User's
Manual

Sequence CPU – Functions
(for F3SP66-4S, F3SP67-6S)

IM 34M6P14-01E

YOKOGAWA
Yokogawa Electric Corporation

IM 34M6P14-01E
1st Edition

Blank Page

Applicable Product

● Range-free-controller FA-M3

- Model Name: F3SP66-4S, F3SP67-6S
- Name: Sequence CPU Module (with network functions)

The document number and document model code for this manual are given below. Refer to the document number in all communications; also refer to the document number or the document model code when purchasing additional copies of this manual.

Document No.	:	IM 34M6P14-01E
Document Model Code	:	DOCIM

Media No. IM 34M6P14-01E (CD) 1st Edition : Jun. 15, 2007-00 (AR)

All Rights Reserved Copyright © 2007, Yokogawa Electric Corporation

Important

■ About This Manual

- This Manual should be passed on to the end user.
- Before using the controller, read this manual thoroughly to have a clear understanding of the controller.
- This manual explains the functions of this product, but there is no guarantee that they will suit the particular purpose of the user.
- Under absolutely no circumstances may the contents of this manual be transcribed or copied, in part or in whole, without permission.
- The contents of this manual are subject to change without prior notice.
- Every effort has been made to ensure accuracy in the preparation of this manual. However, should any errors or omissions come to the attention of the user, please contact the nearest Yokogawa Electric representative or sales office.

■ Safety Precautions when Using/Maintaining the Product

- The following safety symbols are used on the product as well as in this manual.

⚠

Danger. This symbol on the product indicates that the operator must follow the instructions laid out in this user’s manual to avoid the risk of personnel injuries, fatalities, or damage to the instrument. Where indicated by this symbol, the manual describes what special care the operator must exercise to prevent electrical shock or other dangers that may result in injury or the loss of life.

⏚

Protective Ground Terminal. Before using the instrument, be sure to ground this terminal.

⏚

Function Ground Terminal. Before using the instrument, be sure to ground this terminal.

∼

Alternating current. Indicates alternating current.

⎓

Direct current. Indicates direct current.

The following symbols are used only in the user’s manual.

WARNING

Indicates a “Warning”.

Draws attention to information essential to prevent hardware damage, software damage or system failure.

CAUTION

Indicates a “Caution”

Draws attention to information essential to the understanding of operation and functions.

TIP

Indicates a “TIP”

Gives information that complements the present topic.

SEE ALSO

Indicates a “SEE ALSO” reference.

Identifies a source to which to refer.

- For the protection and safe use of the product and the system controlled by it, be sure to follow the instructions and precautions on safety stated in this manual whenever handling the product. Take special note that if you handle the product in a manner other than prescribed in these instructions, the protection feature of the product may be damaged or impaired. In such cases, Yokogawa cannot guarantee the quality, performance, function and safety of the product.
- When installing protection and/or safety circuits such as lightning protection devices and equipment for the product and control system as well as designing or installing separate protection and/or safety circuits for fool-proof design and fail-safe design of processes and lines using the product and the system controlled by it, the user should implement it using devices and equipment, additional to this product.
- If component parts or consumable are to be replaced, be sure to use parts specified by the company.
- This product is not designed or manufactured to be used in critical applications which directly affect or threaten human lives and safety — such as nuclear power equipment, devices using radioactivity, railway facilities, aviation equipment, air navigation facilities, aviation facilities or medical equipment. If so used, it is the user’s responsibility to include in the system additional equipment and devices that ensure personnel safety.
- Do not attempt to modify the product.

■ Exemption from Responsibility

- Yokogawa Electric Corporation (hereinafter simply referred to as Yokogawa Electric) makes no warranties regarding the product except those stated in the WARRANTY that is provided separately.
- Yokogawa Electric assumes no liability to any party for any loss or damage, direct or indirect, caused by the use or any unpredictable defect of the product.

■ Software Supplied by the Company

- Yokogawa Electric makes no other warranties expressed or implied except as provided in its warranty clause for software supplied by the company.
- Use the software with one computer only. You must purchase another copy of the software for use with each additional computer.
- Copying the software for any purposes other than backup is strictly prohibited.
- Store the original media, such as floppy disks, that contain the software in a safe place.
- Reverse engineering, such as decompiling of the software, is strictly prohibited.
- No portion of the software supplied by Yokogawa Electric may be transferred, exchanged, or sublet or leased for use by any third party without prior permission by Yokogawa Electric.

■ General Requirements for Using the FA-M3 Controller

● Avoid installing the FA-M3 controller in the following locations:

- Where the instrument will be exposed to direct sunlight, or where the operating temperature exceeds the range 0°C to 55°C (32°F to 131°F).
- Where the relative humidity is outside the range 10 to 90%, or where sudden temperature changes may occur and cause condensation.
- Where corrosive or flammable gases are present.
- Where the instrument will be exposed to direct mechanical vibration or shock.
- Where the instrument may be exposed to extreme levels of radioactivity.

● Use the correct types of wire for external wiring:

- Use copper wire with temperature ratings greater than 75°C.

● Securely tighten screws:

- Securely tighten module mounting screws and terminal screws to avoid problems such as faulty operation.
- Tighten terminal block screws with the correct tightening torque as given in this manual.

● Securely lock connecting cables:

- Securely lock the connectors of cables, and check them thoroughly before turning on the power.

● Interlock with emergency-stop circuitry using external relays:

- Equipment incorporating the FA-M3 controller must be furnished with emergency-stop circuitry that uses external relays. This circuitry should be set up to interlock correctly with controller status (stop/run).

● Ground for low impedance:

- For safety reasons, connect the [FG] grounding terminal to a Japanese Industrial Standards (JIS) Class D Ground*1 (Japanese Industrial Standards (JIS) Class 3 Ground). For compliance to CE Marking, use braided or other wires that can ensure low impedance even at high frequencies for grounding.

*1 Japanese Industrial Standard (JIS) Class D Ground means grounding resistance of 100 Ω max.

● Configure and route cables with noise control considerations:

- Perform installation and wiring that segregates system parts that may likely become noise sources and system parts that are susceptible to noise. Segregation can be achieved by measures such as segregating by distance, installing a filter or segregating the grounding system.

● Configure for CE Marking Conformance:

- For compliance to CE Marking, perform installation and cable routing according to the description on compliance to CE Marking in the “Hardware Manual” (IM34M6C11-01E).

● Keep spare parts on hand:

- We recommend that you stock up on maintenance parts including spare modules.
- Preventive maintenance (replacement of the module or its battery) is required for using the module beyond 10 years. For enquiries on battery replacement service, contact your nearest Yokogawa Electric representative or sales office. (The module has a built-in lithium battery. Lithium batteries may exhibit decreased voltage, and in rare cases, leakage problems after ten years.)

● Discharge static electricity before operating the system:

- Because static charge can accumulate in dry conditions, first touch grounded metal to discharge any static electricity before touching the system.

● Never use solvents such as paint thinner for cleaning:

- Gently clean the surfaces of the FA-M3 controller with a cloth that has been soaked in water or a neutral detergent and wringed.
- Do not use volatile solvents such as benzine or paint thinner or chemicals for cleaning, as they may cause deformity, discoloration, or malfunctioning.

● Avoid storing the FA-M3 controller in places with high temperature or humidity:

- CPU modules and temperature control modules (F3CT04-□N, F3CR04-□N, F3CV04-1N) have built-in batteries. Avoid storing modules with built-in batteries under high temperature or humidity conditions. Storage at room temperature is recommended.
- The service life of batteries may be shortened when installed or stored at locations of extreme low or high temperatures. Beware that service life of batteries may be drastically shortened under high-temperature conditions (storage temperature should be between –20°C and 75°C).

● Always turn off the power before installing or removing modules:

- Failing to turn off the power supply when installing or removing modules, may result in damage.

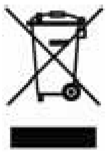
● Do not touch components in the module:

- In some modules you can remove the right-side cover and install ROM packs or change switch settings. While doing this, do not touch any components on the printed-circuit board, otherwise components may be damaged and modules may fail to work.

● Do not use unused terminals:

- Do not connect wires to unused terminals on a terminal block or in a connector. Doing so may adversely affect the functions of the module.

■ Waste Electrical and Electronic Equipment



Waste Electrical and Electronic Equipment (WEEE), Directive 2002/96/EC
(This directive is only valid in the EU.)

This product complies with the WEEE Directive (2002/96/EC) marking requirement.
The following marking indicates that you must not discard this electrical/electronic product in domestic household waste.

Product Category

With reference to the equipment types in the WEEE directive Annex 1, this product is classified as a “Monitoring and Control instrumentation” product.

Do not dispose in domestic household waste.

When disposing products in the EU, contact your local Yokogawa Europe B. V. office.

Introduction

■ Overview of the Manual

This manual describes the sequencing functions of the F3SP66-4S and F3SP67-6S sequence CPU modules (with network functions) of the range-free controller FA-M3.

■ How to Read the Manual

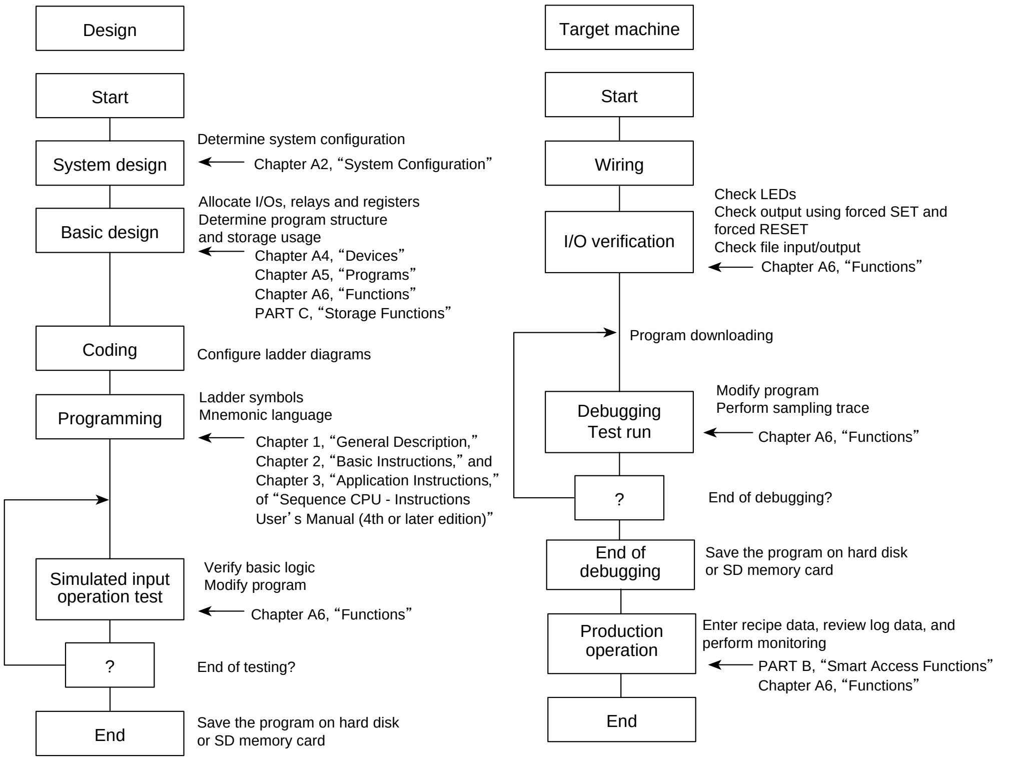
If you are a first-time reader, first read this paragraph, “How to Read the Manual,” and then proceed to Chapter A1, followed by Chapter A3.

For efficiency, you may read only the relevant remaining chapters according to your workflow from system design to system operation.

For details on the network functions of the modules, see "Sequence CPU – Network Functions User’s Manual (for F3SP66-4S, F3SP67-6S)" (IM34M6P14-02E)

The chart below shows the regular workflow, from system design to system operation, as well as chapters you should refer to in each step.

● Work Flow from System Design to System Operation, and Relevant Chapters


Design
Start
System design
Determine system configuration
Chapter A2, “System Configuration”
Basic design
Allocate I/Os, relays and registers
Determine program structure and storage usage
Chapter A4, “Devices”
Chapter A5, “Programs”
Chapter A6, “Functions”
PART C, “Storage Functions”
Coding
Configure ladder diagrams
Programming
Ladder symbols
Mnemonic language
Chapter 1, “General Description,”
Chapter 2, “Basic Instructions,” and
Chapter 3, “Application Instructions,”
of “Sequence CPU - Instructions User’s Manual (4th or later edition)”
Simulated input operation test
Verify basic logic
Modify program
Chapter A6, “Functions”
?
End of testing?
End
Save the program on hard disk or SD memory card
Target machine
Start
Wiring
I/O verification
Check LEDs
Check output using forced SET and forced RESET
Check file input/output
Chapter A6, “Functions”
Program downloading
Debugging Test run
Modify program
Perform sampling trace
Chapter A6, “Functions”
?
End of debugging?
End of debugging
Save the program on hard disk or SD memory card
Production operation
Enter recipe data, review log data, and perform monitoring
PART B, “Smart Access Functions”
Chapter A6, “Functions”
End

F000001.VSD

■ Other User's Manuals

Be sure to read each of the following manuals, in addition to this manual.

● For information on the instructions used with sequence CPUs, refer to:

- Sequence CPU – Instructions User's Manual (IM34M6P12-03E)

● For information on the commands and responses of personal computer link functions, refer to:

- Personal Computer Link Commands User's Manual (IM34M6P41-01E)

● When creating programs using ladder language, refer to:

- FA-M3 Programming Tool WideField2 User's Manual (IM34M6Q15-01E)

● For information on the specifications*, configuration*, installation, wiring, trial operation, maintenance and inspection of the FA-M3, as well as information on the system-wide limitation of module installation, refer to:

- Hardware Manual (IM34M6C11-01E).

*: For information on the specifications of products other than the power supply module, base module, I/O module, cable and terminal block unit, refer to their respective user's manuals.

Read the following user's manuals, as required.

● For information on the network functions of the F3SP66-4S and F3SP67-6S sequence CPU modules, refer to:

- Sequence CPU – Network Functions User's Manual (for F3SP66-4S, F3SP67-6S) (IM34M6P14-02E).

● For information on the functions of fiber-optic FA-bus modules, refer to:

- Fiber-optic FA-bus Module and Fiber-optic FA-bus Type 2 Module User's Manual (IM34M6H45-01E).

● For information on the functions of FA link H and fiber-optic FA link H modules, refer to:

- FA Link H Module, Fiber-optic FA Link H Module User's Manual (IM34M6H43-01E).

● For information on the FL-net functions, refer to:

- FL-net (OPCN-2) Interface Module User's Manual (IM34M6H32-02E).

● For information on the functions of BASIC CPU modules, refer to:

- BASIC CPU Modules and YM-BASIC/FA Programming Language User's Manual (IM34M6Q22-01E).

Copyrights and Trademarks

■ Copyrights

Copyrights of the programs and online manual included in this CD-ROM belong to Yokogawa Electric Corporation.

This online manual may be printed but PDF security settings have been made to prevent alteration of its contents.

This online manual may only be printed and used for the sole purpose of operating this product. When using a printed copy of the online manual, pay attention to possible inconsistencies with the latest version of the online manual. Ensure that the edition agrees with the latest CD-ROM version.

Copying, passing, selling or distribution (including transferring over computer networks) of the contents of the online manual, in part or in whole, to any third party, is strictly prohibited. Registering or recording onto videotapes and other media is also prohibited without expressed permission of Yokogawa Electric Corporation.

■ Trademarks

The trade and company names that are referred to in this document are either trademarks or registered trademarks of their respective companies.

FA-M3

Sequence CPU – Functions (for F3SP66-4S, F3SP67-6S)

IM 34M6P14-01E 1st Edition

CONTENTS

PART A Functions

Appendix A

PART B Smart Access Functions

PART C Storage Functions

FA-M3

Sequence CPU – Functions (for F3SP66-4S, F3SP67-6S)

PART A Functions

IM 34M6P14-01E 1st Edition

Blank Page

A1. Specification and Basic Configuration

This chapter explains the CPU module specifications and the basic configuration of the Range-free Multi-controller FA-M3.

A1.1 Overview

This section describes the overview, features and main functions of the sequence CPU module.

■ Overview

Models F3SP66-4S and F3SP67-6S are sequence CPU modules with built-in network support for use with the Range-free Multi-controller FA-M3.

In addition to high-speed operation and large-scale sequence processing capability, these modules have many more features that help increase development and maintenance efficiency. These features include a RAM disk, an SD memory card slot, a 10BASE-T/100BASE-TX connector, coupled with good compatibility with large data exchange and networking.

■ Features

● High-speed Operation

- 20K steps/1 ms, with shortest scan interval of 200 μs
- High-speed I-P-R-S, which means:
  - High-speed **I**nstruction
  - High-speed **P**erformance
  - High-speed **R**esponse
  - High-speed **S**can

● Sensor Control Function (see Part A of this manual)

In addition to normal scanning, each CPU module has an independent, multiple constant scan function, permitting fast scanning. Fast response is also achievable with a single CPU.

You can execute a block of your program at high speed and fixed intervals (200 μs-minimum), separately from normal scanning. This feature enables you to eliminate the effects of a fault diagnosis program or MMI program, as well as ensure stable control program operation.

● Object Ladder (see user's manuals IM34M6Q15-01E or IM34M6P12-03E)

The FA-M3 Programming Tool WideField2, an object-oriented ladder language development tool, is available with the CPU module. This tool not only increases software development productivity over and above structured programming, but also simplifies program maintenance.

● Built-in 10BASE-T/100BASE-TX Connector (see user's manual IM34M6P14-02E)

The built-in 10BASE-T/100BASE-TX connector can be used with TCP/IP, UDP/IP, FTP client, FTP server network protocols. In addition, it also enables connection to the FA-M3 Programming Tool WideField2 and also data exchange with a display or SCADA software using higher-level link commands.

● Virtual Directory Function (see user's manual IM34M6P14-02E)

The virtual directory function is provided as an extended FTP server function to allow automatic loading of a data file into devices using the FTP put command and retrieval of device data as a data file using the FTP get command, as well as loading of programs, saving of programs and switching of operating mode using FTP commands.

● Built-in SD Memory Card Slot (see Part C of this manual)

An SD memory card can be used for storing programs and data. The module adopts the standard PC FAT16 format so data on the card can be accessed from a PC without special software.

● Built-in RAM Disk (see Part C of this manual)

A 4MB RAM disk is provided for faster file processing.

● Text Processing Support (see Part A of this manual)

Constant definition and M3 escape sequence features are available when the module is used with the Ladder Programming Tool WideField2. The constant definition function simplifies definition of strings and contiguous byte data, as well as reuse of constants. With the use of M3 escape sequences, mixed text and binary data can be defined.

● Maintenance Using Smart Access Functions without Need of a PC (see Part B of this manual)

- By operating the rotary switch (called the MODE switch) together with the push button (called the SET switch), both of which are located on the front panel of the module, you can load and save programs, get log files and perform other maintenance operations without the need for a PC.
- Card batch file functions enable program replacement or device data retrieval to be automatically triggered by a SD memory card insertion, an error, program execution or some other event.

● Other Features

- A compact body allows for reduced panel enclosure size.
- Large-capacity programs and large device sizes are supported to cope with advanced, complex control applications.
- Index modification, indirect designation, structure macro and structured ladder language simplifies program design and maintenance.
- The device size and operating method can be flexibly configured to suit your application needs.
- A rich set of functions are provided to facilitate program debugging and maintenance. For example, a forced SET/RESET function independent of program computation results.
- A carefully designed self-diagnosis function supplements a highly reliable design.
- Macro instruction functions allow you to create and register new instructions.
- Many communication and file processing instructions are provided.
- The sampling trace function acquires and displays the states of multiple devices for a maximum of 1024 scans.
- The SIO port supports the higher-level link service (personal computer link function) and thus enables connection to a host computer or a monitor without the need for adding a module. The 10BASE-T/100BASE-TX connector also supports higher-level link service.
- The module can be mounted in slots 2 to 4 of the main unit, for use as an add-on CPU module for sequence processes added to the main CPU module (F3SP21, F3SP25, F3SP28, F3SP35, F3SP38, F3SP53, F3SP58, F3SP59, F3SP66 and F3SP67).

- Program protection feature, network filter setup, function removal and other features improve security.

■ Functions

[Primarily described in Part A of this manual]

- Configuration (setup of parameters, including device size, range of devices to be latched in case of power failure, and external output to be retained in case of sequence stop)
- CPU properties (communcations setup, etc.)
- Sensor control, constant scan (1 ms to 190 ms)
- Debugging (forced SET/RESET instructions, online edit, sampling trace, user log, etc.)
- Error log (system log), user log
- Clock (year, month, day, hour, minute, second, and day of week)
- Partial download
- Constant definition, M3 escape sequence
- Program protection, block protection, CPU properties protection
- Function removal, network filter
- Self-diagnosis

Etc.

[Primarily described in Part B of this manual]

- Rotary switch function, card batch file function, user LED function
- Boot modes using SD memory card, load project, save project

Etc.

[Primarily described in Part C of this manual]

- SD memory card, RAM disk, file system

Etc.

[Primarily described in user's manual IM 34M6P14-02E]

- TCP/IP socket, UDP/IP socket communications
- FTP client function, FTP server function
- Virtual directory function
- Higher-level link service (personal computer link function)
- Remote programming service

Etc.

SEE ALSO

- "FA-M3 Programming Tool WideField2 User's Manual" (IM34M6Q15-01E)
- "Sequence CPU – Network Functions User's Manual (for F3SP66-4S, F3SP67-6S)" (IM34M6P14-02E)
- "Sequence CPU – Instructions User's Manual" (IM34M6P12-03E)

A1.2 Specifications

The basic specifications of the module are given below.

A1.2.1 List of Specifications

■ General Specification

Table A1.2.1 General Specification (F3SP6□-□S)

Item	Specification
Operating ambient temperature range	0°C to 55°C (excluding SD memory card)
Operating ambient humidity range	10 to 90% RH (non-condensing)
Operating ambient environment	Must be free of corrosive gases, flammable gases or heavy dust.
Storage ambient temperature range	-20°C to 75°C
Storage ambient humidity range	10 to 90% RH (non-condensing)
Current consumption	850 mA (at 5 V DC)
External dimensions	28.9 (W) × 100 (H) × 113.2 (D) mm (excluding protrusions)
Weight	220 g

■ Functional Specification

Table A1.2.2 Functional Specification (F3SP6□-□S) (1/2)

Item		Specifications	
		F3SP66-4S	F3SP67-6S
Control method		Repetitive computation based on stored programs	
I/O computation method		Refresh method/Direct I/O instruction	
Programming language		Object ladder language	
No. of I/O points		4096 max.	8192 max. (including remote I/O points)
No. of internal relays (I)		16384	32768
No. of shared relays (E)		2048	2048
No. of extended shared relays (E)		2048	2048
No. of link relays (L)		8192	16384
No. of special relays (M)		9984	9984
No. of timers (T)		1024	2048
No. of counters (C)		1024	1024
No. of data registers (D)		16384	32768
No. of shared registers (R)		1024	1024
No. of extended shared registers (R)		3072	3072
No. of file registers (B)		32768	262144
No. of link registers (W)		8192	16384
No. of special registers (Z)		1024	1024
No. of labels		1024	1024
No. of input interrupt processing routines		4	4
Constants	Decimal constant	16-bit instruction: -32768 to 32767 (constant definition supported) 32-bit instruction: -2147483648 to 2147483647 (constant definition supported)	
	Hexadecimal constant	16-bit instruction: $0 to $FFFF (hexadecimal number) (constant definition supported) 32-bit instruction: $0 to $FFFFFFFF (hexadecimal number) (constant definition supported)	
	Character-string constant	16-bit instruction: e.g. “AB” 32-bit instruction: e.g. “ABCD” Constant definition: Up to 255 characters	
	Binary constant	Constant definition: Up to 256 bytes of contiguous data	
	IEEE single-precision floating-point constant	32-bit instruction: e.g. 1.23, -3.21 Approximately -3.4×10^{38} to 3.4×10^{38} (constant definition supported)	
	Index constant	0 to 2047	

Table A1.2.3 Functional Specification (F3SP6□-□S) (2/2)

Item		Specifications	
		F3SP66-4S	F3SP67-6S
Program size		56K steps max.	120K steps max.
Program + tag name definition + constant definition		112K steps max.	240K steps max.
No. of program blocks		1024 max.	
No. of instructions	Basic instructions	37	
	Application instructions	389	
Number of macro instructions		256 max.	
Instruction execution time	Basic instructions	0.0175 to 0.07 μs per instruction	
	Application instructions	0.07 μs min. per instruction	
Special module High-speed Read (HRD) Instruction/special module High-speed Write (HWR) Instruction		64 instructions each	
Scan monitoring time		Variable from 10 to 200 ms	
Constant scan		1 to 190 ms, configurable in 0.1 ms increments.	

■ SD Memory Card Slot

Table A1.2.4 SD Memory Card Slot Specifications

Item	Specifications
Interface	SD Memory Card
Power supply to card	3.3 V
Current to card	200 mA max
LED indicator	LED is lit when a card is mounted; LED blinks when a card is being accessed.
Short-circuit protection	Shuts off power supply to SD memory card if short-circuit is detected.
File format	FAT16

● Operating ambient temperature range of SD memory card

Beware that the internal temperature of the SD memory card slot is normally higher than the ambient temperature.

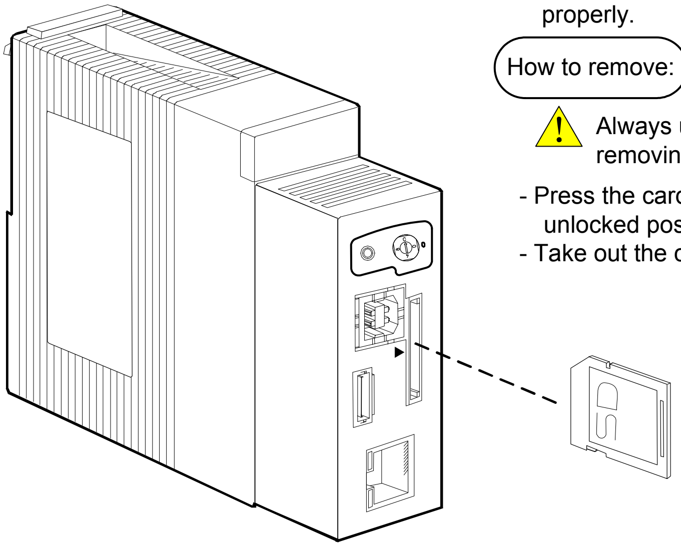
● Short-circuit Protection

The short-circuit protection protects the system against overcurrent due to insertion of a short-circuited SD memory card by shutting off the power supply to the SD memory card.

● Inserting and Removing SD Memory Card

When inserting an SD memory card, ensure that its top side is facing the power supply module.

To remove an SD memory card, press the SD memory card to unlock it before removing the card.



How to insert:

- Insert the card with its face toward the ▾ mark on the slot.
- It clicks into a locked position when inserted properly.

How to remove:

⚠ Always unmount the card before removing it.

- Press the card once so that it clicks into an unlocked position.
- Take out the card.

FA0101.VSD

Figure A1.2.1 Inserting and Removing SD Memory Card

CAUTION

- Do not forcedly pull out an SD memory card, which is inserted and locked in the memory card slot, as this may damage the slot.
- Do not remove an SD memory card, which is mounted or being accessed as it may damage the data on the card. Always unmount an SD memory card and confirm that the **SD** LED has turned off before removing the card.

SEE ALSO

- For details on SD memory card slot and SD memory card, see Chapter C1, "Memory Card."
- For details on recommended memory cards, see "■ Recommended Memory Cards" of Subsection C1.2.1, "Memory Card Specifications."

■ USB Port

Table A1.2.5 USB Port Specifications

Item	Specifications
Connector type	USB type B connector
Interface	USB Rev1.1 compliant device
Transfer rate	12 Mbps max.
Bus power supply	Not used

■ SIO Port

Table A1.2.6 SIO Port Specifications

Item	Description		Configurable? *1
Interface	EIA RS-232-C compliant		
Transmission mode	Half-duplex transmission		
Synchronization	Start-stop synchronization		
Transmission rate (bps)	9600/19200/38400/57600/115200		✓
Data format	Start bit	: 1 bit	
	Data length	: 8 bits (fixed)	
	Parity bit	: None or Even	✓
	Stop bit	: 1 bit (fixed)	
Error detection	Parity check		
	Checksum	: Yes/No	✓
Control line (RS-232-C)	Not used		
Xon/Xoff	Not used		
Protocol	Proprietary protocol		
End character	Yes/No		✓
Protection feature *2	Yes/No		✓
Transmission distance	12 m max.		
External connection	Dedicated cable		

*1: Items marked ✓ are customizable by configuration. However, note that not all combinations of transmission rate and parity check are allowed. For details, see Subsection A6.11.4, "Personal Computer Link Function Setup."

*2: Enabling write protection prohibits writing to the FA-M3.

*3: Connection to WideField2 via the SIO port is not supported.

CAUTION

The SIO port of the module uses neither the RS-232-C control line nor Xon/Xoff characters. Therefore, a monitor or PC may fail to receive data at high transmission rates.

10BASE-T/100BASE-TX Connector

Table A1.2.7 10BASE-T/100BASE-TX Connector Specifications

Item		Specifications	
		Ethernet	
Interface		10BASE-T	100BASE-TX
Transmission specification	Access control	CSMA/CD	
	Transmission rate	10 Mbps	100 Mbps
	Transmission method	Baseband	
	Maximum distance between nodes	100 m[*1]	
Protocol [*2]		TCP, UDP, IP, ICMP, ARP, FTP	
Transmission rate setup		Auto-detect (10 Mbps or 100 Mbps)	
Transmission mode		Full duplex or half duplex	

*1: Distance between a hub and the module

*2: Some parts of the software of the Regents of University of California have been incorporated.

MAC Address

The MAC address is inscribed on a nameplate on the left face of the module.

It is stored in special registers according to the format described in the table below.

Table A1.2.8 MAC Address Special Registers

Category	Sequence CPU Module Status Registers		
No.	Name	Function	Description
Z0114	MAC Address	MAC address low word	Low-order 16 bits [$xxxx]
Z0115		MAC address mid word	Mid-order 16 bits [$64xx]
Z0116		MAC address high word	High-order 16 bits [$0000]

RAM Disk

Table A1.2.9 RAM Disk Specifications

Item	Specification
Capacity	4M bytes (before formatting)
Memory type	SDRAM
Disk type	Fixed disk
Volatility	Volatile
Maximum number of write operations	Unlimited
Sector size	512 bytes
Cluster size	4096 bytes (8 sectors)

SEE ALSO

For details on RAM disk, see Chapter C2, "RAM Disk."

A1.2.2 Device List

Table A1.2.10 Device List

Device		Code	F3SP66-4S		F3SP67-6S		Remarks
			Range	Quantity	Range	Quantity	
Input relay		X	X00201 to X71664 (discontinuous)	4096	X00201 to X71664 (discontinuous)	8192	The range used depends on the module type
Output relay		Y	Y00201 to Y71664 (discontinuos)		Y00201 to Y71664 (discontinuos)		
Internal relay		I	I00001 to I16384	16384	I00001 to I32768	32768	
Shared relay	Non-latched type	E	E0001 to E2048	2048	E0001 to E2048	2048	These devices default to zero in quantity. Be sure to configure the devices when using the CPU module in a multi-CPU configuration.
Extended shared relay			E2049 to E4096	2048	E2049 to E4096	2048	
Link relay	Non-latched type	L	L0001 to L72048 (discontinuous)	8192	L0001 to L72048 (discontinuous)	16384	Used in FA link and FL-net communications.
Special relay		M	M0001 to M9984	9984	M0001 to M9984	9984	
Timer	100 µs Timer	T	T0001 to T0016	2048 in total	T0001 to T0016	3072 in total	Configurable for up to 16 timers
	1 ms Timer		T0001 to T2048		T0001 to T3072		Configuration limit correlated to counters (C) *1.
	10 ms Timer						
	100 ms Timer						
Continuous timer	100 ms Timer						
Counter	Latched type	C	C0001 to C2048		C0001 to C3072		Configuration limit correlated to Timers (T) *1
Data register	Latched type	D	D00001 to D16384	16384	D00001 to D32768	32768	
File register	Latched type	B	B000001 to B32768	32768	B000001 to B262144	262144	
Link register	Non-latched type	W	W00001 to W72048 (discontinuous)	8192	W00001 to W72048 (discontinuous)	16384	Used in FA link and FL-net communications.
Special register		Z	Z0001 to Z1024	1024	Z0001 to Z1024	1024	
Index register		V	V001 to V256	256	V001 to V256	256	
Shared register	Non-latched type	R	R0001 to R1024	1024	R0001 to R1024	1024	These devices default to zero in quantity. Be sure to configure the devices when using the CPU module in a multi-CPU configuration.
Extended shared register			R1025 to R4096	3072	R1025 to R4096	3072	

*1: See Table below.

Table A1.2.11 Device Capacities and Configuration Restrictions

Device	Code	F3SP66-4S		F3SP67-6S	
		Default value	Setup Restrictions	Default value	Setup Restrictions
Timer	T	1024	Total for timers and counters: 2048 max. Default for 100-µs and 1-ms timers: 0	2048	Total for timers and counters: 3072 max. Default for 100-µs and 1-ms timers: 0
Counter	C	1024		1024	
Shared relay	E	0	2048 max.	0	2048 max.
Extended Shared relay	E	0	2048 max.	0	2048 max.
Shared register	R	0	1024 max.	0	1024 max.
Extended shared register	R	0	3072 max.	0	3072 max.

A1.2.3 Components and Their Functions

This subsection describes the components of the module. The description applies to both F3SP66-4S and F3SP67-6S.

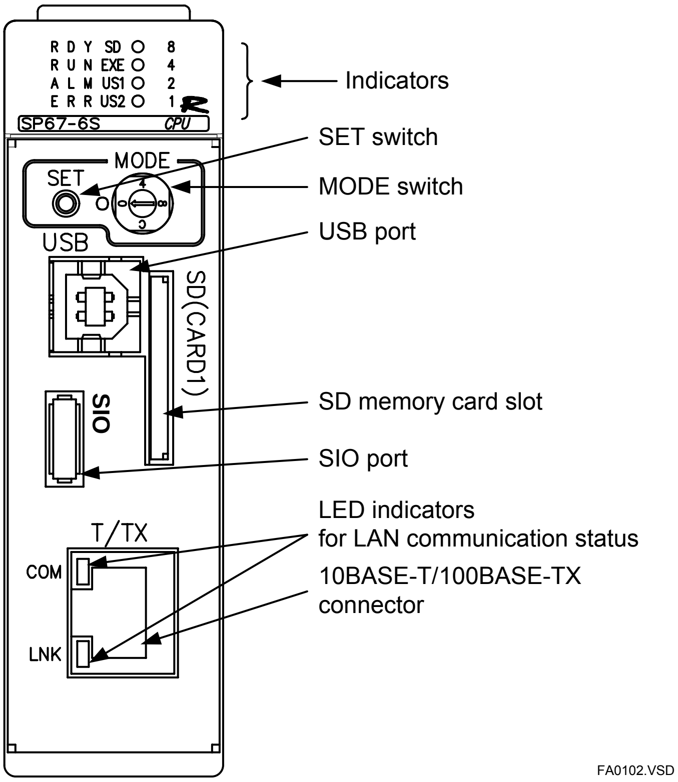


Figure A1.2.2 Names of Components

■ LED Indicators

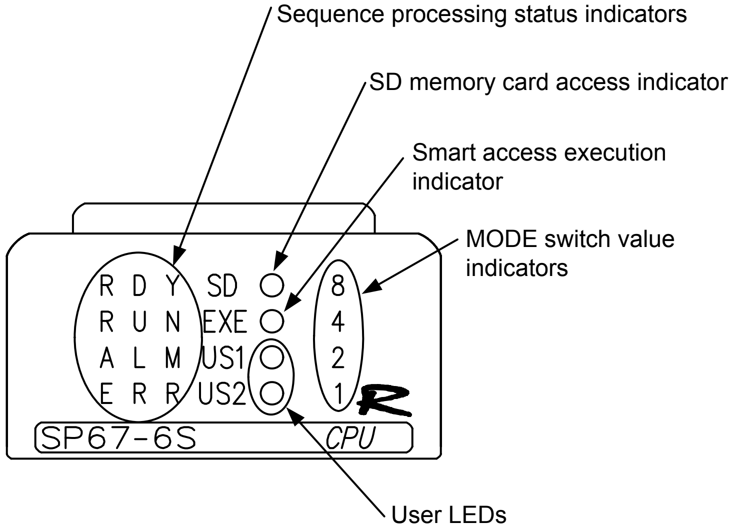

Sequence processing status indicators
SD memory card access indicator
Smart access execution indicator
MODE switch value indicators
RDY
RUN
ALM
ERR
SD
EXE
US1
US2
8
4
2
1
SP67-6S
CPU
User LEDs
FA0103.VSD

Figure A1.2.3 Functions of LED Indicators

● Sequence Processing Status Indicators

These LEDs indicate the operation status of the sequence process function.

Table A1.2.12 Sequence Process Status Indicators

LED		Color	Status	
RDY	(READY)	Green	Lit	Normal
			Off	Major failure
RUN	(RUN)	Green	Lit	Program execution in progress
			Off	Program in Stop mode
			Blinks	Shutdown in progress (Transiting to Program Stop state)
ALM	(ALARM)	Yellow	Lit	Minor failure
			Off	Normal
ERR	(ERROR)	Red	Lit	Moderate failure
			Off	Normal

Note:
- Major failure = No operation possible due to hardware failure.
- Moderate failure = Program execution cannot continue.
- Minor failure = An error is detected but program execution can continue.

The table below summarizes combinations of the LED indicators as classified by the severity of failure.

Table A1.2.13 LED Indicator Combinations Based on the Severity of Failure

Status / LED Indicator	Normal	Major Failure	Moderate Failure	Minor Failure
RDY	○	●	○	○
RUN	○	●	●	○
ALM	●	△	△	○
ERR	●	○	○	●

○: Lit, ●: Off, △: Lit or Off

● SD Memory Card Access Indicator

This LED indicates the status of the SD memory card.

Table A1.2.14 SD Memory Card Access Indicator

LED	Color	Status	
SD	Green	Lit	Card is mounted.
		Blinks	Card is being accessed.
		Off	Card is unmounted.

SEE ALSO

For details on SD memory card access indicator, see Chapter B4, "LED Specifications."

● Smart Access Execution Indicator

This LED indicates the execution status of the smart access function.

Table A1.2.15 Smart Access Function Execution Indicator

LED	Color	Status	
EXE	Green	Lit	Smart access function is running.
		Blinks	Smart access function error
		Off	No smart access function is running.

SEE ALSO

- For details on LED indicators, see Chapter B4, "LED Specifications."
- For details on smart access functions, see Part B, "Smart Access Functions."

● User LEDs

Whether these LED are lit or off is controlled by user programs.

Table A1.2.16 User LEDs

LED	Color	Status	
US1	Green	Lit	Controlled by user program
		Blinks	
		Off	
US2	Green	Lit	Controlled by user program
		Blinks	
		Off	

SEE ALSO

For details on user LED indicators, see Chapter B4, "LED Specifications."

● MODE Switch Value Indicators

These LEDs indicate the current value of the MODE switch (rotary switch).

Table A1.2.17 MODE Switch Value Indicator

LED	Color	Status
8	Green	The current MODE switch value can be calculated by summing the values of lit LEDs in hexadecimal.
4	Green	
2	Green	
1	Green	

SEE ALSO

For details on the MODE switch value indicator, see Chapter B4, "LED Specifications."

■ MODE Switch, SET Switch

These switches can be used to specify the processing to be performed by the smart access function.

Set the MODE switch (16-position rotary switch) to the number corresponding to the required processing, and press the SET switch to invoke the smart access function.

Note that the MODE switch value is indicated by the MODE switch value LED indicators, but not by the position of the arrow of the MODE switch.

SEE ALSO

For details on the MODE switch, see Part B, "Smart Access Functions."

■ USB Port

This USB port can be used to connect to the Programming Tool WideField2 (which runs on a PC).

■ SD Memory Card Slot (CARD1)

The card slot is used for mounting an SD memory card.

SEE ALSO

- For details on SD memory card, see Chapter C1, "Memory Card."
- For details on recommended SD memory card, visit the FA-M3 website (URL: http://www.fa-m3.com). Operation is not guaranteed if an SD memory card, which is not on the recommended list, is used with the module.

■ SIO Port

The SIO port is used for connecting to a monitor or personal computer, using the higher-level link service (personal computer link function) as the protocol.

It cannot be used for connecting to Programming Tool WideField2.

SEE ALSO

For details on SIO port, see Chapter 4, "Higher-level Link Service (Personal Computer Link Function)" of "Sequence CPU – Network Functions User's Manual (for F3SP66-4S, F3SP67-6S)" (IM34M6P14-02E).

■ 10BASE-T/100BASE-TX Connector

This connector is used for connecting to Ethernet.

The transmission rate is automatically determined by auto-negotiation.

SEE ALSO

- For details on the 10BASE-T/100BASE-TX connector, see "Sequence CPU – Network Functions User's Manual (for F3SP66-4S, F3SP67-6S)" (IM34M6P14-02E).
- For details on the setup related to the 10BASE-T/100BASE-TX connector, see Section A9.5, "CPU Property Items."

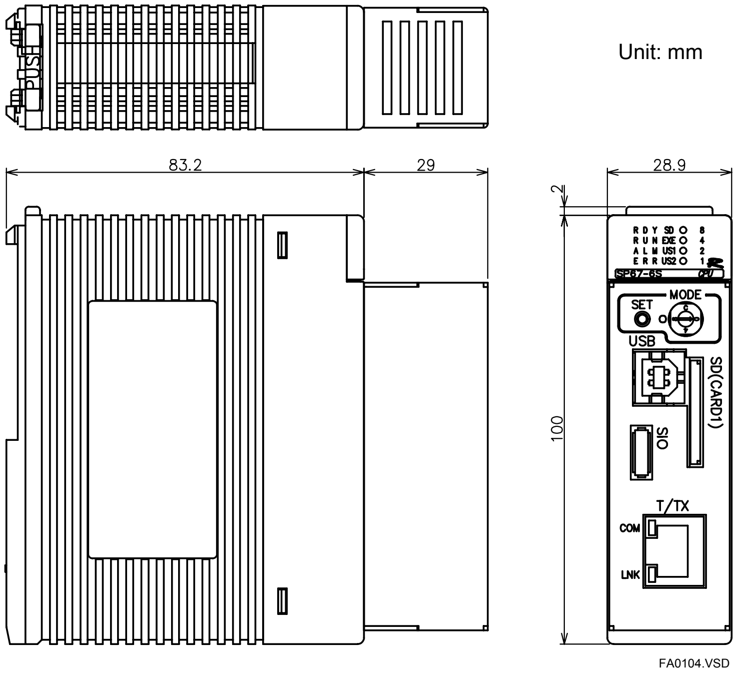
■ LAN Communication Status LED Indicators

These LEDs indicate the Ethernet communication status.

Table A1.2.18 LAN Communication Status LED Indicators

LED	Color		Status
			Ethernet
COM	Yellow	Lit	Sending or receiving
		Off	Neither sending nor receiving
LNK	Green	Lit	A link is established to a network.
		Off	No link is established to any network.

A1.2.4 External Dimensions



Unit: mm

Figure A1.2.4 External Dimensions

A1.3 Basic Configuration

Units, slots, and input/output relays are identified with unique numbers. These numbers are used in parameters of ladder instructions and configuration setup.

A1.3.1 Unit

A unit is a system with the minimum configuration consisting of the following modules. Install these modules on the base module to compose the unit.

The location where you install a module is called a slot.

Table A1.3.1 Unit Components (Modules)

Name	Description
Base module	Five types are available, allowing different number of modules to be mounted.
Power supply module	One power supply module must always be mounted on the base module.
CPU module	At least one CPU module is required. Several types are available with different functionalities.
I/O module	Various types are available with different types of I/O and number of I/O points.
Special module	Various types are available, including analog I/O and communication modules.

■ Main Unit

Install the power supply module in the leftmost slot of the base module and the CPU module in the slot on the immediate right of the power supply module. Then, install required I/O and special modules in the remaining slots. A system with this configuration is called a main unit.

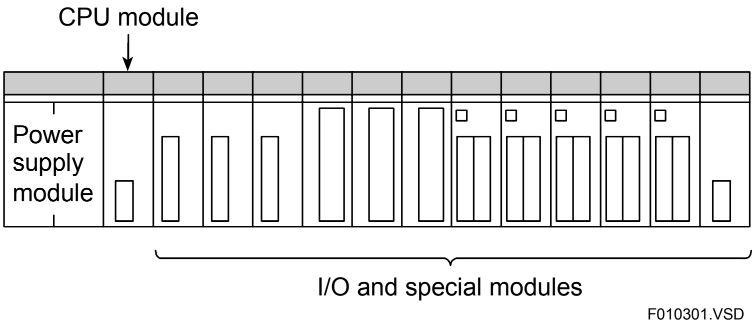


Figure A1.3.1 Main Unit

■ Subunit

A subunit is an I/O expansion unit. It is connected to the main unit through a fiber-optic FA-bus or fiber-optic FA- bus type 2.

A maximum of seven subunits can be connected to the main unit and are identified by their unit numbers. With fiber-optic FA-bus type 2, you can separate any single subunit into a maximum of eight stations.

SEE ALSO

- For details on the method of separation, see “Fiber-optic FA-bus Module, Fiber-optic FA-bus Type 2 Module User's Manual” (IM34M6H45-01E).
- For details on unit number, see Subsection A1.3.2, "Slot Number."

A1.3.2 Slot Number

A slot number indicates the position of a slot where a module is installed.

A slot number is defined as a three-digit integer, as shown below.

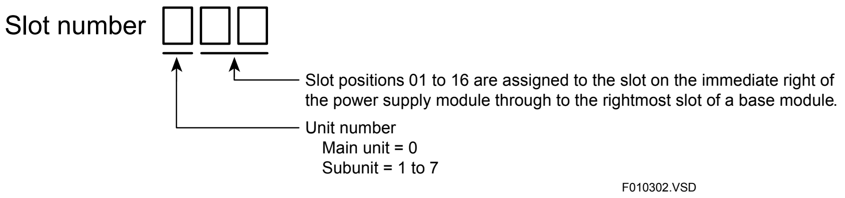

Slot number
Slot positions 01 to 16 are assigned to the slot on the immediate right of the power supply module through to the rightmost slot of a base module.
Unit number
Main unit = 0
Subunit = 1 to 7
F010302.VSD

Figure A1.3.2 Slot Numbers (1 of 2)

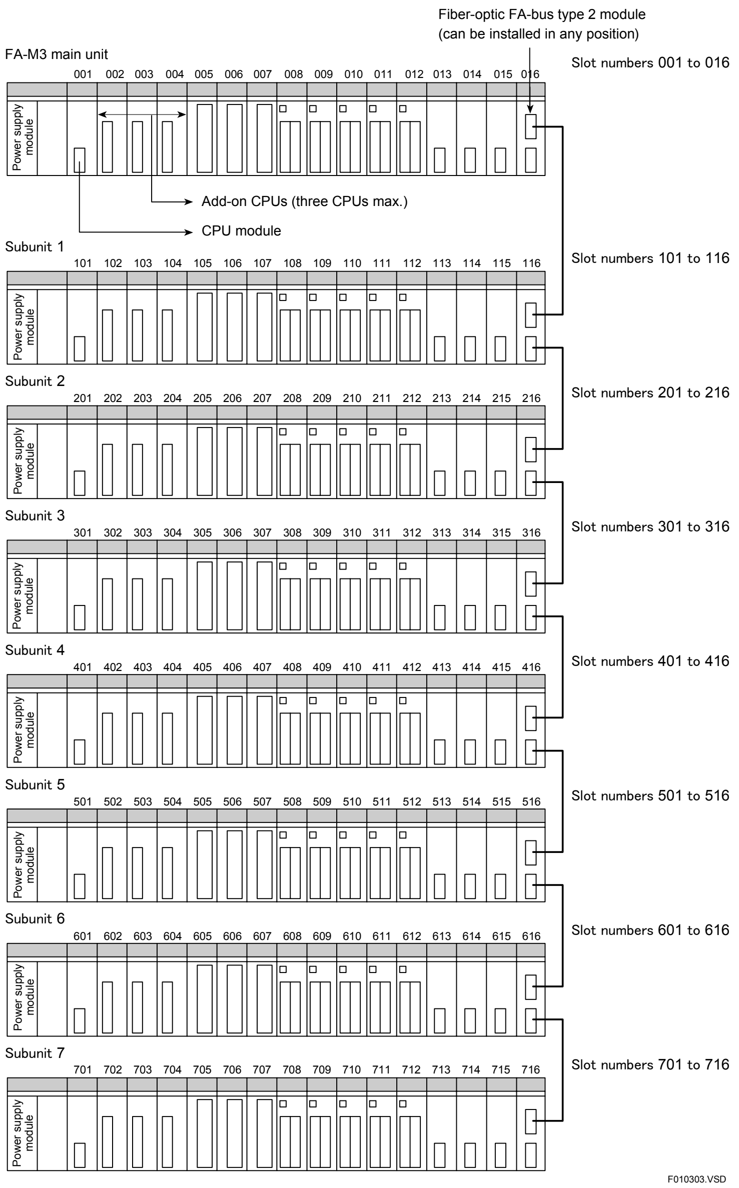


Figure A1.3.3 Slot Numbers (2 of 2)

Install fiber-optic FA-bus type 2 modules in both the main unit and a subunit and connect these modules with a fiber-optic cable.
You can attach up to seven subunits to the main unit. Subunit numbers are assigned by setting the rotary switch on the front panel of each fiber-optic FA-bus type 2 module.

A1.3.3 I/O Relay Number

Each input relay (X) or output relay (Y) number is defined as a slot number followed by an I/O relay number. The I/O relay number is a number corresponding to each terminal of an I/O module.

Example:

The output relay number for terminal 6 of an F3YC08-0N module installed in slot 005 is defined as follows.

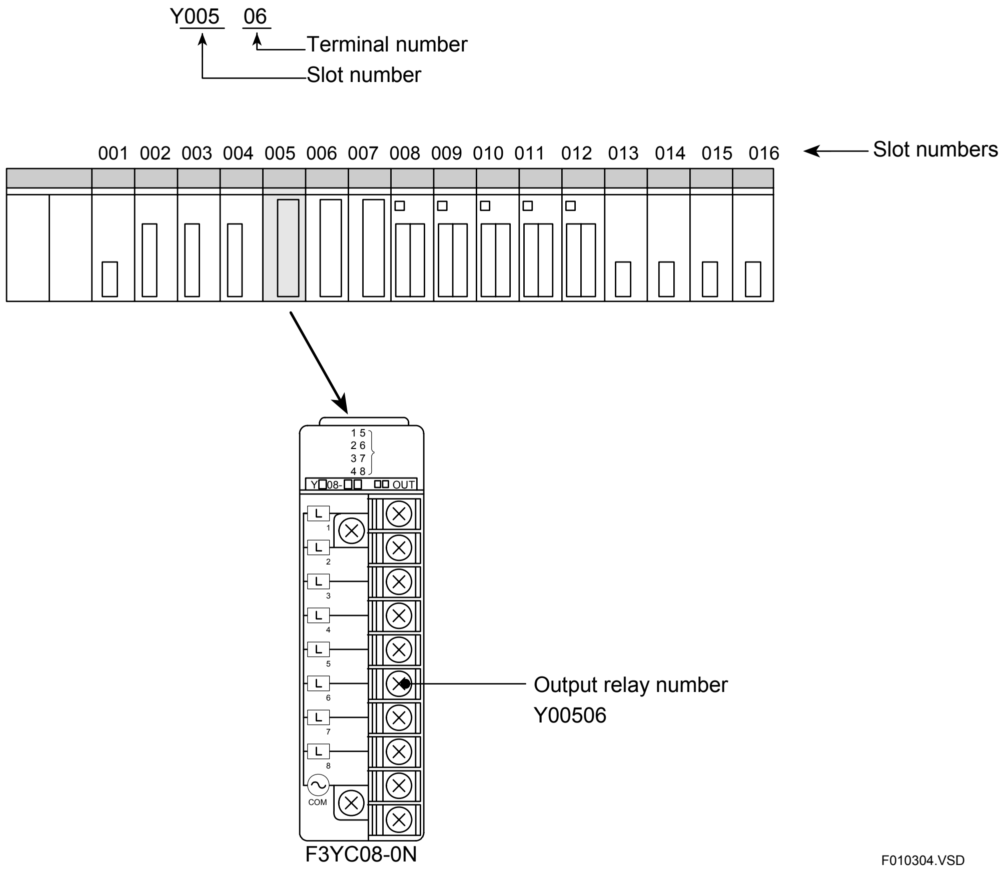


Figure A1.3.4 Output Relay Number

The input and output terminal numbers of a mixed-I/O module or special module with 32 input and output points each are assigned as 1 to 32 and as 33 to 64, respectively.

A2. System Configuration

This chapter describes the FA-M3 system configuration and programming tools.

A2.1 Basic System Configuration

The basic system configuration refers to a system consisting of a main unit only.

SEE ALSO

For details on the main unit, see subsection A1.3.1, “Unit".

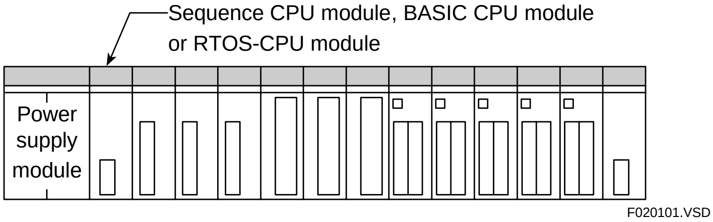


Figure A2.1.1 Example of Basic System Configuration (when a 13-slot base module is used)

A2.2 Multi-CPU System Configuration

This section describes a multi-CPU system configuration and the handling of I/O modules in a multi-CPU system.

A2.2.1 Multi-CPU System Configuration

A multi-CPU system configuration refers to a system with multiple CPU modules installed in the main unit. A maximum of four CPU modules can be installed in slots 001 to 004 on the main unit.

A CPU module installed in slot 001 serves as the main CPU module and CPU modules installed in slots 002 to 004 serve as add-on CPU modules.

Up to four sequence CPU modules can be installed at the same time, while only up to two RTOS-CPU modules and only one F3BP□□ BASIC CPU module is allowed in this system configuration. The maximum number of installed modules is further restricted by the current consumption of individual CPU modules and the power supplied by the power supply module used.

A CPU module installed in the Nth (N = 1 to 4) slot is called "CPU N".

TIP

- A BASIC CPU module refers to a CPU module which runs BASIC programs.
- An RTOS-CPU module refers to a CPU module which runs a real-time OS such as ITRON.

SEE ALSO

- For details on installation restrictions due to the power supply module used, see "Hardware Manual" (IM34M6C11-01E).
- Installation restrictions for individual CPU module types may vary due to functional extensions to the product specifications. Always check the documentation of individual CPU modules before configuring a multi-CPU system.

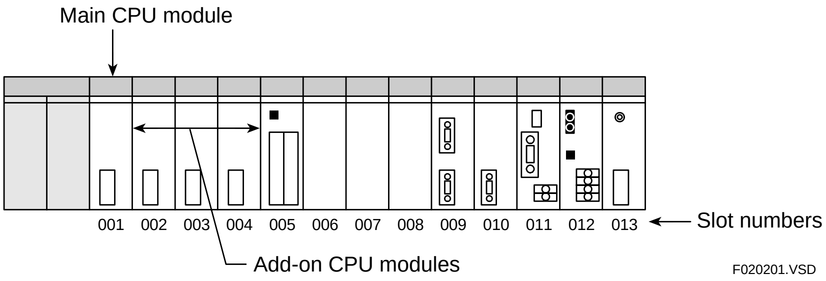


Figure A2.2.1 Example of Multi-CPU System Configuration

CAUTION

Turning on the power with a CPU module installed in the 5th or later slot clears the memory. Do not do this unless it is what you intended.

SEE ALSO

For details on what happens if a CPU module is installed in the 5th or later slot, see Subsection B1.5.4, "Restore Factory Settings".

A2.2.2 Handling I/O Modules in Multi-CPU System

■ Input Modules

With input modules, you can read input data through multiple CPUs. To do this, configure the CPUs so that they share the same input sampling interval for the input module in question.

CAUTION

Beware that the sampling interval that can be set varies depending on the CPU type.

■ Output Modules and Special Modules with Y□□□□□ Output Relays (Y)

● Combination of F3SP28, F3SP38, F3SP53, F3SP58, F3SP59, F3SP66 and F3SP67 Modules

You can output data from multiple sequence CPU modules separately to the output relays of the same output module on 16-point basis. To do this, configure the unused output relays (Y) of the output module as “Not used” on 16-point basis. In addition, all output relays within the same module must be configured with the same output mode (hold or reset) in the event that sequence processing stops.

● Other Combinations of CPU Modules

You may not use the same output module with multiple CPUs. Configure all CPUs that do not use the output module so that the output module is set to “Not Used".

SEE ALSO

For details on how to configure for modules that are not used, see Section A4.1.4, "Configuring DIO Modules".

A2.3 Extended System Configuration

An extended system configuration refers to a system configured by adding remote I/O, personal computer link, FA link, and FL-net to the basic system.

A2.3.1 Remote I/O System

The remote I/O system refers to a system configured using fiber-optic FA-bus and FA-bus type 2 communication modules.

The number of remote I/O points is included in the count of all I/O points.

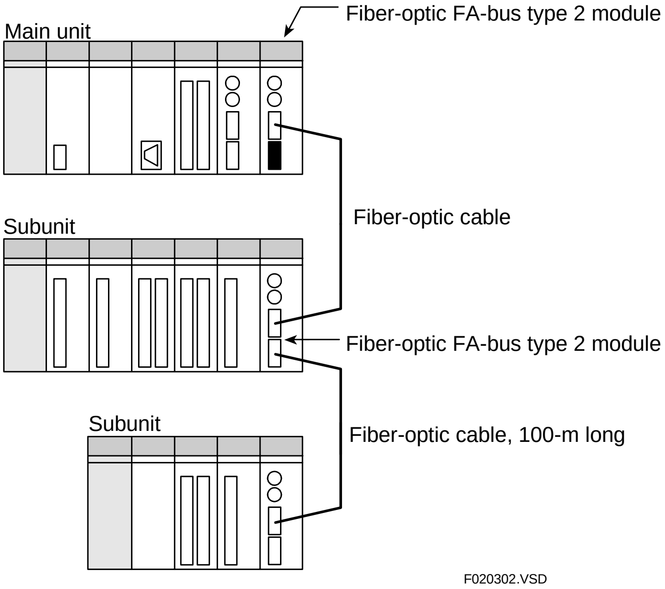

Main unit
Fiber-optic FA-bus type 2 module
Fiber-optic cable
Subunit
Fiber-optic FA-bus type 2 module
Subunit
Fiber-optic cable, 100-m long
F020302.VSD

Figure A2.3.1 Example of System Using Fiber-optic FA-bus Type 2 Modules

A2.3.2 Personal Computer Link System

The personal computer link system refers to a system configured by connecting a personal computer or a monitor to the main unit through a personal computer link module. The personal computer or monitor can also be directly connected to the sequence CPU module via its SIO port or 10BASE-T/100BASE-TX connector.

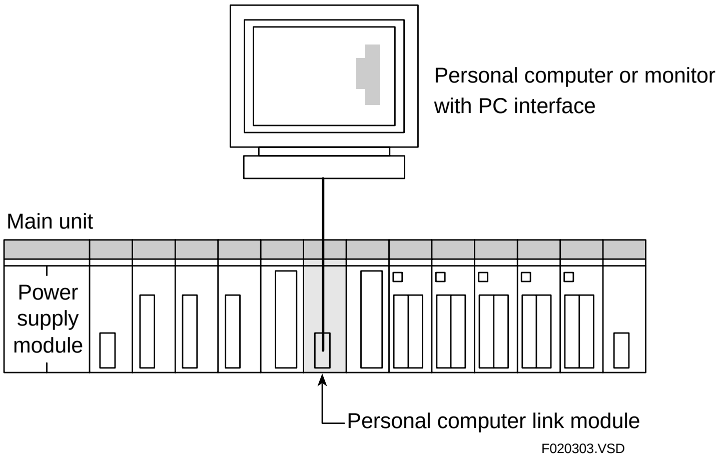


Figure A2.3.2 Example of Personal Computer Link System

A2.3.3 FA Link System

The FA link system refers to a system that employs FA link communication to build a network system with programmable controllers.

The types of communication covered by an FA link system are:

- FA link H communication (FA link H module), and
- Fiber-optic FA link H communication (fiber-optic FA link H module).

Unless otherwise specified, the term “FA link” in this manual comprehensively refers to these two types of communication.

SEE ALSO

For details on FA link, see “FA Link H and Fiber-optic FA Link H Modules User's Manual” (IM34M6H43-01E).

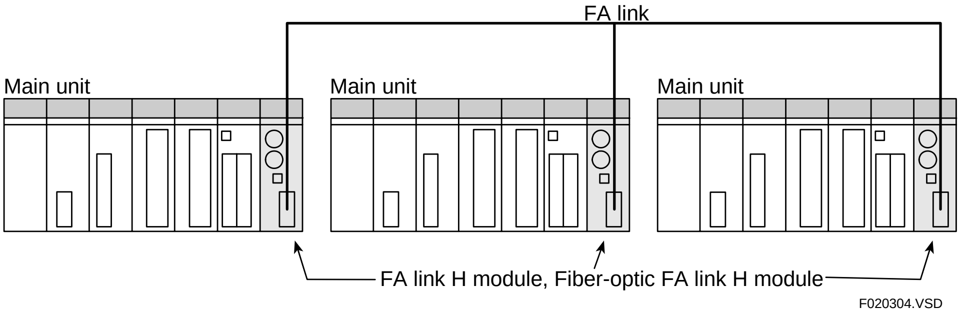


Figure A2.3.3 Example of FA Link System

A2.3.4 FL-net System

An FL-net system is based on FL-net, which is an open network for connecting various programmable controllers (PLC), computerized numerical controllers (CNC) and other factory automation (FA) controllers (including personal computers (PC)) from multiple vendors.

In FA-M3, an FL-net system is configured using FL-net interface modules.

SEE ALSO

For details on FL-net, see "FL-net (OPCN-2) Interface Module User's Manual" (IM34M6H32-02E).

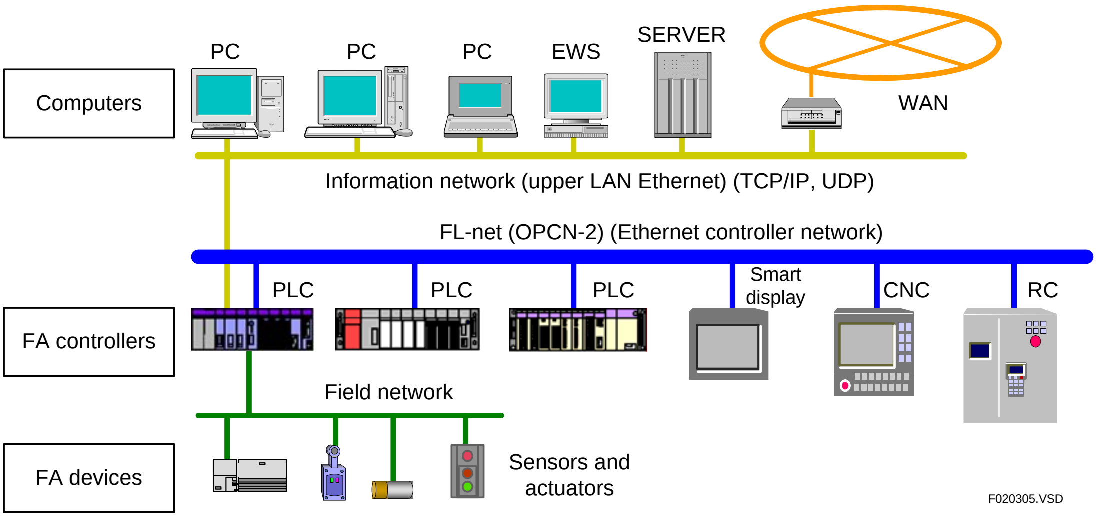


Figure A2.3.4 FL-net System

A2.3.5 LAN System

A LAN system refers to a system for connecting the FA-M3 to PCs or monitors using Ethernet. Remote programming service, socket communications and FTP are carried out via Ethernet.

In FA-M3, there are two ways to configure a LAN system, namely, using the built-in Ethernet function of the module and using the Ethernet function of an Ethernet Interface Module.

SEE ALSO

- For details on the Ethernet function of the module, see "Sequence CPU – Network Functions User's Manual (for F3SP66-4S, F3SP67-6S)" (IM34M6P14-02E).
- For details on Ethernet Interface Module, see "Ethernet Interface Module User's Manual" (IM34M6H24-01E).

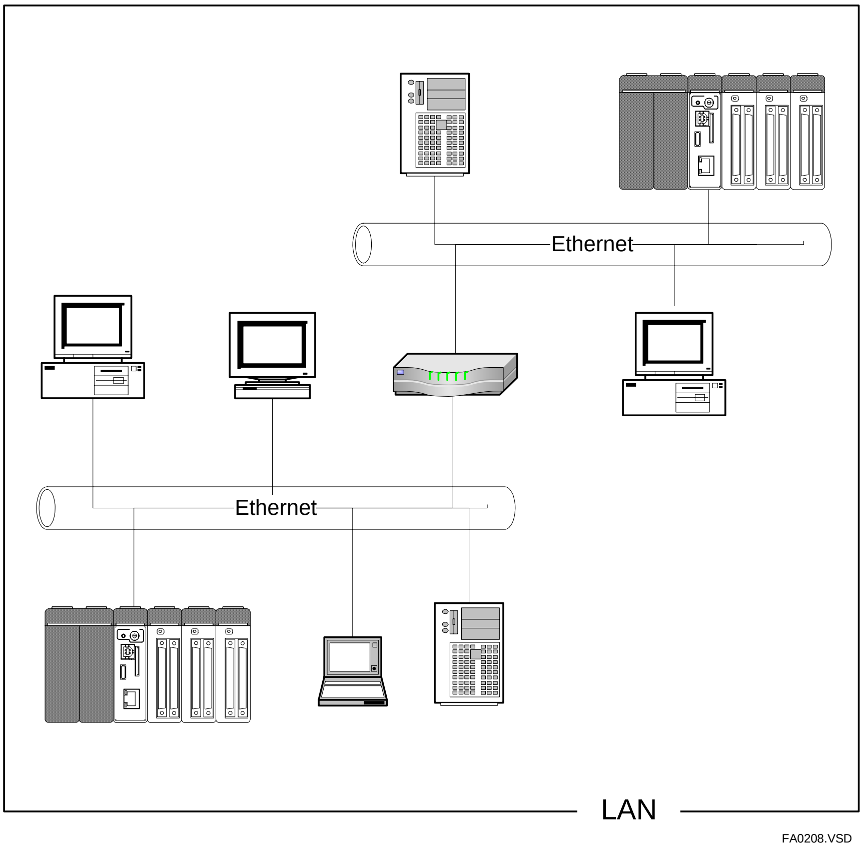


Figure A2.3.5 LAN System

A2.4 Programming Tool

The FA-M3 programming tool WideField2, or simply WideField2, is available as a programming tool for the F3SP66-4S and F3SP67-6S sequence CPU modules.

A2.4.1 WideField2

Table A2.4.1 WideField2

Description	Software Model	Compatible Sequence CPU Modules
FA- M3 Programming Tool WideField2	SF620- ECW	F3SP05 F3SP08 F3SP21 F3SP25 F3SP35 F3SP28 F3SP38 F3SP53 F3SP58 F3SP59 F3FP36 F3SP66 F3SP67

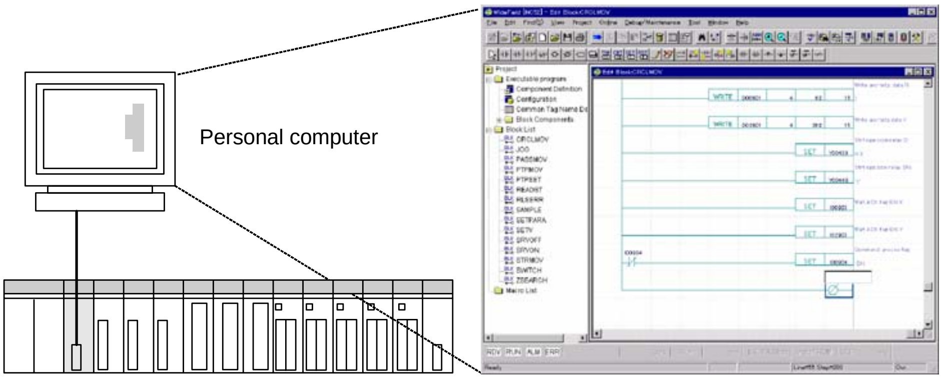


Figure A2.4.1 WideField2

■ Object Ladder

WideField2 defines “blocks” and “instruction macros” that compose a ladder program as “objects,” a term commonly used in the computing world. All objects are responsible for their provided functions and have a high degree of independence. Consequently, the language offers higher productivity, better maintainability and more effective program reuse, as compared to a structured programming language.

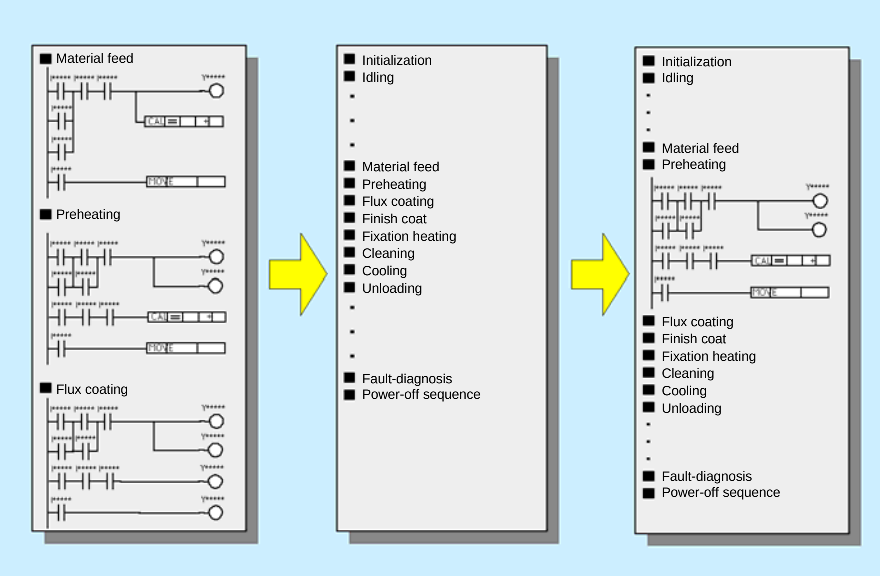
■ Features

● Componentization

With componentization, blocks can be reused as complete components. Devices that are used within a block are defined separately from the main program. Blocks can be easily recombined without undesirable duplicate use of the same device. Macro functions can also be turned into components.

● Index View

You can display an outline view of a large program by “hiding” non-required details. This enables more efficient debugging.

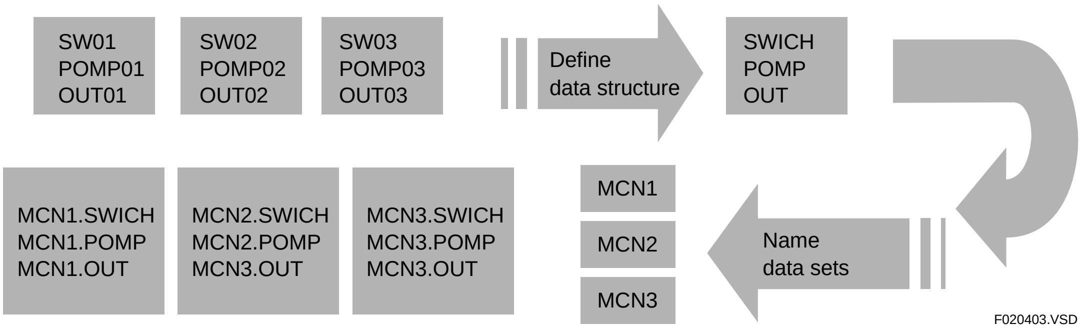


F020402.VSD

Figure A2.4.2 Index View

● Group Tag Names

You can group individual tag names into a group tag name to enable definition of data sets.



F020403.VSD

Figure A2.4.3 Group Tag Name

● Easy Data Exchange with Windows-based Applications

You can select data items such as device names and comments on a Microsoft Excel screen and copy them to WideField2 (drag-and-drop function). You can also copy ladder circuits in WideField2 to Windows applications such as Microsoft Word.

A3. Basic Sequence CPU Module Operations

This chapter describes the basic operation modes of the sequence CPU module and add-on CPU modules, as well as their methods of program execution.

A3.1 Operation Modes of Sequence CPU Module

The sequence CPU module has three operation modes: Run, Debug and Stop. In addition, it has a Shutdown-in-progress transition state, which is an intermediate state when switching from Run mode to Stop mode, or switching from Debug mode to Stop mode.

The LED indicator combination for each operation mode is shown in the table below.

Table A3.1.1 LED Indicator Combinations Based on the Operation Mode

Operation Mode LED Indicator	Run	Debug	Shutdown	Stop
RDY	○	○	○	○
RUN	○	○	☆	●
ALM	●	●	●	●
ERR	●	●	●	●

Note : - ○ =Lit
- ●＝Off
- ☆＝Blinks
- The status of the ALM LED and ERR LED depends on failure (error) events.

■ Run Mode

The Run mode is a state in which the sequence CPU module is running a program, and is used for actual system operation.

In Run mode, the **RDY** and **RUN** LEDs are lit.

■ Debug Mode

The Debug mode is used when debugging and tuning programs.

You can execute programs in the same way as with the Run mode. In Debug mode, you can use debugging functions, such as forced SET/RESET instructions and online edit, through WideField2. These functions, however, affect the scan time. Disable the functions at the end of debugging and tuning, and set the CPU to Run mode.

In Debug mode, the **RDY** and **RUN** LED indicators turn on.

■ Stop Mode

The Stop mode is a state in which the sequence CPU module stops program execution.

In this mode, you can remove programs and clear devices, in addition to using forced SET/RESET instructions, online editing and debug operation. In this mode, the **RUN** LED indicator turns off.

The external outputs being generated by the program are set to ON (hold) or OFF (reset), according to the setting of the configuration item "Output When Stopped" of "DIO Setup". By default, all external outputs are set to OFF.

■ Shutdown-in-progress

Shutdown-in-progress is a state in which the sequence CPU module makes a transition from Run mode to Stop mode, or from Debug to Stop mode. In this state, the **RUN** LED blinks.

If a file access or network access is in progress when a CPU receives a directive to switch to Stop mode, the CPU enters Shutdown-in-progress state to perform close processing. Shutdown-in-progress normally lasts a few seconds to a few minutes. Further switching of operating mode while shutdown is in progress is prohibited.

The external outputs being generated by the program are set to ON (hold) or OFF (reset), according to the setting of the configuration item "Output When Stopped" of "DIO Setup". By default, all external outputs are set to OFF.

A3.2 Operation at Power-on/off

This section describes the operations when power is turned off or turned on.

A3.2.1 Operation at Power-on

When the power is turned on, the CPU performs initialization to get ready for program execution.

During initialization, the CPU performs I/O collation and instruction interpretation to check whether its hardware and programs are normal.

If no error is detected, the CPU executes a program from its beginning.

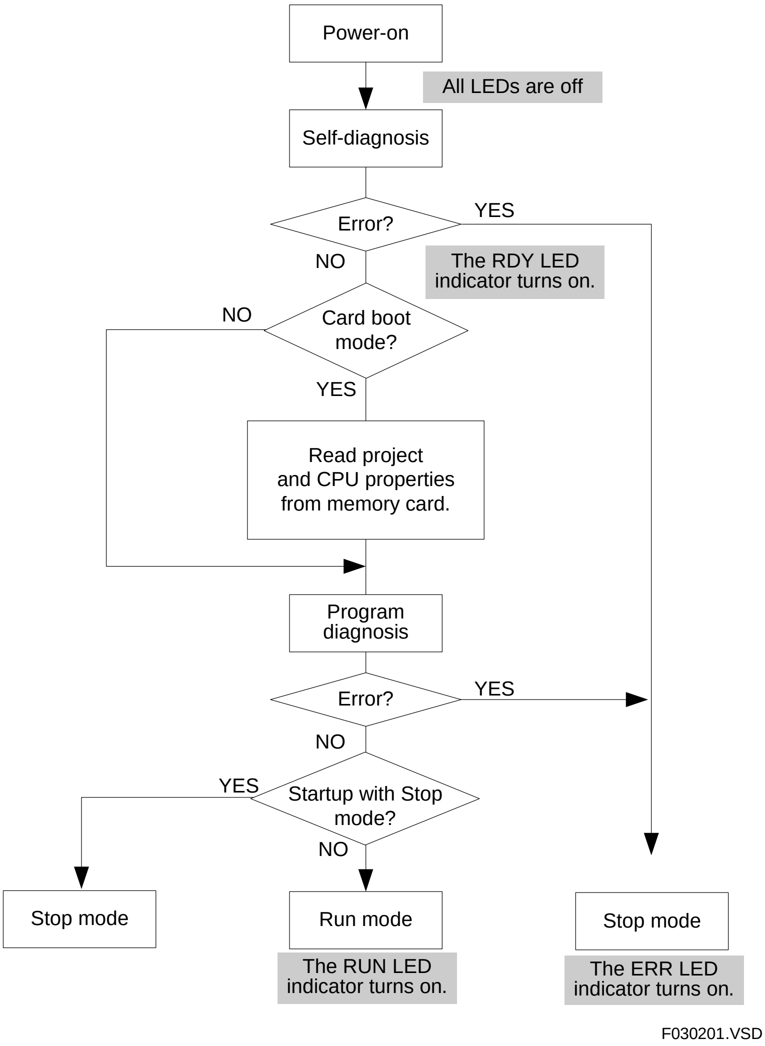


Figure A3.2.1 Operation at Power-on

A3.2.2 Operation at Power-off

When the power is turned off, the sequence CPU module records the date and time in its error log file and stops system operation.

TIP

The error log function saves to an error log file information such as time of occurrence and type of error when a system error occurs or when the power is turned on or turned off.

SEE ALSO

For details on error log (system log), see Chapter A8.1, “Self Diagnosis” and “FA-M3 Programming Tool WideField2 User's Manual” (IM34M6Q15-01E). Error log and system log refer to the same function.

A3.3 Operation in Case of Momentary or Complete Power Failure

This section describes the CPU operation in case of momentary power failure, and how to specify the momentary power failure detection mode, as well as the CPU operation in case of complete power failure, and how to specify the data lock-up range in case of complete power failure.

A3.3.1 Operation in Case of Momentary Power Failure

Two types of power failure detection mode are available for detecting a momentary power failure: the standard mode and the immediate detection mode.

The CPU operates differently in case of a momentary power failure, depending on the type of power failure detection mode selected.

The immediate detection mode can be selected by configuration only when the F3PU10-0□, F3PU16-0N, F3PU20-0□, F3PU26-0N, F3PU30-0□ or F3PU36-0□ power supply module is used.

■ Standard Mode

If a momentary power failure occurs, the sequence CPU module records the date and time in its error log file. The sequence CPU module suspends processing until the power is restored. This causes a delay in the scan time and timer update process.

When the power is restored, the sequence CPU module resumes execution at the point where processing was suspended.

A program can detect a momentary power failure by monitoring a special relay (M195).

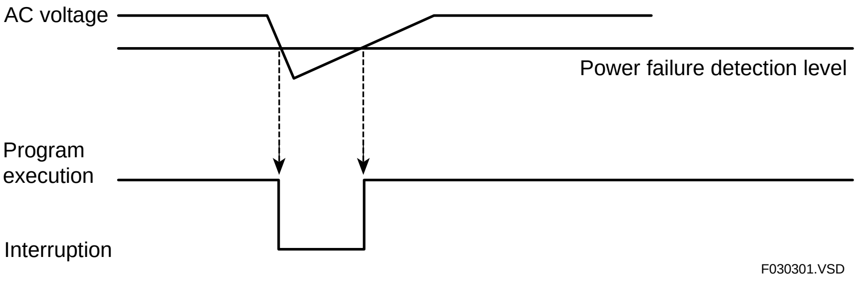


Figure A3.3.1 Operation in Case of Momentary Power Failure

■ Immediate Detection Mode

If a momentary power failure occurs, the sequence CPU module records the date and time in its error log file. The sequence CPU module suspends processing until power is restored. At this point the CPU sets the external outputs generated by a program to OFF, and actuates the FAIL contact.

When power is restored, the sequence CPU module performs a reset-and-start sequence and executes the program from its beginning.

A3.3.2 Momentary Power Failure Detection Mode Setup

This configuration item defines the momentary power failure detection mode.
You can select either the standard mode or the immediate detection mode. The default is the standard mode.

SEE ALSO

For details on each of these modes, see “Hardware Manual” (IM34M6C11-01E).

CAUTION

In a multi-CPU configuration, all CPU modules must be configured with the same momentary power failure detection mode.

A3.3.3 Operation in Case of Complete Power Failure

If a complete power failure occurs, the CPU operates as it does at power off.
You can configure the types and ranges of devices to be latched in case of a complete power failure. This strategy allows the CPU to resume operation from its previous state after power is restored.
When power is restored, the CPU executes the program from its beginning.

TIP

Latching devices at power failure stores device states immediately before a power failure so that a program can continue execution in the same state after power is restored.

A3.3.4 Data Lock-up Range at Power Failure

This configuration item sets the range of devices to be latched in case of a complete power failure.
Specify the starting number and the number of devices for each device type.
The following table shows the default setting and the configurable range of each device type.

Table A3.3.1 Data Lock-up Range at Power Failure of Configuration

Item			Default Value		Configuration Range
			F3SP66	F3SP67	
Extended device configuration	Data Lock-up Range at Power Failure	Internal relay (I)	I0001 to I1024	I0001 to I1024	Configurable on 32-relay basis; contiguous from a starting number*1
		Shared relay (E) Extended shared relay (E)	Non-latching type	Non-latching type	
		Link relay (L)	Non-latching type	Non-latching type	Configurable on 16-relay basis*2
		Timer (T)	Non-latching type (except for continuous timers)	Non-latching type (except for continuous timers)	Configurable for timers/counters; contiguous from a starting number
		Counter (C)	All latched (C0001 to C1024)	All latched (C0001 to C1024)	
		Data register (D)	All latched (D00001 to D16384)	All latched (D00001 to D32768)	Configurable on 2-register basis; contiguous from a starting number*1, *2
		Shared register (R) Extended shared register (R)	Non-latching type	Non-latching type	
		Link register (W)	Non-latching type	Non-latching type	Configurable on 16-register basis*2

*1: If the upper limit of the range of shared relays (E) used is smaller than E2049, the last device number for shared relays (E) is followed by the first device number for extended shared relays (E). Likewise, if the upper limit of shared registers (R) used is smaller than R1025, the last device number for shared registers (R) is followed by the first device number for extended shared registers (R).

*2: The data lock-up range setup for link relay and link register are mapped to contiguous devices starting from their respective starting numbers with the following exceptions:

L/W01024 is followed by L/W10001; L/W11024 is followed by L/W20001;
L/W21024 is followed by L/W30001; L/W31024 is followed by L/W40001;
L/W41024 is followed by L/W50001; L/W51024 is followed by L/W60001;
L/W61024 is followed by L/W70001.

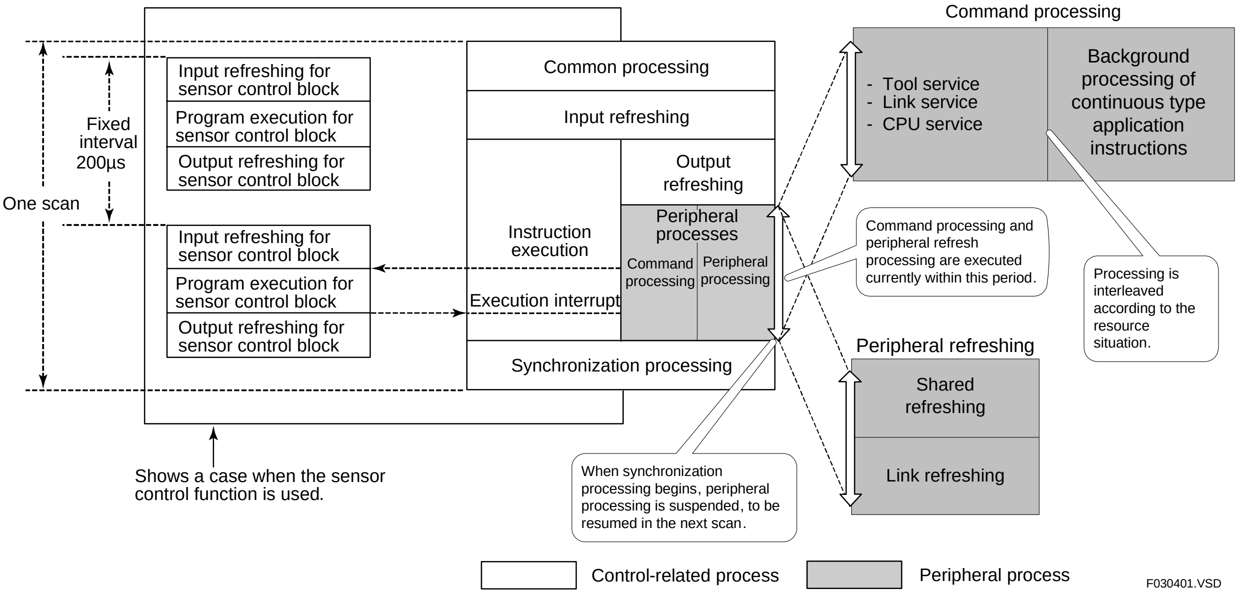
The above rule applies when the number of link relays or registers to be used is defined as 1024.
In the case of 2048, L/Wn2048 is followed by L/Wn0001.
In the case of 8192, L/W08192 is followed by L/W10001.

F030302.VSD

A3.4 Computation Method

This section outlines data computation (scan processing) in the sequence CPU module.

The CPU employs a stored-program iterative computation method.
In this method, a created program is pre-stored in the memory of the sequence CPU module. The sequence CPU executes instructions, one at a time, starting from the first step of the program. After executing the last step in the program, the CPU performs required processing, such as self-diagnosis. It then repeats the instructions from the first step.
Each of these iterative cycles is called “one scan” and the time required for one scan is called a “scan time".
In the case of F3SP66 and F3SP67 CPU modules, the CPU executes instructions and peripheral processes concurrently to perform each scan in a shorter time.
Common processing, instruction execution, input refreshing, output refreshing, and synchronization processing are classified as a system of control-related processes, while tool service, personal computer link service, CPU service, link refreshing, and shared refreshing are classified as a system of peripheral processes. The CPU performs these two kinds of processes concurrently to speed up the control-related processes.



Command processing
One scan
Fixed interval 200µs
Input refreshing for sensor control block
Program execution for sensor control block
Output refreshing for sensor control block
Input refreshing for sensor control block
Program execution for sensor control block
Output refreshing for sensor control block
Common processing
Input refreshing
Instruction execution
Execution interrupt
Output refreshing
Peripheral processes
Command processing
Peripheral processing
Synchronization processing
- Tool service
- Link service
- CPU service
Background processing of continuous type application instructions
Command processing and peripheral refresh processing are executed currently within this period.
Processing is interleaved according to the resource situation.
Peripheral refreshing
Shared refreshing
Link refreshing
Shows a case when the sensor control function is used.
When synchronization processing begins, peripheral processing is suspended, to be resumed in the next scan.
Control-related process
Peripheral process
F030401.VSD

Figure A3.4.1 Computation Method

SEE ALSO

For details on the sensor control function, see Section A6.15, “Sensor Control Function”.

TIP

- Common processing includes self-diagnosis, updating of special relays (M) and special registers (Z), as well as updating of timers. The END processing is sometimes known as an END scan.
- The input refreshing process reflects data from input contacts of, say a DI module, to input relays (X).
- The output refreshing process reflects data from output relays (Y) to output contacts of, say a DO module.
- The synchronization process synchronizes the system of control-related processes with the system of peripheral processes. In particular, it reflects data to the link refresh and shared refresh (intra-PLC communication) devices.

● System of Control-related Processes

This system performs basic operations of the sequence CPU module, such as instruction execution and I/O refreshing.

Execution of the system of control-related processes is called one scan, and the execution time required by the system is usually called the scan time.

● System of Peripheral Processes

This system supports the programming tool WideField2 and performs communication between the CPU module and a personal computer or an FA link module.

The system of peripheral processes is concurrent with and independent of the system of control-related processes. Therefore, neither the number of modules connected nor the content of each peripheral process affects the operation of the system of control-related processes.

The system of peripheral processes is further classified into command processes and peripheral refresh processes, which are processed concurrently. Command processing performs tool service, higher-level link service and CPU service, while peripheral refresh processing performs link refreshing and shared refreshing. In addition, execution of continuous-type application instruction background processes is interleaved with command process execution, taking into consideration the states of resources.

● Synchronization between Systems of Control-related Processes and Peripheral Processes

The system of peripheral processes executes concurrent with and independent of the system of control-related processes. For processes related to operation control (e.g., run or stop) or processes requiring simultaneity of data, however, the CPU synchronizes these two systems using a synchronization process included in the system of control-related processes.

The time required for the synchronization process varies depending on its content. Using debugging functions, such as online editing, affects the scan time.

CAUTION

If the ratio of the instruction execution time to the scan time is too small, insufficient time may be allocated for executing the system of peripheral processes. Consequently, the responses of link refreshing, shared refreshing, tool service, higher-level link service and CPU service will become extremely slow. If this happens, use a constant scan with an interval somewhat longer than the normal scan time, or set up the peripheral processing time to secure sufficient time for executing the system of peripheral processes.

A3.5 Method of Executing Peripheral Processes

Peripheral processes are executed concurrently with instructions in a program. When the execution of program instructions is completed, any peripheral process is interrupted until the next scan, in order to prevent the process from affecting the scan time. This means the peripheral processing time is affected by the program execution time.

