turned ON. If the power supply to the Controller is turned ON after the control power supply, temporary errors may result in incorrect control system signals because the output terminals on Output Units may momentarily turn ON when power supply is turned ON to the Controller.

Actual Operation

- Check the user program, data, and parameter settings for proper execution before you use them for actual operation.

Turning OFF the Power Supply

- Never turn OFF the power supply to the Controller when the BUSY indicator is flashing. While the BUSY indicator is lit, the user program and settings in the CPU Unit are being backed up in the built-in non-volatile memory. This data will not be backed up correctly if the power supply is turned OFF. Also, a major fault level Controller error will occur the next time you start operation, and operation will stop.
- Do not turn OFF the power supply or remove the SD Memory Card while SD Memory Card access is in progress (i.e., while the SD BUSY indicator flashes). Data may become corrupted, and the Controller will not operate correctly if it uses corrupted data. To remove the SD Memory Card from the CPU Unit while the power supply is ON, press the SD Memory Card power supply switch and wait for the SD BUSY indicator to turn OFF before you remove the SD Memory Card.
- Do not disconnect the cable or turn OFF the power supply to the Controller when downloading data or the user program from Support Software.
- Always turn OFF the power supply to the Controller before you attempt any of the following.
 - Mounting or removing I/O Units or the CPU Unit
 - Assembling the Units
 - Setting DIP switches or rotary switches
 - Connecting cables or wiring the system
 - Connecting or disconnecting the connectors

The Power Supply Unit may continue to supply power to the rest of the Controller for a few seconds after the power supply turns OFF. The PWR indicator is lit during this time. Confirm that the PWR indicator is not lit before you perform any of the above.

Operation

- Confirm that no adverse effect will occur in the system before you attempt any of the following.
 - Changing the operating mode of the CPU Unit (including changing the setting of the Operating Mode at Startup)
 - Changing the user program or settings
 - Changing set values or present values
 - Forced refreshing
- Always sufficiently check the safety at the connected devices before you change the settings of an EtherCAT slave or Special Unit.
- If two different function modules are used together, such as when you use CJ-series Basic Output Units and EtherCAT slave outputs, take suitable measures in the user program and external controls to ensure that safety is maintained in the controlled system if one of the function modules stops. The relevant outputs will stop if a partial fault level error occurs in one of the function modules.
- Always confirm safety at the connected equipment before you reset Controller errors with an event level of partial fault or higher for the EtherCAT Master Function Module.
When the error is reset, all slaves that were in any state other than Operational state due to a Controller error with an event level of partial fault or higher (in which outputs are disabled) will go to Operational state and the outputs will be enabled.
Before you reset all errors, confirm that no Controller errors with an event level of partial fault have occurred for the EtherCAT Master Function Module.
- Always confirm safety at the connected equipment before you reset Controller errors for a CJ-series Special Unit. When a Controller error is reset, the Unit where the Controller error with an event level of observation or higher will be restarted.
Before you reset all errors, confirm that no Controller errors with an event level of observation or higher have occurred for the CJ-series Special Unit. Observation level events do not appear on the Controller Error Tab Page, so it is possible that you may restart the CJ-series Special Unit without intending to do so.
You can check the status of the `_CJB_UnitErrSta[0,0]` to `_CJB_UnitErrSta[3,9]` error status variables on a Watch Tab Page to see if an observation level Controller error has occurred.

Battery Backup

- The user program and initial values for the variables are stored in non-volatile memory in the CPU Unit. The present values of variables with the Retain attribute and the values of the Holding, DM, and EM Areas in the memory used for CJ-series Units are backed up by a Battery. If the Battery is not connected or the Battery is exhausted, the CPU Unit detects a Battery-backup Memory Check Error. If that error is detected, variables with a Retain attribute are set to their initial values and the Holding, DM, and EM Areas in memory used for CJ-series Units are cleared to all zeros. Perform thorough verifications and provide sufficient measures to ensure that the devices perform safe operation for the initial values of the variables with Retain attributes and the resulting operation.

Debugging

- Forced refreshing ignores the results of user program execution and refreshes I/O with the specified values. If forced refreshing is used for inputs for which I/O refreshing is not supported, the inputs will first take the specified values, but they will then be overwritten by the user program. This operation differs from the force-set/reset functionality of the CJ-series PLCs.

- You cannot upload or download information for forced refreshing with the Sysmac Studio. After downloading data that contains forced refreshing, change to RUN mode and then use the Sysmac Studio to perform the operation for forced refreshing. Depending on the difference in the forced status, the control system may operate unexpectedly.
- Do not specify the same address for the AT specification for more than one variable. Doing so would allow the same entity to be accessed with different variable names, which would make the user program more difficult to understand and possibly cause programming mistakes.

General Communications

- When you use data link communications, check the error information given in the status flags to make sure that no error has occurred in the source device. Write the user program to use the received data only if there is no error. If there is an error in the source device, the data for the data link may contain incorrect values.
- Unexpected operation may result if inappropriate data link tables are set. Even if appropriate data link tables have been set, confirm that the controlled system will not be adversely affected before you transfer the data link tables. The data links start automatically after the data link tables are transferred.
- All CPU Bus Units are restarted when routing tables are transferred from Support Software to the CPU Unit. Restarting these Units is required to read and enable the new routing tables. Confirm that the system will not be adversely affected by restarting before you transfer the routing tables.
- Tag data links will stop between related nodes while tag data link parameters are transferred during Controller operation. Confirm that the system will not be adversely affected before you transfer the tag data link parameters.

EtherNet/IP Communications

- All related EtherNet/IP nodes are reset when you transfer settings for the built-in EtherNet/IP port (including IP addresses and tag data links settings). This is performed to read and enable the settings. Confirm that the system will not be adversely affected by resetting nodes before you transfer the settings.
- If EtherNet/IP tag data links (cyclic communications) are used with a repeating hub, the communications load on the network will increase. This will increase collisions and may prevent stable communications. Do not use repeating hubs on networks where tag data links are used. Use an Ethernet switch instead.

EtherCAT Communications

- Make sure that the communications distance, number of nodes connected, and method of connection for EtherCAT are within specifications. Do not connect EtherCAT communications to EtherNet/IP, a standard in-house LAN, or other networks. An overload may cause the network to fail or malfunction.
- Malfunctions or unexpected operation may occur for some combinations of EtherCAT revisions of the master and slaves. If you disable the revision check in the network settings, use the Sysmac Studio to check the slave revision settings in the master and the actual slave revisions, and then make sure that functionality is compatible in the slave manuals or other references. You can check the actual slave revisions from the Sysmac Studio or on slave nameplates.
- After you transfer the user program, the CPU Unit is restarted. Communications with the EtherCAT slaves are cut off for up to 45 seconds. During that period, the slave outputs behave according to the slave settings. Before you transfer the user program, confirm that the system will not be adversely affected.
- If the Fail-soft Operation parameter is set to stop operation, process data communications will stop for all slaves when an EtherCAT communications error is detected in a slave. For this reason, if Servo Drives are connected, the Servos for all axes will be turned OFF. Make sure that the Fail-soft Operation parameter setting results in safe operation when a device error occurs.

- EtherCAT communications are not always established immediately after the power supply is turned ON. Use the system-defined variables in the user program to confirm that communications are established before attempting control operations.
- If frames sent to EtherCAT slaves are lost due to noise or other causes, slave I/O data is not communicated, and the intended operation is sometimes not achieved. If noise countermeasures are required, use the `_EC_InDataInvalid` (Input Data Disable) system-defined variable as an interlock condition in the user program.
Refer to the *NJ-series CPU Unit Built-in EtherCAT Port User's Manual* (Cat. No. W505) for details. The slave outputs behave according to the slave settings. Refer to the manuals for the slaves for details.
- When an EtherCAT slave is disconnected, communications will stop and control of the outputs will be lost not only for the disconnected slave, but for all slaves connected after it. Confirm that the system will not be adversely affected before you disconnect a slave.
- If you disconnect the cable from an EtherCAT slave to disconnect it from the network, any current communications frames may be lost. If frames are lost, slave I/O data is not communicated, and the intended operation is sometimes not achieved. Perform the following processing for a slave that needs to be replaced.
 - Program the `_EC_InDataInvalid` (Input Data Disable) system-defined variable as an interlock condition.
 - Set the Impermissible Number of Continuous Timeouts setting in the EtherCAT master to at least 2.
 Refer to the *NJ-series CPU Unit Built-in EtherCAT Port User's Manual* (Cat. No. W505) for details.

Motion Control

- Confirm the axis number carefully before you perform an MC Test Run.
- The motor is stopped if communications are interrupted between the Sysmac Studio and the CPU Unit during an MC Test Run. Connect the communications cable between the computer and CPU Unit securely and confirm that the system will not be adversely affected before you perform an MC Test Run.
- Always execute the Save Cam Table instruction if you change any of the cam data from the user program in the CPU Unit or from the Sysmac Studio. If the cam data is not saved, the previous condition will be restored when the power is turned ON again, possibly causing unexpected machine operation.
- The positive drive prohibit input (POT), negative drive prohibit input (NOT), and home proximity input (DEC) of the Servo Drive are used by the MC Function Module as the positive limit input, negative limit input, and home proximity input. Make sure that the signal widths for all of these input signals are longer than the control period of the MC Function Module. If the input signal widths are shorter than the control period, the MC Function Module may not be able to detect the input signals, resulting in incorrect operation.

Battery Replacement

- The Battery may leak, rupture, heat, or ignite. Never short-circuit, charge, disassemble, heat, or incinerate the Battery or subject it to strong shock.
- Dispose of any Battery that has been dropped on the floor or otherwise subjected to excessive shock. Batteries that have been subjected to shock may leak if they are used.
- UL standards require that only an experienced engineer replace the Battery. Make sure that an experienced engineer is in charge of Battery replacement.
- Apply power for at least five minutes before changing the Battery. Install a new Battery within five minutes (at 25°C) of turning OFF the power supply. If power is not supplied for at least 5 minutes, the saved data may be lost.

Unit Replacement

- We recommend replacing the Battery with the power turned OFF to prevent the CPU Unit's sensitive internal components from being damaged by static electricity and to prevent malfunctions. The Battery can be replaced without turning OFF the power supply. To do so, always touch a grounded piece of metal to discharge static electricity from your body before you start the procedure. After you replace the Battery, connect the Sysmac Studio and clear the Low Battery Voltage error.
- Make sure that the required data, including the user program, configurations, settings, variables, and memory used for CJ-series Units, is transferred to a CPU Unit that was replaced and to externally connected devices before restarting operation.
Be sure to include the routing tables, network parameters, and other CPU Bus Unit data, which are stored in the CPU Unit.

Disposal

- Dispose of the product and Batteries according to local ordinances as they apply.



廢電池請回收

- The following information must be displayed for all products that contain primary lithium batteries with a perchlorate content of 6 ppb or higher when shipped to or transported through the State of California, USA.
Perchlorate Material - special handling may apply.
See www.dtsc.ca.gov/hazardouswaste/perchlorate.
- The CPU Unit contains a primary lithium battery with a perchlorate content of 6 ppb or higher. Place the above information on the individual boxes and shipping boxes when shipping finished products that contain a CPU Unit to the State of California, USA.

Using the User Defined CAN Units

- When adding a new node to the network, make sure that the baud rate is the same as other nodes.
- Use specified communications cables.
- Do not extend connection distances beyond the ranges given in the specifications.

Precautions for Correct Use

Storage, Mounting, and Wiring

- Do not operate or store the Controller in the following locations. Operation may stop or malfunctions may occur.
 - Locations subject to direct sunlight
 - Locations subject to temperatures or humidity outside the range specified in the specifications
 - Locations subject to condensation as the result of severe changes in temperature
 - Locations subject to corrosive or flammable gases
 - Locations subject to dust (especially iron dust) or salts
 - Locations subject to exposure to water, oil, or chemicals
 - Locations subject to shock or vibration
- Take appropriate and sufficient countermeasures when installing the Controller in the following locations.
 - Locations subject to strong, high-frequency noise
 - Locations subject to static electricity or other forms of noise
 - Locations subject to strong electromagnetic fields
 - Locations subject to possible exposure to radioactivity
 - Locations close to power lines
- Before touching a Unit, be sure to first touch a grounded metallic object in order to discharge any static build-up.
- Install the Controller away from sources of heat and ensure proper ventilation. Not doing so may result in malfunction, in operation stopping, or in burning.
- An I/O bus check error will occur and the Controller will stop if an I/O Connecting Cable's connector is disconnected from the Rack. Be sure that the connectors are secure.
- Do not allow foreign matter to enter the openings in the Unit. Doing so may result in Unit burning, electric shock, or failure.
- Do not allow wire clippings, shavings, or other foreign material to enter any Unit. Otherwise, Unit burning, failure, or malfunction may occur. Cover the Units or take other suitable countermeasures, especially during wiring work.
- For EtherCAT and EtherNet/IP, use the connection methods and cables that are specified in the *NJ-series CPU Unit Built-in EtherCAT Port User's Manual* (Cat. No. W505) and the *NJ-series CPU Unit Built-in EtherNet/IP Port User's Manual* (Cat. No. W506). Otherwise, communications may be faulty.
- Use the rated power supply voltage for the Power Supply Units. Take appropriate measures to ensure that the specified power with the rated voltage and frequency is supplied in places where the power supply is unstable.
- Make sure that the current capacity of the wire is sufficient. Otherwise, excessive heat may be generated. When cross-wiring terminals, the total current for all the terminals will flow in the wire. When wiring cross-overs, make sure that the current capacity of each of the wires is not exceeded.
- Do not touch the terminals on the Power Supply Unit immediately after turning OFF the power supply. Residual voltage may cause electrical shock.
- If you use reed switches for the input contacts for AC Input Units, use switches with a current capacity of 1 A or greater.
If the capacity of the reed switches is too low, surge current may fuse the contacts.

Error Processing

- In applications that use the results of instructions that read the error status, consider the affect on the system when errors are detected and program error processing accordingly. For example, even the detection of a minor error, such as Battery replacement during operation, can affect the system depending on how the user program is written.

Unit Replacement

- If you replace a CPU Bus Unit or Special I/O Unit, refer to operation manual for the Unit for information on the data required for individual Units and redo the necessary settings.
- The absolute encoder home offset is backed up with a Battery in the CPU Unit.
When you change the combination of the CPU Unit and Servomotor, e.g., when you add or replace a Servomotor, define home again.
To restore the information without changing the CPU Unit-Servomotor combination, remove the absolute encoder home offset from the data to restore.

Task Settings

- If a Task Period Exceeded error occurs, shorten the programs to fit in the task period or increase the setting of the task period.

Motion Control

- Use the system-defined variable in the user program to confirm that EtherCAT communications are established before you attempt to execute motion control instructions. Motion control instructions are not executed normally if EtherCAT communications are not established.
- Use the system-defined variables to monitor for errors in communications with the slaves that are controlled by the motion control function module. Motion control instructions are not executed normally if an error occur in slave communications.
- Before you start an MC Test Run, make sure that the operation parameters are set correctly.
- Do not download motion control settings during an MC Test Run.

EtherCAT Communications

- Do not disconnect the EtherCAT slave cables during operation. The outputs will become unstable.
- Set the Servo Drives to stop operation if an error occurs in EtherCAT communications between the Controller and a Servo Drive.

Battery Replacement

- Be sure to install a replacement Battery within two years of the production date shown on the Battery label.
- Turn ON the power after replacing the Battery for a CPU Unit that has been unused for a long time. Leaving the CPU Unit unused again without turning ON the power even once after the Battery is replaced may result in a shorter Battery life.
- When you replace the Battery, use the CJ1W-BAT01 Battery Set.

SD Memory Cards

- Insert the SD Memory Card all the way.
- Do not turn OFF the power supply to the Controller during SD Memory Card access. The files may be corrupted.

If there is a corrupted file in the SD Memory Card, the file is automatically deleted by the restoration function when the power supply is turned ON.

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.

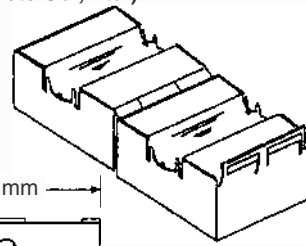
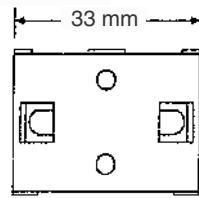
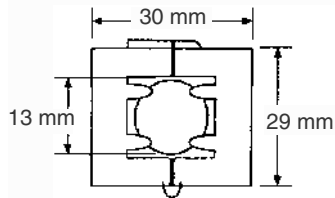
You must therefore confirm that the overall machine or equipment complies with EC Directives.

The following examples show means of reducing noise.

- 1** Noise from the communications cable can be reduced by installing a ferrite core on the communications cable within 10 cm of the User Defined CAN Unit.

Ferrite Core (Data Line Filter): 0443-164151 (manufactured by Fair-Rite Products Co., Ltd.)


Impedance specifications
25 MHZ: 156 Ω
100 MHZ: 250 Ω



[Contact]
Nisshin Electric Co., Ltd.

- 2** Wire the control panel with as thick and short electric lines as possible and ground to 100 Ω min.
- 3** Keep communication cables as short as possible and ground to 100 Ω min.

Trademarks

- Sysmac and SYSMAC are trademarks or registered trademarks of OMRON Corporation in Japan and other countries for OMRON factory automation products.
- Windows, Windows 98, Windows XP, Windows Vista, and Windows 7 are registered trademarks of Microsoft Corporation in the USA and other countries.
- EtherCAT® is a registered trademark of Beckhoff Automation GmbH for their patented technology.
- The SD logo is a trademark of SD-3C, LLC. 
- CAN Protocol is developed by Robert Bosch GmbH and protected by patents.
- SAE is the trademark of The Society of Automotive Engineers.
- CiA is the trademark of CAN in Automation(CiA), CiA is the international users' and manufacturers' organization that develops and supports CAN-based higher-layer protocols.

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

Unit Versions

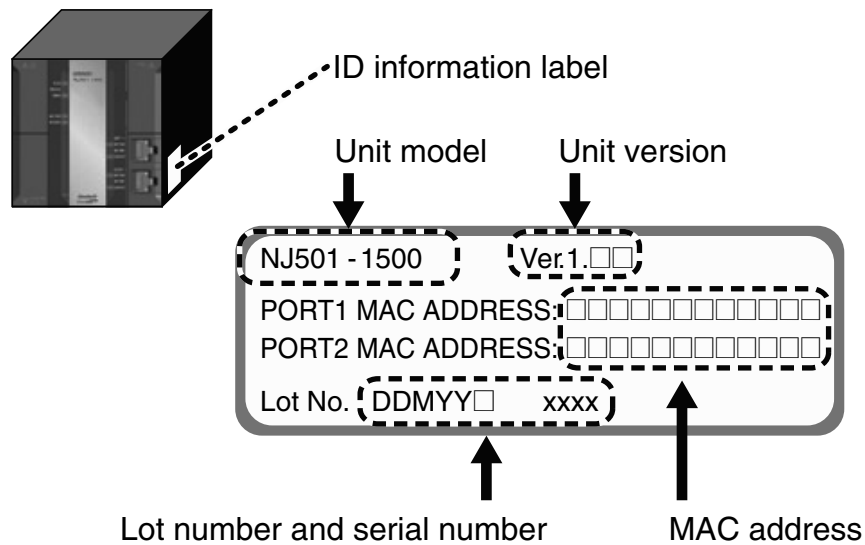
Unit Versions

A “unit version” has been introduced to manage CPU Units in the NJ Series according to differences in functionality accompanying Unit upgrades.

Notation of Unit Versions on Products

The unit version is given on the ID information label of the products for which unit versions are managed, as shown below.

Example for NJ-series NJ501-□□□□ CPU Unit:



The following information is provided on the ID information label.

Item	Description
Unit model	Gives the model of the Unit.
Unit version	Gives the unit version of the Unit.
Lot number and serial number	Gives the lot number and serial number of the Unit. DDMY□: Lot number, □: For use by OMRON, xxxx: Serial number “M” gives the month (1 to 9: January to September, X: October, Y: November, Z: December)
MAC address	Gives the MAC address of the built-in port on the Unit.

Confirming Unit Versions with Sysmac Studio

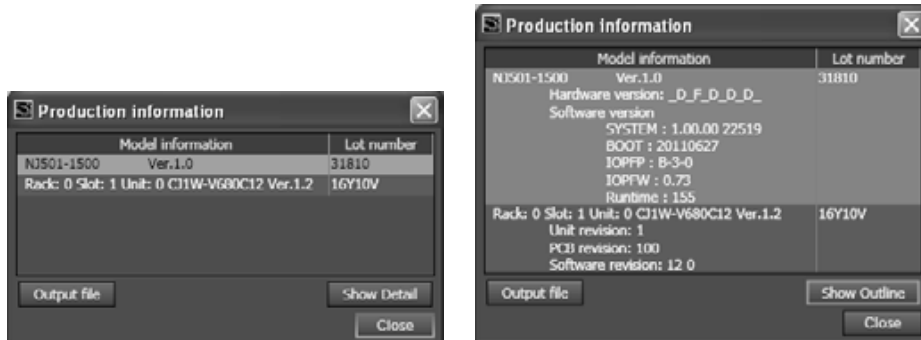
You can use the Unit Production Information on the Sysmac Studio to check the unit version of the CPU Unit, CJ-series Special I/O Units, CJ-series CPU Bus Units, and EtherCAT slaves. The unit versions of CJ-series Basic I/O Units cannot be checked from the Sysmac Studio.

● 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 for the Controller Configurations and Setup layer.

- Right-click any open space in the Unit Editor and select **Production Information**.
The Production Information Dialog Box is displayed.



Simple Display

Detailed Display

In this example, “Ver.1.0” is displayed next to the unit model.

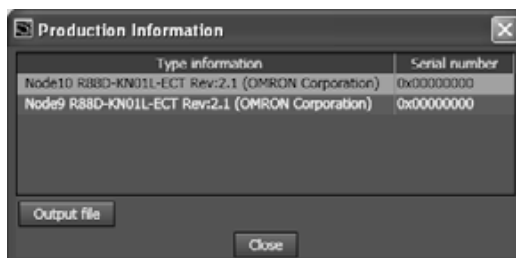
The following items are displayed.

CPU Unit	CJ-series Units
Unit model	Unit model
Unit version	Unit version
Lot number	Lot number
	Rack number, slot number, and unit number

● EtherCAT Slaves

- Double-click **EtherCAT** under **Configurations and Setup** in the Multiview Explorer. Or, right-click **EtherCAT** under **Configurations and Setup** and select **Edit** from the menu.
The EtherCAT Configuration Tab Page is displayed for the Controller Configurations and Setup layer.
- Right-click the master in the EtherCAT Configurations Editing Pane and select **Display Production Information**.

The Production Information Dialog Box is displayed.



The following items are displayed.

Node address
Type information*
Serial number

* If the model number cannot be determined (such as when there is no ESI file), the vendor ID, product code, and revision number are displayed.

Related Manuals

The following manuals are related to the NJ-series Controllers. Use these manuals for reference.

Manual name	Cat. No.	Model numbers	Application	Description
NJ-series CPU Unit Hardware User's Manual	W500	NJ501-□□□□	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 a Controller built with an NJ501 CPU Unit. <ul style="list-style-type: none"> • Features and system configuration • Introduction • Part names and functions • General specifications • Installation and wiring • Maintenance and inspection Use this manual together with the <i>NJ-series CPU Unit Software User's Manual</i> (Cat. No. W501).
NJ-series CPU Unit Software User's Manual	W501	NJ501-□□□□	Learning how to program and set up an NJ-series CPU Unit. Mainly software information is provided.	The following information is provided on a Controller built with an NJ501 CPU Unit. <ul style="list-style-type: none"> • CPU Unit operation • CPU Unit features • Initial settings • Programming based on IEC 61131-3 language specifications Use this manual together with the <i>NJ-series CPU Unit Hardware User's Manual</i> (Cat. No. W500).
Sysmac Studio Version 1 Operation Manual	W504	SYSMAC-SE2□□□	Learning about the operating procedures and functions of the Sysmac Studio.	Describes the operating procedures of the Sysmac Studio.
CJ-series User Defined CAN Units Operation Manual for NJ-series CPU Unit (This document)	W517	CJ1W-CORT21	Learning about the functions and operating procedures when the CJ-series User Defined CAN Unit is used in an NJ-series system configuration.	The functions and operating procedures when the CJ-series User Defined CAN Unit is used in an NJ-series system configuration are described.

Revision History

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

Cat. No. W517-E1-01

↑
Revision code

Revision code	Date	Revised content
01	April 2012	Original production

1

Features and System Configuration

This section provides an introductory overview of CAN bus, its functions and how to setup and configure a network. It also addresses the User Defined CAN Unit its configuration, features and specifications.

1-1	User Defined CAN Unit Features	1-2
1-2	Overview of CAN Bus	1-5
1-2-1	CAN Communication Protocol	1-6
1-2-2	Physical CAN Connection	1-10
1-2-3	Principles of Data Exchange	1-11
1-2-4	Principles of Non-Destructive Bitwise Arbitration	1-11
1-2-5	Efficiency of Bus Allocation	1-12
1-2-6	Message Frame Formats	1-14
1-2-7	Detecting and Signaling Errors	1-15
1-2-8	Data Reliability of the CAN Protocol	1-17
1-2-9	Extended Format CAN Message	1-18
1-2-10	Implementations of the CAN Protocol	1-19
1-2-11	Configuring a CAN Network	1-19
1-3	Specifications	1-20
1-3-1	User Defined CAN Unit	1-20
1-4	Basic Operating Procedures	1-22
1-4-1	Network Installation Procedure	1-22
1-4-2	User Defined CAN Unit Startup Procedure	1-23

1-1 User Defined CAN Unit Features

The User Defined CAN Unit (CJ1W-CORT21) is a CPU Bus Unit which can be installed on an NJ-series Controller. The User Defined CAN Unit will interface between the CPU of the NJ-series controller and the CAN bus. The User Defined CAN Unit is connected to the CPU of the NJ-series through the bus of the controller. The User Defined CAN Unit is connected to the CAN bus with a 5-pin open style connector.

● CPU Bus Unit

A total of up to 16 CPU Bus Units can be mounted on the CPU Rack or an Expansion Rack. The total of 16 must include all User Defined CAN Units and all other CPU Bus Units.

● Unit Control and Status

Up to 25 words of control and status information are exchanged between the User Defined CAN Unit and the CPU Unit. Control bits allow the User program to switch the CAN communication of the Unit between enabled and disabled mode. Status words provide status and diagnostics information of the Unit, the CAN network and the state of messages.

● Message Communications

The User Defined CAN Unit supports message communications exchange with the CPU. Four types of services are supported:

Configuration

- Configure specific memory locations, number of input messages and number of output messages.
- Set the identifiers and method of sending the output messages.
- Set the identifiers of the input messages
- Set the bit rate of the CAN physical layer

Transmission

- Services to send a specific output message

Error Log

- Services to read and clear the error log

Identification Services

- Services to identify the Unit and its firmware version

● Communications Configuration

Before the User Defined CAN Unit is able to operate in a CAN network, the baud rate and sample point must be configured either using the selector switches on the front of the Unit, or message communications.

● Troubleshooting Functions

The User Defined CAN Unit is provided with a variety of troubleshooting functions for prompt recovery if errors occur.

- Extensive self-diagnostic function at startup.
- Communication exchange flags indicating if message buffers are exchanged with the bus-device(s)
- Status and error flags indicating the status of the Unit and the CAN network
- Error log for recording error history data

Sending CAN Messages

Message commands are used to define the CAN messages for the User Defined CAN Unit. These commands define which messages can be sent by the Unit, the mode of sending, and the timing for when messages are sent. Each output message buffer can have one of the three modes SM1, SM2 or SM3. An output message must have the correct length, being less than or equal to 8.

Mode	Name	Description
SM1	Triggered	Sending an output message is triggered with a bit in the CPU memory. This bit is located in the Send Triggers area (see 4-2 <i>Unit Configuration and Control</i>).
SM2	On Change	Sending an output message is triggered as soon as the Unit detects that the message contents have changed or the message length has changed (an incorrect message length > 8 is not considered a change).
SM3	Cyclic	Sending an output message is triggered as soon as a specified time has elapsed.

● Triggered

In send mode SM1, a rising edge in the Send Triggers area determines that an output message will be sent. In state ST5, the Unit will evaluate the trigger. In all other states a rising edge in the Send Triggers area is ignored and messages will not be sent.

● On Change

In send mode SM2, a change in the message content determines that the output message will be sent. In state ST5, the Unit will evaluate the message content. In all other states message content changes are ignored and messages will not be sent.

● Cyclic

In send mode SM3, time determines when an output message is sent. In state ST5, the Unit checks whether the configured cycle time for the output message has elapsed. As soon as it elapses, the message is sent. In all other states the time intervals not monitored and no messages are sent.



Additional Information

It is allowed and possible to combine any of the three send modes mentioned above and all three modes can be active at the same time.

Receiving CAN Messages

The User Defined CAN Unit will process any CAN message it receives. The basic processing is to retrieve the message identifier from the message and compare it with the identifiers defined for the Unit. In total there are four different scenarios whenever a message is processed.

Condition	Unit Processing
The integrity check for the message fails. The check is done by the CAN Unit directly, or the firmware.	The User Defined CAN Unit sends an error frame on the CAN bus. The message is rejected by the CAN controller and/or the firmware, no notification is sent to the application layer.
The integrity check for the message is successful. The User Defined CAN Unit is not able to process the message due to overload conditions.	The User Defined CAN Unit sends an acknowledge on the CAN bus, ignores the message, and in the next cyclic refresh of the CPU the receive queue overflow device variable (<i>*_RcvOver</i>) is turned ON.
The integrity check for the message is successful and the User Defined CAN Unit is able to process the message, but no input message buffer is configured for the received identifier.	The User Defined CAN Unit sends an acknowledge on the CAN bus and ignores the message.
The integrity check for the message is successful. The User Defined CAN Unit is able to process the message and an input message buffer is configured for the received identifier.	The User Defined CAN Unit sends an acknowledge on the CAN bus, and processes the message with the following actions in next Unit Cyclic refresh: <ul style="list-style-type: none"> • Device variable <i>*_MsgRcv</i> is turned ON. • The message content is copied to the first input message buffer that is configured for the received identifier. • Set the receive flag of the first input message buffer that is configured for the received identifier (see <i>4-2 Unit Configuration and Control</i>).



Precautions for Correct Use

Successive messaging, sending or receiving messages with a high rate is not explicitly supported nor implemented.

1-2 Overview of CAN Bus

The Controller Area Network (CAN) is a serial communications protocol which efficiently supports distributed real-time control with a very high level of security. Its domain of application ranges from high speed networks to low cost multiplex wiring. It is especially suited for networking 'intelligent' devices as well as sensors and actuators within a system or sub-system.

In automotive electronics, engine control units, sensors, anti-skid-systems, etc. are connected using CAN with bitrates up to 1 Mbps. At the same time it is cost effective to build into vehicle body electronics, e.g. lamp clusters, electric windows etc. to replace the wiring harness otherwise required. CAN has the following properties:

- prioritization of messages
- guarantee of latency times
- configuration flexibility
- multicast reception with time synchronization
- system wide data consistency
- multi master
- error detection and signalling
- automatic retransmission of corrupted messages as soon as the bus is idle again
- distinction between temporary errors and permanent failures of nodes
- autonomous switching off of defect nodes

● Application Areas

CAN networks can be used as an embedded communication system for micro controllers as well as an open communication system for intelligent devices. The CAN serial bus system was originally developed for use in automobiles and is increasingly implemented in industrial field bus systems because the similarities are remarkable. In both cases some of the major requirements are: low cost, the ability to function in a difficult electrical environment, a high degree of real-time capability and ease of use. Some users, for example in the field of medical engineering, opted for CAN because they have to meet particularly stringent safety requirements. Similar problems are faced by manufacturers of other equipment with very high safety or reliability requirements (e.g. robots, lifts and transportation systems).

● Serial bus

CAN is a serial bus system with multi-master capabilities. All CAN nodes are able to transmit data and several CAN nodes can request the bus simultaneously. The serial bus system with real-time capabilities is the subject of the ISO 11898 international standard and covers the lowest two layers of the ISO/OSI reference model. In CAN networks there is no addressing of subscribers or stations in the conventional sense, but instead, prioritized messages are transmitted.

A transmitter sends a message to all CAN nodes (broadcasting). Each node decides on the basis of the identifier received whether it should process the message or not. The identifier also determines the priority that the message enjoys in competition for bus access. The relative simplicity of the CAN protocol means that very little cost and effort need to be expended on personal training; the CAN chip's interfaces make application programming relatively simple. Introductory courses, function libraries, starter kits, host interfaces, I/O modules and tools are available from a variety of vendors permitting low-cost implementation of CAN networks. Low-cost controller chips implementing the CAN data link layer protocol in silicon and permitting simple connection to micro controllers have been available since 1989. Today there are more than 50 CAN protocol controller chips from more than 15 manufacturers announced and available.

● License of CAN

The CAN protocol is developed by Robert Bosch GmbH and protected by patents.

1-2-1 CAN Communication Protocol

In general, the CAN communication protocol is based on the Open System Interconnection (OSI) reference model in accordance with the international standard ISO-7498 (see the following illustration).

● OSI Reference Model ISO-7498

The model defines 7 layers of communication functions, two of which (layers 1 and 2) are used in CAN. CAN uses layers 1 and 2. Layers 3 to 7 are not defined for CAN. The application layer, OSI layer 7, defines the interface functions for specific application areas. The User Defined CAN Unit has a user defined application layer. The user program configures the Unit, processes received messages, collects data to transmit and triggers the Unit to send messages. This streamlined architecture ensures fast and efficient data transmission. The application functions which are available to the user, as well as the system and device behavior of the various CAN device types are specified in the higher layer protocol.

To achieve design transparency and implementation flexibility, CAN has been subdivided into different layers according to the ISO/OSI Reference Model shown below.

Layer	Description
7	(Application Layer)
3 to 6	Not Defined
2	(Data Link Layer) Logical Link Control: LLC <ul style="list-style-type: none"> • Acceptance filtering • Overload notification • Recovery Management Medium Access Control: MAC <ul style="list-style-type: none"> • Data encapsulation and decapsulation • Frame coding, stuffing • Medium access management • Error detection • Error signalling • Acknowledgement • Serialization, deseriliazation
1	(Physical Layer) <ul style="list-style-type: none"> • Bit encoding/decoding • Bit timing • Synchronization

OSI Layer 1: Transmission Medium

ISO 11898 defines the physical layer. The CAN bus is a balanced (differential) 2-wire interface running over either a Shielded Twisted Pair (STP), Un-shielded Twisted Pair (UTP), or ribbon cable. The bit encoding used is Non Return to Zero (NRZ) encoding (with bit-stuffing) for data communication on a differential two wire bus. The use of NRZ encoding ensures compact messages with a minimum number of transitions and high resistance to external disturbance.

● Serial Bus

A number of different data rates are defined from 1Mbps to 10kbps. Cable length depends on the data rate used. The maximum line length is 5 km and the minimum is 25 meters at 1Mbps. Termination resistors are used at each end of the cable. The worst-case transmission time of an 8-byte frame with an 11-bit identifier is 134 bit times (134 microseconds at the maximum baud rate of 1Mbps).

● Transmission Speed

Transmission speeds between 10 kbps and 1000 kbps can be selected as shown in the table below. One unique transmission speed must be selected for all devices on the bus when the system is commissioned.

Baud rate (kbps)	Distance/segment (meters)
10	5000
20	2500
50	1000
125	500
250	250
500	100
800	50
1000	25

Note For bus lengths greater than 1 km, a bridge or repeater device is recommended.

● Cable Length

The maximum cable length values depend on the transmission speed and are based on a DeviceNet cable. The length can be increased by the use of repeaters. However, it is not recommended to use more than three repeaters in series in a CAN network.

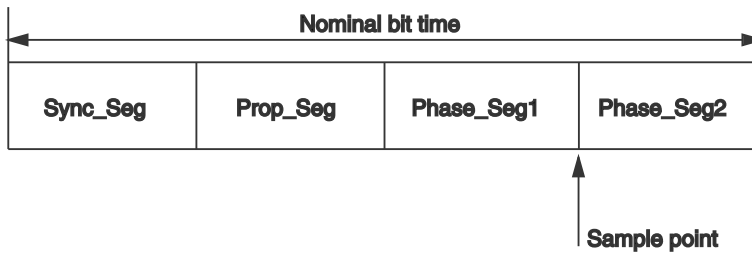
● Bit Time

The User Defined CAN Unit supports bit rates in the range of 10 kbps up to 1000 kbps. Every unit in a CAN network has its own clock generator, usually a quartz oscillator. The frequency of this oscillator determines the bit rate of the unit and therefore the reciprocal of the bit rate, also called the bit time. The timing parameter of the bit time (i.e. the sample point) can be configured individually for each CAN unit, creating a common bit rate even though the CAN units' oscillator periods (f_{osc}) may be different. The nominal bit rate is the number of bits per second transmitted in the absence of resynchronisation by an ideal transmitter. The nominal bit time is the reciprocal of the nominal bitrate:

- nominal bit time = 1 / nominal bit rate

The frequencies of these oscillators are not absolutely stable and small variations are caused by changes in temperature or voltage and by deteriorating components. As long as the variations remain inside a specific oscillator tolerance range, the CAN nodes are able to compensate for the different bit rates by resynchronizing to the bit stream. According to the CAN specification, the nominal bit time is divided into four segments. The Synchronization Segment, the Propagation Time Segment, the Phase Buffer Segment 1, and the Phase Buffer Segment 2. Each segment consists of a specific, programmable number of time quanta. The length of the time quantum (t_q), which is the basic time unit of the bit time, is defined by the CAN controller's system clock f_{sys} and the Baud Rate Prescaler (BRP): $t_q = BRP / f_{sys}$. Typical system clocks are: $f_{sys} = f_{osc}$ or $f_{sys} = f_{osc}/2$. The Synchronization Segment Sync_Seg is that part of the bit time where edges of the CAN bus level are

expected to occur; the distance between an edge that occurs outside of Sync_Seg and the Sync_Seg is called the phase error of that edge. A signal-edge is expected to lie in this segment, and is used to synchronize the bus input to the system clock. The Propagation Time Segment Prop_Seg is intended to compensate for the physical delay times within the CAN network. The Phase Buffer Segments Phase_Seg1 and Phase_Seg2 surround the Sample point.



● **Sample Point**

The sample point is the point of time at which the bus level is read and interpreted as the value of that respective bit. Its location is at the end of Phase_Seg1. The sample point is set as a percentage of the total bit time: 0% - 100%. It can be set indirectly or directly for the User Defined CAN Unit. It is set indirectly with the baud rate switches on the front of the Unit, or directly using a message command (2909).

OSI Layer 2: Datalink Layer

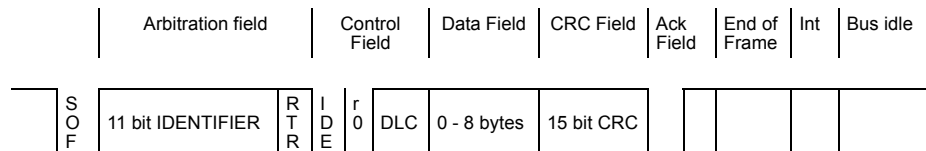
The CAN bus access protocol is implemented by OSI layer 2. This protocol also includes data security and the handling of the transmission protocols and messages. The datalink layer is layer 2 of the ISO/OSI reference model and is built with two sub-layers on top of each other: the Logical Link Control (LLC) sublayer and the Medium Access Control (MAC) sublayer.

● **MAC**

The Medium Access Control (MAC) specifies the procedures which determine when a device is permitted to transmit data. Information from transmitters to receivers are passed in data frames. The MAC sublayer represents the kernel of the CAN protocol. It presents messages received from the LLC sublayer and accepts messages to be transmitted to the LLC sublayer. The MAC sublayer is responsible for Message Framing, Arbitration, Acknowledgement, Error Detection and Signalling.

● **LLC**

The LLC sublayer is concerned with Message Filtering, Overload Notification and Recovery Management. For a detailed description of the MAC and LLC sublayer refer to Bosch CAN Specification Version 2.0.



● Data Frame

The data frame is composed of an Arbitration field, Control field, Data field, CRC field, and an ACK field. The frame begins with a 'Start of frame' [SOF], and ends with an 'End of frame' [EOF] space. The data field may be from 0 to 8 bytes. The frame check sequence is derived from a Cyclic Redundancy Code (CRC); the coefficients are generated modulo-2: $X^{15} + X^{14} + X^{10} + X^8 + X^7 + X^4 + X^3 + 1$. CAN implements five error detection mechanisms; 3 at the message level and 2 at the bit level [Also incorporates error flags].

At the message level

- Cyclic Redundancy Checks (CRC)
- Frame Checks
- Acknowledgment Error Checks

At the bit level

- Bit Monitoring
- Bit Stuffing.

● Messages

Information on the bus is sent in fixed format messages of different but limited length. Whenever the bus is free, any connected unit may start to transmit a new message. In CAN systems a CAN node does not make use of any information about the system configuration (e.g. station addresses). This has several important consequences.

System Flexibility

Nodes can be added to the CAN network without requiring any change in the software or hardware of any node and application layer.

Message Routing

The content of a message is named by an Identifier. The Identifier does not indicate the destination of the message, but describes the meaning of the data, so that all nodes in the network are able to decide by Message Filtering whether the data is to be acted upon by then or not.

Multicast

As a consequence of the concept of Message Filtering, any number of nodes can receive and simultaneously act upon the same message.

Data consistency

Within a CAN network it is guaranteed that a message is simultaneously accepted either by all nodes or by no node. Thus data consistency of a system is achieved by the concepts of multicast and by error handling.

● Message Transfer

The (application) messages transmitted and received by the User Defined CAN Unit are called Data Frames. There are two different formats for a data frame. The two formats differ in the length of the message identifier. This identifier is a field in the frame and part of the arbitration field (see the schematic picture of a data frame above). Data frames that have identifiers with a size of 11 bits are denoted Standard Frames. The other format has frames containing a 29 bit identifier. These frames are referred to as Extended Frames. The maximum length of the data transmitted in one message is 8 bytes. The length of data in a message can be 8 bytes or less. If the data length is less than 8, the frame always has 8 data bytes.

● **Identifier Length**

Standard format

The identifier's length is 11 bits and corresponds to the Base Id in the Extended format. The identifier is interpreted as an integer value in the hexadecimal range 0x0000 - 0x07FF.

Extended format

In contrast to the Standard format, the identifier in this format consists of 29 bits. The format comprises two sections; Base Id and Extended Id. The Base Id consists of 11 bits and is the identifier for the Standard format. The Extended Id consists of 18 bits and with the Base Id this is 29 bits. The identifier is interpreted as an integer value in the hexadecimal range 0x00000000 - 0x1FFFFFFF.

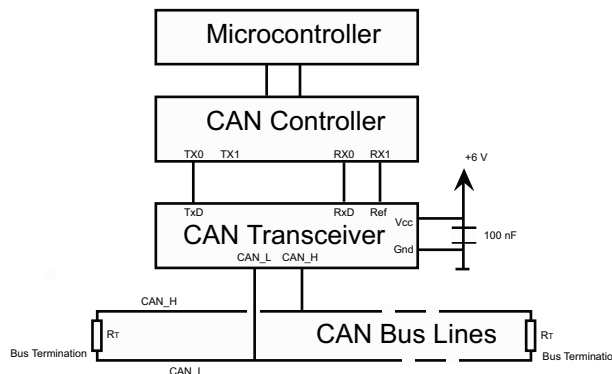
OSI Layer 7: Higher Layer Protocol

Examples of the higher layer protocol in OSI layer 7 are DeviceNet and J1939 (a set of standards concerning the design and use of devices that transmit and receive electronic signals and control information). For the User Defined CAN Unit, the higher layer protocol is implemented in the user program of the controller. Received CAN messages are stored in controller memory areas. The CAN messages to transmit are formatted in the user program and (temporarily) stored in CPU memory areas. The location and size of these memory area's are defined with message commands. Separate message commands can be used for direct transmission of CAN messages (see Section 4, *Message Communications* for details).

1-2-2 Physical CAN Connection

● **ISO 11898**

The data rates (up to 1 Mbps) necessitate a sufficiently steep pulse slope, which can be implemented only by using power elements. A number of physical connections are basically possible. However, the users and manufacturers group CAN in Automation recommends the use of driver circuits in accordance with ISO 11898. Integrated driver chips in accordance with ISO 11898 are available from several companies. The international users and manufacturers group (CiA) also specifies several mechanical connections (cable and connectors).



1-2-3 Principles of Data Exchange

● Unit Addressing

When data is transmitted by CAN, no units are addressed, but instead, the content of the message (e.g. rpm or engine temperature) is designated by an identifier that is unique throughout the network. The identifier defines not only the content but also the priority of the message. This is important for bus allocation when several units are competing for bus access.

● Sending Messages

If the CPU of a given unit wishes to send a message to one or more units, it passes the data to be transmitted and their identifiers to the assigned CAN chip. The CPU only has to initiate data exchange. The message is constructed and transmitted by the CAN chip. As soon as the CAN chip receives the bus allocation, all other units on the CAN network become receivers of this message. Each unit in the CAN network, having received the message correctly, performs an acceptance test to determine whether the data received is relevant for that unit. If the data are of significance for the unit concerned they are processed, otherwise they are ignored.

● Addressing scheme

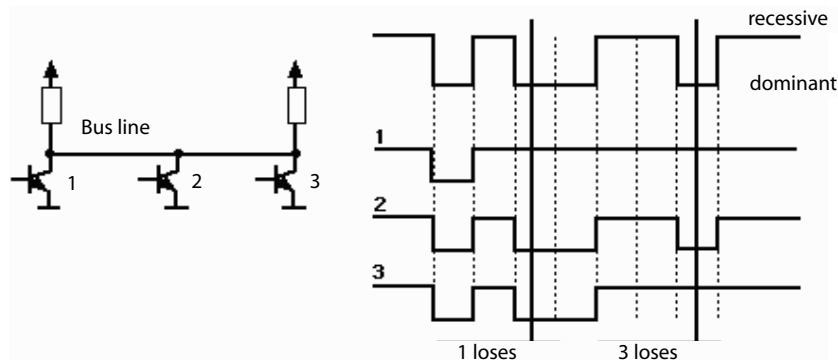
A high degree of system and configuration flexibility is achieved as a result of the content-oriented addressing scheme. It is very easy to add units to the existing CAN network without making any hardware or software modifications to the existing units, provided that the new units are purely receivers. Because the data transmission protocol does not require physical destination addresses for the individual components, it supports the concept of modular electronics and also permits multiple reception (broadcast, multicast) and the synchronization of distributed processes. Measurements needed as information by several controllers can be transmitted via the network, in such a way that it is unnecessary for each controller to have its own sensor.

1-2-4 Principles of Non-Destructive Bitwise Arbitration

For the data to be processed in real time they must be transmitted rapidly. This not only requires a physical data transfer path with up to 1 Mbps but also calls for rapid bus allocation when several units wish to send messages simultaneously.

● Priority

In real-time processing the urgency of messages to be exchanged over the network can differ greatly. A rapidly changing dimension (e.g. engine load) has to be transmitted more frequently and therefore with less delays than other dimensions (e.g. engine temperature) which change relatively slowly.



The priority at which a message is transmitted compared with another less urgent message is specified by the identifier of the message concerned. The priorities are defined during system design in the form of corresponding binary values and cannot be changed dynamically. The identifier with the lowest binary number has the highest priority.

● Bitwise Arbitration

If the bus is free, any unit may start to transmit a message. Bus access conflicts are resolved by bitwise arbitration on the identifiers involved by each unit observing the bus level bit for bit. In accordance with the "wired and" mechanism, by which the dominant state (logical 0) overwrites the recessive state (logical 1), the competition for bus allocation is lost by all those units with recessive transmission and dominant observation. During arbitration every transmitter compares the level of the bit transmitted with the level that is motored on the bus. If these levels are equal the unit may continue to send. All "losers" (levels are not equal) automatically become receivers of the message with the highest priority and do not re-attempt transmission until the bus is available again.

1-2-5 Efficiency of Bus Allocation

Bus Allocation

The efficiency of the bus allocation system is determined mainly by the possible applications for a serial bus system. In order to judge as simply as possibly which bus systems are suitable for which applications the literature includes a method of classifying bus allocation procedures. Generally we distinguish between the following classes:

- Bus allocation on a fixed time schedule.
- Bus allocation on the basis of need.

● Bus Allocation on a Fixed Time Schedule

Bus allocation done on a fixed time schedule, is made sequentially to each participant for a maximum duration regardless of whether this participant needs the bus at this moment or not. Examples of this type of bus allocation are: token slot or token passing.

● Bus Allocation on the Basis of Need

Allocation on the basis of need is defined as: the bus is allocated to one participant on the basis of outstanding transmission requests, i.e. the allocation system only considers participants wishing to transmit. Examples of this type of bus allocation are: CSMA, CSMA/CD, flying master, round robin or bitwise arbitration.

● CAN Bus Allocation

For CAN, bus allocation is negotiated purely among the messages waiting to be transmitted. This means that the procedure specified by CAN is classified as allocation on the basis of need.

Bus Access

Another means of assessing the efficiency of bus arbitration systems is the bus access method:

- Non-destructive bus access
- Destructive bus allocation

● Non-destructive Bus Access

With methods of this type the bus is allocated to one and only one unit either immediately or within a specified time following a single bus access (by one or more units). This ensures that each bus access by one or more units leads to an unambiguous bus allocation. Examples of non-destructive bus access are: token slot, token passing, round robin, bitwise arbitration.

● Destructive Bus Allocation

Simultaneous bus access by more than one unit causes all transmission attempts to be aborted and therefore there is no successful bus allocation. More than one bus access may be necessary in order to allocate the bus at all, the number of attempts before bus allocation is successful being a purely statistical quantity. Examples of destructive bus access are: CSMA/CD, Ethernet.

● Unambiguous Bus Allocation

In order to process all transmission requests of a CAN network while complying with latency constraints at as low a data transfer rate as possible, the CAN protocol must implement a bus allocation method that guarantees that there is always unambiguous bus allocation even when there are simultaneous bus accesses from different units. The method of bitwise arbitration using the identifier of the messages to be transmitted uniquely resolves any collision between a number of units wanting to transmit, and it does this at the latest within 13 (standard format) or 33 (extended format) bit periods for any bus access period. Unlike the message-wise arbitration employed by the CSMA/CD method this non-destructive method of conflict resolution ensures that no bus capacity is used without transmitting useful information.

Even in situations where the bus is overloaded the linkage of the bus access priority to the content of the message proves to be a beneficial system attribute compared with existing CSMA/CD or token protocols: In spite of the insufficient bus transport capacity, all outstanding transmission requests are processed in order of their importance to the overall system (as determined by the message priority). The available transmission capacity is utilized efficiently for the transmission of useful data since "gaps" in bus allocation are kept very small. The collapse of the whole transmission system due to overload, as can occur with the CSMA/CD protocol, is not possible with CAN. Thus, CAN permits implementation of fast, traffic-dependent bus access which is non-destructive because of bitwise arbitration based on the message priority employed.

● Bus Access Control

Non-destructive bus access can be further classified into two types depending on whether the control mechanisms are present in the system only once (centralized) or more than once (decentralized).

- centralized bus access control
- decentralized bus access control

A communication system with a designated unit must provide a strategy to take effect in the event of a failure of the master unit. This concept has the disadvantage that the strategy for failure management is difficult and costly to implement and also that the takeover of the central unit by a redundant unit can be very time-consuming. For these reasons and to circumvent the problem of the reliability of the master unit (and thus of the whole communication system), the CAN protocol implements decentralized bus control. All major communication mechanisms, including bus access control, are implemented several times in the system, because this is the only way to fulfill the high requirements for the availability of the communication system.

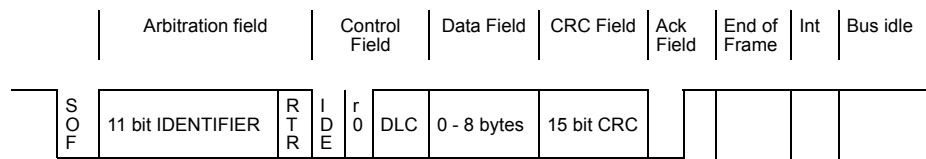
In summary it can be said that CAN implements a traffic-dependent bus allocation system that permits, by means of a non-destructive bus access with decentralized bus access control, a high useful data rate at the lowest possible bus data rate in terms of the bus busy rate for all units. The efficiency of the bus arbitration procedure is increased by the fact that the bus is utilized only by those units with pending transmission requests.

These requests are handled in the order of the importance of the messages for the system as a whole. This proves especially advantageous in overload situations. Since bus access is prioritized on the basis of the messages, it is possible to guarantee low individual latency times in real-time systems.

1-2-6 Message Frame Formats

● Identifier

The CAN protocol supports two message frame formats, the only essential difference being in the length of the identifier (ID). In the standard format the length of the ID is 11 bits and in the extended format the length is 29 bits. The message frame for transmitting messages on the bus comprises seven main fields.



● Arbitration Field

A message in the standard format begins with the start bit (Start Of Frame). This is followed by the arbitration field, which contains the identifier and the RTR (Remote Transmission Request) bit, which indicates whether it is a data frame or a request frame without any data bytes (remote frame).

● Control Field

The control field contains the IDE (Identifier Extension) bit, which indicates either standard format or extended format, a bit reserved for future extensions and, in the last 4 bits, a count of the data bytes in the data field. The number of bytes in the data field is indicated by the Data Length Code (DLC), the data length code is 4 bits wide.

Number of Data Bytes	Data Length Code			
	DLC3	DLC2	DLC1	DLC0
0	Dominant	Dominant	Dominant	Dominant
1	Dominant	Dominant	Dominant	Recessive
2	Dominant	Dominant	Dominant	Recessive
3	Dominant	Dominant	Recessive	Dominant
4	Dominant	Recessive	Recessive	Recessive
5	Dominant	Recessive	Dominant	Dominant
6	Dominant	Recessive	Recessive	Dominant
7	Dominant	Recessive	Recessive	Recessive
8	Recessive	Dominant	Dominant	Dominant

● Data Field

The data field ranges from 0 to 8 bytes in length and is followed by the CRC field, which is used as a frame security check for detecting bit errors.