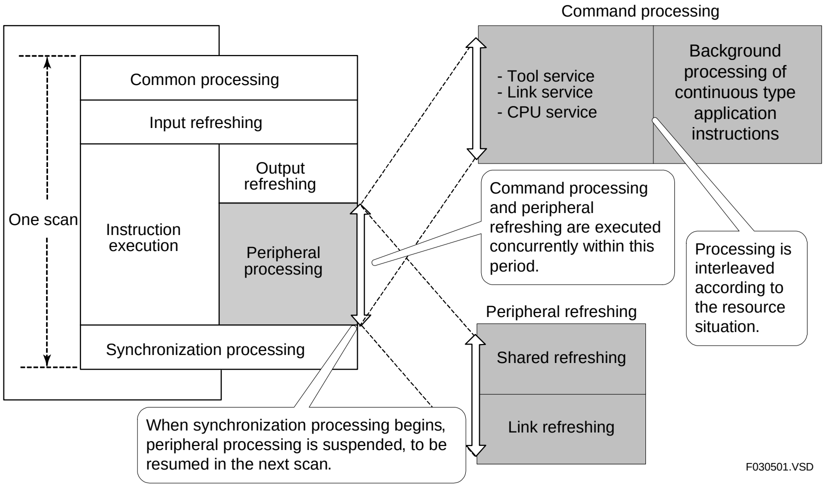

Command processing
One scan
Common processing
Input refreshing
Instruction execution
Output refreshing
Peripheral processing
Synchronization processing
- Tool service
- Link service
- CPU service
Background processing of continuous type application instructions
Command processing and peripheral refreshing are executed concurrently within this period.
Processing is interleaved according to the resource situation.
Peripheral refreshing
Shared refreshing
Link refreshing
When synchronization processing begins, peripheral processing is suspended, to be resumed in the next scan.
F030501.VSD

Figure A3.5.1 Peripheral Processing

■ Specifying Peripheral Processing Time

You can define the minimum peripheral processing time. Use this configuration item when the instruction execution time is so short that insufficient time is allocated to peripheral processes. To secure enough time, the scan time is lengthened, or the delays of shared refreshing, link refreshing and command processing included in the peripheral processes are shortened. If at the end of instruction execution, the actual peripheral processing time is less than the defined peripheral processing time, peripheral processing will be prolonged until the expiry of the defined time. In this case, the scan will also be prolonged accordingly. Beware that the CPU ignores the defined peripheral processing time if a constant scan time is defined.

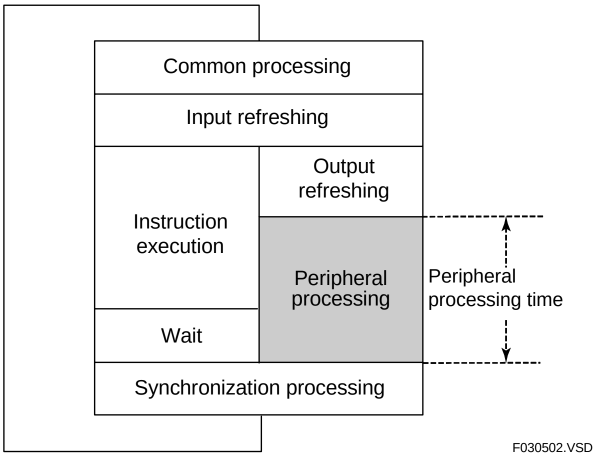


Figure A3.5.2 Peripheral Processing Time

The configurable range is from 0.1 ms to 190 ms, on 0.1 ms basis. If no peripheral processing time is defined, the CPU operates with a peripheral processing time of 0.1 ms by default.

A3.6 Method of I/O Processing

This section describes how I/O processing is performed, I/O response delay, as well as I/O processing in a multi-CPU system.

A3.6.1 Method of I/O Processing

As the method of I/O processing, the CPU uses batch refreshing.

In this method, the sequence CPU module acquires all data changes in the input module into the input relay (X) area of the CPU's data memory before executing each scan.

The sequence CPU module uses data contained in this area when performing computations.

Computation results are output to the output relay (Y) area of the CPU's data memory each time a computation is performed. The results are sent to the output module, collectively and concurrently with the execution of instructions in the next scan.

Only modules that are installed in the system and configured in DIO setup as "Used" will have their input and output refreshed. No error occurs even if a program tries to access input or output relays of a module which is not installed or a module which is configured in DIO setup as "Not used".

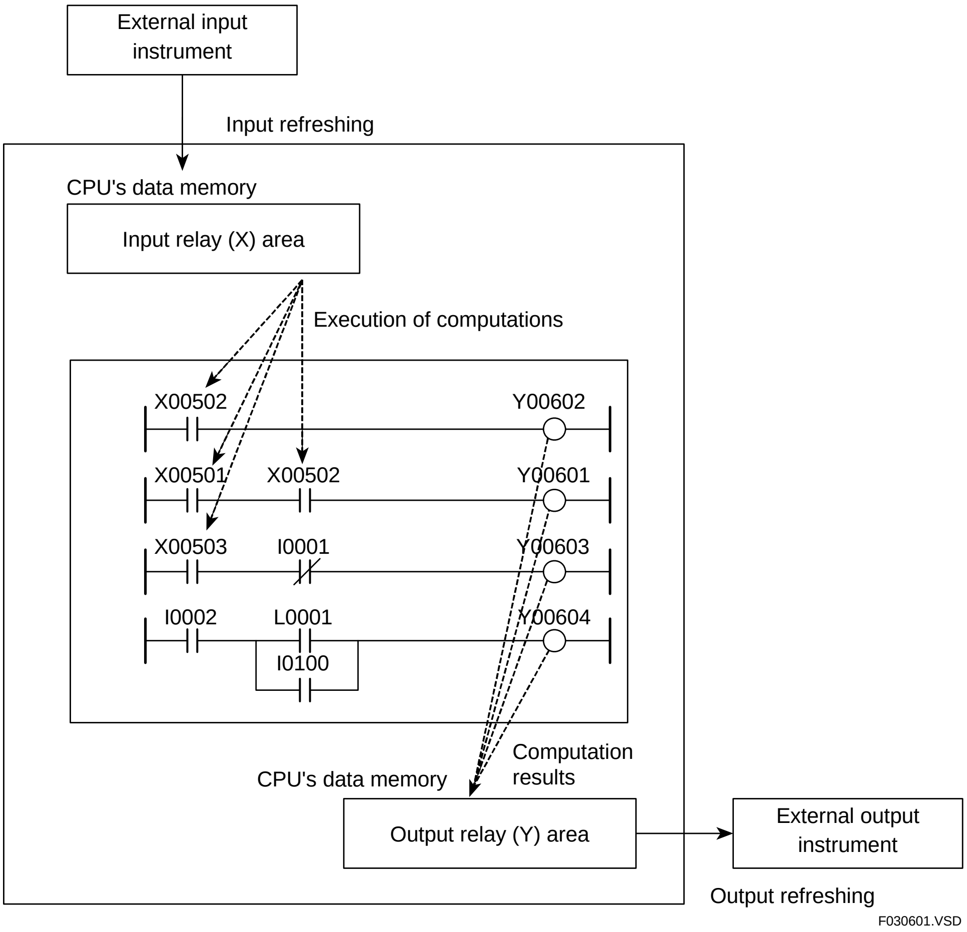


Figure A3.6.1 Method of I/O Processing

A3.6.2 Response Delay

The maximum response delay of the output module against a change in the input module is two scans.

SEE ALSO

For details on I/O response time related to response delay, see Chapter A7, “I/O Response Time Based on Scan Time".

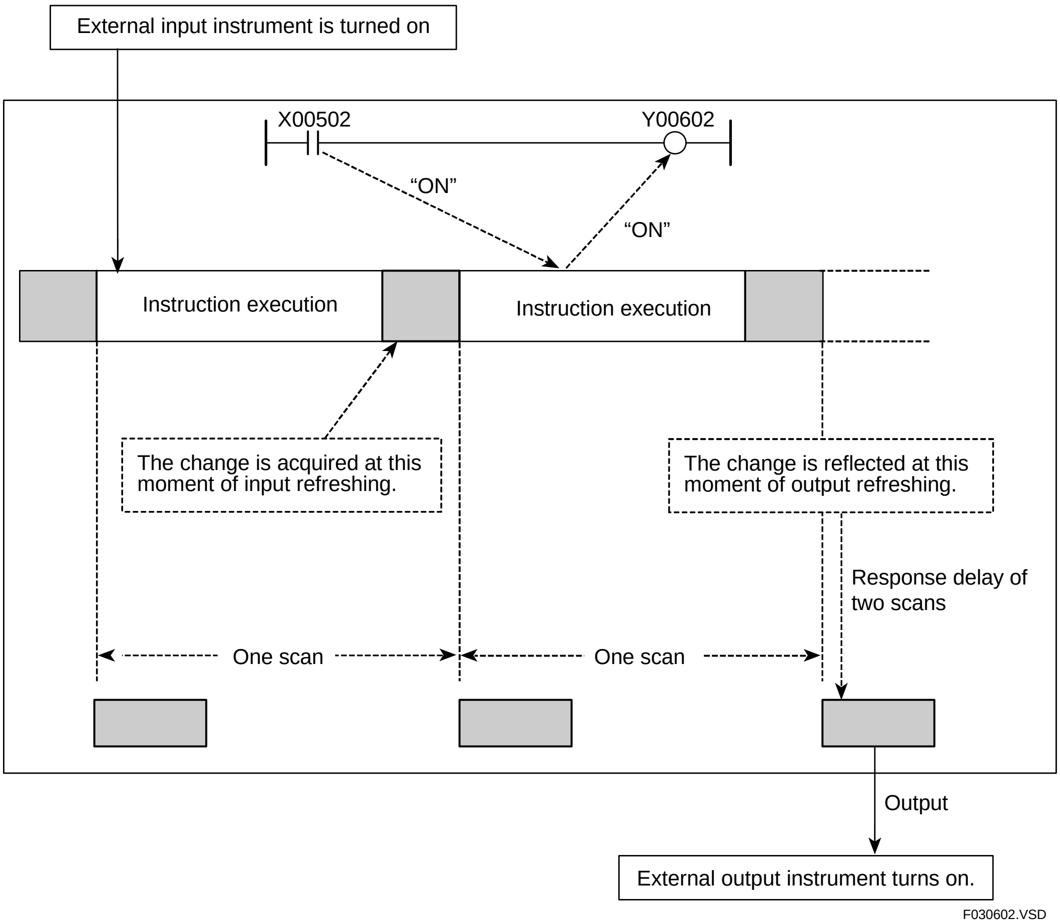

External input instrument is turned on
X00502
Y00602
“ON”
“ON”
Instruction execution
Instruction execution
The change is acquired at this moment of input refreshing.
The change is reflected at this moment of output refreshing.
Response delay of two scans
One scan
One scan
Output
External output instrument turns on.
F030602.VSD

Figure A3.6.2 Response Delay

A3.6.3 I/O Processing in Multi-CPU System

The sequence CPU module performs refreshing when the configuration item “Terminal Usage” of “DIO Setup” is set to “Used” or “Use with SCB”.

In the configuration, define the terminals to be refreshed for each sequence CPU module. Each sequence CPU module refreshes the terminals independently according to the definition.

Be careful not to configure the CPUs so that more than one CPU refreshes the same terminal of the output module. Otherwise, the resultant output of the output module will be indefinite.

SEE ALSO

For details on the parameters set for I/O modules and their limitation of use, see Subsection A2.2.2, “Handling I/O Modules in Multi-CPU System” and subsection A4.1.4, “Configuring DIO Modules".

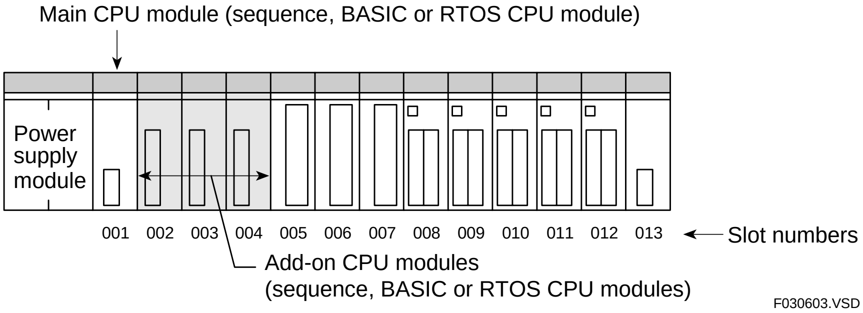

Main CPU module (sequence, BASIC or RTOS CPU module)
Power supply module
001
002
003
004
005
006
007
008
009
010
011
012
013
Slot numbers
Add-on CPU modules (sequence, BASIC or RTOS CPU modules)
F030603.VSD

Figure A3.6.3 Example of Multi-CPU System

A3.7 Method of Executing Commands from WideField2

Commands from WideField2 are executed by the tool service. These commands include downloading and uploading of programs, as well as monitoring of devices.

A3.7.1 Tool Service

The tool service executes commands sent from the FA-M3 programming tool WideField2.

Since the tool service runs concurrently to the execution of instructions, it does not affect the scan time.

The CPU does not execute the tool service if there is no command to be processed.

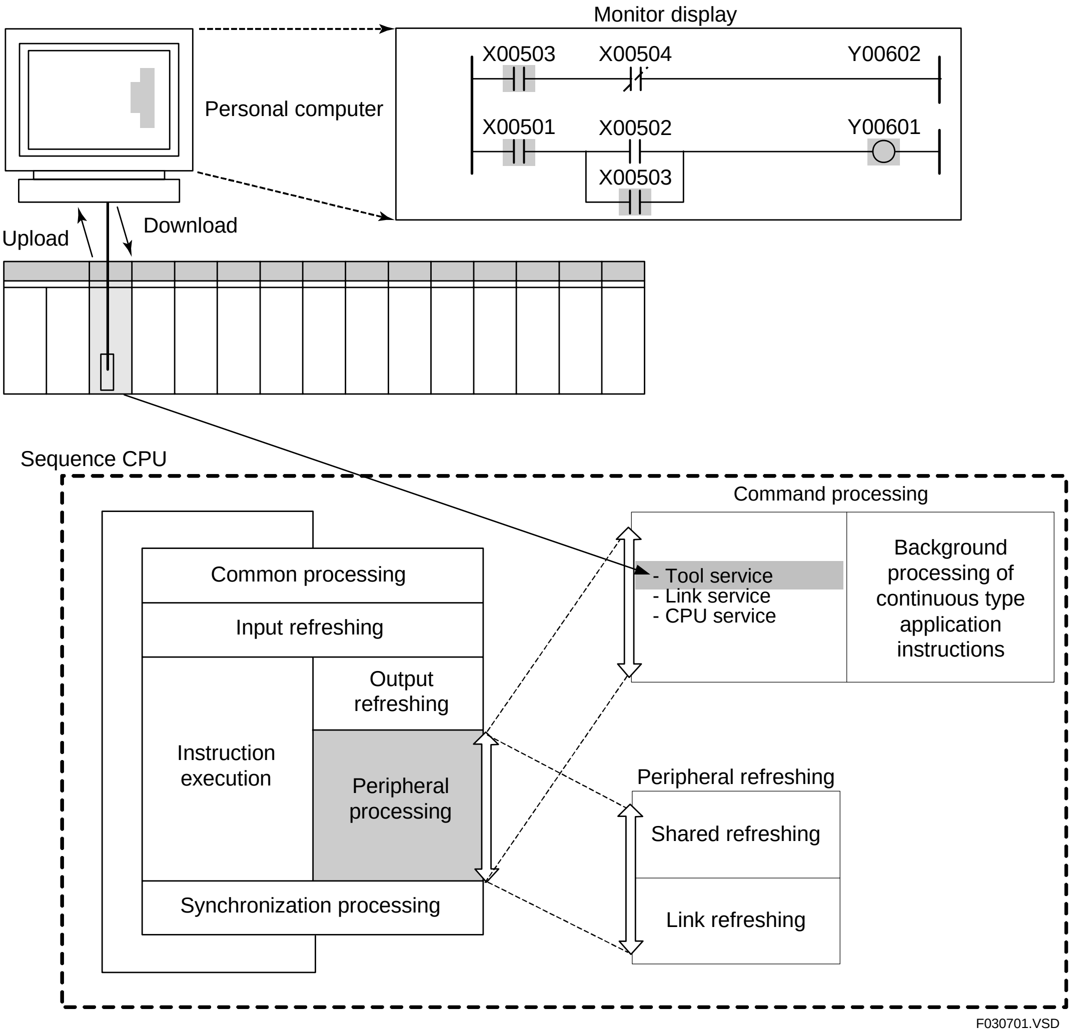


Figure A3.7.1 Execution of Commands Sent from the WideField2

A3.8 Method of Executing Commands through Personal Computer Link

The CPU uses the higher-level link service to execute commands sent through a personal computer link. These commands include downloading and uploading programs and reading from and writing to devices.

A3.8.1 Higher-level Link Service

The higher-level link service executes commands sent from a personal computer or a monitor connected to the personal computer link module.

Since the higher-level link service runs concurrently to the execution of instructions, it does not affect the scan time.

The CPU does not execute the higher-level link service if there is no command to be processed.

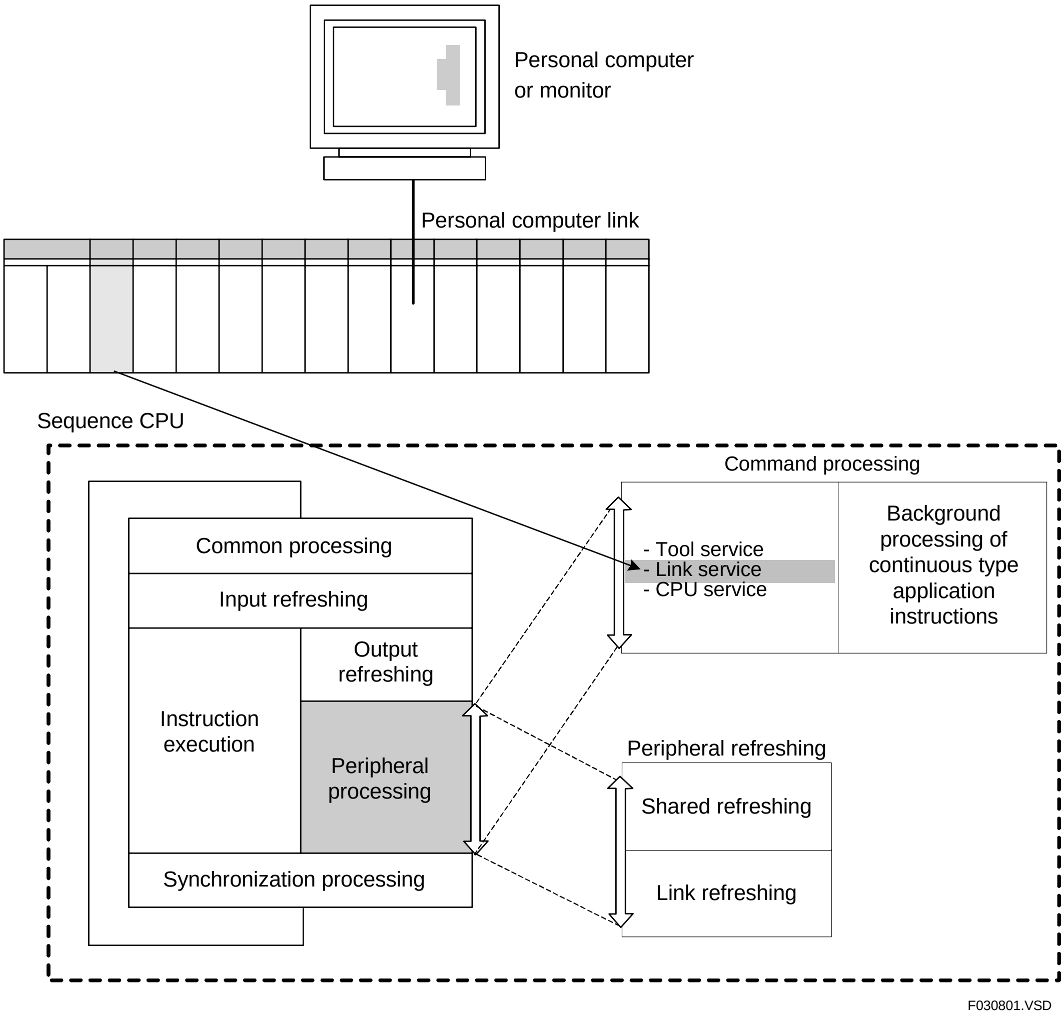


Figure A3.8.1 Execution of Personal Computer Link Commands

A3.9 Method of CPU-to-CPU Data Communication

CPU-to-CPU communication in a multi-CPU system configured using add-on CPUs is carried out using a shared data communication method and CPU services. Communications between sequence CPU modules is carried out using shared data communications, while communications between sequence CPU modules and other types of CPU modules, such as BASIC CPU modules, is carried out using shared data communications or CPU services.
This section describes methods for updating shared data, configuration of shared refreshing and the CPU service.

A3.9.1 Method of Updating Shared Data

CPU-to-CPU data exchange in a multi-CPU system configured using multiple CPU modules is carried out through shared relays (E), extended shared relays (E), shared registers (R) and extended shared registers (R). Hereafter, shared relays (E), extended shared relays (E), shared registers (R) and extended shared registers (R) are collectively referred to as shared devices.

You must configure in advance the range of shared devices to be used for each installed CPU. The configuration of a CPU module must tally with the configuration of the other CPU modules. You can both read from and write to shared devices within the CPU module's own area. However, you can only read from shared devices within the areas of the other CPU modules.

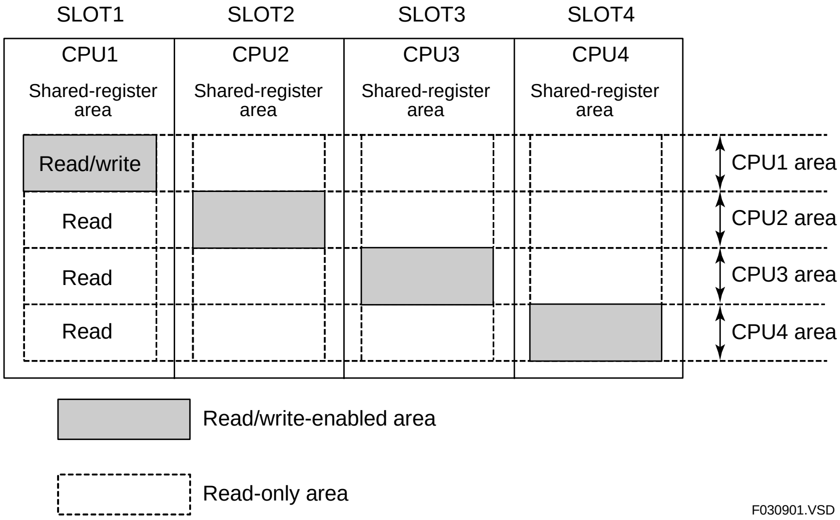


Figure A3.9.1 Example of Configuring Shared Registers (R)

The figure below shows an example of shared refreshing carried out between a sequence CPU module and an add-on CPU module. In this example, shared relays (E) and shared registers (R) are allocated as shown below.

- Sequence CPU module: (Slot1 CPU)	Shared relays (E) = E0001 to E0512 Shared registers (R) = R0001 to R0256
- Add-on CPU module: (Slot2 CPU)	Shared relays (E) = E0513 to E1024 Shared registers (R) = R0257 to R0512

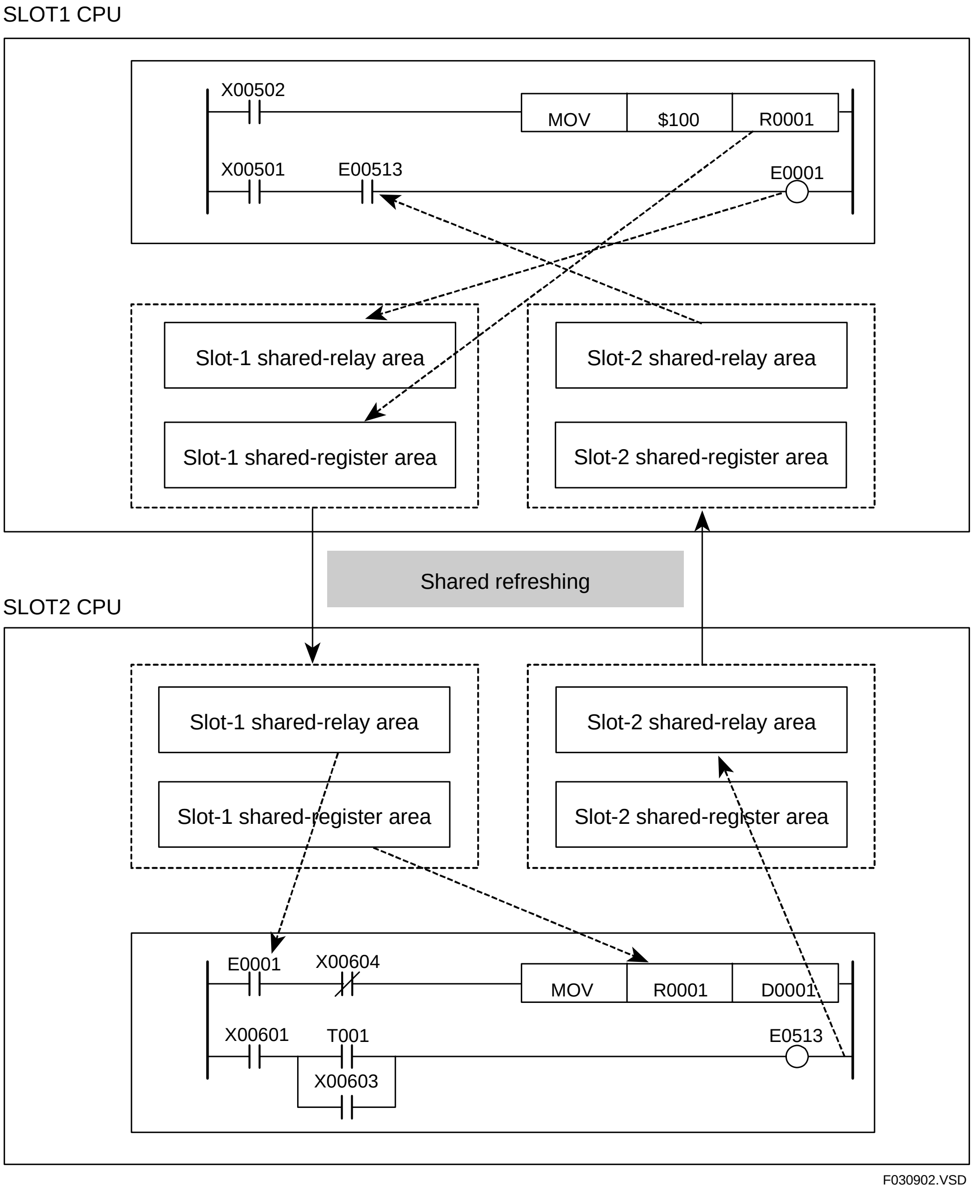


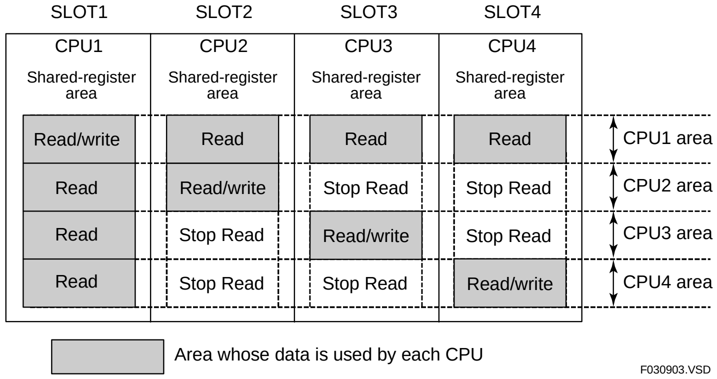
Figure A3.9.2 Shared Refreshing

A3.9.2 Configuration of Shared Refreshing

This subsection describes the shared refreshing range (partial disabling of refreshing), simultaneity of shared refreshed data, and shared refreshing mode (changing to control-related process).

■ Shared Refreshing Range (Partial Disabling of Refreshing)

Using configuration, you can disable shared refreshing for selected device types of shared relay (E), extended shared relay (E), shared register (R) and extended shared register (R) of each CPU module. Disabling shared refreshing between CPU modules that do not need to exchange data shortens the overall shared refreshing interval.



	SLOT1 CPU1 Shared-register area	SLOT2 CPU2 Shared-register area	SLOT3 CPU3 Shared-register area	SLOT4 CPU4 Shared-register area
CPU1 area	Read/write	Read	Read	Read
CPU2 area	Read	Read/write	Stop Read	Stop Read
CPU3 area	Read	Stop Read	Read/write	Stop Read
CPU4 area	Read	Stop Read	Stop Read	Read/write

▨ Area whose data is used by each CPU

F030903.VSD

Figure A3.9.3 Example of Shared Refreshing Configuration

In the example shown in the figure above, if data need not be shared among add-on CPU modules, the refreshing intervals of CPU2, CPU3 and CPU4 are shortened if “CPU3 and CPU4,” “CPU2 and CPU4” and “CPU2 and CPU3” respectively are excluded from shared refreshing.

TIP

If you exclude a CPU module from shared refreshing, its scan time shortens because data updating done by its synchronization process is disabled. However, this prohibits sharing of data in all areas of the other CPU modules.

■ Simultaneity of Shared Refreshed Data

You can specify by configuration whether to maintain simultaneity of shared refreshed data. If you select “Simultaneous” for this configuration item when a sequence CPU module (F3SP28, F3SP38, F3SP53, F3SP58, F3SP59, F3SP66 or F3SP67 module) is combined with any number of F3SP28, F3SP38, F3SP53, FS3P58, F3SP59, F3SP66 and F3SP67 modules defined as add-on CPU modules, simultaneity of shared refreshed data is guaranteed by the unit of shared devices (shared relays (E) and registers (R), or extended shared relays (E) and registers (R)) being refreshed.

If any of the F3SP28, F3SP38, F3SP53, F3SP58, F3SP59, F3SP66 and F3SP67 modules is combined with any of the F3SP21, F3SP25, F3SP35 and F3BP□□ modules, simultaneity of data is not guaranteed irrespective of the configuration settings.

The “Non-simultaneous” option of this configuration item is intended for compatibility with the F3SP21, F3SP25 and F3SP35 modules. Select this option when replacing these modules with the F3SP28, F3SP38, F3SP53, F3SP58, F3SP59, F3SP66 or F3SP67 modules.

■ Shared Refreshing Mode

You can change by configuration the mode of shared refreshing, which updates the data of relays (E) and registers (R) shared with other CPUs, so that it runs as a control-related process. Run shared refreshing as a peripheral process if a short scan time is important. Alternatively, run it as a control-related process if the speed of exchanging shared data is important.

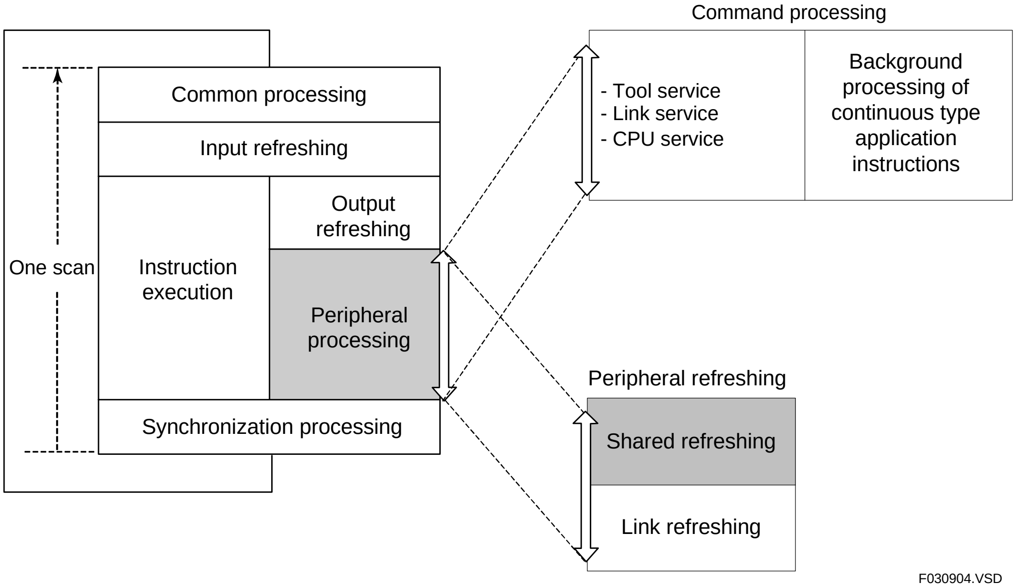


Figure A3.9.4 Executing Shared Refreshing as a Peripheral Process

Executing shared refreshing as a peripheral process reduces its effects on scanning.

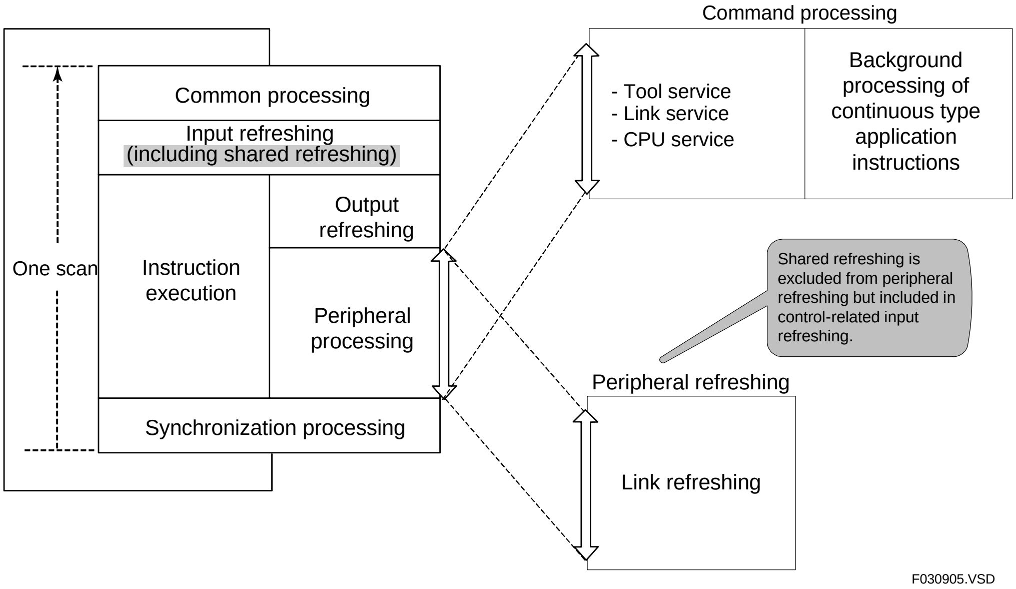

Command processing
Common processing
Input refreshing (including shared refreshing)
Output refreshing
One scan
Instruction execution
Peripheral processing
Synchronization processing
- Tool service
- Link service
- CPU service
Background processing of continuous type application instructions
Shared refreshing is excluded from peripheral refreshing but included in control-related input refreshing.
Peripheral refreshing
Link refreshing
F030905.VSD

Figure A3.9.5 Executing Shared Refreshing as a Control-related Process

If you execute shared refreshing as a control-related process, the scan time lengthens. However, this ensures that shared refreshing is not affected by the link refreshing or command processing time.

TIP

- Sequence of Shared Refreshing
  For a main CPU, shared refreshing is executed in the order of CPU2's shared relays(E)/registers(R), CPU2's extended shared relays(E)/registers(R), CPU3's shared relays(E)/registers(R), CPU3's extended shared relays(E)/registers(R), CPU4's shared relays(E)/registers(R), and CPU4's extended shared relays(E)/registers(R).
- When the configuration item "Shared Refreshing Mode" is set to "Peripheral Process"
  Each single scan of peripheral processing refreshes CPUn's shared relays (E)/registers (R) or extended shared relays (E) /registers (R). The data that has been read is reflected in device areas during the synchronization process occurring after the completion of shared refreshing. Note however that if the configuration item "Shared Refreshing Data" is set to "Simultaneous," refreshing may be delayed by as much as three scans of peripheral processing due to the need for synchronization with the CPUn.
- When the configuration item "Shared Refreshing Mode" is set to "Control Process"
  Each single scan of control-related processing refreshes the CPUn's shared relays(E)/registers(R) or extended shared relays(E)/registers(R). Note however that when the configuration item "Shared Refreshing Data" is set to "Simultaneous," refreshing may be delayed by as much as three scans due to the need for synchronization with the CPUn.
- Reference to a CPU's Own Write Area
  You can read data in a CPU's write area from the other CPUs. That is, you can read the data alternately from shared relays(E)/registers(R) and from extended shared relays(E)/registers(R) in that area during the synchronization process of each scan. Note however that when the configuration item "Shared Refreshing Data" is set to "Simultaneous" for any of the other CPUs, refreshing may be delayed by as much as the longest of those CPUs' scans due to the need for synchronization with the slowest CPU.

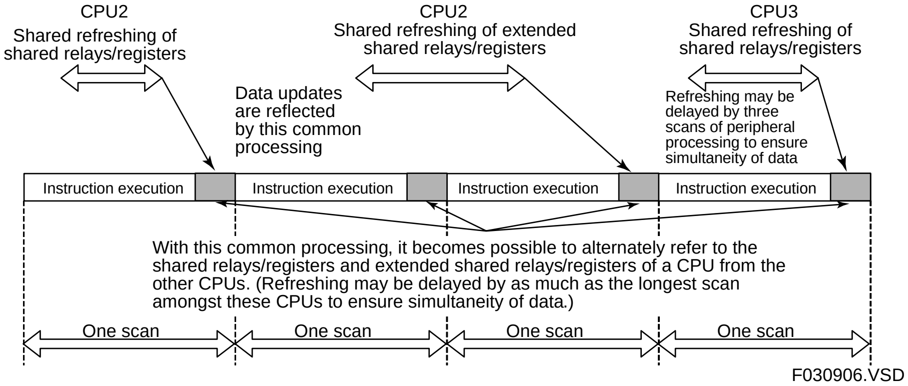

CPU2
Shared refreshing of shared relays/registers
CPU2
Shared refreshing of extended shared relays/registers
CPU3
Shared refreshing of shared relays/registers
Data updates are reflected by this common processing
Refreshing may be delayed by three scans of peripheral processing to ensure simultaneity of data
Instruction execution
Instruction execution
Instruction execution
Instruction execution
With this common processing, it becomes possible to alternately refer to the shared relays/registers and extended shared relays/registers of a CPU from the other CPUs. (Refreshing may be delayed by as much as the longest scan amongst these CPUs to ensure simultaneity of data.)
One scan
One scan
One scan
One scan
F030906.VSD

Figure A3.9.6 Shared Refreshing as a Peripheral Process

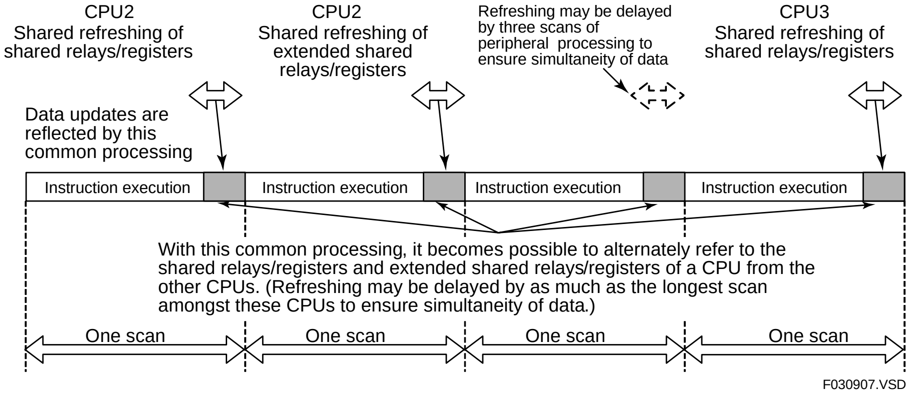

CPU2
Shared refreshing of shared relays/registers
CPU2
Shared refreshing of extended shared relays/registers
Refreshing may be delayed by three scans of peripheral processing to ensure simultaneity of data
CPU3
Shared refreshing of shared relays/registers
Data updates are reflected by this common processing
Instruction execution
Instruction execution
Instruction execution
Instruction execution
With this common processing, it becomes possible to alternately refer to the shared relays/registers and extended shared relays/registers of a CPU from the other CPUs. (Refreshing may be delayed by as much as the longest scan amongst these CPUs to ensure simultaneity of data.)
One scan
One scan
One scan
One scan
F030907.VSD

Figure A3.9.7 Shared Refreshing as a Control-related Process

SEE ALSO

For details on how shared refreshing affects the scan time, see Chapter A7, "I/O Response Time Based on Scan Time".

A3.9.3 CPU Service

CPU service exchanges data and process commands between the sequence CPU and a BASIC CPU.

CPU service is processed concurrently with instruction execution so it does not affect the scan time. The sequence CPU does not execute the CPU service unless it receives a command (ENTER, OUTPUT, etc.) to be processed from a BASIC CPU.

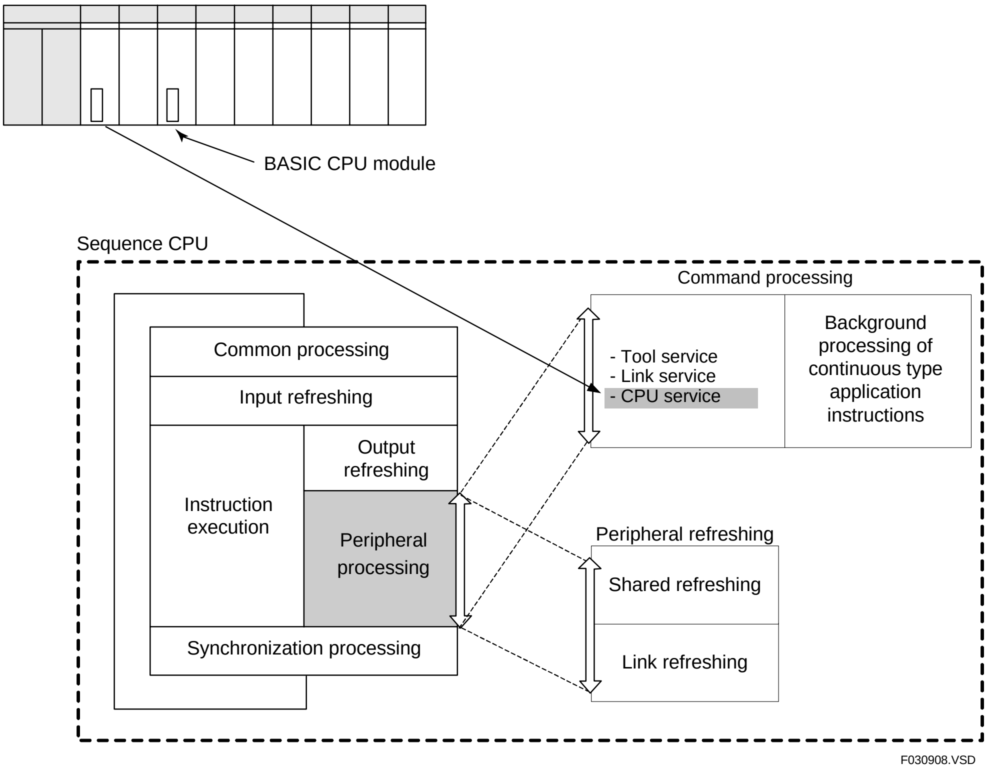


Figure A3.9.8 CPU Service

A3.10 Method of Link Data Updating

This section describes methods of link data updating and link refreshing for FA link systems and FL-net systems.

A3.10.1 Link Data Updating

Link data updating is a process of exchanging data with sequence CPU modules in remote stations through link relays (L) and registers (W).

You must configure in advance the ranges of link relays (L) and registers (W) to which data is written in the local and remote stations.

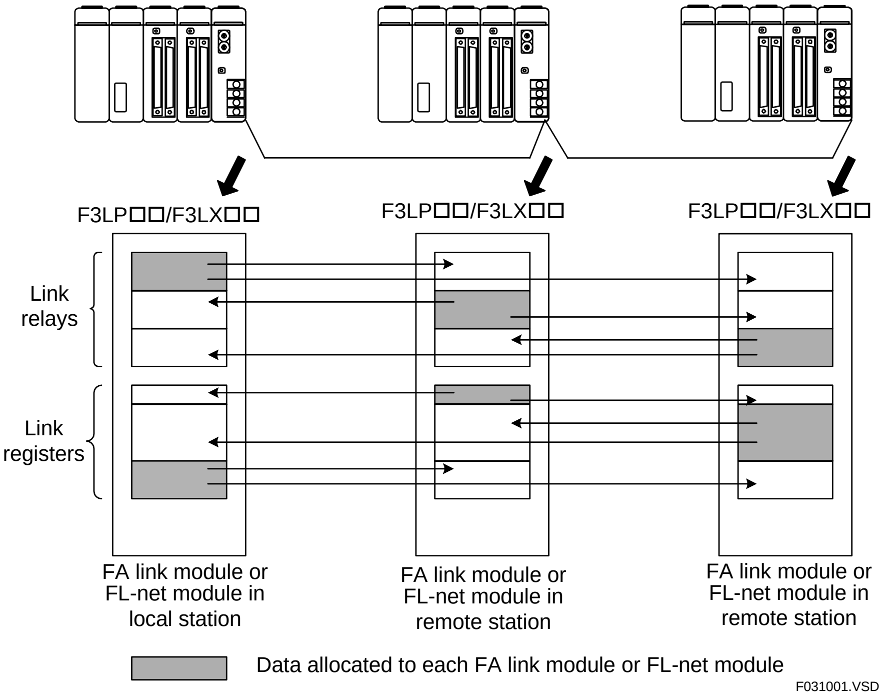

F3LP□□/F3LX□□
F3LP□□/F3LX□□
F3LP□□/F3LX□□
Link relays
Link registers
FA link module or FL-net module in local station
FA link module or FL-net module in remote station
FA link module or FL-net module in remote station
Data allocated to each FA link module or FL-net module
F031001.VSD

Figure A3.10.1 Link Data Updating

SEE ALSO

For details on link data updating and link refreshing, see:

- Section A4.3, “Link Relays (L) and Link Registers (W)” of this manual;
- “FL-net (OPCN-2) Interface Module User's Manual” (IM34M6H32-02E);
- “FA Link H Module F3LP02-0N, Fiber-optic FA Link H Modules F3LP12-0N User's Manual” (IM34M6H43-01E).

A3.10.2 Link Refreshing

Link refreshing reads data from or writes data to devices such as link relays (L) and registers (W) of the sequence CPU module via an FA link module or FL-net (OPCN-2) interface module installed in the local unit. It maps the link data in the storage area of the sequence CPU module to those of the FA link module.

The sequence CPU module reads the link data of the FA link module or FL-net (OPCN-2) interface module automatically so data communication is transparent to a user.

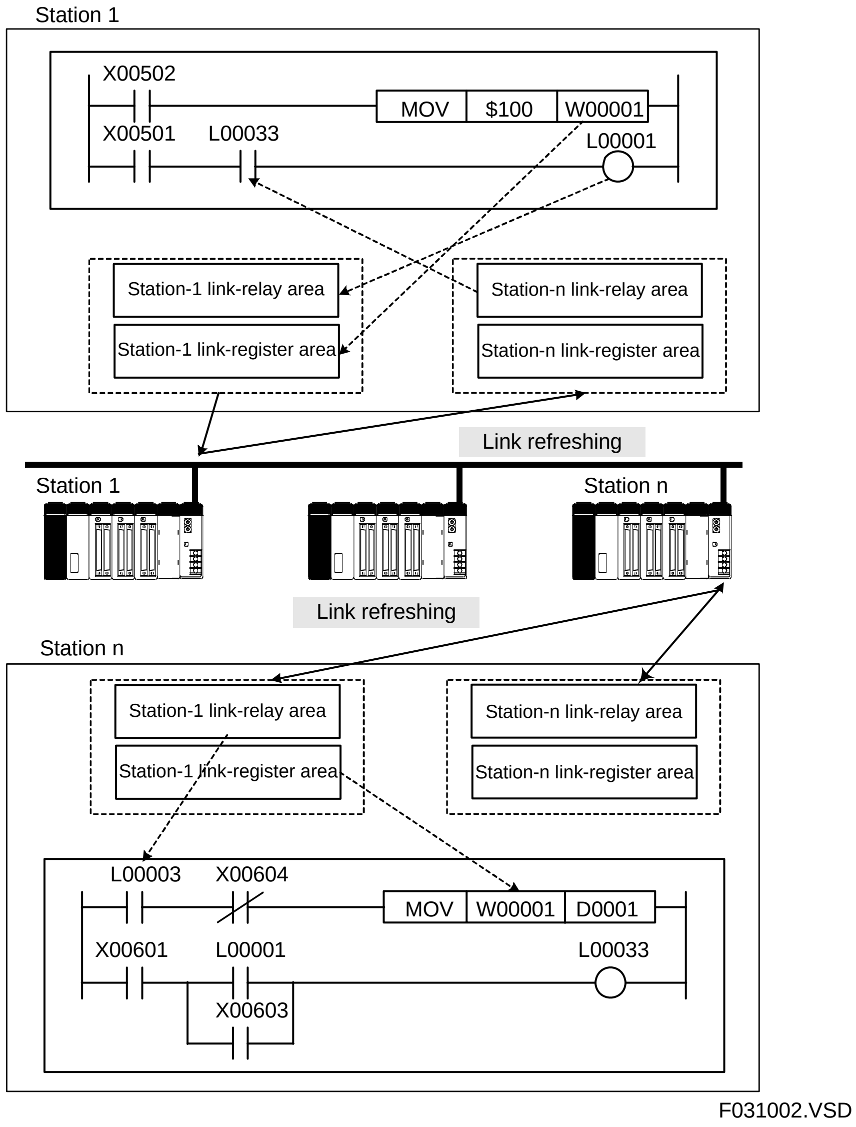

Station 1
X00502
MOV
$100
W00001
X00501
L00033
L00001
Station-1 link-relay area
Station-1 link-register area
Station-n link-relay area
Station-n link-register area
Link refreshing
Station 1
Station n
Link refreshing
Station n
Station-1 link-relay area
Station-1 link-register area
Station-n link-relay area
Station-n link-register area
L00003
X00604
MOV
W00001
D0001
X00601
L00001
L00033
X00603
F031002.VSD

Figure A3.10.2 Link Refreshing

■ Execution of FA Link Refreshing

FA link refreshing is executed in peripheral processing.

Link refreshing runs concurrently with instruction execution so it does not affect the scan time.

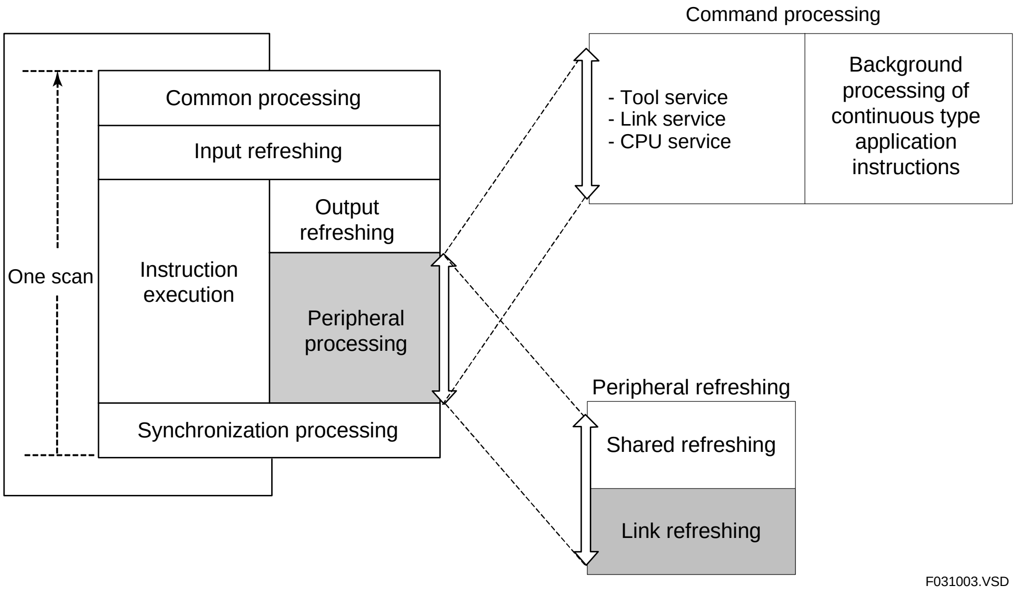


Figure A3.10.3 Executing FA Link Refreshing as a Peripheral Process

Link refreshing updates the link relays (L)/registers (W) of FA link 1 to FA link 8 in each cycle of peripheral processing.

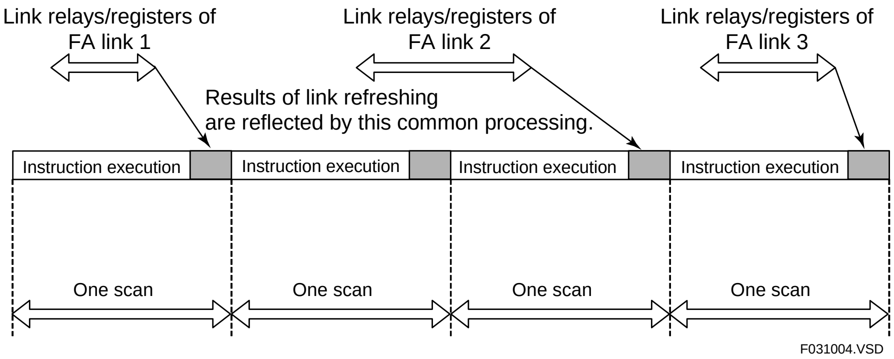

Link relays/registers of FA link 1
Link relays/registers of FA link 2
Link relays/registers of FA link 3
Results of link refreshing are reflected by this common processing.
Instruction execution
Instruction execution
Instruction execution
Instruction execution
One scan
One scan
One scan
One scan
F031004.VSD

Figure A3.10.4 FA Link Refreshing Sequence

SEE ALSO

For details on how FA link refreshing affects the scan time, see Chapter A7, "I/O Response Time Based on Scan Time".

■ Execution of FL-net Link Refreshing

You can specify by configuration whether to execute FL-net link refreshing as a peripheral process or a control-related process.

Link refreshing executed as a peripheral process does not affect the scan time.

Executing link refreshing as a control-related process may lengthen the scan time but it ensures that link refreshing is not affected by the shared refreshing or command processing time.

Include link refreshing in peripheral processes if a short scan time is important. Alternatively, include it in control-related processes if the speed of exchanging link data is important.

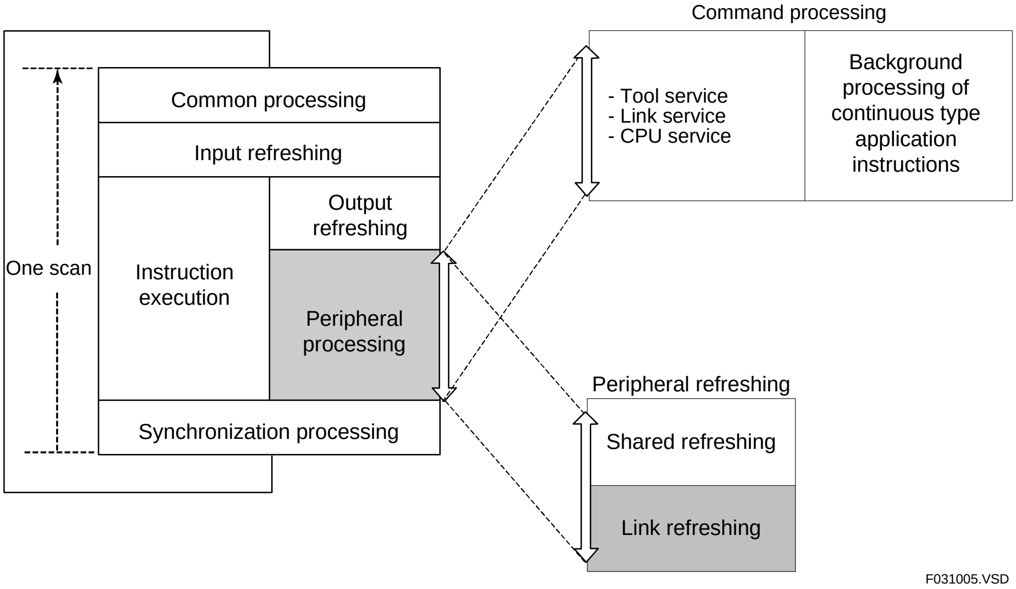


Figure A3.10.5 Executing FL-net Link Refreshing as a Peripheral Process

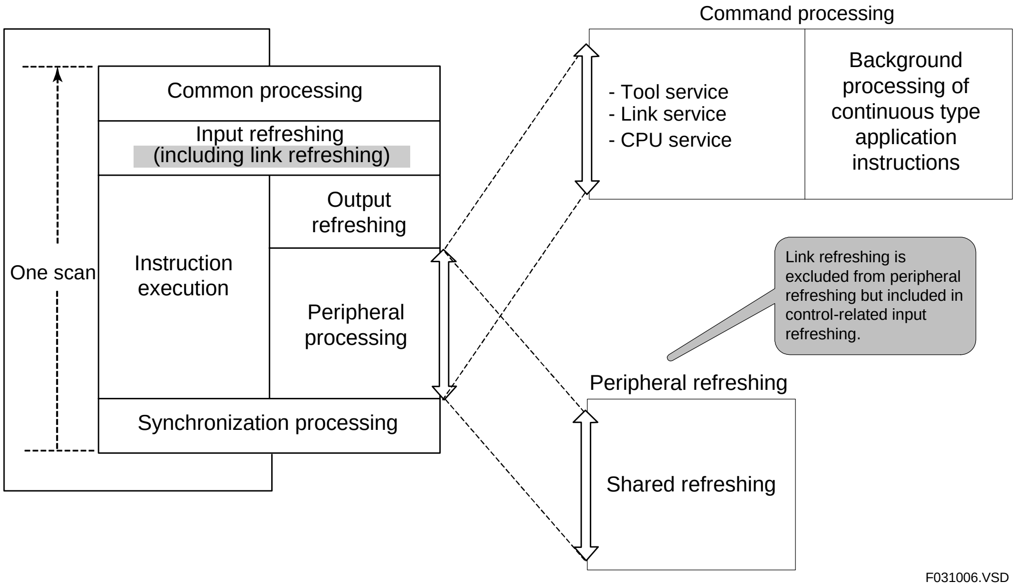

Command processing
Common processing
Input refreshing (including link refreshing)
Output refreshing
One scan
Instruction execution
Peripheral processing
Synchronization processing
- Tool service
- Link service
- CPU service
Background processing of continuous type application instructions
Link refreshing is excluded from peripheral refreshing but included in control-related input refreshing.
Peripheral refreshing
Shared refreshing
F031006.VSD

Figure A3.10.6 Executing FL-net Link Refreshing as a Control-related Process

When executed as a peripheral process, link refreshing updates the link relays (L) and link registers (W) of system 1 or system 2 in each cycle of peripheral processing.

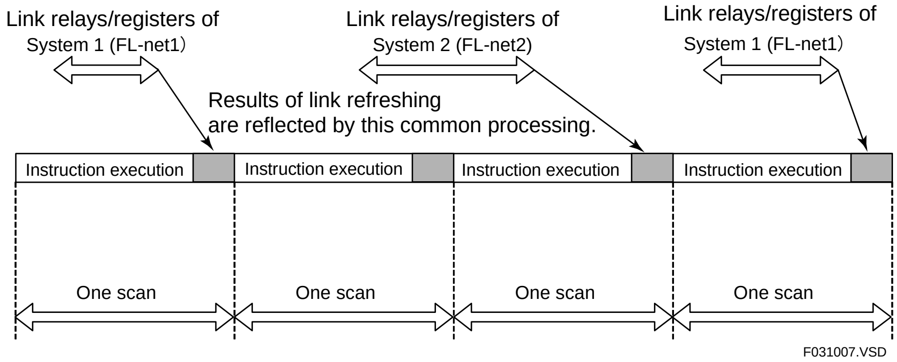

Link relays/registers of System 1 (FL-net1)
Link relays/registers of System 2 (FL-net2)
Link relays/registers of System 1 (FL-net1)
Results of link refreshing are reflected by this common processing.
Instruction execution
Instruction execution
Instruction execution
Instruction execution
One scan
One scan
One scan
One scan
F031007.VSD

Figure A3.10.7 FL-net Link Refreshing as a Peripheral Process

When executed as a control-related process, link refreshing updates the link relays (L) and link registers (W) of system 1 or system 2 in each cycle of control processing

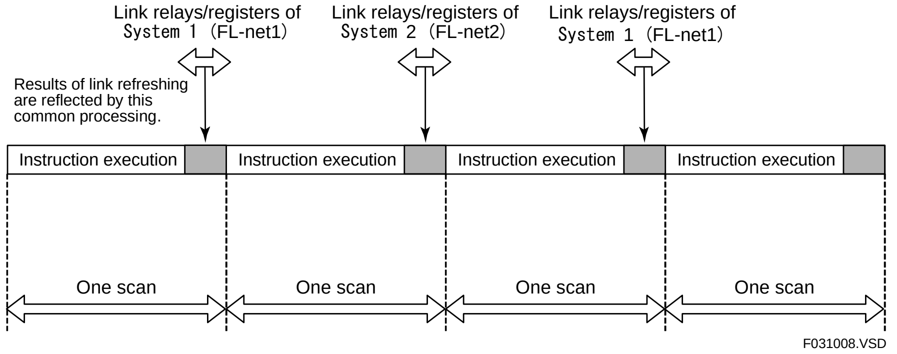

Link relays/registers of System 1 (FL-net1)
Link relays/registers of System 2 (FL-net2)
Link relays/registers of System 1 (FL-net1)
Results of link refreshing are reflected by this common processing.
Instruction execution
Instruction execution
Instruction execution
Instruction execution
One scan
One scan
One scan
One scan
F031008.VSD

Figure A3.10.8 Executing FL-net Link Refreshing as a Control-related Process

SEE ALSO

For details on how FL-net link refreshing affects the scan time, see Chapter A7, “I/O Response Time Based on Scan Time".

■ Inter-mixing FA Link Modules and FL-net Interface Modules

Where FA link and FL-net are intermixed in a system configuration, FA link refreshing and FL-net link refreshing are executed independently of each other.

If FL-net link refreshing is configured as a peripheral process, FA link refreshing and FL-net link refreshing are executed in each cycle of peripheral processing.

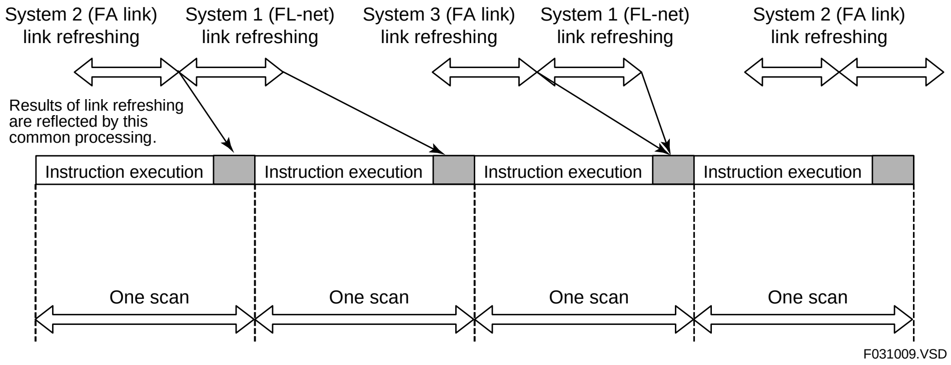

System 2 (FA link) link refreshing
System 1 (FL-net) link refreshing
System 3 (FA link) link refreshing
System 1 (FL-net) link refreshing
System 2 (FA link) link refreshing
Results of link refreshing are reflected by this common processing.
Instruction execution
Instruction execution
Instruction execution
Instruction execution
One scan
One scan
One scan
One scan
F031009.VSD

Figure A3.10.9   Intermixing FA Link and FL-net

Even if FL-net link refreshing is configured as a control-related process, FA link refreshing is still and always executed as a peripheral process.

A3.11 Method of Interrupt Processing

This section describes interrupt processing, interrupt processing control, interrupt timing, and priority of interrupts.

A3.11.1 Interrupt Processing

The sequence CPU module executes an input interrupt program when it detects the rising edge of an interrupt input from an input module.

You can register a maximum of four interrupt programs with the sequence CPU module using interrupt instructions (INTP instructions.)

The module can accept a maximum of eight concurrent interrupts. Interrupt programs are executed in the order of occurrence of their interrupt factors. If any interrupt factor occurs during execution of an interrupt program, the factor is processed when the interrupt program completes execution.

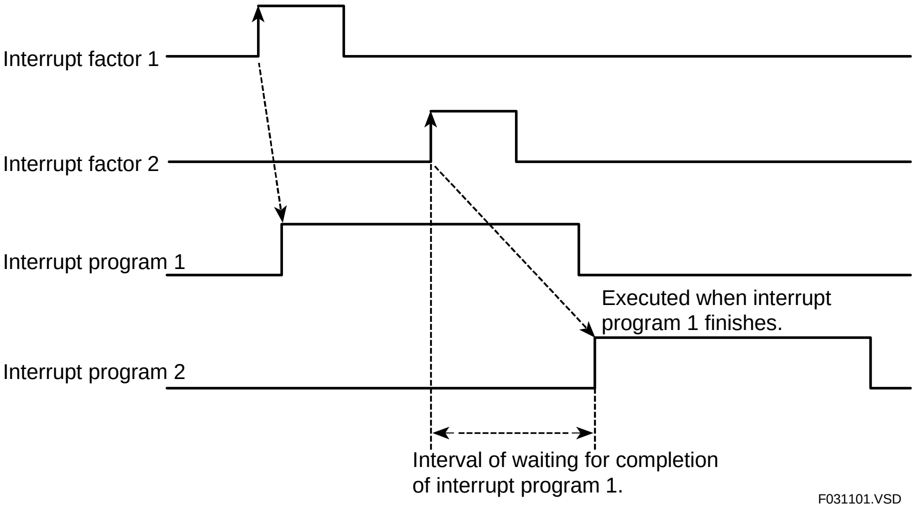

Interrupt factor 1
Interrupt factor 2
Interrupt program 1
Interrupt program 2
Executed when interrupt program 1 finishes.
Interval of waiting for completion of interrupt program 1.
F031101.VSD

Figure A3.11.1 Interrupt Processing

CAUTION

- Do not register an interrupt program for an input module with two or more CPU modules. Otherwise, interrupt processing may fail to be executed.
- Do not use a TIMER instruction in any interrupt program because the instruction may not work correctly.

A3.11.2 Interrupt Processing Control

You can control the execution of interrupt programs by means of programming. Use the Enable Interrupt (EI) instruction and Disable Interrupt (DI) instructions respectively to specify whether to execute (enable) or not execute (disable) an interrupt. Interrupts are enabled by default.

An interrupt that is disabled by a DI instruction continues to be detected by the sequence CPU but its interrupt program is not executed. Such interrupts are processed in order of their occurrence if and after they are enabled by an EI instruction.

A maximum of eight concurrent interrupts are accepted. The ninth or subsequent concurrent interrupt generates an interrupt error.

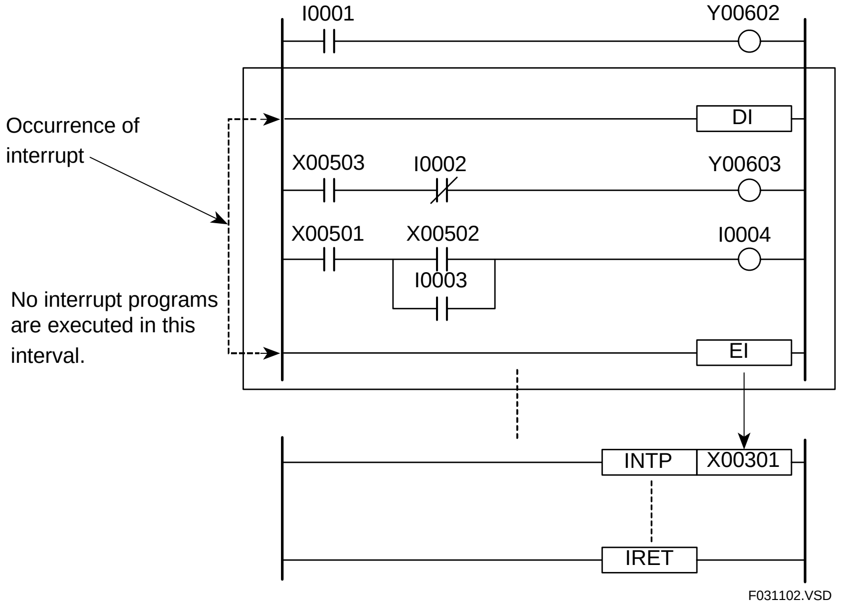


Figure A3.11.2 Interrupt Processing Control

A3.11.3 Interrupt Timing

Using the configuration function of WideField2, you can specify when an interrupt program is to be executed if an interrupt occurs during program execution. The following two options are available.

Table A3.11.1 Interrupt Timing Options of Interrupt Processing

Timing of Interrupt	Description
After instruction (default)	The sequence CPU switches execution to an interrupt program (program code between INTP and IRET instructions) after it finishes executing a ladder instruction. This switching does not take place, however, during synchronization processing, common processing or input refreshing.
Immediate (during instruction execution)	The sequence CPU switches execution to an interrupt program (program code between INTP and IRET instructions) during execution of a ladder instruction. This switching takes place even during synchronization processing, common processing or input refreshing.

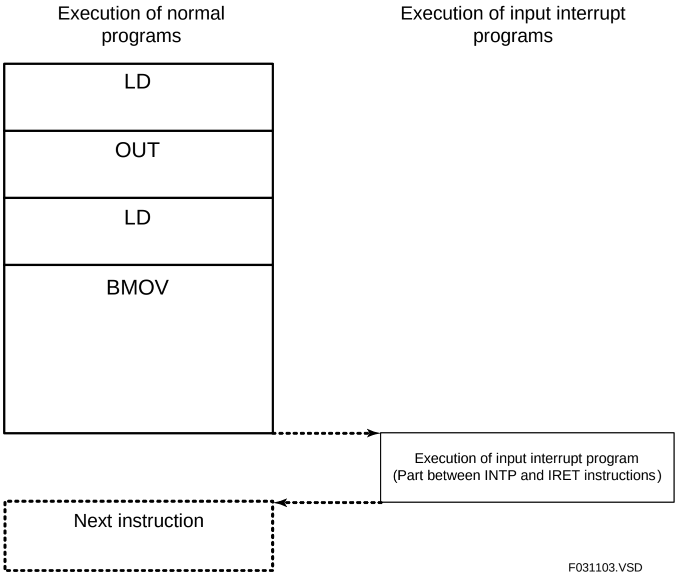

Execution of normal programs
Execution of input interrupt programs
LD
OUT
LD
BMOV
Execution of input interrupt program (Part between INTP and IRET instructions)
Next instruction
F031103.VSD

Figure A3.11.3 Execution of Interrupt Program after the Completion of Instruction Execution

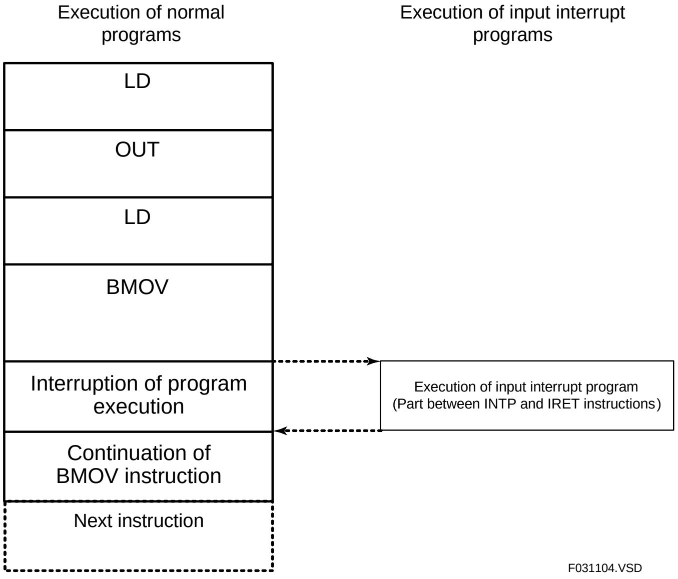

Execution of normal programs
Execution of input interrupt programs
LD
OUT
LD
BMOV
Interruption of program execution
Continuation of BMOV instruction
Execution of input interrupt program (Part between INTP and IRET instructions)
Next instruction
F031104.VSD

Figure A3.11.4 Immediate Execution of Interrupt Program during Instruction Execution

The characteristics of these two interrupt timing options are summarized in the table below.

Table A3.11.2 Characteristics of Interrupt Timing Options

Item	Execution of Interrupt Program after the Completion of Instruction Execution	Immediate Execution of Interrupt Program during Instruction Execution
Execution delay *1	"Processing time of instruction being executed *2 + switching time *3" or "synchronization processing time *4 + common processing time + input refreshing time *4 + switching time *3 "	Switching time only *3
Simultaneity of data	Guaranteed on an instruction basis	None for multiple devices

*1: The indicated time does not include the response time of an input module. For details on the response time of input modules, see "Hardware Manual" (IM34M6C11-01E).

*2: For details on the instruction processing time, see the appendix of "Sequence CPU Modules – Instructions User's Manual" (IM34M6P12-03E),

*3: 120 μs for F3SP28 and F3SP38 modules and 100 μs for F3SP53, F3SP58, F3SP59, F3SP66 and F3SP67 modules.

*4: See Section A7.1, "Description of Scan Time".

CAUTION

- **Output of data to relays with interrupt programs executed immediately during instruction execution**
  Be careful when outputting data to relays using an output instruction (e.g. OUT, SET or RST) if interrupt processing is configured with the timing option of “Immediate (during instruction execution)". In such cases, do not output data in a normal-scan program to any of the relays numbered 1 to 16 if data is output to any of these relays in an interrupt program (there is no limitation on data input, however).
  Example: Interrupt program: OUT I2
  Normal-scan program: OUT I1 - Not allowed.
  OUT I17 - Allowed.
  The same rule applies to relays numbered 17 to 32, 33 to 48, 49 to 64, and so on. If both the interrupt program and normal-scan program output to these relays, no output may be generated.

- **Simultaneity of multi-device data when interrupt programs are executed immediately during instruction execution**
  Simultaneity of data for multiple devices is not guaranteed if interrupt programs are executed immediately during instruction execution.
  Simultaneity of data is required when data of multiple devices is exchanged between a normal-scan program and an interrupt program using a block transfer (BMOV) instruction, a long-word instruction with IEEE single-precision floating point data, or two or more instructions.
  For example, consider the case shown in Figure A3.11.3 where an interrupt program is executed during execution of a block transfer (BMOV) instruction in a normal scan. There is a risk that block data partially transferred may be overwritten after the execution of the interrupt program.
  If simultaneity of data is required when interrupt timing is configured as “Immediate (during instruction execution)”, use any of the following means to ensure data simultaneity:
  1. Use a Disable Interrupt (DI) instruction and an Enable Interrupt (EI) instruction to prevent all interrupt programs from being executed during exchange of multi-device data.
  2. Write an application program to perform flag control between the normal-scan program and the interrupt program using relays.

- **Simultaneity of refreshed data when interrupt programs are executed immediately during instruction execution**
  If interrupt processing is configured with the timing option of “Immediately (during instruction execution)”, an interrupt program may be executed even during synchronization processing, input refreshing and common processing.
  When an interrupt program is executed during synchronization processing or input refreshing, values of devices (I/O relays (X/Y), shared and extended shared relays (E), shared and extended shared registers(R), and link relays and registers (L/W)) which are being refreshed may be read by programs. If these device values are overwritten by the interrupt program, simultaneity of data before and after the execution of the interrupt program is lost.
  To prevent all interrupt programs from being executed during synchronization processing, input refreshing and common processing, execute a Disable Interrupt (DI) instruction at the end of a normal-scan program. Along with this instruction, execute an Enable Interrupt (EI) instruction at the start of the normal-scan program.

A3.11.4 Priority of Interrupts

You can specify the priority of interrupts by configuration using WideField2 (“Priority of Interrupts” of “Interrupt Setup”) for conflict resolution in the event that input interrupt processing coincides with an interrupt from a sensor control block.

The table below lists the two options for “Priority of Interrupts”, along with how they work.

Table A3.11.3 Options for Priority of Interrupts

Priority of Interrupts	Functionality	
	When an interrupt from an input module occurs during execution of a sensor control block	When the time for executing a sensor control block arrives during input interrupt processing
Sensor CB interrupt has priority (default)	Executes the interrupt process after executing the sensor control block.	Suspends the interrupt process and resumes execution after executing the sensor control block.
Input interrupt has priority	Suspends the execution of the sensor control block and resumes execution after executing the interrupt process.	Executes the sensor control block after executing the interrupt process.

CAUTION

The sequence CPU applies the rule of interrupt execution timing (after completion of instruction execution or immediately during instruction execution) discussed earlier, even in the case where execution of the sensor control block or interrupt process is suspended due to interrupt priority.

A4. Devices

This chapter describes the types and functions of devices available with the sequence CPU modules.
Relay devices are accessed on a one-bit basis. Thus a relay device number corresponds to a bit.
Register devices are accessed on a 16-bit basis. Thus a register device number corresponds to 16 bits.

A4.1 I/O Relays (X/Y)

I/O relays (X/Y) are devices used to exchange data with external equipment.
I/O relay (X/Y) numbers are determined by the position of the slot where an I/O module is installed. They are fixed, discontinuous numbers and are assigned in increments of 64 relays for each slot.
The input relay (X) numbers never coincide with any of the output relay (Y) numbers.
Data held in the I/O relays is not retained when the power is turned off.

SEE ALSO

For details on I/O relay (X/Y) number definitions, see Section A1.3, “Basic Configuration".

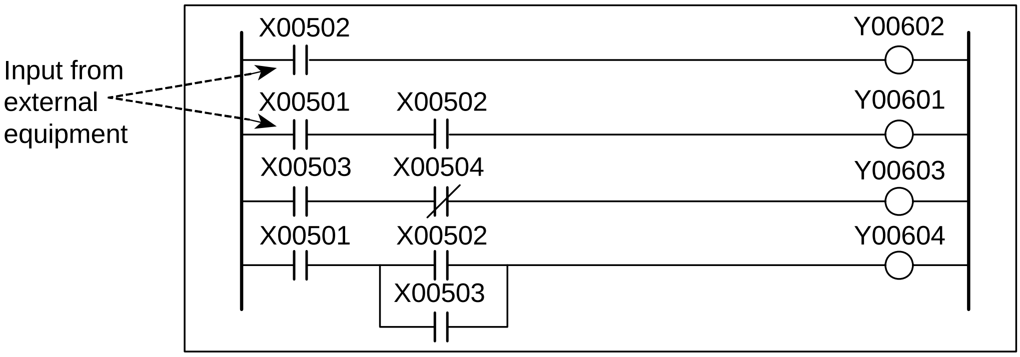
A4.1.1 Input Relays (X)

Input relays are used to input the ON and OFF states of external equipment, such as pushbuttons and limit switches.

In programs, you can use these relays for contacts **a** and **b** and application instructions.

Input relay numbers are coded as X,L mmnn, where:

L mm = Slot number
L = Unit number (0 to 7)
mm = Slot position (01 to 16)
nn = Terminal number (1 to 64)



F040101.VSD

Figure A4.1.1 Input Relays (X)

A4.1.2 Output Relays (Y)

Output relays are used to output the results of program-based control to external equipment, such as actuators. In programs, you can use these relays, for example, for contacts **a** and **b**, coils and application instructions.

Output relay numbers are represented as Y, Lmmnn, where:

Lmm = Slot number
L = Unit number (0 to 7)
mm = Slot position (01 to 16)
nn = Terminal number (1 to 64)

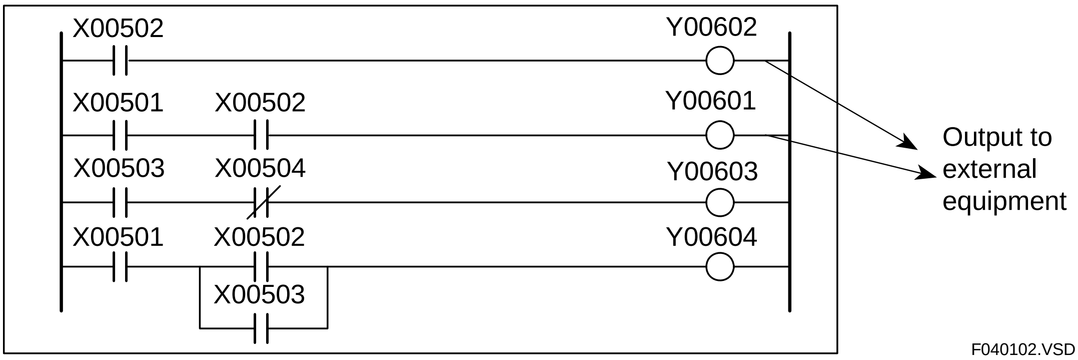


Figure A4.1.2 Output Relays (Y)

A4.1.3 Allocation of I/O Addresses

There is no need to allocate I/O address through WideField2.

I/O relay numbers are determined by the position of the slot where an I/O module is installed. They are fixed, discontinuous numbers and assigned on 64-relay basis for each slot. An empty slot is regarded as being equivalent to 64 relays.

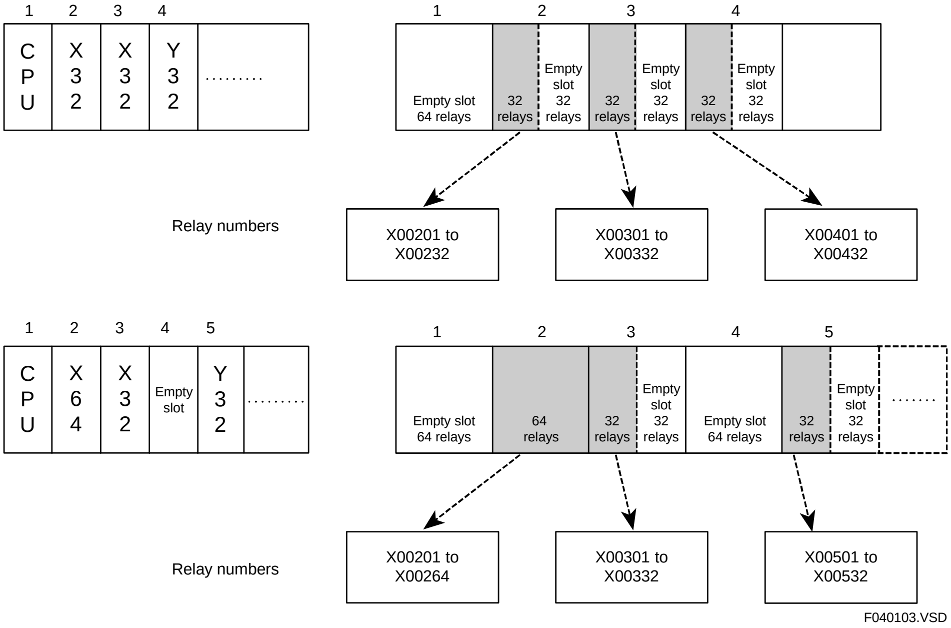


Figure A4.1.3 Allocation of I/O Addresses

A4.1.4 Configuring DIO Modules

This section describes settings for terminal usage (use/not used/sensor control block), data code (BIN/BCD), input sampling interval (16 ms/1.0 ms/250 μs/62.5 μs/Always), and holding/resetting output relays when the program stops.

■ Specifying Terminal Usage

With the configuration function, select one of the three options, “Use,” “Use with SCB,” and “Not Used” to specify whether the I/O module is used in programs, or used in the sensor control block, or not used at all. In this selection, configure the I/O module on 16-point basis (see the second caution below, when the selected option is “Use with SCB”). Configure special modules containing I/O relays (X/Y) in the same way as discussed here.

I/O relays that are included in the option “Not Used” are not refreshed at all. By default, all I/O modules are set to the option “Use".

SEE ALSO

For details on sensor control block, see Section A6.15, "Sensor Control Function".

CAUTION

- **When using output modules and special modules with Y□□□□□ output relays (Y) in a multi-CPU system configuration**

  - Combination of F3SP28, F3SP38, F3SP53, F3SP58, F3SP59, F3SP66 and F3SP67 CPU Modules

    You can output data from multiple sequence CPU modules separately to the output relays (Y) of the same output module on 16-relay basis. To do this, set the unused output terminals to the option “Not Used” on 16-terminal basis.

  - Other Combinations of CPU Modules

    You may not use the same output module with multiple CPUs. Configure all CPUs that do not use the output module so that the output module is set to “Not Used".

- **When using the sensor control block**

  When using an input or output module with the sensor control block, you must configure the input module on long word basis (i.e., 32 relays, 32 terminals, terminals 1 to 32 or terminals 33 to 64); and the output module on word basis (i.e., 16 relays or 16 terminals).

  Let's suppose you configured the input module incorrectly on word basis (for example, you set terminals 1 to 16 to the option “Used” [with normal scan] and terminals 17 to 32 to the option “Use with SCB”). Since input refreshing is performed on long word basis, input (X) relays used under a normal scan are refreshed by the Refresh instruction of the sensor control block when the normal scan is in progress. Consequently, the simultaneity of data is not guaranteed before and after the refreshing. Simultaneity of data is also not guaranteed for input (X) relays used in the sensor control block.

- **When using a Direct Refresh (DREF) instruction**
  Set the output relays (Y) to be refreshed by a DREF instruction of a program to the option “Not Used".

  If you set them to the option “Use” or “Use with SCB,” the values one scan earlier may be overwritten with the values output by the DREF instruction because of the timing of output refreshing, which is executed concurrently with instructions.

■ Specifying Data Code

Specify whether data held in I/O relays (X/Y) should be handled as "BIN" data or “BCD" data when they are used in a Compare, Arithmetic or Move instruction.

All internal computations are based on BIN data. For this reason, if you set the data code of an I/O relay to "BCD,” data is automatically converted from BCD to BIN for an input relay and from BIN to BCD for an output relay.

This option enables you to handle data easily, without worrying about the data representation during programming, especially in cases where data handled by external equipment are in BCD data code.

By default, I/O relays of all I/O modules are handled as “BIN” data. You can specify the data code on 16-relay basis.

■ Specifying Input Sampling Interval

Set the input sampling interval for input relays of input modules.

Note that this setting is not valid for some input modules.

SEE ALSO

For details of individual modules, see data item “response time” in the specifications section of individual input modules given in "Hardware Manual" (IM34M6C11-01E)

You can select from the five options, “16 ms,” “1.0 ms,” “250 μs,” “62.5 μs” and “Always". By default, all input modules are set to “16 ms". You can specify the sampling interval on 16-relay basis.

CAUTION

If a single input module (or special module with input relays X□□□□□) is used with two or more CPU modules in a multi-CPU system configuration, configure the CPUs so that they have the same sampling interval for all relays of that input module. (Also re-configure any CPU whose input relays (X) were set to the option “Not Used,” so that it has the same settings as the other CPUs.) Otherwise, system operation may be unstable.

■ Specifying Output when Stopped (Holding/Resetting Output Relays When Sequence Stops)

Specify whether the output relays (Y) of an output module (or special module with output relays Y□□□□□) should be placed in a “Hold” state or “Reset” state when a program stops (due to a moderate or major failure or a switch to stop mode).

The setting is, however, ignored by some output modules in the event of a major failure.

SEE ALSO

For details of individual modules, see the data item “Output status when the program stops HOLD/RESET” in the specifications section of each individual output module given in "Hardware Manual" (IM34M6C11-01E).

For a special module, the setting for a stop of programs due to a major failure is always ignored.

By default, all output modules are set to the option “Reset". You can perform this configuration on 16-relay basis.

CAUTION

- **When using output modules or special modules with Y□□□□□ output relays in a multi-CPU system configuration**
  - Combination of F3SP28, F3SP38, F3SP53, F3SP58, F3SP59, F3SP66 and F3SP67 CPU Modules

    You can output data from multiple sequence CPU modules separately to the output relays of the same output module on 16-relay basis. To do this, configure the sequence CPU modules so that all of them share the same output option (“Hold” or “Reset”). (Also re-configure any sequence CPU module whose output relays are set to the option “Not Used,” so that it has the same settings as the other CPUs.)
  - Other Combinations of CPU Modules

    You may not use the same output module with multiple CPUs.

A4.2 Internal Relays (I), Shared Relays (E) and Extended Shared Relays (E)

This section describes internal relays (I), shared relays (E), and extended shared relays (E).

Internal relays (I) are 1-bit variables that can be used without restriction in a program.

Shared relays (E) and Extended Shared relays (E) are 1-bit variables that can be used to perform data communications between CPUs in a multi-CPU system.

A4.2.1 Internal Relays (I)

Internal relays are auxiliary relays available for use in programs.

In programs, you can use these relays, for example, for contacts **a** and **b**, coils and application instructions. Unlike I/O relays (X/Y) however, these relays cannot directly exchange signals with external equipment. There is no limit on the number of contacts **a** and **b** that can be used in a program.

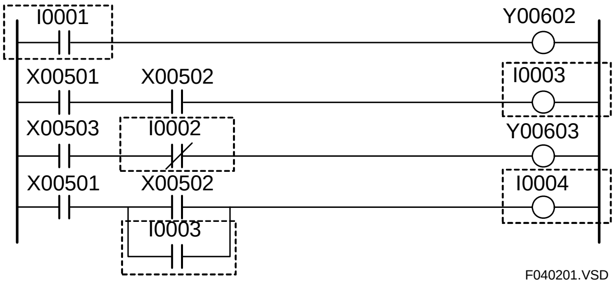


Figure A4.2.1 Internal Relays (I)

Using the configuration function, you can define the data lock-up range at power failure for devices whose computation results are to be latched when power is turned off.

A non-latched device will be cleared to “OFF (0) ” when you:

- power on the module;
- switch the operating mode to Run or Debug using WideField2; or
- execute a Clear Device command from WideField2.

A latched device retains its computation result even after power off and power on, and is cleared to “OFF (0) ” when you:

- execute a Clear Device command from WideField2.

A4.2.2 Shared Relays (E) and Extended Shared Relays (E)

Shared and extended shared relays are used to perform communications between CPU modules in cases where a sequence CPU module and add-on CPU modules are installed.

Shared relays (E) are available with the F3SP21, F3SP25, F3SP28, F3SP35, F3SP38, F3SP53, F3SP58, F3SP59, F3SP66 and F3SP67 sequence CPU modules, as well as with any add-on sequence CPU modules that are combined with one of these sequence CPUs.

Extended shared relays (E) are only available if one of the F3SP25, F3SP28, F3SP35, F3SP38, F3SP53, F3SP58, F3SP59, F3SP66 and F3SP67 sequence CPU modules is combined with any one or more of these CPU modules installed as add-on CPU modules.

In programs, you can use these relays, for example, for contacts **a** and **b**, coils and application instructions.

You can exchange ON/OFF data between CPUs by using shared relays (E) of a CPU as coils and those of the other CPUs as contacts.

CAUTION

If you write data to a device area not belonging to a CPU, data of shared and extended shared relays (E) of the other CPUs are overwritten so computation results are not correctly reflected.

By default, no shared relays are allocated as devices. When using add-on CPU modules, configure the range of shared relays to be used. Allocate the same device range for all of the CPU modules. Otherwise, the shared relays (E) will not be correctly refreshed.

The following figure shows an example of how specific shared relays are shared if you allocate shared relays E0001-0512 to CPU1 and E0513-1024 to CPU2.

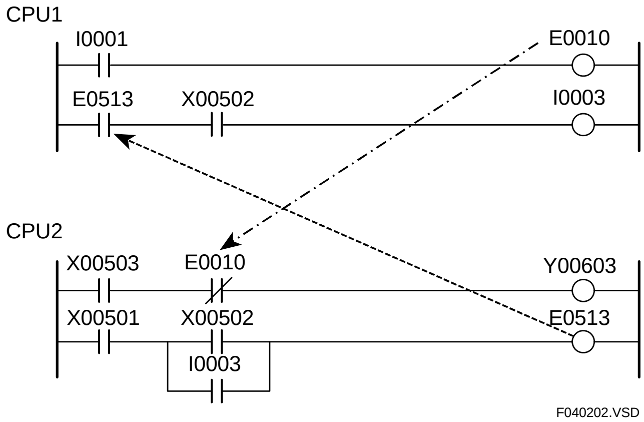


Figure A4.2.2 Shared Relays (E)

Using the configuration function, you can define the data lock-up range at power failure for devices whose computation results are to be latched when power is turned off.

By default, all shared relays (E) are non-latched.

A non-latched device will be cleared to “OFF (0)” when you:

- power on the module;
- switch the operating mode to Run or Debug using WideField2; or
- execute a Clear Device command from WideField2.

A latched device retains its computation result even after power off and power on, and is cleared to “OFF (0) ” when you:

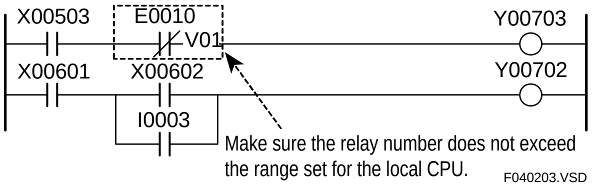
- execute a Clear Device command from WideField2.

CAUTION

When using shared or extended shared relays (E), observe the precautions given below.

(1) Index modification of shared or extended shared relays (E)

When applying index modification to a shared or extended shared relay (E) of the CPU, ensure that the resultant relay number does not exceed the range specified for the CPU in the configuration. Otherwise, information held by a shared or extended shared relay (E) of another sequence CPU module is overwritten and computation result is not correctly reflected.

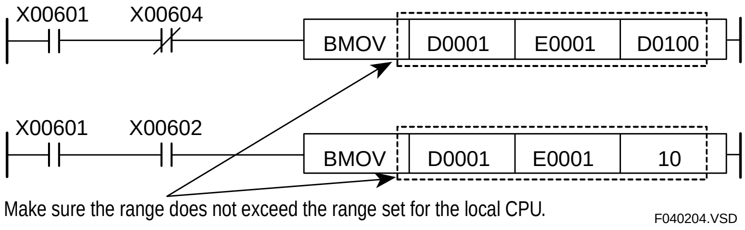


Make sure the relay number does not exceed the range set for the local CPU.

Figure A4.2.3 Precautions when Using Shared or Extended Shared Relays (E) (1 of 2)

(2) Block move and computation of multiple devices

When using shared or extended shared relays (E) in an instruction for transferring or computing data held by multiple devices, ensure that the specified range of these relays does not exceed the range specified for the CPU in the configuration. Otherwise, information held by shared or extended shared relays (E) of another sequence CPU module is overwritten and so computation result is not correctly reflected.



Make sure the range does not exceed the range set for the local CPU.

Figure A4.2.4 Precautions when Using Shared or Extended Shared Relays (E) (2 of 2)

(3) Simultaneity of data

Using the configuration function, you can select either “Simultaneous” or “Non-simultaneous” for simultaneity of data of shared devices.

If you select the “Simultaneous” option, simultaneity of data is guaranteed for units of devices (shared relays (E) /registers or extended shared relays (E) /registers) to be refreshed where one of the F3SP28, F3SP38, F3SP53, F3SP58, F3SP59, F3SP66 and F3SP67 sequence CPU modules is combined with any one or more of these CPUs installed as add-on CPUs. (Simultaneity of data between shared relays (E) /registers and extended shared relays (E) /registers is not guaranteed, however.)

If any of the F3SP28, F3SP38, F3SP53, F3SP58, F3SP59, F3SP66 and F3SP67 CPUs is combined with any of the F3SP21, F3SP25, and F3SP35 sequence CPU modules, simultaneity of shared refreshed data is not guaranteed regardless of the configuration setting.

The “Non-simultaneous” option is provided for compatibility with the F3SP21, F3SP25 and F3SP35 CPUs. Select this option when these CPUs are replaced with the F3SP28, F3SP38, F3SP53, F3SP58, F3SP59, F3SP66 and F3SP67 sequence CPU modules.

■ Configuring Shared and Extended Shared Relays (E) in a Multi-CPU System

Specify the range of shared and extended shared relays (E) to be used by each CPU when add-on CPU modules are installed.

You can allocate any number of relays on 32-relay basis.

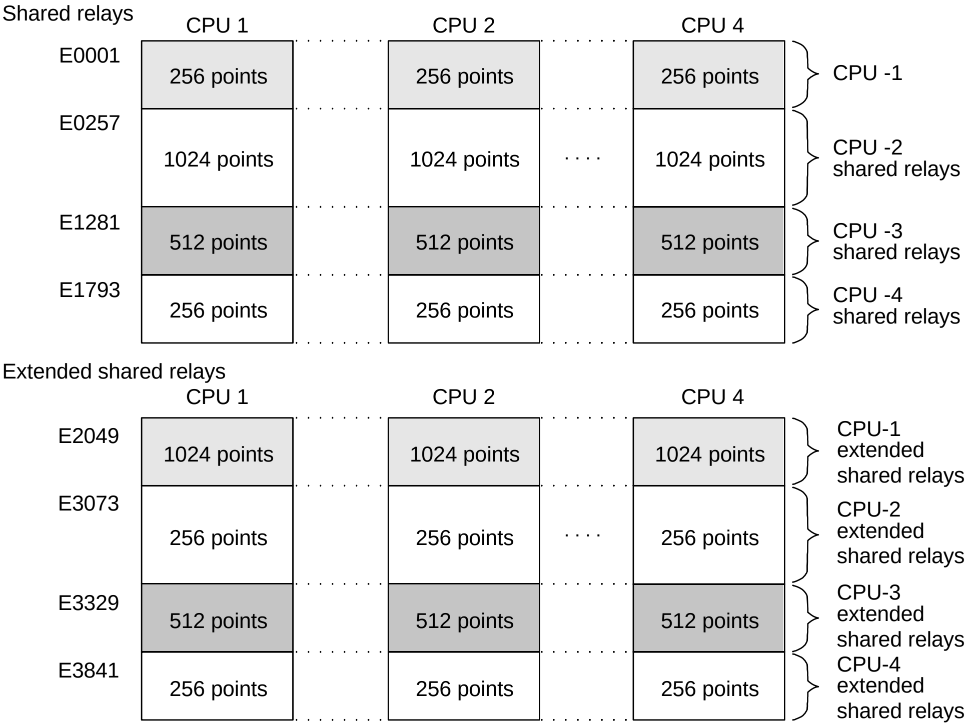
Extended shared relays (E) are only available if one of the F3SP25, F3SP28, F3SP35, F3SP38, F3SP53, F3SP58, F3SP59, F3SP66 and F3SP67 sequence CPU modules is combined with one or more CPUs from the same list installed as add-on CPUs.

Table A4.2.1 Configuration of Shared Relays (E)

Item	F3SP66, F3SP67	
	Default	Configuration Range
Shared relay (E)	0	2048 max. for all CPUs combined in increments of 32 (E0001 to E2048)
Extended shared relay (E)	0	2048 points max. for all CPUs combined in increments of 32 (E2049 to E4096)

CAUTION

- The starting number for extended shared relays (E) is always E2049 even if the range of shared relays (E) used is less than 2048.
- Apply the same allocation of shared/extended shared relays (E) to all CPUs. If the allocation differs among CPUs, shared refreshing will not execute correctly and as a result computation result will not be correctly reflected.



Shared relays

	CPU 1	CPU 2		CPU 4	
E0001	256 points	256 points		256 points	CPU -1
E0257	1024 points	1024 points	· · · ·	1024 points	CPU -2 shared relays
E1281	512 points	512 points		512 points	CPU -3 shared relays
E1793	256 points	256 points		256 points	CPU -4 shared relays

Extended shared relays

	CPU 1	CPU 2		CPU 4	
E2049	1024 points	1024 points		1024 points	CPU-1 extended shared relays
E3073	256 points	256 points	· · · ·	256 points	CPU-2 extended shared relays
E3329	512 points	512 points		512 points	CPU-3 extended shared relays
E3841	256 points	256 points		256 points	CPU-4 extended shared relays

F040205.VSD

Figure A4.2.5 Example of Shared and Extended Shared Relay (E) Allocation with Four Sequence CPU Modules Installed

A4.3 Link Relays (L) and Link Registers (W)

This section describes link relays (L), link registers (W), their settings, system numbers, as well as the link refreshing range.
Link relays (L) are 1-bit variables used for data communications with FA link systems and FL-net systems. Link registers (W) are 16-bit variables used for data communications with FA link systems and FL-net systems.

CAUTION

In this section, FL-net nodes are called "stations".

Link relays (L) and link registers (W) are devices used to exchange data with other programmable controllers via FA link modules and FL-net (OPCN-2) Interface modules.
Before using link relays, specify the range of links for both the local and remote stations.
Using the configuration function, you can define the data lock-up range at power failure for devices whose computation results are to be latched when power is turned off.
By default, all link relays (L) and link registers (W) are non-latched.
A non-latched device will be cleared to “OFF (0)” when you:

- power on the module;
- switch the operating mode to Run or Debug using WideField2; or
- execute a Clear Device command from WideField2.

A latched device retains its computation result even after power off and power on, and is cleared to “OFF (0) ” when you:

- execute a Clear Device command from WideField2.

A4.3.1 Link Relays (L)

Link relays are used to exchange data with other programmable controllers via FA link modules or FL-net (OPCN-2) Interface modules.

In programs, you can use these relays, for example, for contacts **a** and **b**, coils, and application instructions. In addition, you can exchange ON/OFF data between CPUs by using link relays (L) of the local station as coils and those of remote stations as contacts.

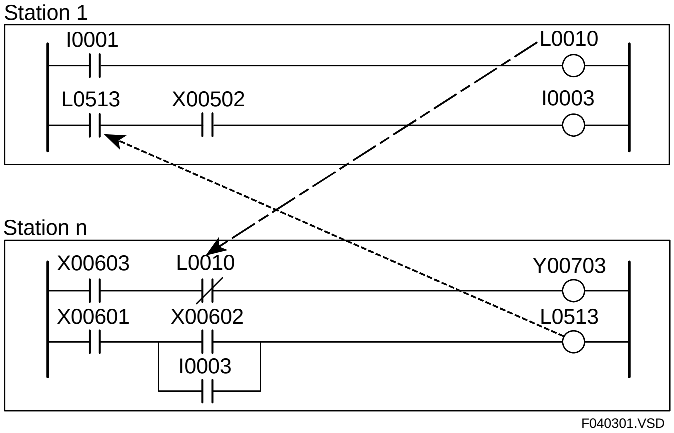


Figure A4.3.1 Link Relays (L)

The relay number is coded as Lmnnnn, where:

m	= System number 1 (0 to 7)
nnnn	= Link relay number

Table 4.3.1 Range of Link Relay Numbers

Module		Configuration Range
FA link H Module Fiber-optic FA Link H Module	High speed configuration	1 to 1024
	Normal configuration	1 to 2048
FL-net (OPCN-2) Interface Module		1 to 8192

A4.3.2 Link Registers (W)

Link registers are used to exchange data with other programmable controllers via FA link modules or FL-net (OPCN-2) Interface modules. In programs, you can read from or write to link registers on 16-bit or 32-bit basis using application instructions.

When you use a long word, the low-order 16 bits are stored in the link register with the number specified in the instruction and the high-order 16 bits are stored in the link register (W) with that number incremented by 1.

Data exchange between the local station and remote stations can be achieved by writing data to link registers (W) of the local station and reading it from a remote station.

Before using link registers, configure the range of links for the local station and remote stations.

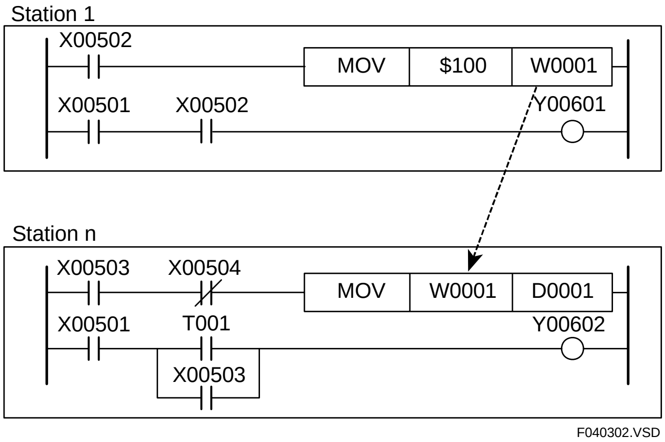


Figure A4.3.2 Link Registers (W)

The register number is coded as Wmnnnn, where:

m = System number 1 (0 to 7)

nnnn = Link register number

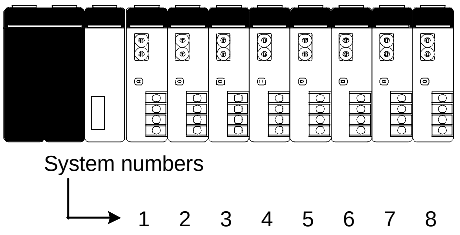
Table 4.3.2 Range of Link Register Number

Module		Configuration Range
FA Link H module Fiber-optic FA Link H module	High speed configuration	1 to 1024
	Normal configuration	1 to 2048
FL-net (OPCN-2) Interface module		1 to 8192

A4.3.3 System Numbers

In an FA link system or FL-net (OPCN-2) system, modules are automatically assigned system numbers based on their slot positions, with the module having the smallest slot number named as system 1.

If FA link modules and FL-net (OPCN-2) modules are intermixed, the modules are assigned system numbers sequentially regardless of the module type.

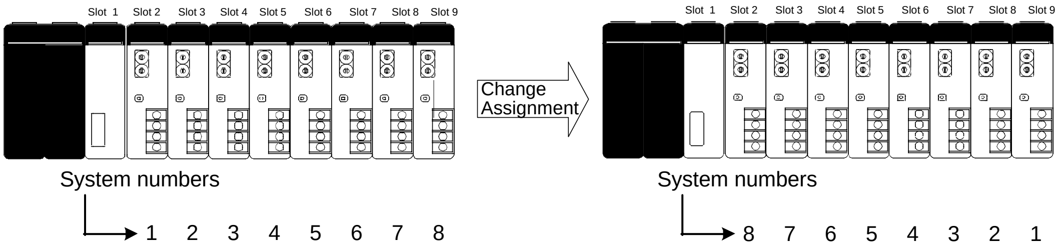


System 1 L (W) 00001 -
System 2 L (W) 10001 -
System 3 L (W) 20001 -
System 4 L (W) 30001 -
System 5 L (W) 40001 -
System 6 L (W) 50001 -
System 7 L (W) 60001 -
System 8 L (W) 70001 -

F040303.VSD

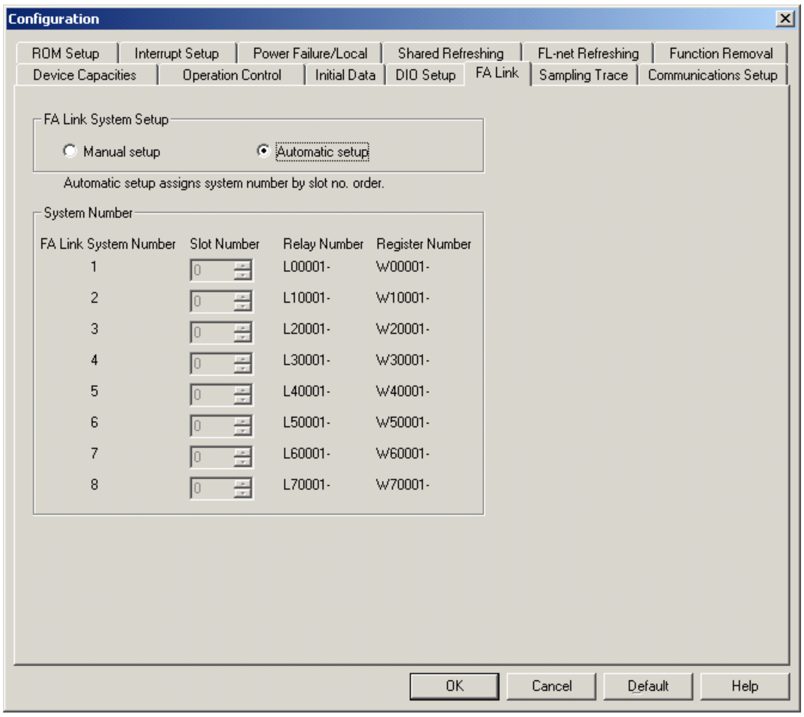
Figure A4.3.3 Assignment of System Numbers

To manually assign system numbers to modules independent of their slot positions, use configuration to assign fixed system numbers to slot positions.


Slot 1 Slot 2 Slot 3 Slot 4 Slot 5 Slot 6 Slot 7 Slot 8 Slot 9
System numbers
1 2 3 4 5 6 7 8
Change Assignment
Slot 1 Slot 2 Slot 3 Slot 4 Slot 5 Slot 6 Slot 7 Slot 8 Slot 9
System numbers
8 7 6 5 4 3 2 1

F040304.VSD

Figure A4.3.4 Changing System Number Assignment


Configuration
ROM Setup | Interrupt Setup | Power Failure/Local | Shared Refreshing | FL-net Refreshing | Function Removal
Device Capacities | Operation Control | Initial Data | DIO Setup | FA Link | Sampling Trace | Communications Setup
OK | Cancel | Default | Help

FA Link System Setup

○ Manual setup ● Automatic setup

Automatic setup assigns system number by slot no. order.

System Number

FA Link System Number	Slot Number	Relay Number	Register Number
1	0	L00001-	W00001-
2	0	L10001-	W10001-
3	0	L20001-	W20001-
4	0	L30001-	W30001-
5	0	L40001-	W40001-
6	0	L50001-	W50001-
7	0	L60001-	W60001-
8	0	L70001-	W70001-

F040305.VSD

Figure A4.3.5 WideField2 Configuration Setup

A4.3.4 Configuring Link Relays (L) and Registers (W)

Specify the range of link relays and registers to be included in each link system.
For each system, specify the number of link relays and link registers to be used.

Table A4.3.3 Device Capacities of Configuration

Item		F3SP66		F3SP67	
		Default	Configuration Range	Default	Configuration Range
Device Capacities	Link relays (L) for each system (FA- link or FL-net (OPCN-2) system)	System 1: 8192	8192 max. for all systems combined in increments of 16	Systems 1 and 2: 8192	16384 max. for all systems combined in increments of 16
		Systems 2 to 8:0		Systems 3-8: 0	
	Link registers (W) for each system (FA link or FL-net (OPCN-2) system)	System 1: 8192	8192 max. for all systems combined in increments of 16	Systems 1 and 2: 8192	16384 max. for all systems combined in increments of 16
		Systems 2 to 8:0		Systems 3-8: 0	

A4.3.5 Link Refreshing Range

This section describes the link refreshing range for an FA link or FL-net (OPCN-2).

■ Setting Link Refreshing Range for an FL-net System

You can select whether to perform link refreshing on per node basis by configuration.

You can specify not to read the common area (i.e. refresh links) of nodes not involved in data exchange so as to shorten the processing time required for link refreshing.

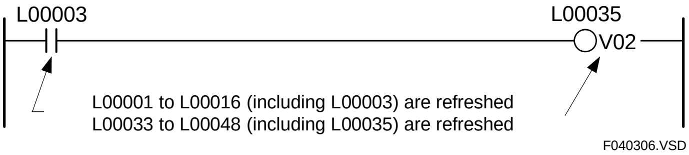
By default, all nodes are refreshed.

■ Setting Link Refreshing Range for an FA Link System

Link refreshing of an FA link module is performed only for link relays (L) and link registers (W) that are used by instructions coded in a program.

● Link Relay (L)

- If link relays (L) are directly coded in a program, each word containing such a link relay is refreshed.
- If link relays (L) are specified by index modification, each word including the link relay (L) designated by the index register with index value of 0 is refreshed.



L00003
L00035
V02
L00001 to L00016 (including L00003) are refreshed
L00033 to L00048 (including L00035) are refreshed
F040306.VSD

Figure A4.3.6 Link Relay (L) Link Refreshing Range

● Link Register (W)

- For an instruction that handles word data, the link register (W) specified in the instruction is refreshed.
- For an instruction that handles long-word data or IEEE single-precision floating-point data, the link register (W) specified in the instruction and the data in link register (W) +1 are refreshed.
- For an instruction that handles two or more words of data, the specified range of words is refreshed if the range is specified by a constant, while only the first word is refreshed if the range is specified by a register.

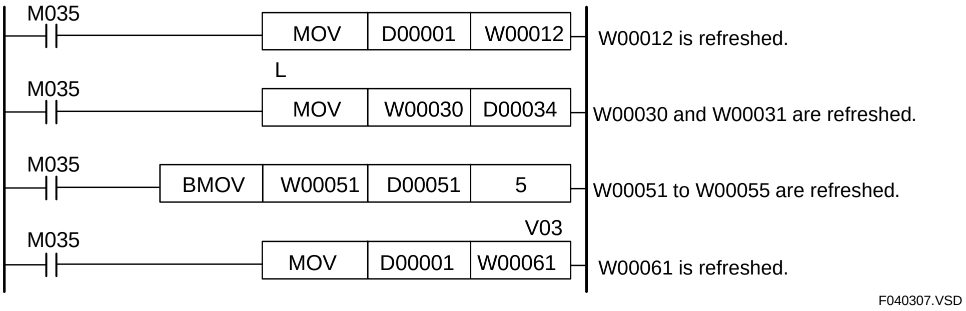
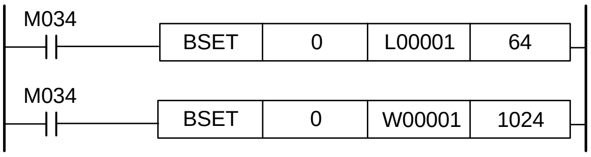


Figure A4.3.7 Link Refreshing Range for Link Registers (W)

TIP

Link relays (L) and link registers (W) specified in a program are included in link refreshing irrespective of whether the relevant instructions are executed.

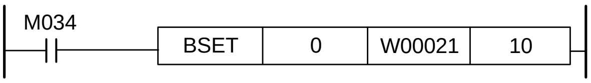
If you want to include all link relays (L) and link registers (W) in link refreshing, include the following code in your program:



F040308.VSD

Figure A4.3.8 To include L00001 to L01024 and W00001 to W01024 in Link Refreshing

If link relays (L) and link registers (W) are specified by index modification, define the index modification range as shown below so that they will be included in link refreshing.



F040309.VSD

Figure A4.3.9 When Using Registers W00021 to W00030 with Index Modification

CAUTION

(1) Index modification/indirect designation

- Index modification/indirect designation must not be made across different systems.
- When specifying link relays and registers with index modification or indirect designation, see TIP to link-refresh to ensure that all relevant devices are link-refreshed.

(2) Block move and computation involving multiple devices

- Block move or computation for multiple devices must not be made across different systems. Be careful especially when specifying the number of bytes of data to be moved or the number of devices for computation using devices.
- When specifying the number of bytes of data to be moved or the number of devices for computation using devices, see TIP to ensure that all relevant devices are link-refreshed.

(3) Multi-CPU configuration

- Multiple CPU modules cannot share the same FA link module or FL-net (OPCN-2) interface module. Ensure that only one CPU module is accessing an FA link module or FL-net (OPCN-2) interface module.

A4.4 Special Relays (M)

Special relays have specific functions, such as indicating the internal state of a sequence CPU module or detecting errors. In programs, these relays are used mainly for contacts a and b.

A4.4.1 Block Start Status

Block Start Status relays indicate which blocks are executed when only specified blocks are executed.

These relays are numbered in ascending order as M001, M002, ... to correlate with block 1, block 2, ...

Table A4.4.1 Block Start Status

No.	Name	Function	Description
M001 to M032	Block n Start Status	ON : Run OFF: Stop	Indicates whether block n is executed when the module is configured to execute specified blocks only.
M2001 to M3024			

Note: The Start Status relays assigned to blocks 1 to 32 are M0001 to M0032 and M2001 to M2032 (M0001 to M0032 have the same values as M2001 to M2032.) Similarly, Start Status relays M2033 to M3024 map to blocks 33 to 1024.

CAUTION

- Do not write to a special relay (M), including those not listed in the tables (e.g. M0067 to M0128) unless it is marked as "write-enabled". Special relays are used by the sequence CPU module. Writing to these relays incorrectly may lead to system shutdown or other failures. Using forced set/reset instruction in debug mode is also prohibited.
- Special relays (M) with index modification cannot be specified as destinations for data output and if specified, will result in instruction processing errors during execution.

CAUTION

Special relays (M) cannot be specified as output destinations in block transfer and table output ladder instructions, and if specified, will cause instruction processing errors during execution.

- Block transfer instructions: Block Move (BMOV), Block Set (BSET), String Move (SMOV), etc.
- Table output instructions: Read User Log (ULOGR), FIFO Write (FIFWR), etc.

A4.4.2 Utility Relays

Utility relays are used to provide timing in a program or issue instructions to the CPU module.

Table A4.4.2 Utility Relays

No.	Name	Function	Description
M033	Always ON	ON OFF	Used for initialization or as a dummy contact in a program.
M034	Always OFF	ON OFF	
M035	1 Scan ON at Program Start	1 Scan	Turns on for one scan only after a program starts execution
M036 *1	0.01 s Clock	0.005s 0.005s	Generates a clock pulse of 0.01 s period.
M037 *1	0.02 s Clock	0.01s 0.01s	Generates a clock pulse of 0.02 s period.
M038 *1	0.1 s Clock	0.05s 0.05s	Generates a clock pulse of 0.1 s period.
M039 *1	0.2 s Clock	0.1s 0.1s	Generates a clock pulse of 0.2 s period.
M040 *1	1 s Clock	0.5s 0.5s	Generates a clock pulse of 1 s period.
M041 *1	2 s Clock	1s 1s	Generates a clock pulse of 2 s period.
M042 *1	1 min Clock	30s 30s	Generates a clock pulse of 60 s period.
M047 *1	1 ms Clock	0.5ms 0.5ms	Generates a clock pulse of 1 ms period.
M048 *1	2 ms Clock	1ms 1ms	Generates a clock pulse of 2 ms period.
M049 to M064	Devices reserved for extended functions		
M066	Normal Subunit Transmission Line	ON :Normal transmission line or no fiber-optic FA-bus installed OFF:Unspecified or abnormal transmission line	
M097	ON for One Scan at Sensor CB Start	ON :At block start OFF:In all other cases	Turns on for one scan when the sensor control block starts (at the first execution of the sensor control block).

*1: Relays M036 to M048 have their rising and falling clock timing synchronized.

SEE ALSO

For details on the M066 Utility relay (Normal Subunit Transmission Line), see "Fiber-optic FA-bus Module and Fiber-optic FA-bus Type 2 Module User's Manual" (IM34M6H45-01E).

A4.4.3 Sequence Operation and Mode Status Relays

Sequence operation and mode status relays indicate the status of sequence operation and various modes.

Table A4.4.3 Sequence Operation and Mode Status Relays

No.	Name	Function	Description
M129	Run Mode Flag	ON : Run mode OFF: Other modes	Indicates the status of CPU operation.
M130	Debug Mode Flag	ON : Debug mode OFF: Other modes	Indicates the status of CPU operation.
M131	Stop Mode Flag	ON : Stop mode OFF: Other modes	Indicates the status of CPU operation.
M132	Pause Flag	ON : Pause OFF: Run	Indicates the status of program execution during debug mode operation.
M133	Execution Flag	ON : Specified blocks OFF: All blocks	Indicates whether all blocks or specified blocks are executed.
M135	ROM/RAM-based Operation Flag	ON : ROM-based operation OFF: RAM-based operation	Indicates whether operation is based on the ROM or RAM.
M136	Power-on Operation Flag	ON : Power-on operation OFF: Other modes of operation	Indicates whether operation was initiated by power on or reset
M137	Sensor CB Execution Status	ON : Run OFF: Stop	Indicates the status of sensor control block operation.
M172 (write-enabled)	Set Clock Time	ON : Time being set OFF:	Requests to set clock data.
M173	Input-offline Flag	ON : Offline OFF: Online	Indicates that input refreshing has stopped.
M174	Output-offline Flag	ON : Offline OFF: Online	Indicates that output refreshing has stopped.
M175	Shared-I/O-offline Flag	ON : Offline OFF: Online	Indicates that shared refreshing has stopped.
M176	Link-I/O-offline Flag	ON : Offline OFF: Online	Indicates that link refreshing has stopped.
M177 to M187	Devices reserved for extended functions		
M188	Carry Flag	ON : Carry enabled OFF: Carry disabled	Carry flag used by shift and rotate operations
M189 TO M192	Devices reserved for extended functions		

SEE ALSO

For details on special registers related to clock data (Z49 to Z54), see Subsection A4.8.3, "Utility Registers."

A4.4.4 Self-diagnosis Status Relays

Self-diagnosis status relays indicate the results of self-diagnosis by the sequence CPU.

Table A4.4.4 Self-diagnosis Status Relays

No.	Name	Function	Description
M193	Self-diagnosis Error	ON : Error OFF: No error	Result of self diagnosis is stored in special registers Z17 to Z19
M194	Battery Error	ON : Error OFF: Normal	Indicates a failure in backup batteries.
M195	Momentary Power Failure	ON : Momentary power failure OFF: No momentary power failure	Indicates that a momentary power failure has occurred.
M196	Inter-CPU Communication Error	ON : Error OFF: Normal	Indicates that a communication failure has occurred in shared relays (E) or shared registers (R).
M197	Existence of CPU1	ON : Exists. OFF: Does not exist.	Indicates whether or not a CPU exists in slot 1.
M198	Existence of CPU2	ON : Exists. OFF: Does not exist.	Indicates whether or not a CPU exists in slot 2.
M199	Existence of CPU3	ON : Exists. OFF: Does not exist.	Indicates whether or not a CPU exists in slot 3.
M200	Existence of CPU4	ON : Exists. OFF: Does not exist.	Indicates whether or not a CPU exists in slot 4.
M201	Instruction Processing Error	ON : Error OFF: No error	Information of instruction processing error is stored in special registers Z22 to Z24.
M202	I/O Comparison Error	ON : Error OFF: Normal	Indicates that the state of module installation is not consistent with the program.
M203	I/O Module Error	ON : Error OFF: Normal	Indicates that no access is possible to I/O modules. The slot number of the error module is stored in special registers Z33 to Z40.
M204	Scan Timeout	ON : Error OFF: Normal	Indicates that scan time has exceeded the scan monitoring time.
M210	Subunit Communication Error	ON : Error OFF: Unspecified or normal line	An error has been detected in the fiber-optic FA-bus module. The slot number of the error module is stored in special registers Z89 to Z96.
M211	Subunit Line Switchover	ON : Error OFF: Unspecified or normal line	
M212	Sensor CB Scan Timeout	ON : Error OFF: Normal	Indicates that the execution interval of the sensor control block cannot be maintained.
M225	CPU1 Sequence Program Execution	ON : Run OFF: Stop	Indicates whether sequence program of the CPU in slot 1 is running.
M226	CPU2 Sequence Program Execution	ON : Run OFF: Stop	Indicates whether sequence program of the CPU in slot 2 is running.
M227	CPU3 Sequence Program Execution	ON : Run OFF: Stop	Indicates whether sequence program of the CPU in slot 3 is running.
M228	CPU4 Sequence Program Execution	ON : Run OFF: Stop	Indicates whether sequence program of the CPU in slot 4 is running.

SEE ALSO

For details on the M210 (Subunit Communication Error) and M211 (Subunit Line Switchover) self-diagnosis relays, see "Fiber-optic FA-bus Module and Fiber-optic FA-bus Type 2 Module User's Manual" (IM34M6H45-01E),

A4.4.5 FA Link Module Status Relays

FA Link module status relays indicate the status of FA link.

Table A4.4.5 FA Link Module Status

No.	Name	Function	Description
M257 to M480 M8321 to M8992	FA Link Error	ON : Error OFF: Normal	Indicates the status of FA links.

SEE ALSO

For details on FA link module status relays, see the sections on special relays and special registers of "FA Link H Module, Fiber-optic FA Link H Module User's Manual" (IM34M5H43-01E),

A4.4.6 FL-net Interface Module Status Relays

FL-net interface module status relays indicate the status of FL-net.

Table A4.4.6 FL-net Interface Module Status

No.	Name	Function	Description
M0235	Invalid communication detected	1: Invalid 0: Valid	Invalid token mode
M0236	Overlapping address detected	1: Overlapping 0: Normal	Overlapping common memory allocation
M0237	Duplicate node number	1: Duplicate 0: Normal	Duplicate node number
M0238	Token monitoring time error	1: Error 0: Normal	Token holding time timeout
M0239	Waiting to receive	1: Error 0: Normal	Token not detected
M0240	Initialization error	1: Invalid 0: Valid	Invalid setup parameter
M3521 to M3774	Node participation status	1: Participating 0: Not participating	FL-net system 1[*1]
M3777 to M4030	Upper layer operation signal error	1: Error 0: Normal	FL-net system 1[*1]
M4033 to M4286	Operation status	1: Run 0: Stop	FL-net system 1[*1]
M4289 to M4542	Common memory data valid	1: Valid 0: Invalid	FL-net system 1[*1]
M4561 to M4814	Node participation status	1: Participating 0: Not participating	FL-net system 2[*2]
M4817 to M5070	Upper layer operation signal error	1: Error 0: Normal	FL-net system 2[*2]
M5073 to M5326	Operation status	1: Run 0: Stop	FL-net system 2[*2]
M5329 to M5582	Common memory data valid	1: Valid 0: Invalid	FL-net system 2[*2]

*1: If FA link modules and FL-net modules are intermixed, FL-net modules will be assigned smaller system numbers.
*2: If FA link modules and FL-net modules are intermixed, FL-net modules will be assigned larger system numbers.

SEE ALSO

For more details, see "FL-net (OPCN-2) Interface Module User's Manual" (IM34M6H32-02E).

A4.5 Timers (T)

There are five types of timer (T): 100-μs, 1-ms, 10-ms and 100-ms timers and a 100-ms continuous timer.
For each type of timer (T), you can assign the number of timers using the configuration function. However, you can only assign a maximum of 16 100-μs timers.

CAUTION

Do not use a timer instruction in the sensor control block or an interrupt program. The timer used will not operate correctly.

A4.5.1 100-μs, 1-ms, 10-ms, and 100-ms Timers

100-μs, 1-ms, 10-ms, and 100-ms timers are synchronized-scan, decremental timers (T) which update their current values and turn on/off their time-out relays using an end-of-scan process.

Preset values:	**100-μs timer**	**0.0001 to 3.2767 s**
	1-ms timer	**0.001 to 32.767 s**
	10-ms timer	**0.01 to 327.67 s**
	100-ms timer	**0.1 to 3276.7 s**

Each timer starts counting at the rising edge of the timer input, and expires when the current value reaches 0. When the timer (T) expires, its time-out relay turns on. The time-out relay is used for a contact **a** or **b**. The timer (T) is reset at the falling edge of the timer input and the current value returns to the timer's preset value.

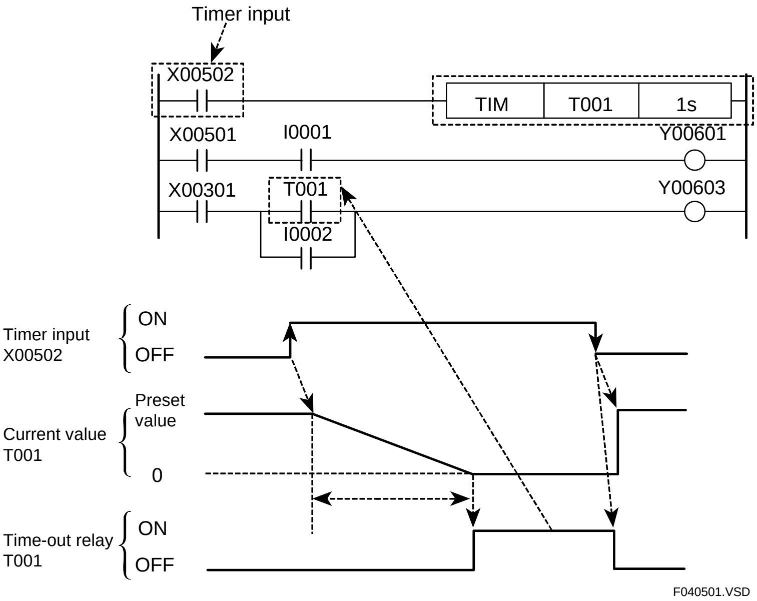


Figure A4.5.1 Timer (T)

TIP

- The preset value of a timer refers to the duration from the time the timer starts running (starting time) until the timer expires. The preset value can be specified using a Timer instruction.
- When a timer is running, its current value decreases as time passes. The current value is set to the preset value when the timer starts running, and becomes 0 when the timer expires.

A4.5.2 100-ms Continuous Timer

A 100-ms continuous timer (T) is a synchronized-scan, decremental timer which updates its current value and turns on/off its time-out relay using an end-of-scan process.

Preset value: 0.1 to 3276.7 s

The 100-ms continuous timer retains its current value and the state of its time-out relay even when its input condition is OFF. When its input condition turns ON again, the timer resumes counting from its retained value.

When its input condition turns off after the continuous timer expires, the timer (T) is reset, its current value returns to the preset value, and the time-out relay is set to OFF.

To reset a continuous timer before expiry, write “0” to the timer using a MOV instruction (MOV 0 Tnnn) when the timer input is in an OFF state.

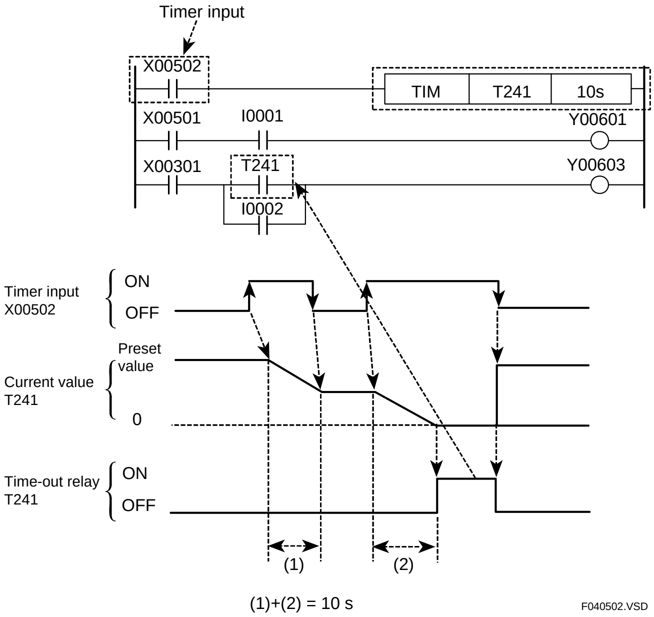


Figure A4.5.2 100-ms Continuous Timers

Using the configuration function, you can define a range of timer devices whose current values are to be latched when power is turned off. By default, all timers are non-latched.

A non-latched timer resets its current value to its preset value when you:

- power on the module;
- switch the operating mode to Run or Debug using WideField2; or
- execute a Clear Device command from WideField2.

A latched timer retains its current value even after power off and power on, and resets its current value to its preset value when you:

- execute a Clear Device command from WideField2.

A4.5.3 Selecting Timers

Configure the device range to be used for each type of 100-µs, 1-ms, 10-ms, and 100-ms timers and 100-ms continuous timers. To do so, specify the number of timers to be allocated for each timer (T) type.

The first device numbers assigned to these timers (T) satisfy the following relationship:

100-µs timer < 1-ms timer < 10-ms timer < 100-ms timer < 100-ms continuous timer

100-µs, 1-ms, 10-ms and 100-ms timers and 100-ms continuous timers are assigned device numbers of the sequence CPU, in the given order.

Table A4.5.1 Configuration of Timers

Item	F3SP66		F3SP67	
	Default	Configuration Range	Default	Configuration Range
100-µs timer	0	2048 for timers and counters combined in increments of 1; 16 max. for 100-µs timers; Timer numbers are continuous.	0	3072 for timers and counters combined in increments of 1; 16 max. for 100-µs timers; Timer numbers are continuous.
1-ms timer	0		0	
10-ms timer	512		1024	
100-ms timer	448		896	
100-ms continuous timer	64		128	

A4.6 Counters (C)

This section describes the function and operation of counters, as well as selection of counters in the configuration.

All counters are decremental counters (C) and have two types of input: count input and counter reset input.

When a counter instruction is executed, the counter decrements its current value each time it detects a rising edge in its count input and terminates when its current value reaches 0.

When the counter (C) terminates, its end-of-count relay turns on. The end-of-count relay is used for a contact a or b.

A counter (C) is reset at the rising edge of its counter reset input and its current value returns to its preset value. Count input is ignored when the counter reset input is on.

Preset value: 1 to 32767

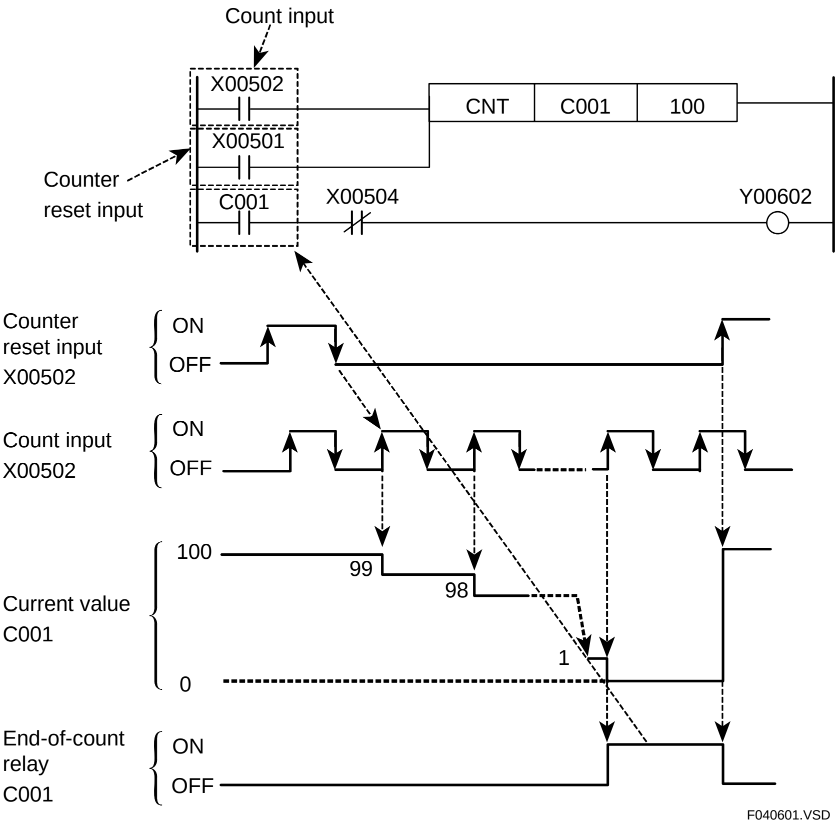


Figure A4.6.1 Counter (C)

Using the configuration function, you can define a range of counters whose current values are to be latched when power is turned off. By default, all counters are latched.

A non-latched counter resets its current value to its preset value when you:

- power on the module;
- switch the operating mode to Run or Debug using WideField2; or
- execute a Clear Device command from WideField2.

A latched counter retains its current value even after power off and power on, and resets its current value to its preset value when you:

- execute a Clear Device command from WideField2.

TIP

- A counter preset value is used by a counter as its current value when it starts counting. The counter preset value can be set using a Counter (CNT) instruction.
- When a counter is running, its current value decrements until it reaches 0, at which time the counter is said to have expired.

A4.6.1 Selecting Counters

Select the range of counters (C) to be used.

Table A4.6.1 Configuration of Counters (C)

Item	F3SP66		F3SP67	
	Default	Configuration Range	Default	Configuration Range
Timer (T)	T0001 to T1024	2048 max. for counters and timers combined in increments of 1; Timer numbers: T0001 to T2048 Counter numbers: C0001 to C2048	T0001 to T2048	3072 max. for counters and timers combined in increments of 1; Timer numbers: T0001 to T3072 Counter numbers: C0001 to C3072
Counter (C)	C0001 to C1024		C0001 to C1024	

A4.7 Data Register (D), Shared Register (R) and Extended Shared Register (R)

This section describes data registers (D), shared registers (R), extended shared register (R), and how to set the initial data.

Data registers (D) are 16-bit variables that can be used without restrictions in a program. Shared registers (R) and extended shared registers (R) are 16-bit variables that can be used for communications between CPUs in a multi-CPU system.

A4.7.1 Data Registers (D)

Data registers serve as memory for storing the results of program-based computation. Each data register has 16 bits (1 word). In programs, you can read from or write to data registers on word or long word basis using application instructions.

When you use data registers on a long word basis, the low-order 16 bits are stored in the data register with the number specified in the instruction and the high-order 16 bits are stored in the data register with that number incremented by 1.

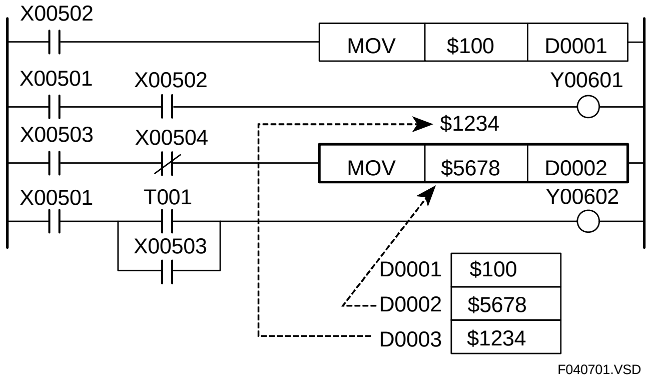


Figure A4.7.1 Data Registers (D)

Using the configuration function, you can define the data lock-up range at power failure for devices whose computation results are to be latched when power is turned off.

By default, all data registers (D) are latched.

A non-latched device will be cleared to “OFF (0)” when you:

- power on the module;
- switch the operating mode to Run or Debug using WideField2; or
- execute a Clear Device command from WideField2.

A latched device retains its computation result even after power off and power on, and is cleared to “OFF (0) ” when you:

- execute a Clear Device command from WideField2.

A4.7.2 Shared Registers (R) and Extended Shared Registers (R)

Shared registers and extended shared registers are used to exchange data between CPUs in a multi-CPU system configuration.

Shared registers (R) can be used regardless of how CPUs are combined.

Extended shared registers (R) can only be used with sequence CPU modules (F3SP25, F3SP28, F3SP35, F3SP38, F3SP53, F3SP58, F3SP59, F3SP66 and F3SP67).

In programs, you can read from or write to data registers on word or long word basis using application instructions.

When you use a long word string, the low-order 16 bits are stored in the data register with the number specified in the instruction and the high-order 16 bits are stored in the data register with that number incremented by 1.

Data can be exchanged between the local CPU module and a remote CPU module by writing the data to shared registers in the local CPU module and reading it from the remote CPU module.

If you write data to a device area not belonging to the local CPU module, information held by shared registers (R) of remote CPUs are overwritten so computation results are incorrectly reflected.

By default, no shared registers are allocated as devices. When using add-on CPU modules, configure the range of shared registers to be used. Allocate the same device range for all of the CPU modules. Otherwise, the shared registers (R) will not be correctly refreshed.

TIP

Shared and extended shared registers (R) are used to exchange data (data sharing) between CPUs in a multi-CPU system configuration between sequence CPU modules and BASIC CPU modules.

SEE ALSO

For details on BASIC CPU modules, see "BASIC CPU Modules and YM-BASIC/FA Programming Language User's Manual" (IM34M6Q22-01E).

The following figure shows an example of how shared or extended shared registers (R) are shared if you allocate shared registers R0001 to R0256 for CPU1 and shared registers R0257 to R0512 for CPU2.

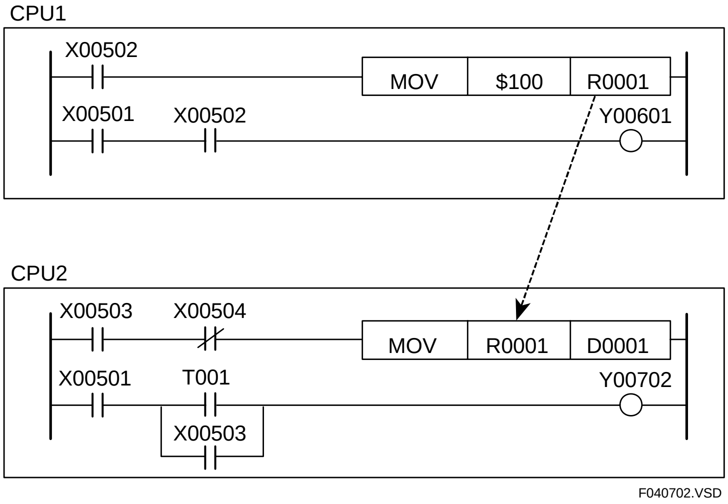


Figure A4.7.2 Shared Register (R)

Using the configuration function, you can define the data lock-up range at power failure for devices whose computation results are to be latched when power is turned off.

By default, all shared registers (R) are non-latched.

A non-latched device will be cleared to “OFF (0)” when you:

- power on the module;
- switch the operating mode to Run or Debug using WideField2; or
- execute a Clear Device command from WideField2.

A latched device retains its computation result even after power off and power on, and is cleared to “OFF (0) ” when you:

- execute a Clear Device command from WideField2.

CAUTION

When using shared or extended shared registers (R), observe the precautions given below.

(1) Index modification of shared or extended shared registers (R)

When applying index modification to a shared or extended shared register (R) of the local sequence CPU module, be careful that the register number, which is directly specified in an instruction, after adding the value of the index register, must not exceed the range specified by configuration for the local CPU. Otherwise, data held by a shared or extended shared register (R) of another CPU module is overwritten and computation result is not correctly reflected.

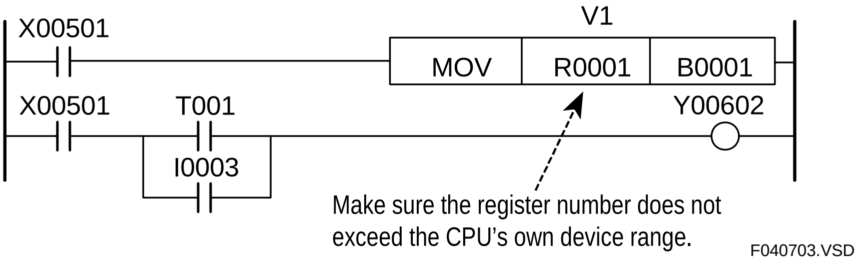

X00501
V1
MOV
R0001
B0001
X00501
T001
Y00602
I0003
Make sure the register number does not exceed the CPU’s own device range.
F040703.VSD

Figure A4.7.3 Precautions when Using Shared or Extended Shared Registers (R) (1 of 2)

(2) Block move and computation of multiple devices

When using shared or extended shared registers (R) in an instruction for transferring or computing data held by multiple devices, be careful that the range of registers, which is defined by the register number specified directly in the instruction and the number of registers included in the transfer and computation, must not exceed the range specified by configuration for the local CPU. Otherwise, information held by shared or extended shared registers (R) of another CPU is overwritten and so computation result is not correctly reflected.

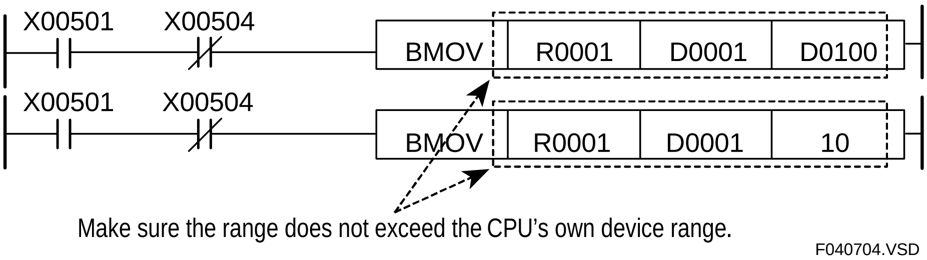


Figure A4.7.4 Precautions when Using Shared or Extended Shared Registers (R) (2 of 2)

(3) Simultaneity of data

- Using the configuration function, you can select either “Simultaneous” or “Non-simultaneous” for simultaneity of data of shared devices.
- If you select the “Simultaneous” option, simultaneity of data is guaranteed for units of devices (shared relays (E) /registers or extended shared relays (E) /registers) to be refreshed where one of the F3SP28, F3SP38, F3SP53, F3SP58, F3SP59, F3SP66 and F3SP67 sequence CPU modules is combined with any one or more of these CPUs installed as add-on CPUs. (Simultaneity of data between shared relays (E) /registers and extended shared relays (E) /registers is not guaranteed, however.)
- If any of the F3SP28, F3SP38, F3SP53, F3SP58, F3SP59, F3SP66 and F3SP67 CPUs is combined with any of the F3SP21, F3SP25, and F3SP35 sequence CPU modules, simultaneity of shared refreshed data is not guaranteed regardless of the configuration setting.
- The “Non-simultaneous” option is provided for compatibility with the F3SP21, F3SP25 and F3SP35 CPUs. Select this option when these CPUs are replaced with the F3SP28, F3SP38, F3SP53, F3SP58, F3SP59, F3SP66 and F3SP67 sequence CPU modules.

SEE ALSO

For details on index modification, see Section 1.8.1, "Index Modification" in “Sequence CPU – Instructions User's Manual” (IM34M6P12-03E).

■ Configuring Shared and Extended Shared Registers (R) for Multiple CPUs

Configure the range of shared and extended shared registers (R) to be used by each CPU in a multi-CPU system configuration where add-on CPU modules are installed. You can allocate any number of registers to each CPU in increments of 2.

Table 4.7.1 Configuration of Shared Registers (R)

Item		F3SP66, F3SP67	
		Default	Configuration Range
Device Capacities	Shared register (R)	0	1024 max. for all CPUs combined in increments of 2
	Extended shared register (R)	0	3072 max. for all CPUs combined in increments of 2

Extended shared registers (R) can only be used with sequence CPU modules (F3SP25, F3SP28, F3SP35, F3SP38, F3SP53, F3SP58, F3SP59, F3SP66 and F4SP67).

CAUTION

Assign the same range of shared and extended shared registers (R) for all CPU modules. No error will result, however, even if the range is not the same among the CPU modules. Rather, data in a remote CPU module may appear wrongly assigned to register numbers or data in the local CPU module may appear that way when referenced from the remote CPU.

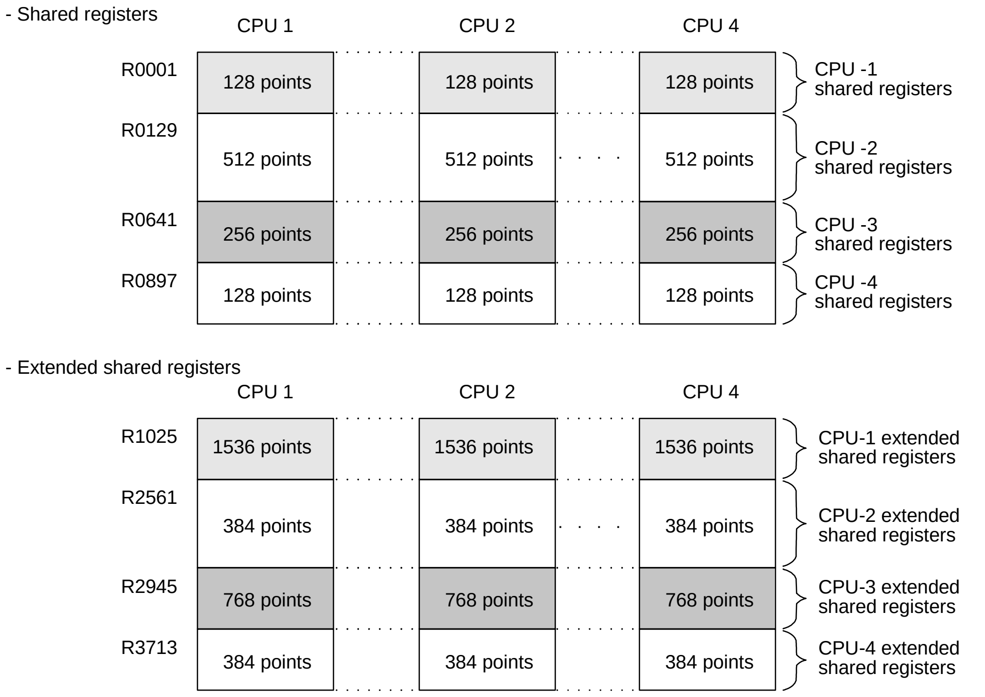


Figure A4.7.5 Example of Shared and Extended Shared Register (R) Allocation when Four Sequence CPU Modules are Installed

CAUTION

Even if the specified range includes less than 1024 shared registers (R), the extended shared registers (R) always begin with the number R1025.

A4.7.3 Setting Initial Data for Data Registers (D)

Using the configuration function, define the initial values of data registers (D) to be used at the beginning of program execution.

Specify the starting number and the number of data registers to be configured, followed by the initial data values. After this configuration, the preset initial data values are stored in the specified data registers when the program starts. This configuration is useful when a large volume of initial data needs to be set by a program or when the initial data needs to be saved. You can set initial data in a maximum of 1024 data registers.

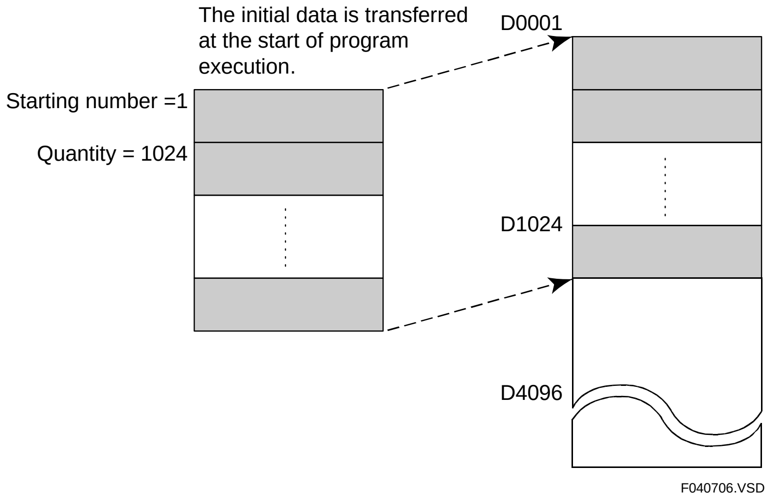

The initial data is transferred at the start of program execution.
Starting number =1
Quantity = 1024
D0001
D1024
D4096
F040706.VSD

Figure A4.7.6 Setting Initial Data for Data Registers (D)

A4.8 Special Registers (Z)

Special registers have specific functions, such as indicating the internal state of a programmable controller or indicating errors.

A4.8.1 Sequence Operation Status Registers

Sequence operation status registers indicate the status of sequence operation.

Table A4.8.1 Sequence Operation Status Registers

No.	Name	Function	Description
Z001	Scan Time (Run mode)	Latest scan time	Stores the latest scan time in 100 µs increments.
Z002	Minimum Scan Time (Run mode)	Minimum scan time	Allows the latest scan time to be read in 100 µs increments if it is shorter than the minimum scan time.
Z003	Maximum Scan Time (Run mode)	Maximum scan time.	Allows the latest scan time to be read in 100 µs increments if it is longer than the maximum scan time.
Z004	Scan Time (Debug mode)	Latest scan time	Stores the latest scan time in 100 µs increments.
Z005	Minimum Scan Time (Debug mode)	Minimum scan time	Allows the latest scan time to be read in 100 µs increments if it is shorter than the minimum scan time.
Z006	Maximum Scan Time (Debug mode)	Maximum scan time.	Allows the latest scan time to be read in 100 µs increments if it is longer than the maximum scan time.
Z007	Peripheral-process Scan Time	Latest scan time	Stores the latest scan time in 100 µs increments. (Tolerance: Scan time of one control process)
Z008	Minimum Peripheral-process Scan Time	Minimum scan time	Allows the latest scan time to be read in 100 µs increments if it is shorter than the minimum scan time. (Tolerance: Scan time of one control process)
Z009	Maximum Peripheral-process Scan Time	Maximum scan time.	Allows the latest scan time to be read in 100 µs increments if it is longer than the maximum scan time. (Tolerance: Scan time of one control process)

CAUTION

- Do not write to a special register (Z), including those not listed in the table above (e.g., Z010 to Z016), unless it is marked as "write-enabled". Special registers are used by the sequence CPU module. Writing to these registers incorrectly may lead to system shutdown or other failures.
- Special registers (Z) with index modification cannot be specified as destinations for data output and if specified, will cause instruction processing errors during execution.

CAUTION

Special registers (Z) cannot be specified as output destinations in block transfer and table output ladder instructions, and if specified, will cause instruction processing error during execution.

- Block transfer instructions: Block Move (BMOV), Block Set (BSET), String Move (SMOV), etc.
- Table output instructions: Read User Log (ULOGR), FIFO Write (FIFWR), etc.

A4.8.2 Self-diagnosis Status Registers

Self-diagnosis status registers indicate the results of self-diagnostics by the sequence CPU.

Table A4.8.2 Self-diagnosis Status Registers

Category	Self-diagnosis Status Registers		
No.	Name	Function	Description
Z017	Self-diagnosis error	Self-diagnosis error No.	Stores the results of self-diagnosis.[*1]
Z018		Self-diagnosis error block No.	
Z019		Self-diagnosis error instruction No.	
Z022	Instruction processing error	Instruction processing error No.	Stores errors detected during instruction processing.[*1]
Z023		Instruction processing error block No.	
Z024		Instruction processing error instruction No.	
Z027	I/O comparison error	I/O comparison error No.	Stores detailed information on I/O comparison error. [*1]
Z028		I/O comparison error block No.	
Z029		I/O comparison error instruction No.	
Z033 to Z040	I/O error	Slot no. with I/O error 16 2 1 0 ····· 1 0	Stores, as a bit pattern, slot numbers where an I/O error is detected. Z033: Main unit Z034: Subunit 1 Z035: Subunit 2 Z036: Subunit 3 Z037: Subunit 4 Z038: Subunit 5 Z039: Subunit 6 Z040: Subunit 7
Z041	Module recognition	Main unit	Slot number 16 1 0 ····· 1 0 0: No modules are recognized. Unable to read/write. 1: Modules are recognized.
Z042		Subunit 1	
Z043		Subunit 2	
Z044		Subunit 3	
Z045		Subunit 4	
Z046		Subunit 5	
Z047		Subunit 6	
Z048		Subunit 7	
Z089	Communication error slot	Main unit	Slot number 16 1 0 ····· 1 0 Fiber-optic FA-bus module 0: Normal transmission line; Unspecified transmission line; or Loaded with a wrong module 1: Abnormal transmission line (Failure or switchover in transmission line)
Z090		Subunit 1	
Z091		Subunit 2	
Z092		Subunit 3	
Z093		Subunit 4	
Z094		Subunit 5	
Z095		Subunit 6	
Z096		Subunit 7	

*1: For details on error codes stored in these special registers, see Chapter A8, "RAS Features".

SEE ALSO

For details on special registers Z089 to Z096 (Communication error slot), see "Fiber-optic FA-bus Module and Fiber-optic FA-bus Type 2 Module User's Manual" (IM34M6H45-01E).

A4.8.3 Utility Registers

Table A4.8.3 Utility Registers

No.	Name	Function	Description
Z049 (write-enabled)	Clock Data	Last two digits of calendar year	Stores "year" as a BCD-coded value. e.g. 1999 as $0099 2000 as $0000
Z050 (write-enabled)		Month	Stores "month" as a BCD-coded value. e.g. January as $0001
Z051 (write-enabled)		Day of month	Stores "day of month" as a BCD-coded value. e.g. 28th as $0028
Z052 (write-enabled)		Hour	Stores "hour" as a BCD-coded value. e.g. 18:00 hours as $0018
Z053 (write-enabled)		Minute	Stores "minute" as a BCD-coded value. e.g. 15 minutes as $0015
Z054 (write-enabled)		Second	Stores "second" as a BCD-coded value. e.g. 30 seconds as $0030
Z055		Day of week ($0000 to $0006)	Stores "day of week" as a BCD-coded value. e.g. Wednesday as $0003
Z056	Constant Scan Time	Value of constant scan time	0.1 ms increments e.g. 10 ms as 100
Z057	Constant Scan Time	Value of constant scan time	1 ms increments e.g. 10 ms as 10
Z058	Scan Monitoring Time	Value of scan monitoring time	1 ms increments e.g. 200 ms as 200

You can set clock data using the Set Date instruction (DATE), Set Time instruction (DATE), Set Date String instruction (SDATE), and Set Time String instruction (STIME).

To set clock data by directly using special registers, use the following procedure.

1. Write the clock data to special registers Z049 to Z054
   (use a MOV P instruction. Using BMOV or BSET instructions will generate an instruction error).
2. Set special relay M172 to ON within the same scan as step (1)
   (use a DIFU instruction).
3. Set special relay M172 to OFF in the scan subsequent to step (2).
   Stop writing the clock data to special registers Z049 to Z054 in the same scan.

Note that no change will be made to clock data, which reverts to its original value if the setup value is invalid.

The accuracy of clock data is specified as:

- Maximum daily error = ±8 s (±2 s, when actually measured)

The clock accuracy is reset to the maximum daily error of -1.2 s/+2 s, however, when the power is turned off and on again. In addition, you can input a correction value from the programming tool. If you specify an appropriate correction value, the clock data is corrected during the power-off-and-on sequence, thus offsetting the cumulative error.

A4.8.4 FA Link Module Status Registers

FA Link module status registers indicate the status of FA links.

Table A4.8.4 FA Link Module Status Registers

No.	Name	Function [*1]	Description
Z075	Local Station No.		System 1 (FA link)
Z076	Local Station No.		System 2 (FA link)
Z077	Local Station No.		System 3 (FA link)
Z078	Local Station No.		System 4 (FA link)
Z079	Local Station No.		System 5 (FA link)
Z080	Local Station No.		System 6 (FA link)
Z081	Local Station No.		System 7 (FA link)
Z082	Local Station No.		System 8 (FA link)
Z065	Local Station Status	0: Initialization in progress 1: Offline 2: Online	System 1 (FA link)
Z066	Cyclic Transmission Time		System 1 (FA link) 1 ms increments
Z070	Local Station Status	0: Initialization in progress 1: Offline 2: Online	System 2 (FA link)
Z071	Cyclic Transmission Time		System 2 (FA link) 1 ms increments
Z257	Local Station Status	0: Initialization in progress 1: Offline 2: Online	System 3 (FA link)
Z258	Cyclic Transmission Time		System 3 (FA link) 1 ms increments
Z262	Local Station Status	0: Initialization in progress 1: Offline 2: Online	System 4 (FA link)
Z263	Cyclic Transmission Time		System 4 (FA link) 1 ms increments
Z267	Local Station Status	0: Initialization in progress 1: Offline 2: Online	System 5 (FA link)
Z268	Cyclic Transmission Time		System 5 (FA link) 1 ms increments
Z272	Local Station Status	0: Initialization in progress 1: Offline 2: Online	System 6 (FA link)
Z273	Cyclic Transmission Time		System 6 (FA link) 1 ms increments
Z277	Local Station Status	0: Initialization in progress 1: Offline 2: Online	System 7 (FA link)
Z278	Cyclic Transmission Time		System 7 (FA link) 1 ms increments
Z282	Local Station Status	0: Initialization in progress 1: Offline 2: Online	System 8 (FA link)
Z283	Cyclic Transmission Time		System 8 (FA link) 1 ms increments

*1: For details on the FA link module status registers, see Special relays/registers sections in "FA Link H Module Fiber-optic FA Link H Module User's Manual" (IM34M5H43-01E).

TIP

Units that make up a system are known as stations.

A4.8.5 Sequence CPU Module Status Registers

CPU module status registers indicate the status of a CPU.

Table A4.8.5 Sequence CPU Module Status Registers

No.	Name	Function	Description
Z105	Number of User Log Records		See Section A6.14, "User Log Management Function" for details on user logs.
Z109	Sensor CB Execution Time	Time taken from starting of input refreshing for the sensor control block through program execution to completion of output refreshing. (Unit: 10 µs)	
Z111	Maximum Sensor CB Execution Time	The maximum time taken to execute the sensor control block. (Unit: 10 µs)	
Z121 to Z128 [*1]	Model Information	CPU model name and revision number of firmware.	

*1: For module ”F3SP67-6S” with firmware Rev1,

- Z121 “F3”
- Z122 “SP”
- Z123 “67”
- Z124 “6S”
- Z125 “/R”
- Z126 “01”
- Z127 “/ “
- Z128 “ “

A4.9 Index Registers (V)

Index registers are used to modify devices numbers.
You can use these registers in both basic instructions and application instructions to make index modifications.
Use these registers to address a device by adding the content of an index register to a device number, which is directly specified in an instruction.

SEE ALSO

For details on index modification, see Section 1.8.1, "Index Modification" in “Sequence CPU – Instructions User's Manual” (IM34M6P12-03E).

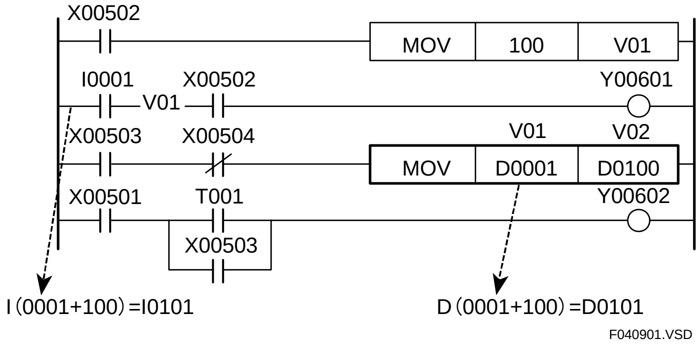


Figure A4.9.1 Index Registers

CAUTION

The sequence CPU module performs no check on whether an index modified device exceeds the device configuration range. If an index register is incorrectly specified, the device configuration range may be exceeded, resulting in inadvertent selection of a different type of device.

CAUTION

An index register can be set to any value between -32768 and 32767. Therefore, for devices such as file registers (B) whose size is larger than 32768, index modification cannot cover the entire device.

A4.10 File Registers (B)

File registers (B) are used as extensions of data registers (D).
Each file register consists of one word.
Like data registers (D), you can read from or write to file registers on a word basis or 32-bit basis using application instructions.

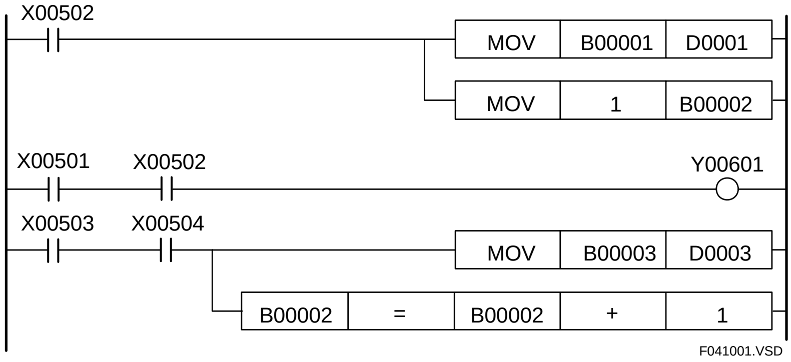


Figure A4.10.1 File Registers (B)

Unlike data registers (D), all file registers (B) retain their computation results when the power is turned off. A file register is cleared to OFF (0) if you:

- write the data value OFF (0) to the file register (B) using the programming tool WideField2.

Unlike data registers (D), file registers are not cleared to OFF (0) even if you:

- execute a Clear Device command from WideField2; or
- clear the memory from Widefield2.

A5. Programs

This chapter describes languages used for programming, program types and program memory.

A5.1 Programming Language

Two types of programming language are available: structured ladder language and mnemonic language. In either case, the written program is read sequentially by the sequence CPU to perform computations according to the program's process details.

A5.1.1 Structured Ladder Language

The structured ladder language is based on relay symbol representation and allows a programmer to do structured programming by breaking a program into functional parts.

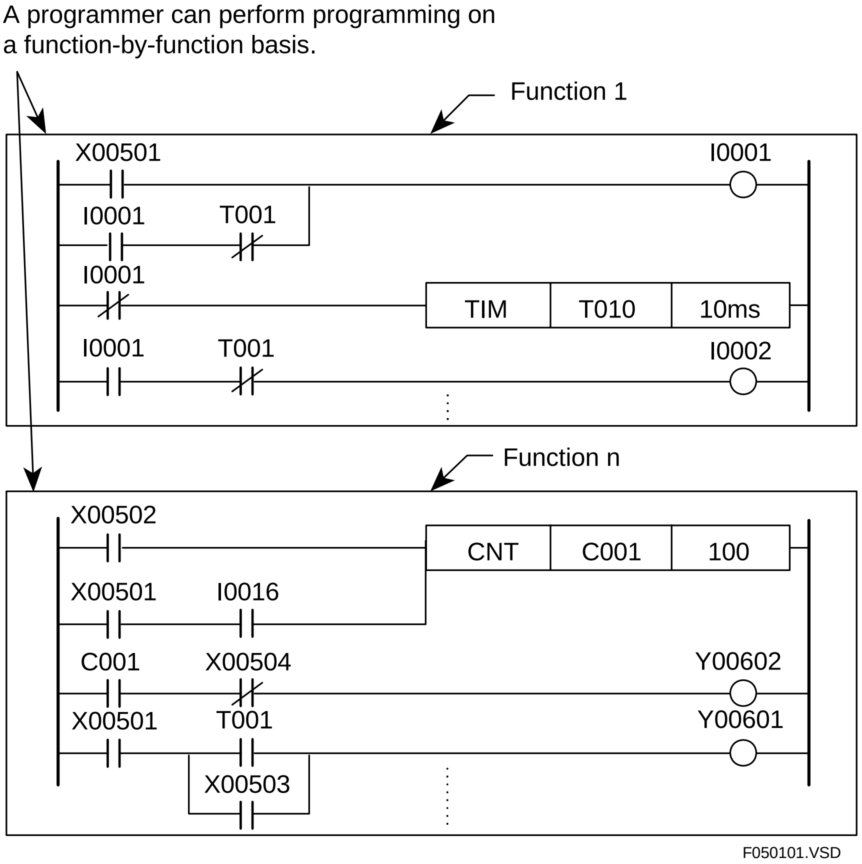


Figure A5.1.1 Structured Ladder Language

A5.2 Program Types and Configuration

There are two types of programs: blocks and executable programs.

A5.2.1 Blocks and Executable Programs

■ Blocks

A block refers to a collection of circuits entered using WideField2.

Parts of a program written on a function-by-function basis using the structured ladder language or mnemonic language are managed as blocks. As a program can be maintained or reused on block basis, program development becomes easier.

The module allows up to 56K steps per block.

CAUTION

An individual block cannot be executed by a CPU.

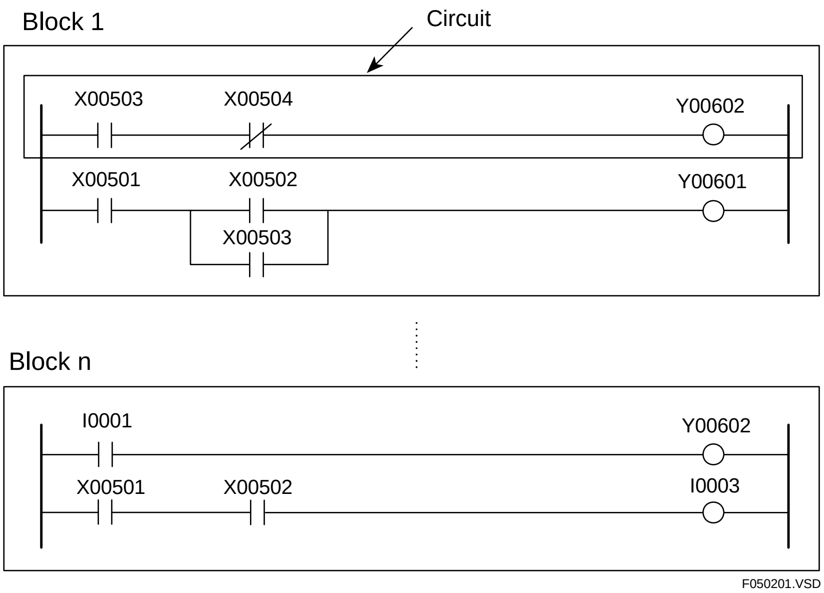


Figure A5.2.1 Blocks

■ Executable Program

An executable program refers to a program, which is stored in a format that allows it to be executed by the CPU. An executable program is composed by combining multiple blocks created using Widefield2. Each executable program can contain a maximum of 1024 blocks.

You can either execute all or selected blocks of an executable program. This simplifies program management.

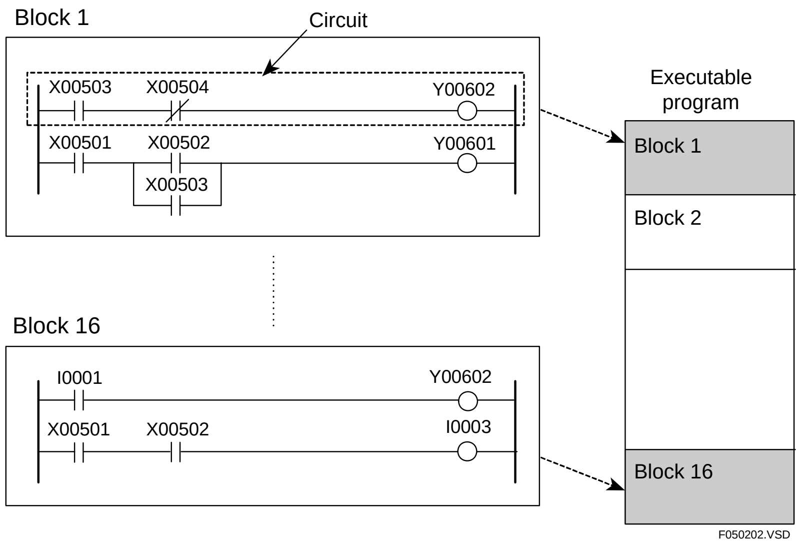


Figure A5.2.2 Example of an Executable Program

A5.2.2 Component Programs of an Executable Program

An executable program contains a maximum of 1024 blocks. The sensor control block is regarded as a single, separate block. Programs that compose an executable program are classified into main routine programs, subroutine programs, interrupt programs and sensor control block programs, according to their functions.

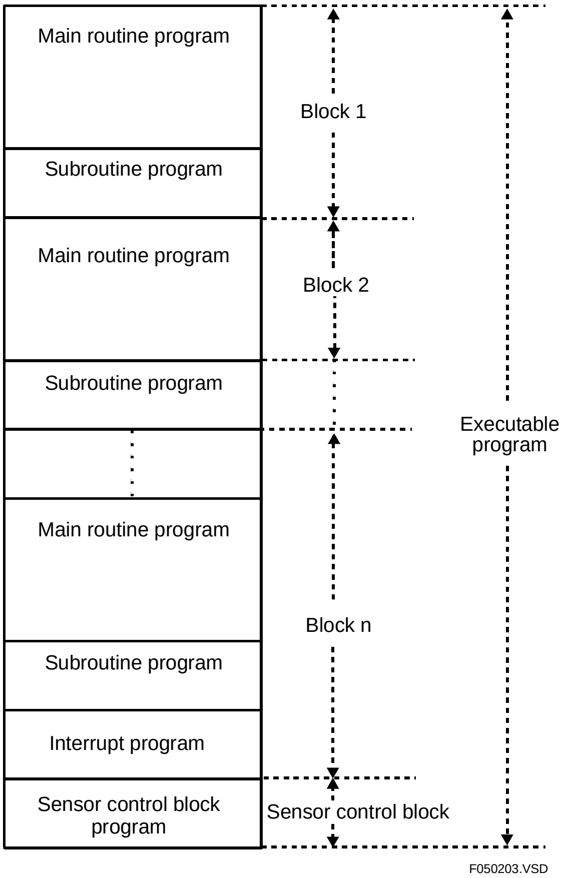


Figure A5.2.3 Component Programs of an Executable Program

■ Main Routine Program

A main routine program is always executed in each scan.

The main routine program is written using structured ladder language, and is composed of multiple blocks.

You can execute a main routine program by either executing all blocks of the program or executing only specified blocks.

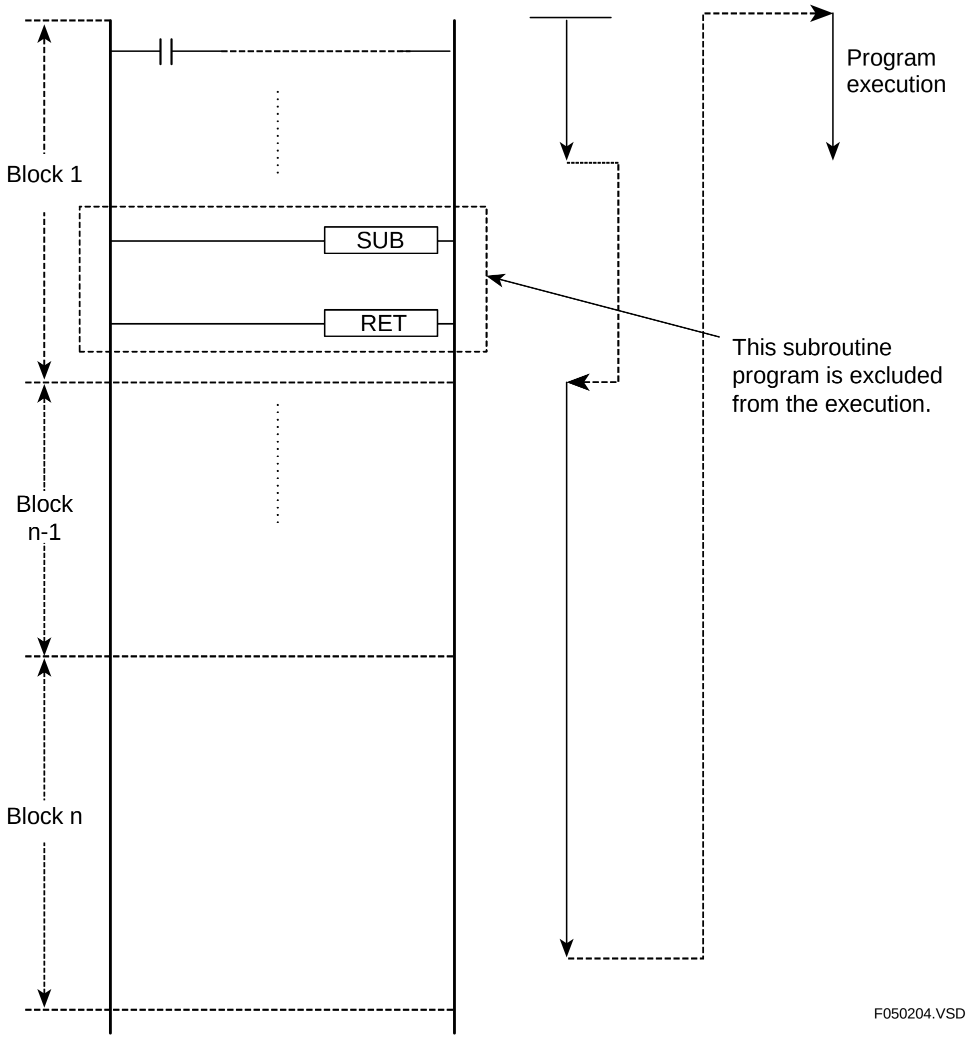


Figure A5.2.4 How a Main Routine Program Is Executed

■ Subroutine Program

A subroutine program is executed when a main routine program executes a CALL instruction. Use a subroutine program when you want to run a specific process two or more times within one scan. A subroutine program can be placed in any location in a block.

In the case where specified blocks are selected for execution, a subroutine program, which is called from a block being executed, will be executed even if it is located in a block, which is not selected for execution.

Subroutine program calls can be nested up to eight levels deep. (To nest a call is to call a subroutine from within another subroutine).

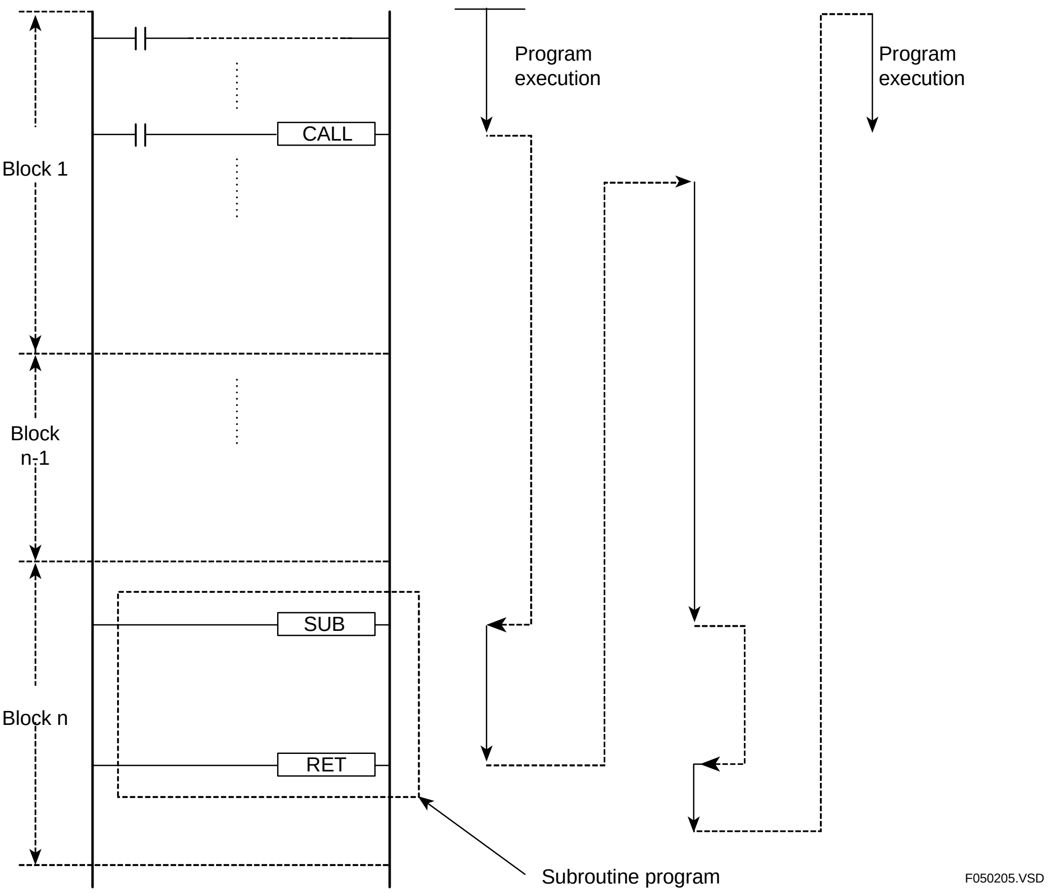


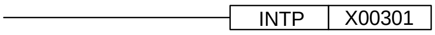
Figure A5.2.5 How a Subroutine Program Is Executed

■ Interrupt Programs

An interrupt program is executed when any cause of interrupt occurs.

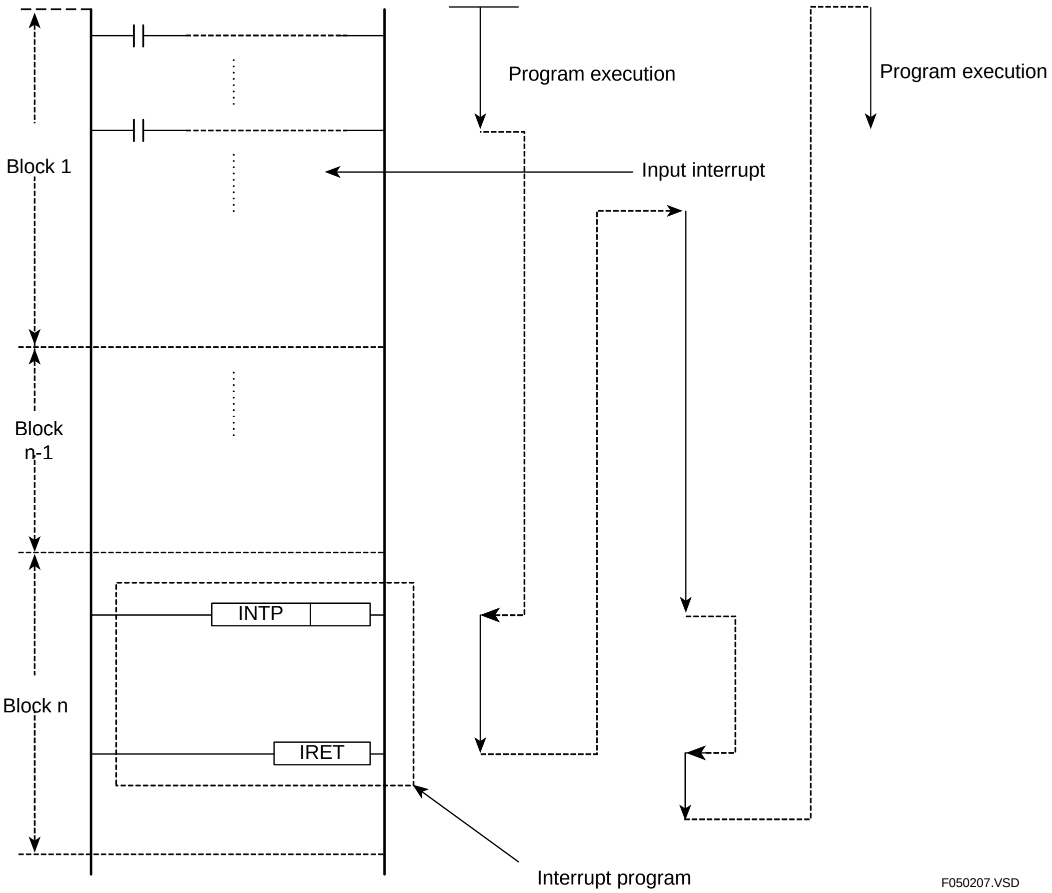
A maximum of four interrupt programs can be included in a program.