● CRC Field

The CRC field contains the CRC sequence, followed by a delimiter. this CTC-delimiter is a single recessive bit. The frame check sequence is derived from a cyclic redundancy code best suited for frames with bit counts less than 127 bits (BCH code). In order to carry out the CRC calculation the polynomial to be divided is defined as the polynomial, the coefficients of which are given by the destuffed bit stream consisting of Start Of Frame (SOF), arbitration field, control field, data field and for the 15 lowest coefficients, by 0. This polynomial is divided (the coefficients are calculated modulo-2) by the generator-polynomial:

- $X^{15} + X^{14} + X^{10} + X^8 + X^7 + X^4 + X^3 + 1$

The remainder of this polynomial division is the CRC sequence transmitted over the bus.

● Ack Field

The ACK field comprises the ACK slot (1 bit) and the ACK delimiter (1 recessive bit). The bit in the ACK slot is sent as a recessive bit and is overwritten as a dominant bit by those receivers which have at this time received the data correctly (positive acknowledgement). Correct messages are acknowledged by the receivers regardless of the result of the acceptance test.

● Intermission

The end of the message is indicated by End Of Frame. Intermission is the minimum number of bit periods separating consecutive messages. If there is no following bus access by any unit, the bus remains idle (bus idle).

1-2-7 Detecting and Signaling Errors

● Error Detection

Unlike other bus systems, the CAN protocol does not use acknowledgement messages but instead signals any errors that occur. For error detection the CAN protocol implements three mechanisms at the message level:

Cyclic Redundancy Check (CRC)

The CRC safeguards the information in the frame by adding redundant check bits at the transmission end. At the receiver end these bits are re-computed and tested against the received bits. If they do not agree there has been a CRC error.

Frame Check

This mechanism verifies the structure of the transmitted frame by checking the bit fields against the fixed format and the frame size. Errors detected by frame checks are designated "format errors".

ACK Errors

As mentioned above, frames received are acknowledged by all recipients through positive acknowledgement. If no acknowledgement is received by the transmitter of the message (ACK error) this may mean that there is a transmission error which has been detected only by the recipients, that the ACK field has been corrupted or that there are no receivers.

● Bit Level Errors

The CAN protocol also implements two mechanisms for error detection at the bit level:

Monitoring

The ability of the transmitter to detect errors is based on the monitoring of bus signals: each node which transmits also observes the bus level and thus detects differences between the bit sent and the bit received. This permits reliable detection of all global errors and errors local to the transmitter.

Bit Stuffing

The coding of the individual bits is tested at bit level. The bit representation used by CAN is NRZ (non-return-to-zero) coding, which guarantees maximum efficiency in bit coding. The synchronization edges are generated by means of bit stuffing, i.e. after five consecutive equal bits the sender inserts into the bit stream a stuff bit with the complementary value, which is removed by the receivers. The code check is limited to checking adherence to the stuffing rule.

● Error Flag

If one or more errors are discovered by at least one unit (any unit) using the above mechanisms, the current transmission is aborted by sending an Error Flag. This prevents other units accepting the message and thus ensures the consistency of data throughout the network.

● Re-transmission

After transmission of an erroneous message has been aborted, the sender automatically re-attempts transmission (automatic repeat request). There may again be competition for bus allocation. As a rule, retransmission will begin within 23 bit periods after error detection; in special cases the system recovery time is 31 bit periods.

However effective and efficient the method described may be, in the event of a defective unit it might lead to all messages (including correct ones) being aborted, thus blocking the bus system if no measures for self-monitoring were taken. The CAN protocol therefore provides a mechanism for distinguishing sporadic errors from permanent errors and localizing unit failures (fault confinement).

This is done by statistical assessment of unit error situations with the aim of recognizing a unit's own defects and possibly entering an operating mode where the rest of the CAN network is not negatively affected. This may go as far as the unit switching itself off to prevent messages erroneously recognized as incorrect from being aborted.

● Bus Off

With respect to fault confinement a CAN unit is in one of three states, the number of the state increases with the severity of the error:

- error active
- error passive
- bus off

An *error active* unit can normally take part in bus communication and sends an Active Error Flag when an error has been detected.

An *error passive* unit must not send an Active Error Flag. It takes part in bus communication but when an error has been detected only a Passive Error Flag is sent. Also after a transmission, an *error passive* unit will wait before initiating a further transmission.

A *bus off* unit is not allowed to have any influence on the bus. (E.g. output drivers switched off.)

● Error Counters

For fault confinement two counters are implemented in every CAN unit:

- Transmit error counter
- Receive error counter

For respectively a transmit error and a receive error the corresponding counter increases, normally the receive counter is increased by one for every error, the transmit error counter is normally increased by eight. See the CAN bus specification for a detailed description of increasing the error counters, and the exceptions in increasing the counters.

A CAN unit is *error passive* when the transmit error counter equals or exceeds 128, or when the receive error counter equals or exceeds 128. A CAN unit is *bus off* when the transmit error counter is greater than or equal to 256.

● Error Reset

In *bus off* mode, the CAN unit has its communication disabled. The user program in the CPU can enable the communication using a message command. Enabling the communication will reset both the receive and transmit counter.

1-2-8 Data Reliability of the CAN Protocol

● Reliability

The introduction of safety-related systems in automobiles brought with it high requirements for the reliability of data transmission. The objective is frequently formulated as not permitting any dangerous situations for the driver to occur as a result of data exchange throughout the whole life of a vehicle.

This goal is achieved if the reliability of the data is sufficiently high or the residual error probability is sufficiently low. In the context of bus systems data, reliability is understood as the capability to identify data corrupted by transmission faults. The residual error probability is a statistical measure of the impairment of data reliability: It specifies the probability that data will be corrupted and that this corruption will remain undetected. The residual error probability should be so small that on average no corrupted data will go undetected throughout the whole life of a system.

● Residual Error

Calculation of the residual error probability requires that:

- the errors which occur are classified, and
- that the whole transmission path is described by a model.

This calculation of the residual error probability of CAN results in a maximum bit error probability, which is approximately 0.02 - in the order of 10^{-13} ($0.02 * 10^{-13}$). The residual error probability of CAN is determined as a function of:

- the bit error probability for message lengths of 80 to 90 bits,
- for system configurations of, for instance, five or ten nodes
- and with an error rate of 1/1000 (an error in one message in every thousand).

Based on the residual error probability it is possible to calculate the maximum number of undetectable errors for a given CAN network.

● Undetectable Errors

For example, if a CAN network operates at a data rate of 1 Mbps/s, at an average bus capacity utilization of 50 percent, for a total operating life of 4000 hours and with an average message length of 80 bits, then the total number of messages transmitted is: 9×10^{10} .

The statistical number of undetected transmission errors during the operating life is thus in the order of less than 10^{-2} . Or, with an operating time of eight hours per day on 365 days per year and an error rate of 0.7 s, one undetected error occurs every thousand years (statistical average).

1-2-9 Extended Format CAN Message

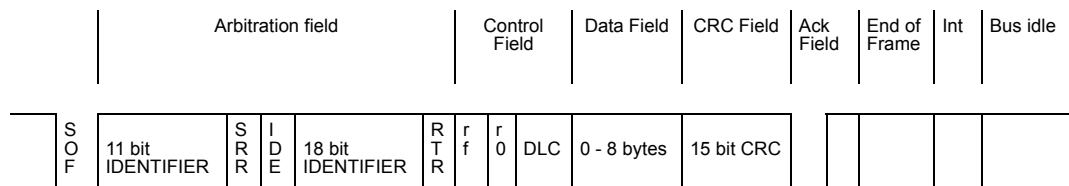
● 29-bit Identifier

The Society of Automotive Engineers (SAE) "Truck and Bus" sub-committee standardized signals and messages as well as data transmission protocols for various data rates. It became apparent that standardization of this kind is easier to implement when a longer identification field is available.

To support these efforts, the CAN protocol was extended by the introduction of a 29-bit identifier. This identifier is made up of the existing 11-bit identifier (base ID) and an 18-bit extension (ID extension). Thus the CAN protocol allows the use of two message formats: StandardCAN (Version 2.0A) and ExtendedCAN (Version 2.0B). As the two formats must coexist on one bus, it is determined which message has higher priority on the bus in the case of bus access collisions with differing formats and the same base identifier. The message in standard format always has priority over the message in extended format.

● Extended Format

CAN controllers which support the messages in extended format can also send and receive messages in standard format. When CAN controllers which only cover the standard format (Version 2.0A) are used on one network, then only messages in standard format can be transmitted on the entire network (messages in extended format would be misunderstood). However there are CAN controllers which only support standard format but recognize messages in extended format and ignore them (Version 2.0B passive).



The distinction between standard format and extended format is made using the IDE bit (Identifier Extension Bit) which is transmitted as dominant in the case of a frame in standard format. For frames in extended format it is recessive.

● RTR Bit

The RTR bit is transmitted dominant or recessive depending on whether data are being transmitted or whether a specific message is being requested from a unit. In place of the RTR bit in standard format the SRR (substitute remote request) bit is transmitted for frames with extended ID. The SRR bit is always transmitted as recessive, to ensure that in the case of arbitration the standard frame always has priority bus allocation over an extended frame when both messages have the same base identifier.

● IDE Bit

Unlike the standard format, in the extended format the IDE bit is followed by the 18-bit ID extension, the RTR bit and a reserved bit (r1).

All the following fields are identical with standard format. Conformity between the two formats is ensured by the fact that the CAN controllers which support the extended format can also communicate in standard format.

1-2-10 Implementations of the CAN Protocol

Overview

Communication is identical for all implementations of the CAN protocol. There are differences, however, with regard to the extent to which the implementation takes over message transmission from the micro controllers which follow it in the circuit.

● CAN Controller with Intermediate Buffer

CAN controllers with intermediate buffer, formerly called basicCAN chips, have implemented as hardware the logic necessary to create and verify the bitstream according to the protocol. However, the administration of messages to be sent and received, acceptance filtering in particular, is carried out only to a limited extent by the CAN controller.

● CAN Controller with Object Storage

CAN objects consist mainly of three components: identifier, data length code and the useful data. CAN controllers with object storage, formerly called FullCAN, function like CAN controllers with intermediate buffers. Additionally they administer a certain number of objects. Where there are several simultaneous requests to transmit objects over the bus, the CAN controllers determine, for example, which object is to be transmitted first. They also carry out acceptance filtering for incoming objects. The micro controller following the CAN controller has to administer only a few bits (e.g. transmission request).

● CAN Slave Controllers for I/O Functions

As well as CAN controllers which support all functions of the CAN protocol there are also CAN implementations possible which do not require a following micro controller. These CAN implementations are called SLIO (Serial Link I/O) acting as CAN slave units and having to be administered by a CAN master unit.

1-2-11 Configuring a CAN Network

In order to operate a CAN network, each unit in the network needs to be configured. This process of network and unit configuration involves the following procedures.

- Installing the physical network topology, i.e. installing User Defined CAN Units in the Controller system, installing any other CAN unit in the network, and wiring the network.
- Configuring the bus parameters which define the baud rate and the bus timing parameter sample point. Configuration for the User Defined CAN Unit is done either with the baud rate switches in the front panel, or with message commands.
- Defining the configuration data, i.e. defining the process data, which will be exchanged between the User Defined CAN Unit and other nodes on the CAN network.
- Defining the parameterization data for the User Defined CAN Unit which establishes the filtering of message identifiers and the configuration of message buffers in the Controller.
- Parameterization of the User Defined CAN Unit with message commands issued from the User program in the Controller.

1-3 Specifications

1-3-1 User Defined CAN Unit

● General Specifications

General specifications of the CJ-series User Defined CAN Unit conform to those of the NJ-series CPU Units.

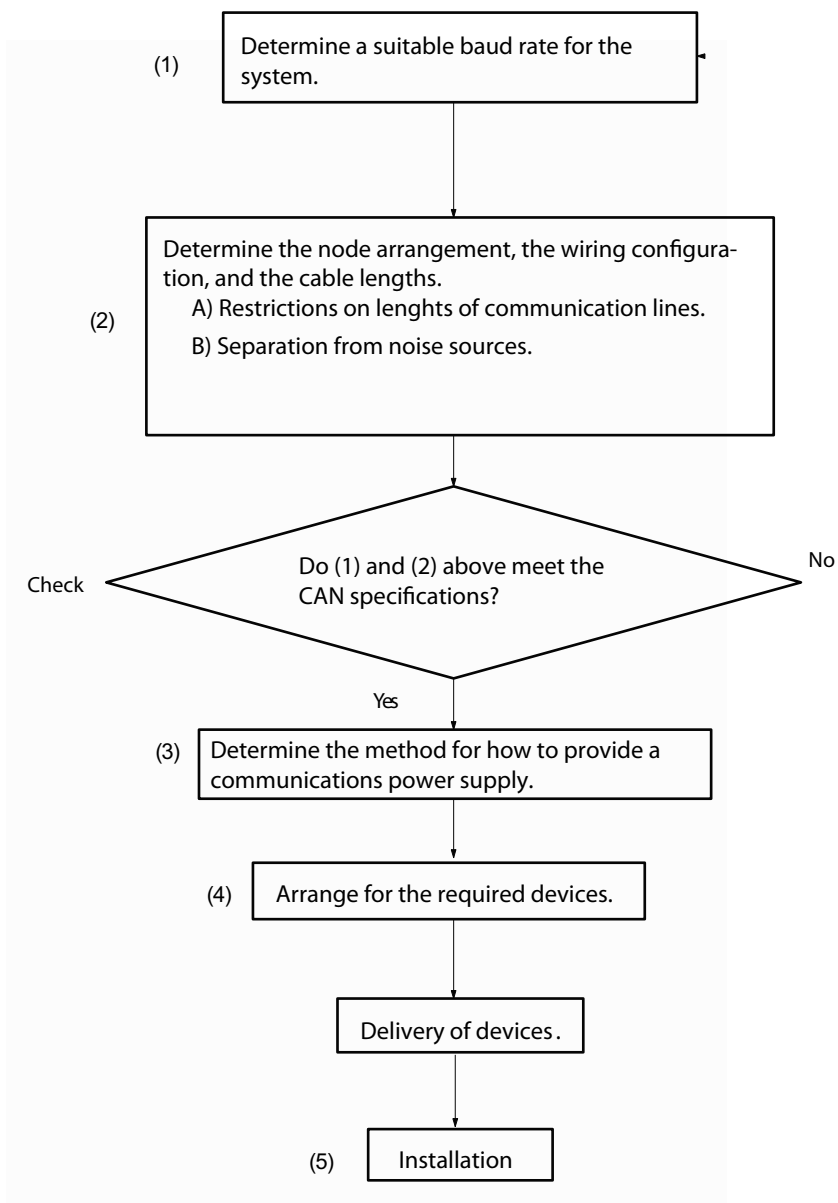
● Functional and Performance Specifications

Item		Specification
User Defined CAN Unit model		CJ1W-CORT21
Applicable Controller		NJ Series
Unit classification		CPU Bus Unit
Applicable unit numbers		0 to F
Mounting position		CPU Rack or Expansion Rack
No. of Units that can be mounted		16 Units max.
Words allocated in the memory used for CJ-series Unit	I/O port (without power OFF retention) (Access via the device variables for CJ-series Unit)	25 words/Unit CPU Unit to User Defined CAN Unit: 10 words for the software switches and status words. The remaining 15 words are reserved for future use.
	I/O port (with power OFF retention) (Access via the device variables for CJ-series Unit)	100 words/Unit Reserved for future use
Supported connections (communications)		Message buffers and message communications (FINS*1). All conform to CAN communications standards.
Message communications*1	Message communications for configuration	<ul style="list-style-type: none"> • Configuration of Input and Output buffers with a maximum of 640 for each type • Configuration of send and receive flags area (limit of one of each type) • Configuration of baud rate and sample point
	Message communications	Message communications to send/receive a CAN message directly
	Other functions	Error history can be accessed using a message command.
Setting section		Rotary switches for Unit No. (hexadecimal x 1) Front panel DIP switch for Baud rate
Display section		Two indicators(2 colors): One green to indicate the Unit is powered. One red to indicate and error status. Two 7-segment displays: Displays CAN communication status and error code. Two dot indicators within the 7-segment display: Left dot indicates communication status configuration set from message communications. Right dot indicates communication is configured and enabled.
Front connector		One communications connector (communications data: CAN H and CAN L, network power supply: CAN_V+, ground CAN_GND and shielded) Use the XW4B-05C1-H1-D connector provided to connect the communications cable. <ul style="list-style-type: none"> • Use the XW4B-05C4-T-D connector sold separately for multi-drop connections.
Communications power supply voltage		11 to 25 VDC (supplied from the communications connector)
Current consumption		Communications power supply: 18 mA at 24 VDC, (supplied from the communications connector) Internal circuit power supply: 290 mA max. at 5 VDC (supplied from the Power Supply Unit)
External dimensions		31 x 90 x 65 mm (W x H x D)
Weight		118 g (including the connector provided)
Standard accessories		One XW4B-05C1-H1-D connector to connect to a node from a T-branch Tap.

- *1 FINS message communications are available with the NJ Series. However, with these functions, not all areas of the NJ-series CPU Unit are accessible. If these functions need to be used, such as to connect to existing equipment, please consult with your OMRON representative.

1-4 Basic Operating Procedures

1-4-1 Network Installation Procedure



1-4-2 User Defined CAN Unit Startup Procedure

The basic operating procedures for the User Defined CAN Unit are described here.

Use Sysmac Studio to create programs and to configure the Unit.

For details on operations of Sysmac Studio, refer to the *Sysmac Studio Version 1 Operation Manual* (Cat. No. W504).

Programming and software settings	Step 1: Create POU and global variables Create unit configuration	<ul style="list-style-type: none"> · Create and register global variables and each POU. Register variables including variables for accessing the slave area and for message communications data. · Create algorithm of each POU and register local variables of each POU. · Register the Unit in the Unit Configuration on the CPU/Expansion Racks screen in Configuration and Setup. At this time, set both the device name and the unit number of the unit to be registered. · Allocate device variables for CJ-series Unit to I/O port. Allocate on the I/O Map View window. You can use one of the following three methods to allocate. <ol style="list-style-type: none"> 1. Select and allocate existing variables. 2. Input a new variable name. 3. Automatically create with "Device variable creation".
	Step 2: Create Initialization Data of the Unit	<ul style="list-style-type: none"> · Create the Unit settings by using the [Edit Special Unit Settings]. Create the User program(s) to send message communications to configure the Unit (see section 4). (When the program is transferred, items set here will be reflected in the device variable for CJ-series Unit.)
Hardware setting and Rack assembly	Step 3: Set up the User Defined CAN Unit's Unit number [UNIT No.] Operation settings for baud rate/communication errors [DIP switch]	<ul style="list-style-type: none"> · Set the hardware switch on the front panel. Set the same unit number as in the Unit Configuration settings.
	Step 4: Set up Remote Devices. Baud rate, etc.	<ul style="list-style-type: none"> · Initial settings for Remote Devices Set by referring to the Manual for Devices.
	Step 5: Mount and wire to the Rack Wire the remote devices	
	Step 6: Turn ON the power to the Controller.	<ul style="list-style-type: none"> · Reflect the settings for the switches on the front panel of the Unit. Switch ON the power in the sequence: Communications power, slave power, controller power, or switch all three ON at the same time.
Program transfer and operation	Step 7: Transfer the user programs	<ul style="list-style-type: none"> · Transfer user programs, Unit Configuration and Setup, and variable information.
	Step 8: Main operation	

Allocation of User-defined Variables to the Memory Used for CJ-series Unit

Bits and words are allocated to the memory used for CJ-series when you configure the User Defined CAN unit with message commands.

- send buffers specification
- send triggers specification
- receive buffers specification
- receive triggers specification

To use this area from the user program, you need to create a user-defined variable of AT specification. User-defined variables are created using Sysmac Studio.

Have the necessary user-defined variables created before creating a program.

For details on operations, refer to the *Sysmac Studio Version 1 Operation Manual* (Cat. No. W504).



Precautions for Correct Use

The user is responsible for management of the memory used for NJ-series Unit. Please take great care to avoid overlapping of reference areas between user-defined variables.

2

Nomenclature and Installation

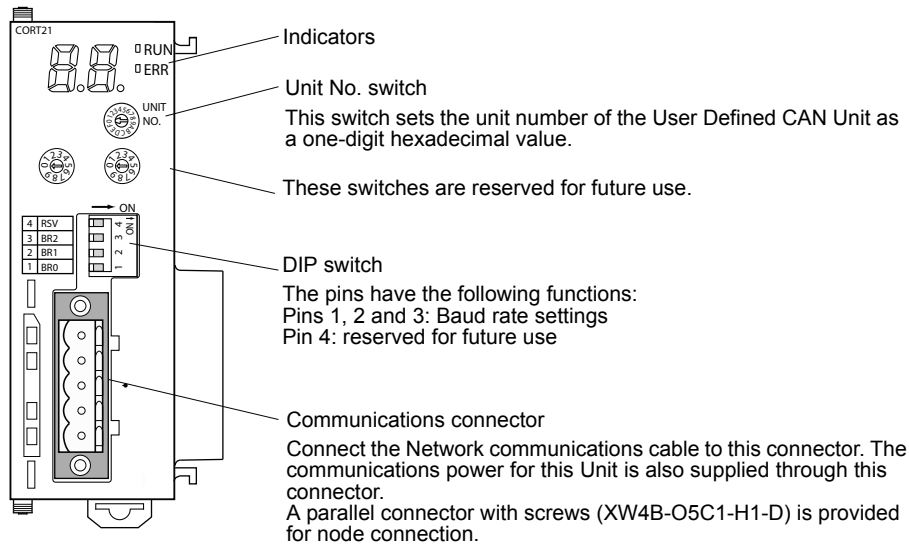
This section describes the nomenclature and installation of the User Defined CAN Unit.

2-1	Nomenclature and Installation	2-2
2-1-1	Nomenclature and Functions	2-2
2-1-2	Switch Settings	2-5
2-2	Installing the User Defined CAN Unit	2-8
2-2-1	System Configuration Precautions	2-8
2-2-2	Mounting	2-8
2-2-3	Handling Precautions	2-9
2-2-4	External Dimensions	2-9
2-2-5	Unit States	2-10

2-1 Nomenclature and Installation

2-1-1 Nomenclature and Functions

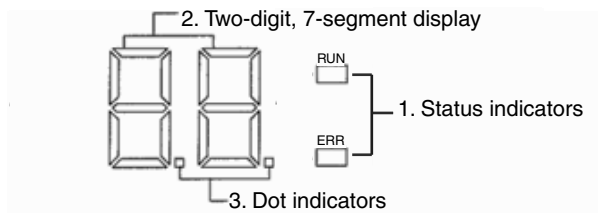
CJ1W-CORT21



Indicators

The User Defined CAN Units are equipped with the following indicators that indicate the operating status of the node itself and the overall network.

1. Two status indicators (two-color: green or red indicators)
2. A two-digit, 7-segment display
3. Two dot indicators



● Status Indicators: Run and ERR

The status indicators on the front panel of the User Defined CAN Unit are labelled RUN and ERR. The RUN and ERR LED indicates the status of the node itself and/or the status of the network.

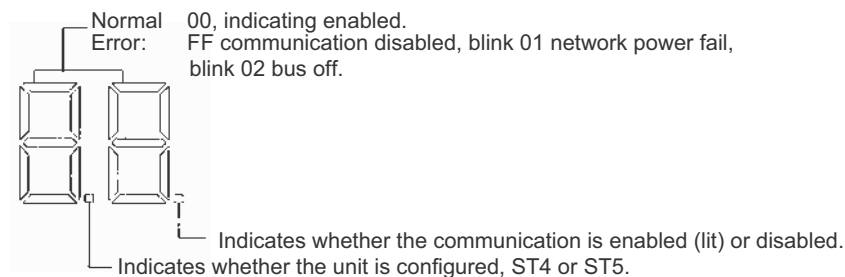
The RUN and ERR indicators can be green or red and they can be turned ON or OFF. The following table lists the different states and colors of the indicators with their meaning. See 2-2-5 *Unit States* for more information on Unit States.

Indicator	Color	Status	Condition
RUN	Green	ON	Normal operating status. The Unit state is ST3, ST4 or ST5.
	Red	ON	<ul style="list-style-type: none"> A non-recoverable fatal error has occurred Unit state is ST2 and a start-up error has occurred.
	---	OFF	Power is not supplied or the Unit is in state ST1 or ST2.
ERR	Red	ON	<p>A fatal communication error has occurred. Network communications are not possible. One or more of the following errors active:</p> <ul style="list-style-type: none"> Hardware error at startup (Unit in ST2). No CAN configuration. CAN network power fail (<i>*_NetPwrErr</i>) Unit is Bus off (<i>*_BusoffErr</i>) Fatal error in Unit CPU Watchdog Time-out CPU communication bus error Cyclic refresh time-out. I/O table error
	---	OFF	The Unit has no active error, diagnostic functions do not detect any error condition.

● Seven-Segment Indicator

In addition to the RUN and ERR indicators, User Defined CAN Units have a 2-digit, 7-segment display that normally indicates the enabled or disabled status of the communication. When an error occurs, the seven-segment indicators will display a (flashing) error code.

There are dot indicators in the lower-right corner of each digit. The left dot of these two indicators shows whether or not the Unit is configured with message commands (ST4 or ST5). The right dot indicator shows whether the communication is enabled or disabled. the dot is lit when the communication is enabled (ST5).



Seven Segment Digits

Status	Display
Unit not mounted in CPU Rack	Not Lit
Initializing (State ST1*)	
Start-up error (State ST2*)	
Unit not configured (State ST3*)	Lit: FF
CAN communications are disabled (State ST4*)	
CAN communications are enabled (State ST5*)	Lit: 00
CAN network power failure	Blink: 01
CAN bus off	Blink: 02

*See section 2-2-5 *Unit States* for more information.

Dot Indicator

The following table shows the functions of the dot indicators.

Indicator	Function	Status
Left dot	Unit operational	ON: The Unit is configured and communication is enabled or disabled and in state ST4* or ST5*. OFF: The Unit has a start-up error or is not yet configured and Unit state is ST1*, ST2* or ST3*. BLINK: The Unit state is ST2*.
Right dot	Communication enabled/disabled	ON: Communication is enabled and Unit state is ST5* (left dot is lit). OFF: Communication is disabled if left dot is lit, otherwise the Unit is in state ST1*, ST2* or ST3*.

*See section 2-2-5 *Unit States* for more information.

● Normal Start-up

During a normal start-up the indicators will show the following information:

Action	Indicator	State
Power OFF State is ST0*	RUN	-
	ERR	-
	7-Segment	--
	Left dot	-
	Right dot	-
Power ON State = ST3*	RUN	Green
	ERR	-
	7-Segment	FF
	Left dot	-
	Right dot	-

Action	Indicator	State
Areas configured, message command 2902 State = ST4*	RUN	Green
	ERR	-
	7-Segment	FF
	Left dot	Red
	Right dot	-
Buffer(s) configured, message command 2903 - 2906 State = ST4*	RUN	Green
	ERR	-
	7-Segment	FF
	Left dot	Red
	Right dot	-

*See section 2-2-5 *Unit States* for more information.