The relationship between a cause of interrupt and an interrupt program is described as a parameter of the Interrupt (INTP) Instruction.



F050206.VSD

Figure A5.2.6 INTP Instruction



F050207.VSD

Figure A5.2.7 How an Interrupt Program Is Executed

■ Sensor Control Block

The sensor control block (SCB) is one block, which is executed at high-speed and at fixed intervals, separately from the normal scan.

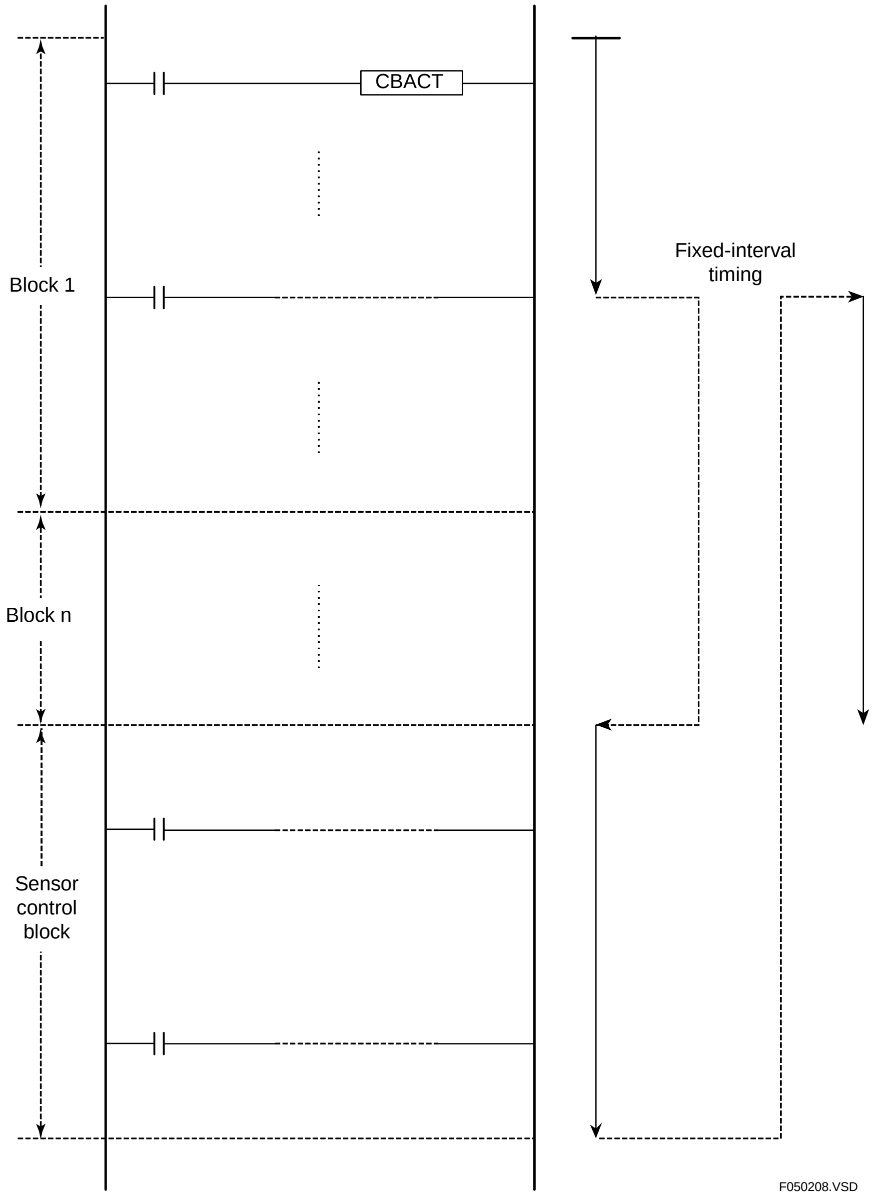


Figure A5.2.8 How the Sensor Control Block Is Executed

A5.3 Program Memory

The program memory contains programs as well as information required for program execution and management. This section describes the structure of the program memory and its initial state with no program.

Table A5.3.1 Structure of Program Memory and Its Initial State

Component	Description	Initial State
Program management table	An area for storing information required for managing all programs including program name, step count, and block management information.	In the initial state, the program name is "PROGRAM, the " block name is "PROGRAM," and the number of steps is zero.
Program	An area for storing programs.	Contains a NOP instruction.
Configuration table*1	An area for storing configuration information, such as device capacities and operation methods.	Contains the initial values.
I/O configuration table*1	An area for storing configuration information such as I/O module setup and output mode (Hold/reset) in case sequence stops.	Contains the initial values.
Program control instruction table	An area for storing information required for managing the execution of program control instructions, such as JMP instructions and subroutine instructions.	Contains "0," indicating that there is no program control instructions such as JMP or subroutine instructions.
Timer/counter preset value table	An area for storing timer and counter preset values.	Contains "0," indicating that there are neither timers nor counters.
Utility	An area for storing information such as circuit comments and subcomments.	Contains "0".

*1: For details, see Section A9.2, “Configuration Items.”

CAUTION

No program can be executed when the program memory is in its initial state.

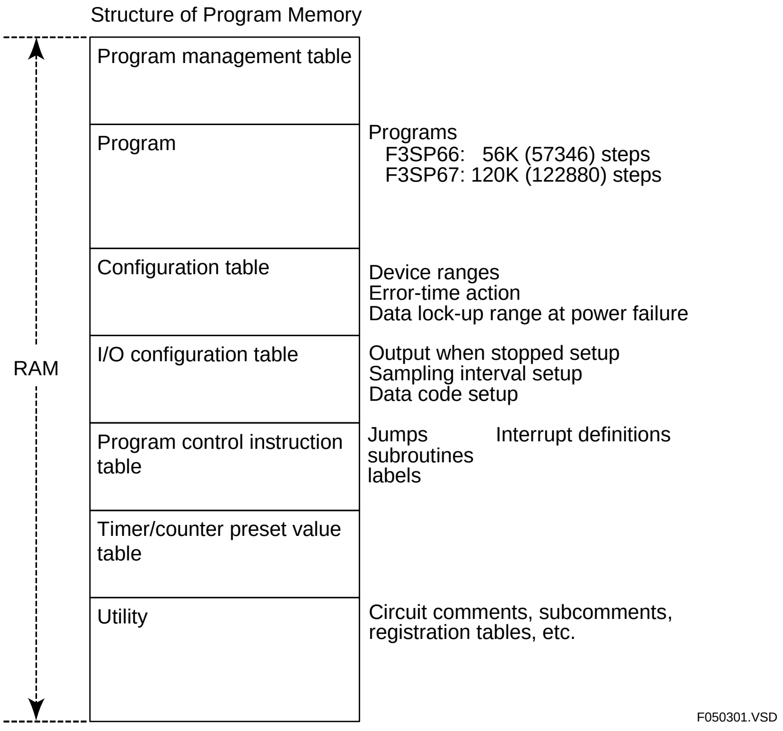

Structure of Program Memory
RAM
Program management table
Program
Programs
F3SP66: 56K (57346) steps
F3SP67: 120K (122880) steps
Configuration table
Device ranges
Error-time action
Data lock-up range at power failure
I/O configuration table
Output when stopped setup
Sampling interval setup
Data code setup
Program control instruction table
Jumps
subroutines
labels
Interrupt definitions
Timer/counter preset value table
Utility
Circuit comments, subcomments, registration tables, etc.
F050301.VSD

Figure A5.3.1 Structure of Program Memory

A6. Functions

This chapter describes the functions provided by the sequence CPU module, such as the execution of specified blocks and debugging operations.

A6.1 Function List

The following tables summarize the functions provided by the sequence CPU module and add-on CPU modules.

Table A6.1.1 Functions Provided by Sequence CPU Modules and Add-on CPU Modules

Function of Sequence CPU Module	Function Overview	Section
Operation setup function	Specifies the operation mode of the sequence CPU module and its actions.	A6.2
Constant scan	Executes a sequence program at certain time intervals.	A6.3
Executing all blocks/specified blocks	Specifies how an executable program is processed. Specified blocks are executed using ACT and INACT instructions.	A6.4
Debugging functions	Functions that support debugging, such as forced set/reset.	A6.5
Program protection	Protects programs by means of password. This function has two modes: executable program protection and block protection.	A6.6
Online editing	Makes on-line modifications or changes to a program in the program memory of the sequence CPU module.	A6.7
Making programs and data resident in "ROM"	Enables programs and data to be stored on portable non-volatile "ROM" memory for backup or other purposes.	A6.8
Exclusive access control	Prohibits operations on program, operating mode, or device data by other users during operation or debugging.	A6.9
Sampling trace function	Acquires and displays states of multiple devices for up to 1024 scans.	A6.10
Personal computer link function	Performs communications equivalent to that of a personal computer link module, when a personal computer or a monitor is connected to the SIO port.	A6.11
Macro instructions	Allows the user to create and register new, customized instructions.	A6.13
User log management function	Allows the user to keep a log of, or record of, errors in the user's system, the way they occurred, the system's operating condition, and so on.	A6.14
Sensor control function	Executes a single block at high speed and at fixed intervals separately from the normal scan.	A6.15
Partial download function	Downloads specified blocks or macros only.	A6.16
Function for storing comments to CPU	Stores circuit comments and subcomments to a sequence CPU module.	A6.17
Function for storing tag name definitions to CPU	Stores tag name definitions to a sequence CPU module.	A6.18
Structures	Represents a group of data items under a unified name.	A6.19
Constant definition function (header file)	Allows constants (word, long word, floating-point, character string and binary constants) to be defined so that defined constant names can be coded in programs in place of constant values.	A6.20
Telegram processing functions	Allows binary data to be included in character strings using the M3 escape sequence function.	A6.21
Log function	The system log function records module error and other events. Other log functions are also available.	A6.22
Security functions	Security related functions including protection of program assets against unauthorized access and function removal for preventing incorrect user operation.	A6.23

Table A6.1.2 Device Management Functions

Device Management Function	Function Overview	Section
Upload device data	Reads device information (data) from the sequence CPU module and saves it to a WideField2 file.	A6.12
Download device data	Reads device information (data) from a WideField2 file and writes it to the sequence CPU module.	
Edit device data	Edits device information (data) saved in a WideField2 file.	
Compare device data	Compares device information saved in the sequence CPU module with that saved in a WideField2 file.	

A6.2 Operation Setup Function

The operation setup function sets up the sequence CPU module operating mode and initializes programs and devices. You can set up operation by issuing a command from WideField2, personal computer link module, or an add-on CPU module.

■ Run Mode

In Run mode, the CPU begins running a program from its first instruction, similarly to when the power is turned on. When the power is turned on or the operating mode is changed from Stop mode to Run mode, the CPU sets all devices to 0, except for latching-type devices, before executing the program. When the CPU switches to Run mode, functions that are available only in Debug or Stop mode are disabled.

■ Debug Mode

In Debug mode, the CPU begins running a program from its first instruction, similarly to when the power is turned on. When the operating mode is changed from Stop mode to Debug mode, the CPU sets all devices to 0, except for latching-type devices, before executing the program. Be sure to exit from Debug mode and enter Run mode after debugging and tuning.

■ Stop Mode

In Stop mode, the CPU stops running the program. The CPU either hold or reset external outputs depending on the settings of the configuration item “Output when stopped.” This function does not work when the CPU has already stopped running the program.

Stop

■ Clear Memory

This function deletes a program or programs and sets all devices except file registers (B) to 0.

You must stop running the program before using this function.

Stop

■ Clear Devices

This function sets all latching-type devices except file registers (B) to 0.

You must stop running the program before using this function. To clear file registers (B), use the edit device function of the device management function to set all the file register (B) data to 0 and then write the data to the sequence CPU module using the write device function.

SEE ALSO

For details on the device management function, see Section A6.12 “Device Management Function.”

CAUTION

Observe the following precautions when using the functions described in this chapter:

- Some functions are only available in some but not all of the operating modes.
- The following marks are used when explaining a function to indicate that the function is available in the cited mode or modes.

  Run Debug Stop

  If no mark is indicated, it means that the function can be used in all operating modes.
- Some functions may lengthen the scan time.

  Be sure to disable such functions after use and before actual operation.

  Be especially careful when using any function that is enabled in Debug mode. Always disable the function and enter Run mode after debugging and tuning.

A6.3 Constant Scan

Run Debug

The constant scan function executes a program repeatedly at certain time intervals.
You can set the constant scan time, i.e., constant-scan time interval, to a value between 1 ms and 190 ms in 0.1 ms increments using the configuration function.

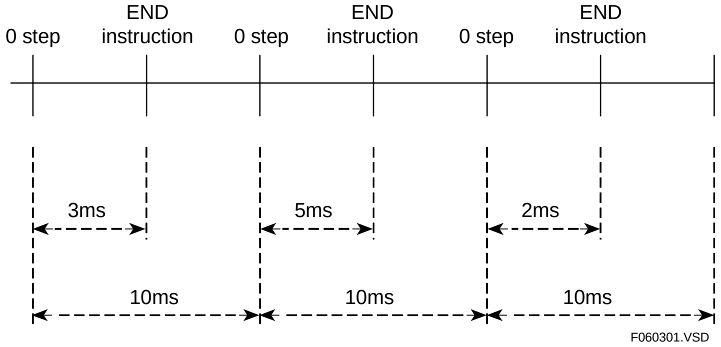


Figure A6.3.1 Operation Based on 10-ms Constant Scan

If the scan time of a sequence program is longer than the preset constant scan time, the constant scan setting is ignored and the program is executed using its own scan time.

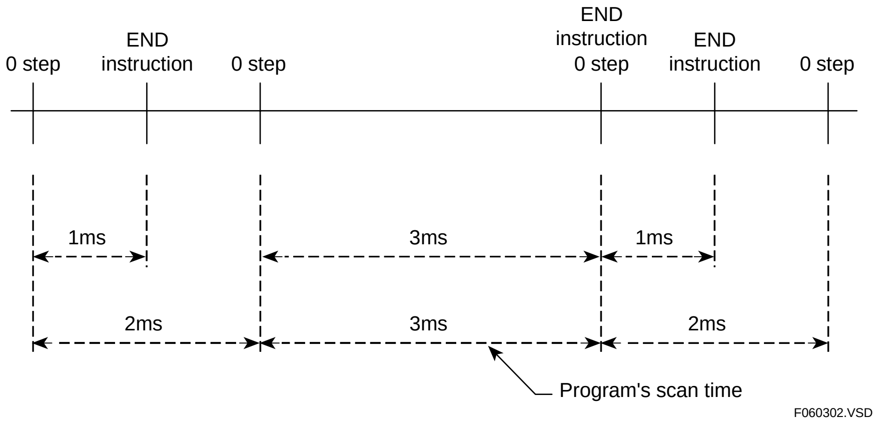


Figure A6.3.2 Operation Based on 2-ms Constant Scan

A6.3.1 Setting the Constant Scan Time

You can set the constant scan time using “Operation Control” of configuration of WideField2.

You can set the constant scan time to a value between 1 ms and 190 ms in 0.1-ms increments. To disable constant scan, select the option “Do not use” (default).

CAUTION

- The constant scan time must be shorter than the scan timeout interval.
- If the constant scan time is longer than the scan timeout interval, a scan timeout error occurs.

A6.4 Executing All Blocks/Specified Blocks

Run Debug

Select the program execution mode ("All Blocks" or "Specified Blocks") using “Operation Control” of configuration of WideField2.

A6.4.1 Executing All Blocks

This mode executes all blocks of an executable program sequentially from block 1. The default program execution mode is “All Blocks.”

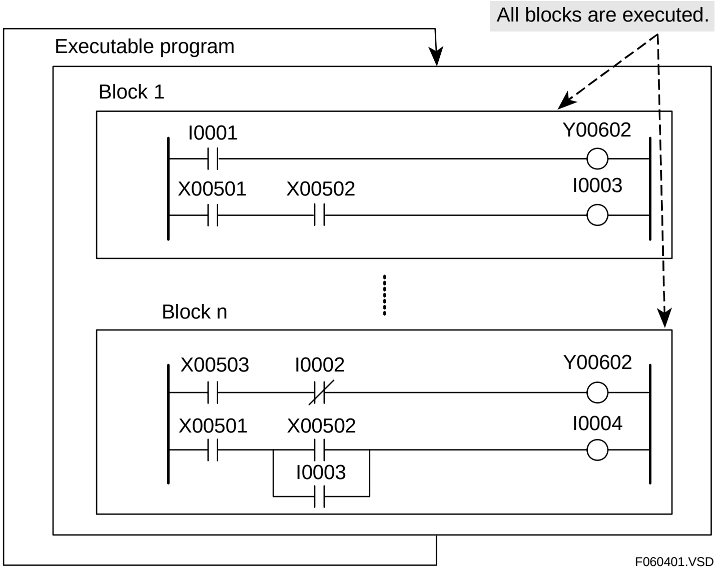


Figure A6.4.1 Executing All Blocks

A6.4.2 Executing Specified Blocks

This mode allows you to specify selected blocks of an executable program for execution using ACT/INACT instructions.

In this way, you can control the execution of blocks, which are created on per function basis in modular programming.

Blocks to be executed are said to be "active" while blocks not to be executed are said to be "inactive." Use an ACT instruction to activate a block and an INACT instruction to inactivate a block. Whether each block is active or inactive is indicated by a special relay (M) given below:

- Special relays M2001 to M3024 for blocks 1 to 1024.

  (Note that special relays M0001 to M0032 have the same values as special relays M2001 to M2032.)

The special relay for a block is set to “1” when the block is active and “0” when the block is inactive.

Active blocks are executed in ascending order of their block numbers. By default, only block 1 is active.

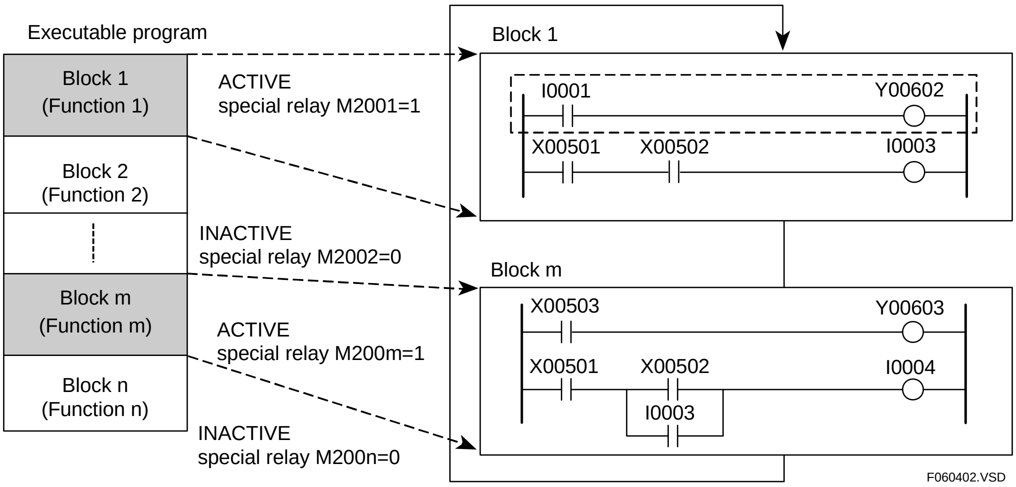


Figure A6.4.2 Execution of Specified Blocks (Executing block 1 and block m only)

A6.4.3 Operation When Specified Blocks Are Activated

A block that is specified for activation by an ACT instruction is initialized at the end of that scan, and is actually started in the next scan.

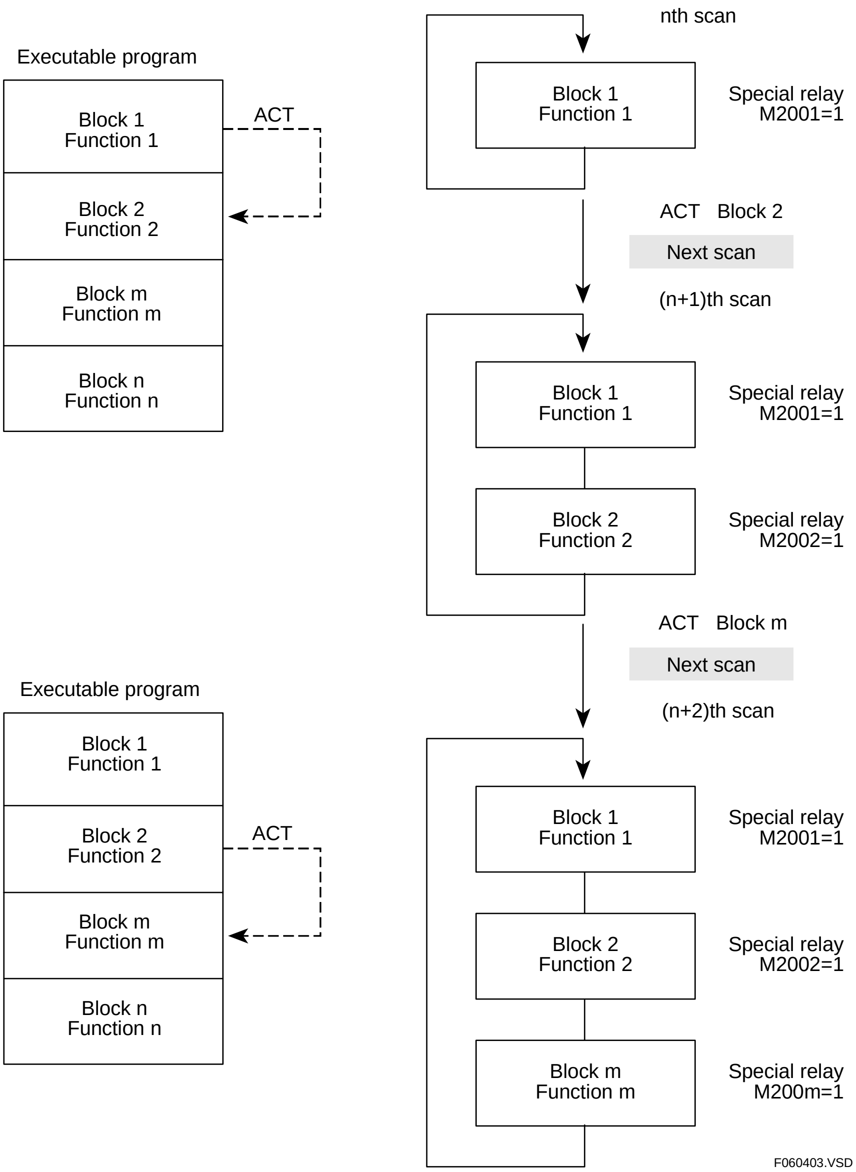


Figure A6.4.3 Operation When Specified Blocks are Activated

Devices that are used in a block, which is activated by an ACT instruction, are put into the following states by block initialization.

Table A6.4.1 State at Block Activation

Device	State at Block Activation
Timer (T)	Resets.
Continuous timer	Retains the value held before block activation.
Counter (C)	Retains the value held before block activation.
Destination of OUT instruction	Goes into an OFF state.
All other devices	Retains the states held before block activation.

Use a SET instruction for a device in a block whose output value is to be retained when the block is activated.

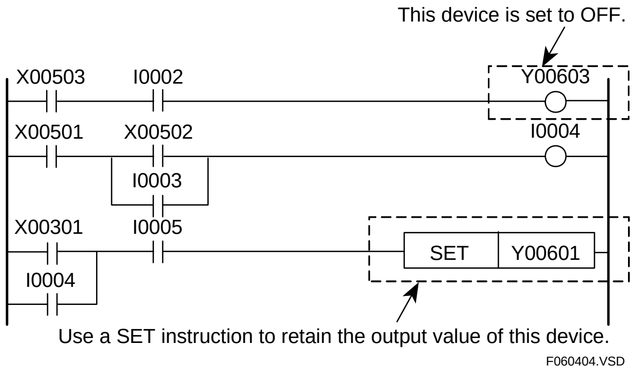

This device is set to OFF.
X00503
I0002
Y00603
X00501
X00502
I0004
I0003
X00301
I0005
SET
Y00601
I0004
Use a SET instruction to retain the output value of this device.
F060404.VSD

Figure A6.4.4 Example of Devices Initialized When a Block is Started

A6.4.4 Operation When Specified Blocks Are Inactivated

A block that is specified for inactivation by an INACT instruction is initialized at the end of that scan, and is actually started in the next scan.

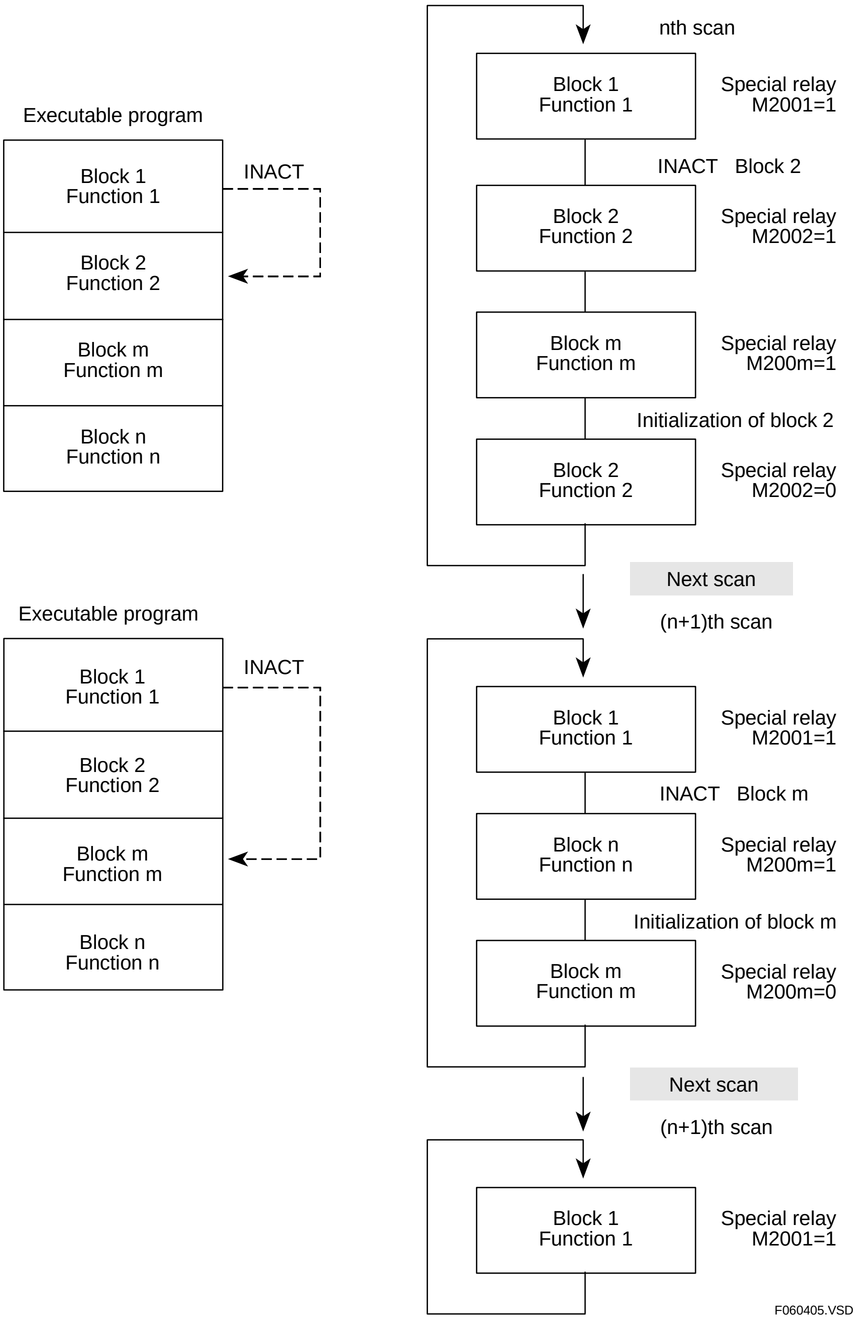


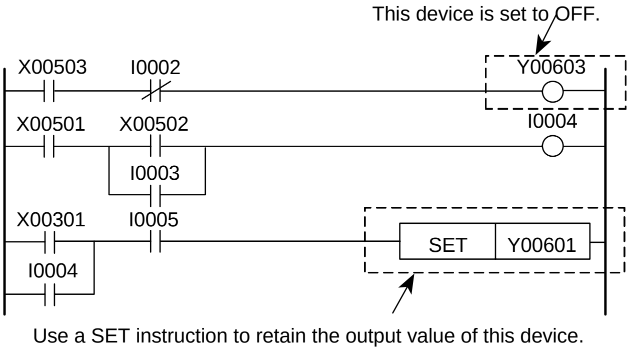
Figure A6.4.5 Operation When Specified Blocks Are Inactivated

Devices that are used in a block, which is stopped by an INACT instruction, are put into the following states by block initialization.

Table A6.4.2 State at Block Inactivation

Device	State at Block Inactivation
Timer (T)	Resets.
Continuous timer	Retains the value held before block inactivation.
Counter (C)	Retains the value held before block inactivation.
Destination of OUT instruction	Goes into an OFF state.
All other devices	Retains the states held before block inactivation.

Use a SET instruction for a device in a block whose output value is to be retained when the block is inactivated.

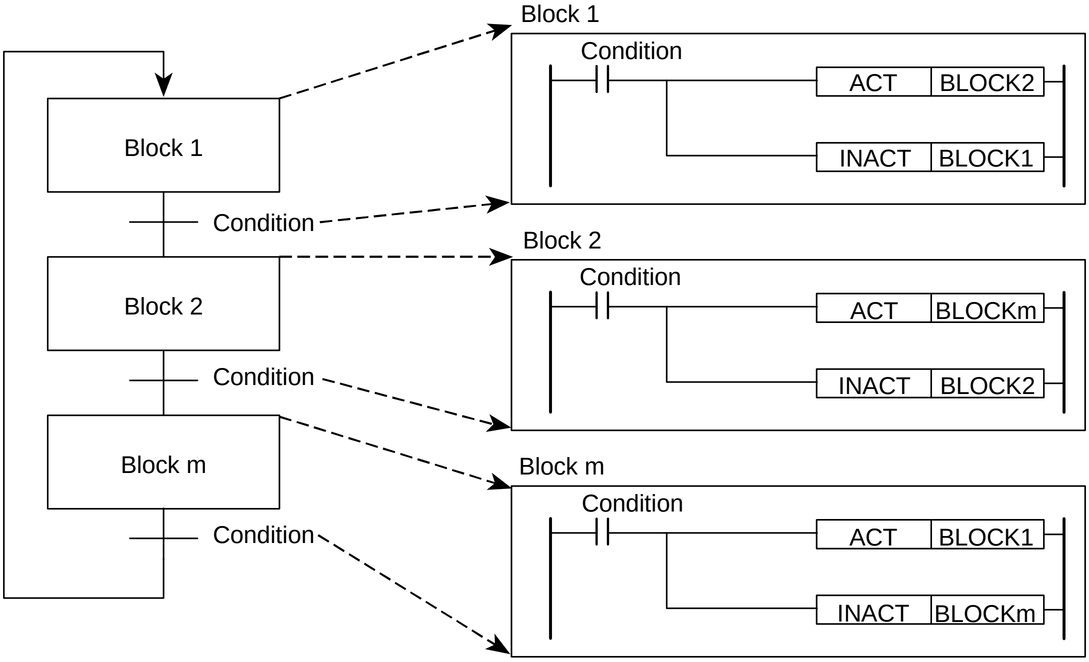

This device is set to OFF.
X00503
I0002
Y00603
X00501
X00502
I0004
I0003
X00301
I0005
SET
Y00601
I0004
Use a SET instruction to retain the output value of this device.

F060406.VSD

Figure A6.4.6 Example of Devices Initialized When a Block is Inactivated

A6.4.5 Operation When Specified Blocks Are Executed

● Example Where Each Block Controls the Next Block to Be Activated



F060407.VSD

Figure A6.4.7 Example Where Each Block Controls the Next Block to Be Activated

● Example Where Block Activation is Controlled by a Scheduler

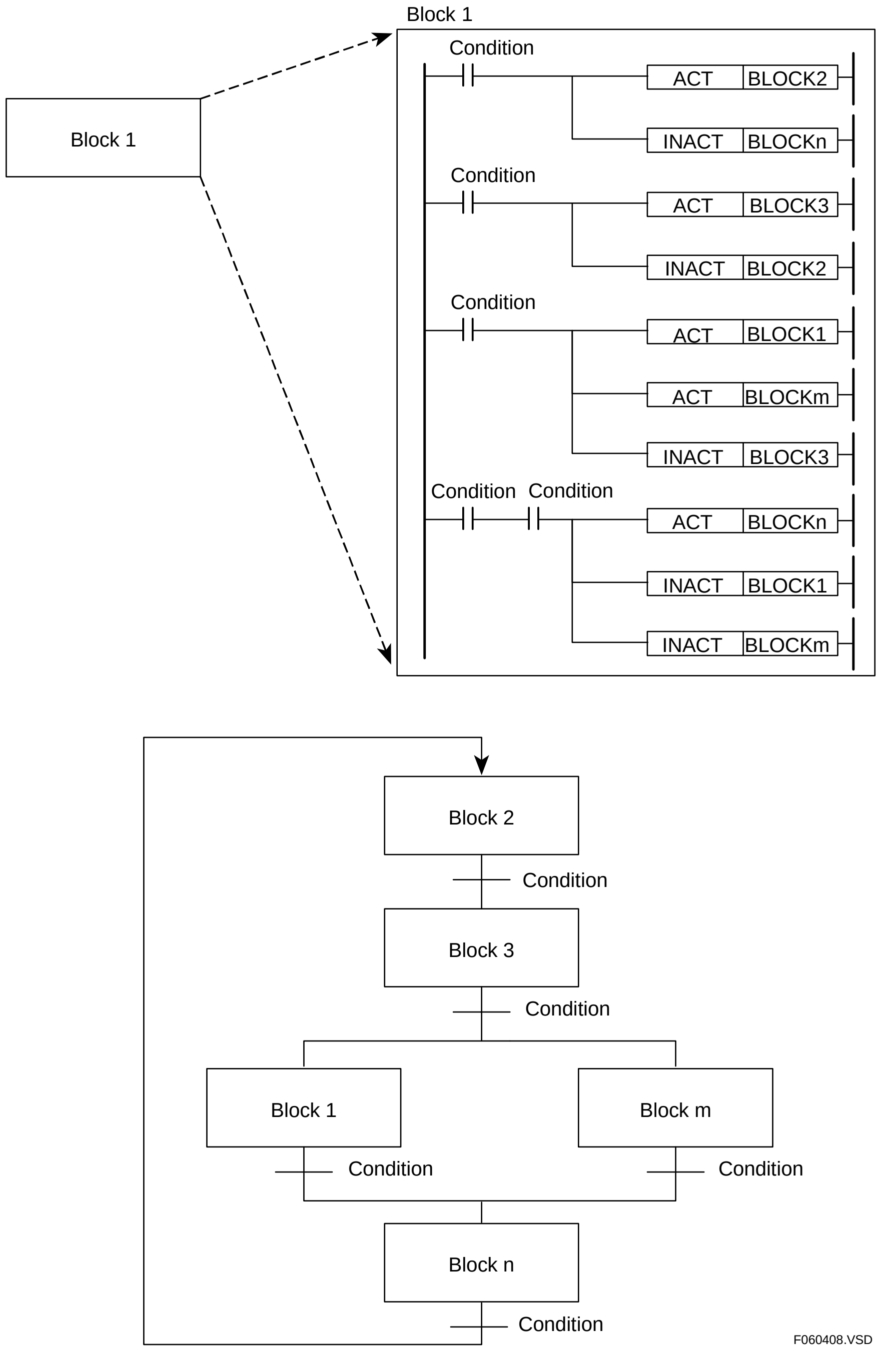


Figure A6.4.8 Example Where Block Activation is Controlled by a Scheduler

Create a scheduler using block 1 which is active by default.

A6.5 Debugging Functions

This section describes the following functions: forced set/reset function for forcibly changing the status of a relay, functions for changing preset values, current values and data values of registers, as well as the stop refreshing function for stopping I/O refreshing, link refreshing and shared refreshing.

A6.5.1 Forced SET/RESET

Debug Stop

A forced SET/RESET forcibly sets a specified bit device to ON/OFF, regardless of program execution. You can apply forced set or forced reset to a maximum of 32 bit devices at one time. Only bit devices are supported (i.e., X, Y, I, E, L, T and C devices).

If a forced SET is applied to a timer (T) or a counter (C), the timer expires or the counter terminates.

A forced SET or forced RESET remains valid until you:

- Cancel the forced set or forced reset;
- Change the operating mode to RUN mode; or
- Turn off the power.

A6.5.2 Changing Preset Values, Current Values and Data Values

Debug Stop

- Changing Preset Values

  You can change the preset values of timers (T) and counters (C).
- Changing Current Values

  You can change the current values of timers (T) and counters (C).

  If you set a current value of “0”, a timer expires and a counter terminates.
- Changing Word or Long-word Data Values

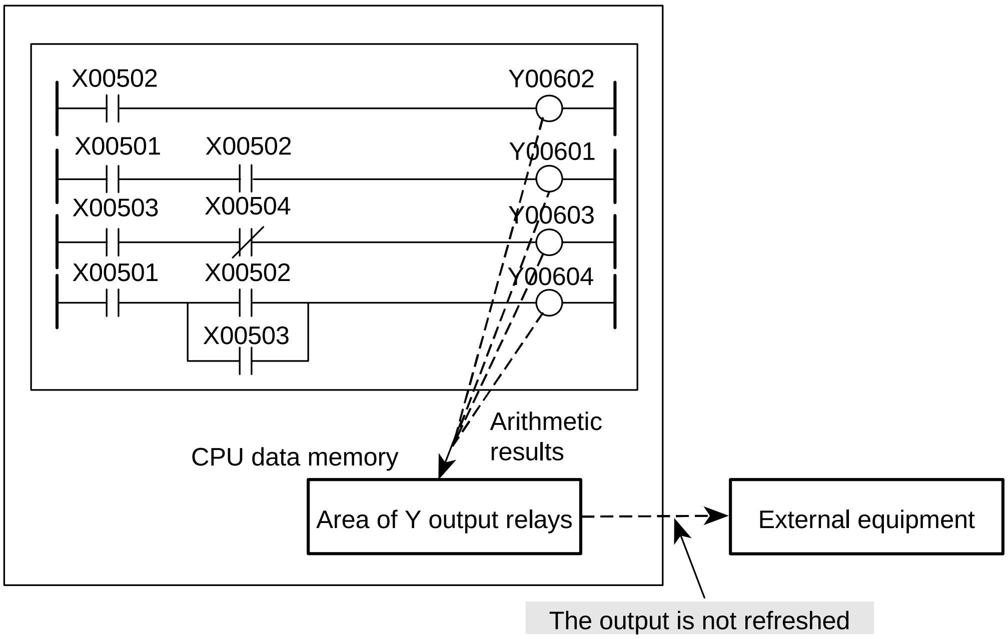
  You can change the data values of word devices other than timers (T) and counters (C), such as data registers (D). If you specify a bit device such as an internal relay (I) instead of a word device, 16 or 32 bits of device data are changed, beginning with the first device address.

A6.5.3 Stopping Refreshing

Stop

You can prevent input relays (X) and output relays (Y) for external equipment, link relays (L) and link registers (W) for FA link and FL-net systems, as well as shared relays (E) and shared registers (R) for add-on CPU modules, from being refreshed by the results of program execution. This allows you to visually check I/O data on the monitor.

In the case of relays (input relays (X) and output relays (Y)) for external equipment, you can stop refreshing X input relays and Y output relays separately.



F060504.VSD

Figure A6.5.1 Stopping Output Refreshing

CAUTION

Refreshing of input relays (X) and output relays (Y) for external equipment, specified in the sensor control block, cannot be stopped.

A6.6 Program Protection

You can protect your programs against unauthorized access for security reasons. There are two modes of protection: executable program protection and block protection. Protection is enabled by defining a password using WideField2. A password can consist of up to eight alphanumeric characters. The protection information is saved with an executable program or block by WideField2.

CAUTION

Program protection is only designed to prevent unauthorized read access. It does not protect against program deletion or CPU operation modification due to erroneous operations or writing.

A6.6.1 Executable Program Protection

Executable program protection protects an entire executable program.

When this protection is enabled, all functions that act upon an executable program (downloading, uploading, monitoring, online editing, etc.) are prohibited.

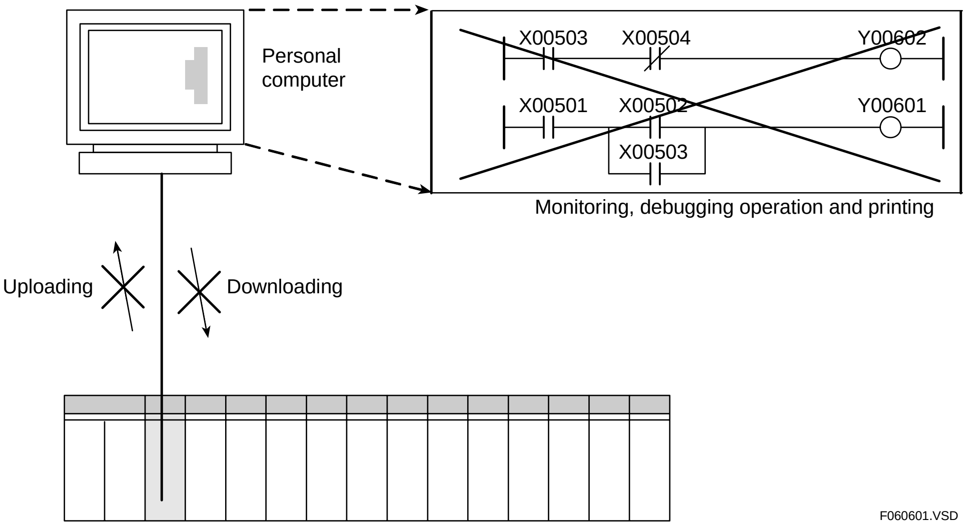


Figure A6.6.1 Executable Program Protection

When executable program protection is enabled, the following functions are prohibited:

- downloading
- uploading
- monitoring (circuit diagram monitoring, debug operation, changing timer (T)/ counter (C) preset values, online edit)
- printing

A6.6.2 Block Protection

Block protection protects programs on per block basis.

This protection mode is only designed to prevent unauthorized read access. In addition, only the specified blocks are protected. When block protection is enabled for a block, its circuit diagrams and instructions are not displayed in WideField2.

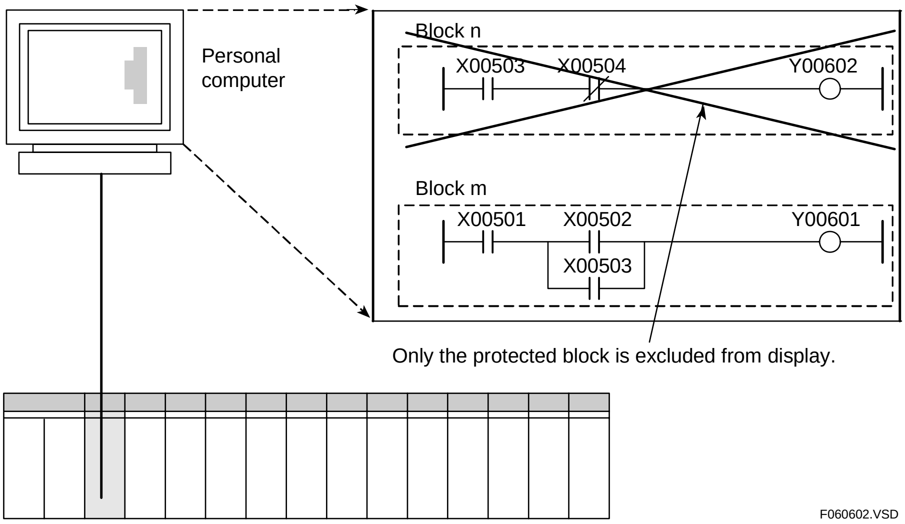


Figure A6.6.2 Block Protection

When block protection is enabled, the following functions are prohibited:

- monitoring (circuit diagram monitoring, debug operation, changing timer (T) /counter (C) preset values, and online edit)
- printing

A6.7 Online Editing

Debug Stop

Online editing allows you to make modifications or additions to your program during program execution. This function is useful for making minor changes to the program during debugging or tuning. Modifications/changes made to the program are reflected in the program memory of the sequence CPU module at the end of a given scan.

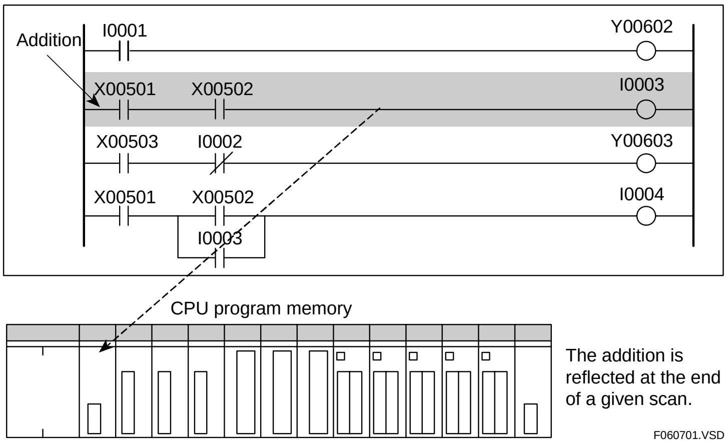


Figure A6.7.1 Online Editing

WARNING

Do not perform online editing when machinery under control is in operation.

When online edited data is written to the sequence CPU module, scan time may become much longer than usual. Scan time lengthens by as much as 10 ms for every 10K step increase in the program size. During this time, external refreshing or communications with external equipment are not allowed.

Edited changes are reflected to the CPU module at the end of conversion, line deletion or online edit operations. Special considerations of sequence processing apply during this update process before all changes are reflected.

If there is a differential type instruction in a circuit that is modified or added online, or in the circuit following a circuit that is modified, inserted or deleted online, beware that the instruction will be executed as if its preceding value is OFF. This means that the instruction may cause a differential output even if its input condition is always ON.

CAUTION

(1) You are not allowed to modify the following instructions and circuits.
- Subroutine Entry (SUB) instruction and Subroutine Return (RET) instruction as well as circuits that contain any of these instructions.
- Interrupt (INTP) instruction and Interrupt Return (IRET) instruction for input modules, as well as circuits that contain any of these instructions.
- Structure Macro Instruction Call (SCALL) instruction, Structure Move (STMOV) instruction, as well as circuits that contain any of these instructions.
- Circuits that contain executing continuous-type application instructions (the continuous-type application instruction is forced to terminate.)

(2) Online editing affects peripheral processing.
Peripheral processing time may lengthen by approximately 200 ms, though this depends on the program size or the location in the program where modifications are made. During this time, the CPU does not perform shared refreshing, link refreshing or command processing.

A6.8 Making Programs and Data Resident in "ROM"

Programs that have been downloaded to the module are stored in the internal flash ROM. The module provides functions equivalent to making programs and data resident in ROM, which can be used to store a copy of programs and data to the SD memory card, separately from the project and setup data stored in the internal flash ROM for backup, transfer or other reasons.

This module does not support the ROM pack function available with older CPU types (F3SP21-0N, F3SP28-3S, F3SP58-6S, etc.). The functions described in this section can be used to replace the ROM pack function for users migrating from an older CPU type.

A6.8.1 Making Programs Resident in "ROM"

You can use the SD memory card to make programs resident. The procedure differs depending on whether your intended purpose is to load a project from the SD memory card at startup or simply to backup data.

■ Loading a Project from SD Memory Card at Startup

To load a project from the SD memory card upon startup, use the Card Boot rotary switch function. This function reads a project in card load format from a specific directory of the SD memory card upon power on or CPU reset. You can select either the Run mode or Stop mode as the operating mode immediately after startup.

SEE ALSO

- For details on Card Boot CARD1, see Subsection B1.3.3, "Card Boot (Run Mode)" or Subsection B1.3.4, "Card Boot (Stop Mode)."
- For details on project in card load format, see Subsection C3.2.3, "FA-M3 File Types."
- For details on how to save a file to an SD memory card, see Subsection C1.4.1, "Accessing a Memory Card."

■ Data Backup

For the simple purpose of data backup, you can save a project to any directory on the SD memory card.

SEE ALSO

For details on how to save a file to an SD memory card, see Subsection C1.4.1, "Accessing a Memory Card."

A6.8.2 Making Data Resident in "ROM"

The module handles data files in both CSV format and binary format. To make data resident, simply store the data files to the SD memory card. You can then read these data files to device using certain instructions.

How to initialize device data at startup is described below.

■ Reading Data from a CSV Formatted File

● Reading data using a ladder program

The Convert CSV File to Device (F2DCSV) instruction can be used to read data from a CSV file to device. Reading can be initiated in the first scan by using the special relay "1 Scan ON at Program Start" (M035) as the input condition of the instruction. The CSV formatted file to be read may be stored in any folder, the pathname of which can be specified in the instruction.

SEE ALSO

- For details on the Convert CSV File to Device (F2DCSV) instruction, see Subsection C3.5.3.10, "Convert CSV File to Device (F2DCSV)"
- For details on how to store files to an SD memory card, see Subsection C1.4.1, "Accessing a Memory Card."

● Reading data using Card Batch File function

The Convert CSV File to Device (F2DCSV) command of the Card Batch File function can be used to read data from a CSV file to device. Reading can be initiated before the first scan by selecting the "Startup event" trigger or the "Run program event" trigger. The CSV formatted file to be read may be stored in any folder, the pathname of which can be specified in the command.

SEE ALSO

- For details on the Convert CSV File to Device (F2DCSV) command of the Card Batch File function, see Subsection B2.8.1.1, "Convert CSV File to Device (F2DCSV)" of Chapter B2, "Card Batch File Function."
- For details on how to store files to an SD memory card, see Subsection C1.4.1, "Accessing a Memory Card."

■ Reading Data from a Binary File

● Reading data using a ladder program

The Convert Binary File to Device (F2DBIN) instruction can be used to read data from a binary file to device. Reading can be initiated in the first scan by using the special relay "1 Scan ON at Program Start" (M035) as the input condition of the instruction. The binary file to be read may be stored in any folder, the pathname of which can be specified in the instruction.

SEE ALSO

- For details on the Convert Binary File to Device (F2DBIN) instruction, see Subsection C3.5.3.12, "Convert Binary File to Device (F2DBIN)".
- For details on how to store files to an SD memory card, see Subsection C1.4.1, "Accessing a Memory Card."

● Reading data using the Card Batch File function

The Convert Binary File to Device (F2DBIN) command of the Card Batch File function can be used to read data from a binary file to device. Reading can be initiated before the first scan by selecting the "Startup event" trigger or the "Run program event" trigger. The binary file to be read may be stored in any folder, the pathname of which can be specified in the command.

SEE ALSO

- For details on the Convert Binary File to Device (F2DBIN) command of the Card Batch File function, see Subsection B2.8.1.3, "Convert Binary File to Device (F2DBIN)" of Chapter B2, "Card Batch File Function."
- For details on how to store files to an SD memory card, see Subsection C1.4.1, "Accessing a Memory Card."

A6.9 Exclusive Access Control

This section describes exclusive access control, a function for restricting operations on program, operating mode, or device data by other users during operation or debugging.
Exclusive access control is used to prevent a program, operating mode, or device data from being changed or a program or device data from being downloaded by other users, say during operation or debugging.
Once you acquire an exclusive access right, all modification- and control-related commands issued from other tools, sequence CPU modules or personal computer links are rejected until you release the right.
While you hold the exclusive access right, all modification- and control-related commands from other users remain disabled so you should release the access right as soon as you have completed the required processing.
If another user has already acquired an exclusive access right, it is not available to you.
The following exclusive access control functions are provided:

● Get access right

This function acquires exclusive access right.

● Release

This function releases exclusive access right.

● Forced Release

This function allows a tool or module that has no exclusive access right to force the holder of the exclusive access right to release it.

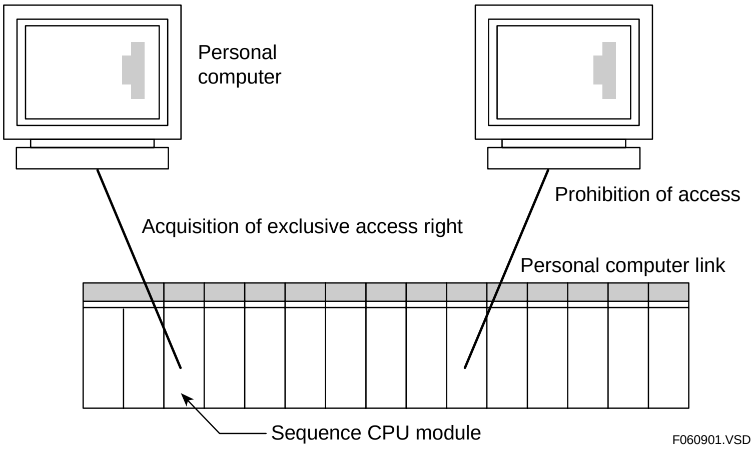


Figure A6.9.1 Exclusive Access Right

Once a user acquires exclusive access right, the system prohibits other tools or modules having no exclusive access right to perform the following operations:

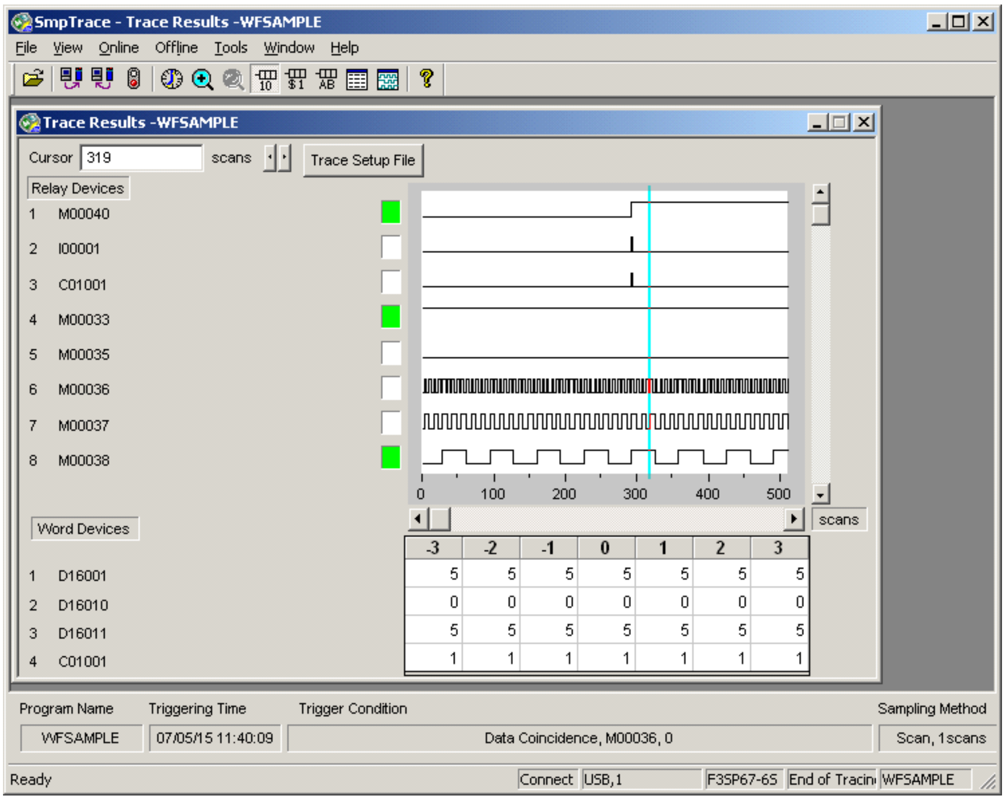
- Runnig a program
- Stopping a program
- Debugging
- Download
- Debug operation and use of debugging functions
- Writing to devices
- Changing preset values of timers (T)/counters (C)

A6.10 Sampling Trace Function

The sampling trace function records the state transitions of specified devices. It stores the states and contents of devices selected to be sampled, sequentially in the sampling trace memory of the sequence CPU module.
Three sampling methods are available:

- **TRC instruction sampling**
- **Scan sampling**
- **Periodic sampling**

You can define the trigger condition for sampling as the rising edge of a specified relay signal, the falling edge of a specified relay signal or data coincidence with a selected register device. The CPU monitors the trigger condition during scan end processing. If the trigger condition becomes true, the CPU takes 1024 samples, starting from a specified negative delay before or a specified positive delay after the condition becomes true.
Using WideField2, you can configure the sampling trace function, and subsequently view sampling trace results in time-chart format, as shown in the figure below.



Cursor 319 scans

1	M00040	🟩
2	I00001	□
3	C01001	□
4	M00033	🟩
5	M00035	□
6	M00036	□
7	M00037	□
8	M00038	🟩

		-3	-2	-1	0	1	2	3
1	D16001	5	5	5	5	5	5	5
2	D16010	0	0	0	0	0	0	0
3	D16011	5	5	5	5	5	5	5
4	C01001	1	1	1	1	1	1	1

Program Name	Triggering Time	Trigger Condition	Sampling Method
WFSAMPLE	07/05/15 11:40:09	Data Coincidence, M00036, 0	Scan, 1scans

Ready | Connect | USB,1 | F3SP67-6S | End of Tracin | WFSAMPLE

Start of sampling trace : Directive from the programming tool
Trigger condition : Defined using the programming tool
- Rising edge of specified relay
- Falling edge of specified relay
- Data coincidence

Sampling method : TRC instruction sampling
Scan sampling (every 1 to 1000 scans)
Periodic sampling (10 to 2000 ms)

Number of samples : 1024 cycles
Delay : Specify a positive or negative delay between -1023 and +1023 scans.
Sampled Devices : 16 X, Y, I, E, L, T, C or M relay devices; or
D, B, R, W, V, Z, T or C word devices; or
4 or 16 X, Y, I, E, L, T, C or M relay devices, starting from the specified first device address.

F061007.VSD

Figure A6.10.1 View of Sampling Trace Results

You can execute sampling trace in either Run or Debug mode. Re-executing sampling trace erases previous data. If you perform sampling trace setup using the configuration function, the CPU begins sampling immediately after power-on. If you perform sampling trace setup using the configuration function and then permanently store the setup to ROM, the CPU reverts to the ROM setup after a power-on-and-off sequence even if you have modified the settings using the programming tool.

SEE ALSO

For details on how to configure the sampling trace function, see, "FA-M3 Programming Tool WideField2 User's Manual" (IM34M6Q15-01E).

How sampling is carried out is explained below.

● TRC Instruction Sampling

In TRC instruction sampling, the CPU samples the states and data of specified contacts whenever the Sampling Trace (TRC) instruction is executed. By executing the TRC instruction in a program, you can perform sampling at any point within a scan.

The CPU collects data when the input-condition relay of the Sampling Trace (TRC) instruction is set to ON. The CPU stores results of up to four cycles of sampling if the TRC instruction is executed multiple times within the same scan. Any fifth or subsequent Sampling Trace (TRC) instruction executions within a scan are ignored. Sampling trace results are stored at the end of a scan.

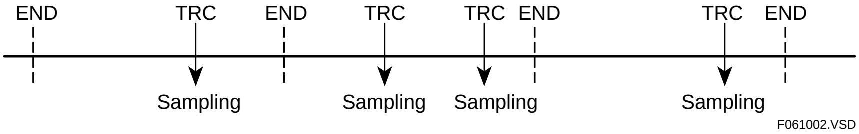


Figure A6.10.2 Sampling when the Sampling Trace (TRC) Instruction is Executed

● Scan Sampling

In scan sampling, the CPU samples the states and data of specified contacts at the end of a scan. It collects and stores the data each time the specified number of scans are completed.

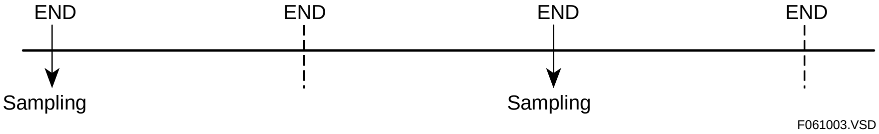


Figure A6.10.3 Scan Sampling at Two-scan Intervals

● Periodic Sampling

In periodic sampling, the CPU samples the states and data of specified contacts at fixed time intervals. It collects and stores the data after the specified period expires and before the next scan begins.

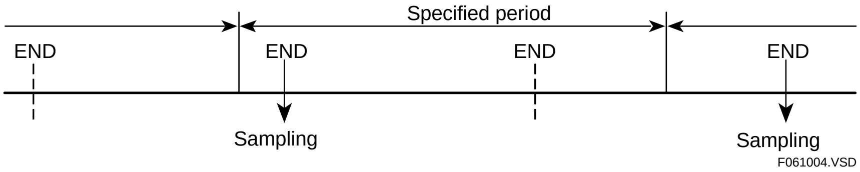


Figure A6.10.4 Periodic Sampling

CAUTION

The sampling trace function checks the trigger condition when an END instruction in a program is processed. Therefore, if the trigger condition becomes true during program execution but becomes false again before processing of the END instruction begins, sampling is not performed.

● Sampling when a Negative Delay Is Defined

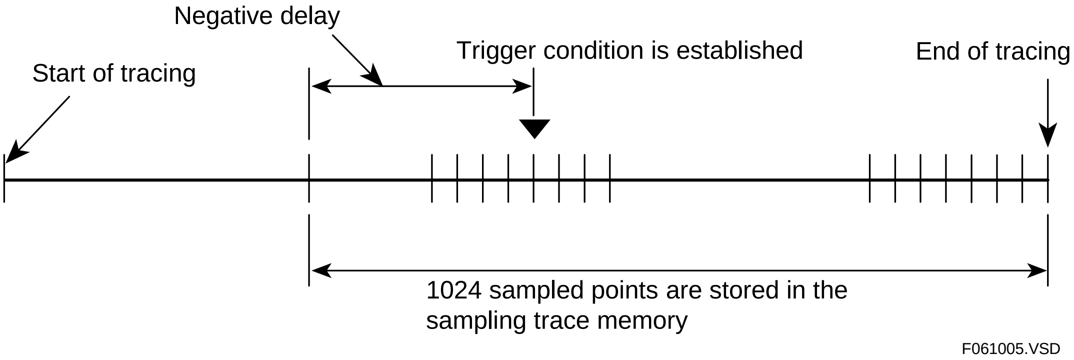

Negative delay
Start of tracing
Trigger condition is established
End of tracing
1024 sampled points are stored in the sampling trace memory
F061005.VSD

Figure A6.10.5 Sampling when a Negative Delay Is Defined

● Sampling when a Positive Delay Is Defined

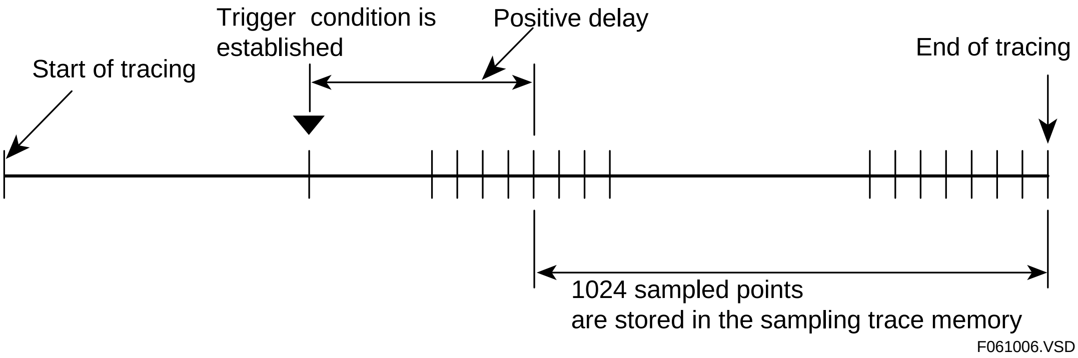

Trigger condition is established
Positive delay
Start of tracing
End of tracing
1024 sampled points are stored in the sampling trace memory
F061006.VSD

Figure A6.10.6 Sampling when a Positive Delay Is Defined

A6.11 Personal Computer Link Function

This section describes the personal computer link function that allows a personal computer or a display device to be connected to a sequence CPU module.

The SIO port on the front of the CPU module functions in the same way as the RS232-C communication port on the F3LC1□-1F personal computer link module. This means you can connect higher-level equipment, such as a personal computer or FA computer, or a monitor to the CPU module to perform one-to-one communication as you do with the personal computer link module. This feature is called the personal computer link function.
You can monitor and configure devices, as well as start, stop, download and upload programs by entering commands from the higher-level computer.

SEE ALSO

For details on the higher-level link service via Ethernet, which is equivalent to the personal computer link function, see Chapter 4, "Higher-level Link Service (Personal Computer Link Function)" of "Sequence CPU – Network Functions User's Manual (for F3SP66-4S, F3SP67-6S)" (IM34M6P14-02E).

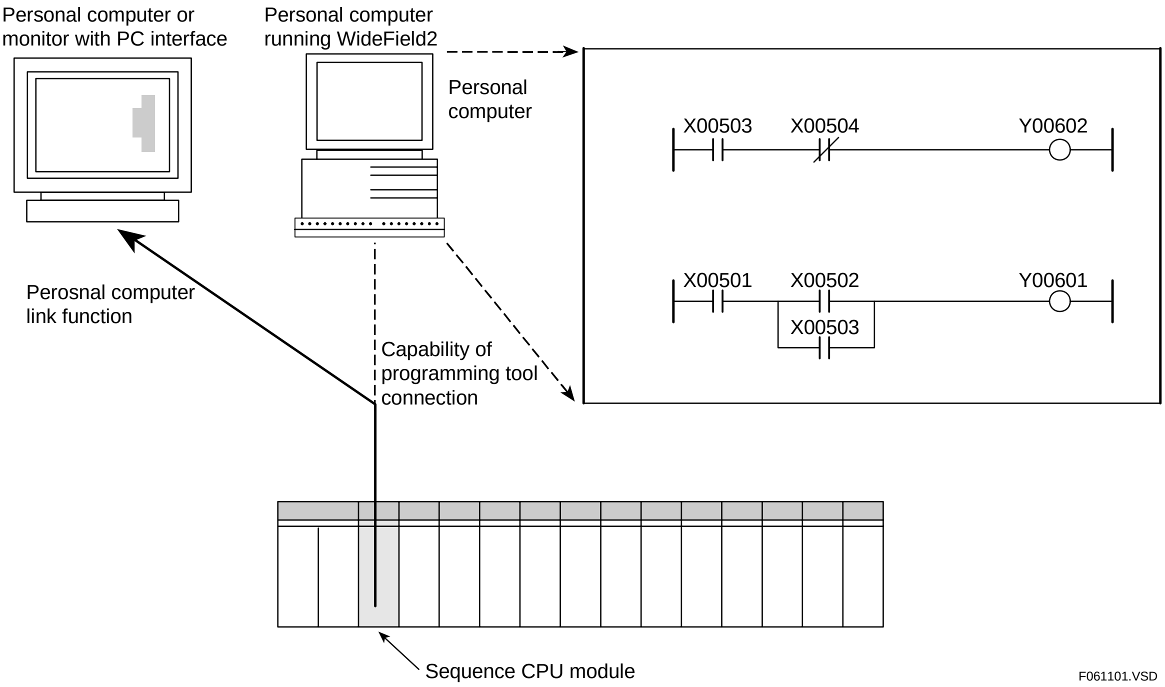


Figure A6.11.1 Personal Computer Link Function

A6.11.1 System Configuration

The figure below shows examples of system configuration using the personal computer link function.

External equipment, such as a personal computer or monitor, is connected to the sequence CPU module of the FA-M3 by using the SIO port on the front of the FA-M3 and a dedicated monitor cable.

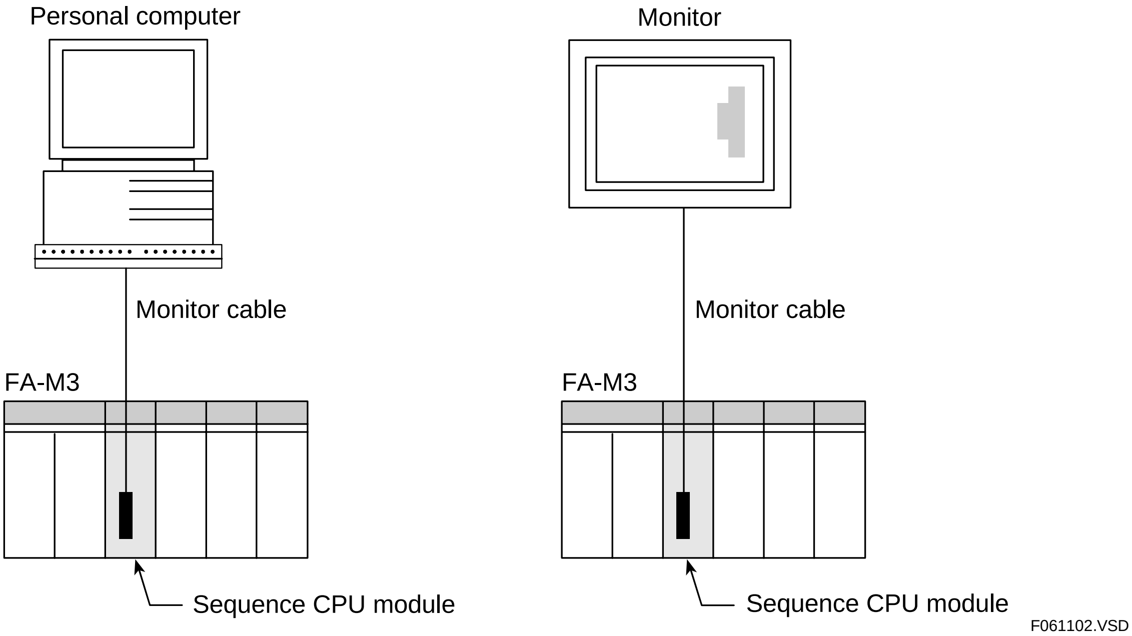


Figure A6.11.2 Examples of Connection between a Sequence CPU Module and External Equipment

Provide the monitor cable with a ferrite core if you want to have the connected device compatible with the CE marking.

Table A6.11.1 Examples of Ferrite Cores

Manufacturer	Product Series Name
Kitagawa Industries K.K.	RFC series
TDK Corporation	ZCAT series
NEC TOKIN Corporation	ESD-SR series

A6.11.2 Differences from Personal Computer Link Module

This subsection describes the differences between the F3LC1□-1F personal computer link module and the personal computer link function of the sequence CPU module.

■ Function

The transmission rate and data format of the CPU's personal computer link function differ from those of the personal computer link module.

SEE ALSO

For details, see Subsection A6.11.4, "Setting up the Personal Computer Link Function."

Table A6.11.2 Transmission Rate and Data Format of CPU's Personal Computer Link Function

Transmission Rate (bps)	Data Length	Parity	Stop Bits
9600	8 bits	Even	1 bit
9600	8 bits	None	1 bit
19200	8 bits	Even	1 bit
19200	8 bits	None	1 bit
38400	8 bits	Even	1 bit
38400	8 bits	None	1 bit
57600	8 bits	Even	1 bit
57600	8 bits	None	1 bit
115200	8 bits	Even	1 bit
115200	8 bits	None	1 bit

A dedicated monitor cable is required to connect a personal computer or monitor to the CPU module. To set the transmission rate, data format, checksum, end character, and protection function, use the configuration item "Communication mode" (setup is by switches in the case of the personal computer link module). The event transmission function is not supported.

If the sequence CPU module receives an MDR module reset command as a PC line command, it resets only the communication port. The maximum number of personal computer link modules that can be installed remains the same even if the CPU's personal computer link function is used.

■ Communications Protocol

A brief description of the communications protocol of the personal computer link function is given below.

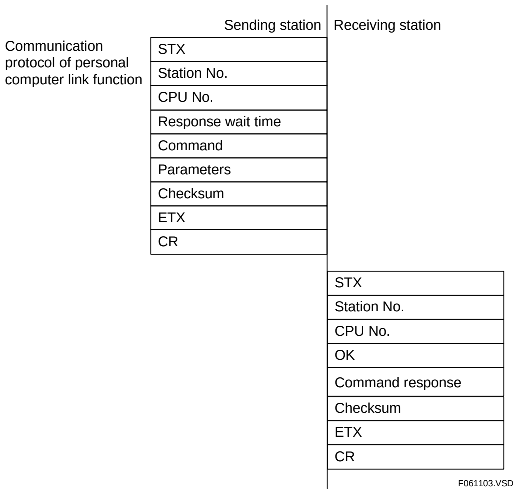


Figure A6.11.3 Communications Protocol of Personal Computer Link Function

In personal computer link communication, the maximum size of text that can be transferred each time is 512 bytes.

A6.11.3 Specifications of Personal Computer Link Function

Table A6.11.3 Specifications of Personal Computer Link Function

Item	Description	Setup [*1]
Interface	EIA RS-232-C compliant	
Transmission mode	Half-duplex transmission	
Synchronization	Start-stop synchronization	
Transmission rate (bps)	9600/19200/38400/57600/115200	✓
Data format	Start bit : 1	
	Data length : Fixed at 8 bits	
	Parity bit : None or Even	✓
	Stop bit : 1 bit (fixed)	
Error checking	Parity check	
	Checksum : Yes/No	✓
Control line (RS-232-C)	Not used.	
Xon/Xoff	Not used.	
Configurable item	Transmission rate, data format, checksum, end character and protection	✓
Protocol	Proprietary protocol	
End character	Yes/No	✓
Protection function [*2]	Yes/No	✓
Access range	Access to all control data, upload/download programs, CPU operation (Run mode)/stop (Stop mode), and read error logs	
Transmission distance	12 m max.	
External connection	Dedicated monitor cable	

*1 The check mark ✓ indicates that a user can configure the item by using the configuration function. However, there are restrictions on the way the transmission rate and parity check are combined. See subsection A6.11.4, “Setting up the Personal Computer Link Function,” for more information.

*2 You can set the protection function to the Yes option to prevent inadvertent writing to the FA-M3.

CAUTION

The personal computer link function uses neither a control line nor Xon/Xoff characters. Be careful when using the function because a communication failure may occur at the higher-level equipment side, depending on the transmission rate.

A6.11.4 Setting Up the Personal Computer Link Function

This subsection describes the items you should define when using the personal computer link function.

■ Transmission Rate and Data Format

You can set up the transmission rate and data format by configuration.

The table below shows the available combinations for transmission rate and data format.

Table A6.11.4 Combinations of Transmission Rate and Data Format

Mode	Transmission Rate and Data Format			
	Transmission Rate (bps)	Data Length	Parity	Stop Bits
Communication mode 0	9600	8 bits	Even	1 bit
Communication mode 1	9600	8 bits	None	1 bit
Communication mode 2	19200	8 bits	Even	1 bit
Communication mode 3	19200	8 bits	None	1 bit
Communication mode 4	38400	8 bits	Even	1 bit
Communication mode 5	38400	8 bits	None	1 bit
Communication mode 6	57600	8 bits	Even	1 bit
Communication mode 7	57600	8 bits	None	1 bit
Communication mode 8	115200	8 bits	Even	1 bit
Communication mode 9	115200	8 bits	None	1 bit

The personal computer link function is set to “communication mode 0” when the sequence CPU module is shipped from the factory, the CPU memory is cleared, or the function is not configured.

■ Checksum, End Character and Protection

Set up these items using the configuration item “communications setup.” By default, all these items are disabled.

SEE ALSO

For details on the configuration function, see, "FA-M3 Programming Tool WideField2 User's Manual" (IM34M6Q15-01E).

A6.11.5 Communication Procedure

To be able to perform communication, the transmission specifications, including the transmission rate and data format, must be consistent between the CPU module and a personal computer, FA computer or monitor.

Use the configuration function to set up the transmission specifications of the sequence CPU module. To set the transmission specifications of a personal computer or FA computer, use a communication software program. To set the transmission specifications of a monitor, follow its configuration procedure.

■ Communication Procedure

How to communicate with the FA-M3 using a BASIC program on a personal computer is briefly described below.

1. Open the RS-232-C communication file by entering a command using the following syntax:

```
OPEN “COM :   ○○○○○” AS#△
```

○○○○○ :Enter communication parameters, such as the parity, data length, and the number of stop bits.

△ :File number. This number is used for subsequent input to and output from the file.

2. Send a command to the FA-M3 using the following syntax.

```
PRINT#△, String variable name (or string)
```

3. To receive a response from the FA-M3, enter a command using the following syntax:

```
LINE INPUT#△, String variable name
INPUT#△, String variable name
```

SEE ALSO

For details on the statements and arguments, refer to the BASIC reference manual for the PC to be used.

■ Overview of Communication

Communication control performed by the CPU module is based on the processing of commands and responses using a dedicated protocol.

At first, the host computer (or monitor) has the transmission right. When the computer sends a command, the transmission right transfers to the CPU module. The CPU module then sends a response to the higher-level computer.

If the configuration item "Use personal computer link function" is enabled, the CPU module does not send any command to the higher-level computer.

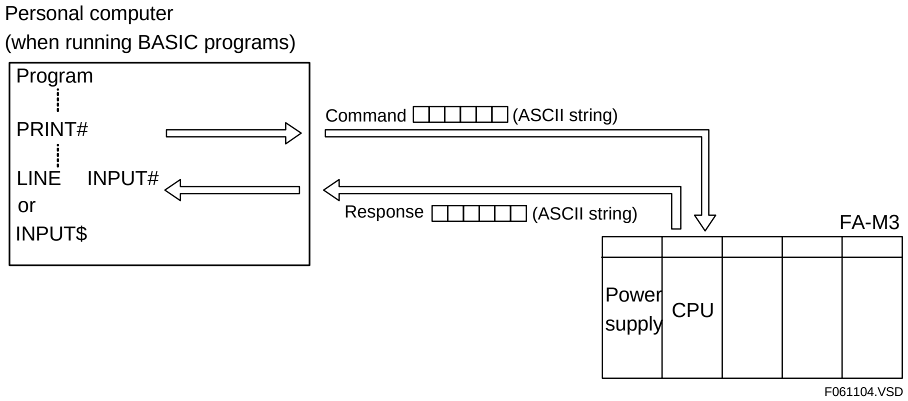


Figure A6.11.4 Interaction between Command and Response

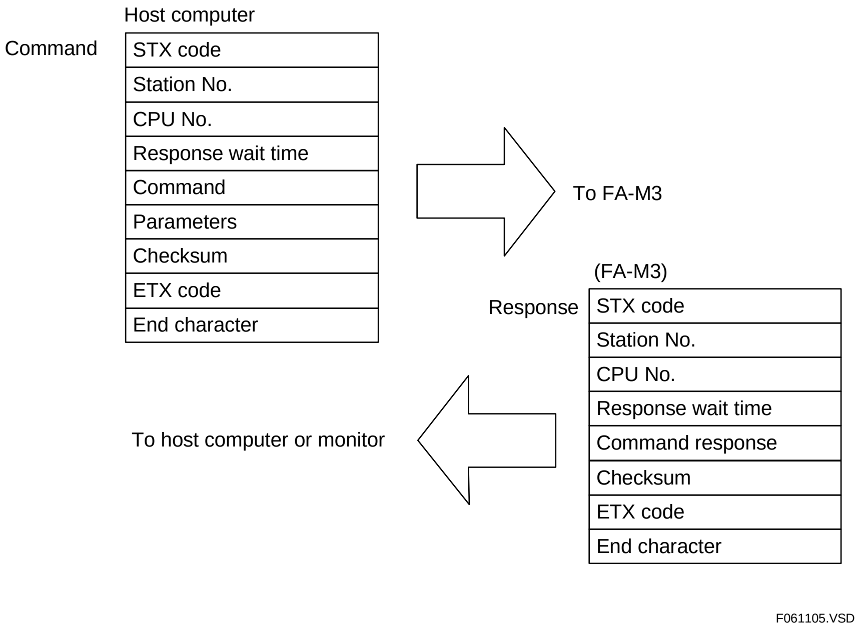


Figure A6.11.5 Brief Description of Command and Response Formats

A6.11.6 Commands and Responses

SEE ALSO

For details on commands and responses, see "Personal Computer Link Commands User's Manual" (IM34M6P41-01E).

■ Command Format and its Elements

The format of a command transmitted from a higher-level computer (or monitor) to the FA-M3 is shown below.

No. of Bytes	Element
1	STX code
2	Station No.
2	CPU No.
1	Response wait time
3	Command
Variable-length	Parameters
2	Checksum
1	ETX code
1	CR code

Checksum: Required only if the configuration item "Checksum" is set to "Yes"

CR code: Required only if the configuration item "End character" is set to "Yes"

F061106.VSD

Figure A6.11.6 Command Format and its Elements

Only uppercase alphabetic characters from A to Z (ASCII codes $41 to $5A in hexadecimal) are used in commands and responses.

The individual elements are described below.

● STX (Start of Text) Code

This control code identifies the beginning of text. The corresponding character code is $02.

● Station No.

The station No. is fixed at 01 when the personal computer link function of the sequence CPU module is used.

● CPU No.

Identifies the target sequence CPU module or add-on CPU module for a command using a number from 01 to 04.

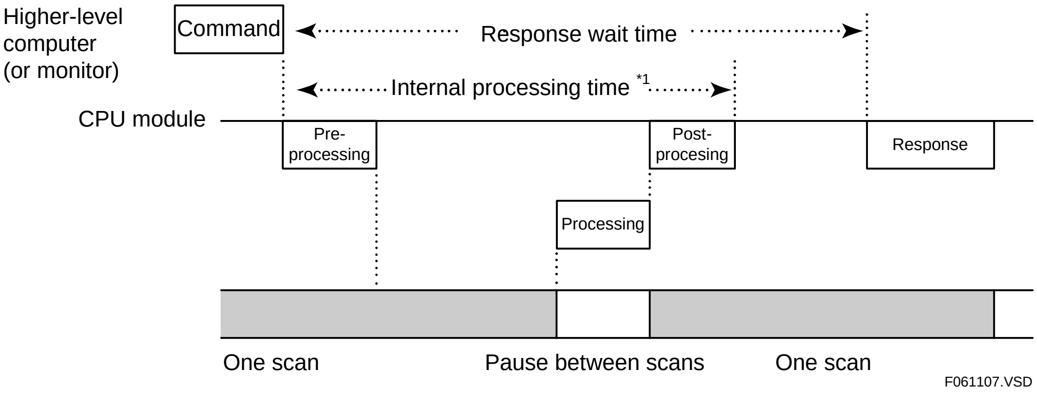
- 01: Sequence CPU module
- 02: Add-on CPU module 1
- 03: Add-on CPU module 2
- 04: Add-on CPU module 3

Response Wait Time

You can specify the maximum waiting time (time delay of up to 600 ms) for a response following a command transmission. Set a longer wait time if the communication software running on the higher-level computer is, say, a BASIC interpreter. Specify this time using one character ('0' to 'F') as shown below.

Table A6.11.5 Response Wait Time

Character	Response Wait Time (ms)	Character	Response Wait Time (ms)
0	0	8	80
1	10	9	90
2	20	A	100
3	30	B	200
4	40	C	300
5	50	D	400
6	60	E	500
7	70	F	600



*1: Even if the response wait time is set at 0, there is a delay of as much as the internal processing time.

Figure A6.11.7 Response Wait Time

Command

Using three letters, specify the type of access, such as reading or writing, from a higher-level computer (or monitor) to the sequence CPU module.

Parameters

These include device name, number of devices, data, etc. The actual parameters vary depending on the command used. Some commands require no parameters.

Checksum

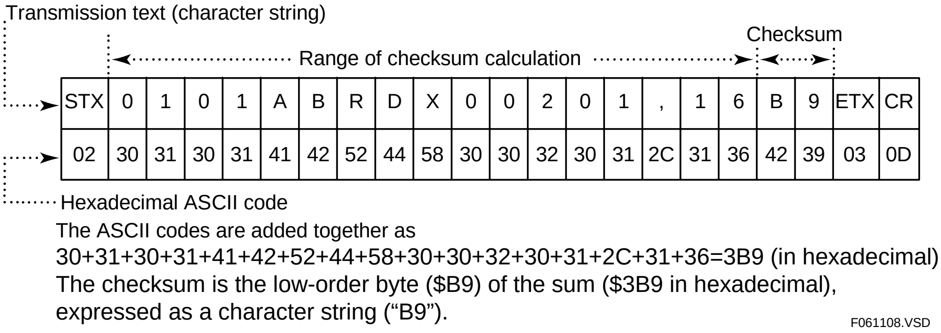
A checksum can be added to the transmission text for data validation. You can select whether to add a checksum in the configuration.

If checksum is set to "Yes", a checksum must be appended to a command before transmission from the higher-level computer (or monitor) to the FA-M3. Moreover, a checksum is automatically appended to the response transmitted from FA-M3.

If checksum is set to "No", this element must not be appended to a command.

How the checksum is calculated is explained below.

- Add the ASCII codes of the characters following the STX character and preceding the checksum.
- Extract the low order byte of the sum and express its hexadecimal value as a character string (2 characters, 2 bytes) to obtain the checksum.


Transmission text (character string)
Checksum
Range of checksum calculation
Hexadecimal ASCII code
F061108.VSD

STX	0	1	0	1	A	B	R	D	X	0	0	2	0	1	,	1	6	B	9	ETX	CR
02	30	31	30	31	41	42	52	44	58	30	30	32	30	31	2C	31	36	42	39	03	0D

The ASCII codes are added together as
30+31+30+31+41+42+52+44+58+30+30+32+30+31+2C+31+36=3B9 (in hexadecimal)
The checksum is the low-order byte ($B9) of the sum ($3B9 in hexadecimal),
expressed as a character string (“B9”).

Figure A6.11.8 Checksum Calculation

ETX (End of Text) Code

This control code identifies the end of text. The corresponding character code is $03.

CR (Carriage Return) Code

This control code identifies the end. The corresponding character code is $0D (ASCII code 13 in decimal)

This control code is required only if the end character is set to "Yes" in the configuration.

■ Response Format and its Elements

The format of a response that is sent from the FA-M3 to a higher-level computer (or monitor) is shown here.

SEE ALSO

For details on individual elements and characters used, see "■ Command Format and its Elements" given earlier in this section.

● If communications is normal

No. of Bytes	Element
1	STX code
2	Station No.
2	CPU No.
2	OK
Variable-length	Command response
2	Checksum
1	ETX code
1	CR code

Appended to the response only if enabled accordingly in the configuration.

F061109.VSD

Figure A6.11.9 Response Format when Communication is Normal

If communications is successful, the character string "OK" and the command response are returned.

● If a communications error occurs

No. of Bytes	Element
1	STX code
2	Station No.
2	CPU No.
2	ER
2	EC 1
2	EC 2
3	Command
2	Checksum
1	ETX code
1	CR code

Appended to the response only enabled accordingly in the configuration.

F061110.VSD

Figure A6.11.10 Response Format when an Error Occurs

If a communications error occurs, the string “ER” is returned along with error codes EC1 and EC2.

EC1 = Error code

EC2 = Detailed error code

If the communication failure is due to an error in the CPU number, the received 2-byte CPU number is returned. If the failure is due to an error in the station number, no response is returned.

If an ETX code in a command is not received, no response may be returned. If this happens, be sure to perform a timeout process on the higher-level computer or monitor.

■ Error Code in a Response

When a communications error or a command error occurs, the module returns an "ER" character string and an error code as a response to the command.

The table below lists the error codes that may be included in a response.

Table A6.11.6 Error Code in a Response

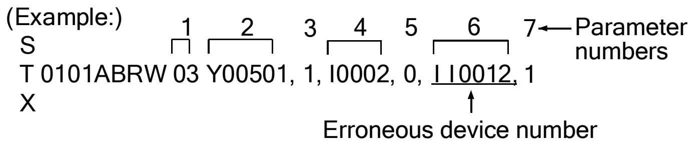
Error Code (EC1)	Semantics	Possible Causes
01	CPU number error	- The CPU number is outside the range of 1 to 4.
02	Command error	- The command does not exist. - The command is not executable.
03	Device specification error	- The device name does not exist.*1 - A relay device is incorrectly specified for read/write access in word units.
04	Value outside the setting range	- Characters other than 0 and 1 are used for bit setting. *1 - Word setting is out of the valid range of 0000 to FFFF. - The specified starting position in a command, such as Load/Save, is out of the valid address range.
05	Data count out of range	- The specified bit count, word count, etc. exceeded the specifications range. *1 - The specified data count and the device parameter count, etc. do not match.
06	Monitor error	- Attempted to execute monitoring without having specified a monitor command (BRS, WRS).
08	Parameter error	- A parameter is invalid for a reason other than those given above. *1
41	Communication error	- An error has occurred during communication. *1
42	Checksum error	- Value of checksum differs. (Bit omitted or changed characters)
43	Internal buffer overflow	- The amount of data received exceeded stipulated value.
51	Timeout error	- No end-of-process response is returned from the CPU for reasons such as CPU power failure. (timeout)
52	CPU processing error	- The CPU has detected an error during processing. *1
F1	Internal error	- A Cancel (PLC) command was issued during execution of a command other than a Load (PLD) or Save (PSV) command. - An internal error was detected.

*1: For details, see the following table.

In the case of a parameter error, the number of the invalid parameter is stored in the detailed error code.

In the case of a communication error, detailed error information is stored in the detailed error code, which is described in the following table.

Table A6.11.7 Detailed Error Codes

Error Code (EC1)	Meaning	Detailed Error Code (EC2) *
03	Device specification error	Error parameter number, expressed in hexadecimal. (The number of the first parameter where an error has occurred, counting from the beginning of the parameters)  (Example:) 1 2 3 4 5 6 7 ← Parameter numbers STX 0101ABRW 03 Y00501, 1, I0002, 0, I I0012, 1 Erroneous device number In this case, Error code EC1=03 Error code EC2=06.
04	Value outside the setting range	
05	Data count out of range	
08	Parameter error	
41	Communication error	b7 ¦ b6 ¦ b5 ¦ b4 ¦ b3 ¦ b2 ¦ b1 ¦ b0 MSB LSB Each bit has the following meaning. b7: Reserved b6: Reserved b5: Framing error b4: Overrun error b3: Parity error b2: Reserved b1: Reserved b0: Reserved
52	CPU processing error	1□: Self-diagnostic error 2□: Program error (including parameter error) 4□: Inter-CPU communication error 8□: Device access error 9□: Communication protocol error A□: Parameter error B□: Operating mode error, protected/exclusive access C□: Device/block specification error F□: Internal system error

* The EC2 error code has no meaning for any value of EC1 other than those listed above.

■ List of Supported Devices

Use a comma (,) or a space ( ) to delimit parameters.

A device name is represented using six or seven characters (or bytes). Abbreviations may be used.

For example, X00201 can be abbreviated as X201 and V00002 as V02 or V2.

Example: To read data from 5 input relays of CPU 1, starting from input relay X00201 with a response wait time of 100 ms.

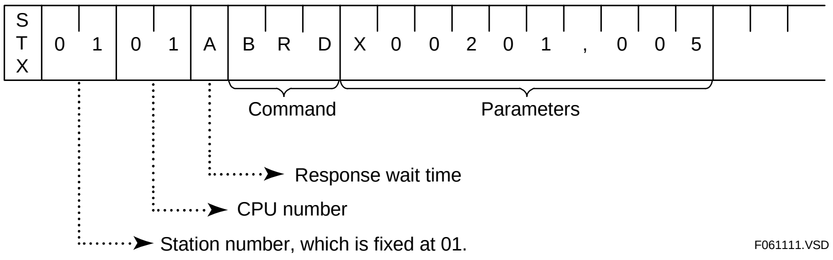

STX
0 1 0 1 A B R D X 0 0 2 0 1 , 0 0 5
Command
Parameters
Response wait time
CPU number
Station number, which is fixed at 01.
F061111.VSD

Figure A6.11.11 Reading Five Input Relays, Starting from X00201

Table A6.11.8 List of Supported Devices

Device name			Read		Write	
		Length	Bit	Word	Bit	Word
Relay Devices	Xnnnnn Input relay	6 bytes	✓	✓	–	–
	Ynnnnn Output relay	6 bytes	✓	✓	✓	✓
	Innnnn Internal relay	6 bytes	✓	✓	✓	✓
	Ennnnn (Extended) shared relay	6 bytes	✓	✓	✓	✓
	Lnnnnn Link relay	6 bytes	✓	✓	✓	✓
	Mnnnnn Special relay	6 bytes	✓	✓	✓*6	✓*6
	Txnnnn Timer	6 bytes	✓*1	✓*2	–	✓*2
	Cxnnnn Counter	6 bytes	✓*1	✓*2	–	✓*2
Word devices	Dnnnnn Data register	6 bytes	–	✓	–	✓
	Rnnnnn (Extended) shared register	6 bytes	–	✓	–	✓
	Vnnnnn Index register	6 bytes	–	✓	–	✓
	Bnnnnn*3 File register	7 bytes	–	✓	–	✓
	Wnnnnn Link register	6 bytes	–	✓	–	✓
	Znnnnn Special register	6 bytes	–	✓	–	✓*6

*1: Specify:
- a time-out relay as TUnnnn
- an end-of-count relay as CUnnnn

*2: Specify:
- the current value of a countdown timer as TPnnnn
- the current value of a countdown counter as CPnnnn
- the current value of a count-up timer*4 as TInnnn
- the current value of a count-up counter *4 as CInnnn
- the preset value of a timer*5 as TSnnnn
- the preset value of a counter*5 as CSnnnn

*3: Only available with the F3SP25, F3SP28, F3SP35, F3SP38, F3SP53, F3SP58, F3SP59, F3SP66 and F3SP67 sequence CPU modules.

*4: In the FA-M3, countdown timers and counters are provided for display on higher-level personal computers. Current value of count-up timer/counter = preset value – current value of countdown timer/counter.

*5: You may not use these preset values in word write commands.

*6: You may not use the BWR, BFL, WWR and WFL commands to write to the module. Use the BRW and WRW commands instead.

■ Precautions for Communications

- You should include timeout handling on the higher-level computer to handle situations where a response is not returned due to say, an incorrect station number specified in the command.
- If the personal computer link function is used to download a program, then you should not load another program from another source (personal computer link module, Ethernet interface module, etc.) at the same time. Otherwise, normal operation is not guaranteed.
- When writing to a shared device, the value may be immediately overwritten if another sequence CPU module is using the same device.
- If a power failure occurs when a monitor command is in use, it is necessary to set it again.
- The maximum text length that can be transmitted or received each time by the personal computer link function is 512 bytes. However, the maximum size that can be received by a higher-level computer may be limited to 256 bytes in some cases. In such cases, make sure that the response text length does not exceed 256 bytes by reducing the number of devices to be read.

A6.12 Device Management Function

The device management function enables you to upload, download, edit and compare device information/data of the sequence CPU module using WideField2.
You can specify the range of device data to be uploaded or downloaded.
You can also use this function to perform initial setup of device data when, for example, replacing the CPU module.
The devices that you can configure using the device management function are:
Internal relays (I), shared relays (E), time-out relays and current values of timers (T), end-of-count relays and current values of timers (C), data registers (D), shared registers (R), link registers (W), index registers (V) and file registers (B).
You cannot configure the following devices:
I/O relays (X/Y), preset values of timers (T) and counters (C), special relays (M) and special registers (Z).
The device management function serves the following four purposes.

■ Upload Device Data

The device management function allows you to read device information/data from the sequence CPU module and saves it to a WideField2 file. You can specify the range of devices to be saved.

■ Download Device Data

The device management function allows you to read device information/data from a WideField2 file and writes it to the sequence CPU module. You can either download all device data from the file or download part of the data by specifying a range of devices.

■ Edit Device Data

The device management function allows you to edit device information/data in a WideField2 file. You can view and change the current value of each device.

■ Compare Device Data

The device management function allows you to compare device information/data in the sequence CPU module with that in a WideField2 file. You can make a comparison of all device data in the file or part of the data by specifying a range of devices. If any mismatch is found, the function shows the device name and content of the mismatch.

A6.13 Macro Instructions

Macro instructions allow reuse of created programs for increased programming efficiency. In addition, the use of macro instructions allows compact program codes structured by function, thus improving program readability and maintainability.

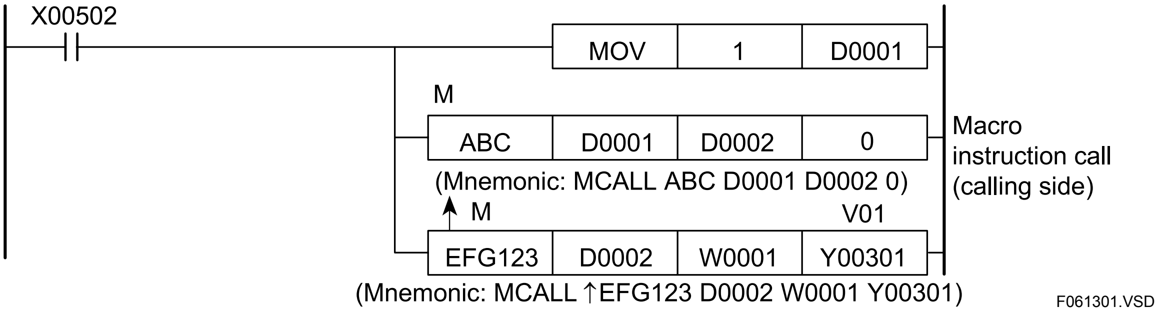
A6.13.1 What Are Macro Instructions?

■ Overview

A macro instruction enables a process requiring multiple instructions/steps to be processed as a single instruction.

The figure below presents an overview of macro instructions.

How to code a macro instruction in ladder diagram editing


X00502
MOV
1
D0001
M
ABC
D0001
D0002
0
(Mnemonic: MCALL ABC D0001 D0002 0)
M
V01
EFG123
D0002
W0001
Y00301
(Mnemonic: MCALL ↑EFG123 D0002 W0001 Y00301)
Macro instruction call (calling side)
F061301.VSD

How to code a macro instruction entity "ABC" in ladder macro editing

"ABC" macro instruction entity (called side)

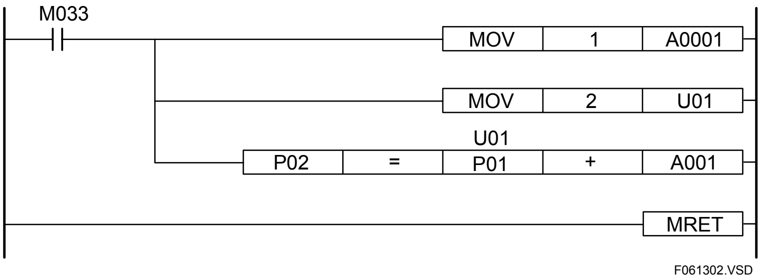


Figure A6.13.1 Examples of Macro Instructions

In the above figure, “ABC” and “EFG123” instructions are macro instructions. When the CPU encounters the “ABC” instruction, it executes the “ABC” macro instruction entity like a subroutine, using “D0001” and “D0002” as parameters. Macro instructions are created using ladder macro editing, separately from normal instructions created using ladder diagram editing.

The Macro Return (MRET) instruction represents the end of a macro instruction entity.

SEE ALSO

- For details on parameters P01, P02, and U01 in the figure, see Subsection A6.13.3, "Devices Dedicated to Macro Instructions.”
- For details on the Macro Return (MRET) instruction, see Section 3.13.4, "Macro Call (MCALL), Parameter (PARA), Macro Return (MRET)” of "Sequence CPU – Instructions User's Manual" (IM34M6P12-03E).

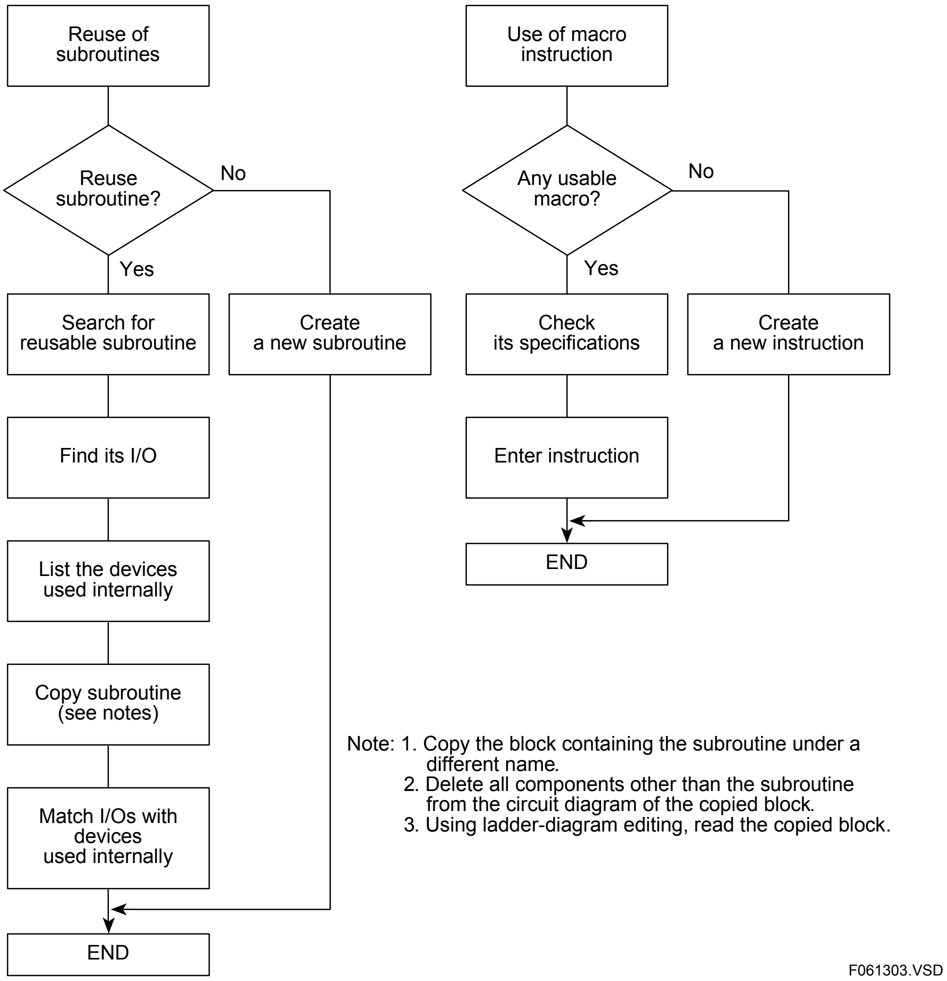
■ Purpose

Using macro instructions offers the following two advantages.

● Increased Programming Efficiency

Like subroutines, macro instructions allow grouping of similar processing. Macro instructions differ from subroutines, however, on the following two points.

- **Parameters can be passed to macro instructions.**
  Subroutines require the use of instructions for passing parameters (e.g. MOV instructions) preceding a CALL instruction.
- **Macros can be handled as instructions.**
  A user need not be aware of the internal processing of a macro, except for its input and output parameters.



Note: 1. Copy the block containing the subroutine under a different name.
2. Delete all components other than the subroutine from the circuit diagram of the copied block.
3. Using ladder-diagram editing, read the copied block.

Figure A6.13.2 Differences between Subroutines and Macro Instructions

● Accumulation of Know-how

Know-how can be accumulated in the form of macro instructions for creation of customized FA-M3 controllers.

Types of Macro Instructions

There are three types of macro instructions, the availability of which depends on CPU types as follows:

- Macro Call (MCALL)
- Input Macro Instruction Call (NCALL)
- Structure Macro Instruction Call (SCALL)

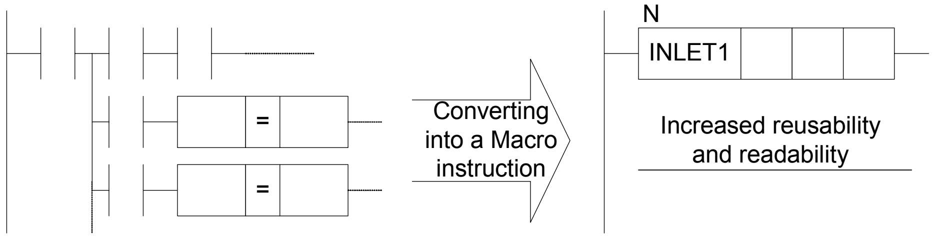
Macro Call

Up to 16 parameters can be passed via a Macro Call instruction.

Input Macro Instruction Call

The Input Macro Instruction Call instruction can be used as an input condition, just like the Load or Compare instruction. It can be used to represent complex or reusable input conditions in a single instruction.

Using an Output of Input Macro (NMOUT) instruction in an Input Macro Instruction call allows you to output the result of logical operations to the next instruction.


=
=
Converting into a Macro instruction
N
INLET1
Increased reusability and readability

F061304.VSD

Figure A6.13.3 Benefits of Input Macro Instruction Call

Structure Macro Instruction Call

The Structure Macro Instruction Call instruction passes multiple data items collectively in a structure to a macro instruction, and is especially useful in reducing the number of items to be passed to a macro instruction and providing better representation of a group of related data items.

A6.13.2 Specification of Macro Instructions

■ Number of Macro Instructions

Macro instruction entities to be called are downloaded, along with user ladder programs, from a personal computer to the sequence CPU module using WideField2. The table below lists the maximum number of macro instruction entities allowed in one executable program during downloading. A macro instruction can be called any number of times in a user ladder program.

Table A6.13.1 Maximum Number of Macro Instructions Allowed by CPU Type

Types	Maximum Number of Instructions Allowed
Macro Call (MCALL)	256 in total
Input Macro Instruction Call (NCALL)	
Structure Macro Instruction Call (SCALL)	

■ Size of Macro Instruction Program

The size of a macro instruction program is limited by the total size of that program and user programs combined.

■ Macro Instruction Execution Time

Table A6.13.2 Macro Instruction Execution Time

FUN NO.	Instruction	Mnemonic	F3SP66, F3SP67	
			When Executed (μs)	When Not Executed (μs)
996	Macro Call	MCALL	4.0	0.175
995	Parameter	PARA	2.5	0.105
998	Macro Return	MRET	2.0	—
981	Input Macro Instruction Call	NCALL	4.0	0.175
309	Output of Input Macro	NMOUT	1.0	0.070
985	Structure Macro Instruction Call	SCALL	15.7	1.265

■ Online Editing of Macro Instructions

You can use the online edit function of WideField2 to edit circuits containing macro instruction calls or input macro instruction calls. However, you can only use macro instructions that are already downloaded, and cannot create any new macro instruction. Circuits containing structure macro instruction calls cannot be edited online.

You can also use the online edit function to edit macro instruction entities already downloaded, but you cannot edit any circuits following the Macro Return (MRET) instruction.

A6.13.3 Devices Dedicated to Macro Instructions

Table A6.13.3 Devices Dedicated to Macro Instructions

Device	Symbol	Range	Number of Devices
Pointer register (P)	P	P01 to P16	16
Macro relay (H)	H	H0001 to H0512	512
Macro register (A)	A	A0001 to A0512	512
Macro index register (U)	U	U01 to U16	16
Structure pointer register (Q)	Q	Q01, Q02	2

■ Pointer (P) Registers

Pointer registers are used specifically to pass parameters to macro instructions. These registers can be used within macro instruction entities. Structure macro instructions use structure pointer registers instead of pointer registers.

The relationship between pointer registers (P) and macro instruction parameters is shown in the following figure.

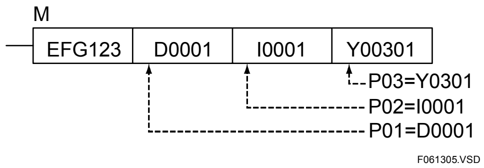


Figure A6.13.4 Relationship between Pointer Registers and Macro Instruction Parameters

Table A6.13.4 Relationship between Pointer Registers and Macro Instruction Parameters

Operand No.	Pointer Register No.
1	P01
2	P02
3	P03
4	P04
⋮	⋮
16	P16

Parameters that can be directly passed using a macro instruction call (1 to 3)

Parameters that can be passed using a parameter instruction (4 to 16)

F061306.VSD

Within a macro instruction entity, you can read from and write to pointer registers using basic or application instructions, in the same way as for devices passed as parameters. You can also perform word/long word processing, index modification, and automatic BIN-to-BCD or BCD-to-BIN conversion on these pointer registers.

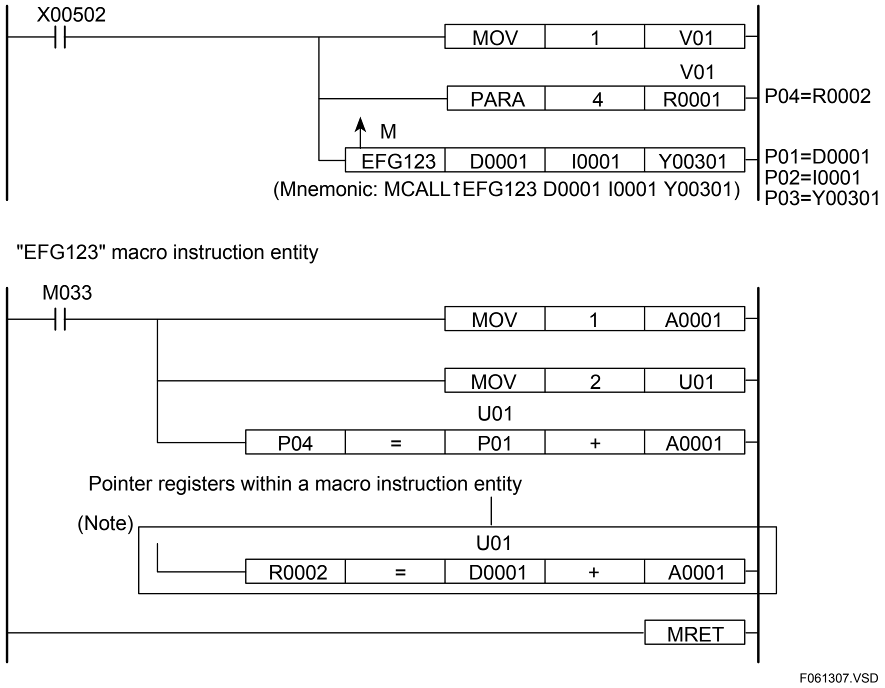
High speed processing of application instructions is not performed, however. More specifically, within a macro instruction entity, high speed processing is not performed for MOV, CAL, CMP, or logical operation instructions with pointer registers specified as parameters.

TIP

When an instruction using a pointer register (P) is to be executed repeatedly, you can first transfer the values of the pointer registers (P) to macro relays (H) and macro registers (A) and then rewrite the instruction to use these relays and registers instead. In this way, you can shorten the execution time.

SEE ALSO

For details on basic and application instructions, see Sections 2.1 and 3.1 of "Sequence CPU - Instructions User's Manual" (IM34M6P12-03E).



Note: Pointer registers can be used within a macro instruction entity.

Figure A6.13.5 Example Using Pointer Registers (P)

CAUTION

- If you pass a device with index modification as a parameter to a macro instruction, the instruction receives the index-modified device. In the example shown in the above figure, parameter R0001;V01 is the same as device R0002 because V01 = 1.
- Any index modification of a pointer register (P) is applied to the parameter that is passed. In the example shown in the above figure, P01;U01 is the same as device D0003 because P01 = D0001 and U01 = 2.

■ Macro Relays (H), Macro Registers (A) and Macro Index Registers (U)

These devices are dedicated to macro instructions. Within a macro instruction entity, you can read from and write to macro relays, macro registers or macro index registers using basic or application instructions, the same way as for internal relays (I), data registers (D) and index registers (V). These devices can be used within a macro instruction entity.

By using these devices in your macro entity, you need not know which devices are used in the macro instruction call. Needless to say, the values of these devices remain unchanged.

■ Structure Pointer Registers (Q)

Structure Pointer Registers are dedicated registers used for passing structure data to structure macro instructions. It is used within structure macro instruction entities.

The relationship between structure pointer registers (Q) and structure macro instruction parameters is shown in the figure below.

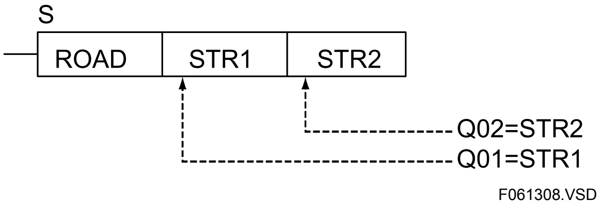


Figure A6.13.6 Relationship between Structure Pointer Registers (Q) and Structure Macro Instruction Parameters

Table A6.13.5 Relationship between Structure Pointer Registers (Q) and Structure Macro Instruction Parameters

Operand	Structure pointer register number
1 (parameter 1)	Q01
2 (parameter 2)	Q02

Within a structure macro instruction entity, you can read from and write to structure data passed as parameters using basic or application instructions and referring to structure members using the "<structure pointer register number>.<structure member name>" syntax.

Word processing, long-word processing and automatic BIN-to-BCD or BCD-to-BIN conversion can be used with structure pointer registers, but index modification is not allowed.

High speed processing of application instructions is not performed, however. More specifically, within a structure macro instruction entity, high speed processing is not performed for MOV, CAL, CMP, or logical operation instructions with structure pointer registers (Q) specified as parameters.

TIP

When an instruction using a structure pointer register (Q) is to be executed repeatedly, you can first transfer the member data to macro relays (H) and macro registers (A) and then rewrite the instruction to use these relays and registers instead. In this way, you can shorten the execution time.

SEE ALSO

- For details on basic and application instructions, see Sections 2.1 and 3.1 of the "Sequence CPU – Instructions User's Manual" (IM34M6P12-03E).
- For details on structures, see "FA-M3 Programming Tool WideField2 User's Manual" (IM34M6Q15-01E).

A6.13.4 Nesting Macro Instructions

Nesting macro instructions is to call another macro instruction or an input macro instruction when executing a macro instruction.

Calling a structure macro instruction within a structure macro instruction body is not allowed.

Calling another macro instruction or an input macro instruction from a structure macro instruction body is allowed but the PARA instruction cannot be used.

Nesting macro instruction calls beyond seven levels will cause an instruction processing error. The nesting depth is stored in special register Z106. A value of “0” is stored in the special register Z106 during non-nested execution of a macro instruction.

Table A6.13.6 Calls between Macros, Input Macros, and Structure Macros

Calling Side	Called Side	Availability
Block	Macro	✓
Block	Input macro	✓
Block	Structure macro	✓
Macro	Macro	△
Macro	Input macro	△
Macro	Structure macro	×
Input macro	Macro	△
Input macro	Input macro	△
Input macro	Structure macro	×
Structure macro	Macro	△
Structure macro	Input macro	△
Structure macro	Structure macro	×

✓ : Call is allowed (PARA instruction can be used).
△: Parameters passed using the PARA instruction are overwritten.
× : Call is not allowed.

CAUTION

- Parameters 1 to 3 passed to macro instructions are saved when macro instructions are nested. However, parameters 4 to 16 passed using PARA (parameter) instructions are not saved. If a Parameter (PARA) instruction is executed in a called macro instruction, the relevant parameters are overwritten.
- Errors generated in nested macro instructions are reported as errors of the first macro instruction.

SEE ALSO

For details on the Parameter (PARA) instruction, see Section 3.13.4, "Macro Call (MCALL), Parameter (PARA), Macro Return (MRET)” of "Sequence CPU – Instructions User's Manual" (IM34M6P12-03E).

When nesting macro instructions, you may mistakenly overwrite macro devices, such as relays, registers and index registers, in a called macro instruction and thereby destroy their data. To avoid this problem, check the depth of macro instruction nesting stored in special register Z106 and use macro devices separately for each level of nesting depth (see the example below).

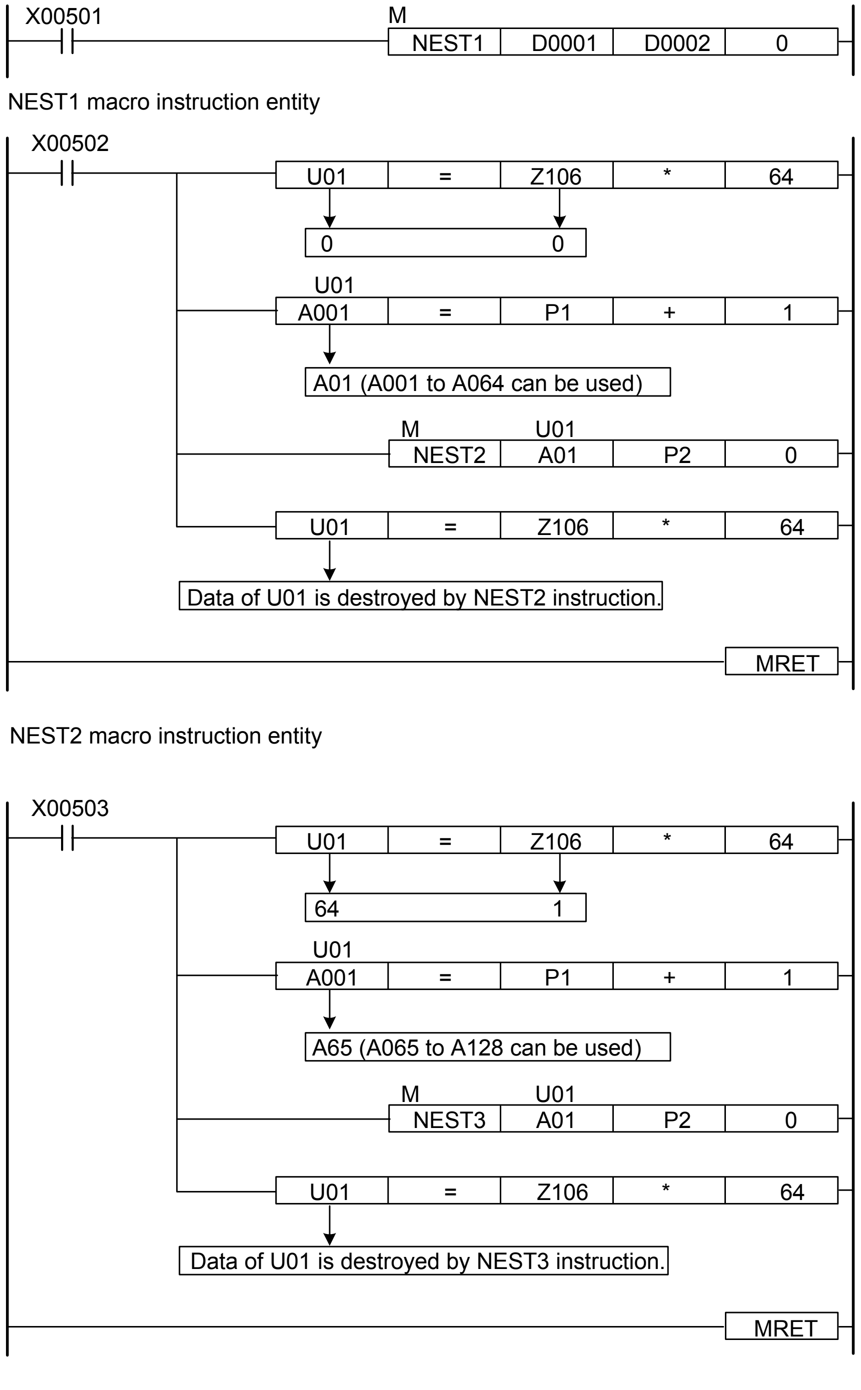

X00501
M
NEST1
D0001
D0002
0
NEST1 macro instruction entity
X00502
U01
=
Z106
*
64
0
0
U01
A001
=
P1
+
1
A01 (A001 to A064 can be used)
M
U01
NEST2
A01
P2
0
U01
=
Z106
*
64
Data of U01 is destroyed by NEST2 instruction.
MRET
NEST2 macro instruction entity
X00503
U01
=
Z106
*
64
64
1
U01
A001
=
P1
+
1
A65 (A065 to A128 can be used)
M
U01
NEST3
A01
P2
0
U01
=
Z106
*
64
Data of U01 is destroyed by NEST3 instruction.
MRET
F0613081.VSD

Figure A6.13.7 Example of Macro Device Separation when Nesting Macro Instructions

A6.13.5 Handling Macro Instruction Errors

When creating a program using a macro instruction tool, an error is generated if:

- There are two or more macro instructions of the same name;
- A macro instruction specified in a macro instruction call (MCALL) is not found; or
- A macro instruction entity contains two or more macro return (MRET) instructions.

An error is also generated and the special relay M201 for instruction processing errors is set to ON if:

- A macro return (MRET) instruction is executed before a macro instruction call (MCALL) (special register Z022 contains the error code $2501); or
- The depth of macro instruction call nesting exceeds 7 levels (special register Z022 contains the error code $2502).

An error detected within a macro instruction entity is seen by the user as an error of the macro instruction. Thus, the user can know which parameters were passed to the macro instruction.

CAUTION

Any error detected by self-diagnosis (except for a memory checksum error) within a macro instruction entity is also seen by the user as an error of the macro instruction execution.

Table A6.13.7 Error Codes for Macro Instructions

Error Type	Error Name	Error Code	Description
Instruction processing	Macro instruction error	$2501	There is no return destination.
		$2502	The maximum nesting depth (seven levels) is exceeded.

A6.13.6 Protecting Macro Instructions

You can protect macro instructions against unauthorized read access. The protection can be configured on per macro instruction basis by entering a password using WideField2. A password can consist of up to eight alphanumeric characters. The protection information is saved in the management information area of a macro instruction file. A protected macro instruction can be edited, printed or monitored only if the password matches.

TIP

Executable program protection and block protection also apply to user-created ladder programs containing macro instructions. For instance, if executable program protection is enabled, downloading, uploading, monitoring, online-editing and other operations on the executable program are not allowed.

A6.13.7 Debugging Operation

■ Forced Set and Forced Reset

You can force bit devices to turn ON or turn OFF in macro instructions (either in a macro instruction call or within a macro instruction entity).

■ Partial Operation

Partial operation is not allowed within a macro instruction entity.

A6.13.8 Input Macro Instructions

An Input Macro Instruction is a type of macro instruction that can be used as an input condition, just like the Load or Compare instruction. It can represent complex, reusable input conditions as a single instruction.

By calling the Output of Input Macro (NMOUT) instruction internally, an input macro instruction can also output the result of logical operation to the next instruction.

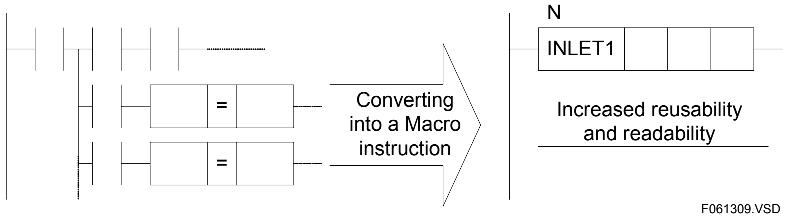

=
=
Converting into a Macro instruction
N
INLET1
Increased reusability and readability
F061309.VSD

Figure A6.13.8 Benefits of Input Macro Instructions

■ How to Use

● Creating an Input Macro Instruction

Input macro instructions can be created like ordinary macro instructions.

Macro instructions called by the Input Macro Instruction Call (NCALL) instruction are called input macro instructions.

Thus, the same macro instruction entity can be either an input macro instruction (if called by NCALL) or a macro instruction (if called by MCALL).

● Calling an Input Macro Instruction

Use the Input Macro Instruction Call (NCALL) instruction to call an input macro instruction.

SEE ALSO

For details on the NCALL instruction, see "Sequence CPU – Instructions User's Manual" (IM34M6P12-03E).

● Where to Code an Input Macro Instruction Call (NCALL) Instruction

You can code an NCALL instruction along with the **LOAD**, **AND**, or **OR** logical operator.

You cannot use it in place of an output instruction (at the right end of a ladder rung).

To call a macro instruction at the position of an output instruction, use the MCALL instruction instead.

● Passing Parameters to an Input Macro Instruction

Use the pointer register (P) to pass parameters to an input macro instruction, the same way as with ordinary macro instructions.

Up to three parameters can be directly coded in an NCALL instruction. To pass more than three parameters, use the Parameter (PARA) instruction. Be careful when using the PARA instruction, because it can be used by both macro and input macro instructions.

SEE ALSO

For details on pointer registers (P), see Section A6.13.3, "Devices Dedicated to Macro Instructions."

● Output of Logical Operation Result to the Power Rail

The NMOUT instruction is used to specify the logical operation result of an input macro instruction. The logical operation result to be output to the step following the Input Macro Instruction Call instruction depends on the status of the input parameter type of the NMOUT instruction.

Input Parameter	Logical Operation Output of Input Macro (device status = output)
Constant	OFF if 0, ON if otherwise
Relay device	OFF if 0, ON If 1
Register device	OFF if 0, ON if otherwise

If the NMOUT instruction is executed more than once, the last instruction takes precedence.

If no NMOUT instruction is executed, the logical operation result of an input macro is OFF.

SEE ALSO

For details on the NMOUT instruction, see "Sequence CPU – Instructions User's Manual" (IM34M6P12-03E).

CAUTION

The NMOUT instruction takes effect only if executed within a macro instruction that has been called by NCALL (that is, an input macro). It is ignored if executed within a macro instruction that has been called by MCALL.

● Nesting Input Macros

Macro and input macro instructions when combined may be nested up to 8 levels.

A6.13.9 Structure Macro Instructions

The Structure Macro Instruction passes a number of parameters collectively in a structure to a macro instruction. By using a structure, it simplifies data passing and improves representation in cases where there are many related parameters.

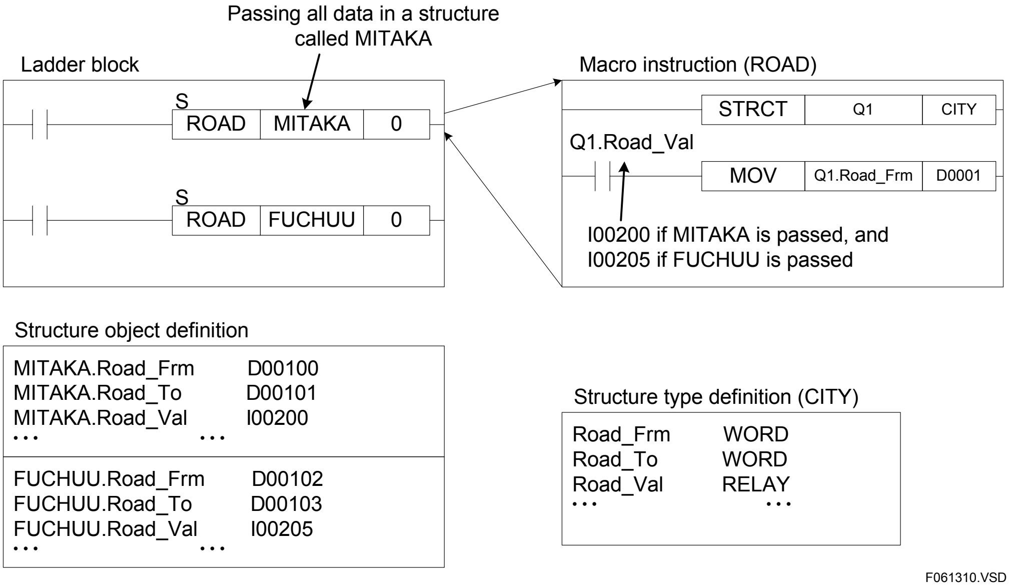


Figure A6.13.9 Benefits of Structure Macro Instructions

SEE ALSO

For details on structures, see "FA-M3 Programming Tool WideField2 User's Manual" (IM34M6Q15-01E).

CAUTION

- A structure macro instruction may not call another structure macro instruction. A structure macro instruction may be called only by a block.
- If the type of a structure passed using a structure macro instruction is different from the structure type declared by a structure pointer declaration instruction in the called structure macro instruction, the latter structure type is used during execution with no error generated.
- Structure macros use pointer registers P4 to P8.

■ How to Use

● Structure Type Definition

Defines the name and members of a structure type.

● Structure Object Definition

Allocates actual registers to structure data.

● Creating Structure Macro Instructions

Structure macro instructions can be created just like ordinary macro instructions.

Macro instructions called by the structure Macro Instruction Call (NCALL) instruction are called structure macro instructions.

● Structure Type Declaration (STRCT) for Structure Macro Instructions

At the very beginning of a structure macro instruction, you must declare the type of the structure to be passed.

One structure type declaration is required if one structure is to be passed. Two structure type declarations are required if two structures are to be passed.

● Calling a Structure Macro Instruction

Use the structure Macro Instruction Call (SCALL) instruction to call a structure macro instruction.

SEE ALSO

For details on the SCALL instruction, see "Sequence CPU – Instructions User's Manual" (IM34M6P12-03E).

● Passing Structure Data to a Structure Macro Instruction

When passing structure data, code the name of the structure in the macro instruction call, but use a structure pointer register (Q) within the macro instruction entity.

SEE ALSO

For details on structure pointer registers (Q), see Section A6.13.3, "Devices Dedicated to Macro Instructions."

● Nesting structure Macros

A structure macro instruction cannot call another structure macro instruction.

A structure macro instruction can call macro and input macro instructions, but cannot use the PARA instruction.

A6.14 User Log Management Function

The user log management function keeps a record of error events in a user system, including information on error occurrence and system operation status, when Save User Log instructions are executed. Stored user log records can then be read using instructions or WideField2. This function is useful for analyzing faults and understanding the operating conditions of machinery.

■ Handling User Logs

A maximum of 64 user log records per CPU can be saved by executing user log instructions in a program.

Four data items, namely, date of occurrence, time of occurrence, main code (one word) and subcode (one word), are saved in each user log record.

Up to sixty-four 32-character messages associated with individual main codes can be stored in the CPU. These stored messages can be retrieved along with main codes when log information is read.

The user log information area is maintained as a rotary buffer. If the maximum number of log records allowed is exceeded, existing log records are overwritten by new log records in chronological order.

Stored user log records can be read using WideField2 or Read User Log instructions. You can check the special register Z105 for the number of stored user log records available.

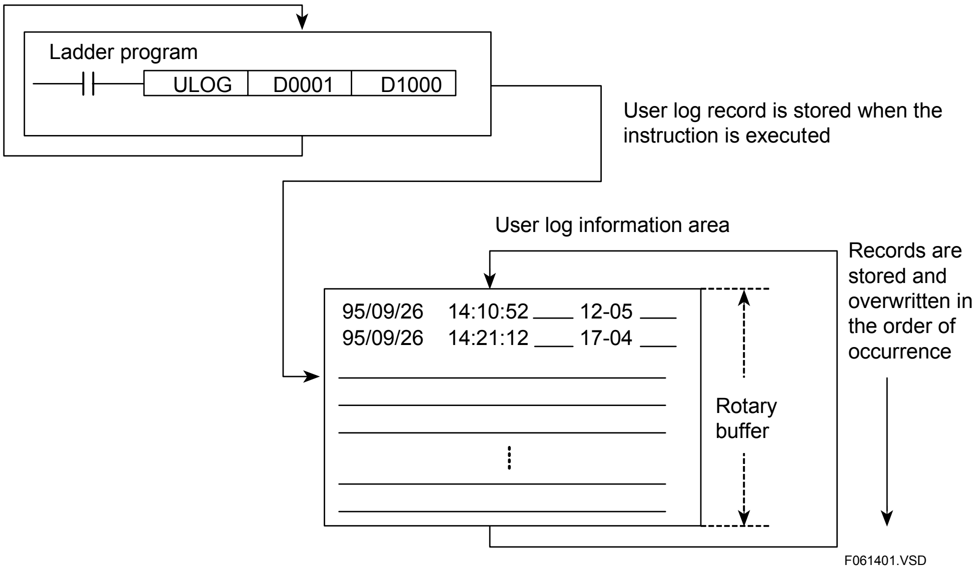

Ladder program
ULOG
D0001
D1000
User log record is stored when the instruction is executed
User log information area
95/09/26 14:10:52 12-05
95/09/26 14:21:12 17-04
Rotary buffer
Records are stored and overwritten in the order of occurrence
F061401.VSD

Figure A6.14.1 Handling User Logs

CAUTION

In some cases, WideField2 may display two identical log records. This happens if a save user log instruction is executed when stored user log records is read using WideField2. To solve the problem, redisplay log records when not executing a save user log instruction.

SEE ALSO

For details on the instructions related to user log, see Subsection 3.22.5, "Save User Log (ULOG), Read User Log (ULOGR) and Clear User Log (UCLR)" of "Sequence CPU – Instructions User's Manual" (IM34M6P12-03E).

TIP

- You may assign any values from -32768 to +32767 to user log main codes. Messages for main codes 1 to 64 can be stored in a CPU module.
- You may also assign any values from -32768 to +32767 to user log sub codes.

A6.15 Sensor Control Function

To enable request responses at speeds of several hundred microseconds, a small PLC or sensor controller is often installed alongside the main PLC. The sensor control function plays the role of such a controller, enabling a program to be scanned at high speeds and fixed intervals independently and not affected by the main scan time, which tends to lengthen due to advanced functionality or high performance of the system.
Using the sensor control function, you can execute one program block at high speeds and fixed intervals (200 µs minimum), independent of the regular scan. This function is useful for control applications requiring higher machining accuracy.

A6.15.1 Schematic Operation Diagram

The figure below shows the operation of a sensor control block.

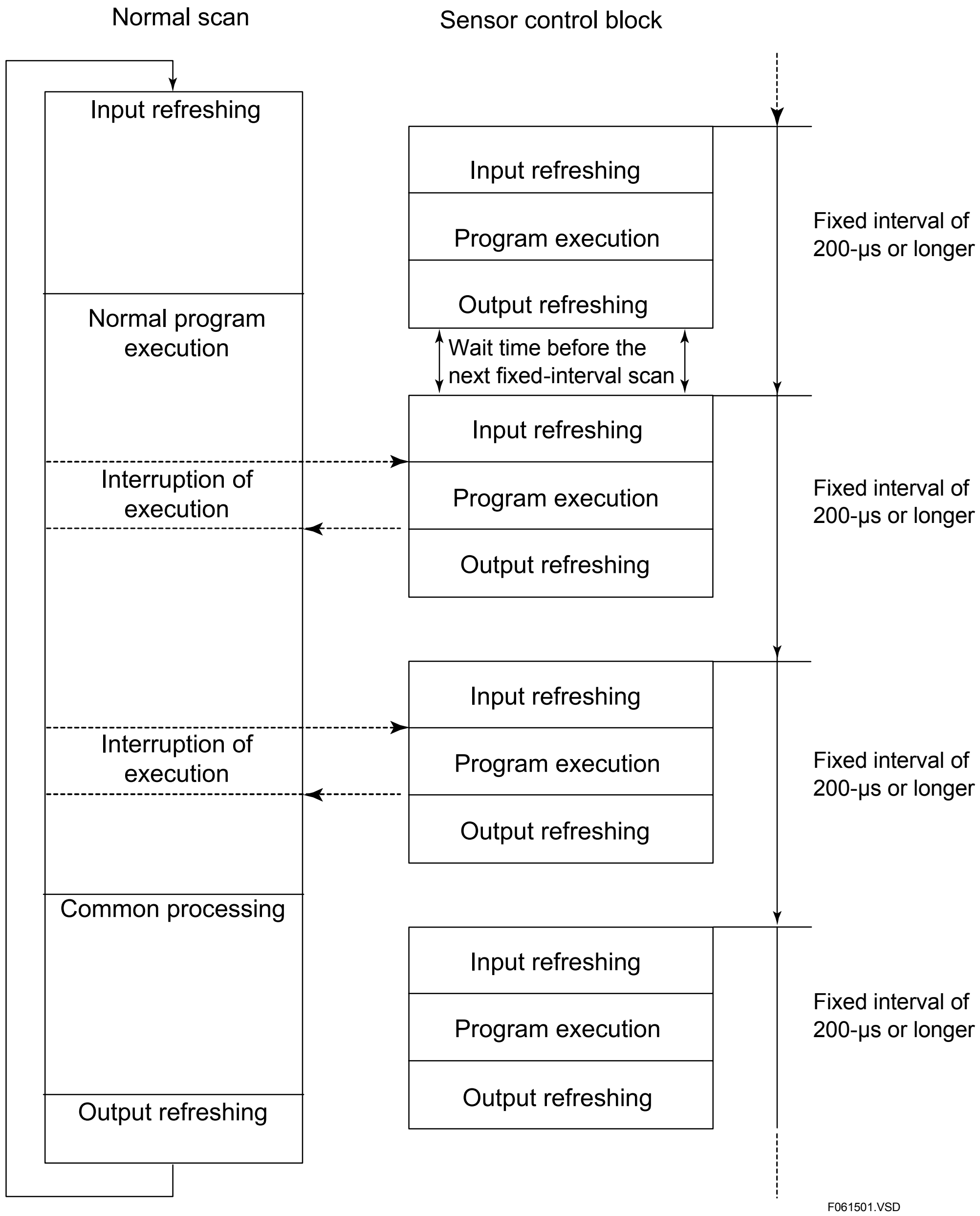


Figure A6.15.1 Schematic Diagram of Sensor Control Block Operation

A6.15.2 Features

■ Features

The sensor control function has the following features.

● High Speed

- The minimum interval of block execution is as short as 200 μs.
- The sequence CPU module operates as if it contains another sequence CPU module with the minimum scan time of 200 μs.
- The maximum I/O response delay is only 400 μs, i.e., twice the minimum interval of block execution.
  You can use it for a process requiring fast I/O response by isolating the process from the regular program.
- It allows the use of a wide choice of modules, including special modules for input/output.

● Fixed Interval

- The sensor control block is executed at fixed intervals.
- The sensor control block function runs even during instruction execution, refreshing or common processing of the normal scan.

A6.15.3 Specifications and Restrictions

■ Specifications

Table A6.15.1 Specifications of Sensor Control Block

Item	Specifications
Number of sensor control blocks	1
Execution interval	200 μs to 25.0 ms, in 100 μs increments
Compatible modules	All types of modules *1
Unit of I/O-refreshed devices	Per word basis, i.e., in units of 16 relays or terminals.
Maximum number of I/O-refreshed words	4 to 512
Applicable instructions	All instructions, except for the Timer, special module High-speed Read (HRD) and special module High-speed Write (HWR) instructions
Applicable devices	All device types *2
Maximum program execution time	50 μs to 24.95 ms*1
Initial condition at normal program execution	Sensor control block at a stop
Interrupt timing	Configurable as "After instruction (execution)" or "Immediate (during instruction execution)." The default is "Immediate."
Priority of interrupts	Selectable as "Sensor CB interrupt has priority" (default) or "Input interrupt has priority." *1
Other functions	Start, stop or interrupt prohibition by means of instruction or tool.

*1: See "CAUTION" under "■ Compatible Modules."

*2: For details on writing to relay devices common to the normal scan, see Subsection A6.15.7, "Programming Precautions."

■ Compatible Modules

The sensor control block function can be used with all CPU modules. However, observe the precautions described below.

CAUTION

Precautions when setting “Terminal Usage” in DIO setup

Using the configuration function of WideField2, specify whether terminals of I/O modules are to be refreshed by the sensor control block or normal scan.

All devices used in the sensor control block including input relays (X) and output relays (Y) are shared with the normal scan.

When using an input or output module with the sensor control block, you must configure the input module on long word basis (i.e., 32 relays, 32 terminals, terminals 1 to 32 or terminals 33 to 64); and the output module on word basis (i.e., 16 relays or 16 terminals).

Let's suppose you configured the input module incorrectly on word basis (for example, you set terminals 1 to 16 to the option “Used” [with normal scan] and terminals 17 to 32 to the option “Use with SCB”). Since input refreshing is performed on long word basis, input relays (X) used under a normal scan are refreshed by the Refresh instruction of the sensor control block when the normal scan is in progress. Consequently, the simultaneity of data is not guaranteed before and after the refreshing. Simultaneity of data is also not guaranteed for input relays (X) used in the sensor control block.

Relationship between shared refreshing and link refreshing

Shared and extended shared relays (E), shared and extended shared registers (R), link relays (L) and link registers (W) are not refreshed by I/O refreshing of the sensor control block. Instead, these devices are refreshed during common (synchronization) processing in a normal scan.

■ Maximum Number of I/O-refreshed Words

The maximum number of words that can be configured for refreshing depends on the type of unit (main unit or subunit) where the module is installed, the execution interval and the number of installed CPU modules (including BASIC CPU modules) as show in the table below.

You can calculate the number of words according to the following equation.

Number of I/O-refreshed words

= (Number of words of I/O modules in main unit to be refreshed)

+ (Number of F3XH04 modules in main unit to be refreshed that use pulse catch function)

+ (Number of words of I/O modules in subunit to be refreshed) x 4

+ (Number of F3XH04 modules in subunit to be refreshed that use pulse catch function) x 4

Table A6.15.2 Maximum Number of I/O-refreshed Words Handled by Sensor Control Block for each CPU

Execution Interval	Maximum Number of I/O-refreshed Words			
	One CPU	Two CPUs	Three CPUs	Four CPUs
200 μs	4	0	0	0
300 μs	8	4	0	0
400 μs	12	4	4	0
500 μs	16	8	4	4
1 ms	36	16	12	8
2 ms	76	36	24	16
3 ms	116	56	36	28
4 ms	156	76	52	36
5 ms	196	96	64	48
10 ms	396	196	132	96
20 ms	512	396	264	196
25 ms	512	496	332	248

Example 1:

Number of CPU modules installed: 2
Execution interval of sensor control block at CPU1: 1 ms
Sum of input-refreshed and output-refreshed words for CPU1
: Should be kept below 16.
Execution interval of sensor control block at CPU2: 500 μs
Sum of input-refreshed and output-refreshed words of CPU2
: Should be kept below 8.

The maximum number of I/O-refreshed words is proportional to the execution interval. You can calculate the maximum number for any execution interval not given in the above table by using the maximum numbers given for the execution intervals immediately above and below the required execution interval.

Example 2:

Number of CPU modules installed:1
Execution interval of sensor control block: 600 μs
Maximum number of I/O-refreshed words: 20

If the maximum number of I/O-refreshed words is exceeded, the execution interval of the sensor control block cannot be maintained. This may result in a sensor CB scan timeout error.

SEE ALSO

For details on the module operation in the event of a sensor CB scan timeout error, see subsection A6.15.6, “Error Handling.”

■ Maximum Program Execution Time

You should keep the program execution time of the sensor CB as short as possible. Otherwise, a program executed under a normal scan will be interrupted for a longer duration, and the time interval of the normal scan will become longer.

Calculate the maximum program execution time using the equation given below. If this maximum time is exceeded, a sensor CB scan timeout error may result because the CPU is unable to maintain the execution interval of the sensor control block.

SEE ALSO

For details on the module operation in the event of a sensor CB scan timeout error, see subsection A6.15.6, "Error Handling".

Maximum program execution time (μs)
= (Maximum number of I/O-refreshed words discussed earlier
- Number of words actually refreshed)
x Number of CPU modules installed x 25 μs + 50 μs

Example 1:
Execution interval: 200 μs
Number of CPU modules installed: 1

Maximum number of I/O-refreshed words: 4 (from previous Table)
Number of words actually refreshed: 2
Maximum program execution time = (4 - 2) x 1 x 25 + 50 = 100 μs

The sensor control block, if composed of basic instructions only, is equivalent to a program with the following number of steps.
100/0.035 = 2856 steps = Approximately 2.8K steps
(for F3SP66 and F3SP67 CPU modules)

Example 2:
Execution interval: 1 ms
Number of CPU modules installed: 1

Maximum number of I/O-refreshed words: 36 (from previous Table)
Number of words actually refreshed: 6
Maximum program execution time = (36 - 6) x 1 x 25 + 50 = 800 μs

The sensor control block, if composed of basic instructions only, is equivalent to a program with the following number of steps.
800/0.035 = 42865 steps = Approximately 41.8K steps
(for F3SP66 and F3SP67 CPU modules)

A6.15.4 Functions

The following sensor control functions are available.

Table A6.15.3 Sensor Control Functions

Function	Description
Execution interval	Allows the execution interval of the sensor control block to be set using WideField2.
Interrupt timing	Allows the interrupt timing of the sensor control block to be set by WideField2 to either of the following options: - After instruction (execution) - Immediate (during instruction execution)
Priority of interrupts	Allows the precedence of sensor control block interrupt and input module interrupt to be set using WideField2.
Activate/Deactivate	Allows the sensor control block to be activated or deactivated using a dedicated instruction.
Disable/Enable	Prohibits the sensor control block from being executed or cancels the prohibition using a dedicated instruction.
On-for-one-scan-at-sensor-CB-start function	A special relay (M) that remains turned on for one scan when the sensor control block is activated.
Execution status	Reflects the status of the sensor control block operation in a special relay (M).
Execution time monitoring	Stores the processing time of the sensor control block in a special register (Z).

■ Execution Interval Setting and Accuracy

Using the configuration function of WideField2, set the execution interval of the sensor control block.

Table A6.15.4 Execution Interval Setpoints of Sensor Control Block

Item	Configuration Range
Setting range	200µs to 25.0 ms. The setting range from 200 µ s to 900 µ s is only valid for the interrupt timing option of “Immediate" (during instruction execution).
Unit of setpoint	100 µs

The accuracy of an execution interval is 100 ppm.

■ Interrupt Timing

Using the configuration function of WideField2, set the timing for sensor control block interrupts during program execution. Two interrupt timing options are available.

Table A6.15.5 Sensor Control Block Interrupt Timing

Interrupt Timing	Description
After instruction	The CPU switches to the sensor control block after the completion of instruction execution. It does not, however, switch to the sensor control block during common processing or refreshing.
Immediate (default)	The CPU switches to the sensor control block during ladder instruction execution. It also switches to the sensor control block during common processing or refreshing.

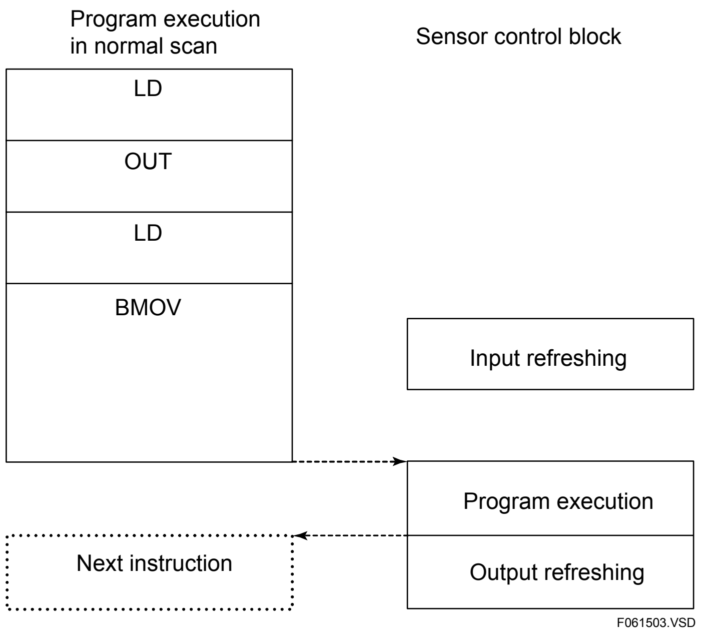


Figure A6.15.2 Interrupt by Sensor Control Block after Instruction Execution

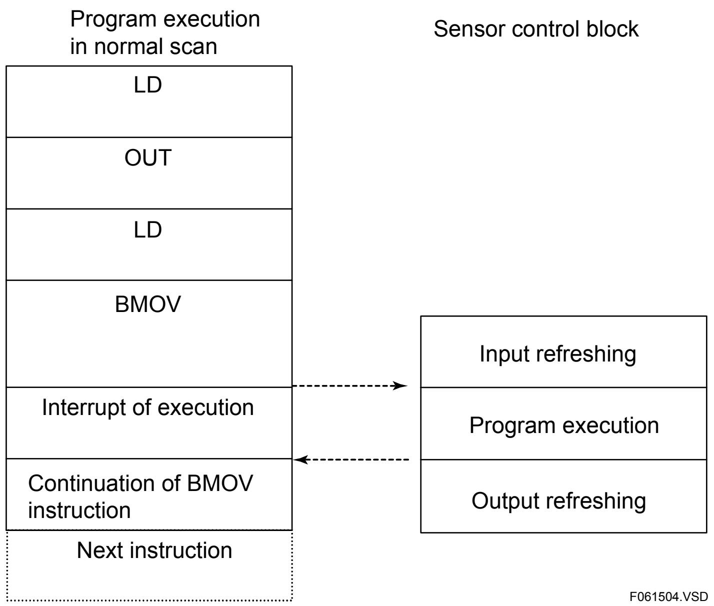


Figure A6.15.3 Immediate Interrupt by Sensor Control Block during Instruction Execution

The table below summarizes the differences between these two interrupt timing options.

Table A6.15.6 Differences between the Two Interrupt Timing Options

Item	When Interrupt Timing is Immediate	When Interrupt Timing is After Instruction Execution
Execution delay	Processing time of instruction being executed[*1] + Switchover processing time[*2], or Synchronization processing time[*3] + Common processing time[*3] + Input refreshing time[*3] + Switchover processing time[*2]	Switchover processing time only[*2]
Simultaneity of data	Guaranteed for each instruction	No simultaneity of multi-device data

[*1] For details on instruction processing time, see the instruction list in the Appendix of "Sequence CPU – Instructions User's Manual" (IM34M6P12-03E).

[*2] 3 to 10 μs for F3SP66 and F3SP67 CPU modules.

[*3] See Section A7.1, "Description of Scan Time."

CAUTION

Setting execution interval when the interrupt timing is after instruction execution

If the interrupt timing is set to be after instruction execution, you should set the execution interval to 1 ms or longer.

With this interrupt timing option, the CPU does not switch to the sensor control block during common processing or refreshing. Although the common processing time or refreshing time varies depending on the duration of synchronization processing, such as shared refreshing or link refreshing, you should set the execution interval to at least 1 ms. Otherwise, the execution interval of the sensor control block cannot be maintained and this may result in a sensor CB scan timeout error.

Debug operation when the interrupt timing is after instruction execution

If the interrupt timing is set to be after instruction execution, a sensor scan timeout error may occur when switching from **Run** to **Debug** mode or from **Debug** to **Run** mode, or when canceling a forced set/reset in **Debug** mode.

Simultaneity of data when the interrupt timing is immediate

Simultaneity of data for multiple devices is not guaranteed if the sensor control block is executed immediately during instruction execution.

For example, consider the case shown in the previous figure where the sensor control block is executed during execution of a block transfer (BMOV) instruction in a normal scan. There is a risk that the source data that is partially transferred may be overwritten after the execution of the sensor control block or data transferred partially to the destination may be read by the sensor control block.

Simultaneity of data is required when data of multiple devices is exchanged between a normal-scan program and the sensor control block program using a block transfer (BMOV) instruction, a long-word instruction with IEEE single-precision floating point data, or two or more instructions.

If simultaneity of data is required when interrupt timing is configured as “Immediate (during instruction execution),” use any of the following means to ensure data simultaneity:

- Use a Disable Sensor Control Block (CBD) instruction and an Enable Sensor Control Block (CBE) instruction to prevent the sensor control block from being executed during exchange of multi-device data.
- Write an application program to perform flag control between the normal-scan program and the sensor control block using relays.

SEE ALSO

For details on the Disable Sensor Control Block (CBD) instruction and an Enable Sensor Control Block (CBE) instruction, see "■ Enabling/Disabling Sensor Control Block" later in this subsection.

■ Priority of Interrupts

You can specify the priority of interrupts by configuration using WideField2 for conflict resolution in the event that interrupt processing of an interrupt from an input module coincides with an interrupt from a sensor control block.

The table below lists the two options for “Priority of Interrupts”, along with how they work.

Table A6.15.7 Options for Priority of Interrupts

Priority of Interrupts	Functionality	
	When an interrupt from an input module occurs during execution of a sensor control block	When the time for executing a sensor control block arrives during input interrupt processing
Sensor CB interrupt has priority (default)	Executes the interrupt process after executing the sensor control block.	Suspends the interrupt process and resumes execution after executing the sensor control block.
Input interrupt has priority	Suspends the execution of the sensor control block and resumes execution after executing the interrupt process.	Suspends the sensor control block after executing the interrupt process.

TIP

The sequence CPU applies the rule of interrupt execution timing (after completion of instruction execution or immediately during instruction execution) discussed earlier, even in the case where execution of the sensor control block or interrupt process is suspended due to priority of interrupts.

■ Activating/Deactivating Sensor Control Block

You can activate the sensor control block using an Activate Sensor Control Block (CBACT) instruction, or stop the sensor control block using an Inactivate Sensor Control Block (CBINA) instruction. At the start of operation, the sensor control block defaults to the STOP status. To activate the sensor control block, execute an Activate Sensor Control Block (CBACT) instruction in a normal-scan program. The first execution of the sensor control block takes place within 100 μs after the execution of a CBACT instruction.

Table A6.15.8 Instructions to Activate or Inactivate the Sensor Control Block

Instruction	Description
CBACT instruction	Activates the sensor control block.
CBINA instruction	Inactivates the sensor control block.

SEE ALSO

For details on individual instructions, see "Sequence CPU – Instructions User's Manual" (IM34M6P12-03E).

TIP

- When the sensor control block stops, the CPU holds or resets the data of output (Y) relays used or refreshed by the sensor control block according to the setting of the configuration item “Output when Stopped.” When the sensor control block is activated, the relays are always set to the Hold option.
- The initialization processing of timers, counters and the destinations of OUT instructions does not apply to the sensor control block activated or inactivated by an ACT/INACT instruction.

■ Enabling/Disabling Sensor Control Block

You can temporarily disable the sensor control block using a CBD (Disable Sensor Control Block) instruction or enable the sensor control block using a CBE (Enable Sensor Control Block) instruction. If the sensor control block is disabled, the CPU does not execute it until it is enabled even if it is time for executing the sensor control block. In this case, the CPU immediately begins executing the block as soon as it is enabled.

Table A6.15.9 Instructions to Disable and Enable Execution of the Sensor Control Block

Instruction	Description
CBD Instruction	Disables execution of the sensor control block.
CBE Instruction	Enables execution of the sensor control block.

SEE ALSO

For details on individual instructions, see "Sequence CPU – Instructions User's Manual" (IM34M6P12-03E).

CAUTION

If the interval of execution disable is too long for the CPU to be able to execute the sensor control block at fixed intervals, a sensor CB scan timeout error will result. Consequently, the CPU stops executing the sensor control block.

SEE ALSO

See subsection A6.15.6, “Error Handling” for more information.

■ On-for-one-scan-at-Sensor-CB-start Function

This function causes a special relay (M) to remain turned on for one scan during the first execution of the sensor control block when the sensor control block starts.

Table A6.15.10 Special Relay (M) which Turns ON for One Scan at Sensor Control Block Startup

No.	Name	Status	Description
M097	On for One Scan at Sensor CB Start	ON: At block start OFF: In all other cases	Turns on for one scan when the sensor control block starts (at the first execution of the sensor control block).

TIP

The on-for-one-scan-at-Sensor-CB-start relay turns on when an Activate Sensor Control Block (CBACT) instruction is executed. It then turns off at the end of the first execution of the sensor control block.

■ Execution Status

The CPU stores in a special relay (M) the execution status of the sensor control block.

Table A6.15.11 Sensor CB Execution Status Special Relay (M)

No.	Name	Status	Description
M137	Sensor CB Execution Status	ON: Run OFF: Stop	Indicates the status of sensor control block operation.

TIP

The status of the sensor control block is updated when an Activate Sensor Control Block (CBACT) instruction is executed, or during normal-scan input refreshing after error detection when an Inactivate Sensor Control Block (CBINA) instruction is executed.

■ Execution Time Monitoring

This function stores in a special register (Z) the time taken from when input refreshing for the sensor control block is started, followed by program execution, to when output refreshing is completed . The time indicates the actual execution time of the sensor control block during the preset execution interval.

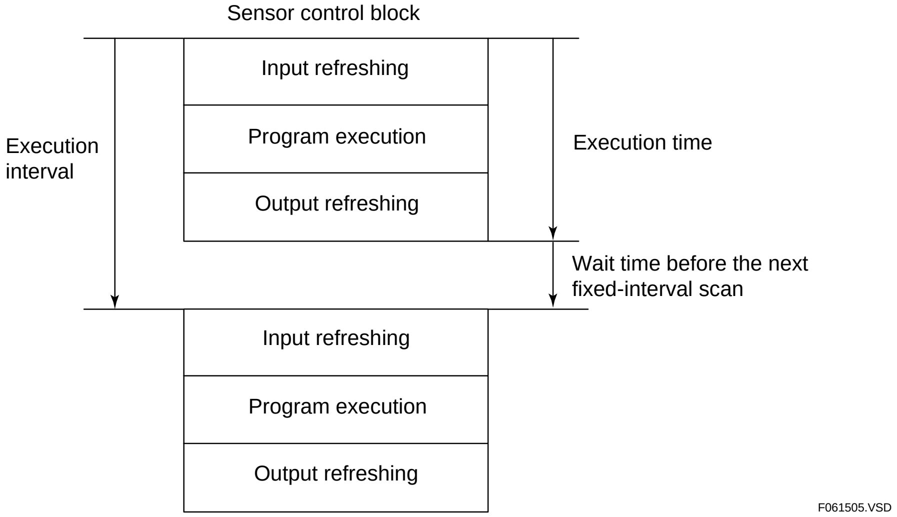


Figure A6.15.4 Schematic Diagram Showing Execution Time of Sensor Control Block

Table A6.15.12 Special Registers (Z) for Execution Time of Sensor Control Block

No.	Name	Description
Z109	Sensor CB Execution Time	Indicates the time taken from starting of input refreshing for the sensor control block through program execution to completion of output refreshing.
Z111	Maximum Sensor CB Execution Time	Indicates the maximum time taken to execute the sensor control block (Unit: 10 µs).

TIP

The measured execution time of the sensor control block is updated at each normal-scan input refreshing process.

A6.15.5 Procedures for Using Sensor Control Function

The table below summarizes the procedures for using the sensor control function.

Table A6.15.13 Procedures for Using Sensor Control Function

Procedure	Setup Item/Edit Item	WideField2 Function	Reference to Items Discussed Earlier
1	Defining I/O-refreshed words of sensor control block	WideField2 project configuration: DIO setup and interrupt setup	"■ Maximum Number of I/O-refreshed Words" of A6.15.3, "Specifications and Restrictions"
	Defining the execution interval		"■ Execution Interval Setting and Accuracy" of A6.15.4, "Functions"
	Defining the timing of interrupts		"■ Interrupt timing" of A6.15.4, "Functions"
	Defining the priority of interrupts		"■ Priority of Interrupts" of A6.15.4, "Functions"
	Creating normal-scan programs	WideField2 block editing	"■ Activating/Deactivating Sensor Control Block" of A6.15.4, "Functions"
	Creating sensor control programs		None (The procedure is the same as creation of normal programs.)
2	Registering sensor control block	WideField2 component definition	None (Register the created block under "SCB" in "Component Definition.")

TIP

You can use the sensor control block, irrespective of whether the configuration item “Program Execution Mode” is set to “All Blocks” or “Specified Blocks.”

A6.15.6 Error Handling

The table below summarizes the errors that may be reported when the sensor control function is in use.

Table A6.15.14 Errors Related to Sensor Control Function

Type of Error	Description
Sensor CB scan timeout error	The CPU fails to maintain the execution interval because it is exceeded by the sum of the fixed-interval I/O refreshing time and the execution time of sensor control programs.
Scan timeout	The CPU has insufficient time to execute a regular program because the execution time of sensor control programs is too long. Thus, the normal scan time exceeds the scan monitoring time.
I/O module error	An I/O module has failed during fixed-interval I/O refreshing in the sensor control block.

CAUTION

If multiple I/O module errors are detected during I/O refreshing in the sensor control block, the CPU only reports the first detected error by means of an alarm indicator.

Table A6.15.15 Sensor-CB-scan-timeout Special Relay (M)

Item	Self-diagnosis Status		
No.	Name	Description	Status
M212	Sensor CB Scan Timeout	ON: Error OFF: Normal	Indicates that the CPU cannot maintain the execution interval of the sensor control block.

Table A6.15.16 Actions When an Error Is Encountered in Sensor Control Block

Type of Error Encountered	Action in Case of Error	
	Sensor Control Block	Effect on Normal Scans
Sensor CB scan timeout (Special relay M212 turns on.)	Stop	The program either runs or stops depending on the "Error-time Action" defined for Sensor CB scan timeout error in "Operation Control" of configuration.
Other errors	For errors configurable with "Operation Control" of configuration, the sensor control block either runs or stops depending on the "Error-time Action" defined. The block stops if the error is not configurable.	

TIP

If an error is encountered in the sensor control block, the error block number stored in a special register (Z) is the last block number of a regular program plus one.

Table A6.15.17 Action of Sensor Control Block when an Error Is Detected in Normal Scan

Type of Error	Action
All types of error	For errors configurable with "Operation Control" of configuration, the sensor control block either runs or stops depending on the "Error-time Action" defined. The block stops if the error is not configurable.

A6.15.7 Programming Precautions

■ Instructions Not Applicable to Sensor Control Block

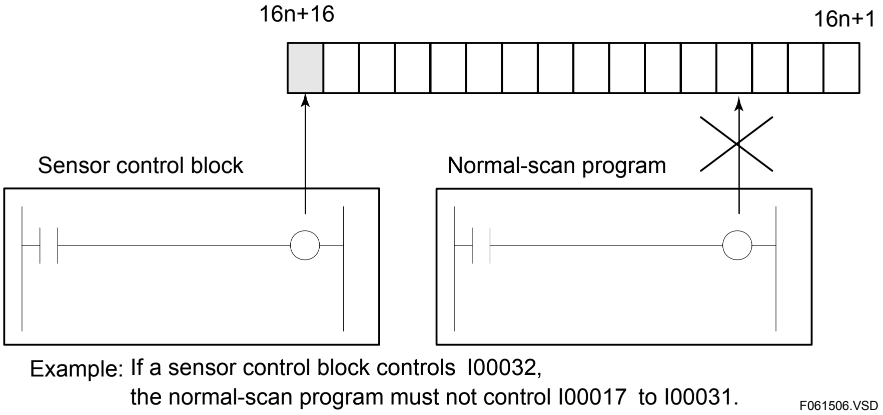
Table A6.15.18 Instructions Not Applicable to Sensor Control Block

Instruction	Corrective Actions
Timer (TIM) instruction	Enable/disable timers in a regular block. Reading timer relays in a sensor control block is allowed, however.
Special module High-speed Read (HRD)	Use the special module Read (READ) instruction instead.
Special module High-speed Write (HWR)	Use the special module Write (WRITE) instruction instead.

■ Precautions when the Interrupt Timing Is Immediate

● Precautions when outputting data to relays

If you have already output data to any of the relays numbered 1 to 16 using an OUT, SET RST, DIFU or DIFD instruction in the sensor control block, do not also output data to any of these relays in a normal-scan program. Otherwise, the output instruction may not be processed correctly. This precaution is also true with other groups of 16 relays numbered 17 to 32, 33 to 48, 49 to 64, and so on. Do not output data to relays within the same group both in the sensor control block and in a normal-scan program.



Example: If a sensor control block controls I00032,
the normal-scan program must not control I00017 to I00031.

Figure A6.15.5 Precautions for Relay Output

● Simultaneity of multi-device data

Simultaneity of data for multiple devices is not guaranteed.

For example, consider the case shown in Figure A6.15.3 where the sensor control block is executed during execution of a block transfer (BMOV) instruction in a normal scan. There is a risk that the source data that is partially transferred may be overwritten after the execution of the sensor control block or data transferred partially to the destination may be read by the sensor control block.

Simultaneity of data is required when data of multiple devices is exchanged between a normal-scan program and the sensor control block program using a block transfer (BMOV) instruction, a long-word instruction with IEEE single-precision floating point data, or two or more instructions.

If simultaneity of data is required when interrupt timing is configured as “Immediate (during instruction execution),” use any of the following means to ensure data simultaneity.

1. Use a Disable Sensor Control Block instruction (CBD) and an Enable Sensor Control Block (CBE) instruction to prevent the sensor control block from being executed during exchange of multi-device data.
2. Write an application program to perform flag control between the normal-scan program and the sensor control block using relays.

● Data simultaneity of devices to be refreshed

Simultaneity of data is not guaranteed if, in the sensor control block, an access is made to I/O relays (X/Y) refreshed in a regular block or to shared/extended shared relays (E), shared/extended shared registers (R), link relays (L), or link registers (W).

The sensor control block is executed even during normal-scan input refreshing, output refreshing and common processing. If you read any of the above-mentioned devices in the sensor control block, a device value being refreshed may be read. Likewise, if you write to the device, the device value being refreshed may be overwritten. Consequently, simultaneity of data may be lost.

To prevent the sensor control block from being executed during normal-scan input refreshing, output refreshing and common processing, execute a Disable Sensor Control Block (CBD) instruction to disable the block at the end of a regular program and execute an Enable Sensor Control Block (CBE) instruction to enable the block at the beginning of the program.

A6.16 Partial Download Function

The partial download function allows only specified blocks/macros to be downloaded to a CPU to replace corresponding blocks/macros of a program that has been downloaded earlier.
This reduces downloading time and improves debugging efficiency, especially in large- scale program development by a group of developers.
This function is available only in STOP mode. It allows multiple blocks or macro instructions to be specified for downloading. Addition or deletion of block/macro instructions by partial downloading is not allowed.

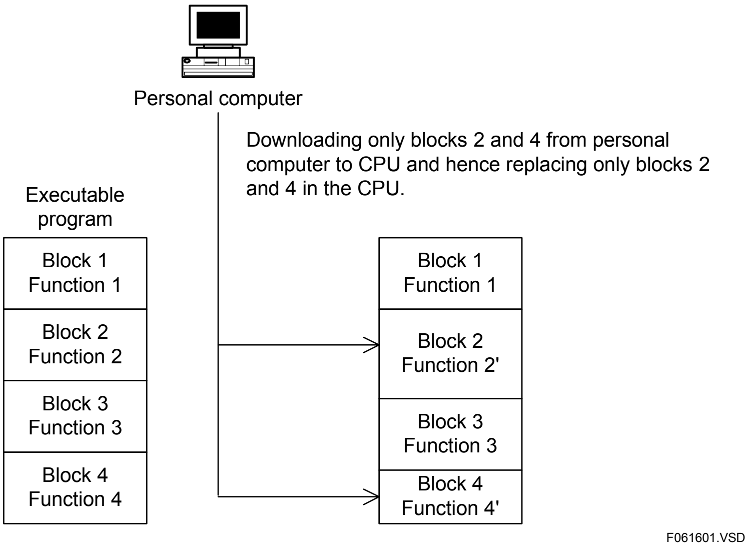


Figure A6.16.1 Partial Download Function

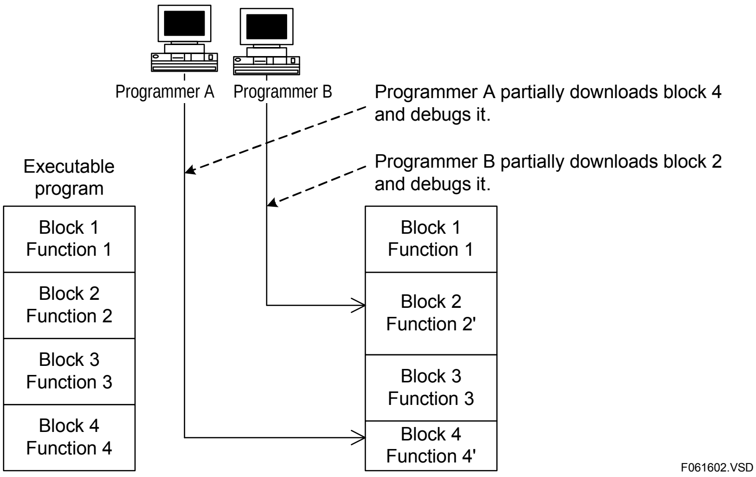


Figure A6.16.2 Partial Download Function Used by Multiple Programmers

CAUTION

- If an error occurs at the time of partial downloading, the step count of the error block becomes 0. If you then upload this defective program to a personal computer, the step count for the corresponding block on the personal computer will also be 0. If you have to upload such a program, save it under a different project name.
- At the completion of partial downloading, program checking and optimization are performed and this may take some time.

A6.17 Function for Storing Comments to CPU

You can store circuit comments and subcomments to the CPU module. Storing comments in the CPU module allows you to display them during circuit monitoring even if there is no project.

TIP

This function can only store circuit comments and subcomments. To store I/O comments, use the function for storing tag name definitions to CPU.

A6.17.1 Performing Setup to Download Comments

You can select whether to store (download) circuit comments and subcomments to a CPU module but you cannot select to download only circuit comments or only subcomments.

In WideField2, setup for downloading comments has to be performed in two places: in the block properties and when you execute the download function.

Firstly, specify to download comments in the block properties for each relevant block (macro instruction).

Then, turn on the download comments checkbox when you execute the download program function. Turning on this checkbox downloads comments to the CPU module according to the block properties.

Turning off the download comments checkbox when you execute the download program function will not download comments regardless of the block properties setup.

SEE ALSO

For details on block properties and program downloading, see "FA-M3 Programming Tool WideField2 User's Manual" (IM 34M6Q15-01E).

CAUTION

Note that if you turn off the checkbox for storing comments to the CPU when you execute the download program function, comments will not be downloaded to the CPU module regardless of the block properties setup.

A6.17.2 Number of Steps Needed for Comments

Like program steps, circuit comments/subcomments also takes up program area. Thus, how much of the program area is consumed in terms of step count also depends on whether comments are downloaded to a CPU module.

■ Calculating the Step Count of Comments

● If comments are downloaded:

The step count of a circuit comment or a circuit subcomment is the sum of the comment offset (1 step) and step count of the character string, as given below.

Step count of a comment = comment offset (1 step) +
step count of the character string

The step count of the character string is calculated as follows:

Sum the step counts of all characters in a character string, using 0.25 steps for each single-byte character and 0.5 steps for each double-byte character, and round up to the nearest integer.

TIP

Example: Assume that a comment is a character string consisting of four single-byte character and five double-byte characters.

Summing the step counts of individual characters yields:

4 × 0.25 (for single-byte characters) + 5 × 0.5 (for double-byte characters) = 3.5 (steps)

Rounding up to the nearest integer yields:

Step count of the character string = 4 (steps)

Adding 1 step for comment offset:

Step count of the comment = 4 + 1 = 5 (steps)

● If no comments are downloaded:

One step of program area is consumed for each comment (as comment offset).

CAUTION

One step of comment offset is added for each comment to the step count for a program even if the comments are not downloaded.

■ Checking the Program Step Count (including Comments)

The step count of a block (or macro instruction) containing comments is displayed on the status bar when the block is opened. The displayed step count includes the step counts of comments and tag names specified to be downloaded to the CPU module in the block properties window. If you select to only download the program, the displayed step count includes only the step count of the program.

A6.17.3 Online Editing of Comments

If circuit comments/subcomments are stored in a CPU module, you can edit or delete them online but you cannot add new comments online.

CAUTION

If you have added circuit comments or subcomments using offline program editing, you should download the circuit comments and subcomments to the CPU module again.

A6.18 Function for Storing Tag Name Definitions to CPU

This function stores common, block, and macro tag name definitions in the internal ROM of the sequence CPU module.
The sum of the program and tag name definition step counts must be within the "Maximum Size for Program Plus Tag Name Definitions" shown in the table below.
In addition, the program step count itself must be within the "Maximum Program Size" shown in the table.

Table A6.18.1 Program Capacity for Storing Tag Name Definitions

	Maximum Program Size	Maximum Size for Program plus Tag Name Definitions
F3SP66-4S	56K steps	112K steps
F3SP67-6S	120K steps	240K steps

For the step count of tag name definitions, check the project properties, the block tag name definition properties for each block, or the macro tag name definition properties for each macro instruction.
You can separately specify whether to download common, block, and macro tag name definitions using the project properties, the block tag name definition properties for each block, or the macro tag name definition properties for each macro instruction.
In addition, at the time you execute the download program function, you can choose to disable the downloading of tag name definitions, regardless of the properties setup.
If you specify not to download tag name definitions when you execute the download program function, any tag name definitions previously downloaded will be erased after the download.
If you edit tag name definitions online, the tag name definition files on the personal computer will be updated but not those in the program memory of the CPU module. If changes are made to the tag name definitions, download them to the CPU module again.

TIP

For programming efficiency, we recommend that you maintain the tag name definitions on the personal computer without storing them in the CPU module during debugging and program development, and download the tag name definitions to the CPU module after the programs are debugged.

A6.19 Structures

A structure represents a group of data under a unified name. It improves device representation and program readability.

The instructions related to structures are:

- **Structure Move (STMOV)**
- **Structure Pointer Declaration (STRCT)**
- **Structure Macro Instruction Call (SCALL)**

SEE ALSO

- For details on structures, see "FA-M3 Programming Tool WideField2 User's Manual" (IM34M6Q15-01E).
- For details on the instructions, see "Sequence CPU – Instructions User's Manual" (IM34M6P12-03E).

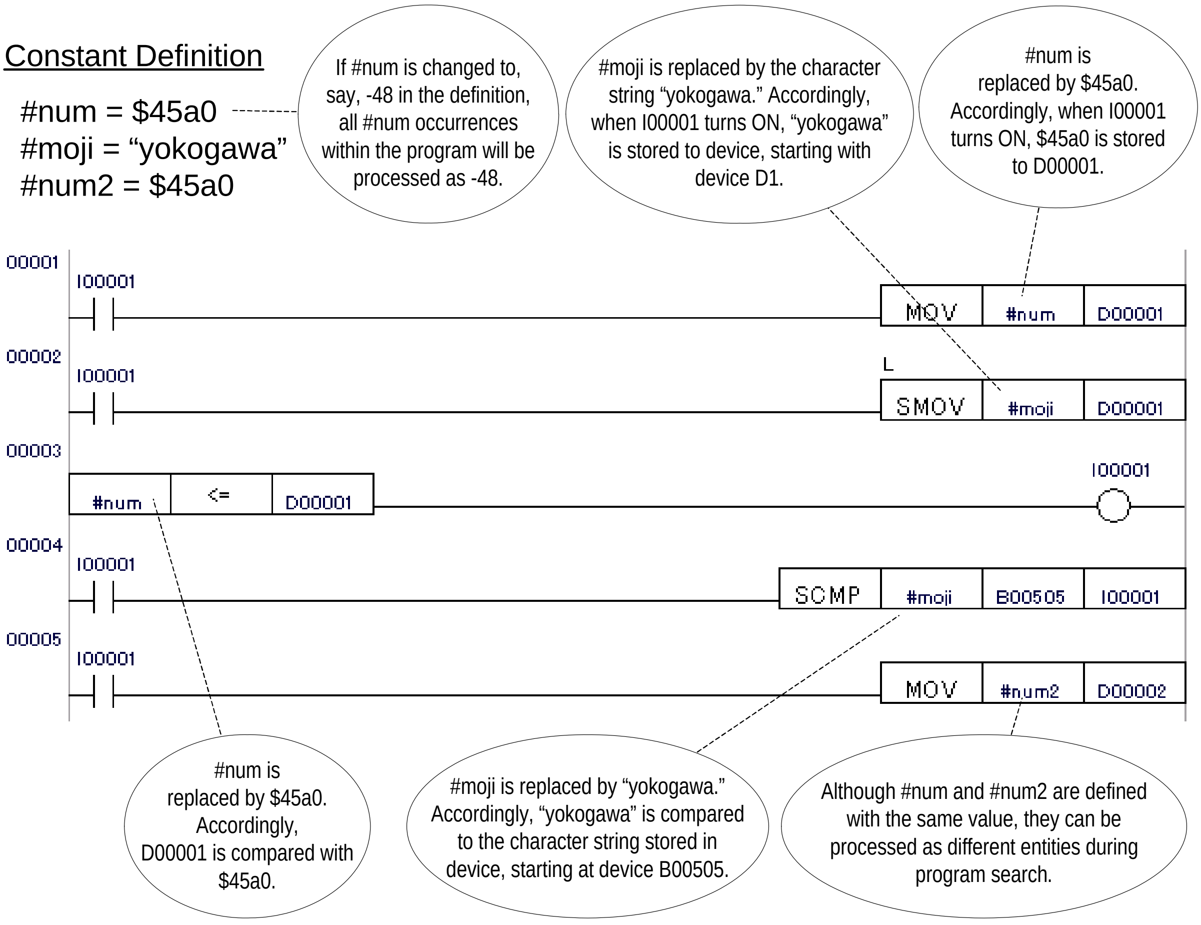
A6.20 Constant Definition Function (Header File)

This section describes the constant definition function.

A6.20.1 Benefits of Constant Definition Function

The constant definition function allows constants to be defined separately from programs, so that defined constant names can be coded in programs in place of constant values. The constant definition function is equivalent to the header file in the C programming language. The function offers the following benefits:

- Using constant names in place of numerical constant values in programs improves program readability and programming efficiency.
- When searching for a specific constant using its numerical value, you may find identical matches that have different meanings. For instance, the number "6" may represent a production volume in one instance and an instrument parameter in another instance. In comparison, searching for a constant using its constant name avoids this problem and is thus much simpler.
- With the use of constant names, changing the value of a specific constant is much simpler as it requires changing only one occurrence in the constant definition.



Constant Definition

#num = $45a0
#moji = “yokogawa”
#num2 = $45a0

If #num is changed to, say, -48 in the definition, all #num occurrences within the program will be processed as -48.

#moji is replaced by the character string “yokogawa.” Accordingly, when I00001 turns ON, “yokogawa” is stored to device, starting with device D1.

#num is replaced by $45a0. Accordingly, when I00001 turns ON, $45a0 is stored to D00001.

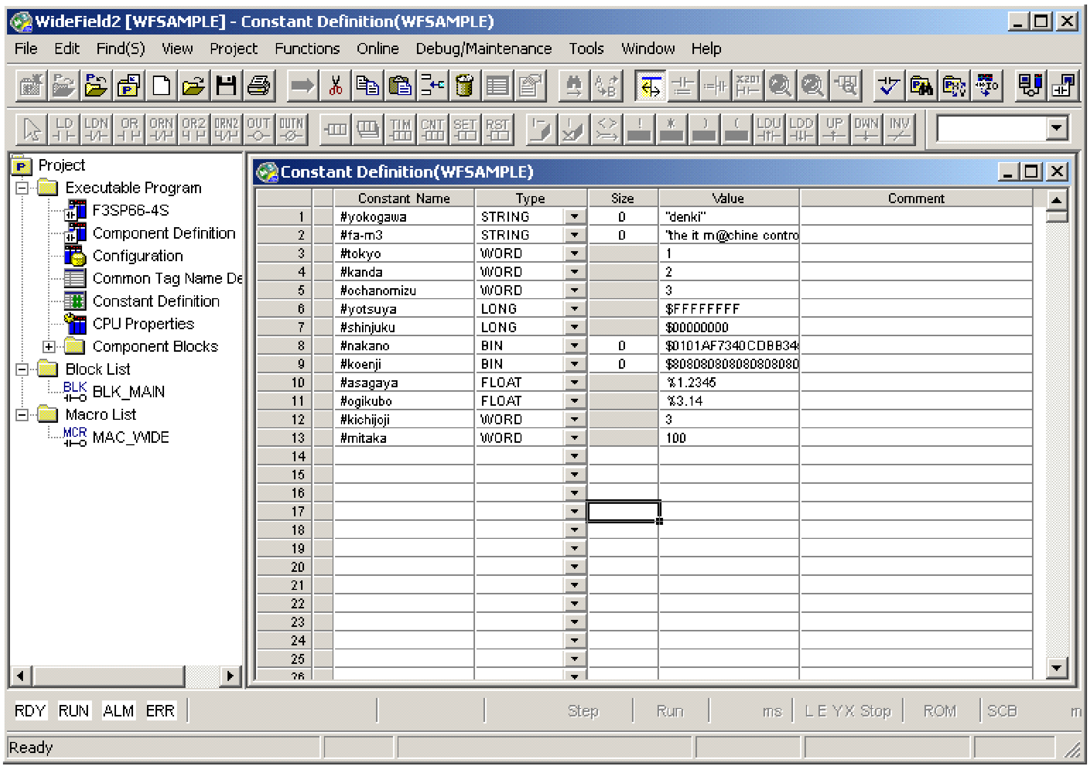
#num is replaced by $45a0. Accordingly, D00001 is compared with $45a0.

#moji is replaced by “yokogawa.” Accordingly, “yokogawa” is compared to the character string stored in device, starting at device B00505.

Although #num and #num2 are defined with the same value, they can be processed as different entities during program search.