2-1-2 Switch Settings

Unit No. Switch



Use the Unit No. Switch to set the unit number for units as a CPU Special Unit. Set a unique unit number for each CPU set Special Unit installed in the CPU Rack and Expansion Rack with the unit numbers in the unit configuration. Turn OFF the Controller before changing the unit number setting.

Setting method: One-digit hexadecimal

Setting range: 0 to F

Note The unit number is set to 0 at the factory.

You can set any unit number from 0 to F as long as it has not been set on another CPU Bus Unit connected to the same CPU Unit.



Precautions for Safe Use

- Use a small flat-blade screwdriver to turn the rotary switches; be careful not to damage the switch.
- Always turn OFF the Controller before changing the unit number setting.



Additional Information

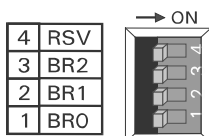
- If the unit number is the same as one set on another CPU Bus Unit connected to the same CPU Unit, a major fault level controller error "Duplicate Unit Number" will occur and it will not be possible to start up the CAN network. After correcting the unit number setting, cycle the power to the Controller.

Rotary Switches

The two 10-position rotary switches below the Unit No. are reserved for future use.

DIP Switch

The DIP switches on the front of the User Defined CAN Unit are used to set the baud rate and implicitly, the sample point. The baud rate is set with switches 1, 2 and 3. Switch number 4 is reserved for future use.



The settings of the DIP switch pins for the baud rate are shown in the following table. All pins are set to OFF at the factory.

Pin 1	Pin 2	Pin 3	Baud rate	Sample point (%)
OFF	OFF	OFF	10	80
ON	OFF	OFF	20	80
OFF	ON	OFF	50	80
ON	ON	OFF	125	80
OFF	OFF	ON	250	80
ON	OFF	ON	500	80
OFF	ON	ON	reserved	reserved
ON	ON	ON	1000	70*

Note *Sample point is reduced due to the clock frequency.



Precautions for Safe Use

Always turn OFF the Controller before changing the DIP switch settings.

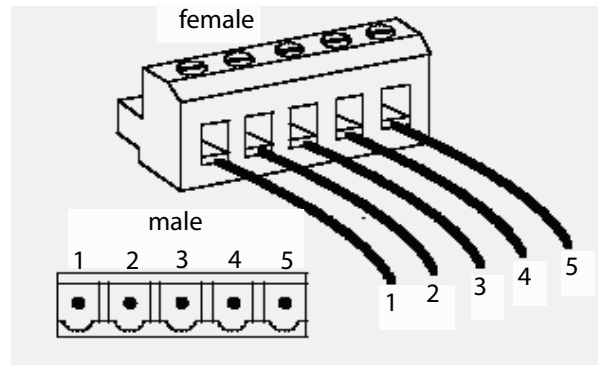


Precautions for Correct Use

Set the same baud rate on all of the nodes in the network. Any unit with a baud rate different from the other units' baud rate won't be able to participate in communications and may cause a communications error between units that have been set properly

CAN Bus Connector

The CAN bus connector is a 5-pin open style connector. This CAN connector is located on the front side of the Unit, and is a male connector with five pins.



Color stickers that match communication cable colors are attached to the communications connectors. Match the colors when connecting communication cables to the connectors, the colors can be found in the next table.

Pin 1	Signal	Color	Description
1	CAN_GND	Black	Power line, negative voltage
2	CAN_L	Blue	Communications line, low
3	CAN_SHLD	----	Shield
4	CAN_H	White	Communications line, high
5	CAN_V+	Red	Power line, positive voltage



Precautions for Safe Use

Before connecting communications cables, turn OFF the Controller power supply, all slave power supplies, and the communications power supply.



Precautions for Correct Use

Required for normal operation of CAN network is an external 24V power supply, connected to the Unit using the front connector.

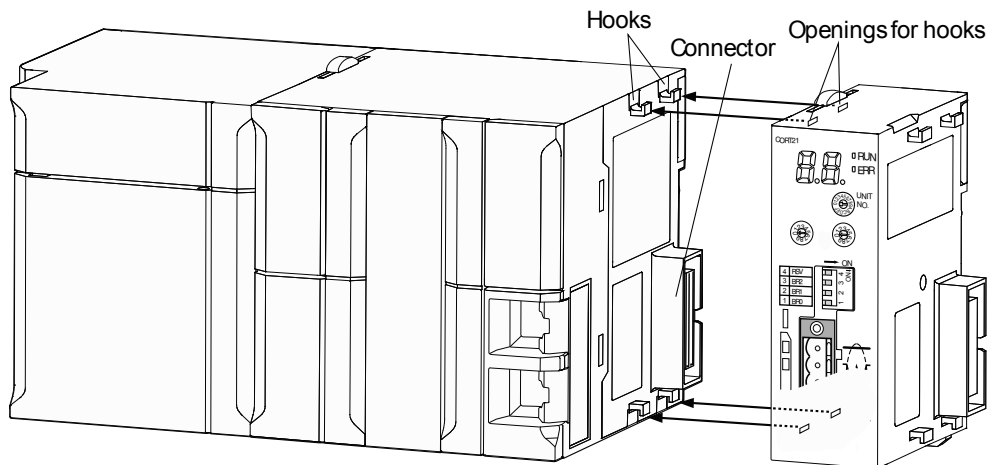
2-2 Installing the User Defined CAN Unit

2-2-1 System Configuration Precautions

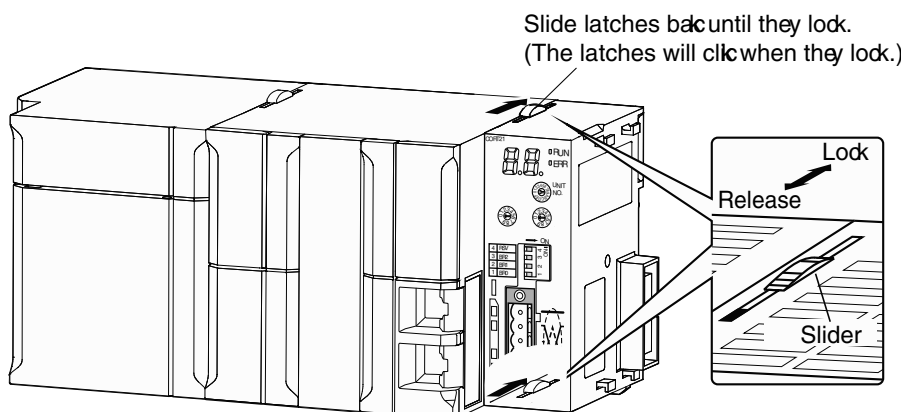
You can mount up to 16 Units on the CPU Rack or an Expansion Rack per CPU (but no more than 10 Units on one Rack).

2-2-2 Mounting

- 1 Carefully align the connectors to mount the User Defined CAN Unit.



- 2 Move the yellow sliders on the top and bottom of the Unit until they click into position, to lock.



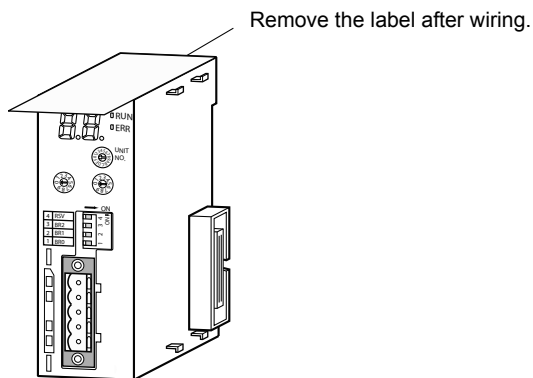
Precautions for Safe Use

If the sliders are not securely locked, the User Defined CAN Unit functions may not operate sufficiently.

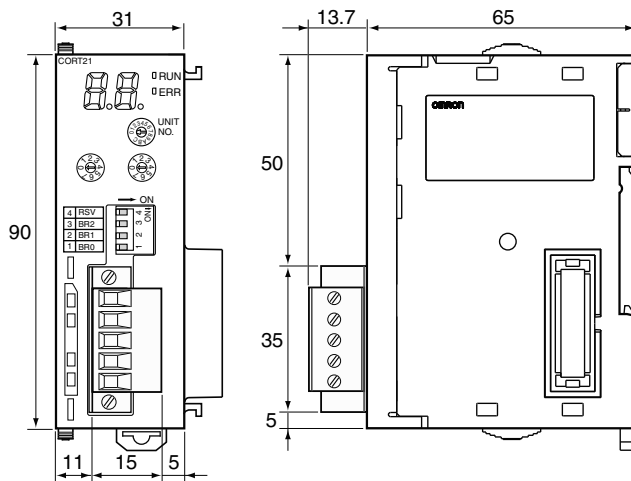
To dismount the Unit, move the sliders to the "Release" direction.

2-2-3 Handling Precautions

- Always turn OFF the Controller before you mount or dismount a Unit or connect or disconnect cables.
- Provide separate conduits or ducts for the I/O lines to prevent noise from high-tension lines or power lines.
- Prevent wire clippings, cutting chips or other materials from getting inside the Unit. They could cause scorching, failure, and malfunction. Pay particular attention to this during installation and take measures such as covering with a cover.
- If the Unit was shipped from the factory with the dust protection label on top of the unit, be sure to remove that label before switching ON the power. The label prevents heat dissipation and could cause a malfunction.



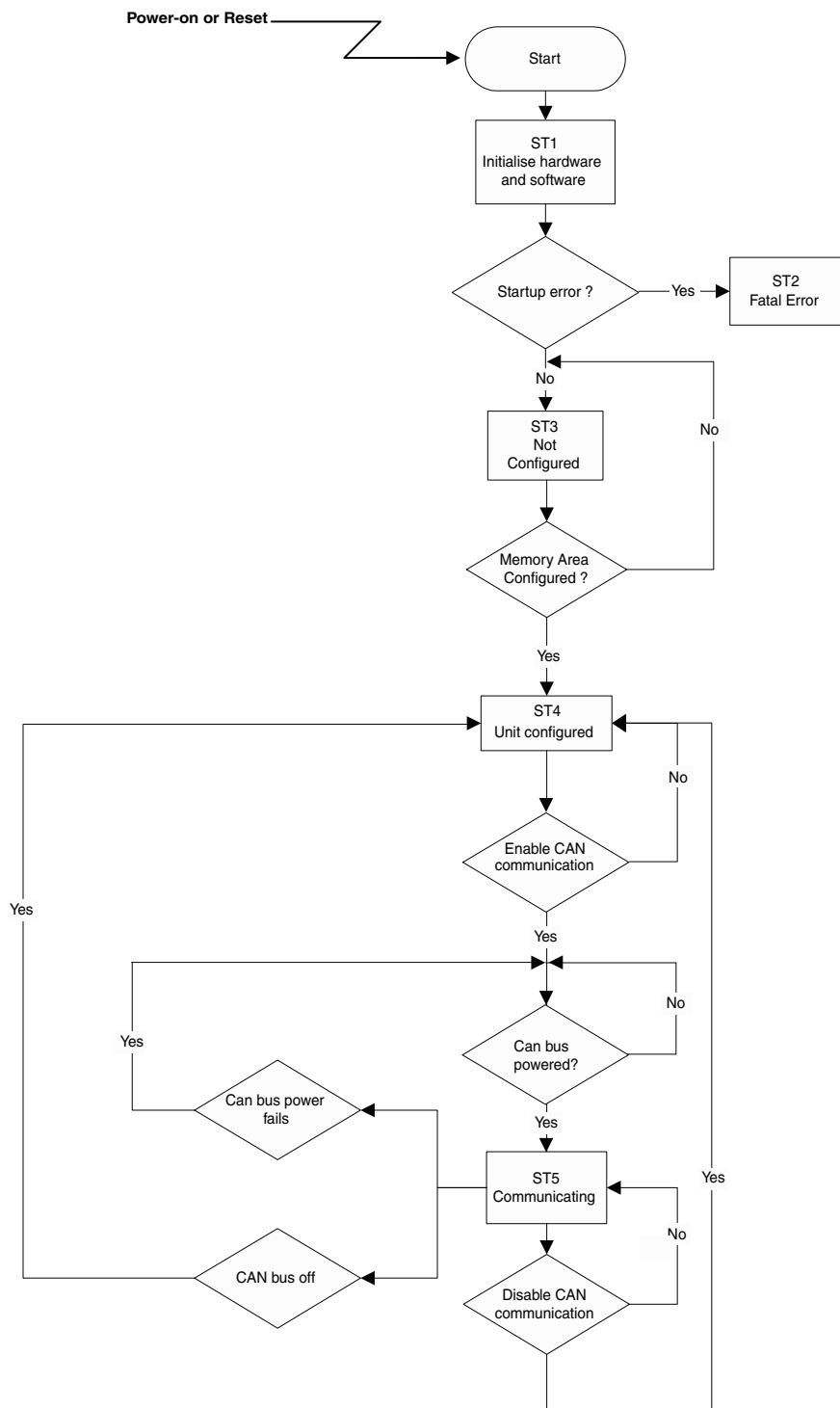
2-2-4 External Dimensions



2-2-5 Unit States

The User Defined CAN Unit has several states. These states are visualized in the figure below. Events will force the Unit to switch to a different state, these events can be user initialized or caused by system behaviour (or error conditions detected in the system).

The different states are defined and described in the first table below. In the second table the events are defined which will force the Unit into a different state. In normal operating mode the Unit is initialized, and no start-up error occurs. The states are refreshed during every Unit Cyclic refresh. Changed state information is reflected as changes in device variables for CJ-series Units.



State	Name	Cyclic refresh of message command(s)	Description
ST1	Initializing	No	The Unit executes start-up tests and initializes the CPU bus communications.
ST2	Start-up error	No	A start-up error was detected during start-up tests or during the initialization of the CPU communication bus.
ST3	Not configured	Yes	The Unit started without start-up errors but is not yet configured using message commands.
ST4	Configured	Yes	The Unit has received the memory locations and a number of output and input message buffers.
ST5	Communicating	Yes	CAN messages can be sent and received if the network is powered.

● Events

Events will force the Unit to switch to a different state. These events can be user initialized, caused by system behavior or error conditions detected in the system. Changed state information is reflected as changes in device variables for CJ-series Units. In normal operating mode the Unit is initialized and no start-up errors occur.

Event	Name	Description
EV1	Configure memory areas	The memory areas for message buffers, send triggers and receive flags have been successfully configured through message commands.
EV2	Configure one CAN message	One of the output message buffers or input message buffers has been successfully configured through message commands.
EV3	Enable communications	CAN messaging has been successfully enabled with device variables for CJ-series Units
EV4	Disable communications	CAN messaging has been disabled with device variables for CJ-series Units
EV5	Bus-off	CAN messaging has been transitioned to the bus off state, e.g. a bus-error occurred.

Bus Off State

In case the User Defined CAN Unit detects an abnormal rate of errors on the bus, the Unit will go in *bus off*. This means that all CAN communication is disabled and the Unit will try to go online after the user enables the communication (normally this is initiated in the user program). In the *bus off* state the Unit will:

- Turn ON the *_*BusoffErr*
- Activate event EV5, i.e. disable communications
- Turn ON the ERR LED indicator

3

Data Exchange with the CPU Unit

This section provides information on exchanging data between the User Defined CAN Unit and NJ-series CPU Units.

3-1	Data Exchange with the CPU Unit	3-2
3-1-1	Data Flow	3-2
3-1-2	Accessing From the User Program	3-5
3-2	Device Variable for CJ-series Unit	3-8
3-2-1	Enable CAN Communications	3-8
3-2-2	Status Communication	3-9
3-2-3	Number of Delayed Messages	3-12
3-2-4	Number of Received Messages Waiting to be Processed	3-12

3-1 Data Exchange with the CPU Unit

Data exchange between this Unit and the CPU Units uses the I/O port and memory for CJ-series Unit allocated to the User Defined CAN Unit.

3-1-1 Data Flow

The CPU Units and CJ-series User Defined CAN Units exchange data as shown in the table and chart below.

Data exchange type

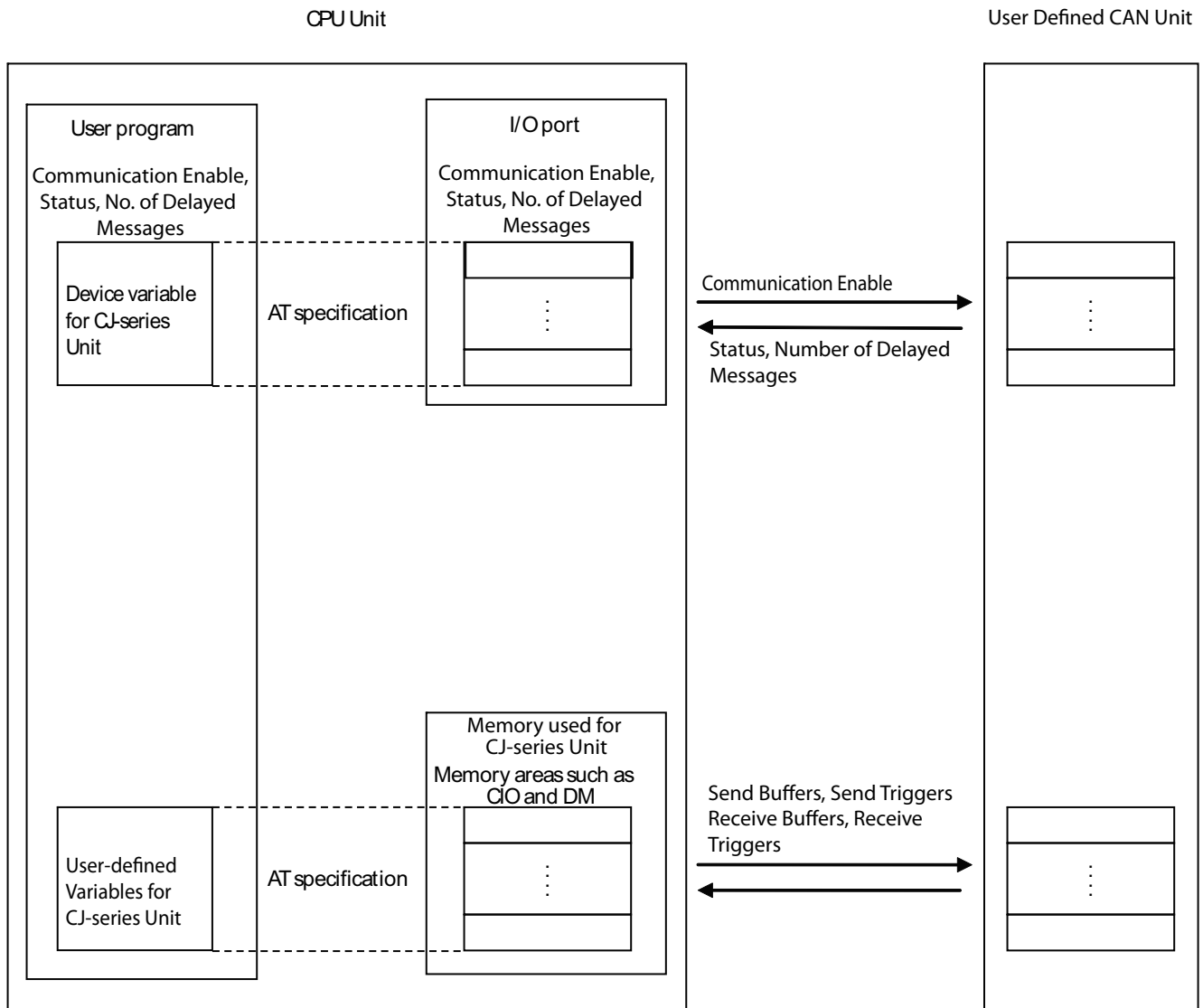
Access methods from the user program	AT specification destination	Data exchange timing	Unit data type
Device variable for CJ-series Unit	I/O port	During I/O refresh	Communication Enable
			Status
			Number of Delayed Send Messages
			Number of Delayed Receive Messages
User-defined variable	Memory used for CJ-series Unit	During I/O refresh	<ul style="list-style-type: none"> • Send Buffers • Send Triggers • Receive Buffers • Receive Triggers
		At I/O refresh after instruction execution	<ul style="list-style-type: none"> • Direct Transmit of 11-bit ID CAN Message • Direct Transmit of 29-bit ID CAN Message

Unit Cyclic Refresh

The User Defined CAN Unit performs a cyclic refresh for five memory areas of the CPU. In one of these memory areas, control and status flags are present and the state information is included in these flags.

During a Unit Cyclic refresh, data is transferred between the CPU and the Unit. The Unit determines independently when to cyclically refresh its data. This data is not processed immediately after the Unit Cyclically refreshes. Several CPU Cyclic refreshes may occur between two Unit Cyclic refreshes. This means that rising/falling edges and/or data changes must be handled with care.

Item	Location/Variable	Description
Control and Status Flags	*_EnbCANC omm, *_StaComm	Flags to control the Unit behavior and to show the status of the Unit.
Send Triggers	Configurable	Triggers to send CAN messages on demand. Bits in the send trigger area will send a message if the bit-value has a rising edge and the bit is associated with an output buffer. Command messages are used to define a send trigger area, to associate an identifier with an output buffer and a send trigger.
Receive Flags		Flags indicating which messages have been received. A received message is identified by its identifier and in case a bit in the receive flags area is associated with the received identifier, this bit is set. The message content is placed in the corresponding input buffer. Message commands are used to define a receive flag area, to associate an identifier with an input buffer and a receive flag.
CAN Output Message Buffers		The message content to send. Every output buffer can only have one message.
CAN Input Message Buffers		The content of the received messages. Every input buffer can have only one message at a time. New incoming messages will overwrite the previous content of the input buffer.



Device Variable for CJ-series Unit

Device variables for CJ-series Units are variables for which AT is specified for the I/O port explained below. The user program uses device variables for CJ-series Unit to access the Configuration Unit such as the User Defined CAN Unit.

For allocation of the device variables for CJ-series Unit to the I/O port, refer *How to Create Device Variables for CJ-series Unit* (P. 3-6).

● I/O Port

An "I/O port" is a logical interface for data exchange by a CPU Unit with a User Defined CAN Unit or other Configuration Unit.

An I/O port has a unique pre-defined name for each unit model and function.

An I/O port is automatically created by preparing the Unit Configuration with Sysmac Studio.

For details on the I/O ports defined for User Defined CAN Unit, refer to *3-2 Device Variable for CJ-series Unit*.

● Communication Enable

Execution instruction for the CAN communication enable function from the CPU Unit to the User Defined CAN Unit.

● Status

User Defined CAN Unit communication status information.

● Number of Delayed Messages

Message delay information for sent and received messages.

User-defined Variable

Bits and words are allocated to the memory used for CJ-series when you configure the User Defined CAN unit with message commands.

- send buffers specification
- send triggers specification
- receive buffers specification
- receive triggers specification

To use this area from the user program, you need to create a user-defined variable of AT specification.

3-1-2 Accessing From the User Program

From the user program, various types of information are exchanged using AT specified device variables for CJ-series Unit that are allocated to the I/O ports, and AT specified user-defined variables that are allocated to slave allocation areas.

From the user program, the following is used to exchange various types of information:

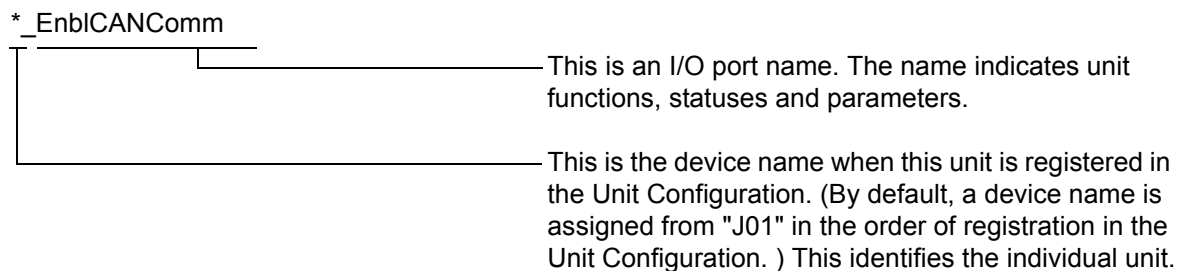
Data type		I/O port, memory used for CJ-series Unit	Access method
Communication Enable, Status, Number of Delayed Messages		Operation Data	Device variables for CJ-series Unit
Memory Areas	Send triggers area	Any area of CIO, DM, WR, HR and EM	User-defined variables
	Receive flags area		
	Output buffers area		
	Input buffers area		

How to Create Device Variables for CJ-series Unit

Use I/O Map in Sysmac Studio to allocate device variables for CJ-series Unit to an I/O port. Specify variable names using one of the methods shown below.

1. Select and allocate existing variables.
2. Input a new variable name.
3. Automatically create with "Create Device Variable".

The following shows the structure of a variable name created automatically with method 3.



For details on device variables for CJ-series Unit, refer to the following:

3-2 Device Variable for CJ-series Unit

In the explanations from here on, the device name automatically created is used as the device variable name for CJ-series Unit, for example *_EnbICANComm.

For details on the memory for CJ-series Unit, refer to *NJ-series CPU Unit Software User's Manual* (Cat. No. W501).

How to Create User-defined Variables

Use the Sysmac Studio to register user-defined variables to the variable table. Use user-defined variables to specify the AT specification of the memory used for CJ-series Unit to which buffer and trigger areas are allocated.

Generally, array variables are created.

Below is an example of allocation to user-defined variables. See *A-2 User Program Example* for more information.

- Configuration of the User Defined CAN unit using the message command 2902
- 1 send buffer, 1 receive buffer

One send or receive buffer occupies 5 words of CPU memory (Data Length Code (DLC) and the 8 bytes of data). One send trigger or receive flag occupies one bit of CPU memory. The number of words occupied by all triggers or flags equals the number of send or receive messages divided by 16, rounded up. Trailing padding bits are ignored (see section *4-2-1 Configure Memory Areas (2902)* for more information).

In this case, buffer and trigger data are allocated as follows:

Words allocated	I/O data
W0000 to W0004	Send Buffer starting word
W100	Send Trigger starting word
EM Bank 0, 000000 to 000004	Receive Buffer starting word
W110	Receive Trigger starting word

Allocate the I/O data to the user-defined variables as shown in the example below.

Name	Data Type	Initial Value	AT	Retain	Constant	Network Publish	Comment
SendBuffer	ARRAY[0..4] OF WORD		%W0	<input type="checkbox"/>	<input type="checkbox"/>	Do not publish	
SendTrigger	ARRAY[0..15] OF BOOL		%W100.00	<input type="checkbox"/>	<input type="checkbox"/>	Do not publish	
ReceiveBuffer	ARRAY[0..4] OF WORD		%E0_0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Do not publish	
ReceiveFlag	ARRAY[0..15] OF BOOL		%W110.00	<input type="checkbox"/>	<input type="checkbox"/>	Do not publish	
J01_EmbiCANComm	BOOL		IOBus://rack#0/slot#0/EmbiCANComm	<input type="checkbox"/>	<input type="checkbox"/>	Do not publish	

This example uses the following data types.

- ARRAY[..**..**]OF WORD
- ARRAY[..**..**]OF BOOL
- BOOL

Use this data type or multiple data types to create user-defined variables according to the user program.



Additional Information

For details on memory used for CJ-series Unit, variable allocation, and user-defined variable registration, refer to the *Sysmac Studio Version 1 Operation Manual* (Cat. No. W504).

3-2 Device Variable for CJ-series Unit

When you operate and reference software switches and statuses, use the following device variables for CJ-series Unit allocated to the I/O port of this Unit.

Name of device variable for CJ-series Unit	Type	R/W	Area
*_EnbICANComm	BOOL	RW	Enable CAN Communications
*_StaComm	WORD	R	Status Communication
*_DelayMsgNo	WORD	R	Number of Delayed Messages
*_ProcMsgNo	WORD	R	Number of Messages to be Processed

The function of each device variable for CJ-series Unit is explained below described with variable names automatically created on the I/O Map View window.

3-2-1 Enable CAN Communications

Use the following BOOL-type device variable for CJ-series Units to enable CAN communications from the user program.

The variable executes some functions when changed to TRUE or changed to FALSE and has an effect on variables *_RcvOver, *_BusoffErr and *_SendOver variables. For details on these switches, refer to 3-2-2 Status Communication.

Once a function is set by changing a switch, it is not saved if the power is turned OFF and ON again and will return to the OFF state

Name of device variable for CJ-series Unit	Type	R/W	Area	Function
*_EnbICANComm	BOOL	RW	Enable CAN Communications	<Operation specifications> FALSE→TRUE: <ul style="list-style-type: none"> • In state ST1 (initializing) or ST3 (not configured), generate error ER9 (network parameter file lost). • In state ST4 (configured): <ol style="list-style-type: none"> 1. Generate event EV3 (enable communications). 2. Reset *_RcvOver. 3. Reset *_BusoffErr. 4. Reset all receive flags. TRUE→FALSE: <ul style="list-style-type: none"> • In state ST5 (communicating): <ol style="list-style-type: none"> 1. Generate event EV4 (disable communications) 2. Reset *_SendOver. 3. Reset all send triggers • In all other states, there is no action.

3-2-2 Status Communication

Use one of the following device variables for CJ-series Unit to monitor Status. Communication from the user program:

- WORD-type device variable for CJ-series Unit holding all switch functions contained in Status Communication
- BOOL-type device variable for CJ-series Unit separating functions per each switch contained in Status Communication

Name of device variable for CJ-series Unit	Type	R/W	Area	Function
*_StaComm	WORD	R	Status Communication	Bit 00: Reserved by System Bit 01: Reserved by System Bit 02: Enabled Communication Bit 03: CAN Message Received Bit 04: Reserved by System Bit 05: Reserved by System Bit 06: Send Queue Overflow Bit 07: Receive Queue Overflow Bit 08: Reserved by System Bit 09: Network Power-Failure Bit 10: Bus Off Event Bit 11: Reserved by system Bit 12: Reserved by system Bit 13: Reserved by system Bit 14: Reserved by system Bit 15: Error in Error Log Default: 16#0000

The following device variables for CJ-series Unit are used to reference individual information.

Name of device variables for CJ-series Unit	Type	R/W	Area	Function
*_EnblComm	BOOL	R	Enabled Communication	<p><Operation specifications></p> <p>TRUE: CAN communications enabled and unit state is ST5 (communicating).</p> <p>FALSE: CAN communications disabled and unit state is not ST5 (communicating).</p> <p>Default: FALSE</p>
*_MsgRcv	BOOL	R	CAN Message Received	<p><Operation specifications></p> <p>TRUE: No new configured CAN message received since last Unit Cyclic refresh.</p> <p>FALSE: New configured CAN message received since last Unit Cyclic refresh.</p> <p>Default: FALSE</p>
*_SendOver	BOOL	R	Send Queue Overflow	<p><Operation specifications></p> <p>TRUE: Send queue overflow occurred. Overflowed messages will be delayed. Reset to FALSE when *_EnblCANComm is switched from TRUE→FALSE in state ST5 (communicating).</p> <p>FALSE: The send queue has accepted all messages to be sent.</p> <p>Default: FALSE</p>
*_RcvOver	BOOL	R	Receive Queue Overflow	<p><Operation specifications></p> <p>TRUE: Receive queue overflow occurred. Some messages have been discarded. Reset to FALSE when *_EnblCANComm is switched from FALSE→TRUE in state ST4 (configured).</p> <p>FALSE: All messages received were processed.</p> <p>Default: FALSE</p>

Name of device variables for CJ-series Unit	Type	R/W	Area	Function
*_NetPwrErr	BOOL	R	Network-Power Failure	<Operation specifications> TRUE: Error ER10 active indicating a power failure while communicating. FALSE: Network power is within acceptable range. Default: FALSE
*_BusoffErr	BOOL	R	Bus Off Event	<Operation specifications> TRUE: A bus off event EV5 (bus off) has been generated. Reset to FALSE when *_EnbICANComm is switched from FALSE→TRUE in state ST4 (configured). FALSE: A bus off event did not occur since the last EV3 event (enable communications). Default: FALSE
*_ErrInErrLog	BOOL	R	Error in Error Log	<Operation specifications> TRUE: New errors present in error log since: <ul style="list-style-type: none"> • Startup • Last service of message command 2102 • Last service of message command 2103 FALSE No new errors in error log since: <ul style="list-style-type: none"> • Startup • Last service of message command 2102 • Last service of message command 2103 Default: FALSE

3-2-3 Number of Delayed Messages

When the *_SendOver variable is TRUE, *_DelayMsgNo will provide the number of delayed messages in BCD format. For details on the *_SendOver variable, see 3-2-2 *Status Communication*.

Name of device variable for CJ-series Unit	Type	R/W	Area	Function
*_DelayMsgNo	WORD	R	Number of Delayed Messages (BCD)	Number of delayed messages in BCD format. Data range: 0 to 15 Default: 0

3-2-4 Number of Received Messages Waiting to be Processed

When the *_RcvOver variable is TRUE, *_ProcMsgNo will provide the number of messages that are awaiting processing in BCD format. For details on the *_RcvOver variable, see 3-2-2 *Status Communication*.

Name of device variable for CJ-series Unit	Type	R/W	Area	Function
*_ProcMsgNo	WORD	R	Number of Messages to be processed (BCD)	Number of Messages to be processed (BCD) Data range: 0 to 15 Default: 0

4

Message Communications

This section describes message communications sent from the user program in the CPU Unit.

4

4-1 Overview	4-2
4-1-1 Outline of Message Communications	4-2
4-2 Unit Configuration and Control	4-4
4-2-1 Configure Memory Areas (2902)	4-4
4-2-2 Configure 11-Bit ID Output Message Buffer (2903)	4-10
4-2-3 Configure 29-Bit ID Output Message Buffer (2904)	4-12
4-2-4 Configure 11-Bit ID Input Message Buffer (2905)	4-14
4-2-5 Configure 29-Bit ID Input Message Buffer (2906)	4-16
4-2-6 Direct Transmit of an 11-bit ID CAN Message (2907)	4-17
4-2-7 Direct Transmit of an 29-bit ID CAN Message (2908)	4-19
4-2-8 Setting the CAN Bit Rate and Sample Point (2909)	4-21
4-2-9 Error Log Read (2102)	4-22
4-2-10 Error Log Clear (2103)	4-23

4-1 Overview

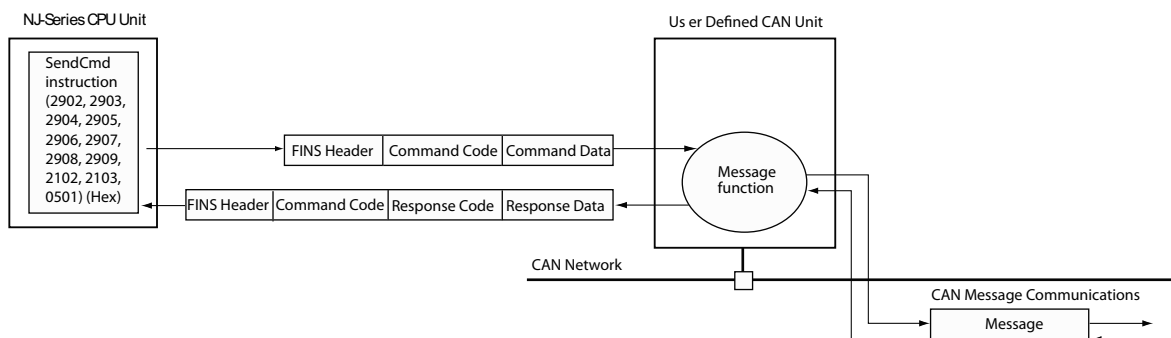
Message communications enable messages to be sent between nodes on a CAN network when required by system conditions. The messages can be sent between CPUs, between an OMRON CPU and a User Defined CAN unit, or between slaves. They can be used to send/receive data, read time data, error histories, and other data or control operation, e.g., refreshing with user-specified values.

To operate a CAN network, the User Defined CAN Unit must be configured using message communications. Bus parameters, configuration data and defining parameterization data are set with message communications in the user program (see 4-2 Unit Configuration and Control for more information).

4-1-1 Outline of Message Communications

Message communications are sent using the SendCmd instruction for NJ-series controllers.

● Overall Structure



● Command Codes

Command codes are represented by a 2-byte hexadecimal code. Commands always begin with a 2-byte command code and any parameters that are required follow the command code. More information about command codes and parameters can be found in the following pages.

Command Code	Name	Type	Page
2902	Configure Memory Areas	User Defined CAN Unit	4
2903	Configure 11-Bit ID Output Message Buffer		10
2904	Configure 29-Bit ID Output Message Buffer		12
2905	Configure 11-Bit ID Input Message Buffer		14
2906	Configure 29-Bit ID Input Message Buffer		16
2907	Direct Transmit of an 11-Bit ID Can Message		17
2908	Direct Transmit of a 29-Bit ID Can Message		19
2909	Setting the CAN Bit Rate and Sample Point		21
2102	Error Log Read	General Service	22
2103	Error Log Clear		23



Additional Information

The User Defined CAN Unit supports sending and receiving functions for messages. Every message in CAN communication has an identifier. The configurable options for these message identifiers are 29-bit or 11-bit size for each input and output message buffer. See sections below for more information.

● Response Codes

Response codes are represented by a 2-byte hexadecimal code that indicates the results of command execution. The first byte provides the main response code (MRC), which classifies the results, and the second byte provides the sub-response code (SRC), which provides details on the results.

The main response codes are listed below. More information about response codes and sub-response codes can be found in the following pages.



Additional Information

Refer to the *SYSMAC CS/CJ Series Communication Commands Reference Manual* (Cat. No. W342) for further details on response codes, including sub-response codes (SRC).

Response Code (MRC)	Name
00	Normal Completion
01	Local Node Error
02	Destination Node Error
03	Communications Controller Error
04	Unsupported Setting Error (Service Not Supported)
05	Routing Error
10	Command Format Error
11	Parameter Error
20	Read Not Possible
21	Write Not Possible
22	Not Executable In Current Mode
23	No Such Device
24	Start/Stop Not Possible
25	Unit Error
26	Command Error
30	Access Right Error
40	Abort

4-2 Unit Configuration and Control

In order to operate a CAN network, each unit in the network needs to be configured with software (user program) configuration.

The total process of network and unit configuration involves:

- Setting up the physical network topology
- Setting up the bus parameters, which define the baud rate and the bus timing parameter sample point
- Defining the configuration data, i.e. defining the process data, which will be exchanged between the User Defined CAN Unit and other nodes on the CAN network
- Defining the parameterization data for the User Defined CAN Unit, which defines the filtering of message identifiers and the configuration of message buffers in the CPU system.

For parameterization of the User Defined CAN Unit, memory areas and message parameterization need to be configured upon power-up. The software configuration steps (user program) that must be done after every 'power on' of the unit, to insure correct operation are:

- (1) Setting memory area (buffer) allocations with command code 2902
- (2) Setting parameters for sending messages with 11-bit identifier, or 29-bit identifier
- (3) Setting parameters for receiving messages with 11-bit identifier, or 29-bit identifier
- (4) Enable CAN communications, device variable `*_EnbICANComm`

4-2-1 Configure Memory Areas (2902)

With this command code, the following items are configured:

- send buffer location
- send trigger location
- receive buffer location
- receive flag location
- number of send and receive messages

One send or receive buffer occupies 5 words of CPU memory (Data Length Code (DLC) and the 8 bytes of data). One send trigger or receive flag occupies one bit of CPU memory. The number of words occupied by all triggers or flags equals the number of send or receive messages divided by 16, rounded up. Trailing padding bits are ignored.

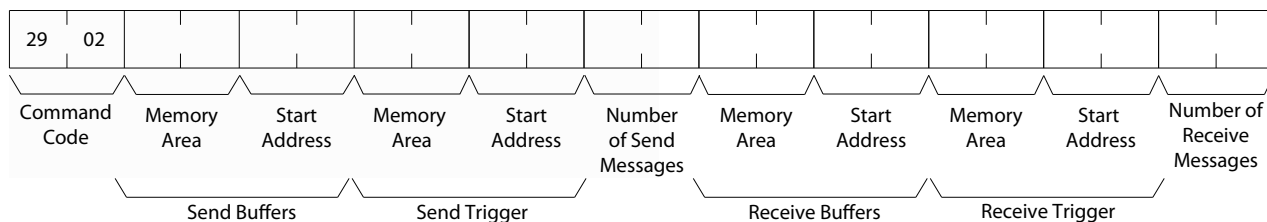
Note This command can be used only if communication has been disabled (by turning OFF device variable `*_EnbICANComm`) and this disabling is confirmed by the unit (by an OFF state of device variable `*_EnbIComm`).



Additional Information

See section A-2 *User Program Example* for more information about the Configure Memory Area (2902) message command.

Command Block



The following table defines the minimum and maximum values for each item in the command block.

Setting	Minimum Value (Hex)	Maximum Value (Hex)
Memory area of send buffers	0001	0014
Start address of send buffer	0000	7FFF
Memory area of send trigger	0001	0014
Start address of send trigger	0000	7FFF
Number of send messages	0000	0280
Memory area of receive buffers	0001	0014
Start address of receive buffer	0000	7FFF
Memory area of receive trigger	0001	0014
Start address of receive trigger	0000	7FFF
Number of receive messages	0000	0280

The following table defines the numerical values for CPU memory area types. The address range for each memory area type is also provided.

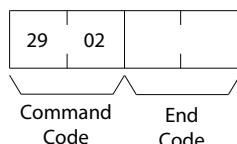
Memory Area Numerical Value (hex)	Memory Area Type	Maximum Address (hex) ¹
0001	CIO area	17FF
0002	DM area	7FFF
0003	Work area	01FF
0004	Holding area	05FF
0008	EM bank 0	7FFF
0009	EM bank 1	7FFF
000A	EM bank 2	7FFF
000B	EM bank 3	7FFF
000C	EM bank 4	7FFF
000D	EM bank 5	7FFF
000E	EM bank 6	7FFF
000F	EM bank 7	7FFF
0010	EM bank 8	7FFF
0011	EM bank 9	7FFF
0012	EM bank A	7FFF
0013	EM bank B	7FFF
0014	EM bank C	7FFF

Note 1 Using the maximum address send/receive buffers will result in an error because the buffers consume 5 words each. Therefore, use the following equation to determine the maximum allowable memory area for a start address buffer:

- Maximum Buffer Start Address = Maximum Address - (Number of Send/Receive Messages x 5)

Response Block

Issuing a Configure Memory Area command will result in the following response End Codes.



End Code (hex)	Description	Condition	Correction
No response	---	Unit state = ST1 or ST2	Restart the CPU, make sure that the unit is mounted and wired correctly
0000 ^{1,2}	Normal completion	Unit state = ST3 or ST4 and all parameters are in range	Command is executed and operation is normal.
1001	Command length exceeds maximum command length	---	Too many parameters sent in the command. Correct the command and re-send.
1002	The command length is insufficient for the smallest command	---	Too few parameters sent in the command. Correct the command and re-send.
1101	No area type	Unit state = ST3 or ST4 and the memory area is not available	Specified memory area type is not valid. Specify a correct memory area type.
1104	Address range over	Unit state = ST3 or ST4 and the end address of an area is not in range	Correct the memory area size.
1109	Mutual relation error	Unit state = ST3 or ST4 and the memory areas overlap	Memory area overlap. Correct the memory area specification.
110C	Parameter error	Unit state = ST3 or ST4 and one or more parameters are not in range	Correct the parameters. The number of send and receive message may not exceed a number of 0280 Hex.
2201	Not executable in current mode	Unit state = ST5	This message command can only be executed if communication is disabled (by clearing *_EnbICANComm and this disabling is confirmed when *_EnbIComm is FALSE)

Note 1 On response code 0000, event EV1 is generated. The unit configures:

- The Send Buffers area using the provided memory area and start address
- The Send Trigger area using the provided memory area and start address
- The quantity of Send Message Buffers
- The Receive Buffers area using the provided memory area and start address.
- The Receive Trigger area using the provided memory area and start address
- The quantity of Receive Message Buffers

2 On response code 0000, the unit will set to zero:

- The Receive Message buffers that are added to the configuration
- The Receive Trigger bits that are added to the configuration (the Send Trigger bits are not cleared)

● Send Trigger Area

Every bit in this area can be a trigger to send a CAN message located in the send buffer area. The bit is only a send trigger if:

- a corresponding send buffer is defined for the bit
- the send mode SM1 is part of the total send mode for the send buffer
- the bit has a rising edge transition (OFF to ON)

The words in this memory area are controlled by both the CPU and/or the Unit.

Controlled By	Action	Function
CPU	Bit Set	In mode SM1 (output messages are sent by triggering a bit in trigger send area table). After the next Unit Cyclic refresh, the unit will send the corresponding message. In other modes: no function.
	Bit Reset	Unspecified functionality: the corresponding message may send or not send.
User Defined CAN Unit	Bit Set	The unit shall not set bits in this area.
	Bit Reset	In mode SM1 (output messages are sent by triggering a bit in trigger send area table). The unit indicates that the corresponding message has been sent by clearing the send bit in the send area table. In other modes: the unit shall not reset the bit.

The location R of the send trigger words is configured with the command code 2902. The length of the area is determined via the number of configured send message buffers. If the number of configured send message buffers is A , the length L is:

$$L = \frac{(A + 15)}{16}$$

The result is rounded down to the nearest whole number and remaining trigger bits will be ignored.

The send trigger area table has the following layout:

Word	Bit															
R	15	14	13	12	11	10	09	08	07	06	05	04	30	02	01	00
R+1	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
~																
R+L	16*L +15	16*L +14	16*L +13	16*L +12	16*L +11	16*L +10	16*L +09	16*L +08	16*L +07	16*L +06	16*L +05	16*L +04	16*L +03	16*L +02	16*L +01	16*L +00

● Receive Trigger Area

This area has flags indicating that a CAN message is received in the receive buffer corresponding to the receive bit. Every input message buffer has its corresponding bit. The bit value means:

ON = An input message with the configured message identifier of this input message buffer has been received since last Unit Cyclic refresh.

OFF = No input message with the configured message identifier of this input message buffer has been received since last cyclic refresh or the message has been discarded due to receive queue overflow (*_RcvOver is ON).

If any of these bits is set, *_MsgRcv is ON.

If all these bits are reset, *_MsgRcv is OFF.

The location *S* of the receive trigger word is configured with the command code 2902. This memory area contains input words. The length of the area is determined by the number of configured receive message buffers. If the number of configured receive message buffers is *B*, the length is:

$$L = \frac{(B + 15)}{16}$$

The result is rounded down to the nearest whole number and remaining receive bits will be ignored.

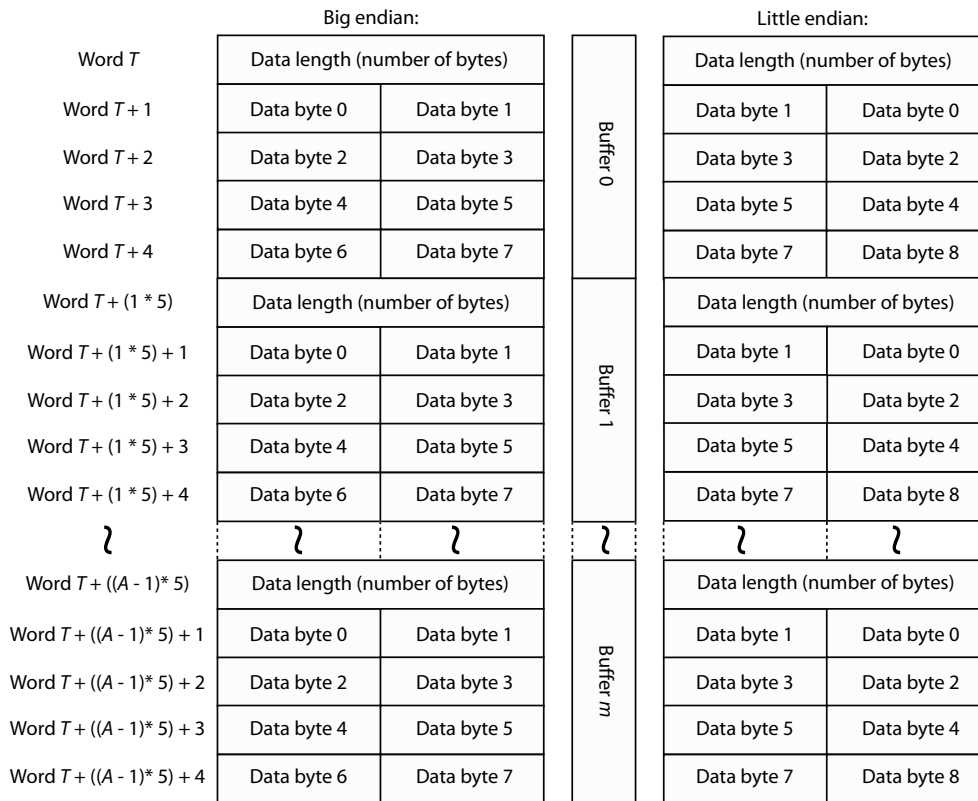
The receive trigger area table has the following layout:

Word	Bit															
S	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
S+1	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
~																
S+L	16*L +15	16*L +14	16*L +13	16*L +12	16*L +11	16*L +10	16*L +09	16*L +08	16*L +07	16*L +06	16*L +05	16*L +04	16*L +03	16*L +02	16*L +01	16*L +00

● **Send Buffer Area**

In this area one or more send buffers are present. Every output buffer can have only one CAN message content for sending.

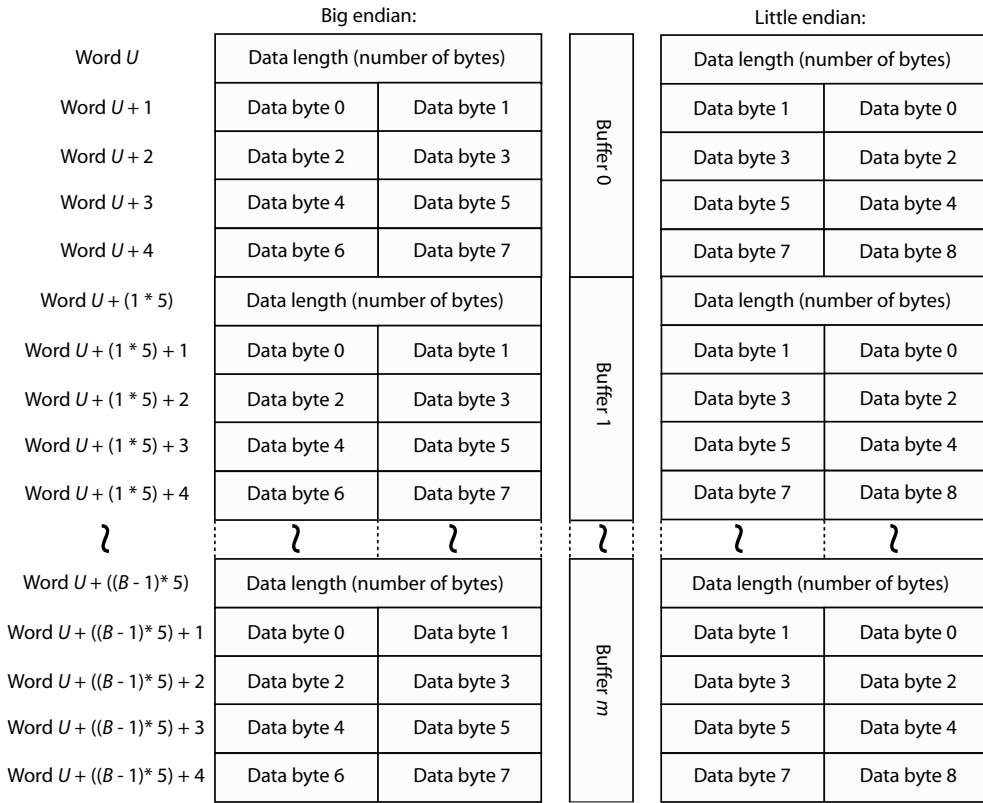
The location *T* of the CAN send message buffer is configured with the command code 2902. This memory area contains output words. Given the number of configured send message buffers as *A*, the area has the layout specified below. The layout depends on the configuration supplied via the command codes 2903 or 2904. By selecting little or big endian format (adjusted when the command is executed) a byte swap is performed (see section 4-2-2 *Configure 11-Bit ID Output Message Buffer (2903)* and 4-2-3 *Configure 29-Bit ID Output Message Buffer (2904)* for more details).



● **Receive Buffer Area**

In this area one or more receive buffers are present. Every input buffer can have only one CAN message content for receiving.

The location U of the CAN receive message buffers is configured with the command code 2902. This memory area contains input words. Given the number of configured receive message buffers as B , the area has the layout specified below. The layout depends on the configuration supplied via the command codes 2905 or 2906. By selecting little of big endian format (adjust when the command is executed) a byte swap is performed (see section 4-2-4 *Configure 11-Bit ID Input Message Buffer (2905)* and 4-2-5 *Configure 29-Bit ID Input Message Buffer (2906)* for more details).



Additional Information

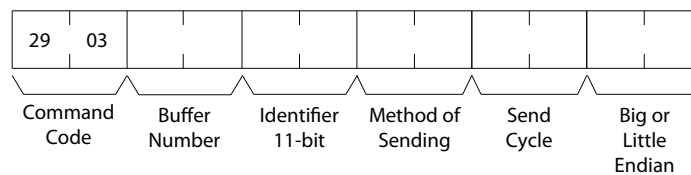
See section 3-1-2 *Accessing From the User Program* for more detail regarding Send and Receive Buffer areas.

4-2-2 Configure 11-Bit ID Output Message Buffer (2903)

Each Output (Send) Buffer requires the following settings that are configured using the command code 2903.

- the 11-bit identifier
- the method of sending (SM1 = triggered, SM2 = on change or SM3 = cyclic)
- the send cycle
- the use of little endian or big endian

Command Block

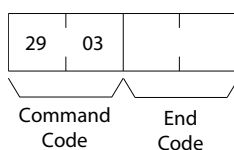


The following table defines the minimum and maximum values for each item in the command block with setting details.