FA0652.VSD

Figure A6.20.1 Using Constant Definitions in Programs


WideField2 [WFSAMPLE] - Constant Definition(WFSAMPLE)
File Edit Find(S) View Project Functions Online Debug/Maintenance Tools Window Help
LD LDN OR ORN OR2 ORN2 OUT OUTN TIM CNT SET RST LDU LDD UP DWN INV
Project
Executable Program
F3SP66-4S
Component Definition
Configuration
Common Tag Name De
Constant Definition
CPU Properties
Component Blocks
Block List
BLK_MAIN
Macro List
MAC_WIDE
RDY RUN ALM ERR
Step Run ms L E YX Stop ROM SCB
Ready

Constant Definition(WFSAMPLE)

		Constant Name	Type	Size	Value	Comment
1		#yokogawa	STRING	0	"denki"	
2		#fa-m3	STRING	0	"the it m@chine contro	
3		#tokyo	WORD		1	
4		#kanda	WORD		2	
5		#ochanomizu	WORD		3	
6		#yotsuya	LONG		$FFFFFFFF	
7		#shinjuku	LONG		$00000000	
8		#nakano	BIN	0	$0101AF7340CDBB34	
9		#koenji	BIN	0	$8080808080808080	
10		#asagaya	FLOAT		%1.2345	
11		#ogikubo	FLOAT		%3.14	
12		#kichijoji	WORD		3	
13		#mitaka	WORD		100	
14						
15						
16						
17						
18						
19						
20						
21						
22						
23						
24						
25						

FA0653.VSD

Figure A6.20.2 Constant Definition Window

CAUTION

To enter a backslash (\) in a constant definition, you need to enter two backslashes (\\) for compatibility with M3 escape sequence. For instance, to enter the file pathname "\CARD1\ABC", enter "\\CARD1\\ABC" instead.

SEE ALSO

For details on the M3 escape sequence function, see Subsection A6.21.1, "M3 Escape Sequence Function."

A6.20.2 Constant Definition Function Specifications

This subsection describes the specifications of the constant definition function.

■ Constant Definition Specifications

The table below shows the specifications of the constant definition function.

Table A6.20.1 Constant Definition Specifications

Item	F3SP66-4S	F3SP67-6S
Size of constant area *1	16384 (words)	32768 (words)
Constant name length	16 max. (including the constant name identifier '#')	
Online modification	Not allowed	

*1: This is the size of the system area for storing constants. If all defined constants are word constants, the maximum number of constants that can be defined is the same as the size indicated here. For details on the consumption of the constant definition area by individual constant data types, see "■ Maximum Number of Constant Names."

■ Supported Constant Data Types

● Word constants

Word constants are word sized constants.
Their values range from -32768 to 32767.

● Long word constants

Long word constants are long word sized constants.
Their values range from -2147483648 to 2147483647.

● Binary constants

Binary constants are multiple bytes long. Up to 256 bytes of contiguous data can be defined as a binary constant.

● String constants

Up to 255 characters can be defined as a string constant. A NULL byte is appended automatically to the end of the string.

M3 escape sequence can be included in string constants.

● Floating-point constants in IEEE single precision floating-point format

These are long-word-sized constants. Their values range from 1E-38 to 1E+38, and must be prefixed with the '%' character.
(Example: %1.234560E-012)

SEE ALSO

For details on the M3 escape sequence function, see Subsection A6.21.1, "M3 Escape Sequence Function."

■ Maximum Number of Constant Names

The maximum number of constant names that can be defined varies, depending on the data types of individual defined constant names. This is because the consumption of the constant definition area by a defined constant varies with its data type. The table below lists constant data types along with their consumption of the constant definition area. Once the total consumption of the constant definition area by all constant names in use reaches the size of the constant definition area, no more constant names can be defined.

Table A6.20.2 Constant Data Type and Consumption of Constant Definition Area

Data Type	Consumption of Constant Area (words)
Word	1
Long word	2
Binary	Size of defined data ÷2 *1
String	Length of defined string÷2+0.5 *1
Floating-point	2

*1: Fractional values 0.5 and above are to be rounded to 1.
Note: Besides consuming the constant definition area, constants also consume the program steps.

■ Types of Constant Definitions

The constant definition function supports common constant definition. In other words, defined constants are visible to all blocks of a project. Definition of local constants on block basis is not supported.

● Common constant definition

Defines constants that can be used by all blocks of a project.

● Local constant definition

Definition of local constants is not supported.

■ Special Relays and Special Registers

There are no special relays and special registers related to constant definition.

A6.20.3 Constant Definition Function Setup

The constant definition function requires no setup before use.

A6.20.4 Using Constant Definitions

This subsection briefly describes how to use the constant definition function.

SEE ALSO

For details on how to use the constant definition function, see "FA-M3 Programming Tool WideField2 User's Manual" (IM 34M6Q15-01E).

■ Editing Constant Definitions

Select [Project]–[Constant Definition] from the menu bar of WideField2 to open the Constant Definition Window. You can edit constant definitions displayed in the Constant Definition Window.

You can also find, replace, print, sort and perform other operations in the Constant Definition Window.

■ Downloading (Loading) Constant Definitions

Constant definitions are included as part of a project, and are therefore stored to the module automatically when you download or load a project.

■ Uploading (Saving) Constant Definitions

Constant definitions are included as part of a project, and are therefore retrieved from the module automatically when you upload or save a project.

■ Comparing Constant Definitions

Select [Online]–[Compare File and CPU]–[Project] or [Online]–[Compare File and CPU]–[Block/Macro] from the menu bar of WideField2. Constant definitions are compared along with project data or block data.

A6.21 Telegram Processing Functions

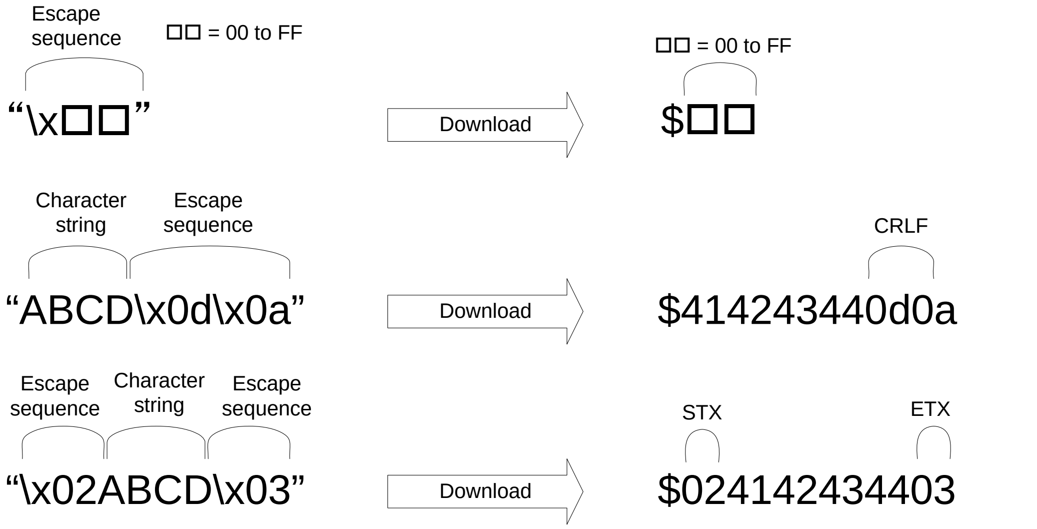
This section describes the telegram processing function.
Telegram processing function group supports the creation of messages (telegrams) in communication processing and file processing, thus increasing programming efficiency.

A6.21.1 M3 Escape Sequence

This subsection describes the M3 escape sequence function.

■ Merits of M3 Escape Sequence

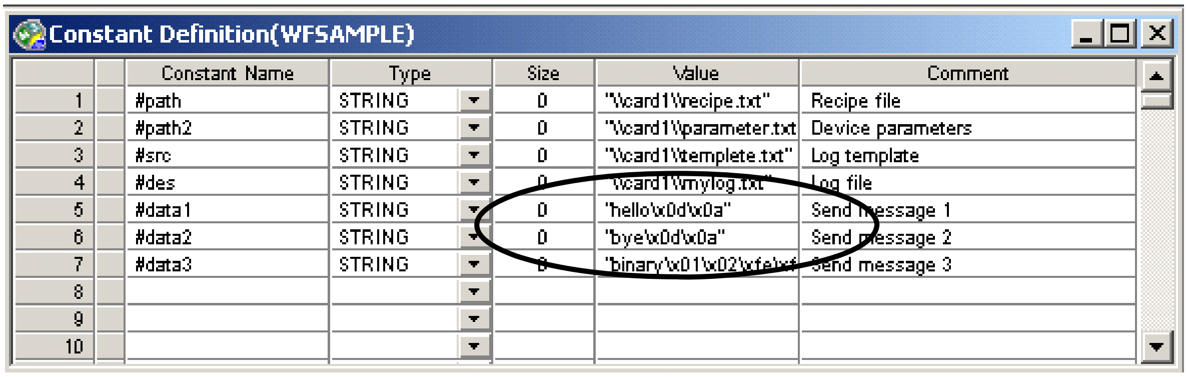
An escape sequence is a binary representation of a character string. When characters coded in a defined format (escape sequence) is included within a character string, WideField2 replaces the escape sequence with its binary data before downloading.



FA0654.VSD

Figure A6.21.1 Downloading Escape Sequence (Converting Escape Sequence to Binary Data)

Conversely, when reading a character string containing binary data, WideField2 replaces the binary data with an escape sequence character string before display.


Constant Definition(WFSAMPLE)

		Constant Name	Type	Size	Value	Comment
1		#path	STRING	0	"\\card1\\recipe.txt"	Recipe file
2		#path2	STRING	0	"\\card1\\parameter.txt	Device parameters
3		#src	STRING	0	"\\card1\\templete.txt"	Log template
4		#des	STRING	0	"\\card1\\mylog.txt"	Log file
5		#data1	STRING	0	"hello\x0d\x0a"	Send message 1
6		#data2	STRING	0	"bye\x0d\x0a"	Send message 2
7		#data3	STRING	0	"binary\x01\x02\xfe\xf	Send message 3
8						
9						
10						

FA0655.VSD

Figure A6.21.2 Replacing Binary Data by Character String

The following are some benefits of using M3 escape sequence.

- M3 escape sequence can be used to define control characters (ETX, STX, etc.) along with transmission text when creating a telegram. In the past, when the M3 escape sequence function was not available, transmission text and control characters had to be created separately and combined using application instructions.
- M3 escape sequence can be used to easily combine text with tab characters and line feed characters (CRLF, LF, etc.) when creating a CSV formatted file.

■ M3 Escape Sequence Specifications

The specifications of M3 escape sequence is described here.

● List of M3 escape sequences

The table below lists M3 escape sequences.

Table A6.21.1 List of M3 Escape Sequences

M3 Escape Sequence	Corresponding Binary Value (in hexadecimal)
\x00 – \x0F	$00 – $0F
\x10 – \x1F	$10 – $1F
\x20 – \x2F	$20 – $2F
\x30 – \x3F	$30 – $3F
\x40 – \x4F	$40 – $4F
\x50 – \x5F	$50 – $5F
\x60 – \x6F	$60 – $6F
\x70 – \x7F	$70 – $7F
\x80 – \x8F	$80 – $8F
\x90 – \x9F	$90 – $9F
\xA0 – \xAF	$A0 – $AF
\xB0 – \xBF	$B0 – $BF
\xC0 – \xCF	$C0 – $CF
\xD0 – \xDF	$D0 – $DF
\xE0 – \xEF	$E0 – $EF
\xF0 – \xFF	$F0 – $FF

● Range of values that can be represented using M3 escape sequence

M3 escape sequence can be used to represent hexadecimal values from $00 to $FF.

● Syntax of M3 escape sequence

Specify a hexadecimal value between "00" and "FF" prefixed by the "\x" character string.

As an example, specify "\xD0" for $D0.

M3 escape sequences are case-insensitive.

● Representing backslash (\) character

To represent a backslash (\) character, code it as two backslash characters (\\) (the first backslash character acts as the escape character).

● Scope of M3 escape sequence

The table below lists the applicable scope of M3 escape sequence.

Table A6.21.2 Applicable Scope of M3 Escape Sequence

Category	Function	Operation
Input/edit	Constant definition	Defining and editing character string constants
	Block/macro edit	Entering or editing character string constants in instruction parameters
Display	Constant definition	Display of character string constants
	Block/macro edit	Display of character string constants in TIP help.
	Circuit monitor	Display of character string constants in TIP help.

● Special relays and special registers

There are no special relays and special registers related to the M3 escape sequence function.

■ M3 Escape Sequence Setup

The M3 escape sequence function requires no setup before use.

■ Using M3 Escape Sequence

● Entering and editing M3 escape sequence

- Entering Escape Sequence in Constant Definition

  Select [Project]–[Constant Definition] from the menu bar of WideField2 to open the constant definition window. M3 escape sequence can be entered directly into the constant definition window when defining a character string constant.

- Entering Escape Sequence in Block/Macro Edit

  Open a block or macro. M3 escape sequence can be entered directly as a character string constant for an instruction parameter.

● Display of M3 escape sequence

- Display of M3 Escape Sequence in Constant Definition

  Select [Project]–[Constant Definition] from the menu bar of WideField2 to open the constant definition window. If binary data is included in a defined character string constant, it is displayed as an escape sequence.

- Display of M3 Escape Sequence in Block/Macro Edit Window and Circuit Monitor

  Moving the mouse cursor over a constant name displays its defined value as TIP help. If binary data is included in a defined character string constant, it is displayed as an escape sequence.

A6.22 Log Function

The section describes the log function.
The module supports the following types of log data.

Table A6.22.1 Summary of Log Function

Log Type	Description
System log (error log)	Records module startup, power off, error and other events.
FTP server log	Records accesses to the FTP server.
User log	Records user-defined events.

A6.22.1 System Log (Error Log) Function

The system log records module startup, power off, error and other events.

SEE ALSO

For details on the system log (error log), see Section A8.1, "Self Diagnosis" of this manual, as well as the "FA-M3 Programming Tool WideField2 User's Manual" (IM34M6Q15-01E).

A6.22.2 FTP Server Log Function

The FTP server log records accesses to the FTP server and commands processed by the FTP server.

SEE ALSO

For details on the FTP server log, see Subsection 3.6.4, "FTP Server Log" of "Sequence CPU – Network Functions User's Manual (for F3SP66-4S, F3SP67-6S)" (IM34M6P14-02E)

A6.22.3 User Log Function

The user log function records user-defined events when dedicated instructions are executed by a program.

SEE ALSO

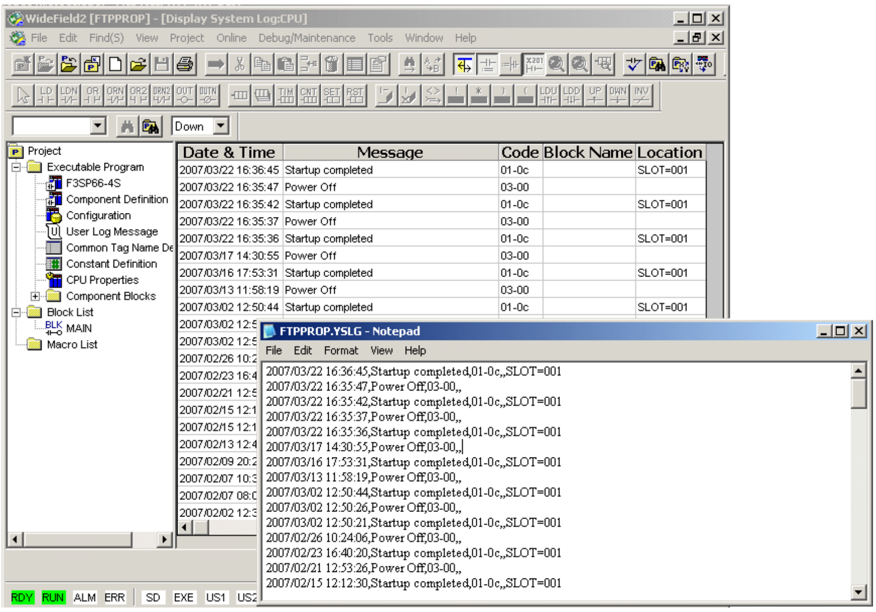
For details on the user log function, see Section A6.14, "User Log Management Function."

A6.22.4 Log to Text Conversion Function

The log to text conversion function converts log data to a text file. In this way, log data can be viewed without using WideField2. The conversion is done using the smart access functions.

The following types of log data can be converted:

- system log
- FTP server log



Date & Time	Message	Code	Block Name	Location
2007/03/22 16:36:45	Startup completed	01-0c		SLOT=001
2007/03/22 16:35:47	Power Off	03-00		
2007/03/22 16:35:42	Startup completed	01-0c		SLOT=001
2007/03/22 16:35:37	Power Off	03-00		
2007/03/22 16:35:36	Startup completed	01-0c		SLOT=001
2007/03/17 14:30:55	Power Off	03-00		
2007/03/16 17:53:31	Startup completed	01-0c		SLOT=001
2007/03/13 11:58:19	Power Off	03-00		
2007/03/02 12:50:44	Startup completed	01-0c		SLOT=001
2007/03/02 12:5				
2007/03/02 12:5				
2007/02/26 10:2				
2007/02/23 16:4				
2007/02/21 12:5				
2007/02/15 12:1				
2007/02/15 12:1				
2007/02/13 12:4				
2007/02/09 20:2				
2007/02/07 10:3				
2007/02/07 08:0				
2007/02/02 12:3				

```
2007/03/22 16:36:45,Startup completed,01-0c,,SLOT=001
2007/03/22 16:35:47,Power Off,03-00,,
2007/03/22 16:35:42,Startup completed,01-0c,,SLOT=001
2007/03/22 16:35:37,Power Off,03-00,,
2007/03/22 16:35:36,Startup completed,01-0c,,SLOT=001
2007/03/17 14:30:55,Power Off,03-00,,
2007/03/16 17:53:31,Startup completed,01-0c,,SLOT=001
2007/03/13 11:58:19,Power Off,03-00,,
2007/03/02 12:50:44,Startup completed,01-0c,,SLOT=001
2007/03/02 12:50:26,Power Off,03-00,,
2007/03/02 12:50:21,Startup completed,01-0c,,SLOT=001
2007/02/26 10:24:06,Power Off,03-00,,
2007/02/23 16:40:20,Startup completed,01-0c,,SLOT=001
2007/02/21 12:53:26,Power Off,03-00,,
2007/02/15 12:12:30,Startup completed,01-0c,,SLOT=001
```

FA0656.VSD

Figure A6.22.1 Log Text Conversion Function

■ Text Conversion Using Smart Access Functions

All log data retrieved using smart access functions are returned as text files. You can get log data using the following smart access functions:

- Card batch file function
- Rotary switch function

SEE ALSO

For details on smart access functions, see Part B, "Smart Access Functions."

A6.23 Security Functions

Security functions help protect user program assets against unauthorized external access and incorrect user operation.

Table A6.23.1 List of Security Functions

Security Functions	Overview	SEE ALSO
Executable Program Protection	Protects an entire project using a password.	A6.23.1, "Executable Program Protection"
Block Protection	Protects a block using a password.	A6.23.2, "Block Protection"
CPU Properties Protection	Protects CPU property data using a keyword.	A6.23.3, "CPU Properties Protection"
Network Filter Function	Restricts network nodes that are allowed to connect to the module.	A6.23.4, "Network Filter Function"
Function Removal	Removes selected functions of the module.	A6.23.5, "Function Removal"

CAUTION

Security functions do not guarantee 100% protection of user program assets against unauthorized external access and incorrect user operation. They are provided to support users' security operations.

A6.23.1 Executable Program Protection

Executable program protection protects an entire executable program.

When executable program protection is enabled, downloading, uploading and other operations on the executable program can be executed only if a valid password is entered.

When executable program protection is enabled, the following operations are restricted:

- downloading
- uploading
- monitoring (ciruit diagram monitoring, debugging operations, changing preset values of timers (T) and counters (C))
- online editing
- printing

SEE ALSO

For details on how to enable executable program protection, see Subsection B7.1.2, "Changing CPU Type And Executable Program Properties" of "FA-M3 Programming Tool WideField2 User's Manual" (IM34M6Q15-01E).

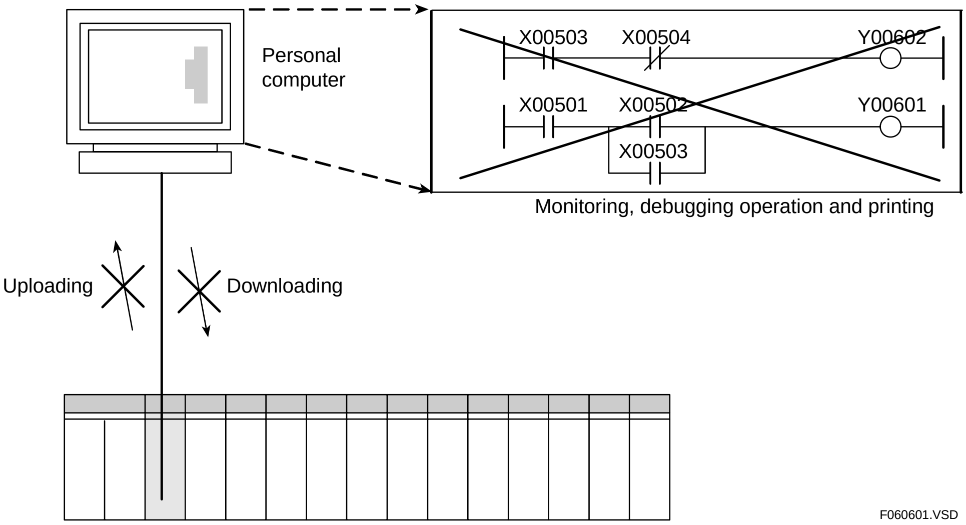


Figure A6.23.1 Executable Program Protection

A6.23.2 Block Protection

Block protection protects programs on per block basis.

When block protection is enabled for a block, its circuit diagrams and instructions are not displayed in WideField2. To display such information, a valid password must be entered.

When block protection is enabled for a block, the following operations on the block are prohibited:

- monitoring (ciruit diagram monitoring, debugging operations, changing preset values of timers (T) and counters (C))
- online editing
- printing

SEE ALSO

For details on how to enable block protection, see Section B3.4, "Editing Local Devices and Properties of Blocks and Macros" of "FA-M3 Programming Tool WideField2 User's Manual" (IM34M6Q15-01E).

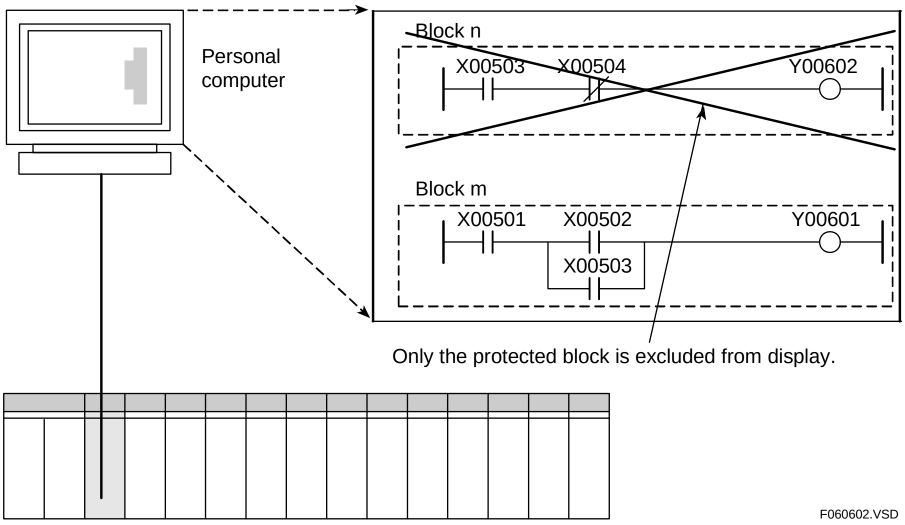


Figure A6.23.2 Block Protection

A6.23.3 CPU Properties Protection

This subsection describes the security function for protecting CPU properties.

■ Overview of CPU Properties Protection

CPU properties protection protects CPU property data stored in the module against read and write access using a keyword. You can set (enable) or remove (disable) this protection using WideField2 or smart access functions. When this protection is enabled, the following restrictions are imposed on operations.

Table A6.23.2 Restrictions Imposed by CPU Properties Protection

Operation on CPU Properties	Restriction
Loading, downloading, writing program code	A keyword is required.
Saving, uploading and reading program code	A keyword is required.
Clearing CPU properties	A keyword is required.

■ Specification of CPU Properties Protection

The table below shows the specification of the CPU properties protection function.

Table A6.23.3 Specifications of CPU Properties Protection

Item	Specifications
Protection method	Keyword validation
Keyword length	0 to 8 characters
Valid characters for keyword	ASCII characters ’0’ to ’9’, ’a’ to ’z’ and ’A’ to ’Z’

● Special relays and special registers

There are no special relays and special registers related to CPU properties protection.

■ CPU Properties Protection Setup

CPU properties protection requires no special setup before use.

■ Setting (Enabling) Protection

● Setting protection using WideField2

In WideField2, select [Online]–[Extended Functions]–[Protection of CPU Properties]–[Set] from the menu bar, and specify a security keyword.

When CPU properties protection is set from WideField2, the PROTECT part of the CPU property file is not used.

● Setting protection using smart access functions

In the PROTECT part of the CPU property file, specify 1 for the PROTECT_ENABLE property, and specify a keyword. When the CPU property file is loaded to the module, protection is enabled for the stored CPU property data.

Table A6.23.4 CPU Property File Setup (for setting protection)

CPU Property Setup Name	CPU Property Item	Property Value for Setting Protection
[PROTECT]	PROTECT_ENABLE	1
	PROTECT_KEYWORD	Up to 8 ASCII characters

CAUTION

Apply stringent control to any CPU property file containing a security keyword to avoid inadvertent disclosure as the CPU property file is a readable text file. To avoid such security risks, use WideField2 to set and remove protection of CPU properties instead.

■ Removing (Disabling) Protection

● Removing protection using WideField2

In WideField2, select [Online]–[Extended Functions]–[Protection of CPU Properties]–[Remove] from the menu bar to remove the protection.

When CPU properties protection is enabled from WideField2, the PROTECT part of the CPU property file is not used. It is automatically overwritten by WideField2.

● Removing protection using smart access functions

In the PROTECT part of the CPU property file, specify 0 for the PROTECT_ENABLE property, and specify the keyword used to set protection earlier. When the CPU property file is loaded to the module, CPU properties protection is disabled.

Table A6.23.5 CPU Property File Setup (for removing protection)

CPU Property Setup Name	CPU Property Item	Property Value for Removing Protection
[PROTECT]	PROTECT_ENABLE	0
	PROTECT_KEYWORD	Keyword used to set protection previously

■ Modifying Protected CPU Properties

● Using WideField2

When you attempt to upload or download CPU properties after connecting to the module with CPU properties protection enabled, an input dialog is displayed prompting for a keyword. Entering the keyword used to set protection previously allows processing to proceed.

● Using smart access functions

In the PROTECT part of the CPU property file, specify 1 for the PROTECT_ENABLE property, and specify the keyword used to set protection previously. When the CPU property file is loaded to the module, CPU property data in the module is updated while CPU property protection remains enabled.

Beware of disabling CPU property protection inadvertently by loading a CPU property file with 0 specified for the PROTECT_ENABLE property.

Table A6.23.6 CPU Property File Setup (for setting protection)

CPU Property Setup Name	CPU Property Item	Property Value for Updating CPU Properties When Protection is Enabled
[PROTECT]	PROTECT_ENABLE	1
	PROTECT_KEYWORD	Keyword used to set protection previously

■ Modifying Security Keyword

● Using WideField2

In WideField2, select [Online]–[Extended Functions]–[Protection of CPU Properties]–[Set] from the menu bar to change the security keyword. You need to enter both the old keyword and the new keyword.

When CPU properties protection is set from WideField2, the PROTECT part of the CPU property file is not used. It is automatically overwritten by WideField2.

● Using smart access functions

Disable protection and then enable protection again.

A6.23.4 Network Filter Function

The network filter function restricts IP addresses that are allowed to establish connections to the module.

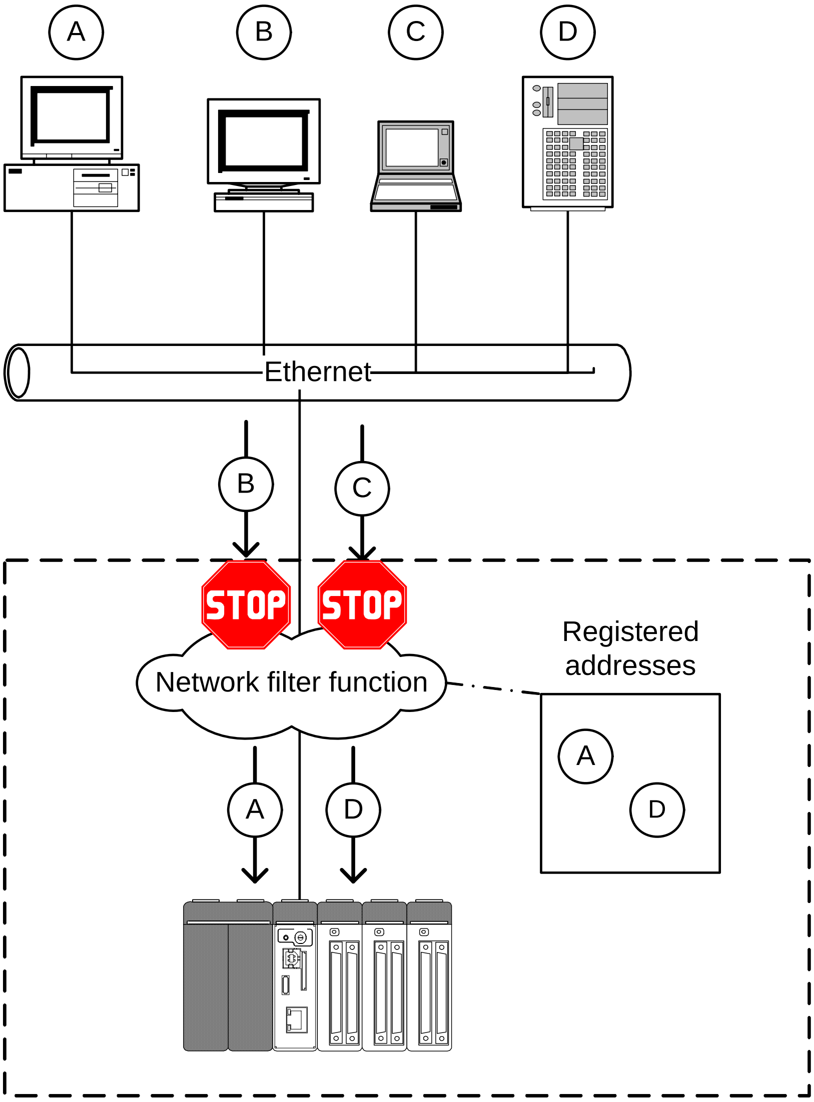
■ Overview of Network Filter Function

The subsection briefly describes the network filter function.

● Merits of Network Filter Function

- Protecting against malicious connections

The network filter function protects equipment against malicious connections by restricting IP addresses that are allowed to establish connections to the module. In particular, it is safer to use the network filter when the module is connected to an external network such as the Internet.



FA0659.VSD

Figure A6.23.3 Operation of Network Filter

Specification of Network Filter Function

The specification of the network filter function is described here.

Applicable network functions

The network filter function restricts IP addresses that are allowed to establish connections to the module using the functions listed in the table below.

Table A6.23.7 Module Functions Supporting Network Filter

Interface Name
Remote programming service (WideField2 connection)
Higher-level link service
Socket communications
FTP server
FTP client

Special relays and special registers

- Special relays

  There are no special relays related to the network filter function.

- Special registers

Table A6.23.8 Special Registers (related to Network Filter function)

Category	Sequence CPU Module Status Registers		
No.	Name	Function	Description
Z113	Number of Invalid Accesses	Counts the number of invalid accesses	Counts the number of connection requests received from IP addresses that are not registered with the network filter function. The counter is incremented from 0 to 65535, and resets to zero when it exceeds 65535. To reset the counter value, write a zero value to the register, or restore the module to factory settings.

Network Filter Function Setup

How to configure the network filter before use is described here.

Basic Setup

The table below shows required setup for the network filter before use.

Table A6.23.9 Basic Setup for Network Filter

Name of Setup	Type of Setup	SEE ALSO
Network filter setup	CPU properties	A9.5.8, "Network Filter Setup"

- Network filter setup

  Register IP addresses that are allowed to establish connections to the module. By default, connections from all IP addresses are allowed. This setup affects all functions (e.g. remote programming service, FTP server, etc.) supporting network filtering.

Optional Setup

The network filter function has no optional setup.

■ Using Network Filter Function

How to use the network filter function is described below.

● Preparing the network filter function

In the network filter setup of CPU properties, register IP addresses and/or hostnames that are allowed to establish connection to the module. The network filter function is enabled so long as at least one IP address or hostname is registered. Once enabled, the network filter function rejects connection requests from all unregistered IP addresses or hostnames.

A subnet mask can also be used to grant access to a group of IP addresses based on a specific network address. For instance, specifying IP address 192.168.5.1 and subnet mask 255.255.255.0 as a pair grants access permissions to IP addresses 192.168.0.xxx (where xxx = 0 to 255). Subnet masks can also be used with hostnames.

Registering multiple IP addresses or hostnames creates OR conditions for granting access permissions. For instance, if both 192.168.0.1 and 192.168.0.3 are registered with subnet mask 255.255.255.255, both IP addresses 192.168.0.1 and 192.168.0.3 are granted permission to connect.

If hostnames are specified in network filter setup, the module converts these hostnames to IP addresses at startup to be used for subsequent filtering. Therefore, if the mapping between hostnames and IP addresses on the DNS is changed after the module is started, such changes are not known to the module and filtering will be incorrect. To solve the problem, either restart the module (by power off and on or by reset) or update the network filter setup using the RENEW part of CPU properties.

CAUTION

- If the network filter function fails to locate the DNS server when it tries to convert hostnames to IP addresses, communications may be disabled for several to tens of seconds.
- If the network filter function fails to resolve a hostname, it ignores the hostname during filter processing.
- The use of network filtering may degrade commnication performance. The more registered IP addresses and hostnames, the greater the impact on performance. Moreover, the impact is greater for UDP communications as filtering is performed on each receiving.

● Using the network filter function

The network filter function is automatically started at module startup.

A6.23.5 Function Removal

Function removal allows you to disable selected functions of the module.

SEE ALSO

If you remove all functions, you will no longer be able to access the module (except via an external module such as an Ethernet interface module). To recover from such a state, you need to restore the module to its factory settings.

For details on how to restore the module to its factory settings, see Subsection B1.5.4, "Restore Factory Settings."

■ Overview of Function Removal

Function removal disables selected CPU module functions. It can be used to remove functions not to be used by end users before equipment delivery. It is also useful for reducing risks of mishandling in a situation where equipment may be operated by untrained personnel.

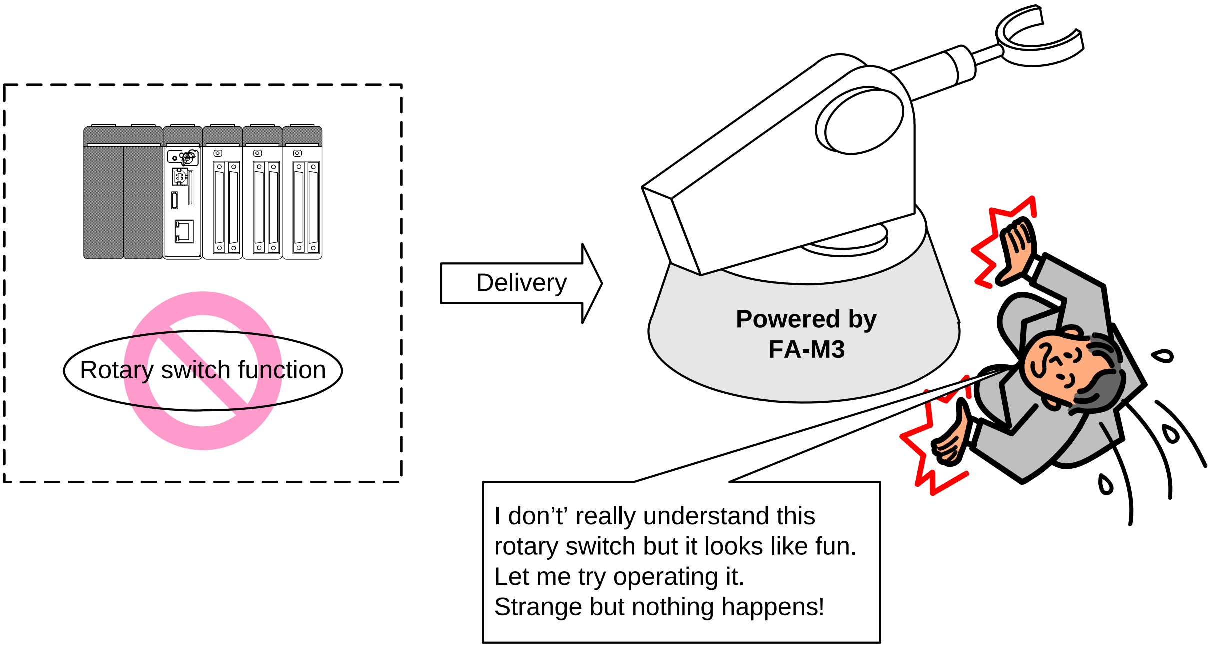

Rotary switch function
Delivery
Powered by FA-M3
I don’t’ really understand this rotary switch but it looks like fun. Let me try operating it. Strange but nothing happens!
FA0660.VSD

Figure A6.23.4 Function Removal (an example of rotary switch function removal)

■ Specification of Function Removal

The specification of function removal is described here.

● Function removal

The table below lists the functions that can be removed along with the outcome and limitations relating to their removal.

Table A6.23.10 List of Functions that Can be Removed

Function that can be Removed	Outcome and Limitations of Removal
Remote programming service	The CPU built-in remote programming service via the USB port and the 10BASE-T/100BASE-TX connector are removed. Access via an external module such as an Ethernet interface module is not affected.
Higher-level link service	The CPU built-in higher-level link service via the SIO port and the 10BASE-T/100BASE-TX connector are removed. Access via an external module such as a personal computer link module is not affected.
FTP server	If the FTP server function is removed, virtual directory function is also automatically removed.
Rotary switch function	All rotary switch functions except boot modes are removed.
Virtual directory function	All virtual directory commands are removed.
Card batch file function	All card batch file functions are removed.

● Special relays and special registers

There are no special relays and special registers related to function removal.

■ Function Removal Setup

How to configure function removal before use is described here.

● Basic Setup

The table below shows required setup for function removal before use.

Table A6.23.11 Basic Setup for Function Removal

Name of Setup	Type of Setup	SEE ALSO
Function removal setup	Configuration	A9.2.12, "Function Removal"

- Function removal setup

  Specify the functions to be removed (disabled). By default, all functions are enabled.

● Optional Setup

Function removal has no optional setup.

■ Using Function Removal

Specify the functions to be removed (disabled) using function removal setup of configuration. By default, all functions are enabled.

A7. I/O Response Time Based on Scan Time

This chapter discusses examples of calculating the scan time and I/O response time. It also explains such parameters as instruction execution time.

A7.1 Description of Scan Time

The sequence CPU module is designed so that two systems of processes, i.e., a system of control-related processes and a system of peripheral processes, run concurrently and independently. For this reason, the system of control-related processes whose main purpose is to execute programs and control-related processes is not affected by the system of peripheral processes whose purpose is to support communication and WideField2. Thus, the system of control-related processes can run at extremely high speeds. Under normal conditions, the scan time of the sequence CPU module is equivalent to the time taken by the system of control-related processes.

SEE ALSO

For details on the system of processes, see Section A3.4, "Computation Method."

The following paragraphs explain the processing tasks and time of each of these systems.

● System of Control-related Processes

The latest, minimum and maximum of scan times taken by the system of control-related processes are stored in special registers Z001 to Z003 in that order.

Table A7.1.1 Scan Time of System of Control-related Processes

Item	Processing Task	Processing Time
Common processing	Self-diagnosis	Fixed at 0.16 ms
Program execution	Executes ladder programs. The scan time is calculated using the program execution time or output refreshing time, whichever is greater.	The scan time is the sum of the execution times of basic and application instructions. It varies depending on the execution time of each instruction word.*1
Output refreshing	Writes the contents of output relays (Y) to an output module.	9 μs x number of modules calculated on a 16-points basis*2
Shared refreshing	Updates the contents of shared/extended shared relays (E) and shared/extended shared registers (R) when add-on CPU(s) are installed and shared refreshing is configured as a control-related process. In a single refreshing cycle, this task updates the contents of shared/extended shared relays (E) and shared/extended shared registers (R) included in the configuration for each CPU.	When an add-on CPU module is installed and shared refreshing is configured as a control-related process: - 0.0015 x (A/32 + B/2)+ 0.16 ms if the sequence CPU module for which the devices are refreshed is F3SP28, 38, 53, 58, 59, 66, 67 - 0.01 x (A/32 + B/2)+ 0.14 ms if the sequence CPU for which the devices are refreshed is other than those listed above. A = Number of shared relays configured in the sequence CPU module for refreshing B = Number of shared registers configured in the sequence CPU module for refreshing
	Not performed if no add-on CPU module is installed or shared refreshing is configured as a peripheral process.	0.00 ms
FL-net link refreshing	Updates the contents of link relays (L) and link registers (W) when FL-net interface module(s) is installed and FL-net link refreshing is configured as a control-related process.	0.001 x (A/16 + B)+ 0.39 ms A = Number of link relays to be refreshed B = Number of link registers to be refreshed
	Not performed if no FL-net interface module is installed or FL-net link refreshing is configured as a peripheral process.	0.00 ms
Input refreshing	Write the contents of input modules to CPU input relays (X).	6 μs x number of modules calculated on a 16-point basis*2
Synchronization processing	Ensures synchronization of operation control related processing and the simultaneity of data between the system of control-related processes and the system of peripheral processes.	- When output relays (Y) are used: 8μs x number of modules calculated on a 16-point basis*2
		- When FA link modules are used: 0.002 x (A/16 + B)+ 0.09 ms A = number of relays used in FA link to be refreshed B = number of registers used in FA link to be refreshed
		- When an FL-net module is used and FL-net link refreshing is configured as a peripheral process: 0.0014 x (A/16 + B)+ 0.1 ms A = Number of link relays to be refreshed B = Number of link registers to be refreshed
		- When an add-on CPU is installed and shared refreshing is configured as a peripheral process: 0.0015 x (A/32 + B/32 + C/2 + D/2)+ 0.03 ms A = number of relays set in CPU for refreshing B = number of relays set in the CPU itself C = number of registers set in CPU for refreshing D = number of registers set in the CPU itself
		- When an add-on CPU module is installed and shared refreshing is configured as a control-related process: 0.0015x (A/32 + B/2)+0.03 ms A = number of relays set in the CPU itself B = number of registers set in the CPU itself
Peripheral processing	Performs peripheral processes.	Minimum peripheral processing time (0.1 ms if not configured) or sum of program execution time + output refreshing time, whichever is greater.

*1: For details, see Section A7.5, "Instruction Execution Time."

*2: For the relationship between the types of I/O modules and the number of modules calculated on a 16-point basis, see the following table.

Table A7.1.2 Relationship between Types of I/O Module and Number of Modules Calculated on a 16-point Basis

Type of I/O Module	Number of Modules Calculated on a 16-point Basis	Type of I/O Module	Number of Modules Calculated on a 16-point Basis
4-point I/O module	1	16-point I/O relay	1
8-point I/O module	1	32-point I/O relay	2
14-point I/O module	1	64-point I/O relay	4

● System of Peripheral Processes

The latest, maximum and minimum of scan times taken by the system of peripheral processes are stored in special registers Z007 to Z009 in that order.

Table A7.1.3 Scan Time of System of Peripheral Processes

Item	Processing Task	Processing Time
Shared refreshing	Updates the contents of shared/extended shared relays (E) and shared/extended shared registers (R) when add-on CPU module(s) is installed and shared refreshing is configured as a peripheral process. In a single refreshing cycle, this task updates the contents of shared/extended shared relays (E) and shared/extended shared registers (R) included in the configuration setting, for each CPU.	When an add-on CPU module is installed and shared refreshing is set as a peripheral process: - 0.0015 x (A/32 + B/2) + 0.16 ms if the CPU module for which the devices are refreshed is F3SP28, 38, 53, 58, 59, 66, 67. - 0.01 x (A/32 + B/2) + 0.14ms if the CPU module for which the devices are refreshed is other than those listed above. A = number of relays set in CPU for refreshing B = number of registers set in CPU for refreshing
	Not performed if no add-on CPU module is installed or shared refreshing is configured as a control-related process.	0.00 ms
FA-link refreshing	Updates the contents of link relays and registers when an FA link module is installed.	- When an FA link module is installed: 0.02 x (A/16 + B)+ 0.59 ms A = number of relays used in FA link for refreshing B = number of registers used in FA link for refreshing
	Not performed if no FA link module is installed.	0.00ms
FL-net link refreshing	Updates the contents of link relays (L) and link registers (W) when FL-net interface module(s) is installed and FL-net link refreshing is configured as a peripheral process.	0.001 x (A/16 + B)+ 0.39 ms A = Number of link relays to be refreshed B = Number of link registers to be refreshed
	Not performed if no FL-net interface module is installed or FL-net link refreshing is configured as a control-related process.	0.00ms
Tool service	Processes commands input from the WideField2 connected to the sequence CPU module. Executes one command per service.	Varies with the type of command.
Link service	Processes commands input from a personal computer link module. Executes one command per service.	Varies with the type of command.
CPU service	Processes commands input from another CPU module. Executes one command per service.	Varies with the type of command.

A7.2 Setting Scan Monitoring Time

This configuration item sets the scan monitoring time. You can set the time to any value from 10 ms to 200 ms, in 10 ms increments. By default, the time is set at 200 ms.

A7.3 Example of Scan Time Calculation

Module configuration	:	Four 32-point input modules
	:	Four 32-point output modules
User program	:	20K steps consisting of LD and OUT instructions only, where the average execution time of these instructions is assumed to be 0.035 µs

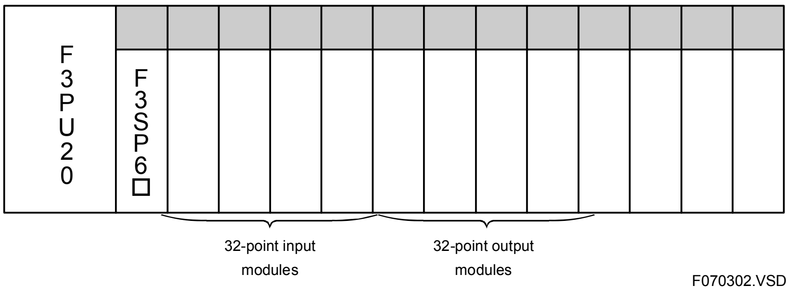


Figure A7.3.1 Module Configuration for Scan Time Calculation Example

Table A7.3.1 Example of Scan Time Calculation

Item	Calculation	Processing Time
Common processing	Fixed 0.16ms	0.16 ms
Program execution	0.035µs x 20480 = 717 µs	0.72 ms
Output refreshing	Number of modules calculated on a 16-point basis: 2 x 4 = 8 9 µs x 8 = 72 µs	0.072 ms*1
Shared refreshing	When no add-on CPU module is installed: 0.00 ms	0.00 ms
Input refreshing	Number of modules calculated on a 16-point basis: 2 x 4 = 8 6 µs x 8 = 48 µs	0.05 ms
Synchronization Processing	Number of modules calculated on a 16-point basis: 2 x 4 = 8 8 µs x 8 = 64 µs	0.06 ms
Peripheral processing	Minimum peripheral processing time, if not yet defined: 0.1 ms	0.1 ms*1
Scan time, which is the sum of all time spans listed above		1.0 ms

*1: The output refreshing time and the minimum peripheral processing time are excluded from scan time calculation because the sum of these time spans is smaller than the program execution time.

A7.4 Example of I/O Response Time Calculation

● Calculation of the minimum I/O response time

Input response time: 16 ms
Output response time: 1 ms
Scan time: 2 ms
Minimum I/O response time = Input response time + Scan time + Output response time
= 16 ms + 2 ms + 1 ms
= 19 ms

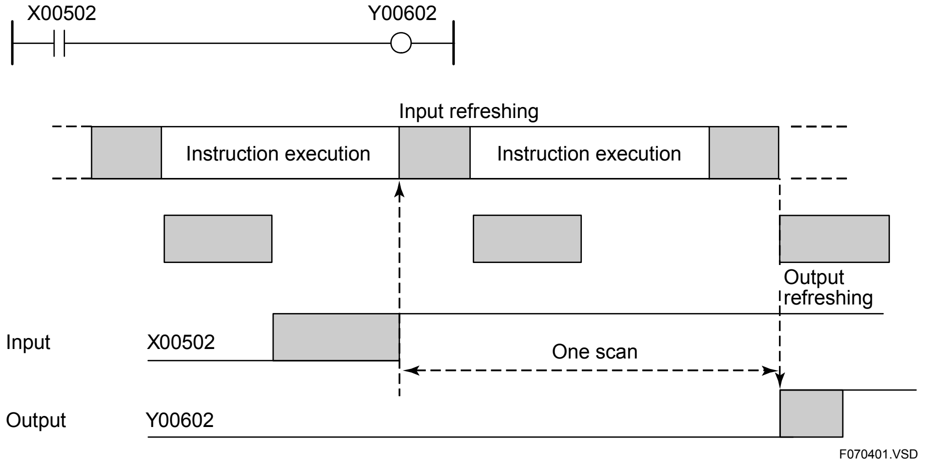


Figure A7.4.1 Minimum I/O Response Time

● Calculation of the maximum I/O response time

Input response time: 16 ms
Output response time: 1 ms
Scan time: 2 ms
Maximum I/O response time
= Input response time + (Scan time x 2) + Output response time
= 16 ms + (2 x 2) ms + 1 ms
= 21 ms

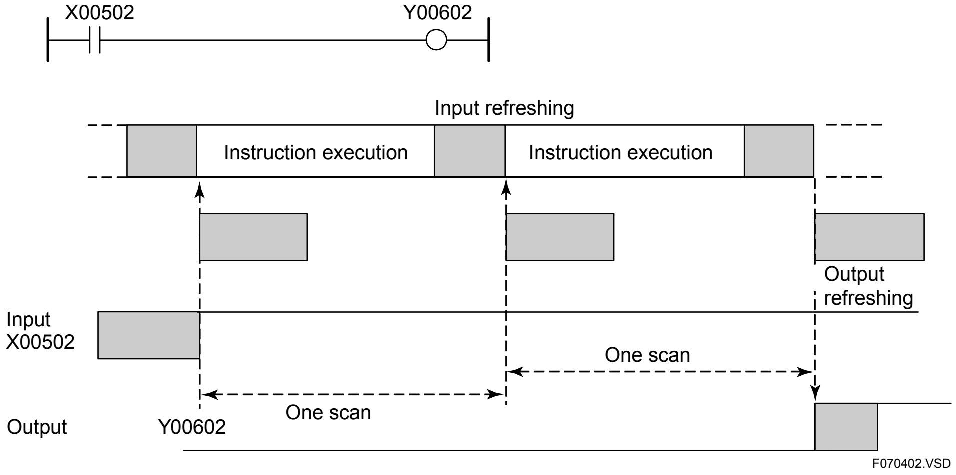


Figure A7.4.2 Maximum I/O Response Time

TIP

- The I/O response time refers to the total time taken to receive signal input from external input equipment, execute instructions and turn on external output equipment.
- Input response time refers to the time taken to load external input tag name using the input refreshing process.
- Output response time refers to the time taken to reflect the result of instruction execution in external output equipment using the output refreshing process.

A7.5 Instruction Execution Time

SEE ALSO

For details on the execution time of each instruction, see Appendix 3, "List of Ladder Sequence Instructions" of “Sequence CPU – Instructions User's Manual (IM34M6P12-03E).”

The instruction execution time varies somewhat depending on the contents of the input parameter or output parameter devices or the number of devices included in data transfer. The execution time lengths listed in the “List of Ladder Sequence Instructions” are typical. Use these values of the instruction execution time for reference purposes only when calculating the scan time. The instruction execution time varies somewhat with the conditions under which an instruction is executed as shown below. Use the instruction execution time (T) values given in the "List of Ladder Sequence Instructions" to calculate the instruction execution time under certain execution conditions.

Table A7.5.1 Calculation of Instruction Execution Time

Execution Conditions		F3SP66 F3SP67
		Instruction Execution Time (µs)
Differential type instruction	When executed	T+0.07
	When not executed	
Relay (BIN format)	16 bits	T+1.0xN1
	32 bits	T+1.4xN1
I/O relays (X/Y) defined in BCD format	16 bits	T+1.4xN2
	32 bits	T+1.8xN2
Index modification	Basic instruction	T+0.4xN3
	Application instruction	T+0.8xN3

Note:
- T = Instruction execution time given in “List of Ladder Sequence Instructions”.
- N1 = Number of relay devices
- N2 = Number of relay devices defined in BCD format
- N3 = Number of index-modified devices

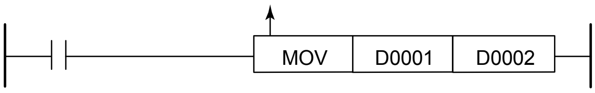
■ Examples of Calculation

Some examples for calculating the instruction execution time are given below.

SEE ALSO

For information on the execution time of an MOV instruction, see Appendix 3, “List of Ladder Sequence Instructions” of “Sequence CPU – Instructions User's Manual” (IM34M6P12-03E).

(1) Differential Type Instructions



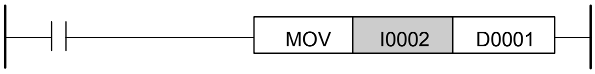
0.07+0.07=0.14 µs (for F3SP66, F3SP67)

F070501.VSD

Figure A7.5.1 Differential Type Instructions

(2) Relays (BIN Format)

Use parenthesized execution time value from the “List of Ladder Sequence Instructions,” if any.



Number of relay devices: N_1 = 1

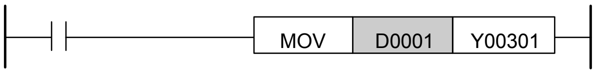
1.2+1.0=2.2 µs (for F3SP66, F3SP67)

F070502.VSD

Figure A7.5.2 Relays (BIN Format)

(3) I/O Relays (X/Y) Defined in BCD Format

Use parenthesized execution time value from the “List of Ladder Sequence Instructions,” if any.



Number of relay device defined in BCD format: N_2 = 1

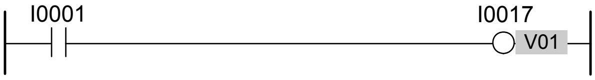
1.2+1.4=2.6 µs (for F3SP66, F3SP67)

F070503.VSD

Figure A7.5.3 I/O Relays (X/Y) Defined in BCD Format

(4) Index Modification

- Basic Instructions



Number of index-modified relay devices: N_3 = 1

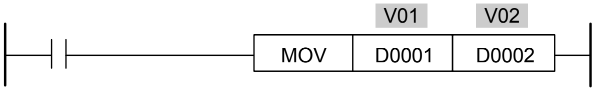
0.035+0.4=0.435 µs (for F3SP66, F3SP67)

F070504.VSD

Figure A7.5.4 Basic Instructions

- Application Instructions

Use parenthesized execution time value from the “List of Ladder Sequence Instructions,” if any.



Number of index-modified relay devices:N_3 = 2

1.2+0.8x2=2.8 µs (for F3SP66, F3SP67)

F070505.VSD

Figure A7.5.5 Application Instruction

A8. RAS Features

This Chapter describes self-diagnosis and the actions taken by the module in the event of errors.

SEE ALSO

For details on network-related diagnosis, see Subsection 1.3.3, "Troubleshooting Communications Problems" of "Sequence CPU – Network Functions User's Manual (for F3SP66-4S, F3SP67-6S)" (IM34M6P14-02E).

A8.1 Self Diagnosis

■ Basic Operation of Self Diagnosis

The sequence CPU performs self-diagnosis when it is powered on and during program execution. This includes checking its memory, processor and ladder instruction execution statuses. If the module detects an error, it performs error-time actions (to be described later) according to the severity of the error.

TIP

Error-time actions are not performed for errors detected by continuous-type application instruction errors (an error status is returned). It is the responsibility of a user program to take appropriate actions such as retry processing or error exit in such cases.

■ Severity of Failure Modes (Errors)

Errors are classified into three levels of severity: major failure, moderate failure and minor failure.

Table A8.1.1 Severity of Failure Modes (Errors)

Severity	Failure Condition	Failure Mode (Error)
Major failure	The core hardware is disabled.	- CPU error - Memory crash
Moderate failure	The user program cannot be started or run any further.	- Program error - I/O comparison error*1 - I/O module error*1 - Memory error - CPU error - Instruction error*1 - Scan timeout*1 - Startup error - Invalid instruction found - Excess number of I/O points - Subroutine error*1 - Interrupt error*1 - Subunit communication error*1 - Sensor CB scan timeout*1
Minor failure	An error has occurred but the program can continue execution	- Momentary power failure - Inter-CPU communication error - Subunit line switchover

*1: The table indicates the default severity level for this error, which can be reclassified as a minor failure by configuration.

Some failure modes (errors) classified by default as moderate failures can be reclassified by configuration as minor failures and vice versa. The severity of an error is not modified directly but indirectly by specifying a program operating mode (Run/Stop) as the "error-time action" to be taken for the error using the configuration function. Selecting the "Run" option for a moderate failure reclassifies it as a minor failure. Similarly, selecting the "Stop" option for a minor failure reclassifies it as a moderate failure.

■ Error Severity and Error-time Actions

When an error is detected, the module automatically undertakes the following error-time actions according to the severity of the error.

- Update LED indicators
  The LED indicator statuses are updated as described in the table below.
- Output to error log (system log)
  An error code is output to the error log (system log).
- Update statuses of special relays (M) and special registers (Z)
  The module turns on a special relay (M) and stores an error code to a special register (Z) to inform a user program of the error. For details on the corresponding error codes and device numbers, see "■ Details of Failure Modes (Errors)" given later in this chapter.
- Change program operating mode
  The module changes the operating mode of the program as described in the table below.
- FAIL signal contact output
  The FAIL signal contact output changes as described in the table below. The FAIL signal contact is an external output contact used for reporting a failure condition. For details, see the "Hardware Manual" (IM34M6C11-01E).
- Update external outputs of output modules
  The output contacts of output modules are updated as described in the table below. If an output contact is configured to "Reset" in the DIO Setup of the configuration, it will be reset to OFF. If it is configured to "Hold" in the DIO Setup of the configuration, the contact output is latched when an error occurs. In a multi-CPU configuration, a CPU module only updates contact outputs that are configured as "Used" in the DIO Setup of its configuration. For details on update of outputs of output modules, see "■ Specifying Output when Stopped (Holding/Resetting Output Relays When Sequence Stops)" of Subsection A4.1.4, "Configuring DIO Modules."

Table A8.1.2 Error Severity and Error-time Actions

Severity of Error	Program Operating Mode	Update LED Indicators	FAIL Signal Contact Output		Update Outputs of Output Modules	
			Between FAIL1 and COM	Between FAIL2 and COM	Group A[*1]	Group B[*2]
Major failure	Stop	RDY LED Off (green)	Shorted	Open	Default: RESET Configurable to HOLD or RESET on 16-terminal basis.	Always HOLD Not configurable
Moderate failure	Stop	ERR LED Lit (red)	Shorted	Open	Default: RESET Configurable to HOLD or RESET on 16-terminal basis.	Default: RESET Configurable to HOLD or RESET on 16-terminal basis.
Minor failure	Run (continue execution)	ALM LED Lit (yellow)	Open	Shorted	No update performed	No update performed

*1: Group A includes output modules with 32 or less outputs: F3YD64-1F, F3YD64-1P and F3WD64-□F

*2: Group B includes output modules with 64 outputs excluding those in Group A: F3WD64 and advanced function modules with output relays (Y)

CAUTION

The default DIO setup in the configuration is such that all contacts of output modules are set to OFF in the event of a moderate failure. This applies similarly to major failures for some modules.

■ Details of Failure Modes (Errors)

Table A8.1.3 Details on Self-diagnosis (1 of 4)

Failure Mode		Special Relay that Turns ON	Special Registers that Store Error Codes, Etc.	Stored Error Code	Failure Description		Corrective Actions
Major failure	CPU error	–	–	–	The CPU malfunctions due to noise or for other reasons.	System error	Check the installation environment for possible problems, such as noise sources. If the failure recurs, replace the module. In the event of a major failure (RDY LED is off), a power-off time is incorrectly output to the system log. If both an SPU error ($1104) and a power-off message are output to the system log, the SPU error is the real reason why the RDY LED is turned OFF.
Moderate failure	Startup error	M193	Z017 to Z019	$10nn	A failure has occurred during CPU initialization.	System error	1. Restrictions on module installation may have been violated. *1 2. Check the installation environment for possible problems, such as noise sources. If the failure recurs, replace the module.
	SPU error			$1101 $1102	The CPU for sequence processing has failed.	System error	The error may be due to a transient memory failure caused by effects of noise. Check the installation environment for possible problems, such as noise sources. If the failure recurs, replace the module.
				$1103 $1104	The CPU for sequence processing has failed (In the case of error $1104, the RDY is OFF)	System error	The error may be due to a transient memory failure caused by effects of noise. Check the installation environment for possible problems, such as noise sources. As this error may also be caused by a failure of the base module or a fiber-optic FA bus module, check for the presence of a subunit communication error ($8301). If the failure recurs, check the entire system.
	Memory error			$1201	A program checksum error has occurred.	Transient memory failure or system error *2.	The error may be due to a transient memory failure caused by effects of noise. Check the installation environment. Clear the memory and download the program again. If the failure recurs, replace the module.*3
				$1202	Inadvertent writing has been done to the M129 to M131 special relays for Run, Debug and Stop mode flags.	Application error	Check if there is any error in the values of index registers or in the parameters defining the number of devices in an instruction, which writes to multiple devices, such as a Block Move (BMOV) instruction.
					A device memory read/write check error has occurred.	System error	The error may be due to a transient memory failure caused by noise. Check the installation environment. Clear the memory and download the program again. If the failure recurs, replace the module. *3
				$1203	A system memory read/write check error has occurred.		
	Invalid instruction found			$1701	An invalid instruction has been encountered.		
	Program error			$1702	There is no END instruction in the program.		
				$2001	There is a mismatch in SUB, RET or JMP instructions.	System error	1. Verify that JMP, SUB and RET instructions are paired correctly. 2. The error may be due to a transient memory failure caused by effects of noise. Check the installation environment. Clear the memory and download the program again. If the failure recurs, replace the module. *3
	Excess number of I/O points			$2002	The number of I/O points has been exceeded.	Configuration error	Restrictions on module installation may have been violated. *1

*1: Check the modules according to Section A1.2, "Restrictions on Module Installation" of the "Hardware Manual" (IM34M6C11-01E).

*2: Try powering off and then powering on the module or downloading again. If error recovery is successful, it indicates a transient memory error. Otherwise, it is probably a system error.

*3: See "CAUTION" given at the end of the tables.

Table A8.1.4 Details on Self-diagnosis (2 of 4)

Failure Mode		Special Relay that Turns ON	Special Registers that Store Error Codes, Etc.	Stored Error Code	Failure Description		Corrective Actions
Moderate failure	Battery error	M194	—	$1801	There is a memory backup circuit failure or error in the pattern data used for backup check.	Transient memory error or system error	The error may be due to a transient memory failure caused by effects of noise. Check the installation environment. The module restarts with its factory settings when it is powered on after this error is detected. Download the program again. If the same failure recurs, replace the module.
	Subroutine error*4	M201	Z022 to Z024	$2201	The subroutine return (RET) instruction was not executed or there is no return destination.	Application error	1. Check if there is a jump out of or into the subroutine. 2. Check if a scan timeout has been detected within the subroutine.
				$2202	The maximum nesting depth of eight levels has been exceeded.	Application error	Check the depth of nesting when calling another subroutine in a given subroutine.
	Interrupt error*4	M201	Z022 to Z024	$2301	The interrupt return (IRET) instruction was not executed or there is no return destination.	Application error	1. Check if there is a jump out of or into the input module interrupt program. 2. Check if a scan timeout was detected within the input module interrupt program.
				$2302	There are more than eight pending interrupts.	Application error	There are more than eight pending interrupts. Check the detailed process of each interrupt, the number of interrupts, their frequency, etc. Check whether there are more than eight pending interrupts after powering on and before executing the program.
	Instruction error /Macro instruction error*4			$2101	A parameter is invalid.	Application error	Check if any abnormal value is set in the instruction parameter.
				$2102	Data is invalid.	Application error	Check if any abnormal value, such as one based on division by 0, is set in the instruction parameter.
				$2103	There is an error in BIN-to-BCD conversion	Application error	An invalid value may have been set in BIN-to-BCD conversion. Check the parameter where the error has occurred.
				$2104	There is an error in the pointers of the FIFO table.	Application error	1. Check if data written to the FIFO table has exceeded its capacity. 2. Check if an attempt has been made to read data values from the FIFO table when there is none. 3. Check if the default settings of the FIFO table are correct. Also check if the table has been destroyed by any other part of the program.
				$2105	The value defining a boundary between devices has been exceeded.	Application error	Check if there is any error in the values of index registers or in the parameters defining the number of devices in an instruction, which writes to multiple devices, such as a Block Move (BMOV) instruction.
				$2106	The FOR-NEXT loop is not consistent.	Application error	1. Check if there is a jump out of or into the FOR-NEXT loop. 2. Check if a scan timeout has been detected within the FOR-NEXT loop.
				$2107	The IL-ILC loop is not consistent.	Application error	1. Check if there is a jump out of or into the IL-ILC loop. 2. Check if a scan timeout has been detected within the IL-ILC loop.
				$2501	The macro return (MRET) instruction was not executed or there is no return destination.	Application error	1. Check if there is a jump out of or into the macro instruction. 2. Check if a scan timeout has been detected within a macro instruction.
				$2502	Macro call nesting is deeper than 7 levels.	Application error	Although a macro instruction may call another macro instruction (nesting), the nesting may not be more than 7 levels deep.

*4: You can specify by configuration whether to stop or continue program execution if this error is detected.

Table A8.1.5 Details on Self-diagnosis (3 of 4)

Failure Mode		Special Relay that Turns ON	Special Registers that Store Error Codes, Etc.	Stored Error Code	Failure Description		Corrective Actions
Moderate failure	I/O comparison error	M202	Z027 to Z029	$2401	- The condition of module installation is not consistent with the program. - The number of special module High-speed Read (HRD) instructions or special module High-speed Write (HWR) instructions exceeded the limit (error code: $2401).	Application error	1. There may be a mismatch between the I/O relay (X/Y) devices specified in the program and those contained in the installed I/O module. Check if the instruction parameter in question is consistent with the installed I/O module. 2. Check if the number of special module High-speed Read (HRD) instructions or the number of special module High-speed Write (HWR) instructions exceeded 64.
				$2402 (READ/ WRITE)	$2402 (READ/WRITE)	Application error	There may be a mismatch between the slot number in READ/WRITE instructions used in the program and that of the installed I/O module. Check if the instruction parameter in question is consistent with the installed I/O module.
				$2403 (HRD/ HWR)	$2403 Special module High-speed Read instruction (HRD)/special module High-speed Write instruction (HWR)	Application error	There may be a mismatch between the slot number in a special module High-speed Read (HDR) instruction or a special module High-speed Write (HWR) instruction used in the program and that of the installed I/O module. Check if the instruction parameter in question is consistent with the installed I/O module.
	I/O module error	M203	Z033 to Z040	$8000	- There is a failure to read from or write to the I/O module. - There is a communication failure in the fiber-optic FA-bus module. - An attempt has been made to reset one of the other sequence CPU modules in a multi-CPU system.	Application error	1. Check if the subunit is turned off. 2. Check if there is any problem with the cable of the fiber-optic FA-bus module. 3. Do not reset the CPU modules individually. Rather, reset them all at once from the main CPU. 4. The I/O module may be defective. Replace it.
	Scan timeout*4	M204	–	$1401	The scan monitoring time has been exceeded.	Application error	1. Check if the iteration-counter values of the FOR-NEXT loop are correct. 2. Check for the presence of endless loop caused by JMP instructions. 3. Adjust the scan monitoring time according to the execution time of the application program.
	Subunit communication error*4	M210	Z089 to Z096	$8301	There is a failure to read from or write to the subunit.	Open-circuited cable, loss of power to subunit or system error	1. Check if the subunit is turned off. 2. Check if there is any problem with the cable of the fiber-optic FA-bus module. 3. The fiber-optic FA-bus module may be defective. Replace it.

*4: You can specify by configuration whether to stop or continue program execution if this error is detected.

Table A8.1.6 Details on Self-diagnosis (4 of 4)

Failure Mode		Special Relay that Turns ON	Special Registers that Store Error Codes, Etc.	Stored Error Code	Failure Description		Corrective Actions
Moderate failure	Sensor CB scan timeout*4	M212	—	$1402	The CPU fails to maintain the execution interval of the sensor control block as it is exceeded by the sum of its I/O refreshing time and execution time.	Application error	1. In the case of interruption by the sensor control block after completion of instruction execution, set the execution interval at 1 ms or longer, preferably at the largest possible value. 2. Check the number of input/output words in the sensor control block and the block's execution time and minimize both values. 3. Check and minimize any code section between CBD and CBE instructions, during which execution of the sensor control block is disabled.
Minor failure	Momentary power failure	M195	—	$1302	The CPU indicates that a momentary power failure has occurred.	Momentary power failure	If this error occurs too frequently, check the power supply for possible problems. If a UPS is in use, check that it has captured peak values of its supply voltage waveform. If the failure still occurs frequently while there is no problem with the wave-form, the power supply module and/or sequence CPU module may be defective. Replace it.
	Inter-CPU communication error	M196	—	$400n	There is a communication failure in shared devices.	System error	There may be a failure in one of the other CPUs in a multi-CPU system. Do not reset the CPU modules individually. Rather, reset them all at once from the main CPU. If this error recurs, replace the CPU modules.
	Subunit line switchover	M211	Z089 to Z096	$8401	There is a problem with the paired cables attached to remote I/O modules in a loop configuration.	Open-circuited cable	1. Check if there is any problem with the cable of the fiber-optic FA-bus module. 2. The fiber-optic FA-bus module may be defective. Replace it.

*4: You can specify by configuration whether to stop or continue program execution if this error is detected.

SEE ALSO

- For details on error log (system log) messages, see "FA-M3 Programming Tool WideField2 User’s Manual" (IM34M6Q15-01E) along with Section B3, "Common Specifications of Smart Access Functions."
- For details on the M210 (Subunit Communication Error) and M211 (Subunit Line Switchover) relays, see "Fiber-optic FA-bus Module and Fiber-optic FA-bus Type 2 Module User's Manual" (IM34M6H45-01E).

TIP

How to Restore the CPU Module to its Factory Settings:

Execute the "Restore Factory Settings" rotary switch function. For details on rotary switch functions, see Chapter B1, "Rotary Switch Functions."

CAUTION

In the event of transient memory errors caused by noise or other external factors, first try downloading the application program at a later time. If the error disappears, you can continue to use the module.

If the error recurs, this may indicate a hardware failure so you should consider replacing the module.

CAUTION

Power off time may be incorrectly recorded in any of the following situations:

(a) A major failure (RDY turned off) is detected.

(b) The CPU module is removed from the base unit with power turned on.

(c) Power off is not correctly detected due to power supply module failure.

If power off time is not correctly recorded, check for causes (a) and (b). If both causes are ruled out but the problem recurs frequently with no apparent cause, the power supply module or the base module may be faulty.

CAUTION

The CPU module may sometimes record a power off time that is later than the startup completion time of the next startup. This may happen, in particular, when the CPU module is switched off after running for an extended period of time. This apparent discrepancy is not an error but happens because the power off time is obtained by adding the elapsed time maintained by the system timer to the most recent power on time but the power on time is recorded based on the backup real time clock reading.

A8.2 Updating Error Status Indicators after Correcting Moderate or Minor Failures

After eliminating the cause of a moderate or minor failure, initialize the states of special relays (M), special registers (Z) and LED indicators as instructed below.

■ Updating Error Status after Removing Moderate Failures

To reset special relays (M) and special registers (Z), as well as turn off the ERR LED indicator after eliminating the cause of a moderate failure, use the following procedure.

(1) Turn on the power again.

(2) Switch the sequence CPU to **Run** or **Debug** mode using WideField2.

■ Updating Error Status after Removing Minor Failures

To reset special relays (M) and special registers (Z), as well as turn off the ALM LED indicator after eliminating the cause of a minor failure, use the following procedure.

(1) Turn on the power again.

(2) Switch the sequence CPU to **Run** or **Debug** mode using WideField2.

(3) Perform an “alarm acknowledgement.”

A9. Setup Description

This chapter describes how to configure the module.
There are primarily two means of configuring the module, namely, through configuration and CPU properties.

Table A9.1 Overview of Configuration

Setup Name (Window Tab)	Sub Item	Description
Device Capacities	–	Define and allocate the number of device units for each device type
Operation Control	Scan monitoring time	Scan timeout interval
	Constant scan	Constant scan time
	Error-time action	Define whether to stop program execution in case of various error events
	Program execution mode	Execution mode for program blocks
	Momentary power failure detection mode	Configure power failure detection
	Peripheral processing time	Processing time of peripheral processes
Initial Data	–	Configure initial data of data registers
DIO Setup	Terminal usage	Specify I/O terminals to be refreshed
	Data code	Select BIN/BCD data code for I/O
	Input sampling interval	Sampling of input modules
	Output when stopped (Reset/Hold external outputs when sequence stops)	Specify output when system fails
FA Link	–	Configure FA link/FL-net systems
Sampling Trace	–	Configure sampling trace
Communications Setup	–	Communications Setup for SIO port
Interrupt Setup	Sensor control block	Sensor CB execution interval and timing of interrupt
	Input interrupt	Timing of input interrupt
	Priority of Interrupts	Priority of sensor CB interrupt versus input interrupt
Power Failure/Local	–	Configure data lock-up range at power failure
Shared Refreshing	–	Configure refreshing of shared devices
FL-net Refreshing	–	Configure FL-net refreshing
Function Removal	–	Disable selected functions

Note: This module does not support ROM setup.

Table A9.2 Overview of CPU Properties

Setup Name (CPU Property Group)	Description
Network setup	Select the network type for the 10BASE-T/100BASE-TX connector. The module only supports Ethernet with auto-negotiation.
Ethernet setup	Ethernet setup (IP address, subnet mask, DNS, gateway, etc.)
Socket setup	Socket communications function setup (UDP/IP broadcast, TCP/IP keep-alive, etc.)
Higher-level link service setup	Higher-level link service (personal computer link) setup (port setup, data format, etc.)
FTP client setup	FTP client function setup (Destination FTP server, etc.)
FTP server setup	FTP server function setup (password, log, maximum connections, port setup, etc.)
Rotary switch setup	Disable selected rotary switch values. Boot modes cannot be disabled.
Network filter setup	Specify remote hosts that are allowed to establish connection using socket function, FTP function, etc.

A9.1 Setup Types and Characteristics

The module has two types of setup:

- **Configuration**
- **CPU properties**

The table below summarizes the characteristics of configuration and CPU properties.

Table A9.1.1 Characteristics of Configuration and CPU Properties

Characteristic	Configuration	CPU Properties
Included settings	Settings that are unlikely to require modification during operation	Settings that are likely to require modification during operation
Relationship with project	Integrated	Independent
Online modification	No	Yes
Modification by program	No	Yes
Modification without using programming tool	No	Yes
Security	Executable program protection	CPU property protection

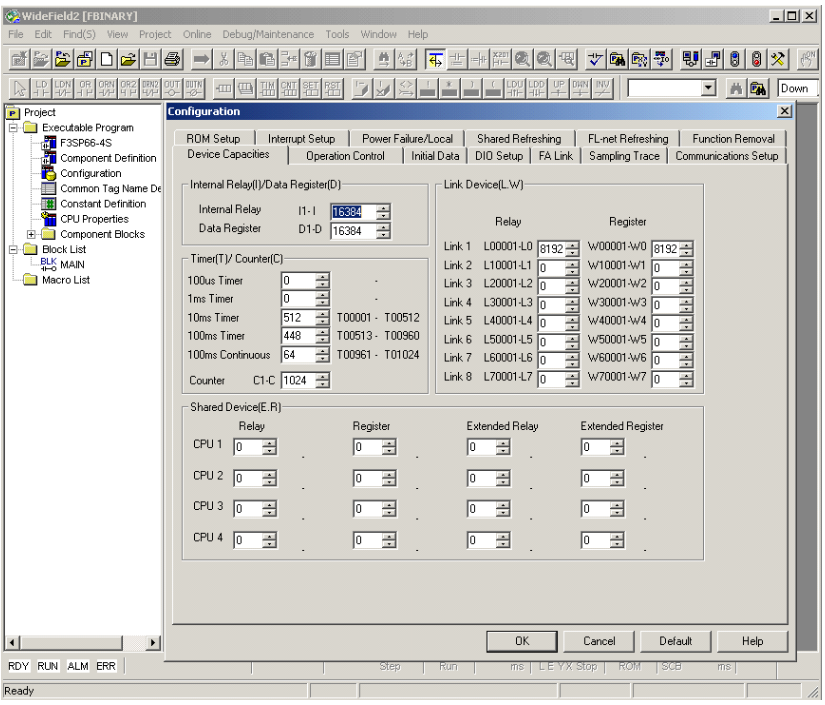
■ Configuration

Configuration is part of a project, and cannot be modified after a project is stored to the module. It primarily includes setup items that are unlikely to require modification during operation (e.g. operation mode and I/O allocation).

The WideField2 software is required for editing configuration data.

SEE ALSO

For details on configuration, see Section A9.2, "Configuration Items" and Section A9.3, "Editing and Saving Configuration."



Internal Relay(I)/Data Register(D)

Internal Relay	I1-I	16384
Data Register	D1-D	16384

Timer(T)/ Counter(C)

100us Timer	0	-
1ms Timer	0	-
10ms Timer	512	T00001 - T00512
100ms Timer	448	T00513 - T00960
100ms Continuous	64	T00961 - T01024
Counter C1-C	1024	

Link Device(L.W)

	Relay		Register	
Link 1	L00001-L0	8192	W00001-W0	8192
Link 2	L10001-L1	0	W10001-W1	0
Link 3	L20001-L2	0	W20001-W2	0
Link 4	L30001-L3	0	W30001-W3	0
Link 5	L40001-L4	0	W40001-W4	0
Link 6	L50001-L5	0	W50001-W5	0
Link 7	L60001-L6	0	W60001-W6	0
Link 8	L70001-L7	0	W70001-W7	0

Shared Device(E.R)

	Relay		Register		Extended Relay		Extended Register	
CPU 1	0	-	0	-	0	-	0	-
CPU 2	0	-	0	-	0	-	0	-
CPU 3	0	-	0	-	0	-	0	-
CPU 4	0	-	0	-	0	-	0	-

FA0901.VSD

Figure A9.1.1 Configuration Screen (shown with default values for F3SP66-4S)

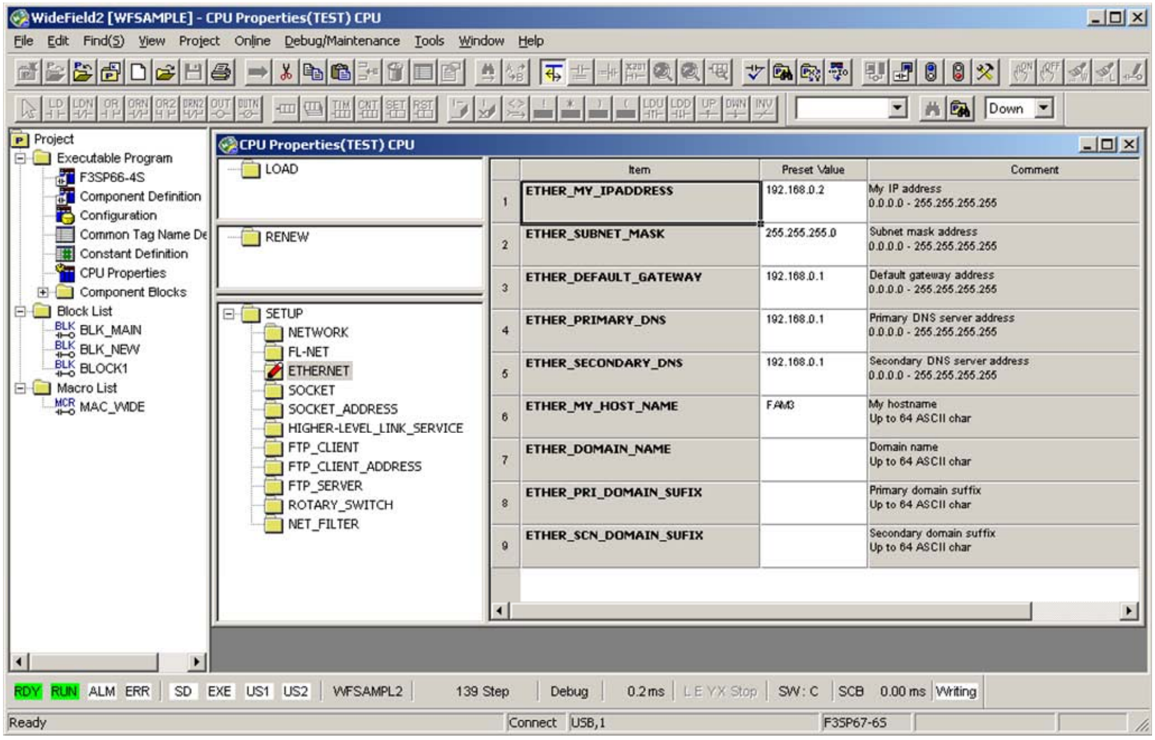
■ CPU Properties

CPU properties are independent of any project, and can be overwritten even when a project is stored in the module. CPU properties data can be stored to the module by itself even when no project is stored in the module. It primarily includes settings that are likely to require modification during operation (e.g. IP address and FTP login account).

Another feature of CPU properties is that it can be saved in text format in a CPU property file, which can then be edited using any generic text editor. Furthermore, CPU properties can be saved to a CPU property file using smart access functions (e.g. rotary switch function) without requiring the use of the WideField2 software.

SEE ALSO

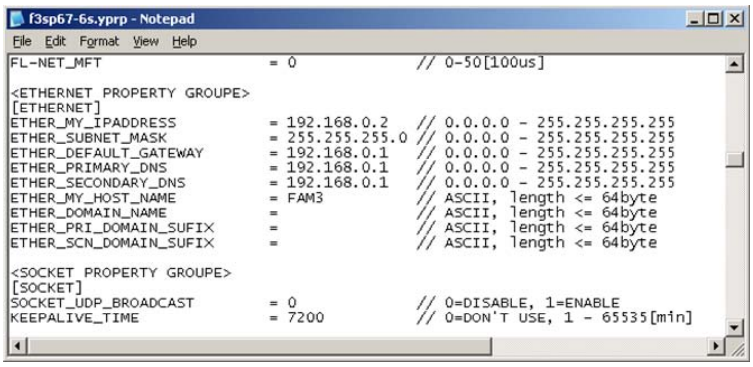
For details on CPU properties, see Section A9.4, "CPU Property File Specifications," Section A9.5, "CPU Property Items" and Section A9.6, "Editing and Saving CPU Properties."



	Item	Preset Value	Comment
1	ETHER_MY_IPADDRESS	192.168.0.2	My IP address 0.0.0.0 - 255.255.255.255
2	ETHER_SUBNET_MASK	255.255.255.0	Subnet mask address 0.0.0.0 - 255.255.255.255
3	ETHER_DEFAULT_GATEWAY	192.168.0.1	Default gateway address 0.0.0.0 - 255.255.255.255
4	ETHER_PRIMARY_DNS	192.168.0.1	Primary DNS server address 0.0.0.0 - 255.255.255.255
5	ETHER_SECONDARY_DNS	192.168.0.1	Secondary DNS server address 0.0.0.0 - 255.255.255.255
6	ETHER_MY_HOST_NAME	FAM3	My hostname Up to 64 ASCII char
7	ETHER_DOMAIN_NAME		Domain name Up to 64 ASCII char
8	ETHER_PRI_DOMAIN_SUFIX		Primary domain suffix Up to 64 ASCII char
9	ETHER_SCN_DOMAIN_SUFIX		Secondary domain suffix Up to 64 ASCII char

FA0902.VSD

Figure A9.1.2 CPU Properties (as displayed in WideField2)



```
FL-NET_MFT                  = 0             // 0-50[100us]

<ETHERNET PROPERTY GROUPE>
[ETHERNET]
ETHER_MY_IPADDRESS          = 192.168.0.2   // 0.0.0.0 - 255.255.255.255
ETHER_SUBNET_MASK           = 255.255.255.0 // 0.0.0.0 - 255.255.255.255
ETHER_DEFAULT_GATEWAY       = 192.168.0.1   // 0.0.0.0 - 255.255.255.255
ETHER_PRIMARY_DNS           = 192.168.0.1   // 0.0.0.0 - 255.255.255.255
ETHER_SECONDARY_DNS         = 192.168.0.1   // 0.0.0.0 - 255.255.255.255
ETHER_MY_HOST_NAME          = FAM3          // ASCII, length <= 64byte
ETHER_DOMAIN_NAME           =               // ASCII, length <= 64byte
ETHER_PRI_DOMAIN_SUFIX      =               // ASCII, length <= 64byte
ETHER_SCN_DOMAIN_SUFIX      =               // ASCII, length <= 64byte

<SOCKET PROPERTY GROUPE>
[SOCKET]
SOCKET_UDP_BROADCAST        = 0             // 0=DISABLE, 1=ENABLE
KEEPALIVE_TIME              = 7200          // 0=DON'T USE, 1 - 65535[min]
```

FA0903.VSD

Figure A9.1.3 CPU Properties (as displayed in text editor)

A9.2 Configuration Items

The table below summarizes the configuration items.

Table A9.2.1 Summary of Configuration Items

Setup Name (Window Tab)	Sub Item	SEE ALSO
Device Capacities	–	A9.2.1, "Device Capacities"
Operation Control	Scan monitoring time	A9.2.2, "Operation Control"
	Constant scan	
	Error-time action	
	Program execution mode	
	Momentary power failure detection mode	
	Peripheral processing time	
Initial Data	–	A9.2.3, "Initial Data"
DIO Setup	Terminal usage	A9.2.4, "DIO Setup"
	Data code	
	Input sampling interval	
	Output when stopped (Reset/Hold external outputs when sequence stops)	
FA Link	–	A9.2.5, "FA Link"
Sampling Trace	–	A9.2.6, "Sampling Trace"
Communications Setup	–	A9.2.7, "Communications Setup"
Interrupt Setup	Sensor control block	A9.2.8, "Interrupt Setup"
	Input interrupt	
	Priority of interrupts	
Power Failure/Local	–	A9.2.9, "Power Failure/Local"
Shared Refreshing	–	A9.2.10, "Shared Refreshing"
FL-net Refreshing	–	A9.2.11, "FL-net Refreshing"
Function Removal	–	A9.2.12, "Function Removal"

Note: This module does not support ROM setup.

A9.2.1 Device Capacities

Table A9.2.2 Device Capacities of Configuration (F3SP66-4S)

Item		Default Value	Value Range
Shared Device (E,R)	Shared Relay (E)	0	2048 max. for all CPUs combined in increments of 32
	Extended Shared Relay (E)	0	2048 max. for all CPUs combined in increments of 32
	Shared Register (R)	0	1024 max. for all CPUs combined in increments of 2
	Extended Shared Register (R)	0	3072 max. for all CPUs combined in increments of 2
Link Device (L,W)	Link Relay (L)	Link 1 : 8192 Links 2 to 8 : 0	8192 max. for all links combined in increments of 16
	Link Register (W)	Link 1 : 8192 Links 2 to 8 : 0	8192 max. on for all links combined in increments of 16
Timer (T)/ Counter(C)	100μs Timer	0	2048 for timers and counters combined in increments of 1; 16 max. for 100 μs timers; timer numbers are continuous.
	1ms Timer	0	
	10ms Timer	512	
	100ms Timer	448	
	100ms Continuous Timer	64	
	Counter	1024	2048 max. for timers and counters combined in increments of 1

Table A9.2.3 Device Capacities of Configuration (F3SP67-6S)

Item		Default Value	Value Range
Shared device (E,R)	Shared Relay (E)	0	2048 max. for all CPUs combined in increments of 32
	Extended Shared relay (E)	0	2048 max. for all CPUs combined in increments of 32
	Shared Register (R)	0	1024 max. for all CPUs combined in increments of 2
	Extended Shared Register (R)	0	3072 max. for all CPUs combined in increments of 2
Link device (L,W)	Link Relay (L)	Links 1 to 2: 8192 Links 3 to 8: 0	16384 max. for all links combined in increments of 16
	Link Register (W)	Links 1 to 2: 8192 Links 3 to 8: 0	16384 max. for all links combined in increments of 16
Timer (T)/ Counter(C)	100μs Timer	0	3072 for timers and counters combined in increments of 1; 16 max. for 100 μs timers; timer numbers are continuous.
	1ms Timer	0	
	10ms Timer	1024	
	100ms Timer	896	
	100ms Continuous Timer	128	
	Counter	1024	3072 max. for timers and counters combined in increments of 1

A9.2.2 Operation Control

Table A9.2.4 Operation Control of Configuration (for F3SP66-4S, F3SP67-6S)

Item		Default Value	Value Range
Scan Monitoring Time		200 ms	10 to 200 ms in increments of 10 ms
Constant Scan		Do Not Use	1.0 ms to 190.0 ms in increments of 0.1 ms
Error-Time Action	I/O module error	Stop	Run/Stop
	I/O comparison error	Stop	
	Instruction parameter error	Stop	
	Scan timeout	Stop	
	Subroutine error	Stop	
	Interrupt error	Stop	
	Subunit communication error	Run	
	Sensor CB scan timeout	Stop	
Program Execution Mode		All Blocks	All Blocks/Specified Blocks
Momentary Power Failure Detection Mode	Valid for all power supply modules except F3PU01-0N	Standard mode	Standard/Immediate
Peripheral Processing Time		Not set up	100 μs to 190 ms in increments of 100 μs

A9.2.3 Initial Data

Table A9.2.5 Initial Data of Configuration (for F3SP66-4S, F3SP67-6S)

Item		Default Value	Value Range
Initial Data of Data Registers	Data Register (D)	None	Configurable for up to 1024 contiguous registers from a starting number

A9.2.4 DIO Setup

Table A9.2.6 DIO Setup of Configuration (for F3SP66-4S, F3SP67-6S)

Item		Default Value	Value Range
DIO Setup	Terminal Usage (Module used/not used)	Used	Used/Not used/Use with SCB Configurable on 16-terminal basis
	Data Code	BIN	BIN/BCD Configurable on 16-terminal basis
	Input Sampling Interval	16 ms	16 ms/1 ms/250 μs/62.5 μs/Always Configurable on 16-terminal basis
	Output When Stopped (Reset/hold external outputs when sequence stops)	Reset	Reset/Hold Configurable on 16-terminal basis

Note: When the same input module is used with sensor control block and normal block, you must perform setup on 32-bit basis.