Setting	Minimum Value (Hex)	Maximum Value (Hex)	Setting Details
Buffer Number	0000	027F	Defines which buffer to configure (0 to 639)
Identifier 11-bit	0000	07FF	Identifier value
Method of Sending	0001	0004	<ul style="list-style-type: none"> 0001: Message will be sent when the corresponding bit in the send trigger area changes from OFF to ON. 0002: Message will be sent on a cyclic period based on the Send Cycle value in milliseconds. 0003: Invalid setting 0004: Message will be sent on change of data.
Send Cycle (milliseconds)	0000	FFFF	<ul style="list-style-type: none"> 0000: Setting when Method of Sending is 0001 or 0004 0001 to FFFF: Send Cycle in milliseconds when Method of Sending is 0002
Big or Little Endian	0000	0001	<ul style="list-style-type: none"> 0000: Big Endian 0001: Little Endian

Response Block

Issuing a Configure 11-Bit ID Output (Send) Message Buffer command will result in the following response End Codes.



End Code (hex)	Description	Condition	Correction
No response	---	Unit state = ST1 or ST2	Restart the CPU and make sure that the unit is mounted and wired correctly.
0000 ¹	Normal completion	Unit state = ST4 or ST5 and all parameters are in range	Command is executed and operation is normal.
1001	Command length exceeds maximum command length	---	Too many parameters sent in the command. Correct the command and re-send.
1002	The command length is insufficient for the smallest command	---	Too few parameters sent in the command. Correct the command and re-send.

End Code (hex)	Description	Condition	Correction
1103	No area type	Unit state = ST3 or ST4 and the memory area is not available	Number of output buffers exceeds the number of output buffers configured by message command 2902. Increase the number of output buffers with message command 2902 or select an output buffer smaller than de maximum of output buffers configured by message command 2902.
1109	Mutual relation error	Unit state = ST4 or ST5 and send mode = 2 and send cycle = 0	Set the send cycle different from zero or change the send mode.
110C	Parameter error	Unit state = ST4 or ST5 and one or more parameters are not in range	Correct the parameters.
2201	Not executable in current mode	Unit state = ST3	Nothing configured. Perform message command 2902 before executing this command
220F	The specified service is being executed	Unit state = ST5 and send mode "changed" and the message is being sent	Disable communications, reconfigure the buffer and enable communications.

Note 1 On response code 0000, the unit configures the output message buffer identified with buffer number with the following properties:

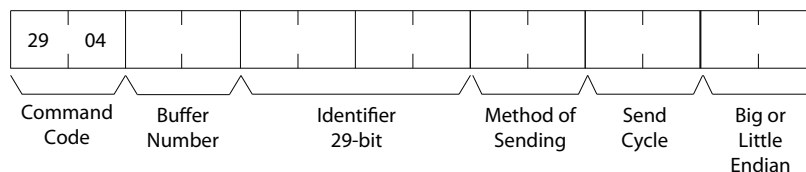
- Extended identifier length (11-bit)
- Identifier number
- Send mode, (see *Sending CAN Messages* in Section 1)
- Send cycle time (if the send mode is 2)
- Big or little endian format, see *Send Buffer Area* on page 8

4-2-3 Configure 29-Bit ID Output Message Buffer (2904)

Each Output (Send) Buffer requires the following settings that are configured using the command code 2904.

- the 29-bit identifier
- the method of sending (SM1 = triggered, SM2 = on change or SM3 = cyclic)
- the send cycle
- the use of little endian or big endian

Command Block

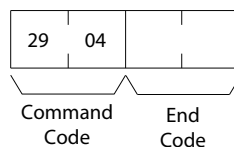


The following table defines the minimum and maximum values for each item in the command block with setting details.

Setting	Minimum Value (Hex)	Maximum Value (Hex)	Setting Details
Buffer Number	0000	027F	Defines which buffer to configure (0 to 639)
Identifier 29-bit	00000000	1FFF FFFF	Identifier value
Method of Sending	0001	0004	<ul style="list-style-type: none"> 0001: Message will be sent when the corresponding bit in the send trigger area changes from OFF to ON 0002: Message will be sent on a cyclic period based on the Send Cycle value in milliseconds. 0003: Invalid setting 0004: Message will be sent on change of data
Send Cycle (milliseconds)	0000	FFFF	<ul style="list-style-type: none"> 0000: Setting when Method of Sending is 0001 or 0004 0001 to FFFF: Send Cycle in milliseconds when Method of Sending is 0002
Big or Little Endian	0000	0001	<ul style="list-style-type: none"> 0000: Big Endian 0001: Little Endian

Response Block

Issuing a Configure 29-Bit ID Output (Send) Message Buffer command will result in the following response End Codes.



End Code (hex)	Description	Condition	Correction
No response	---	Unit state = ST1 or ST2	Restart the CPU, make sure that the unit is mounted and wired correctly
0000 ¹	Normal completion	Unit state = ST4 or ST5 and all parameters are in range	Command is executed and operation is normal.
1001	Command length exceeds maximum command length	---	Too many parameters sent in the command. Correct the command and re-send.

End Code (hex)	Description	Condition	Correction
1002	The command length is insufficient for the smallest command	---	Too few parameters sent in the command. Correct the command and re-send.
1103	Address range destination error	Unit state = ST4 or ST5 and buffer number is greater than or equal to the number of output buffers configured with command 2902	Number of output buffers exceeds the number of output buffers configured by message command 2902. Increase the number of output buffers with message command 2902 or select an output buffer smaller than the maximum of output buffers configured by message command 2902.
1109	Mutual relation error	Unit state = ST4 or ST5 and send mode = 2 and send cycle = 0	Set the send cycle different from zero or change the send mode.
110C	Parameter error	Unit state = ST4 or ST5 and one or more parameters are not in range	Correct the parameters.
2201	Not executable in current mode	Unit state = ST3	Nothing configured. Perform message command 2902 before executing this command.
220F	The specified service is being executed	Unit state = ST5 and send mode "changed" and the message is being sent	Disable communications, reconfigure the buffer and enable communications.

Note 1 On response code 0000, the unit configures the output message buffer identified with buffer number with the following properties:

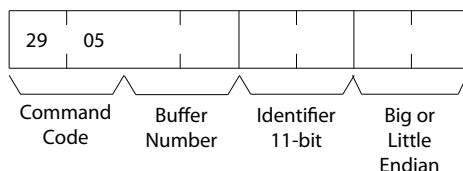
- Extended identifier length (29-bit)
- Identifier number
- Send mode (see *Sending CAN Messages* in Section 1)
- Send cycle time (if the send mode is 2)
- Big or little endian format, see *Send Buffer Area* on page 8

4-2-4 Configure 11-Bit ID Input Message Buffer (2905)

Each Input (Receive) Buffer requires the following settings that are configured using the command code 2905.

- the 11-bit identifier
- the use of little endian or big endian

Command Block

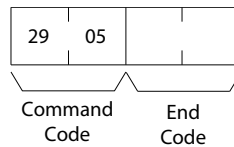


The following table defines the minimum and maximum values for each item in the command block with setting details.

Setting	Minimum Value (Hex)	Maximum Value (Hex)	Setting Details
Buffer Number	0000	027F	Defines which buffer to configure (0 to 639)
Identifier 11-bit	0000	07FF	Identifier value
Big or Little Endian	0000	0001	<ul style="list-style-type: none"> • 0000: Big Endian • 0001: Little Endian

Response Block

Issuing a Configure 11-Bit ID Input (Receive) Message Buffer command will result in the following response End Codes.



End Code (hex)	Description	Condition	Correction
No response	---	Unit state = ST1 or ST2	Restart the CPU, make sure that the unit is mounted and wired correctly
00001	Normal completion	Unit state = ST4 or ST5 and all parameters are in range	Command is executed and operation is normal.
1001	Command length exceeds maximum command length	---	Too many parameters sent in the command. Correct the command and re-send.
1002	The command length is insufficient for the smallest command	---	Too few parameters sent in the command. Correct the command and re-send.
1103	Address range destination error	Unit state = ST4 or ST5 and buffer number is greater than or equal to the number of input buffers configured with command 2902	Number of input buffers exceeds the number of input buffers configured by message command 2902. Increase the number of input buffers with message command 2902 or select a input buffer, smaller than the maximum of input buffers configured by message command 2902.
110C	Parameter error	Unit state = ST4 or ST5 and one or more parameters are not in range	Correct the parameters.
2201	Not executable in current mode	Unit state = ST3	Nothing configured, please perform message command 2902 before executing this command.

Note 1 On response code 0000, the unit configures the input (receive) message buffer identified with buffer number with the following properties:

- Extended identifier length (see additional information below)
- Identifier number
- Big or little endian format, see *Send Buffer Area* on page 8



Additional Information

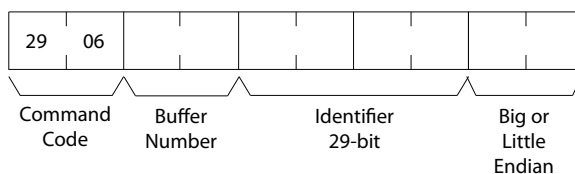
The last command 2905 or 2906 sent to the unit determines whether 11-bit or 29-bit identifiers are used for all input message buffers. If the last command is 2905, 11-bit identifiers will be used. If the last command is 2906, 29-bit identifiers will be used.

4-2-5 Configure 29-Bit ID Input Message Buffer (2906)

Each Input (Receive) Buffer requires the following settings that are configured using the command code 2906.

- the 29-bit identifier
- the use of little endian or big endian

Command Block

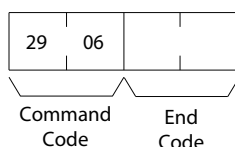


The following table defines the minimum and maximum values for each item in the command block with setting details.

Setting	Minimum Value (Hex)	Maximum Value (Hex)	Setting Details
Buffer Number	0000	027F	Defines which buffer to configure (0 to 639)
Identifier 29-bit	00000000	1FFF FFFF	Identifier value
Big or Little Endian	0000	0001	<ul style="list-style-type: none"> • 0000: Big Endian • 0001: Little Endian

Response Block

Issuing a Configure 29-Bit ID Input (Receive) Message Buffer command will result in the following response End Codes.



End Code (hex)	Description	Condition	Correction
No response	---	Unit state = ST1 or ST2	Restart the CPU, make sure that the unit is mounted and wired correctly
0000 ¹	Normal completion	Unit state = ST4 or ST5 and all parameters are in range	Command is executed and operation is normal.
1001	Command length exceeds maximum command length	---	Too many parameters sent in the command. Correct the command and re-send.
1002	The command length is insufficient for the smallest command	---	Too few parameters sent in the command. Correct the command and re-send.
1103	Address range destination error	Unit state = ST4 or ST5 and buffer number is greater than or equal to the number of input buffers configured with command 2902	Number of input buffers exceeds the number of input buffers configured by message command 2902. Increase the number of input buffers with message command 2902 or select a input buffer, smaller than the maximum of input buffers configured by message command 2902.
110C	Parameter error	Unit state = ST4 or ST5 and one or more parameters are not in range	Correct the parameters.
2201	Not executable in current mode	Unit state = ST3	Nothing configured, please perform message command 2902 before executing this command.

Note 1 On response code 0000, the unit configures the input (receive) message buffer identified with buffer number with the following properties:

- Extended identifier length (see additional information below)
- Identifier number
- Big or little endian format, see *Send Buffer Area* on page 8



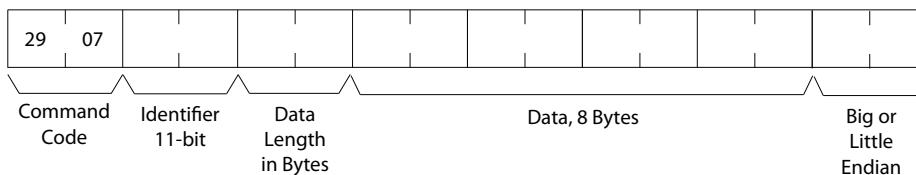
Additional Information

The last command 2905 or 2906 sent to the unit determines whether 11-bit or 29-bit identifiers are used for all input message buffers. If the last command is 2905, 11-bit identifiers will be used. If the last command is 2906, 29-bit identifiers will be used.

4-2-6 Direct Transmit of an 11-bit ID CAN Message (2907)

The instruction is used to transmit a user defined CAN message. This function is only for advanced users that have knowledge of the message structure of the higher layer protocol. The User Defined CAN Unit will not check the validity of the CAN message.

Command Block

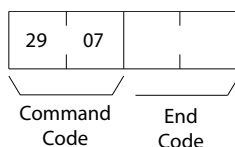


The following table defines the minimum and maximum values for each item in the command block with setting details.

Setting	Minimum Value (Hex)	Maximum Value (Hex)	Setting Details
Identifier 11-bit	0000	07FF	Identifier value
Data Length in Bytes	0000	0008	Data Length
Data	00	FF	Data
Big or Little Endian	0000	0001	<ul style="list-style-type: none"> • 0000: Big Endian • 0001: Little Endian

Response Block

Issuing a Direct Transmit of an 11-bit ID CAN Message command will result in the following response End Codes.



End Code (hex)	Description	Condition	Correction
No response	---	Unit state = ST1 or ST2	Restart the CPU, make sure that the unit is mounted and wired correctly
0000 ¹	Normal completion	Unit state = ST5 and all parameters are in range	Command is executed and operation is normal.
1001	Command length exceeds maximum command length	---	Too many parameters sent in the command. Correct the command and re-send.
1002	The command length is insufficient for the smallest command	---	Too few parameters sent in the command. Correct the command and re-send.

End Code (hex)	Description	Condition	Correction
110C	Parameter error	Unit state = ST5 and one or more parameters are not in range	Correct the parameters,
2201	Not executable in current mode	Unit state = ST3 or ST4	Nothing configured. Perform message command 2902 before executing this command and enable communications by setting *_EnbCANComm ON (confirmed with *_EnbComm ON).
220F	The specified service is being executed	Unit state = ST5 and send mode "changed" and the message is being sent	Sent the message less frequently.

Note 1 On response code 0000, the unit sends a message with an extended identifier length with the provided identifier, data length and data depending on the provided endian flag.



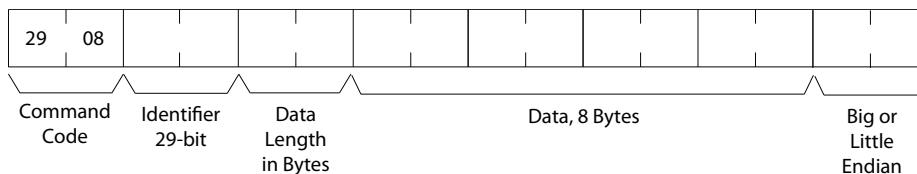
Additional Information

- On all response codes different from 0000, the unit will not transmit the user defined CAN message.
- All of the 8 data bytes are always part of the command message. The actual data length of the CAN message (data length in bytes) can be less than 08. The length of the command must always equal 8 words.

4-2-7 Direct Transmit of an 29-bit ID CAN Message (2908)

The instruction is used to transmit a user defined CAN message. The User Defined CAN Unit will not check the validity of the CAN message.

Command Block

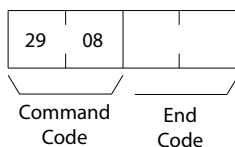


The following table defines the minimum and maximum values for each item in the command block with setting details.

Setting	Minimum Value (Hex)	Maximum Value (Hex)	Setting Details
Identifier 29-bit	00000000	1FFF FFFF	Identifier value
Data Length in Bytes	0000	0008	Data Length
Data	00	FF	Data
Big or Little Endian	0000	0001	<ul style="list-style-type: none"> • 0000: Big Endian • 0001: Little Endian

Response Block

Issuing a Direct Transmit of an 29-bit ID CAN Message command will result in the following response End Codes.



End Code (hex)	Description	Condition	Correction
No response	---	Unit state = ST1 or ST2	Restart the CPU, make sure that the unit is mounted and wired correctly
0000 ¹	Normal completion	Unit state = ST5 and all parameters are in range	Command is executed and operation is normal.
1001	Command length exceeds maximum command length	---	Too many parameters sent in the command. Correct the command and re-send.
1002	The command length is insufficient for the smallest command	---	Too few parameters sent in the command. Correct the command and re-send.
110C	Parameter error	Unit state = ST5 and one or more parameters are not in range	Correct the parameters,
2201	Not executable in current mode	Unit state = ST3 or ST4	Nothing configured, please perform message command 2902 before executing this command and enable communications by setting *_EnbICANComm ON (confirmed with *_EnbIComm ON).
220F	The specified service is being executed	Unit state = ST5 and send mode "changed" and the message is being sent	Send the message less frequently.

Note 1 On response code 0000, the unit sends a message with an extended identifier length with the provided identifier, data length and data depending on the provided endian flag.



Additional Information

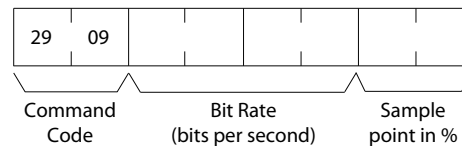
- On all response codes different from 0000, the unit will not transmit the user defined CAN message.
- All of the 8 data bytes are always part of the command message. The actual data length of the CAN message (data length in bytes) can be less than 08. The length of the command must always equal 8 words.

4-2-8 Setting the CAN Bit Rate and Sample Point (2909)

The bit rate can be set in 2 ways:

- The 8 bit rates recommended by CANopen can be selected with the DIP-switches on the front of the Unit. This gives the user some pre-defined bit rates.
- If the User wants to set another bit rate, the bit rate can be set via the message command Setting the CAN bit rate and sample point (2909). If the values for these settings are within the ranges specified in the table below, the bit rate is set according these settings. If a wrong setting is made, the bit rate is set to the bit rate selected with the DIP-switches at the front of the Unit. If the command is correct, the Unit will return a response of the specified bit rate and sample point which are in use.

Command Block

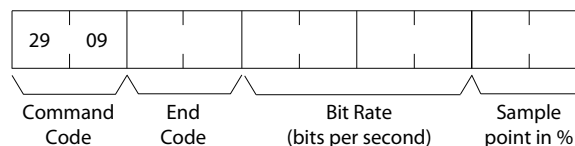


The following table defines the minimum and maximum values for each item in the command block with setting details.

Setting	Minimum Value (Hex)	Maximum Value (Hex)	Setting Details
Bit Rate	0000 0000	0012 12D0 (1.25 Mbit/sec)	Bit rate
Sample Point	0000	0064	See 1-2-1 CAN Communication Protocol

Response Block

Issuing a Setting the CAN Bit Rate and Sample Point command will result in the following response End Codes (apart from the standard command length checks).



End Code (hex)	Description	Condition	Correction
No response	---	Unit state = ST1 or ST2	Restart the CPU and make sure that the unit is mounted and wired correctly
0000 ¹	Normal completion	Unit state = ST3 or ST4 and all parameters are in range	Command is executed and operation is normal.
1001	Command length exceeds maximum command length	---	Too many parameters sent in the command. Correct the command and re-send.

End Code (hex)	Description	Condition	Correction
1002	The command length is insufficient for the smallest command	---	Too few parameters sent in the command. Correct the command and re-send.
110C	Parameter error	Unit state = ST3 or ST4 and one or more parameters are not in range	Correct the parameters.
2201	Not executable in current state	Unit state = ST5	This message command can only be executed if communication is disabled (by turning OFF *_EnbICANComm and confirmed when *_EnbIComm is OFF)

Note 1 On response code 0000, the unit will set a bit rate and a sample point. Not all bit rates and all sample points in the valid range are available.



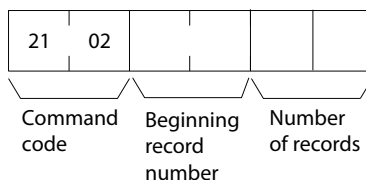
Additional Information

- Setting the bit rate and sample point to zero instructs the unit to use the DIP switch settings. The zero values cause the unit to revert to the DIP switch setting. Any other values will result in an override of the DIP switch settings with the commanded values.

4-2-9 Error Log Read (2102)

The ERROR LOG READ command reads a specified number of error records from the error log.

Command Block

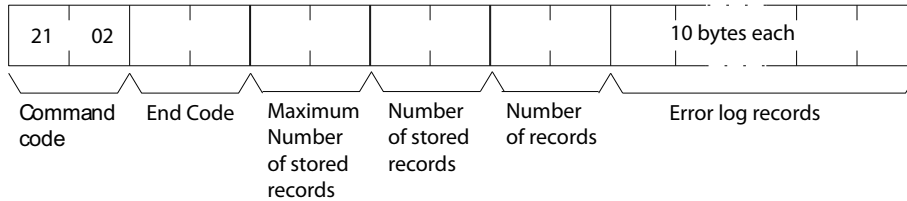


The following table defines the minimum and maximum values for each item in the command block with setting details.

Setting	Minimum Value (Hex)	Maximum Value (Hex)	Setting Details
Beginning Record Number	0000	0050	The first record to read
Number of Records	0000	0050	The number of records to read

Response Block

Issuing an Error Log Read command will result in the following response End Codes.

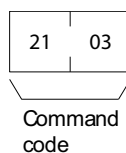


End Code (hex)	Description	Condition	Correction
0000	Normal Completion	---	
1001	Command length exceeds maximum command length	---	Too many parameters sent in the command. Correct the command and re-send.
1002	The command length is insufficient for the smallest command	---	Too few parameters sent in the command. Correct the command and re-send.
1103	Beginning Record Number is Out of Range	Requested beginning record number is invalid	Correct the beginning record number and resend.
110C	The Number of Read Records is 0	There is an error in one of the parameter settings	Check the command data and correct the parameters.

4-2-10 Error Log Clear (2103)

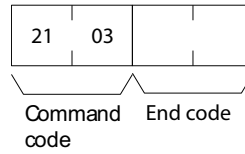
The Error Log Clear command clears the number of records stored in the User Defined CAN Unit Error Log.

Command Block



Response Block

Issuing an Error Log Clear command will result in the following response End Codes.



End Code (hex)	Description	Condition	Correction
0000	Normal Completion	---	
250F	Memory Writing Error.	Error Log was not cleared normally	---
260B	Cannot Clear Error Log	The error cause still exists	Correct the error and re-issue the command

5

Communications Timing

This section describes the overall specifications and the communication performance of the User Defined CAN Unit.

5-1 Performance	5-2
5-1-1 I/O Refresh Time	5-2
5-1-2 Output Message Evaluation Time	5-2
5-1-3 Input Message Processing Time	5-4
5-1-4 CAN Interface	5-5
5-1-5 I/O Response Time	5-5
5-1-6 Transmission of CAN Messages	5-8
5-1-7 Reception of CAN Messages	5-9

5-1 Performance

The overall performance of the User Defined CAN Unit depends on the performance of the host CPU interface and the performance of the CAN interface. In the next two sub-sections, these factors will be described.

5-1-1 I/O Refresh Time

The I/O refresh time of the User Defined CAN Unit depends on the size of the receive flags area, the size of the send trigger area, on the number of input buffers and on the number of output buffers that have been defined. Refer to Section 3, *Data Exchange with the CPU Unit* and Section 4, *Message Communications* for more information.

5-1-2 Output Message Evaluation Time

The output message evaluation time is the time between completion of the I/O refresh of the unit and the moment an output message is set ready for sending. The duration of this period is user-dependent and can introduce a significant delay before the message is actually sent. The message evaluation time depends on the length and structure of the user program and the send mode used for the User Defined CAN Unit. It is advised to review the timing, as the performance of the complete system (or application) is directly related to this timing. The evaluation time depends on the send mode used, and as different send modes can be combined, the total evaluation time is the sum of the individual send mode evaluation times.



Precautions for Correct Use

- Sending messages in the Triggered mode is the fastest way to send messages. It is faster than evaluation in the On Change mode and Cyclic mode.
- Sending messages in Cyclic mode is more than 4 times faster than when in On Change mode.

SM1, Triggered Mode

The evaluation time in Trigger mode differs with the state of the trigger (the trigger can be OFF or ON). When the rising edge of the trigger is detected, the message in the output buffer is sent.

Every time the trigger is detected and the message is sent, the evaluation time is 10 microseconds. The evaluation time is 9 microseconds every time the trigger is OFF.

The total number of output buffers is configured with the message command 2902. Message command 2903 and 2904 are used to associate a buffer with an identifier. A buffer with an identifier associated is called a configured buffer. The evaluation time for buffers that are not configured is 4.3 microseconds. The following examples are provided for calculation of the evaluation time.

Example 1

- Number of output buffers: 2
- Number of configured output buffers: 1

Every time the trigger of the configured message is ON, the evaluation time of all output buffers (being 2) will be: $10.0 + 4.3 = 14.3$ (microseconds).

The evaluation time is $9.0 + 4.3 = 13.3$ (microseconds) for the configured output buffer, in case the trigger is OFF.

Example 2

- Number of output buffers: 640
- Number of configured output buffers: 1

Every time the trigger configured message is ON, the evaluation time of all output buffers (being 640) will be: $10.0 + (4.3 * 639) = 2.76$ (milliseconds).

**Precautions for Correct Use**

To optimize the evaluation time for the triggered send mode, set the number of output buffers with message command 2902 to the same amount as the number of configured output buffers (the total number of buffers configured with message command 2903 or 2904).

SM2, On Change Mode

When using On Change mode, the evaluation time differs for output buffers depending on whether they contain information changes or not. In the case that data in the buffer has changed since the last evaluation, the evaluation time is 52 microseconds. If the output buffer has not changed, the evaluation time depends on the length of data in all the output buffers and is in the range of 15.4 - 47.0 microseconds.

The given times are for messages using the big endian format. Using little endian format will increase the evaluation time with approximately 13%.

The total number of output buffers is configured with message command 2902. Message command 2903 and 2904 are used to associate a buffer with an identifier. A buffer with an identifier associated is called a configured buffer. The evaluation time for buffers that are not configured is 4.3 microseconds.

Example 1

- Number of output buffers: 2
- Number of configured output buffers: 1

Every time the configured message is changed, the evaluation time of all output buffers (being 2) will be: $52.0 + 4.3 = 56.3$ (microseconds).

The evaluation time is $47.0 + 4.3 = 51.3$ (microseconds) for output buffers that are not changed since the last evaluation. The length of the configured output buffer is assumed to be 8 characters.

Example 2

- Number of output buffers: 640
- Number of configured output buffers: 1

Every time the configured message is changed, the evaluation time of all output buffers (being 640) will be: $52.0 + (4.3 * 639) = 2.8$ (milliseconds).

**Precautions for Correct Use**

The following guidelines apply to optimizing the evaluation time for the On Change Mode:

- Set the number of output buffers with message command 2902 to the same amount as the number of configured output buffers (that is the total number of buffers configured with message command 2903 or 2904).
- Use the big endian format in message commands 2903 and 2904 whenever possible because the little endian format will increase the evaluation time.

SM3, Cyclic Mode

In Cyclic mode the evaluation time differs for output buffers that should be sent and for which the cyclic time has not yet elapsed. Output buffers with elapsed cyclic time have an evaluation time of 11.6 microseconds. The evaluation time will be 10 microseconds as long as the cyclic time is not reached.

The total number of output buffers is configured with message command 2902, 2903 and 2904 are used to associate a buffer with an identifier. A buffer with an identifier associated is called a configured buffer. The evaluation time for buffers that are not configured is 4.3 microseconds.

Example 1

- Number of output buffers: 2
- Number of configured output buffers: 1

Every time the configured message is sent (cyclic time elapsed), the evaluation time of all output buffers (being 2) will be: $11.6 + 4.3 = 15.9$ (microseconds)

The evaluation time is $10.0 + 4.3 = 14.3$ (microseconds) in case the configured output buffer is not sent.

Example 2

- Number of output buffers: 640
- Number of configured output buffers: 1

Every time the configured message is sent (cyclic time elapsed), the evaluation time of all output buffers (being 640) will be: $11.6 + (4.3 * 639) = 2.76$ (milliseconds).



Precautions for Correct Use

The following guidelines apply to optimizing the evaluation time for the Cyclic Mode:

- Set the number of output buffers with message command 2902 to the same amount as the number of configured output buffers (that is the total number of buffers configured with message command 2903 or 2904).

5-1-3 Input Message Processing Time

The input message process time is the time between reception of a message and the processing of that message, i.e. data is ready to be refreshed. The process time is user dependant and can significantly slow the process of receiving CAN messages. If processing time become excessive, messages will be lost (and missed) due to insufficient buffer capacity.

It is strongly advised to configure the input and output buffers for CAN messages according to good practice, as this configuration directly influences the performance of the application.

● Processing Time

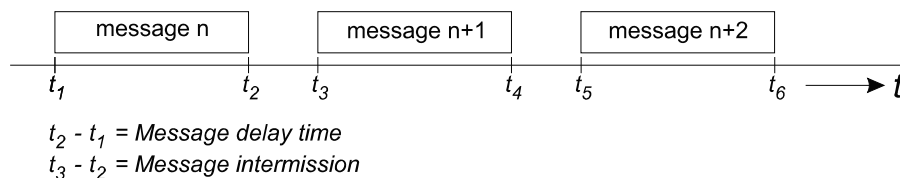
The process time of a received message is 65 microseconds only in the case that the message identifier is the one corresponding with input buffer 0. The User Defined CAN Unit compares the received identifier with the configured identifier for buffer0, next with buffer1, next with buffer 2 etc. The comparing stops if a match between the received and configured identifier is found. Therefore the process time will be 2.55 milliseconds if the received identifier matches with the configured identifier for buffer 639. If the received identifier is not found and a total of 640 input buffers is defined, the process time of the message will be 2.50 microseconds.

The following guidelines apply to optimizing the process time:

- Set the number of input buffers with message command 2902 to the same amount as the number of configured input buffers (that is the total number of buffers configured with message command 2905 or 2906).
- Configure the order of input buffers according to the frequency of received messages, i.e. in descending order. Buffer 0 is configured with the identifier of the most frequent received message. The least frequent message should be received in the buffer with the highest number.
- If the total number of input buffers is high and the unit will receive messages that are not configured (especially if they are very frequent), configure these message using same logic from the previous guideline.

5-1-4 CAN Interface

This section describes the performance of the physical layer of CAN and the performance of the CAN interface of the User Defined CAN Unit. The following figure depicts the transmission of CAN messages on the bus



● Message Delay Time

Every message has a certain message delay time which is mainly determined by the size of the data field in the message. The CAN bit-stuff mechanism (after every 5 consecutive equal value bits, a bit of the opposite polarity is added) can increase the message delay time by a maximum of 15%. The following formulas give the minimum and maximum message delay time.

Message Delay Time (seconds) = number of bits in message / bit rate (bit/s)

Message Delay Time_{min}(seconds) = 44 + (8 x number of data bytes) / bit rate (bit/s)

Message Delay Time_{max}(seconds) = trunc (47.8 + (9.6 x number of data bytes) / bit rate (bits/s))

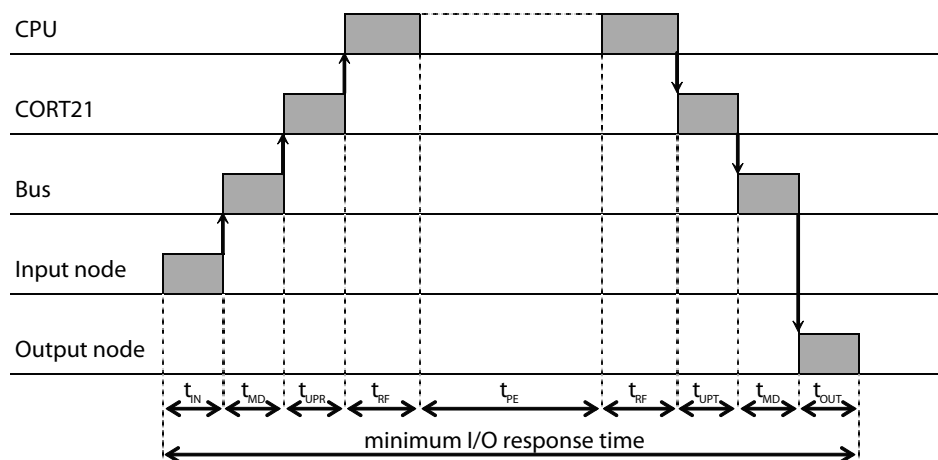
Note The maximum number of data bytes = 8

5-1-5 I/O Response Time

The I/O response time (CAN Bus response time) is the time between an input being set (or reset) on an Input Node and an output being set (or reset) on an Output Node, all under the condition that the Input Node and Output Node are linked to the User Defined CAN Unit.

The figures below show the minimum and maximum I/O response time. In the bottom figure, the processing of received messages is finished immediately after a CPU I/O refresh. The received application data cannot be transferred to the CPU until the next I/O refresh. The other factors that can influence the minimum and maximum I/O response times are explained below.

Minimum I/O Response Time



- | | |
|--|---|
| t_{IN} : Input Node ON (OFF) delay | t_{UPR} : Unit Processing time for Received messages |
| t_{OUT} : Output Node ON (OFF) delay | t_{UPT} : Unit Processing time for Transmitted messages |
| t_{MD} : Message Delay time | t_{PE} : Program Execution time |
| | t_{RF} : I/O Refresh time |

- **Message Delay Time**

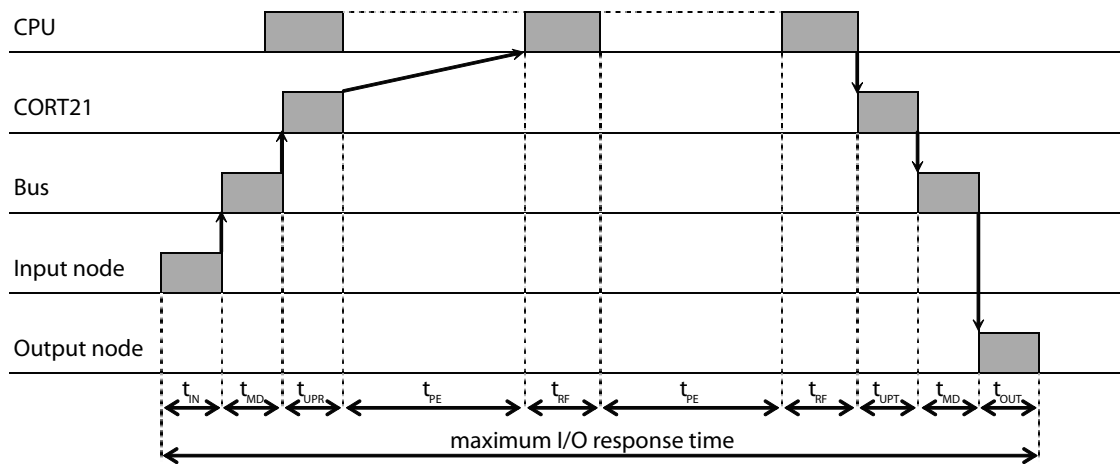
The message delay time is described in section 5-1-4 *CAN Interface*.

- **Unit Processing Time for Transmitted / Received Messages**

The unit processing time for received messages depends on the message rate of all the units connected to the user Defined CAN Unit (the rate of received messages by the unit).

If the message rate is higher than the unit is capable of processing (see 5-1-4 *CAN Interface*), the messages will queue, causing a delay in processing. If messages are received when the queue is full, queue overflow occurs (see section 6, *Troubleshooting and Maintenance*).

Maximum I/O Response Time



t_{IN} : Input Node ON (OFF) delay	t_{UPR} : Unit Processing time for Received messages
t_{OUT} : Output Node ON (OFF) delay	t_{UPT} : Unit Processing time for Transmitted messages
t_{MD} : Message Delay time	t_{PE} : Program Execution time
	t_{RF} : I/O Refresh time

The unit processing time of messages to be transmitted depends on the number of send triggers and the (send) mode of each output buffer. If many send triggers change at (or near) the same time, or if many output buffers are changed at the same time, or have the same time triggering, the transmission of output buffers will queue causing a delay in transmission. If the rate at which the output buffers change state requires a higher transmission rate than the unit is capable of (see 5-1-4 CAN Interface), a transmit queue overflow occurs.

Transmit queue overflow errors can be solved by reducing the rate at which the output buffers change state and therefore requests for transmission are generated.

Receive queue overflow errors can be solved by decreasing the CPU cycle time or by reducing the rate at which the other units transmit messages.

In case of high bus loads (over 50%), additional delays can be caused by high-priority (low identifier) messages which delay the transmission of lower priority messages.



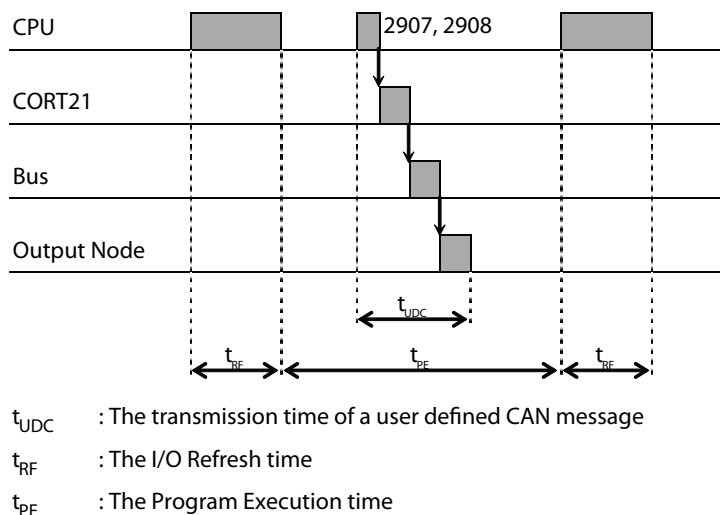
Additional Information

The CPU Unit's task period is the sum of the User Program execution time plus the User Defined CAN Unit's refresh time on the CPU Unit.

5-1-6 Transmission of CAN Messages

Transmitting With Direct Transmit Mode

The following figure shows the sequence of direct transmitting a CAN message using message command 2907 or 2908.

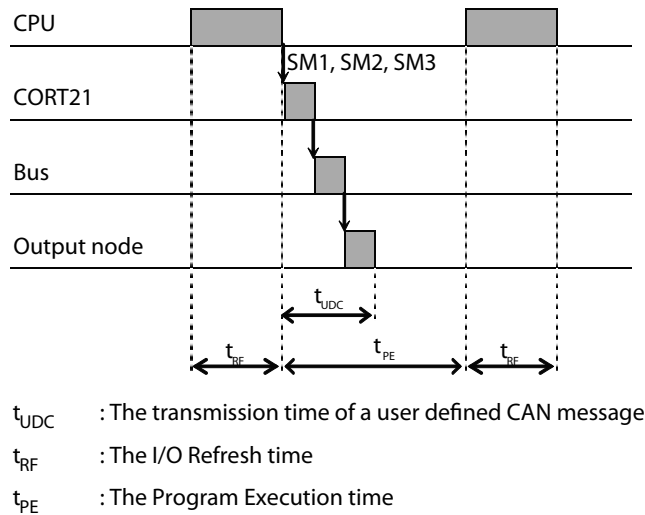


After the content of the message command is transferred to the User Defined CAN Unit, the unit immediately relays the contained message to the CAN transmission interface. The actual transmission delay of the message depends on the bus load and the message priority (identifier). Refer to 5-1-4 *CAN Interface* for the message delay time on the bus. The time required for an output to be set or a value in the Output Node to be changed depends on the characteristics of the Output node.

The minimum time between the execution of the message command in the user program and the message being transmitted on the bus is 0.5 milliseconds. Depending on the configuration, (message command 2902) this time can be sufficient longer and up to a maximum of 20 milliseconds.

Transmitting With Send Mode

Sending messages with SM1, SM2 or SM3 differs from direct sending CAN messages. There is no delay between the start of 'program execution' and passing the message to the User Defined CAN Unit.



5-1-7 Reception of CAN Messages

The User Defined CAN Unit can store up to 15 received messages in its internal buffer, the receive queue. These messages in the receive queue are filtered (according to the configuration of the input buffers) and transferred to the CPU. It may take several CPU cycles to filter and transfer a CAN message to the CPU.

The performance of the unit depends on the number of input and output buffers. If several input and/or output buffers are used, the unit will require more time for processing. Limiting the number of input and/or output buffers can increase performance.

The CPU cycle time also influences the performance. If the CPU cycle time is long, the receive queue in the unit may overflow. Refer to the *NJ-series CPU Unit Software User's Manual* (Cat. No. W501) for more info.

6

Troubleshooting and Maintenance

This section describes error processing, periodic maintenance operations and troubleshooting procedures needed to keep the User Defined CAN Unit operating properly. It is recommended to read through the error processing procedures before operation so that operating errors can be identified and corrected quickly.

6-1 Overview	6-2
6-1-1 Troubleshooting the User Defined CAN Unit	6-2
6-2 Troubleshooting with the User Defined CAN Unit Indicators	6-4
6-2-1 Run LED Indicator	6-4
6-2-2 ERR LED Indicator	6-5
6-2-3 Two 7-segment Display	6-5
6-2-4 Two Dot Indicators	6-6
6-3 Error Log Functions	6-7
6-3-1 Error Log Table	6-7
6-3-2 Error Codes and Detail Codes	6-8
6-3-3 Status Information	6-8
6-4 Troubleshooting	6-10
6-4-1 CPU Unit's ERR/ALM Indicator Lit or Flashing	6-10
6-5 Event Logs	6-11
6-5-1 Overview of the Event Logs	6-11
6-5-2 Error Table	6-12
6-5-3 Error Descriptions	6-12
6-6 Maintenance and Replacement	6-16
6-6-1 Cleaning	6-16
6-6-2 Inspection	6-16
6-6-3 Replacing Faulty Units	6-17

6-1 Overview

6-1-1 Troubleshooting the User Defined CAN Unit

The User Defined CAN Unit uses several error detection and error handling mechanisms.

User Defined CAN is based on the serial bus protocol of CAN. The data link layer of the CAN protocol combines 5 error detection mechanisms (CRC check, frame check, Ack check, bit check, bit stuffing check). This combination results in a Hamming distance of 6. This means that at least 6 bits in the message frame must have been disturbed to possibly remain undetected. The overall residual error probability is extremely low and this makes CAN-based protocols very reliable and suitable for harsh environments.

The User Defined CAN Unit also has manufacturer specific status indication mechanisms

Status

Device Variable `*_StaComm` indicates the current state of the User Defined CAN Unit communications.

LED Indicators, 7-segment Display

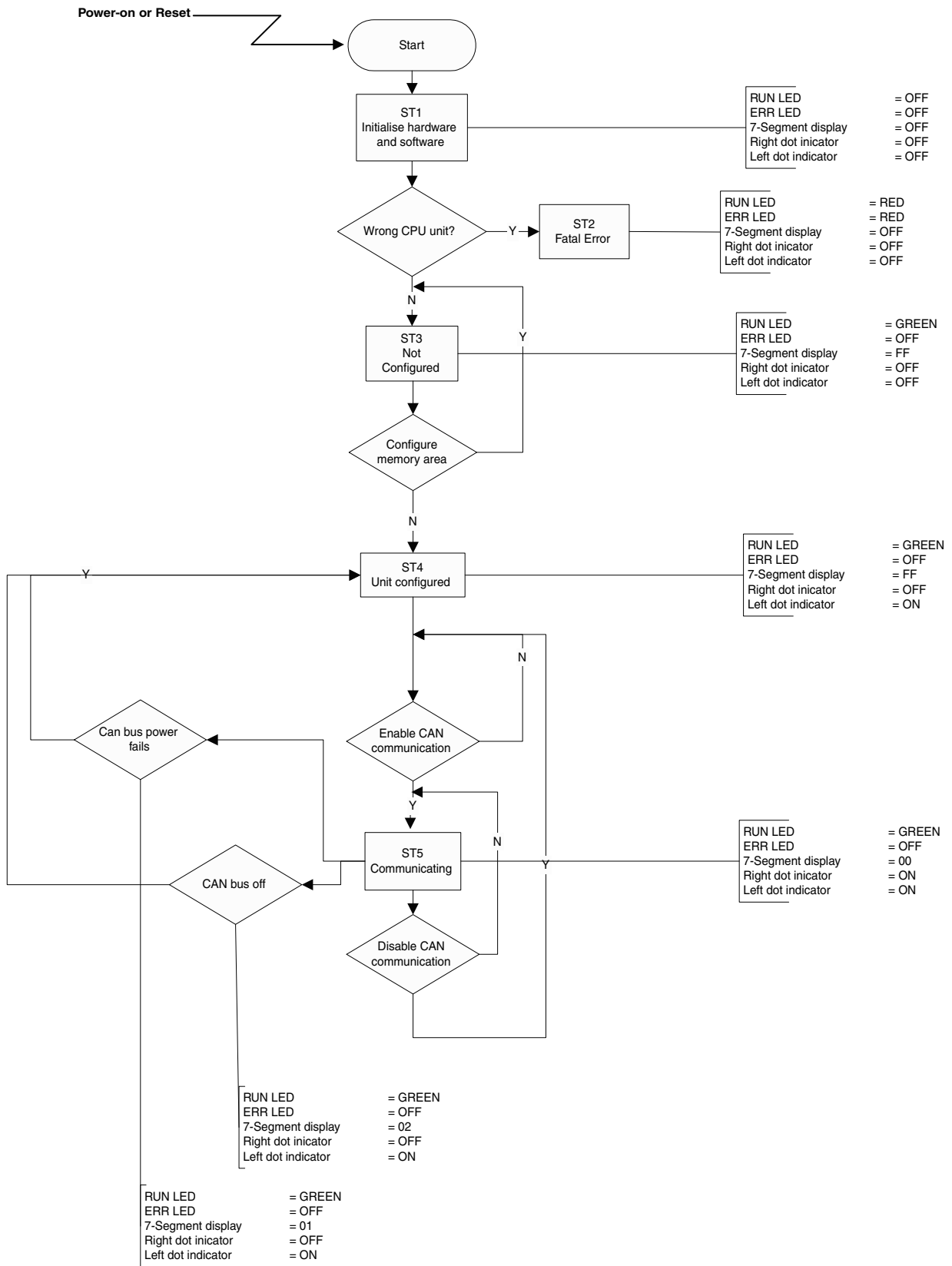
Visual status indication of the User Defined CAN Unit.

Error Log

Some errors are logged in the error log. These errors can be read after a restart.

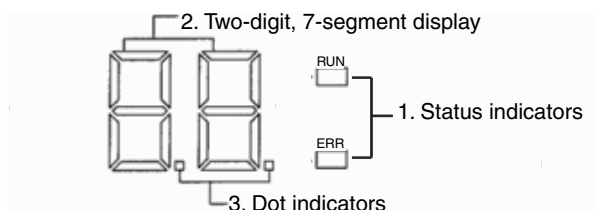
Startup Sequence

In the following diagram the startup sequence is shown, including the status of all indicators of the unit.



6-2 Troubleshooting with the User Defined CAN Unit Indicators

The User Defined CAN Unit has several indicators on the front of the Unit. Two 7-segment displays with dots and two error LED indicators.



6-2-1 Run LED Indicator

Indicator Status		Network/Unit Status	Comments
Color	Status		
Green	OFF	No power or in ST1 or ST2 state	Make sure that the unit is wired and mounted correctly, switch the power on or wait for until unit to be initialized.
	ON	State is ST3, ST4 or ST5	Configure the unit with message command 2902 or enable communication by turning ON <i>*_EnbICANComm</i> . Enabled communication is confirmed when <i>*_EnbIComm</i> is ON or the unit is waiting for CAN messages.
Red	OFF	No power or in ST1 or ST2	Make sure that the unit is wired and mounted correctly, switch the power on or wait for the until unit to be initialized.
	Flashing	ST1 (PC21 startup error) or ST2 (hardware error)	Verify proper unit number setting. If problem persists, contact an Omron representative.
	ON	Fatal error in unit	Restart Unit. If problem persists, contact and Omron representative.

6-2-2 ERR LED Indicator

Indicator Status		Unit Status	Comments
Color	Status		
Red	OFF	No error active	The unit has no active error and diagnostic functions do not detect any error condition.
	ON	One of more of the following errors are active: <ul style="list-style-type: none"> • Hardware error • No CAN configuration • CAN network power fail, Unit is Bus off • Fatal error in Unit • CPU Watchdog Time-out • PC21 bus error • Cyclic refresh time-out • I/O table error 	<ul style="list-style-type: none"> • Make sure that the unit is wired and mounted correctly, switch the power on or wait until the unit is initialized. • Configure the Unit. • Correct the CAN network, restore the power on the network. • Create the CPU I/O table. • Make sure the Cyclic refresh is enabled • Restart the Unit If problem persists, contact a local Omron representative.

6-2-3 Two 7-segment Display

Two 7-segment Display			
Color	Status	Unit Status	Comments
Red	All Off	No power or in ST1 or ST2	Make sure that the unit is wired and mounted correctly, switch the power on or wait for unit to initialize. Configure the User Defined CAN Unit with message commands.
	Displays FF	State is ST3 or ST4	If unit is not configured, then configure the unit with message command 2902. When unit already has a configuration, enable communication by turning ON <code>*_EnbICANComm</code> . This enabling is confirmed when <code>*_EnbComm</code> is set.
	Displays 00	State is ST5	CAN messages can be sent and received
	Flashes 01	CAN Bus power fail	Restore power on the CAN bus
	Flashes 02	CAN Bus off	Check the CAN network
	Displays H1	ST1 (PC21 Startup error)	Create I/O table or change the unit number.
	Displays EC	ST1 (PC21 Startup error)	Create I/O table or change the unit number.
	Flashes EC	ER2	Create I/O table, change the unit number.

6-2-4 Two Dot Indicators

Indicator	Color	Status	Unit Status	Comments
Left Dot	Red	ON	Unit is configured, and communication is enabled or disabled, state is ST4 or ST5	Enable communication by turning ON <i>*_EnbICANComm</i> . This is confirmed when <i>*_EnbIComm</i> is ON or communication is enabled. CAN message can be sent and received
		Flashing	Unit state is ST2, Initialization error.	Make sure that the unit is wired and mounted correctly, switch the power on or wait for unit to initialize. Configure the unit with message command 2902
		OFF	Unit state is ST1, ST2 or ST3, unit has a startup error or is not (yet) configured	Make sure that the unit is wired and mounted correctly, switch the power on or wait for unit to initialize. Configure the unit with message command 2902
Right Dot		ON	Unit state is ST5	Communication is enabled
		OFF	Communication is disabled, or unit is in state ST1, ST2 or ST3 or ST4	Enable communication by turning ON <i>*_EnbICANComm</i> . This is confirmed when <i>*_EnbIComm</i> is ON. Make sure that the unit is wired and mounted correctly, switch the power on or wait for until the unit has initialized. Configure the unit with message command 2902.

6-3 Error Log Functions

Errors detected by the User Defined CAN Unit are stored in the error log along with the date and time of their occurrence. The error log can be read, cleared, and monitored using message commands.

6-3-1 Error Log Table

Each time an error occurs, one error record is recorded in the User Defined CAN Unit's RAM error log table. The table can record up to 15 records. If another error occurs when the table is full, the oldest record will be erased to make room for the new error record.

The error log table records the following information.

- Error code
- Detail code
- Time of occurrence (The CPU Unit's time is used for the time stamp.)

● Error Log Storage Area

When an error is detected, information on the error and the time stamp are stored in the Unit's internal RAM as an error log record. Serious errors are recorded in EEPROM as well as RAM*. The error log records in EEPROM are retained even when the Unit's power is turned OFF or the Unit is restarted. The error log records in EEPROM are copied to RAM when the User Defined CAN Unit is powered-up.

When the error log is read with a message command, only the error log records in RAM are read. When the error log is cleared with a message command, the error log records in RAM and EEPROM are erased.

Refer to 6-3-2 *Error Codes and Detail Codes* for a table listing the error codes.

● Reading and Clearing the Error Log

The error log table can be read or cleared by sending a message command to the User Defined CAN Unit. Use the User Defined CAN Unit's unit address as the message command's destination unit address. (The unit address is the unit number +10 Hex.).

Message command 2102 is used to read the error history. Issuing the message command 2103 will clear the error log. Refer to 4-2 *Unit Configuration and Control* for details on reading and clearing the error log with message commands.



Additional Information

The CPU Unit's time information is used for the time stamps in the User Defined CAN Unit's error log records. If the time information cannot be read from the CPU Unit, the time stamp will contain all zeros.

Moreover, if the battery is replaced in an NJ-series controller, the time of the CPU Unit's built-in clock must be set again the next time that power is turned ON. If the built-in clock time is not set, the correct time information will not be recorded. If this error log is read from the CPU Unit, the time information will not be consistent.

6-3-2 Error Codes and Detail Codes

Error code (Hex)	ID	Error	Detail code		Record stored in EEPROM
			First byte	Second byte	
0001	ER1	CPU Unit watchdog timer error	00 Hex	00 Hex	Yes
0002	ER2	CPU Unit service monitoring error (Servicing from the CPU Unit is not performed at fixed intervals.)	Monitoring time (ms)		Yes
000E	ER3	PC21 Bus Error	00 Hex	00 Hex	Yes
0601	ER5	CPU Bus Unit Error	Undefined Contents		Yes
0602	ER5	Error Log Read Error	01 Hex	06 Hex	Yes
	ER6	Error log write error	02 Hex	06 Hex	No
	ER7	Network parameter read error	01 Hex	02 Hex	Yes
	ER8	Network parameter write error	02 Hex	02 Hex	No
0201	ER8	Network parameter file lost	00 Hex	00 Hex	No
0340	ER10	Network power fail	00 Hex	02 Hex	Yes

Note The error information is not written to EEPROM when a memory error occurs in the error log area (EEPROM).