Note: By default, all contacts of output modules are set to OFF in the event of a moderate error.

A9.2.5 FA Link

Table A9.2.7 FA Link of Configuration (for F3SP66-4S, F3SP67-6S)

Item	Default Value	Value Range
FA Link System Setup (Mapping between FA link and FL-net system numbers and slot numbers)	Automatic setup	Manual setup/Automatic setup System numbers from 1 to 8 Slot numbers from 1 to 16

A9.2.6 Sampling Trace

Table A9.2.8 Sampling Trace of Configuration (for F3SP66-4S, F3SP67-6S)

Item	Default Value	Value Range
Sampling Trace	Disabled	Enabled/Disabled
Sampling Method	Undefined	Sampling Method: TRC instruction/Scan/Periodic Delay: -1023 to 1023
Trigger Condition	Undefined	Device Address: Any device Trigger: Rising Edge of Specified Relay/Falling Edge of Specified Relay/Data Coincidence
Sampled Devices	Undefined	Any device

A9.2.7 Communications Setup

Table A9.2.9 Communications Setup of Configuration (for F3SP66-4S, F3SP67-6S)

Item		Default Value	Value Range
SIO Port	Communication Mode	Mode 0: 9600bps Even Parity	Mode 0: 9600bps Even Parity Mode 1: 9600bps No Parity Mode 2: 19200bps Even Parity Mode 3: 19200bps No Parity Mode 4: 38400bps Even Parity Mode 5: 38400bps No Parity Mode 6: 57600bps Even Parity Mode 7: 57600bps No Parity Mode 8: 115200bps Even Parity Mode 9: 115200bps No Parity
CPU Personal Computer Link	Use Personal Computer Link	Used	Used (fixed)
	Checksum	No	Yes/No
	End Character	No	Yes/No
	(Program) Protection	No	Yes/No

A9.2.8 Interrupt Setup

Table A9.2.10 Interrupt Setup of Configuration (for F3SP66-4S, F3SP67-6S)

Item		Default Value	Value Range
Sensor CB	Execution Interval	200 μs	200 μs to 25.0 ms in increments of 100 μs
	Timing of Interrupt	Immediate (during instruction execution)	After Instruction /Immediate (during instruction execution)
Input Interrupt	Timing of Interrupt	After instruction	After Instruction /Immediate (during instruction execution)
Priority of Interrupts		Sensor CB interrupt has priority	Sensor CB interrupt has priority /Input interrupt has priority

A9.2.9 Power Failure/Local

Table A9.2.11 Data Lock-up Range at Power Failure of Configuration (for F3SP66-4S, F3SP67-6S)

Item		Default Value	Value Range
Data Lock-up Range at Power Failure	Internal Relay (I)	I0001 to I1024	Configurable on 32-relay basis; contiguous from a starting number
	Shared Relay (E) Extended Shared Relay (E)	Non-latching type	
	Link Relay (L)	Non-latching type	Configurable on 16-relay basis
	Timer (T)	Non-latching type (except for continuous timers)	Configurable on per timer or per counter basis; contiguous from a starting number
	Counter (C)	All latched	
	Data Rregister (D)	All latched	Configurable on 2-register basis; contiguous from a starting number
	Shared Register (R) Extended Shared Register (R)	Non-latching type	
	Link Register (W)	Non-latching type	Configurable on 16-register basis

Note:

- If the number of shared relays (E) used is less than 2048, there is a gap between the last device number for shared relays and the first device number for extended shared relays, which is always fixed at E2049. However, the data lock-up range setup is mapped as if the two device areas are contiguous with no gap in between.
  e.g.: Assuming 1024 shared relays (E) and 2048 extended shared relays (E) are used,
  if the data lock-up range for power failure is specified with starting number as 513 and number of units as 1024, shared relays (E) E0513 to E1024 and extended shared relays E2049 to E2560 will be latched.
- If the number of shared registers (R) used is less than 1024, there is a gap between the last device number for shared registers and the first device number for extended shared registers, which is always fixed at R1025. However, the data lock-up range setup is mapped as if the two device areas are contiguous with no gap in between.
- The data lock-up range setup for link relay and link register are mapped to contiguous devices starting from their respective starting numbers with the following exceptions:
  L/W01024 is followed by L/W10001; L/W11024 is followed by L/W20001.
  L/W21024 is followed by L/W30001; L/W31024 is followed by L/W40001.
  L/W41024 is followed by L/W50001; L/W51024 is followed by L/W60001.
  L/W61024 is followed by L/W70001.
  The above rule applies when the number of link relays or registers used is defined as 1024.
  If the number is defined as 2048, L/W02048 is followed by L/W10001.
  e.g.: Assuming there are 1024 link relays (L) each for link 1, link 2 and link 3,
  if the data lock-up range is specified with starting number as 10513 and the number of units as 1024,
  the devices latched will be L10513 to L11024 for link 1 and L20001 to L20512 for link 2.

Table A9.2.12 Local Devices of Configuration (F3SP66-4S)

Item		Default Value	Value Range
Local Devices	Internal Relay (/I)	Starting No.=0; Points =0	I00001 to I16384
	Data Register (/D)	Starting No.=0; Points =0	D00001 to D16384
	File Register (/B)	Starting No.=0; Points =0	B00001 to B32768
	Timer (/T)	Starting No.=0; Points =0	T0001 to T2048
	Counter (/C)	Starting No.=0; Points =0	C0001 to C2048

Note: The upper limit of the configuration range follows the Device Capacities tab of configuration.

Table A9.2.13 Local Devices of Configuration (F3SP67-6S)

Item		Default Value	Value Range
Local Devices	Internal Relay (/I)	Starting No.=0; Points =0	I00001 to I32768
	Data Register (/D)	Starting No.=0; Points =0	D00001 to D32768
	File Register (/B)	Starting No.=0; Points =0	B00001 to B262144
	Timer (/T)	Starting No.=0; Points =0	T0001 to T3072
	Counter (/C)	Starting No.=0; Points =0	C0001 to C3072

Note: The upper limit of the configuration range follows the Device Capacities tab of configuration.

A9.2.10 Shared Refreshing

Table A9.2.14 Shared Refreshing of Configuration (for F3SP66-4S, F3SP67-6S)

Item	Default Value	Value Range
Shared Refreshing Range (partial disabling of refreshing)	All Refreshed	Enable/Disable refreshing, configurable separately for shared relays (E), shared registers (R), extended shared relays (E) and extended shared registers (R) of each CPU module
Shared Refreshed Data	Simultaneous	Simultaneous/Non-simultaneous
Shared Refreshing Mode	Peripheral Process	Peripheral Process Control Process

A9.2.11 FL-net Refreshing

Table A9.2.15 FL-net Refreshing of Configuration (for F3SP66-4S, F3SP67-6S)

Item	Default Value	Value Range
Common Data Refreshing Mode	Peripheral Process	Peripheral Process or Control Process
Common Data Refreshing Range	All Nodes	All Nodes or Some Nodes Node numbers 1 to 254

A9.2.12 Function Removal

Table A9.2.16 Function Removal of Configuration (for F3SP66-4S, F3SP67-6S)

Item	Default Value	Value Range
Remove Remote Programming Service	Enabled	Enabled/Disabled
Remove Higher-level Link Service	Enabled	Enabled/Disabled
Remove FTP Server Function	Enabled	Enabled/Disabled
Remove Rotary Switch Function	Enabled	Enabled/Disabled
Remove Virtual Directory Function	Enabled	Enabled/Disabled
Remove Card Batch File Function	Enabled	Enabled/Disabled

TIP

If you remove all functions, you will no longer be able to access the CPU module, unless via an external module such as an Ethernet module. To recover from this state, you need to restore the factory settings. To do so, power off the CPU module, set the MODE switch to 'C,' mount the CPU module in slot 5 or a higher slot, and then switch on the power. Processing begins and the **1** LED, **2** LED, **4** LED and **8** LED turn on and off in turn. When processing ends, the CPU module is restored to its factory settings and the **RDY** LED turns on. Beware, however, that this procedure clears all user information such as ladder programs and CPU properties.

A9.3 Editing and Saving Configuration

This section describes how to edit a configuration and save it to the module.

A9.3.1 Editing Configuration

Configuration data can be edited using the WideField2 software. It must be done separately for each project.

SEE ALSO

For details on how to edit configuration data, see "FA-M3 Programming Tool WideField2 User's Manual" (IM34M6Q15-01E).

A9.3.2 Saving Configuration

When you save a project to the module, configuration data is saved with it. The format of a project file depends on how the project is saved, as summarized in the table below.

Table A9.3.1 Methods for Saving Configuration and the Corresponding Project File Format

If Project is Saved Using:	Format of Project File	SEE ALSO
WideField2	WideField2 format	"FA-M3 Programming Tool WideField2 User's Manual" (IM34M6Q15-01E)
Rotary switch function	Card load format	B1, "Rotary Switch Functions"
Card batch file function	Card load format	B2, "Card Batch File Function"
Virtual directory command	Card load format	Section 3.7, "Virtual Directory Commands" of "Sequence CPU – Network Functions User's Manual (for F3SP66-4S, F3SP67-6S)" (IM34M6P14-02E)
Personal computer link command	Personal computer link command format	"Personal Computer Link Commands User's Manual" (IM34M6P41-01E)

SEE ALSO

For details on card load format, see Subsection C3.2.3, "FA-M3 File Types."

A9.4 CPU Property File Specifications

This section describes the specifications of a CPU property file.
This information is intended for users who wish to edit a CPU property file directly using a text editor.

SEE ALSO

Users who intend to edit CPU properties using WideField2 need only read "■ Structure of CPU Property File."

A CPU property file is a text file coded with setup items for the module. It can be directly edited using a generic text editor or edited using the WideField2 software, which provides a friendlier user interface with displayed comments.

This section discusses the following aspects about the CPU property file:

- **Structure of CPU Property File**
- **LOAD Part (LOAD PROPERTY SELECTOR PART)**
- **RENEW Part (RENEW PROPERTY SELECTOR PART)**
- **PROTECT Part (PROPERTY PROTECT PART)**
- **SETUP Part (CPU PROPERTY PART)**
- **Basic Syntax of CPU Property File**

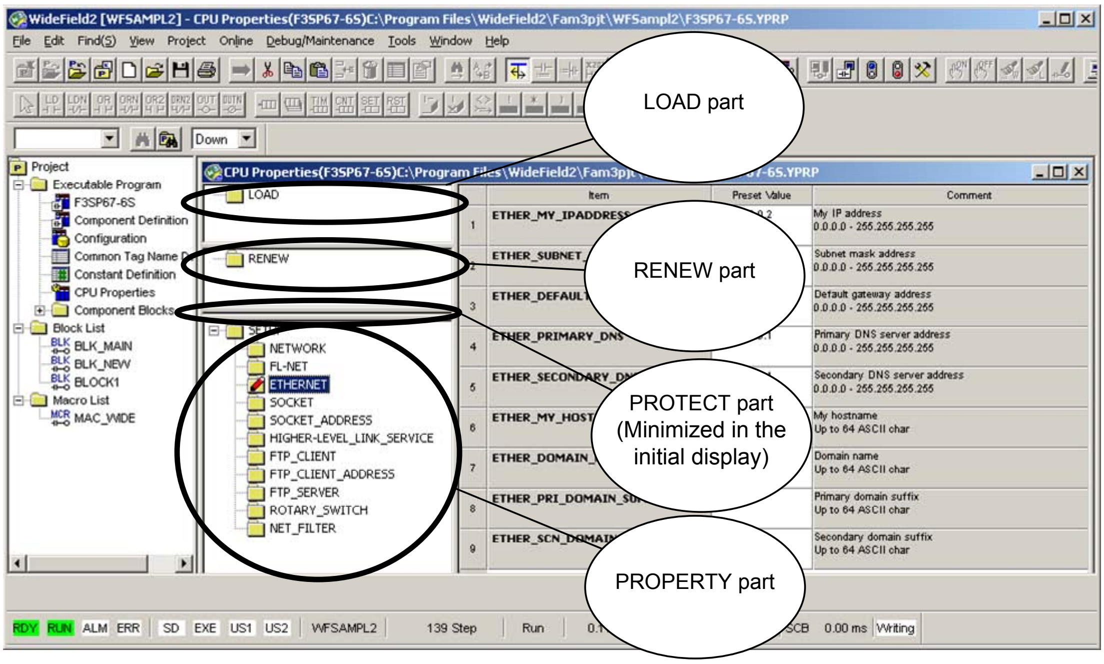
■ Structure of CPU Property File

The CPU property file is divided into four main parts.

Table A9.4.1 Parts of a CPU Property File

Part	Description
LOAD part	The LOAD part can be used to load selected CPU property setups. This is useful for updating some but not all setups.
RENEW part	The RENEW part can be used to immediately apply selected CPU property setups after saving CPU properties. Only some CPU property setups allow immediate application.
PROTECT part	The PROTECT part can be used to set protection of CPU properties. *1
PROPERTY part	This is a collection of various setup items.

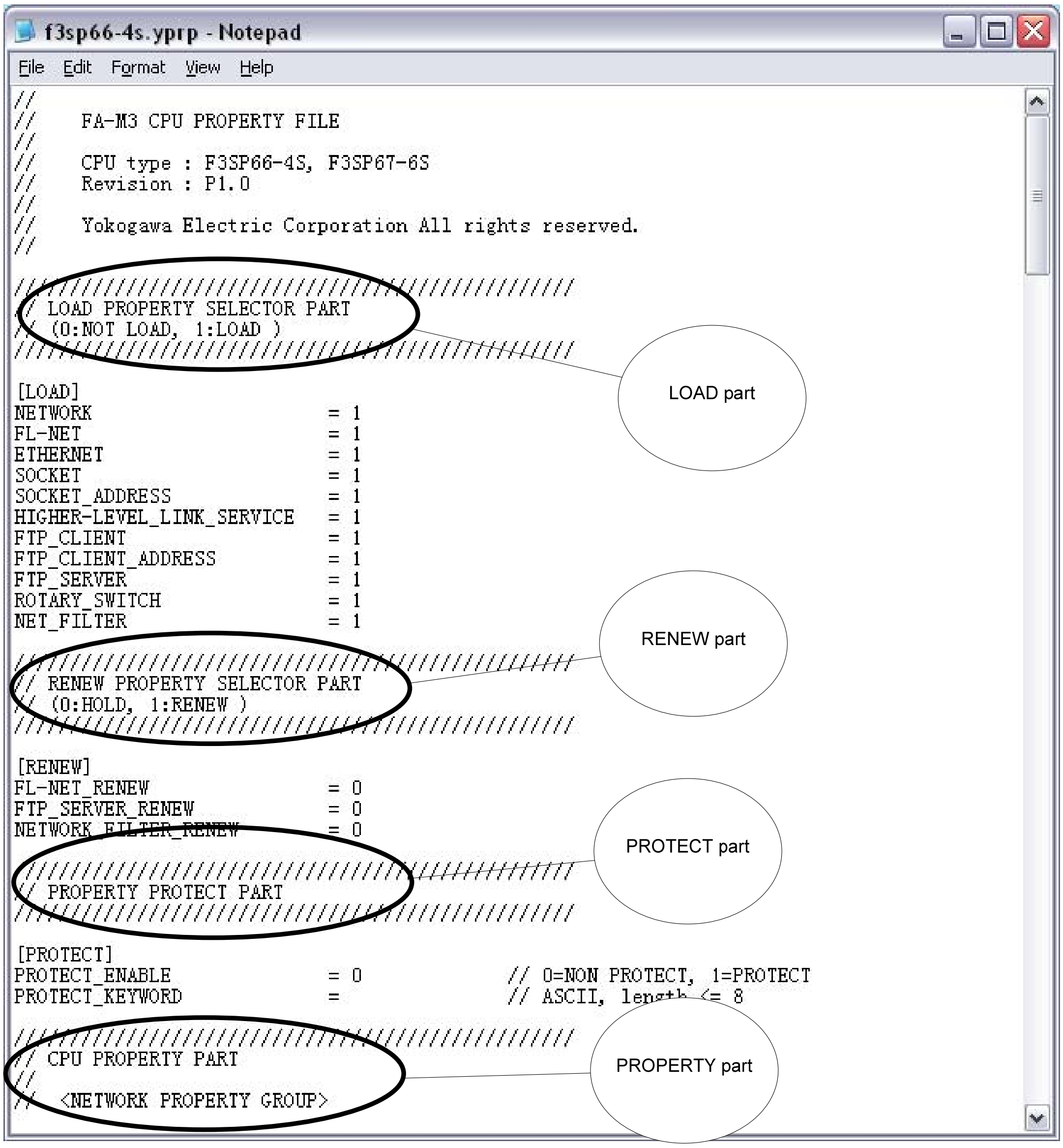
*1: In WideField2, protection for CPU properties can be set and removed using menu operations without using the PROTECT part, which is therefore minimized by default in the CPU properties window of WideField2 (see Figure below). For details on how to set and remove protection of CPU properties in WideField2, see Subsection A6.23.3, "Protection of CPU Properties."



	Item	Preset Value	Comment
1	ETHER_MY_IPADDRESS	0.2	My IP address 0.0.0.0 - 255.255.255.255
2	ETHER_SUBNET_		Subnet mask address 0.0.0.0 - 255.255.255.255
3	ETHER_DEFAULT		Default gateway address 0.0.0.0 - 255.255.255.255
4	ETHER_PRIMARY_DNS	.1	Primary DNS server address 0.0.0.0 - 255.255.255.255
5	ETHER_SECONDARY_DN		Secondary DNS server address 0.0.0.0 - 255.255.255.255
6	ETHER_MY_HOST		My hostname Up to 64 ASCII char
7	ETHER_DOMAIN_		Domain name Up to 64 ASCII char
8	ETHER_PRI_DOMAIN_SU		Primary domain suffix Up to 64 ASCII char
9	ETHER_SCN_DOMAIN		Secondary domain suffix Up to 64 ASCII char

FA0904.VSD

Figure A9.4.1 CPU Properties (as displayed in WideField2)



```
//
//    FA-M3 CPU PROPERTY FILE
//
//    CPU type : F3SP66-4S, F3SP67-6S
//    Revision : P1.0
//
//    Yokogawa Electric Corporation All rights reserved.
//

//////////////////////////////////////////////////
// LOAD PROPERTY SELECTOR PART
// (0:NOT LOAD, 1:LOAD )
//////////////////////////////////////////////////

[LOAD]
NETWORK                     = 1
FL-NET                      = 1
ETHERNET                    = 1
SOCKET                      = 1
SOCKET_ADDRESS              = 1
HIGHER-LEVEL_LINK_SERVICE   = 1
FTP_CLIENT                  = 1
FTP_CLIENT_ADDRESS          = 1
FTP_SERVER                  = 1
ROTARY_SWITCH               = 1
NET_FILTER                  = 1

//////////////////////////////////////////////////
// RENEW PROPERTY SELECTOR PART
// (0:HOLD, 1:RENEW )
//////////////////////////////////////////////////

[RENEW]
FL-NET_RENEW                = 0
FTP_SERVER_RENEW            = 0
NETWORK_FILTER_RENEW        = 0

//////////////////////////////////////////////////
// PROPERTY PROTECT PART
//////////////////////////////////////////////////

[PROTECT]
PROTECT_ENABLE              = 0              // 0=NON PROTECT, 1=PROTECT
PROTECT_KEYWORD             =                // ASCII, length <= 8

//////////////////////////////////////////////////
// CPU PROPERTY PART
//
//   <NETWORK PROPERTY GROUP>
```

FA0905.VSD

Figure A9.4.2   CPU Properties (as displayed in a text editor)

■ LOAD Part (LOAD PROPERTY SELECTOR PART)

● CPU Property File Representation

```
//////////////////////////////////////////////////
// LOAD PROPERTY SELECTOR PART
// (0:NOT LOAD, 1:LOAD )
//////////////////////////////////////////////////

[LOAD]
NETWORK                     = 1
FL-NET                      = 1
ETHERNET                    = 1
SOCKET                      = 1
SOCKET_ADDRESS              = 1
HIGHER-LEVEL_LINK_SERVICE   = 1
FTP_CLIENT                  = 1
FTP_CLIENT_ADDRESS          = 1
FTP_SERVER                  = 1
ROTARY_SWITCH               = 1
NET_FILTER                  = 1
```

● Description

This part of the CPU property file specifies whether to load (store) each CPU property setup of the CPU property file into the CPU module. It is useful for replacing selected CPU property setups in the CPU module with data in a CPU property file. For each CPU property setup, specify 1 to load new setup data to the CPU module; and specify 0 to retain the setup data stored in the CPU module.

Table A9.4.2 Default Values for Setup Items of LOAD Part

Setup Item in LOAD Part (Setup Name)	Corresponding CPU Property Setup Name	Default Value *1
NETWORK	[NETWORK]	1 (=load)
FL-NET *2	[FL-NET] *2	1 (=load)
ETHERNET	[ETHERNET]	1 (=load)
SOCKET	[SOCKET]	1 (=load)
HIGHER-LEVEL_LINK_SERVICE	[HIGHER-LEVEL_LINK_SERVICE]	1 (=load)
FTP_CLIENT	[FTP_CLIENT]	1 (=load)
FTP_CLIENT_ADDRESS	[FTP_CLIENT_ADDRESS]	1 (=load)
FTP_SERVER	[FTP_SERVER]	1 (=load)
ROTARY_SWITCH	[ROTARY_SWITCH]	1 (=load)
NETWORK_FILTER	[NETWORK_FILTER]	1 (=load)

*1: Loading saves edited values to the module.
*2: System reserved. This item is not applicable to the module.

■ RENEW Part (RENEW PROPERTY SELECTOR PART)

● CPU Property File Representation

```
//////////////////////////////////////////////////
// RENEW PROPERTY SELECTOR PART
// ( 0:HOLD, 1:RENEW )
//////////////////////////////////////////////////
[RENEW]
FL-NET_RENEW          = 0
FTP_SERVER_RENEW      = 0
NEWWORK_FILTER_RENEW  = 0
```

● Description

This part of the CPU property file specifies whether to immediately apply CPU property setup stored to the CPU ROM to executing functions. Specify 1 to immediately apply a CPU property setup by automatically restarting the executing function.

The table below shows the list of CPU property setups that allow immediate application, along with their default values.

Table A9.4.3 Corresponding CPU Property Setups and Default Values for Setup Items in RENEW Part

Setup Item in RENEW Part	CPU Property Setup that is Immediately Applied	Default Value
FL-NET_RENEW [*1]	[FL-NET] [*1]	0 (Do not apply)
FTP_SERVER_RENEW	[FTP_SERVER]	0 (Do not apply)
NETWORK_FILTER_RENEW	[NETWORK_FILTER]	0 (Do not apply)

*1: System reserved. This property is not applicable to the module.

The RENEW part contains transient property values that are not stored to the internal ROM of the CPU module. Each time you write new CPU property data, you need to set up the RENEW part as required.

You can also request for immediate application of a CPU property setup by executing a Write CPU Properties (PWRITE) instruction, specifying 100 for instruction parameter n1. Before executing the instruction, set the device designated by instruction parameter "s" to the required setup number as shown in the table below.

Table A9.4.4 CPU Property Instruction Data Format (for NEW Part) [Setup No.=100]

Setup Item	Description	Offset from Specified Device [words]	Size [words]
Setup number of CPU property setup to be applied	9 = FTP server setup 11 = Network filter setup	+0	1

*: The RENEW part is a transient setup so execution of the Read CPU Properties (PREAD) instruction always returns a zero value.

SEE ALSO

For details on when CPU Properties are applied, see "■ When are CPU Properties Applied" of Section A9.5, "CPU Property Items”.

CAUTION

Using the RENEW part may cause a function to restart in order to immediately apply a CPU property setup. For instance, a connection to an FTP server may be disconnected as a result. Before using the RENEW part, ensure that its use will not cause any problems to user applications.

■ PROTECT Part (PROPERTY PROTECT PART)

● CPU Property File Representation

```
//////////////////////////////////////////////////
// PROPERTY PROTECT PART
//////////////////////////////////////////////////

[PROTECT]
PROTECT_ENABLE              = 0      // 0=NON PROTECT, 1=PROTECT
PROTECT_KEYWORD             =        // ASCII, length <= 8
```

● Description

This part of the CPU property file can be used to set protection of CPU properties after they are stored in the CPU module.

The protection specified here applies to loading of CPU properties using the Smart Access function. It does not apply to access via WideField2. Protection of CPU properties against access via WideField2 can be set by specifying a security keyword using the menu of the WideField2 software.

SEE ALSO

For details on how to set and remove protection of CPU properties, see Subsection A6.23.3, "CPU Properties Protection."

Table A9.4.5   Setup Items of PROTECT Part and their Default Values

Setup Item of PROTECT Part (CPU Property Name)	Value Range	Default Value
PROTECT_ENABLE	0: Disabled (No protection) 1: Enabled (Set protection)	0 (Disabled)
PROTECT_KEYWORD	CPU properties security keyword (Up to 8 ASCII characters)*1	–

*1: Valid characters are ‘0’ to ‘9’, ‘a’ to ‘z, ‘A’ to ‘Z’

CAUTION

Apply stringent control to any CPU property file containing a security keyword to avoid inadvertent disclosure as the CPU property file is a readable text file. To avoid such security risks, use WideField2 to set and remove protection of CPU properties instead.

■ SETUP Part (CPU PROPERTY PART)

● CPU Property File Representation

```
//////////////////////////////////////////////////
// CPU PROPERTY PART
//
//  <NETWORK PROPERTY GROUP>
//      [NETWORK]
//
//  <CONTROLLER-LINK PROPERTY GROUP>
//      [FL-NET]
//
//  <ETHERNET PROPERTY GROUP>
//      [ETHERNET]
//
//  <SOCKET PROPERTY GROUP>
//      [SOCKET]
//      [SOCKET_ADDRESS]
//
//  <HIGHER-LEVEL LINK PROPERTY GROUP>
//      [HIGHER-LEVEL_LINK_SERVICE]
//
//  <FTP PROPERTY GROUP>
//      [FTP_CLIENT]
//      [FTP_CLIENT_ADDRESS]
//      [FTP_SERVER]
//
//  <SMART ACCESS PROPERTY GROUP>
//      [ROTARY_SWITCH]
//
//  <SECURITY PROPERTY GROUP>
//      [NETWORK-FILTER]
//
//////////////////////////////////////////////////
<NETWORK PROPERTY GROUP>
[NETWORK]
NETWORK_SELECT              = 1             // 1=ETHER

<CONTROLLER-LINK PROPERTY GROUP>
[FL-NET]
FL-NET_IPADR                = 192.168.250.0 // Reserved
FL-NET_COM1_TOP             = 0             // Reserved
FL-NET_COM1_SIZE            = 0             // Reserved
FL-NET_COM2_TOP             = 0             // Reserved
FL-NET_COM2_SIZE            = 0             // Reserved
FL-NET_TOKEN_TIMEOUT        = 50            // Reserved
```

```
FL-NET_NODE_NAME            = SP6xNo.000    // Reserved
FL-NET_MFT                  = 0             // Reserved

<ETHERNET PROPERTY GROUP>
[ETHERNET]
ETHER_MY_IPADDRESS          = 192.168.0.2   // 0.0.0.0 - 255.255.255.255
ETHER_SUBNET_MASK           = 255.255.255.0 // 0.0.0.0 - 255.255.255.255
ETHER_DEFAULT_GATEWAY       = 192.168.0.1   // 0.0.0.0 - 255.255.255.255
ETHER_PRIMARY_DNS           = 192.168.0.1   // 0.0.0.0 - 255.255.255.255
ETHER_SECONDARY_DNS         = 192.168.0.1   // 0.0.0.0 - 255.255.255.255
ETHER_MY_HOST_NAME          = FAM3          // ASCII, length <= 64byte
ETHER_DOMAIN_NAME           =               // ASCII, length <= 64byte
ETHER_PRI_DOMAIN_SUFIX      =               // ASCII, length <= 64byte
ETHER_SCN_DOMAIN_SUFIX      =               // ASCII, length <= 64byte

<SOCKET PROPERTY GROUP>
[SOCKET]
SOCKET_UDP_BROADCAST        = 0             // 0=DISABLE, 1=ENABLE
KEEPALIVE_TIME              = 30            // 0=DON'T USE, 1 - 65535[sec]

[SOCKET_ADDRESS]
SOCKET_PORT_1               = 0             // 0-65535
SOCKET_ADR_IP_1             =               // 0.0.0.0 - 255.255.255.255, ex.)192.168.0.4
SOCKET_ADR_HOSTNAME_1       =               // ASCII <= 64byte, ex.)Servername
SOCKET_PORT_2               = 0             // 0 - 65535
SOCKET_ADR_IP_2             =               // 0.0.0.0 - 255.255.255.255, ex.)192.168.0.4
SOCKET_ADR_HOSTNAME_2       =               // ASCII <= 64byte, ex.)Servername
SOCKET_PORT_3               = 0             // 0 - 65535
SOCKET_ADR_IP_3             =               // 0.0.0.0 - 255.255.255.255, ex.)192.168.0.4
SOCKET_ADR_HOSTNAME_3       =               // ASCII <= 64byte, ex.)Servername
SOCKET_PORT_4               = 0             // 0 - 65535
SOCKET_ADR_IP_4             =               // 0.0.0.0 - 255.255.255.255, ex.)192.168.0.4
SOCKET_ADR_HOSTNAME_4       =               // ASCII <= 64byte, ex.)Servername
SOCKET_PORT_5               = 0             // 0 - 65535
SOCKET_ADR_IP_5             =               // 0.0.0.0 - 255.255.255.255, ex.)192.168.0.4
SOCKET_ADR_HOSTNAME_5       =               // ASCII <= 64byte, ex.)Servername
SOCKET_PORT_6               = 0             // 0 - 65535
SOCKET_ADR_IP_6             =               // 0.0.0.0 - 255.255.255.255, ex.)192.168.0.4
SOCKET_ADR_HOSTNAME_6       =               // ASCII <= 64byte, ex.)Servername
SOCKET_PORT_7               = 0             // 0 - 65535
SOCKET_ADR_IP_7             =               // 0.0.0.0 - 255.255.255.255, ex.)192.168.0.4
SOCKET_ADR_HOSTNAME_7       =               // ASCII <= 64byte, ex.)Servername
SOCKET_PORT_8               = 0             // 0 - 65535
SOCKET_ADR_IP_8             =               // 0.0.0.0 - 255.255.255.255, ex.)192.168.0.4
SOCKET_ADR_HOSTNAME_8       =               // ASCII <= 64byte, ex.)Servername
SOCKET_PORT_9               = 0             // 0 - 65535
SOCKET_ADR_IP_9             =               // 0.0.0.0 - 255.255.255.255, ex.)192.168.0.4
```

```
SOCKET_ADR_HOSTNAME_9       =               // ASCII <= 64byte, ex.)Servername
SOCKET_PORT_10              = 0             // 0 - 65535
SOCKET_ADR_IP_10            =               // 0.0.0.0 - 255.255.255.255, ex.)192.168.0.4
SOCKET_ADR_HOSTNAME_10      =               // ASCII <= 64byte, ex.)Servername
SOCKET_PORT_11              = 0             // 0 - 65535
SOCKET_ADR_IP_11            =               // 0.0.0.0 - 255.255.255.255, ex.)192.168.0.4
SOCKET_ADR_HOSTNAME_11      =               // ASCII <= 64byte, ex.)Servername
SOCKET_PORT_12              = 0             // 0 - 65535
SOCKET_ADR_IP_12            =               // 0.0.0.0 - 255.255.255.255, ex.)192.168.0.4
SOCKET_ADR_HOSTNAME_12      =               // ASCII <= 64byte, ex.)Servername
SOCKET_PORT_13              = 0             // 0 - 65535
SOCKET_ADR_IP_13            =               // 0.0.0.0 - 255.255.255.255, ex.)192.168.0.4
SOCKET_ADR_HOSTNAME_13      =               // ASCII <= 64byte, ex.)Servername
SOCKET_PORT_14              = 0             // 0 - 65535
SOCKET_ADR_IP_14            =               // 0.0.0.0 - 255.255.255.255, ex.)192.168.0.4
SOCKET_ADR_HOSTNAME_14      =               // ASCII <= 64byte, ex.)Servername
SOCKET_PORT_15              = 0             // 0 - 65535
SOCKET_ADR_IP_15            =               // 0.0.0.0 - 255.255.255.255, ex.)192.168.0.4
SOCKET_ADR_HOSTNAME_15      =               // ASCII <= 64byte, ex.)Servername
SOCKET_PORT_16              = 0             // 0 - 65535
SOCKET_ADR_IP_16            =               // 0.0.0.0 - 255.255.255.255, ex.)192.168.0.4
SOCKET_ADR_HOSTNAME_16      =               // ASCII <= 64byte, ex.)Servername

<HIGHER-LEVEL LINK PROPERTY GROUP>
[HIGHER-LEVEL_LINK_SERVICE]
HLLINK_PROTOCOL_A           = 0             // 0=TCP/IP, 1=UDP/IP
HLLINK_DATA_FORMAT_A        = 0             // 0=ASCII, 1=BINARY
HLLINK_PROTOCOL_B           = 0             // 0=TCP/IP, 1=UDP/IP
HLLINK_DATA_FORMAT_B        = 1             // 0=ASCII, 1=BINARY
HLLINK_PROTECT              = 0             // 0=NON PROTECT 1=PROTECT

<FTP PROPERTY GROUP>
[FTP_CLIENT]
FTPC_NETACK_TOUT            = 60            // 1 - 32767sec, 0=LONGEST (2147483sec)
FTPC_M3ACK_TOUT             = 60            // Reserved
FTPC_GATEWAY_INTERVAL_TOUT  = 3600          // Reserved
FTPC_INUNIT_INFLNET_LOG     = 0             // Reserved

[FTP_CLIENT_ADDRESS]
FTPC_SRV_ACCOUNT_1          = anonymous     // ASCII <= 32byte
FTPC_SRV_PASSWORD_1         = fam3@         // ASCII <= 32byte
FTPC_SRV_PORT_1             = 21            // 1 - 65535
FTPC_SRV_IP_1               = 192.168.0.3
   // 0.0.0.0 - 255.255.255.255, ex.192.168.0.4, For Ethernet connection
FTPC_SRV_HOSTNAME_1         =     // ASCII <= 64byte, ex.)Servername, For Ethernet connection
FTPC_SRV_SEAMLESS_1         =               // Reserved
FTPC_SRV_ACCOUNT_2          = anonymous     // ASCII <= 32byte
```

```
FTPC_SRV_PASSWORD_2         = fam3@         // ASCII <= 32byte
FTPC_SRV_PORT_2             = 21            // 1 - 65535
FTPC_SRV_IP_2               = 192.168.0.3
   // 0.0.0.0 - 255.255.255.255, ex.192.168.0.4, For Ethernet connection
FTPC_SRV_HOSTNAME_2         =     // ASCII <= 64byte, ex.)Servername, For Ethernet connection
FTPC_SRV_SEAMLESS_2         =               // Reserved
FTPC_SRV_ACCOUNT_3          = anonymous     // ASCII <= 32byte
FTPC_SRV_PASSWORD_3         = fam3@         // ASCII <= 32byte
FTPC_SRV_PORT_3             = 21            // 1 - 65535
FTPC_SRV_IP_3               = 192.168.0.3
   // 0.0.0.0 - 255.255.255.255, ex.192.168.0.4, For Ethernet connection
FTPC_SRV_HOSTNAME_3         =     // ASCII <= 64byte, ex.)Servername, For Ethernet connection
FTPC_SRV_SEAMLESS_3         =               // Reserved
FTPC_SRV_ACCOUNT_4          = anonymous     // ASCII <= 32byte
FTPC_SRV_PASSWORD_4         = fam3@         // ASCII <= 32byte
FTPC_SRV_PORT_4             = 21            // 1 - 65535
FTPC_SRV_IP_4               = 192.168.0.3
   // 0.0.0.0 - 255.255.255.255, ex.192.168.0.4, For Ethernet connection
FTPC_SRV_HOSTNAME_4         =     // ASCII <= 64byte, ex.)Servername, For Ethernet connection
FTPC_SRV_SEAMLESS_4         =               // Reserved

[FTP_SERVER]
FTPS_MY_PORT                = 21            // 1 - 65535
FTPS_MAX_CLIENT             = 4             // 1 - 4
FTPS_LOG                    = 1             // 0=DISABLE, 1=ENABLE
FTPS_ANONYMOUS              = 1             // 0=DISABLE, 1=ENABLE
FTPS_INTERVAL_TOUT          = 3600          // 1 - 32767sec, 0=LONGEST (2147483sec)
FTPS_NETACK_TOUT            = 60            // 1 - 32767sec, 0=LONGEST (2147483sec)
FTPS_M3ACK_TOUT             = 60            // Reserved
FTPS_PASSWD                 = fam3@         // ASCII <= 32byte

<SMART ACCESS PROPERTY GROUP>
[ROTARY_SWITCH]
SW_NO_0                     = 1             // 0=DISABLE, 1=ENABLE
SW_NO_1                     = 1             // 0=DISABLE, 1=ENABLE
SW_NO_2                     = 1             // 0=DISABLE, 1=ENABLE
SW_NO_3                     = 1             // 0=DISABLE, 1=ENABLE
SW_NO_4                     = 1             // 0=DISABLE, 1=ENABLE
SW_NO_5                     = 1             // 0=DISABLE, 1=ENABLE
SW_NO_6                     = 1             // 0=DISABLE, 1=ENABLE
SW_NO_7                     = 1             // 0=DISABLE, 1=ENABLE
SW_NO_8                     = 1             // 0=DISABLE, 1=ENABLE
SW_NO_9                     = 1             // 0=DISABLE, 1=ENABLE
SW_NO_A                     = 1             // 0=DISABLE, 1=ENABLE
SW_NO_B                     = 1             // 0=DISABLE, 1=ENABLE
SW_NO_C                     = 1             // 0=DISABLE, 1=ENABLE
SW_NO_D                     = 1             // 0=DISABLE, 1=ENABLE
```

```
SW_NO_E                     = 1             // 0=DISABLE, 1=ENABLE
SW_NO_F                     = 1             // 0=DISABLE, 1=ENABLE

<SECURITY PROPERTY GROUP>
[NET_FILTER]
NET_FILTER_IP_1             =               // 0.0.0.0 - 255.255.255.255
NET_FILTER_HOSTNAME_1       =               // ASCII <= 64byte, ex.)Servername
NET_FILTER_MASK_1           = 0.0.0.0       // 0.0.0.0 - 255.255.255.255, ex.)255.255.255.0
NET_FILTER_IP_2             =               // 0.0.0.0 - 255.255.255.255
NET_FILTER_HOSTNAME_2       =               // ASCII <= 64byte, ex.)Servername
NET_FILTER_MASK_2           = 0.0.0.0       // 0.0.0.0 - 255.255.255.255, ex.)255.255.255.0
NET_FILTER_IP_3             =               // 0.0.0.0 - 255.255.255.255
NET_FILTER_HOSTNAME_3       =               // ASCII <= 64byte, ex.)Servername
NET_FILTER_MASK_3           = 0.0.0.0       // 0.0.0.0 - 255.255.255.255, ex.)255.255.255.0
NET_FILTER_IP_4             =               // 0.0.0.0 - 255.255.255.255
NET_FILTER_HOSTNAME_4       =               // ASCII <= 64byte, ex.)Servername
NET_FILTER_MASK_4           = 0.0.0.0       // 0.0.0.0 - 255.255.255.255, ex.)255.255.255.0
NET_FILTER_IP_5             =               // 0.0.0.0 - 255.255.255.255
NET_FILTER_HOSTNAME_5       =               // ASCII <= 64byte, ex.)Servername
NET_FILTER_MASK_5           = 0.0.0.0       // 0.0.0.0 - 255.255.255.255, ex.)255.255.255.0
NET_FILTER_IP_6             =               // 0.0.0.0 - 255.255.255.255
NET_FILTER_HOSTNAME_6       =               // ASCII <= 64byte, ex.)Servername
NET_FILTER_MASK_6           = 0.0.0.0       // 0.0.0.0 - 255.255.255.255, ex.)255.255.255.0
NET_FILTER_IP_7             =               // 0.0.0.0 - 255.255.255.255
NET_FILTER_HOSTNAME_7       =               // ASCII <= 64byte, ex.)Servername
NET_FILTER_MASK_7           = 0.0.0.0       // 0.0.0.0 - 255.255.255.255, ex.)255.255.255.0
NET_FILTER_IP_8             =               // 0.0.0.0 - 255.255.255.255
NET_FILTER_HOSTNAME_8       =               // ASCII <= 64byte, ex.)Servername
NET_FILTER_MASK_8           = 0.0.0.0       // 0.0.0.0 - 255.255.255.255, ex.)255.255.255.0
NET_FILTER_IP_9             =               // 0.0.0.0 - 255.255.255.255
NET_FILTER_HOSTNAME_9       =               // ASCII <= 64byte, ex.)Servername
NET_FILTER_MASK_9           = 0.0.0.0       // 0.0.0.0 - 255.255.255.255, ex.)255.255.255.0
NET_FILTER_IP_10            =               // 0.0.0.0 - 255.255.255.255
NET_FILTER_HOSTNAME_10      =               // ASCII <= 64byte, ex.)Servername
NET_FILTER_MASK_10          = 0.0.0.0       // 0.0.0.0 - 255.255.255.255, ex.)255.255.255.0
NET_FILTER_IP_11            =               // 0.0.0.0 - 255.255.255.255
NET_FILTER_HOSTNAME_11      =               // ASCII <= 64byte, ex.)Servername
NET_FILTER_MASK_11          = 0.0.0.0       // 0.0.0.0 - 255.255.255.255, ex.)255.255.255.0
NET_FILTER_IP_12            =               // 0.0.0.0 - 255.255.255.255
NET_FILTER_HOSTNAME_12      =               // ASCII <= 64byte, ex.)Servername
NET_FILTER_MASK_12          = 0.0.0.0       // 0.0.0.0 - 255.255.255.255, ex.)255.255.255.0
NET_FILTER_IP_13            =               // 0.0.0.0 - 255.255.255.255
NET_FILTER_HOSTNAME_13      =               // ASCII <= 64byte, ex.)Servername
NET_FILTER_MASK_13          = 0.0.0.0       // 0.0.0.0 - 255.255.255.255, ex.)255.255.255.0
NET_FILTER_IP_14            =               // 0.0.0.0 - 255.255.255.255
NET_FILTER_HOSTNAME_14      =               // ASCII <= 64byte, ex.)Servername
NET_FILTER_MASK_14          = 0.0.0.0       // 0.0.0.0 - 255.255.255.255, ex.)255.255.255.0
```

```
NET_FILTER_IP_15            =               // 0.0.0.0 - 255.255.255.255
NET_FILTER_HOSTNAME_15      =               // ASCII <= 64byte, ex.)Servername
NET_FILTER_MASK_15          = 0.0.0.0       // 0.0.0.0 - 255.255.255.255, ex.)255.255.255.0
NET_FILTER_IP_16            =               // 0.0.0.0 - 255.255.255.255
NET_FILTER_HOSTNAME_16      =               // ASCII <= 64byte, ex.)Servername
NET_FILTER_MASK_16          = 0.0.0.0       // 0.0.0.0 - 255.255.255.255, ex.)255.255.255.0
```

● Description

CPU property items of a setup are coded with one item per line. You can customize a property by editing its value.

Adding an invalid CPU property item or coding an invalid CPU property value in a CPU property file generates an error when the file is loaded. CPU property items that are coded with invalid CPU property values in a loaded CPU property file retains their original property values.

Deleting CPU property items from a CPU property file does not generate an error. CPU property items that are not present in a loaded CPU property file retains their original property values.

TIP

Items coded with a "Reserved" comment in the CPU property file representation shown earlier are system reserved items, which are not used with the module. Do not modify their default values.

SEE ALSO

For details on CPU property setup and CPU property items, see, Section A9.5, "CPU Property Items."

■ Basic Syntax of CPU Property File

● CPU Property Group

A CPU property group name is represented as a string enclosed within angle brackets ("< >"). A CPU property group is a collection of related CPU property setups. CPU property group names are system-defined and cannot be modified by a user.

● CPU Property Setup

A CPU property setup name is represented as a string enclosed within square brackets ("[ ]"). A CPU property setup is a collection of related CPU properties. CPU property setup names are system-defined and cannot be modified by a user.

● CPU Property Name, CPU Property Value

An expression of "A = B" where A is a CPU property name and B is a CPU property value assigns a value to a CPU property. A property name corresponds to a setup item while a property value corresponds to a setup value. CPU property values can be edited. CPU property names are system-defined and cannot be modified by a user.

● Comment

All text on a line following a double slash ("//") is treated as a comment and ignored.

You can add comments to a CPU property file but beware that the module does not save user comments. In other words, all user comments added to a CPU property file will be lost after the file is loaded (downloaded) to the module and then saved (uploaded) again.

● Newline

The CPU property file is designed for one property per line. Multiple properties on one line constitute a syntax error. Therefore, beware of deleting newline codes (CRLF) inadvertently. Furthermore, you must not change the newline code, say, to LF for UNIX operating systems.

● End of File

The module terminates reading of a CPU property file when it encounters the end of file.

A9.5 CPU Property Items

This section describes the CPU property items included in each CPU property setup.

Table A9.5.1 Summary of CPU Properties Setups

Setup	Description	SEE ALSO
Network setup	Select the network type for the 10BASE-T/100BASE-TX connector. The module only supports Ethernet with auto- negotiation.	A9.5.1, "Network Setup"
Ethernet setup	Settings for Ethernet communications (IP address, subnet mask, DNS, gateway, etc.)	A9.5.2, "Ethernet Setup"
Socket setup	Settings for socket communications function (UDP/IP broadcast, TCP/IP keep-alive, etc.)	"■ Socket setup" of A9.5.3 , "Socket Setup"
Socket address setup	Define remote hosts for socket communications function.	"■ Socket Address Setup" of A9.5.3, "Socket Setup"
Higher-level link service setup	Settings for higher-level link service (personal computer link) (port settings, data format, etc.)	A9.5.4, "Higher-level Link Service Setup"
FTP client setup	FTP client timeout interval, etc.	"■ FTP Client Setup" of A9.5.5, "FTP Client Setup"
FTP client address setup	Define destination FTP servers for FTP client	"■ FTP Client Address Setup" of A9.5.5, "FTP Client Setup"
FTP server setup	Settings for FTP server function (password, log, maximum connections, port setting, timeout interval, etc.)	A9.5.6, "FTP Server Setup"
Rotary switch setup	Disable selected rotary switch values.	A9.5.7, "Rotary Switch Setup"
Network filter setup	Specify remote hosts that are allowed to establish connection using socket function, FTP function, etc.	A9.5.8, "Network Filter Setup"

■ When are CPU Properties Applied

Table A9.5.2 When are CPU Properties Applied

Setup	Items	Property Values are Applied:
Network setup	All items	- at power on or reset
Ethernet setup	All items	- at power on or reset
Socket setup	UDP broadcast enable	- upon execution of a related instruction
	TCP/IP keep-alive time	- at power on or reset
Socket address setup	All items	- upon execution of a related instruction
Higher-level link service setup	All items	- at power on or reset
FTP client setup	All items	- when an FTP client is opened
FTP client address setup	All items	- when an FTP client is opened
FTP server setup	All items	- at power on or reset - upon request via CPU Properties RENEW part
Rotary switch setup	All items	- at execution of rotary switch function
Network filter setup	All items	- at power on or reset - upon request via CPU Properties RENEW part

A9.5.1 Network Setup

SEE ALSO

- For details on the specifications of CPU property items included in Network Setup, see Table A9.5.3, "Network Setup."
- For details on the CPU property file representation for Network Setup, see Table A9.5.4, "CPU Property File Representation (for Network Setup)."

Table A9.5.3 Network Setup

Setup Item	Description	Value Range	Default Value
Network select	Select the network type for the 10BASE-T/100BASE-TX connector located on the front panel of the module. The module only supports Ethernet with auto-negotiation.	1 = Ethernet (fixed)	1

Table A9.5.4 CPU Property File Representation (for Network Setup)

Group Name	<NETWORK PROPERTY GROUP>	
Setup Name	[NETWORK]	
Setup Item	Property Name	Property Value Format
Network select	NETWORK_SELECT	“1” (fixed)[*1]

*1: Double-quoted expressions denote ASCII character strings, which should be coded without the double quotation marks in the CPU property file.

A9.5.2 Ethernet Setup

SEE ALSO

- For details on the specifications of CPU property items included in Ethernet Setup, see Table A9.5.5, "Ethernet Setup."
- For details on the CPU property file representation for Ethernet Setup, see Table A9.5.6, "CPU Property File Representation (for Ethernet Setup)."
- For details on the format of device data for Ethernet Setup used in CPU property instructions (PREAD, PWRITE), see Table A9.5.7, "CPU Property Instruction Data Format (for Ethernet Setup)."

Table A9.5.5 Ethernet Setup

Setup Item	Description	Value Range	Default Value
My IP address	Specify the IP address of the module.	0.0.0.0 to 255.255.255.255	192.168.0.2
Subnet mask	Specify the mask to be used for deriving the network address from an IP address. This value must match the network to which the module belongs. If 0.0.0.0 is specified, the default mask corresponding to the class of an IP address is used.	0.0.0.0 to 255.255.255.255	255.255.255.0
Default gateway	Specify the gateway (router etc.) to be used when communicating with other networks. This value must match the system or network to which the module belongs.	0.0.0.0 to 255.255.255.255	192.168.0.1
Primary DNS server	When using hostnames in FTP client instructions or socket instructions, specify the IP address of one primary and one secondary DNS server. (A DNS server is a server that converts hostnames to IP addresses.) The secondary DNS server is automatically used when the primary DNS server is down.	0.0.0.0 to 255.255.255.255	192.168.0.1
Secondary DNS server		0.0.0.0 to 255.255.255.255	192.168.0.1
My hostname	Specify an alphanumeric string as the hostname of the module when using hostnames in FTP client instructions and socket instructions. "My hostname" and "My IP address" defined in the Ethernet setup must be registered with the DNS server. Dynamic DNS is not supported.	Up to 64 ASCII characters	FAM3
Domain name	When using hostnames in FTP client instructions and socket instructions, specify an ASCII character string for the name of the domain to which "My hostname" belongs.	Up to 64 ASCII characters	–
Primary domain suffix	When using hostnames in FTP client instructions and socket instructions, specify the alternative domain suffixes to be used for querying the DNS server when it failed to find the IP address for "My hostname"+"Domain name". "My hostname"+"Primary domain suffix" is used as a first attempt while "My hostname"+"Secondary domain suffix" is used as a second attempt.	Up to 64 ASCII characters	–
Secondary domain suffix		Up to 64 ASCII characters	–

Table A9.5.6 CPU Property File Representation (for Ethernet Setup)

Group Name	<ETHERNET PROPERTY GROUP>	
Setup Name	[ETHERNET]	
Setup Item	Property Name	Property Value Format
My IP address	ETHER_MY_IPADDRESS	“xxx.xxx.xxx.xxx” (xxx = 0 to 255)[*1]
Subnet mask	ETHER_SUBNET_MASK	“xxx.xxx.xxx.xxx” (xxx = 0 to 255)[*1]
Default gateway	ETHER_DEFAULT_GATEWAY	“xxx.xxx.xxx.xxx” (xxx = 0 to 255)[*1]
Primary DNS server	ETHER_PRIMARY_DNS	“xxx.xxx.xxx.xxx” (xxx = 0 to 255)[*1]
Secondary DNS server	ETHER_SECONDARY_DNS	“xxx.xxx.xxx.xxx” (xxx = 0 to 255)[*1]
My hostname	ETHER_MY_HOST_NAME	Up to 64 ASCII characters
Domain name	ETHER_DOMAIN_NAME	Up to 64 ASCII characters
Primary domain suffix	ETHER_PRI_DOMAIN_SUFIX	Up to 64 ASCII characters
Secondary domain suffix	ETHER_SCN_DOMAIN_SUFIX	Up to 64 ASCII characters

*1: Double-quoted expressions denote ASCII character strings, which should be coded without the double quotation marks in the CPU property file.

Table A9.5.7 CPU Property Instruction Data Format (for Ethernet Setup) [Setup No.=3]

Setup Item	Value Range	Offset from Specified Device (words)	Size (words)
My IP address	IP address low (binary)	+0	2
	IP address high (binary)	+1	
Subnet mask	Subnet mask low (binary)	+2	2
	Subnet mask high (binary)	+3	
Default gateway	IP address low (binary)	+4	2
	IP address high (binary)	+5	
Primary DNS server	IP address low (binary)	+6	2
	IP address high (binary)	+7	
Secondary DNS server	IP address low (binary)	+8	2
	IP address high (binary)	+9	
My hostname[*1]	Up to 64 ASCII characters	+10 to 42	33
Domain name[*1]	Up to 64 ASCII characters	+43 to 75	33
Primary domain suffix[*1]	Up to 64 ASCII characters	+76 to 108	33
Secondary domain suffix[*1]	Up to 64 ASCII characters	+109 to 141	33

*1: Append a NULL byte at the end of the string.

A9.5.3 Socket Setup

■ Socket Setup

SEE ALSO

- For details on the specifications of CPU property items included in Socket Setup, see Table A9.5.8, "Socket Setup."
- For details on the CPU property file representation for Socket Setup, see Table A9.5.9, "CPU Property File Representation (for Socket Setup)."
- For details on the format of device data for Socket Setup used in CPU property instructions (PREAD, PWRITE), see Table A9.5.10, "CPU Property Instruction Data Format (for Socket Setup)."

Table A9.5.8 Socket Setup

Setup Item	Description	Value Range	Default Value
UDP broadcast enable	Specify whether to enable or disable sending and receiving of broadcast packets using UDP/IP socket instructions.	0 = Disabled 1 = Enabled	Disabled
TCP/IP keep-alive time	To prevent disconnection by a remote peer due to interval timeout, specify the time interval for sending keep-alive ACK for TCP/IP socket instructions.	0 = Do not send 1 to 65535 (s)	30

Table A9.5.9 CPU Property File Representation (for Socket Setup)

Group Name	<SOCKET PROPERTY GROUP>	
Setup Name	[SOCKET]	
Setup Item	Property Name	Property Value Format
UDP broadcast enable	SOCKET_UDP_BROADCAST	“0” or “1” [*1]
TCP/IP keep-alive time	SOCKET_KEEPALIVE_TIME	“0” to “65535”[*1]

*1: Double-quoted expressions denote ASCII character strings, which should be coded without the double quotation marks in the CPU property file.

Table A9.5.10 CPU Property Instruction Data Format (for Socket Setup) [Setup No.=4]

Setup Item	Value Range	Offset from Specified Device (words)	Size (words)
UDP broadcast enable	0 = Disabled 1 = Enabled	+0	1
TCP/IP keep-alive time [*1]	0 = Do not send 1 to 65535 (s)	+1	1

*1: Set a numeric value compatible with an unsigned word data type. Beware that the value may be displayed as a signed integer in a device monitor.

■ Socket Address Setup

SEE ALSO

- For details on the specifications of CPU property items included in Socket Address Setup, see Table A9.5.11, "Socket Address Setup."
- For details on the CPU property file representation for Socket Address Setup, see Table A9.5.12, "CPU Property File Representation (for Socket Address Setup)."
- For details on the format of device data for Socket Address Setup used in CPU property instructions (PREAD, PWRITE), see Table A9.5.13, "CPU Property Instruction Data Format (for Socket Address Setup)."

Table A9.5.11 Socket Address Setup

Setup Item	Description	Value Range	Default Value
Socket port no. (1)	Specify the port number for a destination no., which can then be used in socket instructions. This item should be specified in tandem with the Socket Address of the destination.	0 to 65535	0
Socket address (1) *1	Specify the address for a destination no., which can then be used in socket instructions. This item should be specified in tandem with the Socket Port No. of the destination.	<IP address> 0.0.0.0 to 255.255.255.255 <Hostname> Up to 64 ASCII characters	–
:*2	:	:	:
Socket port no. (16)	Same as above	Same as above	Same as above
Socket address (16) *1	Same as above	Same as above	Same as above

*1: Specify either the IP address or hostname for a socket. If both are specified, the hostname takes precedence.
*2: The port no. and IP address/hostname for up to 16 sockets can be registered for setting numbers 1 to 16.

Table A9.5.12 CPU Property File Representation (for Socket Address Setup)

Group Name	<SOCKET PROPERTY GROUP>	
Setup Name	[SOCKET_ADDRESS]	
Setup Item	Property Name	Property Value Format
Socket port no. (1)	SOCKET_PORT_1	"0" to "65535"
Socket address (1) *1	SOCKET_ADR_IP_1	"0.0.0.0" to "255.255.255.255"
	SOCKET_ADR_HOSTNAME_1	Up to 64 ASCII characters
: *2	:	:
Socket port no. (16)	SOCKET_PORT_16	"0" to "65535"
Socket address (16) *1	SOCKET_ADR_IP_16	"0.0.0.0" to "255.255.255.255"
	SOCKET_ADR_HOSTNAME_16	Up to 64 ASCII characters

Note: Double-quoted expressions denote ASCII character strings, which should be coded without the double quotation marks in the CPU property file.
*1: Specify either the IP address or hostname for a socket. If both are specified, the hostname takes precedence.
*2: The port no. and IP address/hostname for up to 16 sockets can be registered for setting numbers 1 to 16.

Table A9.5.13 CPU Property Instruction Data Format (for Socket Address Setup) [Setup No.=5]

Setup Item		Value Range	Offset from Specified Device (words)	Size (words)
Socket address setting no.		1 to 16 Specify the target socket address setting no. (n) of CPU properties.	+0	1
Socket port no. (n)		0 to 65535*1	+1	1
Socket address type		0 = IP address 1 = Hostname Select IP address or hostname as the data type for socket address (n).	+2	1
Socket address (n) *2	Socket IP address	Low ($0000 to $FFFF) (C and D of "A.B.C.D" in dotted decimal notation)	+3	1
		High ($0000 to $FFFF) (A and B of "A.B.C.D" in dotted decimal notation)		1
	Socket hostname	Up to 64 ASCII characters Terminated with a NULL byte.		1 to 33

*1: Set a numeric value compatible with an unsigned word data type. Beware that the value may be displayed as a signed integer in a device monitor.
*2: When writing CPU properties, specify either the IP address or hostname, and indicate which is specified in the address type. When reading CPU properties, address data is read according to how it was written.

A9.5.4 Higher-level Link Service Setup

SEE ALSO

- For details on the specifications of CPU property items included in Higher-level Link Service Setup, see Table A9.5.14, "Higher-level Link Service Setup."
- For details on the CPU property file representation for Higher-level Link Service Setup, see Table A9.5.15, "CPU Property File Representation (for Higher-level Link Service Setup)."
- For details on the format of device data for Higher-level Link Service Setup used in CPU property instructions (PREAD, PWRITE), see Table A9.5.16, "CPU Property Instruction Data Format (for Higher-level Link Service Setup)."

Table A9.5.14 Higher-level Link Service Setup

Setup Item	Description	Value Range	Default Value
Higher-level link service port A protocol	Select the protocol to be used for port A for the higher-level link service via Ethernet. The port number for port A is 12289 ($3001).	0 = TCP/IP 1 = UDP/IP	TCP/IP
Higher-level link service Port A command data format	Select the command data format to be used for port A for the higher-level link service via Ethernet.	0 = ASCII 1 = Binary	ASCII
Higher-level link service port B protocol	Select the protocol to be used for port B for the higher-level link service via Ethernet. The port number for port A is 12291 ($3003).	0 = TCP/IP 1 = UDP/IP	TCP/IP
Higher-level link service Port B command data format	Select the command data format to be used for port B for the higher-level link service via Ethernet.	0 = ASCII 1 = Binary	Binary
Write protection	Specify whether to enable write protection to prohibit writing to the module for the higher-level link service via Ethernet.	0 = Disabled 1 = Enabled	Disabled

Table A9.5.15 CPU Property File Representation (for Higher-level Link Service Setup)

Group Name	<HIGHER-LEVEL LINK PROPERTY GROUP>	
Setup Name	[HIGHER-LEVEL_LINK_SERVICE]	
Setup Item	Property Name	Property Value Format
Higher-level link service port A protocol	HLLINK_PROTOCOL_A	“0”, “1”
Higher-level link service Port A command data format	HLLINK_DATA_FORMAT_A	“0”, “1”
Higher-level link service port B protocol	HLLINK_PROTOCOL_B	“0”, “1”
Higher-level link service Port B command data format	HLLINK_DATA_FORMAT_B	“0”, “1”
Write protection	HLLINK_PROTECT	“0”, “1”

Table A9.5.16 CPU Property Instruction Data Format (for Higher-level Link Service Setup) [Setup No.=6]

Setup Item	Value Range	Offset from Specified Device (words)	Size (words)
Higher-level link service port A protocol	0 = TCP/IP 1 = UDP/IP	+0	1
Higher-level link service Port A command data format	0 = ASCII 1 = Binary	+1	1
Higher-level link service port B protocol	0 = TCP/IP 1 = UDP/IP	+2	1
Higher-level link service Port B command data format	0 = ASCII 1 = Binary	+3	1
Write protection	0 = Disabled 1 = Enabled	+4	1

A9.5.5 FTP Client Setup

■ FTP Client Setup

SEE ALSO

- For details on the specifications of CPU property items included in FTP Client Setup, see Table A9.5.17, "FTP Client Setup."
- For details on the CPU property file representation for FTP Client Setup, see Table A9.5.18, "CPU Property File Representation (for FTP Client Setup)."
- For details on the format of device data for FTP Client Setup used in CPU property instructions (PREAD, PWRITE), see Table A9.5.19, "CPU Property Instruction Data Format (for FTP Client Setup)."

Table A9.5.17 FTP Client Setup

Setup Item	Description	Value Range	Default Value
FTP client network timeout	Specify the response timeout interval in seconds for TCP/IP communications (via Ethernet). The FTP client generates an internal communications timeout error (error code -100) if timeout occurs. Automatic adjustment of timeout interval: If the timeout interval specified here is longer than the timeout interval specified in an FTP client instruction parameter, the shorter interval applies.	0 = Longest (2147483 s) 1 to 32767 (s)	60

Table A9.5.18 CPU Property File Representation (for FTP Client Setup)

Group Name	<FTP PROPERTY GROUP>	
Setup Name	[FTP_CLIENT]	
Setup Item	Property Name	Property Value Format
FTP client network timeout	FTPC_NETACK_TOUT	“0“ to “32767“
(Reserved)	FTPC_M3ACK_TOUT	“60“ (fixed)
(Reserved)	FTPC_GATEWAY_INTERVAL_TOUT	“3600” (fixed)
(Reserved)	FTPC_INUNIT_INFLNET_LOG	“0” (fixed)

Note: Double-quoted expressions denote ASCII character strings, which should be coded without the double quotation marks in the CPU property file.

Table A9.5.19 CPU Property Instruction Data Format (for FTP Client Setup) [Setup No.=7]

Setup Item	Value Range	Offset from Specified Device (words)	Size (words)
FTP client network timeout	0 = Longest (2147483 s) 1 to 32767 (s)	+0	1
(Reserved)*1	60 (fixed)	+1	1
(Reserved)*1	3600 (fixed)	+2	1
(Reserved)*1	0 (fixed)	+3	1

*1: Before executing a PWRITE instruction, set the device data for Setup Item indicated as "(Reserved)" in the above table to the constant value specified in the "Value Range" column.

■ FTP Client Address Setup

SEE ALSO

- For details on the specifications of CPU property items included in FTP Client Address Setup, see Table A9.5.20, "FTP Client Address Setup."
- For details on the CPU property file representation for FTP Client Address Setup, see Table A9.5.21, "CPU Property File Representation (for FTP Client Address Setup)."
- For details on the format of device data for FTP Client Address Setup used in CPU property instructions (PREAD, PWRITE), see Table A9.5.22, "CPU Property Instruction Data Format (for FTP Client Address Setup)."

Table A9.5.20 FTP Client Address Setup

Setup Item	Description	Value Range	Default Value
Destination FTP server account (1)	Specify the user name for logging in to the FTP server.	Up to 32 ASCII characters	anonymous
Destination FTP server password (1)	Specify the password for logging in to the FTP server.	Up to 32 ASCII characters	fam3@
Destination FTP server port no. (1)	Specify the port no. of the destination FTP server. This item applies only if the FTP server is connected via Ethernet; the item is ignored otherwise.	1 to 65535	21
Destination FTP server address (1) *1	Specify the IP address or hostname of the destination FTP server.	<IP address> 0.0.0.0 to 255.255.255.255 <hostname> Up to 64 ASCII characters	192.168.0.3
: *2	:	:	:
Destination FTP server account (4)	Same as above	Same as above	Same as above
Destination FTP server password (4)	Same as above	Same as above	Same as above
Destination FTP server port no. (4)	Same as above	Same as above	Same as above
Destination FTP server address (4)	Same as above	Same as above	Same as above

*1: Specify either the IP address or hostname for a destination FTP server. If both are specified, the hostname takes precedence.

*2: Up to 4 destination FTP servers can be registered for setting numbers 1 to 4.

Table A9.5.21 CPU Property File Representation (FTP Client Address Setup)

Group Name	<FTP PROPERTY GROUP>	
Setup Name	[FTP_CLIENT_ADDRESS]	
Setup Item	Property Name	Property Value Format
Destination FTP server account (1)	FTPC_SRV_ACCOUNT_1	Up to 32 ASCII characters
Destination FTP server password (1)	FTPC_SRV_PASSWORD_1	Up to 32 ASCII characters
Destination FTP server port no. (1)	FTPC_SRV_PORT_1	"1" to "65535"
Destination FTP server address (1) [*1]	FTPC_SRV_IP_1	"0.0.0.0" to "255.255.255.255"
	FTPC_SRV_HOSTNAME_1	Up to 64 ASCII characters
	FTPC_SRV_SEAMLESS_1	(Reserved)
: [*2]	:	:
Destination FTP server account (4)	FTPC_SRV_ACCOUNT_4	Same as above
Destination FTP server password (4)	FTPC_SRV_PASSWORD_4	Same as above
Destination FTP server port no. (4)	FTPC_SRV_PORT_4	Same as above
Destination FTP server address (4) [*1]	FTPC_SRV_IP_4	Same as above
	FTPC_SRV_HOSTNAME_4	Same as above
	FTPC_SRV_SEAMLESS_4	Same as above

Note: Double-quoted expressions denote ASCII character strings, which should be coded without the double quotation marks in the CPU property file.

*1: Specify either the IP address or hostname for a destination FTP server. If both are specified, the hostname takes precedence. The FTPC_SRV_SEAMLESS_n (n=1 to 4) items are reserved and not used with the module.

*2: Up to 4 destination FTP servers can be registered for setting numbers 1 to 4.

Table A9.5.22 CPU Property Instruction Data Format (for FTP Client Address Setup) [Setup No.=8]

Setup Item		Value Range	Offset from Specified Device (words)	Size (words)
Destination FTP server setting no.		1 to 4 Specify the target destination FTP server setting no. (n) of CPU properties.	+0	1
Destination FTP server account (n)		Up to 32 ASCII characters Terminated with a NULL byte.	+1	17
Destination FTP server password (n)		Up to 32 ASCII characters Terminated with a NULL byte.	+18	17
Destination FTP server port no. (n)		1 to 65535	+35	1
Destination FTP server address type		0 = IP address 1 = Hostname 2 = (Reserved) Select IP address or hostname as the data type for the destination FTP server address setting (n).	+36	1
Destination FTP server address (n) [*2]	IP address	Low ($0000 to $FFFF) (C and D of "A.B.C.D" in dotted decimal notation)	+37	1
		High ($0000 to $FFFF) (A and B of "A.B.C.D" in dotted decimal notation)		1
	Hostname	Up to 64 ASCII characters Terminated with a NULL byte.[*3]		1 to 33

*1: Specify a numeric value compatible with an unsigned word data type. Beware that the value may be displayed as a signed integer in a device monitor.

*2: When writing CPU properties, specify either the IP address or hostname, and indicate which is specified in the address type. When reading CPU properties, address data is read according to how it was written.

*3: When specifying a hostname, always append NULL byte at the end of the character string.

A9.5.6 FTP Server Setup

SEE ALSO

- For details on the specifications of CPU property items included in FTP Server Setup, see Table A9.5.23, "FTP Server Setup."
- For details on the CPU property file representation for FTP Server Setup, see Table A9.5.24, "CPU Property File Representation (for FTP Server Setup)."
- For details on the format of device data for FTP Server Setup used in CPU property instructions (PREAD, PWRITE), see Table A9.5.25, "CPU Property Instruction Data Format (for FTP Server Setup)."

Table A9.5.23 FTP Server Setup

Setup Item	Description	Value Range	Default Value
FTP server my port no.	Specify the port number of the FTP server of the module.	1 to 65535*1	21
FTP server maximum connections	Specify the maximum number of clients that can be connected concurrently to the FTP server of the module.	1 to 4	4
FTP server log	Specify whether to output to the FTP server log. *2	0 = No 1 = Yes	Yes
Anonymous login enable	Specify whether to allow anonymous login to the FTP server. When anonymous login is enabled, a user can log in successfully using any password by specifying the account name (user name) as "anonymous".	0 = Disabled 1 = Enabled	Enabled
FTP server Interval timeout	Specify the timeout interval in seconds. If the FTP server receives no request from an FTP client within the specified time, it terminates the connection with the FTP client.	0 = Longest (2147483 s) 1 to 32767 (s)	3600
FTP server network timeout	Specify the response timeout interval in seconds for TCP/IP communications (via Ethernet). The FTP server terminates processing of a request from an FTP client if a timeout occurs.	0 = Longest (2147483 s) 1 to 32767 (s)	60
FTP server password	Specify the password for logging in to the FTP server of the module. Specifying NULL disables password checking.	Up to 32 ASCII characters	fam3@

*1: Do not specify my port number as 12289, 12290, 12291, 12305 or 12307 as these numbers are used by the higher-level link service and remote programming service.

*2: For details, see Subsection 3.6.4, "FTP Server Log" of "Sequence CPU – Network Functions User's Manual (for F3SP66-4S, F3SP67-6S)" (IM34M6P14-02E).

Table A9.5.24 CPU Property File Representation (for FTP Server Setup)

Group Name	<FTP PROPERTY GROUP>	
Setup Name	[FTP_SERVER]	
Setup Item	Property Name	Property Value Format
FTP server my port no.	FTPS_MY_PORT	"1" to "65535" *1
FTP server maximum connections	FTPS_MAX_CLIENT	"1" to "4"
FTP server log	FTPS_LOG	"0", "1"
Anonymous login enable	FTPS_ANONYMOUS	"0", "1"
FTP server Interval timeout	FTPS_INTERVAL_TOUT	"0" to "32767"
FTP server network timeout	FTPS_NETACK_TOUT	"0" to "32767"
(Reserved)	FTPS_M3ACK_TOUT	"60" (fixed)
FTP server password	FTPS_PASSWORD	Up to 32 ASCII characters

Note: Double-quoted expressions denote ASCII character strings, which should be coded without the double quotation marks in the CPU property file.

*1: Do not specify my port number as 12289, 12290, 12291, 12305 or 12307 as these numbers are used by the higher-level link service and remote programming service.

Table A9.5.25 CPU Property Instruction Data Format (for FTP Server Setup) [Setup No.=9]

Setup Item	Value Range	Offset from Specified Device (words)	Size (words)
FTP server my port no.	1 to 65535*2	+0	1
FTP server maximum connections	1 to 4	+1	1
FTP server log	0 = No 1 = Yes	+2	1
Anonymous login enable	0 = Disabled 1 = Enabled	+3	1
FTP server Interval timeout	0 = Longest (2147483 s) 1 to 32767 (s)	+4	1
FTP server network timeout	0 = Longest (2147483 s) 1 to 32767 (s)	+5	1
(Reserved)*3	60 (fixed)	+6	1
FTP server password *1	Up to 32 ASCII characters	+7 to 23	17

*1: Append a NULL byte at the end of the character string.

*2: Do not specify my port number as 12289, 12290, 12291, 12305 or 12307 as these numbers are used by the higher-level link service and remote programming service.

*3: Before executing a PWRITE instruction, set the device data for Setup Item indicated as "(Reserved)" in the above table to the constant value specified in the "Value Range" column.

A9.5.7 Rotary Switch Setup

SEE ALSO

- For details on the specifications of CPU property items included in Rotary Switch Setup, see Table A9.5.26, "Rotary Switch Setup."
- For details on the CPU property file representation for Rotary Switch Setup, see Table A9.5.27, "CPU Property File Representation (for Rotary Switch Setup)."
- For details on the format of device data for Rotary Switch Setup used in CPU property instructions (PREAD, PWRITE), see Table A9.5.28, "CPU Property Instruction Data Format (for Rotary Switch Setup)."

Table A9.5.26 Rotary Switch Setup

Setup Item	Description	Value Range	Default Value
MODE switch 0	Specify whether to enable or disable the MODE switch value. When disabled, the press operation and press and hold operation associated with this switch value are disabled. Boot modes, however, cannot be disabled.	0 = Disabled 1 = Enabled	1 (Enabled)
:[*1]	:	:	:
MODE switch F	Same as above	Same as above	Same as above

*1: There is one setting each for MODE switch values $0 to $F.

Table A9.5.27 CPU Property File Representation (for Rotary Switch Setup)

Group Name	<SMART ACCESS PROPERTY GROUP>	
Setup Name	[ROTARY_SWITCH]	
Setup Item	Property Name	Property Value Format
MODE switch 0	SW_NO_0	“0”, “1”[*1]
:[*2]	:	:
MODE switch F	SW_NO_F	“0”, “1”[*1]

*1: Double-quoted expressions denote ASCII character strings, which should be coded without the double quotation marks in the CPU property file.
*2: There is one setting each for MODE switch values $0 to $F.

Table A9.5.28 CPU Property Instruction Data Format (for Rotary Switch Setup) [Setup No.=10]

Setup Item	Value Range	Offset from Specified Device (words)	Size (words)
MODE switch 0	0 = Disabled; 1 = Enabled	+0	Bit 0
MODE switch 1	0 = Disabled; 1 = Enabled		Bit 1
MODE switch 2	0 = Disabled; 1 = Enabled		Bit 2
MODE switch 3	0 = Disabled; 1 = Enabled		Bit 3
MODE switch 4	0 = Disabled; 1 = Enabled		Bit 4
MODE switch 5	0 = Disabled; 1 = Enabled		Bit 5
MODE switch 6	0 = Disabled; 1 = Enabled		Bit 6
MODE switch 7	0 = Disabled; 1 = Enabled		Bit 7
MODE switch 8	0 = Disabled; 1 = Enabled		Bit 8
MODE switch 9	0 = Disabled; 1 = Enabled		Bit 9
MODE switch A	0 = Disabled; 1 = Enabled		Bit 10
MODE switch B	0 = Disabled; 1 = Enabled		Bit 11
MODE switch C	0 = Disabled; 1 = Enabled		Bit 12
MODE switch D	0 = Disabled; 1 = Enabled		Bit 13
MODE switch E	0 = Disabled; 1 = Enabled		Bit 14
MODE switch F	0 = Disabled; 1 = Enabled		Bit 15

Note: Note that each of these 16 setup items uses 1 bit of a 16-bit word data.

A9.5.8 Network Filter Setup

SEE ALSO

- For details on the specifications of CPU property items included in Network Filter Setup, see Table A9.5.29, "Network Filter Setup."
- For details on the CPU property file representation for Network Filter Setup, see Table A9.5.30, "CPU Property File Representation (for Network Filter Setup)."
- For details on the format of device data for Network Filter Setup used in CPU property instructions (PREAD, PWRITE), see Table A9.5.31, "CPU Property Instruction Data Format (for Network Filter Setup)."

Table A9.5.29 Network Filter Setup

Setup Item	Description	Value Range	Default Value
Allowed host (1)	Only remote hosts with their IP addresses or hostname on the Ethernet registered with this setup are allowed to access the module. A subnet mask can be used to grant permission to an entire subnet. The module performs a logical AND of a subnet mask with the registered IP address (or in the case of a hostname, the IP address returned by the DNS), as well as with the IP address of a remote host requesting connection. If the results of the two AND operations are the same, the remote host is allowed to connect. Specify either the IP address or hostname. If both are specified, the hostname takes precedence.	<IP address> 0.0.0.0 to 255.255.255.255 <hostname> Up to 64 ASCII characters <subnet mask> 0.0.0.0 to 255.255.255.255	<IP address> – <hostname> – <subnet mask> 0.0.0.0
:*1	:	:	:
Allowed host (16)	Same as above	Same as above	<IP address> – <hostname> – <subnet mask> 0.0.0.0

Note: For details on the network filter function, see Subsection A6.23.4, "Network Filter Function."
*1: Up to 16 addresses or hostnames can be registered for setting numbers 1 to 16.

Table A9.5.30 CPU Property File Representation (for Network Filter Setup)

Group Name	<SECURITY PROPERTY GROUP>	
Setup Name	[NETWORK_FILTER]	
Setup Item	Property Name	Property Value Format
Allowed host (1)	NET_FILTER_IP_1*2	“0.0.0.0” to “255.255.255.255”
	NET_FILTER_HOSTNAME_1*2	Up to 64 ASCII characters
	NET_FILTER_MASK_1	“0.0.0.0” to “255.255.255.255”
:*1	:	:
Allowed host (16)	NET_FILTER_IP_16*2	“0.0.0.0” to “255.255.255.255”
	NET_FILTER_HOSTNAME_16*2	Up to 64 ASCII characters
	NET_FILTER_MASK_16	“0.0.0.0” to “255.255.255.255”

*1: Up to 16 addresses or hostnames can be registered for setting numbers 1 to 16.
*2: Specify either the IP address or hostname. If both are specified, the hostname takes precedence.

Table A9.5.31 CPU Property Instruction Data Format (for Network Filter Setup) [Setup No.=11]

Setup Item		Value Range	Offset from Specified Device (words)	Size (words)
Allowed host setting no.		1 to 16 Specify the target allowed host setting no. (n) of CPU properties.	+0	1
Allowed host address type		0 = IP address; 1 = hostname Select IP address or hostname as the data type for the "Allowed host" setting. This item is not used when reading CPU properties.	+1	1
Allowed host (n) *1	Subnet mask	Low $0000 to $FFFF	+2	2
		High $0000 to $FFFF		
	IP address	Low $0000 to $FFFF	+4	1
		High $0000 to $FFFF		1
	Hostname	Up to 64 ASCII characters; terminated with a NULL byte.		1 to 33

*1: When writing CPU properties, specify either the IP address or hostname, and indicate which is specified in the address type. When reading CPU properties, address data is read according to how it was written.

A9.6 Editing and Saving CPU Properties

Use the following procedure for editing and saving CPU properties to the module.

1. Get CPU properties.
2. Edit CPU properties.
3. Store CPU properties to the module.

A9.6.1 Getting CPU Property File

There are primarily four means of getting a CPU property file:

- Getting from the module
- Getting from WideField2
- Getting from Yokogawa's FA-M3 website (URL: http://www.fa-m3.com)
- Using a CPU property file saved previously

■ Getting CPU Property File from the Module

You can get a CPU property file using WideField2 or the smart access function.

● Using WideField2

Connect WideField2 and the module. In WideField2, select [Online]–[Upload]–[CPU Properties] from the menu bar to upload CPU property data. You can then save the data to a CPU property file.

SEE ALSO

For details on how to obtain a CPU property file using WideField2, see "FA-M3 Programming Tool WideField2" (IM34M6Q15-01E).

● Using Rotary Switch Function

By operating the MODE switch and SET switch in tandem, you can save a CPU property file in a specific directory (\CARD1\PROJECT) on the memory card CARD1.

SEE ALSO

For details on how to obtain a CPU property file using the rotary switch function, see Chapter B1, "Rotary Switch Functions."

● Using Card Batch File Function

By executing a Save project (SAVE) command from a card batch file, you can generate a CPU property file to a directory specified in the command.

SEE ALSO

For details on how to obtain a CPU property file using the card batch file function, see Chapter B2, "Card Batch File Function."

■ Getting CPU Property File from WideField2

The following two CPU property files are located in a sub-directory named "\CPUPROPERTY" in the installed directory of the WideField2 software. You can copy these property files to a work directory and edit them.

- f3sp66-4s.yprp (for F3SP66-4S)
- f3sp67-6s.yprp (for F3SP67-6S)

■ Getting CPU Property File from Yokogawa's FA-M3 Website (http://www.fa-m3.com)

You can get CPU property files for individual CPU types from the member page on Yokowaga's FA-M3 website (URL: http://www.fa-m3.com). To do so, you must first register as a member.

■ Using a CPU Property File Saved Previously

You can use a CPU property file obtained using any of the methods described above. Before using the file, you must check the CPU type coded at the beginning of the file.

A9.6.2 Editing CPU Properties

A CPU property file can be edited using the WideField2 software or a generic text editor.

● Using WideField2

To edit a CPU property file in WideField2, open the file by selecting [File]-[Open]-[CPU Properties] from the menu bar.

SEE ALSO

For details on how to edit a CPU property file using WIdeFIeld2, see "FA-M3 Programming Tool WideField2" (IM34M6Q15-01E).

● Using Text Editor

You can open and edit a CPU property file using Notepad or any generic text editor.

SEE ALSO

For details on the syntax of a CPU property file, see Section A9.4, "CPU Property File Specifications" and Section A9.5, "CPU Property Items."

A9.6.3 Saving CPU Properties to the Module

You can save edited CPU property data or file to the module using WideField2 or smart access functions.

You can store CPU property data only without affecting project data as CPU properties are independent of any project. Furthermore, you can save CPU property data even in Run mode.

● Using WideField2

From WideField2, connect online to the module, and download the CPU property file to the module.

- Downloading both CPU properties and project

  Select [Online]-[Download]-[Project+CPU Properties] from the menu bar.
- Downloading only CPU properties

  Select [Online]-[Download]-[CPU Properties] from the menu bar, or select [Download] from the right-mouse-click pop-up menu of the CPU Properties edit window.

SEE ALSO

For details on how to store a CPU property file to the module using WideField2, see "FA-M3 Programming Tool WideField2" (IM34M6Q15-01E).

● Using Rotary Switch Function

Save a CPU property file in the "\CARD1\PROJECT" directory of memory card CARD1, and then load the file to the module by operating the MODE switch and SET switch in tandem.

SEE ALSO

For details on how to store a CPU property file to the module using the rotary switch function, see Subsection B1.4.6, "Load Project from CARD1."

● Using Card Batch File Function

To load a CPU property file to the module, first save the file in any directory on memory card CARD1, and then execute a Load Project (LOAD) command from a card batch file, specifying the pathname of the file.

SEE ALSO

For details on how to store a CPU property file to the module using the card batch file function, see Subsection B2.8.2.1 "Load Project (LOAD)."

Blank Page

A9.6.4 Modifying CPU Properties by Program

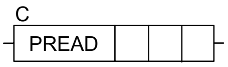
You can modify CPU property values by executing CPU property instructions from a program.

A9.6.4.1 CPU Property Instructions

■ Read CPU Properties (PREAD)

Reads and stores a specified setup of CPU properties to device, starting with a specified first device.

Table A9.6.1 Read CPU Properties

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro-cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	Read CPU Properties	PREAD		✓	–	6	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see "■ Continuous Type Application Instructions" in Section C3.5.1, "Using File System Instructions".

Parameter

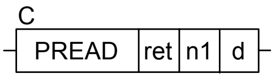
Read CPU Properties 

Table A9.6.2 Parameters

Parameter	Description
ret*1	Device for storing status (W)
n1	Setup no. (W) [1= (Reserved) *2 2= (Reserved) *2 3=Ethernet setup 4=Socket setup 5=Socket address setup 6=Higher-level link service setup 7=FTP client setup 8=FTP client address setup 9=FTP server setup 10=Rotary switch setup 11=Network filter setup]
d	First output device for CPU Properties (W)

*1: ret (status) is table data. For details on the return status (ret), see "Status (Return Value)

*2: Do not specify setup no. indicated as "(reserved)" in the above table.

Table A9.6.3 Text Parameter

Parameter		Description
1	n2	Security keyword string Specify a valid security keyword if CPU properties is protected.

SEE ALSO

Specify text parameters using the Text Parameter (TPARA) instruction. For details on text parameters and the Text Parameter (TPARA) instruction, see "■ Text Parameter" in Section C3.5.1, "Using File System Instructions".

Status (Return Value)

Table A9.6.4 Status (Return Value)

Offset (word)		Description	
ret	ret+0	= 0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see "● Error Status of Continuous Type Application Instructions" of "■ Continuous Type Application Instructions" in Section C3.5.1, "Using File System Instructions."

Available Devices

Table A9.6.5 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	W	Z	R	V	Con-stant	Index Modification	Indirect Designation, Pointer P
ret									✓	✓	✓		✓	✓		Yes	Yes
n1									✓	✓	✓		✓	✓	✓	Yes	Yes
d									✓	✓	✓		✓	✓		Yes	Yes

Note: See Section 1.15, "Restrictions on Devices Used as Instruction Parameters" of "Sequence CPU – Instructions User's Manual" (IM34M6P12-03E).

Resource Relays

None

Function

Reads and stores a specified setup of CPU properties to device, starting from a specified first device.

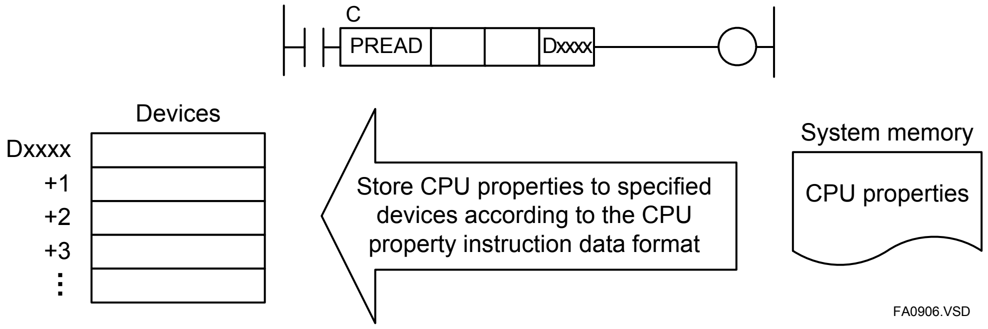

C
PREAD
Dxxxx
Devices
Dxxxx
+1
+2
+3
⋮
Store CPU properties to specified devices according to the CPU property instruction data format
System memory
CPU properties
FA0906.VSD

Figure A9.6.1 Read CPU Properties to Device

The instruction reads current CPU property data in the system memory but not the backup data in the internal ROM. For details on the relationship between CPU property data stored in the system memory and in the internal ROM, see the description for the Write CPU Properties (PWRITE) instruction.

The data read is stored to the output device area designated by the First Output Device for CPU Properties parameter according to the CPU property instruction data format.

SEE ALSO

For details on the format of the returned data, see the Table, "CPU Property Instruction Data Format" for the respective setup in Section A9.5, "CPU Property Items."

Before executing the instruction, you must specify values for the setup items listed in the table below in the output device area according to the CPU Property Instruction Data Format. The required setup items are similar to those of the Write CPU Properties (PWRITE) instruction, except that some setup items do not apply to the PREAD instruction. For instance, a destination can be written as an IP address or a hostname but is always read according to the way it was written.

Table A9.6.6 Required Setup in CPU Property Instruction Data Format

Setup Name [Setup No.]	Required Setup Items	Description
Socket address setup [5]	Socket address setting no.	Specify the target socket address setting no. (n) of CPU properties for reading.
	Socket address type	This item is ignored by the PREAD instruction. Data is read according to how it was written.
FTP client address setup [8]	Destination FTP server setting no.	Specify the target destination FTP server setting no. (n) of CPU properties for reading.
	Destination FTP server address type	This item is ignored by the PREAD instruction. Data is read according to how it was written.
Network filter setup [11]	Allowed host setting no.	Specify the target allowed host setting no. (n) of CPU properties for reading.
	Allowed host address type	This item is ignored by the PREAD instruction. Data is read according to how it was written.

If CPU property data is protected with a security keyword, you must specify a valid security keyword as a text parameter.

CAUTION

No timeout interval can be specified for this instruction, which always waits indefinitely. Furthermore, instruction execution always completes regardless of any cancellation request.

Programming Example

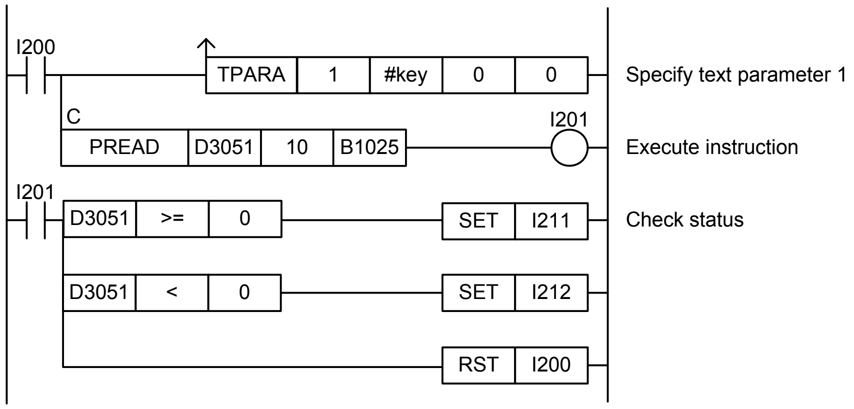


Figure A9.6.2 Read CPU Properties Sample Program

This sample code reads the rotary switch setup of CPU properties data, which is protected. It assumes that the security keyword is defined by constant name #key. The rotary switch setup data is stored to the devices starting from B1025.

The table below shows the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter Name
ret=D3051	0	Status

■ Write CPU Properties (PWRITE)

Writes CPU property values stored in device starting from a specified device to the module.

Table A9.6.7 Write CPU Properties

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro-cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	Write CPU Properties	PWRITE	C ┤PWRITE│ │ │ │├	✓	–	6	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see "■ Continuous Type Application Instructions" in Section C3.5.1, "Using File System Instructions".

Parameter

C
Write CPU Properties ┤PWRITE│ret│n1│s├

Table A9.6.8 Parameters

Parameter	Description
ret*1	Device for storing status (W)
n1	Setup no. (W) [1= (Reserved) *2 2= (Reserved) *2 3=Ethernet setup 4=Socket setup 5=Socket address setup 6=Higher-level link service setup 7=FTP client setup 8=FTP client address setup 9=FTP server setup 10=Rotary switch setup 11=Network filter setup 100=RENEW part 900=Write to internal ROM]
s	First device of CPU property data (W)

*1: ret (status) is table data. For details on the return status (ret), see "Status (Return Value)"
*2: Do not specify setup no. indicated as "(reserved)" in the above table.