6-3-3 Status Information

The Unit status can be read by analyzing the status words. In the next table there is an overview of all information which can be analyzed. The Unit uses only device variables **_StaComm*, **_DelayMsgNo* and **_ProcMsgNo* for status information

Device Variable <i>*_StaComm</i> (Word)				
Device Variable	Name	Status	Controlled by	Unit Operation
<i>*_MsgRcv</i> (Bool)	CAN Message Received	OFF	Unit	New configured CAN message received since last cyclic refresh
		ON	Unit	No new configured CAN message received since last cyclic refresh
<i>*_SendOver</i> (Bool)	Send Overflow	OFF	Unit	All messages to be sent fit in the send queue
		ON	Unit	Send queue overflow, some messages will be delayed
<i>*_RcvOver</i> (Bool)	Receive Overflow	OFF	Unit	Receive queue overflow, some received messages have been discarded
		ON	Unit	All messages received were handled
<i>*_NetPwrErr</i> (Bool)	Network Power Failure	OFF	Unit	Network power OK
		ON	Unit	Error ER10 active (Power fail when communicating)

Device Variable *_StaComm (Word)				
Device Variable	Name	Status	Controlled by	Unit Operation
*_BusoffErr (Bool)	Bus Off Event	OFF	Unit	Bus off event did not happen since last event EV3 (enable communications)
		ON	Unit	Bus off event EV5 (bus off) has been generated
*_ErrInErrLog (Bool)	Error in Error Log	OFF	Unit	No new errors in error log since: <ul style="list-style-type: none"> • Startup • Last service of message command 2102 • Last service of message command 2103
		ON	Unit	New errors in error log since: <ul style="list-style-type: none"> • Startup • Last service of message command 2102 • Last service of message command 2103

Device Variable *_DelayMsgNo (Word)				
Device Variable	Name	Status	Controlled by	Unit Operation
*_DelayMsgNo (Word)	Number of Delayed Messages	BCD Coded	Unit	Number of messages that are delayed, see send overflow device variable (*_SendOver) Range 0 to 15 (BCD)

Device Variable *_ProcMsgNo (Word)				
Device Variable	Name	Status	Controlled by	Unit Operation
*_ProcMsgNo (Word)	Number of Messages to be Processed	BCD Coded	Unit	Number of input messages, see receive overflow device variable (*_RcvOver). Range 0 to 15 (BCD)

6-4 Troubleshooting

6-4-1 CPU Unit's ERR/ALM Indicator Lit or Flashing

Error	Probable cause
An I/O setting check error occurred.	<ul style="list-style-type: none"> • Make sure that the Unit is mounted properly. • The CPU Bus Unit model registered in the Unit Configuration in the CPU Unit does not match the actual Unit Configuration. Compare using "Synchronize" operation and use one of the following procedures. <ul style="list-style-type: none"> • Correct the Unit number setting • Correct the project Unit Configuration and transfer to the CPU Unit
Special Unit access is denied.	<ul style="list-style-type: none"> • Make sure that the Unit is mounted properly. • Restart the Unit. If operation is not restored even after the Unit is restarted, replace the Unit.
An I/O Bus check error occurred.	<ul style="list-style-type: none"> • Make sure that the Unit is mounted properly. • Restart the Unit. If operation is not restored even after the Unit is restarted, replace the Unit.

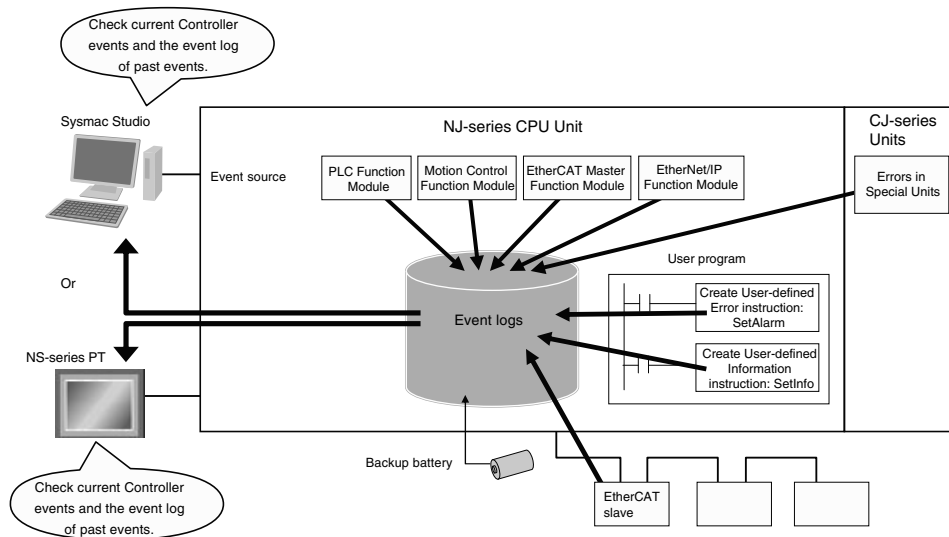
Refer to the *NJ-Series CPU Unit Hardware User's Manual* (Cat. No. W500) and *NJ-Series CPU Unit Software User's Manual* (Cat. No. W501) for more details on these errors.

6-5 Event Logs

6-5-1 Overview of the Event Logs

You use the same methods to manage all of the events that occur on the NJ-series Controller. (The events include errors and information.) You can use the Sysmac Studio or an NS-series PT to confirm current Controller events and the logs of events that have occurred. These logs are called event logs. Controller errors that occur for this Unit are also reported as events in the NJ-series CPU Unit.

Refer to the *NJ-series CPU Unit Software User's Manual* (Cat. No. W501) for details on the event logs in an NJ-series CPU Unit. Refer to the *NJ-series Troubleshooting Manual* (Cat. No. W503) for details on Controller errors, confirmation methods, and corrections.



6-5-2 Error Table

The errors that may occur for this Unit are listed below. Event levels are given in the table as follows:

Maj: Major fault level

Prt: Partial fault level

Min: Minor fault level

Obs: Observation

Info: Information

Refer to the *NJ-series Troubleshooting Manual* (Cat. No. W503) for all of the event codes that may occur in an NJ-series Controller.

Event code	Event name	Meaning	Assumed cause	Level					Reference
				Maj	Prt	Min	Obs	Info	
04C60000 hex	Network Power Error	CAN network power is insufficient	<ul style="list-style-type: none"> Communications power is not being supplied sufficiently from the network 			√			page 6-14
84C20000 hex	Send Queue Overflow	The send queue was overflowed	<ul style="list-style-type: none"> If the message rate exceeds the processing capacity of the unit, a delay occurs. The send queue was full already when sending the message. 				√		page 6-14
84C30000 hex	Receive Queue Overflow	The received queue was overflowed	<ul style="list-style-type: none"> If the message rate exceeds the processing capacity of the unit, a delay occurs. The receive queue was full already when receiving the message. 				√		page 6-15
84C40000 hex	Bus Off Event	Bus off event EV5 (bus off) has occurred	<ul style="list-style-type: none"> A buss off condition was detected 			√			page 6-15

6-5-3 Error Descriptions

This section describes the information that is given for individual errors.

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
Error attributes	Level	Tells the level of influence on control.*1	Recovery	Gives the recovery method.*2	Log category

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 the Sysmac Studio or an NS-series PT.			
Precautions/Remarks	Provides precautions, restrictions, and supplemental information.			

*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.

Error Descriptions

Event name	Network Power Error			Event code	04C60000 hex	
Meaning	CAN network power is insufficient.					
Source	PLC function module		Source details	CJ-series Unit	Detection timing	During CAN communications
Error attributes	Level	Minor Fault	Recovery	Cycle the power supply	Log category	System
Effects	User program	Continues	Operation	CAN communications are stopped		
System-defined variables	Variable		Data type		Name	
	None		---		---	
Cause and correction	Assumed cause		Correction		Prevention	
	Communications power is not being supplied sufficiently		Check the wiring of the network power and the network cable and correct the mistake		Check that the network power and the network cable are wired correctly	
Attached information	None					
Precautions/Remarks	None					

Event name	Send Queue Overflow			Event code	84C20000 hex	
Meaning	The send queue was overflowed					
Source	PLC function module		Source details	CJ-series Unit	Detection timing	During CAN communications
Error attributes	Level	Observation	Recovery	All messages to be sent fit in the send queue.	Log category	System
Effects	User program	Continues	Operation	Some messages will be discarded		
System-defined variables	Variable		Data type		Name	
	None		---		---	
Cause and correction	Assumed cause		Correction		Prevention	
	If the processing capacity of the unit exceeds the message rate, messages are discarded when the send queue is full (overflowed). The send queue was full already when sending the message.		Reduce the communications load.		Consider the communications load and the rate the output buffers change state.	
Attached information	None					
Precautions/Remarks	None					

Event name	Receive Queue Overflow			Event code	84C30000 hex	
Meaning	The receive queue was overflowed					
Source	PLC function module		Source details	CJ-series Unit	Detection timing	During CAN communications
Error attributes	Level	Observation	Recovery	All messages received were handled.	Log category	System
Effects	User program	Continues	Operation	Some messages will be discarded		
System-defined variables	Variable		Data type		Name	
	None		---		---	
Cause and correction	Assumed cause		Correction		Prevention	
	If the message rate exceeds the processing capacity of the unit, messages are discarded when the receive queue is full (overflowed). The receive queue was full already when receiving the message.		Reduce the communications load.		Consider the communications load (the rate the other devices transmit messages) and the CPU cycle time.	
Attached information	None					
Precautions/Remarks	None					

Event name	Bus Off Event			Event code	84C40000 hex	
Meaning	Bus off event EV5 (bus off) has occurred.					
Source	PLC function module		Source details	CJ-series Unit	Detection timing	During CAN communications
Error attributes	Level	Minor Fault	Recovery	Disable and enable the communication (EV3) with the *_EnbCANCom device variable.	Log category	System
Effects	User program	Continues	Operation	None		
System-defined variables	Variable		Data type		Name	
	None		---		---	
Cause and correction	Assumed cause		Correction		Prevention	
	A bus off condition was detected.		Check the following: <ul style="list-style-type: none"> • Matching device baud rates • Proper cable lengths • Broken or loose cables • Installation of terminators at both ends of the lines • Excessive noise 		Implement the preventative measures for each error.	
Attached information	None					
Precautions/Remarks	None					

6-6 Maintenance and Replacement

This section describes the routine cleaning and inspection recommended as regular maintenance as well as the Unit replacement procedure.

6-6-1 Cleaning

Clean the User Defined CAN Units regularly as described below in order to keep the network in its optimal operating condition.

- Wipe the Unit daily with a dry, soft cloth.
- When a spot can't be removed with a dry cloth, dampen the cloth with a neutral cleanser (2% solution), wring out the cloth, and wipe the Unit.
- A smudge may remain on the Unit from gum, vinyl, or tape that was left on for a long time. Remove the smudge when cleaning.



Precautions for Correct Use

Never use volatile solvents such as paint thinner, benzene, or chemical wipes. These substances could damage the surface of the Unit.

6-6-2 Inspection

Be sure to inspect the system periodically to keep it in its optimal operating condition. In general, inspect the system once every 6 to 12 months, but inspect more frequently if the system is used with high temperature or humidity or under dirty/dusty conditions.

Inspection Equipment

Prepare the following equipment before inspecting the system.

● **Required Equipment**

Have a standard and Phillips-head screwdriver, multimeter, alcohol, and a clean cloth.

● **Equipment Required Occasionally**

Depending on the system conditions, a synchroscope, oscilloscope, thermometer, or hygrometer (to measure humidity) might be needed.

Inspection Procedure

Check the items in the following table and correct any items that are below standard.

	Item	Standard	Equipment
Environmental conditions	Ambient and cabinet temperature	0°C to 55°C	Thermometer
	Ambient and cabinet humidity	10% to 90% (with no condensation or icing)	Hygrometer
	Dust/dirt accumulation	None	Check visually
Installation	Are the Units installed securely?	No looseness	Phillips head screwdriver
	Are the communications connectors fully inserted?	No looseness	Phillips head screwdriver
	Are the external wiring screws tight?	No looseness	Phillips head screwdriver
	Are the connecting cables undamaged?	No damage	Check visually

6-6-3 Replacing Faulty Units

Replace a faulty User Defined CAN Unit as soon as possible. We recommend having spare Units available to restore network operation as quickly as possible.



Precautions for Correct Use

Observe the following precautions when replacing a faulty Unit.

- After replacement make sure that there are no errors with the new Unit.
- When a Unit is being returned for repair, attach a sheet of paper detailing the problem and return the Unit to your OMRON dealer.
- If there is a faulty contact, try wiping the contact with a clean, lint-free cloth dampened with alcohol.



Precautions for Safe Use

To prevent electric shock when replacing a Unit, be sure to stop communications in the network and turn OFF the power supplies to all of the nodes (master and slaves) before removing the faulty Unit.



Appendices

A-1 Differences in Available Functions Depending on the CPU Unit (NJ/CJ-series) to be Connected	A-2
A-1-1 Differences in Available Functions	A-2
A-1-2 Differences in Accessing from User Program	A-2
A-2 User Program Example	A-4

A-1 Differences in Available Functions Depending on the CPU Unit (NJ/CJ-series) to be Connected

A-1-1 Differences in Available Functions

Some functions available to the CJ series may be unavailable when you operate this Unit with the NJ series.

Note This manual describes the information you must know when using CJ1W-CORT21 by connecting it to NJ-series CPU Unit. You can use FINS message communications with NJ-series, but not all areas of the NJ-series CPU Unit can be accessible. If you need to use these functions, such as to connect to existing equipment, please consult with your OMRON representative.

A-1-2 Differences in Accessing from User Program

When this Unit is operated with an NJ-series device, a user program accesses various functions provided by the User Defined CAN Unit through device variables for CJ-series Unit that specifies AT specification for the memory used for CJ-series Unit.

The device variables for CJ-series Unit in the NJ-series CPU Unit's memory for CJ series Unit that correspond to the addresses and bit positions in the CJ-series CPU Unit's I/O memory are listed below.

First word in Special I/O Unit CIO Area: $n = \text{CIO } 1,500 + \text{Unit number} \times 25$ (Unit number: 0 to 15)

First word in Special I/O Unit DM Area: $m = \text{D}30,000 + \text{Unit number} \times 100$ (Unit number: 0 to 15)

CPU Bus Unit Words Allocated in CIO Area

- **CIO n (Enable CAN Communications)**

The device variable for CJ-series Unit that corresponds to all bits of a word starting with CIO n is as follows:

CJ-series I/O memory address		NJ-series device variables for CJ-series Unit	
Word address	Bit	Variable name	Description
CIO n	04	*_EnbICANComm	Enable CAN Communications (The functions of bit 4 of a word starting with CIO n correspond to that of bit 4 of this device variable for CJ-series Unit.)

- **CIO n+3 (Status Communication)**

The device variable for CJ-series Unit that corresponds to all bits of a word starting with CIO n+3 is as follows:

CJ-series I/O memory address		NJ-series device variables for CJ-series Unit	
Word address	Bit	Variable name	Description
CIO n+3	0 to 15	*_StaComm	Status Communication (The functions of bits 0 to 15 of a word starting with CIO n+3 correspond to those of bits 0 to 15 of this device variable for CJ-series Unit.)

The device variables for CJ-series Units that correspond to bits 0 to 15 of a word starting with CIO n+3 are as follows:

CJ-series I/O memory address		NJ-series device variables for CJ-series Unit	
Word address	Bit	Variable name	Description
CIO n+3	0 to 1	---	Reserved by System
	2	*_EnblComm	Enabled Communication
	3	*_MsgRcv	CAN Message Received
	4 to 5	---	Reserved by System
	6	*_SendOver	Send Queue Overflow
	7	*_RcvOver	Receive Queue Overflow
	8	---	Reserved by System
	9	*_NetPwrErr	Network Power-Failure
	10	*_BusoffErr	Bus Off Event
	11 to 14	---	Reserved by System
	15	*_ErrInErrLog	Error in Error Log

● **CIO n+8 (Number of Delayed Messages)**

The device variable for CJ-series Unit that corresponds to all bits of a word starting with CIO n+8 is as follows:

CJ-series I/O memory address		NJ-series device variables for CJ-series Unit	
Word address	Bit	Variable name	Description
CIO n+8	0 to 15	*_DelayMsgNo	Number of Delayed Messages (Corresponding to the word CIO n+8 of this device variable for CJ-series Unit.)

● **CIO n+9 (Number of Messages to be Processed)**

The device variable for CJ-series Unit that corresponds to all bits of a word starting with CIO n+9 is as follows:

CJ-series I/O memory address		NJ-series device variables for CJ-series Unit	
Word address	Bit	Variable name	Description
CIO n+9	0 to 15	*_ProcMsgNo	Number of Messages to be Processed (Corresponding to the work CIO n+9 of this device variable for CJ-series Unit.)

A-2 User Program Example

The following example uses the message command 2902 (Configure Memory Areas) to configure the User Defined CAN Unit for correct operation. This user program example is created with the Structured Text language using the Sysmac Studio ST Editor. See the *Sysmac Studio Operation Manual* (Cat. No. W504) for more details about program creation using the ST Editor.

The example is created based on the following configuration requirements (specific application requirements may vary).

- Send buffer location of CIO 100 to 104.
- Send trigger location of W100.
- Receive buffer location of E0_0 to E0_4.
- Receive flag location of W110.
- Number of send messages of 1.
- Number of receive messages of 1.

Internal Variables

The following Internal Variables are used in the user program example.

Internals	Name	Data Type	Initial Value	AT	Retain	Constant	Comment
Externals	SEND_CMD	SendCmd			<input type="checkbox"/>	<input type="checkbox"/>	
	InExecute	BOOL	False		<input type="checkbox"/>	<input type="checkbox"/>	
	InDNetAdr	_sDNET_ADR			<input type="checkbox"/>	<input type="checkbox"/>	
	InOption	_sRESPONSE			<input type="checkbox"/>	<input type="checkbox"/>	
	InCmdDat	ARRAY[0..22] OF B...			<input type="checkbox"/>	<input type="checkbox"/>	Command def...
	Comm	_ePORT	_NONE		<input type="checkbox"/>	<input type="checkbox"/>	
	OutResDat	ARRAY[0..255] OF...			<input type="checkbox"/>	<input type="checkbox"/>	
	OutDone	BOOL	False		<input type="checkbox"/>	<input type="checkbox"/>	
	OutBusy	BOOL	False		<input type="checkbox"/>	<input type="checkbox"/>	
	OutError	BOOL	False		<input type="checkbox"/>	<input type="checkbox"/>	
	OutErrorID	WORD	16#0		<input type="checkbox"/>	<input type="checkbox"/>	
	OutErrorIDex	DWORD	16#0		<input type="checkbox"/>	<input type="checkbox"/>	
	Cnt_1	ULINT	0		<input type="checkbox"/>	<input type="checkbox"/>	
	SendingFlag	BOOL	False		<input type="checkbox"/>	<input type="checkbox"/>	

External Variables

The following External Variables are used in the user program example.

Internals	Name	Data Type	Constant	Comment
Externals	gStartFlag	BOOL	<input type="checkbox"/>	
	gErrorID	WORD	<input type="checkbox"/>	
	gErrorIDex	DWORD	<input type="checkbox"/>	
	gErrorNo	ErrorNo	<input type="checkbox"/>	
	gArrFINSresp	ARRAY[0..255] OF...	<input type="checkbox"/>	
	_Port_isAvailable	BOOL	<input checked="" type="checkbox"/>	
	gStrFINSresp	STRING[256]	<input type="checkbox"/>	

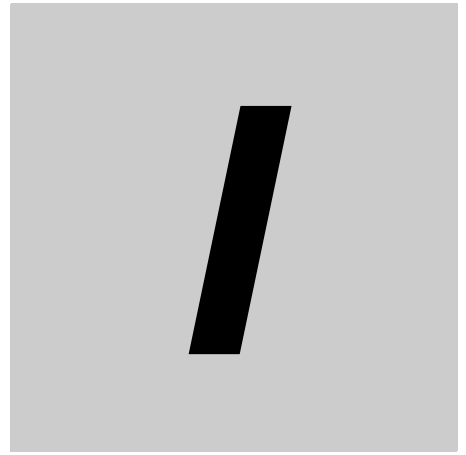
ST Program

The following ST Program example configures the User Defined CAN Unit based on the provided details.

```

1 //SendCmd命令
2
3
4 IF (gStartFlag = TRUE) AND (Port.isAvailable = TRUE) THEN
5   FOR Cnt_1 := ULINT#0 TO ULINT#2 BY ULINT#1 DO
6
7     IF Cnt_1 = 1 THEN
8       (****Parameter Set****)
9       OutDone:=FALSE;
10
11       InExecute := TRUE;
12       InDNetAdr.NetNo := 0;
13       InDNetAdr.NodeNo := 0;
14       //Cort21 unit number 0
15       InDNetAdr.UnitNo := BYTE#16#10;
16
17       InOption.isNonResp := FALSE;
18       InOption.TimeOut := 200;
19       InOption.Retry := 0;
20       // Setting memory areas   FINS 2902:
21       InCmdDat[0] := BYTE#16#29;
22       InCmdDat[1] := BYTE#16#02;
23       // Send Buffer C10 100 - C10 104
24       InCmdDat[2] := BYTE#16#00;
25       InCmdDat[3] := BYTE#16#01;
26       InCmdDat[4] := BYTE#16#00;
27       InCmdDat[5] := BYTE#16#64;
28       // Send Trigger W100
29       InCmdDat[6] := BYTE#16#00;
30       InCmdDat[7] := BYTE#16#03;
31       InCmdDat[8] := BYTE#16#00;
32       InCmdDat[9] := BYTE#16#64;
33       // 1 Message Send
34       InCmdDat[10] := BYTE#16#00;
35       InCmdDat[11] := BYTE#16#01;
36
37       // Receive Buffer E0 0 - E0 4
38       InCmdDat[12] := BYTE#16#00;
39       InCmdDat[13] := BYTE#16#08;
40       InCmdDat[14] := BYTE#16#00;
41       InCmdDat[15] := BYTE#16#00;
42       // Receive trigger W110
43       InCmdDat[16] := BYTE#16#00;
44       InCmdDat[17] := BYTE#16#03;
45       InCmdDat[18] := BYTE#16#00;
46       InCmdDat[19] := BYTE#16#6E;
47       // 1 Message Received
48       InCmdDat[20] := BYTE#16#00;
49       InCmdDat[21] := BYTE#16#01;
50
51       SendingFlag := TRUE;
52     END_IF;
53
54     SEND_CMD(
55       Execute      := InExecute,
56       DstNetAdr    := InDNetAdr,
57       CommPort    := Comm,
58       CmdDat      := InCmdDat[0], // command array assignment
59       CmdSize     := 22, // configure command size, in bytes
60       ResDat      := OutResDat[0], // response array assignment
61       Option      := InOption,
62       Done        => OutDone,
63       Busy        => OutBusy,
64       Error       => OutError,
65       ErrorID     => OutErrorID,
66       ErrorIDEx  => OutErrorIDEx
67     );
68
69   END_FOR;
70
71   IF (OutError=TRUE) THEN
72     gErrorID := OutErrorID;
73     gErrorIDEx := OutErrorIDEx;
74   END_IF;
75
76   IF (OutDone=TRUE) THEN
77     IF (OutBusy=TRUE) THEN
78       gErrorID := OutErrorID;
79     ELSIF (OutError=TRUE) THEN
80       gErrorID := OutErrorID;
81       gErrorIDEx := OutErrorIDEx;
82     ELSIF (OutErrorID <> WORD#16#0) THEN
83       gErrorID := OutErrorID;
84       gErrorIDEx := OutErrorIDEx;
85     ELSE
86       // FINS response
87       gArrFINSresp[0] := OutResDat[0];
88       gArrFINSresp[1] := OutResDat[1];
89       gArrFINSresp[2] := OutResDat[2];
90       gArrFINSresp[3] := OutResDat[3];
91
92       gStartFlag := FALSE;
93
94       InExecute := FALSE;
95     END_IF;
96   END_IF;
97
98   IF (SendingFlag=TRUE) AND (OutBusy=FALSE) THEN
99     gErrorID := OutErrorID;
100    gErrorIDEx := OutErrorIDEx;
101  END_IF;
102 END_IF;

```

Index



Index

Numerics

11-bit Identifier	4-3
29-bit Identifier	1-18, 4-3

A

ACK Errors	1-15
Ack Field	1-15
Acknowledge	1-4
Addressing Scheme	1-11
Application Areas	1-5
Arbitration Field	1-14
Assumed cause	6-12
AT specification	3-5
AT specification destination	3-2

B

Bit Level Errors	1-15
Bit Stuffing	1-16
Bit Time	1-7
Bitwise Arbitration	1-12
Bus Access	1-12
Bus Access Control	1-13
Bus Allocation	1-12
Bus Off	1-16
Bus Off Event	6-12
Bus Off State	2-11

C

Cable Length	1-7
CAN Input Message Buffers	3-3
CAN Output Message Buffers	3-3
CAN Slave Controllers	1-19
CIO Area	A-2
Codes	4-3
Command Codes	4-2
Communication Enable	3-2, 3-5
Communications Configuration	1-2
Control and Status Flags	3-3
Control Field	1-14
CPU	1-2
CPU Unit watchdog timer error	6-8
CRC Field	1-15
Cyclic	1-3
Cyclic Mode	5-4
Cyclic Redundancy Check (CRC)	1-15
Cyclic Refresh	3-2

D

Data Exchange with the CPU Unit	3-2
Data Field	1-14
Data Frame	1-9
Destructive Bus Allocation	1-13
Device name	3-6
Device Variable	3-8
Device variable for CJ-series Unit	3-2, 3-5
DIP Switch	2-6
DM Area	A-2
Dot Indicator	2-4

E

ERR Indicator	2-3
Error Counters	1-16
Error Detection	1-15
Error Flag	1-16
Error Frame	1-4
Error Log Storage Area	6-7
EV1 Event	2-11
EV2 Event	2-11
EV3 Event	2-11
EV4 Event	2-11
EV5 Event	2-11
Event code	6-12
Event level	6-12
Event name	6-12
Events	2-11
Extended Format	1-18

F

FINS message communications	1-21
Frame Check	1-15

I

I/O Bus check error	6-10
I/O Map	3-6
I/O Port	3-5
I/O port	3-2, 3-5
I/O setting check error	6-10
IDE Bit	1-18
Identifier	1-14
Identifier Length	1-10
Indicators	2-2
Intermediate Buffer	1-19
Intermission	1-15
ISO 11898	1-5, 1-10

L

License of CAN	1-5
LLC Sublayer	1-8

M

maintenance	6-16
Medium Access Control (MAC)	1-8
Message Communications	1-2, 4-2
Message Transfer	1-9
Messages	1-9
Monitoring	1-15

N

Network Power Error	6-12
Normal Start-up	2-4
Number of Delayed Messages	3-5
Number of Delayed Receive Messages	3-2
Number of Delayed Send Messages	3-2

O

Object Storage	1-19
On Change	1-3
On Change Mode	5-3
OSI Layer 1, Transmission Medium	1-6
OSI Layer 2, Datalink Layer	1-8
OSI Layer 7, Higher Layer Protocol	1-10
OSI Reference Model ISO-7498	1-6

P

Performance	5-2
Power OFF retention	1-20
Priority	1-11

R

Reading and Clearing the Error Log	6-7
Receive Buffer Area	4-9
Receive Flags	3-3
Receive Trigger Area	4-7
Reliability	1-17
Residual Error	1-17
Re-transmission	1-16
RTR Bit	1-18
RUN Indicator	2-3

S

Send Queue Overflow	6-12
Send Trigger Area	4-7
Send Triggers	3-3
Sending Messages	1-11
Serial Bus	1-5, 1-7
Seven-Segment Indicator	2-3
SM1 Mode	1-3
SM2 Mode	1-3
SM3 Mode	1-3
ST1 State	2-11
ST2 State	2-11
ST3 State	2-11
ST4 State	2-11
ST5 State	2-11
Startup Sequence	6-2
Status	3-2, 3-5
Status Indicators	2-2
Sysmac Studio	1-23

T

Transmission Speed	1-7
Triggered	1-3
Triggered Mode	5-2

U

Unambiguous Bus Allocation	1-13
Undetectable Errors	1-17
Unit Addressing	1-11
Unit Configuration	3-6
Unit Control and Status	1-2
Unit No. Switch	2-6
User program	3-5
User-defined Variable	3-5
User-defined variable	1-24, 3-2, 3-5