Table A9.6.9 Text Parameter

Parameter		Description
1	n2	Security keyword

SEE ALSO

Specify text parameters using the Text Parameter (TPARA) instruction. For details on text parameters and the Text Parameter (TPARA) instruction, see "■ Text Parameter" in Section C3.5.1, "Using File System Instructions".

Status (Return Value)

Table A9.6.10 Status (Return Value)

Offset (word)		Description	
ret	ret+0	= 0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see "● Error Status of Continuous Type Application Instructions" of "■ Continuous Type Application Instructions" in Section C3.5.1, "Using File System Instructions."

Available Devices

Table A9.6.11 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	W	Z	R	V	Con-stant	Index Modification	Indirect Designation, Pointer P
ret									✓	✓	✓		✓	✓		Yes	Yes
n1									✓	✓	✓		✓	✓	✓	Yes	Yes
s									✓	✓	✓		✓	✓		Yes	Yes

Note: See Section 1.15, "Restrictions on Devices Used as Instruction Parameters" of "Sequence CPU – Instructions User's Manual" (IM34M6P12-03E)

Resource Relays

None

Function

This is an instruction for setting CPU property values from a ladder program.

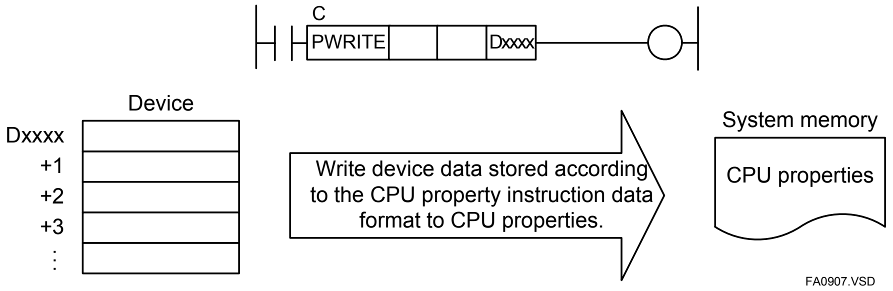

C
PWRITE
Dxxxx
Device
Dxxxx
+1
+2
+3
⋮
Write device data stored according to the CPU property instruction data format to CPU properties.
System memory
CPU properties
FA0907.VSD

Figure A9.6.3 Write CPU Properties

This instruction writes CPU properties on per CPU property setup basis. Before executing the instruction, store the CPU property data in devices according to the required data format. During execution, the data is copied to the system memory. The modified CPU property values in the system memory are then applied to various functions according to the corresponding application timing for each CPU property setup.

SEE ALSO

- For details on the required format for the CPU property data, see Table, "CPU Property Instruction Data Format" for the respective CPU property setup in Section A9.5, "CPU Property Items" and " ■ RENEW Part (RENEW PROPERTY SELECTOR PART)" of Section A9.4, "CPU Property File Specifications."
- For details on when CPU Properties are applied, see "■ When are CPU Properties Applied" of Section A9.5, "CPU Property Items."

Before executing the instruction, you must specify values for the setup items listed in the table below in the CPU property data device area designated by the First Device of CPU Property Data parameter according to the CPU Property Instruction Data Format.

Table A9.6.12 Required Setup in CPU Property Data Device Area

Setup Name [Setup No.]	Required Setup Items	Description
Socket address setup [5]	Socket address setting no.	Specify the target socket address setting no. (n) of CPU properties for writing.
	Socket address type	Select IP address or hostname as the data type for the socket address setting.
FTP client address setup [8]	Destination FTP server setting no.	Specify the target destination FTP server setting no. (n) of CPU properties for writing.
	Destination FTP server address type	Select IP address or hostname as the data type for the destination FTP server address setting.
Network filter setup [11]	Allowed host setting no.	Specify the target allowed host setting no. (n) of CPU properties for writing.
	Allowed host address type	Select IP address or hostname as the data type for the allowed host setting. This item is not used when reading CPU properties.

To ensure that CPU property data copied to the system memory remains valid after power off, you need to write the data to the internal ROM. To do so, execute this instruction, specifying 900 for the setup number parameter. This writes all CPU property data in the system memory to the internal ROM in one go so for faster processing, you can do this after copying CPU property data for multiple setup numbers to the system memory.

If CPU property data is protected with a security keyword, you must specify a valid security keyword as a text parameter.

Multiple concurrent executions of this instruction are not allowed. An error is also generated if this instruction is executed while CPU property data is being written by WideField2 or a smart access function.

CAUTION

- No timeout interval can be specified for this instruction, which always waits indefinitely. Furthermore, instruction execution always completes regardless of any cancellation request.
- Always write CPU property data to the internal ROM eventually. Otherwise, all modifications will be lost after power off and CPU property values will revert to their old values after power on.
- Specifying an out-of-range property value generates a data processing error (error code -9015). When this happens, the invalid property value is not written but all other valid property values are written to system memory. Beware of possible loss of data integrity among property values in this case.

Programming Example

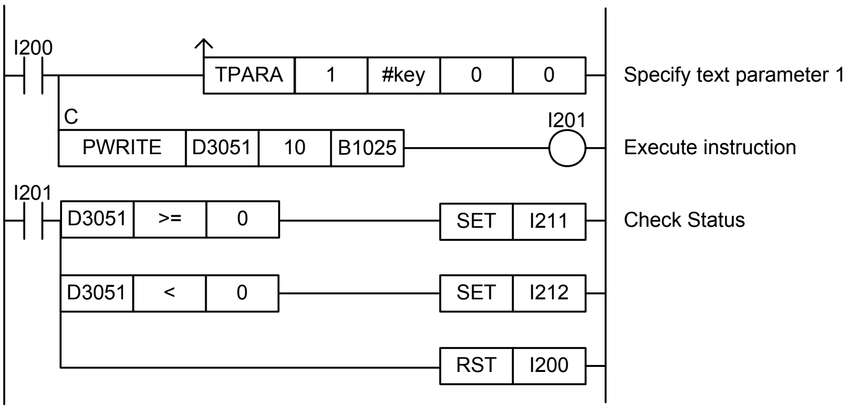


Figure A9.6.4   Write CPU Properties Sample Program

This sample code modifies the rotary switch setup of CPU properties data, which is protected. It assumes that the security keyword is defined by constant name #key, and the new rotary switch setup is stored to device, starting from device B1025.

The table below shows the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter Name
ret=D3051	0	Status

A9.6.5 Restoring Default Values for CPU Properties

This subsection describes how to restore default values for CPU properties.

● Using WideField2

Select [Online]–[Extended Functions]–[Clear CPU Properties] from the menu bar.

● Using smart access function

Execute the Restore Factory Settings smart access function. Beware that this clears other information (e.g. ladder program, device data, log information) together with CPU properties.

SEE ALSO

For details on the Restore Factory Setting smart access function, see Subsection B1.5.4, "Restore Factory Settings”.

A9.7 CPU Property Instruction Sample Program

This section describes a sample program for CPU property instructions. This sample program is intended to help a user better understand the instruction specifications and is not intended to be used directly in user applications.

■ Overview of Sample Program

The table below shows the file structure of the sample program provided for CPU property instructions. Files of the sample program are automatically copied to their respective folders shown below when WideField2 is installed.

Table A9.7.1 Sample Program Components and Location

Sample Program Name	Component	Location
FTP client address setup	Project	~\Fam3pjt\CPUSample\F3SP66\FTPPROP\FTPPROP.YPJT
	CPU properties	Undefined
	Files	None

Note: "~" in the "Location" column denotes the folder where Widefield2 is installed.

A9.7.1 FTP Client Address Setup Example

■ Function and Usage

This sample program writes and subsequently reads FTP client setup of CPU properties. One F3SP66-4S module is required to run the sample program.

The sample program performs the following processing:

1. Initializes devices.
2. Reads destination FTP server settings 1 to 4 of FTP client address setup to device, starting from device B1.
3. Creates setup data in device area according to the required data format for FTP client address setup.
4. Sets destination FTP server settings 1 to 4 of FTP client address setup in the system memory.
5. Writes CPU property data in the system memory to the internal ROM.
6. Reads and stores modified destination FTP server settings 1 to 4 of FTP client address setup into devices starting from device B201.

■ Structure of Sample Program

● List of Instructions Used

The table below lists the main ladder instructions used in the sample program.

Table A9.7.2 List of File Access Instructions Used

Ladder Instruction Mnemonic	Purpose
PREAD	Reads FTP client address setup of CPU properties to device.
PWRITE	Sets FTP client address setup of CPU properties in the system memory. Next, writes the data to the internal ROM.

● List of Special Relays Used

The table below lists the main special relays used in the sample program.

Table A9.7.3 List of Special Relays Used

Name of Special Relay	No. of Special Relay	Function
Always ON	M0033	Used for Always on circuit
1 Scan ON at Program Start	M0035	Turns on for one scan after program starts execution
US1 LED Lit	M0125	Used for turning on US1 LED

Project

The table below shows the content of the WideField2 project containing the sample program.

Table A9.7.4 Project Content

Name	Component	Description	
FTPPROP	Configuration	SP66 configuration with default setup. You can also F3SP67-6S provided you change the CPU type in the configuration.	
	Blocks	Total no. of blocks	1
		Block 1	MAIN
	Macros	Total no. of macros	0
	Constant definition	#USER1	Value to be written to destination FTP server account (1) property
		#PASS1	Value to be written to the destination FTP server password (1) property
		#PORT1	Value to be written to the destination FTP server port no. (1) property
		#TYPE1	Address type for destination FTP server setting 1
		#TARGET1	Value to be written to the destination FTP server IP address (1) property or the destination FTP server hostname (1) property
		#IP	Indicates IP address as address type
		#HOST	Indicates hostname as address type
		#PRPFTPA	Setup number representing FTP client address setup to be specified as a parameter in a CPU property instruction.
		Others, 35 definitions in total	

CPU Properties

This sample program does not define any CPU property. Any CPU property file can be used.

Files

This sample program uses no data file.

■ Ladder Program Listing

The figure on the following pages shows the ladder program listing. For details on the purpose of individual devices used in the ladder program, see the I/O comments of the block tag names.

● Project (FTPPROP) Block (MAIN)

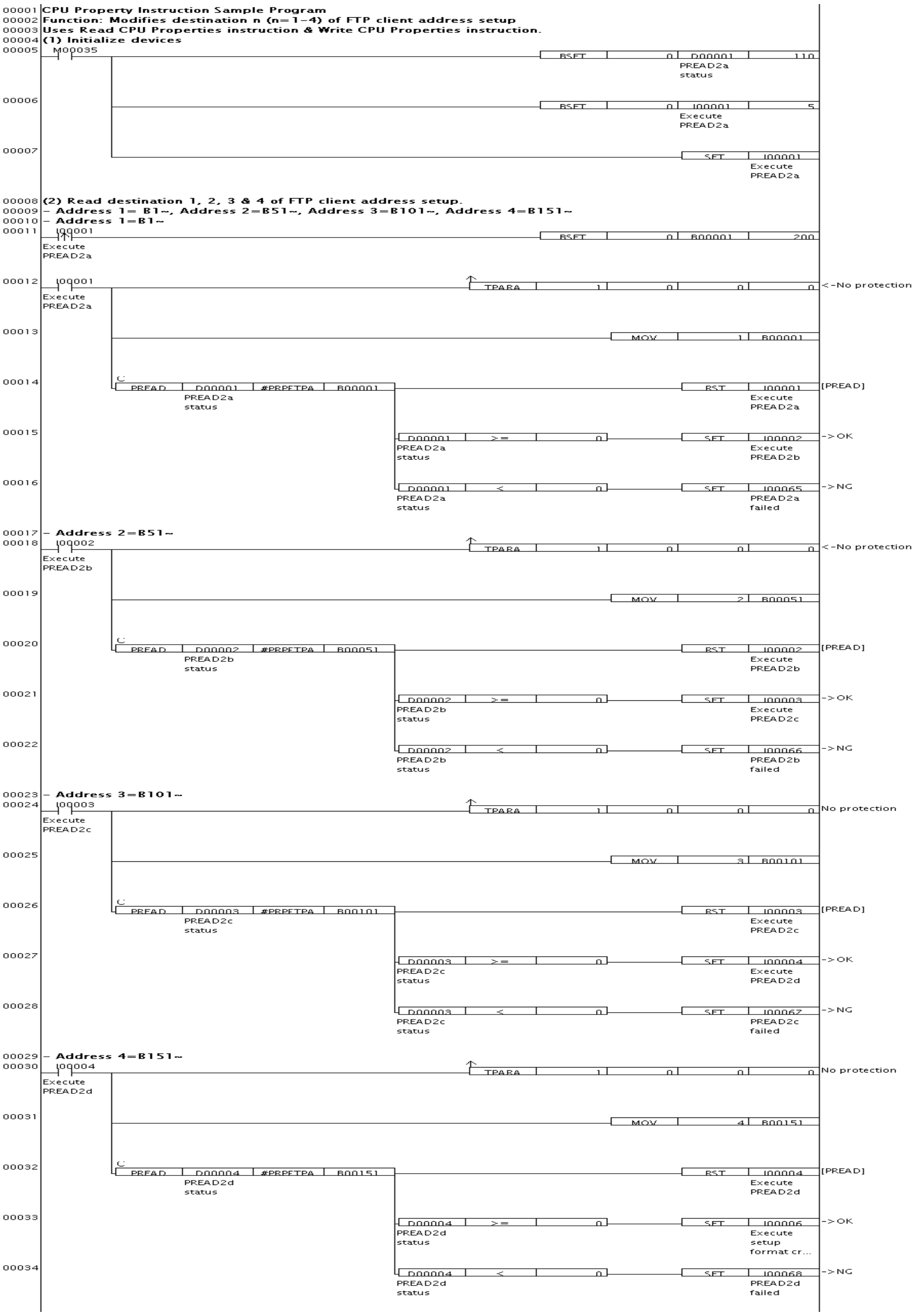
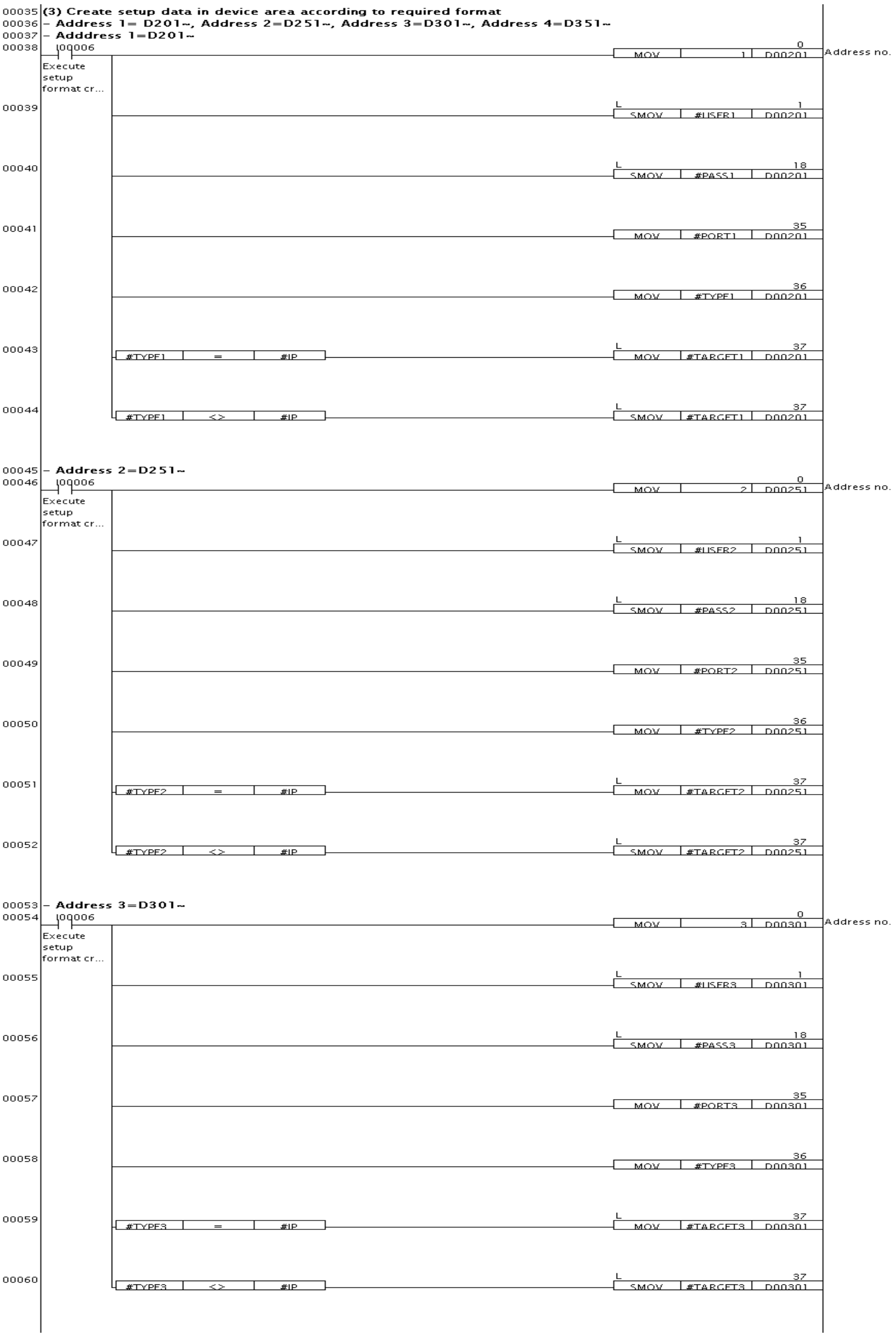
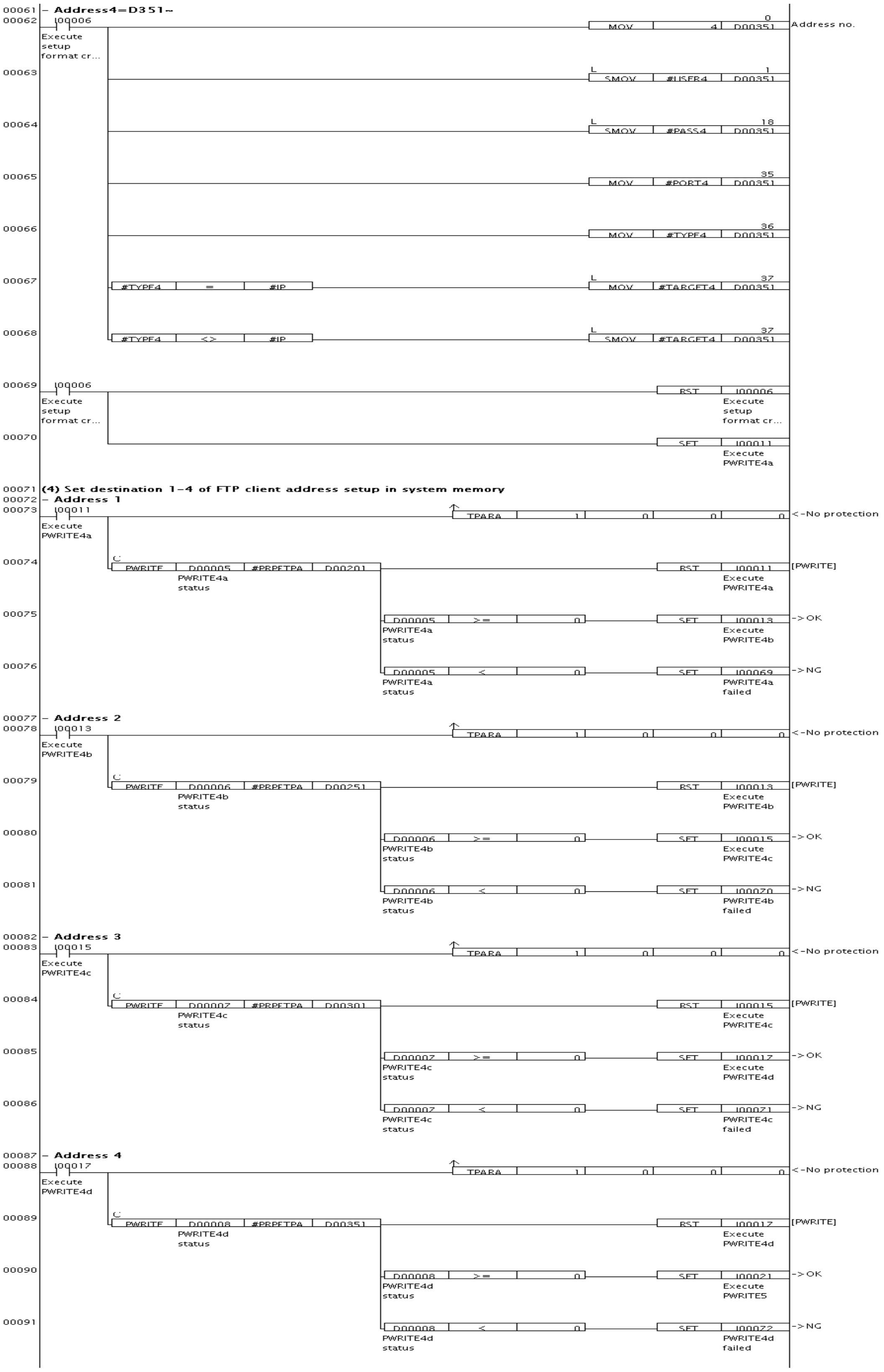

00001 CPU Property Instruction Sample Program
00002 Function: Modifies destination n (n=1-4) of FTP client address setup
00003 Uses Read CPU Properties instruction & Write CPU Properties instruction.
00004 (1) Initialize devices
00005 M00035 BSET 0 D00001 110 PREAD2a status
00006 BSET 0 I00001 5 Execute PREAD2a
00007 SET I00001 Execute PREAD2a
00008 (2) Read destination 1, 2, 3 & 4 of FTP client address setup.
00009 - Address 1= B1~, Address 2=B51~, Address 3=B101~, Address 4=B151~
00010 - Address 1=B1~
00011 I00001 Execute PREAD2a BSET 0 B00001 200
00012 I00001 Execute PREAD2a TPARA 1 0 0 0 <-No protection
00013 MOV 1 B00001
00014 PREAD D00001 PREAD2a status #PRPFTPA B00001 RST I00001 Execute PREAD2a [PREAD]
00015 D00001 PREAD2a status >= 0 SET I00002 Execute PREAD2b ->OK
00016 D00001 PREAD2a status < 0 SET I00065 PREAD2a failed ->NG
00017 - Address 2=B51~
00018 I00002 Execute PREAD2b TPARA 1 0 0 0 <-No protection
00019 MOV 2 B00051
00020 PREAD D00002 PREAD2b status #PRPFTPA B00051 RST I00002 Execute PREAD2b [PREAD]
00021 D00002 PREAD2b status >= 0 SET I00003 Execute PREAD2c ->OK
00022 D00002 PREAD2b status < 0 SET I00066 PREAD2b failed ->NG
00023 - Address 3=B101~
00024 I00003 Execute PREAD2c TPARA 1 0 0 0 No protection
00025 MOV 3 B00101
00026 PREAD D00003 PREAD2c status #PRPFTPA B00101 RST I00003 Execute PREAD2c [PREAD]
00027 D00003 PREAD2c status >= 0 SET I00004 Execute PREAD2d ->OK
00028 D00003 PREAD2c status < 0 SET I00067 PREAD2c failed ->NG
00029 - Address 4=B151~
00030 I00004 Execute PREAD2d TPARA 1 0 0 0 No protection
00031 MOV 4 B00151
00032 PREAD D00004 PREAD2d status #PRPFTPA B00151 RST I00004 Execute PREAD2d [PREAD]
00033 D00004 PREAD2d status >= 0 SET I00006 Execute setup format cr... ->OK
00034 D00004 PREAD2d status < 0 SET I00068 PREAD2d failed ->NG

Figure A9.7.1 CPU Property Instruction Sample Program Listing: MAIN (1/5)


00035 (3) Create setup data in device area according to required format
00036 – Address 1= D201~, Address 2=D251~, Address 3=D301~, Address 4=D351~
00037 – Adddress 1=D201~
00038 I00006 Execute setup format cr... MOV 1 D00201 0 Address no.
00039 L SMOV #USER1 D00201 1
00040 L SMOV #PASS1 D00201 18
00041 MOV #PORT1 D00201 35
00042 MOV #TYPE1 D00201 36
00043 #TYPE1 = #IP L MOV #TARGET1 D00201 37
00044 #TYPE1 <> #IP L SMOV #TARGET1 D00201 37
00045 – Address 2=D251~
00046 I00006 Execute setup format cr... MOV 2 D00251 0 Address no.
00047 L SMOV #USER2 D00251 1
00048 L SMOV #PASS2 D00251 18
00049 MOV #PORT2 D00251 35
00050 MOV #TYPE2 D00251 36
00051 #TYPE2 = #IP L MOV #TARGET2 D00251 37
00052 #TYPE2 <> #IP L SMOV #TARGET2 D00251 37
00053 – Address 3=D301~
00054 I00006 Execute setup format cr... MOV 3 D00301 0 Address no.
00055 L SMOV #USER3 D00301 1
00056 L SMOV #PASS3 D00301 18
00057 MOV #PORT3 D00301 35
00058 MOV #TYPE3 D00301 36
00059 #TYPE3 = #IP L MOV #TARGET3 D00301 37
00060 #TYPE3 <> #IP L SMOV #TARGET3 D00301 37

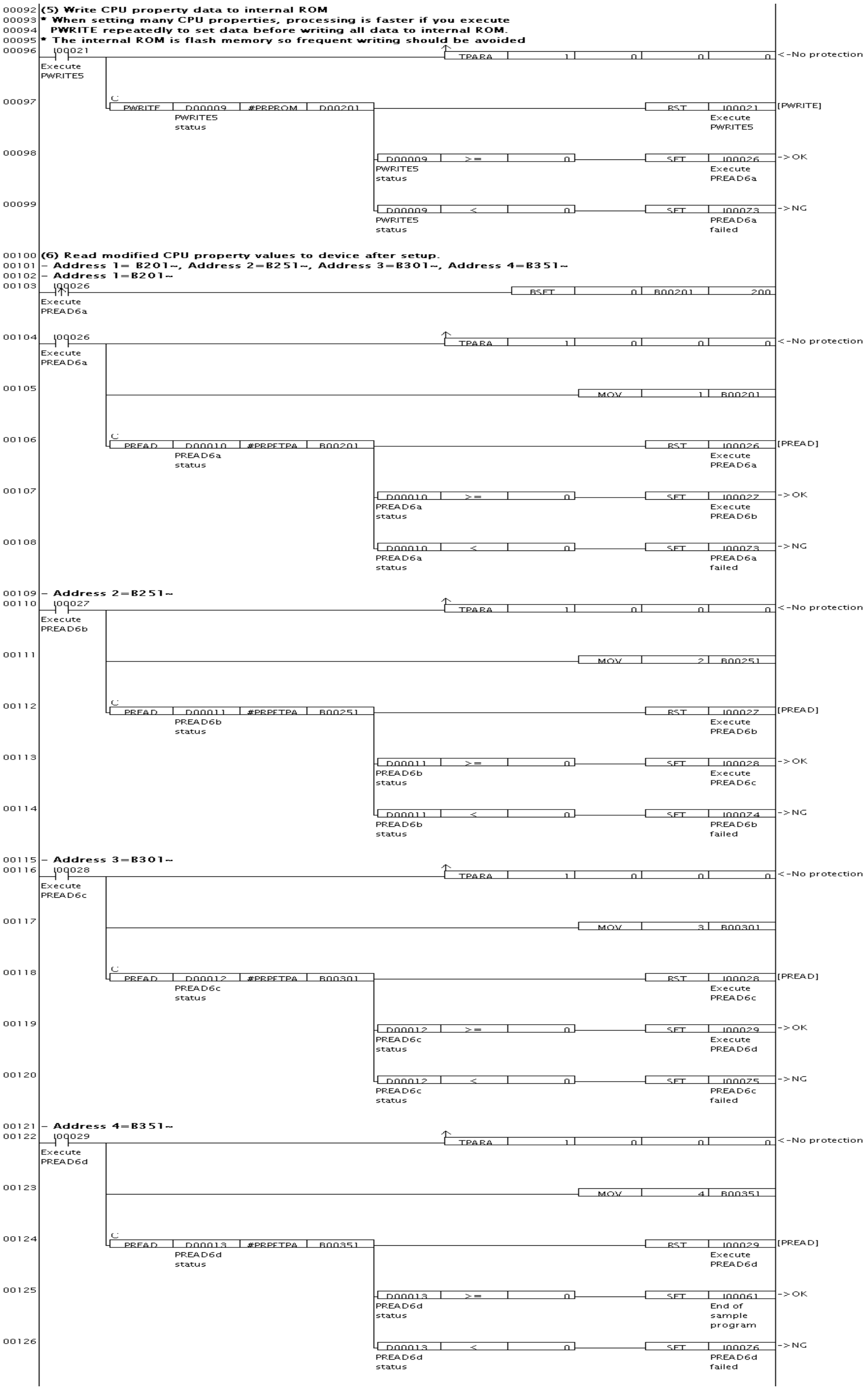
FA0909.VSD

Figure A9.7.2 CPU Property Instruction Sample Program Listing: MAIN (2/5)


00061 - Address4=D351-
00062 I00006 Execute setup format cr... MOV 4 D00351 0 Address no.
00063 L SMOV #USER4 D00351 1
00064 L SMOV #PASS4 D00351 18
00065 MOV #PORT4 D00351 35
00066 MOV #TYPE4 D00351 36
00067 #TYPE4 = #IP L MOV #TARGET4 D00351 37
00068 #TYPE4 <> #IP L SMOV #TARGET4 D00351 37
00069 I00006 Execute setup format cr... RST I00006 Execute setup format cr...
00070 SET I00011 Execute PWRITE4a
00071 (4) Set destination 1-4 of FTP client address setup in system memory
00072 - Address 1
00073 I00011 Execute PWRITE4a TPARA 1 0 0 0 <-No protection
00074 C PWRITE D00005 PWRITE4a status #PRPFTPA D00201 RST I00011 Execute PWRITE4a [PWRITE]
00075 D00005 PWRITE4a status >= 0 SET I00013 Execute PWRITE4b ->OK
00076 D00005 PWRITE4a status < 0 SET I00069 PWRITE4a failed ->NG
00077 - Address 2
00078 I00013 Execute PWRITE4b TPARA 1 0 0 0 <-No protection
00079 C PWRITE D00006 PWRITE4b status #PRPFTPA D00251 RST I00013 Execute PWRITE4b [PWRITE]
00080 D00006 PWRITE4b status >= 0 SET I00015 Execute PWRITE4c ->OK
00081 D00006 PWRITE4b status < 0 SET I00070 PWRITE4b failed ->NG
00082 - Address 3
00083 I00015 Execute PWRITE4c TPARA 1 0 0 0 <-No protection
00084 C PWRITE D00007 PWRITE4c status #PRPFTPA D00301 RST I00015 Execute PWRITE4c [PWRITE]
00085 D00007 PWRITE4c status >= 0 SET I00017 Execute PWRITE4d ->OK
00086 D00007 PWRITE4c status < 0 SET I00071 PWRITE4c failed ->NG
00087 - Address 4
00088 I00017 Execute PWRITE4d TPARA 1 0 0 0 <-No protection
00089 C PWRITE D00008 PWRITE4d status #PRPFTPA D00351 RST I00017 Execute PWRITE4d [PWRITE]
00090 D00008 PWRITE4d status >= 0 SET I00021 Execute PWRITE5 ->OK
00091 D00008 PWRITE4d status < 0 SET I00072 PWRITE4d failed ->NG

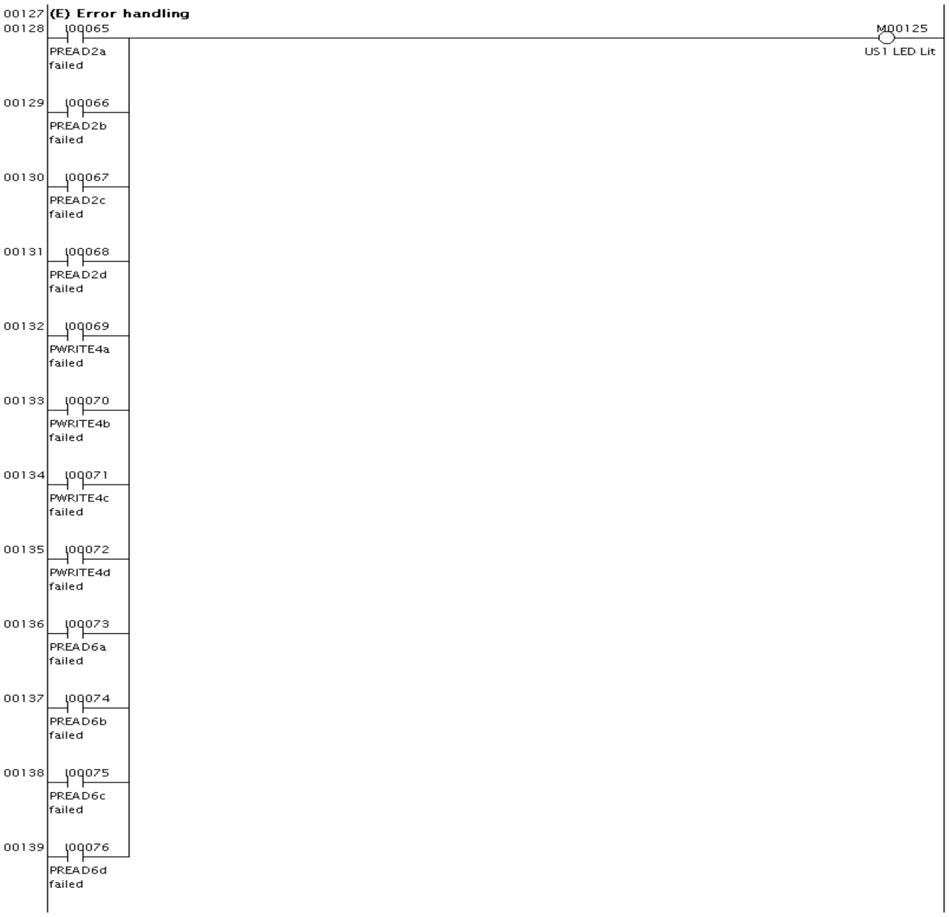
FA0910.VSD

Figure A9.7.3 CPU Property Instruction Sample Program Listing: MAIN (3/5)


00092 (5) Write CPU property data to internal ROM
00093 * When setting many CPU properties, processing is faster if you execute
00094 PWRITE repeatedly to set data before writing all data to internal ROM.
00095 * The internal ROM is flash memory so frequent writing should be avoided
00096 I00021 Execute PWRITE5
TPARA 1 0 0 0
<-No protection
00097 C PWRITE D00009 #PRPROM D00201
PWRITE5 status
RST I00021 Execute PWRITE5
[PWRITE]
00098 D00009 >= 0
PWRITE5 status
SET I00026 Execute PREAD6a
->OK
00099 D00009 < 0
PWRITE5 status
SET I00073 PREAD6a failed
->NG
00100 (6) Read modified CPU property values to device after setup.
00101 – Address 1= B201~, Address 2=B251~, Address 3=B301~, Address 4=B351~
00102 – Address 1=B201~
00103 I00026 Execute PREAD6a
BSET 0 B00201 200
00104 I00026 Execute PREAD6a
TPARA 1 0 0 0
<-No protection
00105 MOV 1 B00201
00106 C PREAD D00010 #PRPETPA B00201
PREAD6a status
RST I00026 Execute PREAD6a
[PREAD]
00107 D00010 >= 0
PREAD6a status
SET I00027 Execute PREAD6b
->OK
00108 D00010 < 0
PREAD6a status
SET I00073 PREAD6a failed
->NG
00109 – Address 2=B251~
00110 I00027 Execute PREAD6b
TPARA 1 0 0 0
<-No protection
00111 MOV 2 B00251
00112 C PREAD D00011 #PRPETPA B00251
PREAD6b status
RST I00027 Execute PREAD6b
[PREAD]
00113 D00011 >= 0
PREAD6b status
SET I00028 Execute PREAD6c
->OK
00114 D00011 < 0
PREAD6b status
SET I00074 PREAD6b failed
->NG
00115 – Address 3=B301~
00116 I00028 Execute PREAD6c
TPARA 1 0 0 0
<-No protection
00117 MOV 3 B00301
00118 C PREAD D00012 #PRPETPA B00301
PREAD6c status
RST I00028 Execute PREAD6c
[PREAD]
00119 D00012 >= 0
PREAD6c status
SET I00029 Execute PREAD6d
->OK
00120 D00012 < 0
PREAD6c status
SET I00075 PREAD6c failed
->NG
00121 – Address 4=B351~
00122 I00029 Execute PREAD6d
TPARA 1 0 0 0
<-No protection
00123 MOV 4 B00351
00124 C PREAD D00013 #PRPETPA B00351
PREAD6d status
RST I00029 Execute PREAD6d
[PREAD]
00125 D00013 >= 0
PREAD6d status
SET I00061 End of sample program
->OK
00126 D00013 < 0
PREAD6d status
SET I00076 PREAD6d failed
->NG

FA0911.VSD

Figure A9.7.4 CPU Property Instruction Sample Program Listing: MAIN (4/5)


00127 (E) Error handling
00128 I00065 PREAD2a failed
M00125 US1 LED Lit
00129 I00066 PREAD2b failed
00130 I00067 PREAD2c failed
00131 I00068 PREAD2d failed
00132 I00069 PWRITE4a failed
00133 I00070 PWRITE4b failed
00134 I00071 PWRITE4c failed
00135 I00072 PWRITE4d failed
00136 I00073 PREAD6a failed
00137 I00074 PREAD6b failed
00138 I00075 PREAD6c failed
00139 I00076 PREAD6d failed

FA0912.VSD

Figure A9.7.5 CPU Property Instruction Sample Program Listing: MAIN (5/5)

A10. Differences between F3SP6□-□S and F3SP5□-□S CPUs

This chapter describes the functional differences between F3SP6□-□S and older F3SP5□-□S CPU types.

TIP

Even for functions which are functionally equivalent between F3SP6□-□S and F3SP5□-□S CPUs, there is no guarantee of equivalent performance. For instance, the performance of shared refreshing, link refreshing and higher-level link service differ between the CPU types.

A10.1 New Functions

This section describes new functions of F3SP6□-□S CPUs, as compared to F3SP5□-□S CPUs.

SEE ALSO

For details on the new functions, see the appropriate chapters or sections of this manual given in the "SEE ALSO" column of each table. For details on some of these functions, see "Sequence CPU – Network Functions User's Manual (for F3SP66-4S, F3SP67-6S)" (IM34M6P14-02E).

A10.1.1 Network Functions and Communications Functions

This subsection describes the new network and communications functions added in F3SP6□-□S.

■ Built-in Ethernet-based Functions of the CPU Module

The F3SP6□-□S is equipped with a 10BASE-T/100BASE-TX connector on its front panel. The following Ethernet-based functions are available with F3SP6□-□S.

Table A10.1.1 Ethernet-based Functions

Function	Description	SEE ALSO
Socket communications	Enables bi-directional socket communications with network equipment using TCP/IP or UDP/IP protocols. Dedicated ladder instructions are provided.	2, "Socket communications Function" *1
FTP client	Enables requests to get (receive) or put (send) files to be sent from the module to an FTP server. Dedicated ladder instructions are provided.	3, "FTP Function" *1
FTP server	Replies to requests to get (send) or put (receive) files, which are sent from FTP clients.	3, "FTP Function" *1
Higher-level link service	Enables reading and writing of device data using personal computer link commands, without the need for ladder programming.	4, "Higher-level Link Service (Personal Computer Link Function)"*1
Remote programming service	Enables connection to WideField2 to download or debug programs.	5., "Remote Programming Service" *1
Network filter function	Restricts the IP addresses and hostnames that are allowed to connect to the module for security purposes.	A6.23.4, "Network Filter Function"
Virtual directory commands	Enables reading and writing of device data, as well as maintenance operations to be carried out without the need for ladder programming by coding dedicated commands in FTP put/get commands.	3.7, "Virtual Directory Commands" *1

*1: For details, see the indicated section or chapter of "Sequence CPU – Network Functions User's Manual" (for F3SP66-4S, F3SP67-6S) (IM34M6P14-02E).

■ Built-in USB-based Functions of the CPU Module

F3SP6□-□S is equipped with USB1.1 connector on its front panel. The following USB-based functions are available with F3SP6□-□S.

Table A10.1.2 USB-based Function

Function	Description	SEE ALSO
Remote programming service	Enables connection to WideField2 for programming or debugging purposes.	5, "Remote Programming Service"*1

*1: For details, see the indicated section or chapter of "Sequence CPU – Network Functions User's Manual" (for F3SP66-4S, F3SP67-6S) (IM34M6P14-02E).

A10.1.2 Storage Functions

F3SP6□-□S is equipped with a card slot, which supports SD memory cards, and a RAM disk, which support high-speed access. It provides various means of accessing these disks via the file system. The table below lists the available disk and file system related functions.

Table A10.1.3 Storage Functions

Function	Description	SEE ALSO
SD memory card slot (CARD1)	The card slot supports the use of non-volatile, portable disks, which can be accessed by ladder programs, FTP and smart access functions.	C1, "Memory Card"
RAM disk	The RAM disk is a volatile, high-speed disk, which can be accessed by ladder programs, FTP and smart access functions.	C2, "RAM Disk"
File system	The module supports the FAT16 file system format. This enables easy file exchange between the module and a PC.	C3, "File System"
Card batch file	Enables the module to be operated using a batch file stored in the memory card.	B2, "Card Batch File Function"
FTP client	Enables files on the memory card and RAM disk to be sent or received via a network.	3, "FTP Function" *1
FTP server	Enables files on the memory card and RAM disk to be sent or received via a network.	3, "FTP Function"*1

*1: For details, see the indicated section or chapter of "Sequence CPU – Network Functions User's Manual" (for F3SP66-4S, F3SP67-6S) (IM34M6P14-02E).

A10.1.3 Smart Access Functions

F3SP6□-□S provides a new interface which allows the CPU module to be operated without the need of WideField2 or a personal computer. This new interface is called the Smart Access Function.

Table A10.1.4 Smart Access Functions

Function	Description	SEE ALSO
Rotary switch functions	Enables the module to be operated simply by turning the rotary switch (MODE switch) and pressing the push button (SET switch) located on the front panel of the module.	B1, "Rotary Switch Functions"
Card batch file function	Enables the module to be operated through execution of dedicated commands coded in a batch file stored on the memory card.	B2, "Card Batch File Function"

A10.1.4 Other Functions

The table below describes the other new functions of the F3SP6□-□S module.

Table A10.1.5 Other Functions

Function	Description	SEE ALSO
CPU properties	These are module settings, which can be modified during operation. It can be edited as a text file, and thus enables module setup without the need of WideField2.	A9, "Setup Description"
Constant definition	Enables names to be assigned to constants. This function is similar to the header file of C and other programming languages.	A6.20, "Constant Definition Function (Header File)"
M3 escape sequence	Enables binary data to be included in character strings. It simplifies creation of transmission messages containing text intermixed with binary codes.	A6.21.1, "M3 Escape Sequence"
Function removal	Enables selected CPU module functions to be disabled for security reasons.	A6.23.5, "Function Removal"

A10.2 Obsoleted Functions

The section describes the functions of F3SP5□-□S, which are removed from F3SP6□-□S.

SEE ALSO

For details on replacement functions for obsoleted functions, see the references given in the "SEE ALSO" column of the following table.

Table A10.2.1 Obsoleted Functions

Function	Description	SEE ALSO
ROM pack	The ROM pack is used for storing ladder programs for backup purposes. It is replaced by the SD memory, which is a new function of the F3SP6□-□S module.	A6.8, "Making Programs and Data Resident in ROM"
WideField2 connection of serial communication port	The F3SP6□-□S module no longer supports WideField2 connection using the serial communication port (called the "Programming tool port" in the F3SP5□-□S modules or "SIO port" in the F3SP6□-□S modules). USB and Ethernet connections are supported instead. In addition, connection to a monitor is still supported.	5, "Remote Programming Service" *1
μ -bus system	This is not supported by the module.	None

*1: For details, see the indicated section or chapter of "Sequence CPU – Network Functions User's Manual" (for F3SP66-4S, F3SP67-6S) (IM34M6P14-02E).

A10.3 New Instructions

F3SP6□-□S provides new groups of instructions to support its new and improved network and storage functions.

SEE ALSO

For details on individual instruction groups, see the reference given in the "SEE ALSO" column of the following table.

Table A10.3.1 New Instruction Groups

Instruction Group	Description	SEE ALSO
File access instructions	A set of instructions for file access operations (reading, writing, etc.).	C3.5.3, "File Access Instructions"
File operation instructions	A set of instructions for file and directory operations (copying, moving, etc.)	C3.5.4, "File Operation Instructions"
Disk operation instructions	A set of instructions for disk operations (unmounting memory card, etc.)	C3.5.5, "Disk Operation Instructions"
UDP/IP communications preparation instructions	A set of instructions for preparation of UDP/IP socket communications.	2.6, "Socket Instructions"*1
UDP/IP send and receive instructions	A set of instructions for sending and receiving using UDP/IP socket communications.	2.6, "Socket Instructions"*1
TCP/IP communications preparation instructions	A set of instructions for preparation of TCP/IP socket communications.	2.6, "Socket Instructions"*1
TCP/IP send and receive instructions	A set of instructions for sending and receiving using TCP/IP socket communications.	2.6, "Socket Instructions"*1
FTP client instructions	A set of instructions for issuing requests to an FTP server to send or receive files when the module is running as an FTP client.	3.4, "FTP Client Instructions"*1
FTP server instructions	A set of instructions for suspending and resuming the FTP server request service.	3.6.5, "FTP Server Instructions"*1
CPU properties instructions	A set of instructions for modifying and reading CPU properties from a ladder program.	A9.6.4, "Modifying CPU Properties by Program"

*1: For details, see the indicated section or chapter of "Sequence CPU – Network Functions User's Manual" (for F3SP66-4S, F3SP67-6S) (IM34M6P14-02E).

A10.4 Changes in Related Equipment

This section describes changes in equipment related to F3SP5□-□S in order to support F3SP6□-□S.

SEE ALSO

For details on the new functions, see the references given in the "SEE ALSO" column of the following table. Where appropriate, see "Sequence CPU – Network Functions User's Manual (for F3SP66-4S, F3SP67-6S)" (IM34M6P14-02E).

Table A10.4.1 Changes in Related Equipment

Product	Description	SEE ALSO
Programming tool cable	The programming tool cable used with the F3SP5□-□S modules cannot be used with the F3SP6□-□S modules as the shape of the connector on the CPU module end has been changed. The product to be used with the F3SP6□-□S module is called the "monitor cable."	4.3, "Personal Computer Link Function via SIO Port"*1

*1: For details, see the indicated section or chapter of "Sequence CPU – Network Functions User's Manual" (for F3SP66-4S, F3SP67-6S) (IM34M6P14-02E).

Blank Page

Appendix A

Blank Page

Appendix A1. Special Relays (M)

Special relays have specific functions, such as indicating the internal state of a sequence CPU module or detecting errors. In programs, these relays are used mainly for contacts a and b.

CAUTION

Do not write to a special relay unless it is marked as "write-enabled". Special relays are used by the sequence CPU module. Writing to these relays incorrectly may lead to system shutdown or other failures. Using forced set/reset instruction in debug mode is also prohibited.

CAUTION

Special relays with index modification cannot be specified as destinations for data output and if specified, will result in instruction processing errors during execution.

CAUTION

Special relays cannot be specified as output destinations in block transfer and table output ladder instructions, and if specified, will cause instruction processing errors during execution.

- Block transfer instructions: BMOV, BSET, SMOV, etc.
- Table output instructions: ULOGR, FIFWR, etc.

Appendix A1.1 Block Start Status

Block Start Status relays indicate which blocks are executed when only specified blocks are executed.
These relays are numbered in ascending order as M001, M002, ... to correlate with block 1, block 2, ...

Table Appendix A1.1 Block Start Status

Item	Block Start Status		
No.	Name	Function	Description
M001 to M032	Block n Start Status	ON : Run OFF: Stop	Indicates whether block n is executed when the module is configured to execute specified blocks only.
M2001 to M3024			

Note: The Start Status relays assigned to blocks 1 to 32 are M0001 to M0032 and M2001 to M2032 (M0001 to M0032 have the same values as M2001 to M2032.) Similarly, Start Status relays M2033 to M3024 map to blocks 33 to 1024.

Appendix A1.2 Utility Relays

Utility relays are used to provide timing in a program or issue instructions to the CPU module.

Table Appendix A1.2 Utility Relays

Item	Utility Relays		
No.	Name	Function	Description
M033	Always ON	ON OFF	Used for initialization or as a dummy contact in a program.
M034	Always OFF	ON OFF	
M035	1 Scan ON at Program Start	1 Scan	Turns on for one scan only after a program starts execution
M036 *1	0.01 s Clock	0.005s 0.005s	Generates a clock pulse of 0.01 s period.
M037 *1	0.02 s Clock	0.01s 0.01s	Generates a clock pulse of 0.02 s period.
M038 *1	0.1 s Clock	0.05s 0.05s	Generates a clock pulse of 0.1 s period.
M039 *1	0.2 s Clock	0.1s 0.1s	Generates a clock pulse of 0.2 s period.
M040 *1	1 s Clock	0.5s 0.5s	Generates a clock pulse of 1 s period.
M041 *1	2 s Clock	1s 1s	Generates a clock pulse of 2 s period.
M042 *1	1 min Clock	30s 30s	Generates a clock pulse of 60 s period.
M047 *1	1 ms Clock	0.5ms 0.5ms	Generates a clock pulse of 1 ms period.
M048 *1	2 ms Clock	1ms 1ms	Generates a clock pulse of 2 ms period.
M066	Normal Subunit Transmission Line	ON : Normal transmission line or no fiber-optic FA-bus installed OFF: Unspecified or abnormal transmission line	
M097	ON for One Scan at Sensor CB Start	ON : At block start OFF: In all other cases	Turns on for one scan when the sensor control block starts (at the first execution of the sensor control block).

*1: Relays M036 to M048 have their rising and falling clock timing synchronized.

SEE ALSO

For details on the M066 Utility relay (Normal Subunit Transmission Line), see "Fiber-optic FA-bus Module and Fiber-optic FA-bus Type 2 Module User's Manual" (IM34M6H45-01E).

Appendix A1.3 Sequence Operation and Mode Status Relays

Sequence operation and mode status relays indicate the status of sequence operation and various modes.

Table Appendix A1.3 Sequence Operation and Mode Status Relays (1/2)

Item	Sequence Operation and Mode Status Relays		
No.	Name	Function	Description
M079	Press Event	Turns on for 1 scan to report a press event	The User Event Press 1 (or 2) function causes this relay to turn on for one scan cycle when a user presses and releases the SET switch with the MODE switch set to $E (or $F). A program may monitor this relay together with special register Z0117 to detect and process a user-defined event. This is a read-only relay.
M080	Press & Hold Event	Turns on for 1 scan to report a press and hold event	The User Event Press 1 (or 2) function causes this relay to turn on for one scan cycle when a user presses and holds the SET switch with the MODE switch set to $E (or $F). A program may monitor this relay together with special register Z0117 to detect and process a user-defined event. This is a read-only relay.
M113	RDY LED	ON : Lit OFF : Off	Indicates whether the RDY LED is lit or off. Read-only.
M115	RUN LED	ON : Lit OFF : Off	Indicates whether the RUN LED is lit or off. Read-only.
M117	ALM LED(1)	ON : Lit OFF : Off	Indicates whether the ALM LED is lit or off. If the LED is blinking, ALM LED (2) is ON. Read-only.
M118	ALM LED(2)	ON : Blinking OFF : Not blinking	
M119	ERR LED(1)	ON : Lit OFF : Off	Indicates whether the ERR LED is lit or off. If the LED is blinking, ERR LED (2) is ON. Read-only.
M120	ERR LED(2)	ON : Blinking OFF : Not blinking	
M121	SD LED(1)	ON : Lit OFF : Off	Indicates whether the SD LED is lit or off. If the LED is blinking, SD LED (2) is ON. Read-only.
M122	SD LED(2)	ON : Blinking OFF : Not blinking	
M123	EXE LED(1)	ON : Lit OFF : Off	Indicates whether the EXE LED is lit or off. If the LED is blinking, EXE LED (2) is ON. Read-only.
M124	EXE LED(2)	ON : Blinking OFF : Not blinking	
M125 (write-enabled)	US1 LED(1)	ON : Lit OFF : Off	Indicates whether the US1 LED is lit or off. If the LED is blinking, US1 LED (2) is ON. You can also manipulate the US1 LED status by writing to this relay.
M126 (write-enabled)	US1 LED(2)	ON : Blinking OFF : Not blinking	
M127 (write-enabled)	US2 LED(1)	ON : Lit OFF : Off	Indicates whether the US2 LED is lit or off. If the LED is blinking, US2 LED (2) is ON. You can also manipulate the US2 LED status by writing to this relay.
M128 (write-enabled)	US2 LED(2)	ON : Blinking OFF : Not blinking	

Table Appendix A1.4 Sequence Operation and Mode Status Relays (2/2)

Item	Sequence Operation and Mode Status Relays		
No.	Name	Function	Description
M129	Run Mode Flag	ON : Run mode OFF: Other modes	Indicates the status of CPU operation.
M130	Debug Mode Flag	ON : Debug mode OFF: Other modes	Indicates the status of CPU operation.
M131	Stop Mode Flag	ON : Stop mode OFF: Other modes	Indicates the status of CPU operation.
M132	Pause Flag	ON : Pause OFF: Run	Indicates the status of program execution during debug mode operation.
M133	Execution Flag	ON : Specified blocks OFF: All blocks	Indicates whether all blocks or specified blocks are executed.
M136	Power-on Operation Flag	ON : Power-on operation OFF: Other modes of operation	Indicates whether operation was initiated by power on or reset
M137	Sensor CB Execution Status	ON : Run OFF: Stop	Indicates the status of sensor control block operation.
M172 (write-enabled)	Set Clock Time	ON : Time being set OFF:	Requests to set clock data.
M173	Input-offline Flag	ON : Offline OFF: Online	Indicates that input refreshing has stopped.
M174	Output-offline Flag	ON : Offline OFF: Online	Indicates that output refreshing has stopped.
M175	Shared-I/O-offline Flag	ON : Offline OFF: Online	Indicates that shared refreshing has stopped.
M176	Link-I/O-offline Flag	ON : Offline OFF: Online	Indicates that link refreshing has stopped.
M188	Carry Flag	ON : Carry enabled OFF: Carry disabled	Carry flag used by shift and rotate operations
M197	Existence of CPU1	ON : Exists. OFF: Does not exist.	Indicates whether or not a CPU exists in slot A1.
M198	Existence of CPU2	ON : Exists. OFF: Does not exist.	Indicates whether or not a CPU exists in slot 2.
M199	Existence of CPU3	ON : Exists. OFF: Does not exist.	Indicates whether or not a CPU exists in slot 3.
M200	Existence of CPU4	ON : Exists. OFF: Does not exist.	Indicates whether or not a CPU exists in slot 4.
M225	CPU1 Sequence Program Execution	ON : Run OFF: Stop	Indicates whether sequence program of CPU in slot 1 is running.
M226	CPU2 Sequence Program Execution	ON : Run OFF: Stop	Indicates whether sequence program of CPU in slot 2 is running.
M227	CPU3 Sequence Program Execution	ON : Run OFF: Stop	Indicates whether sequence program of CPU in slot 3 is running.
M228	CPU4 Sequence Program Execution	ON : Run OFF: Stop	Indicates whether sequence program of CPU in slot 4 is running.
M250	CARD1 Mounted	ON : Mounted OFF: Not mounted	Turns on if memory card CARD1 is mounted. Turns off if otherwise.

SEE ALSO

For details on clock data, see Appendix A2, "Special Registers (Z)".

Appendix A1.4 Self-diagnosis Status Relays

Self-diagnosis status relays indicate the results of self-diagnosis by the sequence CPU.

Table Appendix A1.5 Self-diagnosis Status Relays

Item	Self-diagnosis Status Relays		
No.	Name	Function	Description
M193	Self-diagnosis Error	ON : Error OFF: No error	Result of self diagnosis is stored in special registers Z17 to Z19
M194	Battery Error	ON : Error OFF: Normal	Indicates a failure in backup batteries.
M195	Momentary Power Failure	ON : Momentary power failure OFF: No momentary power failure	Indicates that a momentary power failure has occurred.
M196	Inter-CPU Communication Error	ON : Error OFF: Normal	Indicates that a communication failure has occurred in shared relays (E) or shared registers (R).
M201	Instruction Processing Error	ON : Error OFF: No error	Information of instruction processing error is stored in special registers Z22 to Z24.
M202	I/O Comparison Error	ON : Error OFF: Normal	Indicates that the state of module installation is not consistent with the program.
M203	I/O Module Error	ON : Error OFF: Normal	Indicates that no access is possible to I/O modules. The slot number of the error module is stored in special registers Z33 to Z40.
M204	Scan Timeout	ON : Error OFF: Normal	Indicates that scan time has exceeded the scan monitoring time.
M210	Subunit Communication Error	ON : Error OFF: Unspecified or normal line	An error has been detected in the fiber-optic FA-bus module. The slot number of the error module is stored in special registers Z89 to Z96.
M211	Subunit Line Switchover	ON : Error OFF: Unspecified or normal line	
M212	Sensor CB Scan Timeout	ON : Error OFF: Normal	Indicates that the execution interval of the sensor control block cannot be maintained.

SEE ALSO

For details on the M210 (Subunit Communication Error) and M211 (Subunit Line Switchover) self-diagnosis relays, see "Fiber-optic FA-bus Module and Fiber-optic FA-bus Type 2 Module User's Manual" (IM34M6H45-01E).

Appendix A1.5 FA Link Module Status Relays

FA Link module status relays indicate the status of FA link.

SEE ALSO

For details on FA link module status relays, see the sections on special relays and special registers of “FA Link H Module, Fiber-optic FA Link H Module User's Manual” (IM34M5H43-01E),

Table Appendix A1.6 FA Link Module Status Relays

Item	FA Link Module Status Relays		
No.	Name	Function	Description
M257 to M480 M8321 to M8992	FA Link Error	ON : Error OFF: Normal	Indicates the status of FA links.

Appendix A1.6 FL-net Interface Module Status

FL-net interface module status relays indicate the status of FL-net.

Table Appendix A1.7 FL-net Interface Module Status Relays

Item	FL-net Interface Module Status Relays		
No.	Name	Function	Description
M3521 to M3774	Node Participation Status	1: Participating 0: Not participating	FL-net system 1[*1]
M3777 to M4030	Upper Layer Operation Signal Error	1: Error 0: Normal	FL-net system 1[*1]
M4033 to M4286	Operation Status	1: Run 0: Stop	FL-net system 1[*1]
M4289 to M4542	Common Memory Data Valid	1: Valid 0: Invalid	FL-net system 1[*1]
M4561 to M4814	Node Participation Status	1: Participating 0: Not participating	FL-net system 2[*2]
M4817 to M5070	Upper Layer Operation Signal Error	1: Error 0: Normal	FL-net system 2[*2]
M5073 to M5326	Operation Status	1: Run 0: Stop	FL-net system 2[*2]
M5329 to M5582	Common Memory Data Valid	1: Valid 0: Invalid	FL-net system 2[*2]

*1: If both FL-net and FA link are installed, FL-net are allocated smaller system numbers.
*2: If both FL-net and FA ink are installed, FL-net are allocated larger system numbers.

SEE ALSO

For details, see "FL-net (OPCN-2) Interface Module User's Manual" (IM34M6H32-02E)

TIP

A system refers to a group of units connected to one FL-net.

Appendix A1.7 Continuous Type Application Instruction Resource Relays

These relays indicate the usage of resources of continuous type application instructions.

SEEL ALSO

For details on continuous type application instruction resource relays, see the description of individual continuous type application instructions.

Table Appendix A1.8 Resource Relays (related to file system instructions)

Category	Continuous Type Application Instruction Resource Relays		
No.	Name	Function	Description
M1026	No Unused File ID	No unused file ID is available.	Turns on when all file IDs are in use. This is a read-only relay. Do not write to it.
M1025	File/Disk Operation Group Busy	File operation instruction group or disk operation instruction group is running.	Turns on during execution of any file operation instruction or disk operation instruction. Execution of any other file operation instruction or disk operation instruction is not allowed while this relay is ON. This relay is not affected by file access instructions. This is a read-only relay. Do not write to it.
M1041 to M1056	File ID Open	File ID is open.	Each file ID is associated with one special relay. The relay for a file ID turns on while the file ID is open. When the relay for a file ID is OFF, no instruction using the file ID can be executed. This is a read-only relay. Do not write to it.
M1057 to M1072	File ID Busy	File ID is busy.	Each file ID is associated with one special relay. The relay for a file ID turns on during execution of any file system instruction using the file ID. When the relay for a file ID is ON, no other file system instruction using the same file ID can be executed. This is a read-only relay. Do not write to it.

Table Appendix A1.9 Resource Relays (related to socket instructions)

Category	Continuous Type Application Instruction Resource Relays		
No.	Name	Function	Description
M1028	No Unused UDP Socket	No unused UDP socket is available.	Turns on when all UDP/IP sockets are in use. This is a read-only relay. Do not write to it.
M1029	No Unused TCP Socket	No unused TCP socket is available.	Turns on when all TCP/IP sockets are in use. This is a read-only relay. Do not write to it.
M1105 to M1120	Socket Open	Socket is open.	Each socket ID is associated with one special relay. The relay for a socket ID turns on while the socket ID is open. When the relay for a socket ID is OFF, the socket ID cannot be used. This is a read-only relay. Do not write to it.
M1121 to M1136	Socket Busy	Socket is busy.	Each socket ID is associated with one special relay. The relay for a socket ID turns on during execution of any socket instruction using the socket ID. When the relay for a socket ID is ON, no other socket communication instruction using the same socket ID can be executed except for concurrent execution of sending and receiving. This is a read-only relay. Do not write to it.
M1073 to M1088	Socket Sending	Socket is performing send processing.	Each socket ID is associated with one special relay. The relay for a socket ID turns on during send processing of the socket. When the relay for a socket ID is ON, no send request is allowed for the same socket ID. This is a read-only relay. Do not write to it.
M1089 to M1104	Socket Receiving	Socket is performing receive processing.	Each socket ID is associated with one special relay. The relay for a socket ID turns on during receive processing of the socket. When the relay for a socket ID is ON, no receive request is allowed for the same socket ID. This is a read-only relay. Do not write to it.

Table Appendix A1.10 Resource Relays (related to FTP Client instructions)

Category	Continuous Type Application Instruction Resource Relays		
No.	Name	Function	Description
M1027	FTP Client Busy	An FTP client instruction is being executed.	This relay turns on during execution of any FTP client instruction. When the relay is ON, no other FTP client instruction can be executed. By inserting this relay in the input condition of a FTP client instruction, you can prevent inadvertent duplicate execution. This is a read-only relay. Do not write to it.

Appendix A2. Special Registers (Z)

Special registers have specific functions, such as indicating the internal state of a programmable controller or indicating errors.

Appendix A2.1 Sequence Operation Status Registers

Sequence operation status registers indicate the status of sequence operation.

Table Appendix A2.1 Sequence Operation Status Registers

Category	Sequence Operation Status Registers		
No.	Name	Function	Description
Z001	Scan Time (Run mode)	Latest scan time	Stores the latest scan time in 100 µs increments.
Z002	Minimum Scan Time (Run mode)	Minimum scan time	Allows the latest scan time to be read in 100 µs increments if it is shorter than the minimum scan time.
Z003	Maximum Scan Time (Run mode)	Maximum scan time.	Allows the latest scan time to be read in 100 µs increments if it is longer than the maximum scan time.
Z004	Scan Time (Debug mode)	Latest scan time	Stores the latest scan time in 100 µs increments.
Z005	Minimum Scan Time (Debug mode)	Minimum scan time	Allows the latest scan time to be read in 100 µs increments if it is shorter than the minimum scan time.
Z006	Maximum Scan Time (Debug mode)	Maximum scan time.	Allows the latest scan time to be read in 100 µs increments if it is longer than the maximum scan time.
Z007	Peripheral-process Scan Time	Latest scan time	Stores the latest scan time in 100 µs increments. (Tolerance: Scan time of one control process)
Z008	Minimum Peripheral-process Scan Time	Minimum scan time	Allows the latest scan time to be read in 100 µs increments if it is shorter than the minimum scan time. (Tolerance: Scan time of one control process)
Z009	Maximum Peripheral-process Scan Time	Maximum scan time.	Allows the latest scan time to be read in 100 µs increments if it is longer than the maximum scan time. (Tolerance: Scan time of one control process)

CAUTION

- Do not write to a special register (Z), including those not listed in the table above (e.g., Z010 to Z016), unless it is marked as "write-enabled". Special registers are used by the sequence CPU module. Writing to these registers incorrectly may lead to system shutdown or other failures.
- Special registers (Z) with index modification cannot be specified as destinations for data output and if specified, will cause instruction processing errors during execution.
- Special registers (Z) cannot be specified as output destinations in block transfer and table output ladder instructions, and if specified, will cause instruction processing error during execution.

  Block transfer instructions: BMOV, BSET, SMOV, etc.

  Table output instructions: ULOGR, FIFWR, etc.

Appendix A2.2 Self-diagnosis Status Registers

Self-diagnosis status registers indicate the results of self-diagnostics by the sequence CPU.

Table Appendix A2.2 Self-diagnosis Status Registers

Category	Self-diagnosis Status Registers		
No.	Name	Function	Description
Z017	Self-diagnosis Error	Self-diagnosis error No.	Stores the results of self-diagnosis.*
Z018		Self-diagnosis error block No.	
Z019		Self-diagnosis error instruction No.	
Z022	Instruction Processing Error	Instruction processing error No.	Stores errors detected during instruction processing.*
Z023		Instruction processing error block No.	
Z024		Instruction processing error instruction No.	
Z027	I/O Comparison Error	I/O comparison error No.	Stores detailed information on I/O comparison error.*
Z028		I/O comparison error block No.	
Z029		I/O comparison error instruction No.	
Z033 To Z040	I/O Error	Slot no. with I/O error 16 2 1 0 ····· 1 0	Stores, as a bit pattern, slot numbers where an I/O error is detected. Z033: Main unit Z034: Subunit 1 Z035: Subunit 2 Z036: Subunit 3 Z037: Subunit 4 Z038: Subunit 5 Z039: Subunit 6 Z040: Subunit 7
Z041	Module Recognition	Main unit	Slot number 16 1 0 ····· 1 0 0: No modules are recognized. Unable to read/write. 1: Modules are recognized.
Z042		Subunit 1	
Z043		Subunit 2	
Z044		Subunit 3	
Z045		Subunit 4	
Z046		Subunit 5	
Z047		Subunit 6	
Z048		Subunit 7	
Z089	Communication Error Slot	Main unit	Slot number 16 1 0 ····· 1 0 Fiber-optic FA-bus module 0: Normal transmission line; Unspecified transmission line; or Loaded with a wrong module 1: Abnormal transmission line (Failure or switchover in transmission line)
Z090		Subunit 1	
Z091		Subunit 2	
Z092		Subunit 3	
Z093		Subunit 4	
Z094		Subunit 5	
Z095		Subunit 6	
Z096		Subunit 7	

*: For details on error codes stored in these special registers, see Tables A8.1.3 to A8.1.6,

SEE ALSO

For details on the Z089 to Z096 special registers (Communication error slot), see "Fiber-optic FA-bus Module and Fiber-optic FA-bus Type 2 Module User's Manual" (IM34M6H45-01E).

Appendix A2.3 Utility Registers

Table Appendix A2.3 Utility Registers

Category	Utility Registers		
No.	Name	Function	Description
Z049 (write-enabled)	Clock Data	Last two digits of calendar year	Stores "year" as a BCD-coded value. e.g. 1999 as $0099 2000 as $0000
Z050 (write-enabled)		Month	Stores "month" as a BCD-coded value. e.g. January as $0001
Z051 (write-enabled)		Day of month	Stores "day of month" as a BCD-coded value. e.g. 28th as $0028
Z052 (write-enabled)		Hour	Stores "hour" as a BCD-coded value. e.g. 18:00 hours as $0018
Z053 (write-enabled)		Minute	Stores "minute" as a BCD-coded value. e.g. 15 minutes as $0015
Z054 (write-enabled)		Second	Stores "second" as a BCD-coded value. e.g. 30 seconds as $0030
Z055		Day of week ($0000 to $0006)	Stores "day of week" as a BCD-coded value. e.g. Wednesday as $0003
Z056	Constant Scan Time	Value of constant scan time	0.1 ms increments e.g. 10 ms as 100
Z057	Constant Scan Time	Value of constant scan time	1 ms increments e.g. 10 ms as 10
Z058	Scan Monitoring Time	Value of scan monitoring time	1 ms increments e.g. 200 ms as 200

You can set clock data using the Set Date instruction (DATE), Set Time instruction (DATE), Set Date String instruction (SDATE), and Set Time String instruction (STIME).

- **Procedure for Setting Clock Data without Using Ladder Instructions**

(1) Write the clock data to special registers Z049 to Z054
(use a MOV P instruction. Using BMOV or BSET instructions will generate an instruction error).

(2) Set special relay M172 to ON within the same scan as step (1)
(use a DIFU instruction).

(3) Set special relay M172 to OFF in the scan subsequent to step (2).
Stop writing the clock data to special registers Z049 to Z054 in the same scan.

Note that no change will be made to clock data, which reverts to its original value if the setup value is invalid.

- **Accuracy of Clock Data**

The accuracy of clock data is specified as:
Maximum daily error = ±8 s (±2 s, when actually measured)
The clock accuracy is reset to the maximum daily error of -1.2 s/+2 s, however, when the power is turned off and on again. In addition, you can input a correction value from the programming tool. If you specify an appropriate correction value, the clock data is corrected during the power-off-and-on sequence, thus offsetting the cumulative error.

Appendix A2.4 FA Link Module Status Registers

FA Link module status registers indicate the status of FA links.

SEE ALSO

For details on the FA link module status registers, see Special relays/registers sections in "FA Link H Module Fiber-optic FA Link H Module User's Manual" (IM34M5H43-01E).

Table Appendix A2.4 FA Link Module Status Registers

Category	FA Link Module Status		
No.	Name	Function	Description
Z075	Local Station No.		System 1 (FA link)
Z076	Local Station No.		System 2 (FA link)
Z077	Local Station No.		System 3 (FA link)
Z078	Local Station No.		System 4 (FA link)
Z079	Local Station No.		System 5 (FA link)
Z080	Local Station No.		System 6 (FA link)
Z081	Local Station No.		System 7 (FA link)
Z082	Local Station No.		System 8 (FA link)
Z065	Local Station Status	0: Initialization in progress 1: Offline 2: Online	System 1 (FA link)
Z066	Cyclic Transmission Time		System 1 (FA link) 1 ms increments
Z070	Local Station Status	0: Initialization in progress 1: Offline 2: Online	System 2 (FA link)
Z071	Cyclic Transmission Time		System 2 (FA link) 1 ms increments
Z257	Local Station Status	0: Initialization in progress 1: Offline 2: Online	System 3 (FA link)
Z258	Cyclic Transmission Time		System 3 (FA link) 1 ms increments
Z262	Local Station Status	0: Initialization in progress 1: Offline 2: Online	System 4 (FA link)
Z263	Cyclic Transmission Time		System 4 (FA link) 1 ms increments
Z267	Local Station Status	0: Initialization in progress 1: Offline 2: Online	System 5 (FA link)
Z268	Cyclic Transmission Time		System 5 (FA link) 1 ms increments
Z272	Local Station Status	0: Initialization in progress 1: Offline 2: Online	System 6 (FA link)
Z273	Cyclic Transmission Time		System 6 (FA link) 1 ms increments
Z277	Local Station Status	0: Initialization in progress 1: Offline 2: Online	System 7 (FA link)
Z278	Cyclic Transmission Time		System 7 (FA link) 1 ms increments
Z282	Local Station Status	0: Initialization in progress 1: Offline 2: Online	System 8 (FA link)
Z283	Cyclic Transmission Time		System 8 (FA link) 1 ms increments

TIP

Units that make up a system are known as stations.

Appendix A2.5 Sequence CPU Module Status Registers

CPU module status registers indicate the status of a CPU.

Table Appendix A2.5 Sequence CPU Module Status Registers

Category	Sequence CPU Module Status Registers		
No.	Name	Function	
Z105	Number of User Log Records	See Section A6.14, "User Log Management Function" for details on user logs.	
Z109	Sensor CB Execution Time	Time taken from starting of input refreshing for the sensor control block through program execution to completion of output refreshing. (Unit: 10 µs)	
Z111	Maximum Sensor CB Execution Time	The maximum time taken to execute the sensor control block. (Unit: 10 µs)	
Z113 (write-enabled)	Number of Invalid Accesses	Counts the number of connection requests received from IP addresses that are not registered with the network filter function. The counter is incremented from 0 to 65535, and resets to zero when it exceeds 65535. To reset the counter value, write a zero value to the register, or restore the module to factory settings.	
Z114	MAC Address	MAC address (low word)	Low-order 16 bits [$xxxx]
Z115		MAC address (mid word)	Mid-order 16 bits [$64xx]
Z116		MAC address (high word)	High-order 16 bits [$0000]
Z117	MODE Switch	Stores the MODE switch value. Its value is updated when the SET switch is pressed or pressed and held. Its value does not change within the same scan. Read-only.	
Z121～Z128[*1]	Model Information	CPU model name and revision number of firmware.	

[*1] For module ”F3SP67-6S” with firmware Rev1,

Z121	“F3”
Z122	“SP”
Z123	“67”
Z124	“6S”
Z125	“/R”
Z126	“01”
Z127	“/ “
Z128	“ “

FA-M3
Sequence CPU – Functions (for F3SP66-4S, F3SP67-6S)
PART A Functions

IM 34M6P14-01E 1st Edition

INDEX

FA-M3

Sequence CPU – Functions (for F3SP66-4S, F3SP67-6S)

PART B Smart Access Functions

IM 34M6P14-01E 1st Edition

PART B describes the smart access functions. These functions enable a user to operate the module during debug and maintenance without requiring the use of the WideField2 software.

The table below briefly describes the smart access functions.

Table B.1 List of Smart Access Functions

Smart Access Function	Feature	Overview
Rotary switch function	- Does not require a PC - Lower versatility	Various functions are assigned to the MODE switch. A user can execute these functions by changing the MODE switch value and pressing the SET switch.
Card batch file function	- Does not require a PC - Higher versatility	A user can code card batch commands, which perform CPU operations, in a card batch file and store the file on a memory card to be automatically executed by the module when a trigger event such as a power on or operating mode switchover occurs.

These functions can be disabled in the configuration by a manufacturer or developer to make them inaccessible to end users.

B1. Rotary Switch Functions

Rotary switch functions allow you to operate the module using the MODE switch (rotary switch) and SET switch located on the front panel of the module. For example, you can replace project data or read device data into a memory card easily using rotary switch functions without bringing in a PC to the site or having knowledge of WideField2.

You can also use the MODE switch to specify the boot mode of the module after power on or reset, select either the internal ROM or the memory card as the source for programs, as well as specify the program operating mode after startup. You can also use the MODE switch to invoke user-defined functions. This means that you can create different maintenance modes for different user devices.

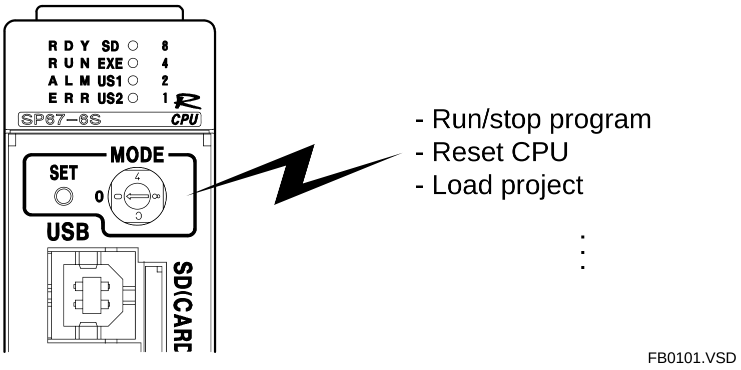


Figure B1.1.1 Rotary Switch Functions

■ List of Rotary Switch Functions

The rotary switch functions are classified into four categories, namely, boot mode selection, maintenance operation, memory card operation, and user event. They are briefly described in Tables B1.1.1 to B1.1.4 respectively. Table B1.1.1 lists the functions related to boot mode selection. For details, read the detailed description given later in this manual.

Table B1.1.1 Boot Mode Selection

Function Name	Description	Operation
ROM Boot (Run mode)	The module starts up in Run mode using the project and CPU properties stored in the internal ROM.	Startup 0 or 4 to F
ROM Boot (Stop mode)	The module starts up in Stop mode using the project and CPU properties stored in the internal ROM.	Startup 1
Card Boot (Run mode)	The module starts up in Run mode using project and CPU properties copied from source directory "\CARD1\PROJECT\" on the memory card (CARD1) into the internal ROM.	Startup 2
Card Boot (Stop mode)	The module starts up in Stop mode using project and CPU properties copied from source directory "\CARD1\PROJECT\" on the memory card (CARD1) into the internal ROM.	Startup 3

Note: - 'Startup' means a power on or reset operation.
- The given numeric value denotes the MODE switch value in hexadecimal representation.

Table B1.1.2 Maintenance Operation

Function Name	Description	Operation	Availability in Run Mode
Load Project from CARD1	Loads project of card load format and CPU properties from the memory card to the internal ROM. Source directory: \CARD1\PROJECT\	Press 8	Not executable for project; Executable for CPU properties
Save Project to CARD1	Saves project of card load format and CPU properties from the internal ROM to the memory card. Destination directory: \CARD1\PROJECT\	Press 6	Executable
Module Info	- Gets system log as a text file (syslog.txt). - Gets FTP server log as a text file (ftpslog.txt). - Gets CPU information (operating mode, LED statuses, etc.) as a text file (cpuinfo.txt). - Gets application information (project information, configuration, I/O setup information, etc.) as a text file (apinfo.txt). Destination directory: \CARD1\INFO\	Press 7	Executable
Switch Operating Mode	Switches the operating mode from Run or Debug mode to Stop mode, or from Stop mode to Run mode.	Press 1	Executable
Clear Alarms	Clears all alarms and outputs a log of cleared alarms to a text file. Destination directory: \CARD1\INFO\	Press 5	Executable
Copy RAM Disk	Copies the contents of the RAM disk to memory card. Source directory: \RAMDISK\ Destination directory: \CARD1_RAMDISK\	Press 9	Executable
Copy Memory Card	Copies the contents of the memory card to RAM disk. Destination directory: \RAMDISK\ Source directory: \CARD1_RAMDISK\	Press B	Executable
Yokogawa Maintenance	Reserved for maintenance by Yokogawa.	Press C	Not executable
Reset CPU	Resets the CPU.	Press & hold 0-3	Executable
Restore Factory Settings	Restores the module to factory settings.	Press & hold C	Not executable
Technical Support Info	Gets information file for use in technical support (techinfo.bin). Destination directory: \CARD1\INFO\	Press & hold 7	Not executable

Note: - 'Press' means to press and release the SET switch.
- Press & hold means to press and hold the SET switch.
- The given numeric value denotes the MODE switch value in hexadecimal representation.

Table B1.1.3 Memory Card Operation

Function Name	Description	Operation	Availability in Run Mode
Unmount CARD1	Unmounts memory card CARD1 so that it can be safely removed.	Press 4	Executable
Format CARD1	Formats the SD memory card in CARD1 in FAT16 format.	Press & hold A	Not executable

Note: - 'Press' means to press and release the SET switch.
- Press & hold means to press and hold the SET switch.
- The given numeric value denotes the MODE switch value in hexadecimal representation.

Table B1.1.4 User Event (press operation or press & hold operation)

Function Name	Description	Operation	Availability in Run Mode
User Event Press 1	Notifies user application of a user event (press 1) through a special relay and a special register.	Press E	Executable
User Event Press 2	Notifies user application of a user event (press 2) through a special relay and a special register.	Press F	Executable
User Event Press & Hold 1	Notifies user application of a user event (press & hold 1) through a special relay and a special register.	Press & hold E	Executable
User Event Press & Hold 2	Notifies user application of a user event (press & hold 2) through a special relay and a special register.	Press & hold F	Executable

Note: - 'Press' means to press and release the SET switch.
- Press & hold means to press and hold the SET switch.
- The given numeric value denotes the MODE switch value in hexadecimal representation.

B1.1 Rotary Switch Function Setup

This section describes how to configure the rotary switch functions.

■ Basic Setup

The rotary switch function requires no basic setup.

■ Optional Setup

The rotary switch functions may be configured as required before use.

● Removing rotary switch function

To disable all rotary switch functions, remove the rotary switch function using function removal of configuration.

SEE ALSO

For details on function removal of configuration, see Subsection A9.2.12, "Function Removal".

● Disabling selected MODE switch positions

To disable selected MODE switch positions (values), use rotary switch setup of CPU properties. However, you may not disable the boot mode function.

SEE ALSO

For details on rotary switch setup of CPU properties, see Subsection A9.5.7, "Rotary Switch Setup".

B1.2 Executing Rotary Switch Functions

This section describes how to select and execute a rotary switch function.

■ MODE and SET Switches

To select and execute a rotary switch function, operate the MODE switch, which is a rotary switch, together with the SET switch.

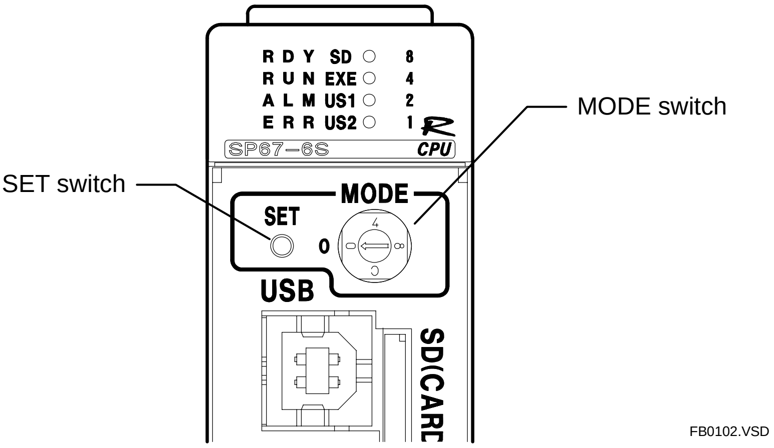


Figure B1.2.1 MODE and SET Switches

The table below describes the roles of the MODE and SET switches.

Table B1.2.1 Roles of the MODE and SET Switches

Switch	Description
MODE switch	The MODE switch is a rotary switch with 16 positions for selecting hexadecimal values 0 to F. It is used together with the SET switch to select boot modes and other functions.
SET switch	The SET switch is used together with the MODE switch to allow up to 32 selections. You press and release the SET switch (press operation) or press and hold the SET switch (press & hold operation) for a longer duration.

● Checking the value of the MODE switch

The current value of the MODE switch is indicated by the combined statuses of four LEDs (**1** LED, **2** LED, **4** LED, and **8** LED) located on the front panel of the module. You can obtain the current Mode switch value by summing up the values of lit LEDs in hexadecimal.

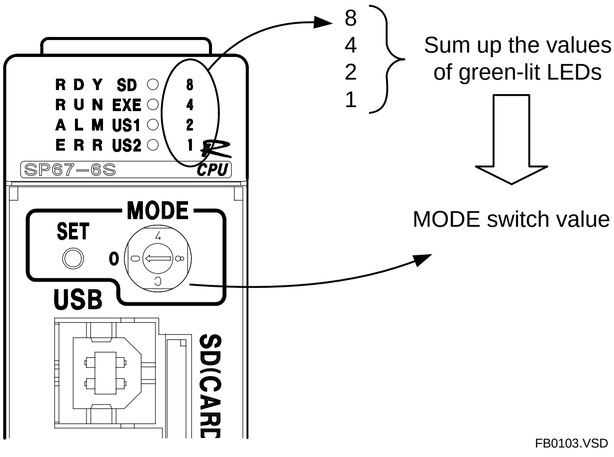


Figure B1.2.2 Value of the MODE Switch

■ Executing Rotary Switch Functions

There are three ways to execute a rotary switch function.

Table B1.2.2 Executing Rotary Switch Function

Operation	Description
Boot mode selection	The current MODE switch value is read at power on or module reset to determine the boot mode.
Press operation	The current MODE switch value is read when the SET switch is pressed and released within 1 second and the assigned press function is executed.
Press & hold operation	The current MODE switch value is read when the SET switch is pressed and held for 3 seconds or longer and the assigned press & hold function is executed.

● Boot mode selection

To select a boot mode, turn the MODE switch to the corresponding position with a screwdriver and either power on the module or reset the module. The module reads the Mode switch value and starts up in the selected boot mode.

SEE ALSO

For details on the boot modes, see Section B1.3, "Boot Modes".

● Press operation

To execute one of the rotary switch functions assigned to the press operation, turn the MODE switch to the corresponding position with a screwdriver and perform a press operation by pressing and releasing the SET switch within 1 second.

SEE ALSO

For details on the press operation, see Section B1.4, "Press Rotary Switch Functions".

● Press & hold operation

To execute one of the rotary switch functions assigned to the press & hold operation, turn the MODE switch to the corresponding position with a screwdriver and perform a press & hold operation by pressing and holding the SET switch for 3 seconds or longer.

SEE ALSO

For details on the press & hold operation, see Section B1.5, "Press & Hold Rotary Switch Functions".

B1.3 Boot Modes

The boot mode determines how a module starts up. It determines the source from which project or CPU properties are loaded, as well as the operating mode of the CPU after startup. The table below lists the available boot modes.

Table B1.3.1 List of Boot Modes

MODE Switch Value	Boot Mode	Source for Reading Project or CPU Properties	Operating Mode after Startup
0	ROM boot (Run mode)	Internal ROM	Run mode
1	ROM boot (Stop mode)	Internal ROM	Stop mode
2	Card boot (Run mode)	Memory card CARD1 \CARD1\PROJECT	Run mode
3	Card boot (Stop mode)	Memory card CARD1 \CARD1\PROJECT	Stop mode
4	(= MODE switch 0)	(= MODE switch 0)	(= MODE switch 0)
5	(= MODE switch 0)	(= MODE switch 0)	(= MODE switch 0)
6	(= MODE switch 0)	(= MODE switch 0)	(= MODE switch 0)
7	(= MODE switch 0)	(= MODE switch 0)	(= MODE switch 0)
8	(= MODE switch 0)	(= MODE switch 0)	(= MODE switch 0)
9	(= MODE switch 0)	(= MODE switch 0)	(= MODE switch 0)
A	(= MODE switch 0)	(= MODE switch 0)	(= MODE switch 0)
B	(= MODE switch 0)	(= MODE switch 0)	(= MODE switch 0)
C	(= MODE switch 0)	(= MODE switch 0)	(= MODE switch 0)
D	(= MODE switch 0)	(= MODE switch 0)	(= MODE switch 0)
E	(= MODE switch 0)	(= MODE switch 0)	(= MODE switch 0)
F	(= MODE switch 0)	(= MODE switch 0)	(= MODE switch 0)

Notes:
- MODE switch values that are disabled in the rotary switch setup of CPU properties are disabled for press and press & hold operations but remain enabled for boot mode selection. Boot mode selection remains enabled even if the rotary switch function is removed using function removal of configuration.
- The current MODE switch value can be determined from the LEDs.

TIP

MODE switch values with no assigned function have the same effect as MODE switch value 0. Therefore, powering on or resetting the module with one of these MODE switch values will execute an internal ROM boot with Run mode startup.

B1.3.1 ROM Boot (Run Mode)

The module starts up in Run mode using project and CPU properties stored in the internal ROM.

■ Operation

Table B1.3.2 Operation Specifications

MODE Switch Value	0, 4 to F
Operation	Boot up

■ LED Indication

Table B1.3.3 LED Status

LED Meaning	LED Status
Boot in progress	No LED is lit.
Normal exit[*1]	The RDY and RUN LEDs are lit. The other LEDs reflect the present module status.
Error exit	The RDY and ERR LEDs are lit. The other LEDs reflect the present module status.

*1: Indicates the status of the module right after boot up. The LED statuses may change due to subsequent events (I/O module error, etc.).

■ Function

The module starts up in Run mode using project and CPU properties stored in the internal ROM.

■ If an Error Occurs

The module switches to Stop mode.

The **ERR** LED is lit.

A message is output to the system log as shown in the table below.

Table B1.3.4 Error Message

Error Message	Code	Description
Boot mode error	10-1n	Error related to boot mode n = MODE switch value (0 or 4-F)

B1.3.2 ROM Boot (Stop Mode)

The module starts up in Stop mode using project and CPU properties stored in the internal ROM.

■ Operation

Table B1.3.5 Operation Specifications

MODE Switch Value	1
Operation	Boot up

■ LED Indication

Table B1.3.6 LED Status

LED Meaning	LED Status
Boot in progress	No LED is lit.
Normal exit[*1]	The RDY LED is lit. The RUN LED is not lit. The other LEDs reflect the present module status.
Error exit	The RDY and ERR LEDs are lit. The RUN LED is not lit. The other LEDs reflect the present module status.

*1: Indicates the status of the module right after boot up. The LED statuses may change due to subsequent events (I/O module error, etc.).

■ Function

The module starts up in Stop mode using project and CPU properties stored in the internal ROM.

■ If an Error Occurs

The module switches to Stop mode.

The **ERR** LED is lit.

A message is output to the system log as shown in the table below.

Table B1.3.7 Error Message

Error Message	Code	Description
Boot mode error	10-11	Error related to boot mode. MODE switch value = 1

B1.3.3 Card Boot (Run Mode)

The module starts up in Run mode using project file of card load format and CPU property file loaded (copied) from source directory "\CARD1\PROJECT\" on the memory card (CARD1) to the internal ROM.

■ Operation

Table B1.3.8 Operation Specifications

MODE Switch Value	2
Operation	Boot up

■ LED Indication

Table B1.3.9 LED Status

LED Meaning	LED Status
Boot in progress	The EXE LED is lit. The SD LED blinks. No other LED is lit.
Normal exit[*1]	The RDY and RUN LEDs are lit. The EXE LED is not lit. The other LEDs reflect the present module status.
Error exit	The RDY and ERR LEDs are lit. The EXE LED blinks. The other LEDs reflect the present module status.

[*1] Indicates the status of the module right after boot up. The LED statuses may change due to subsequent events (I/O module error, etc.).

■ Function

The module starts up in Run mode using project file of card load format and CPU property file copied from source directory "\CARD1\PROJECT\" on the memory card (CARD1) to the internal ROM.

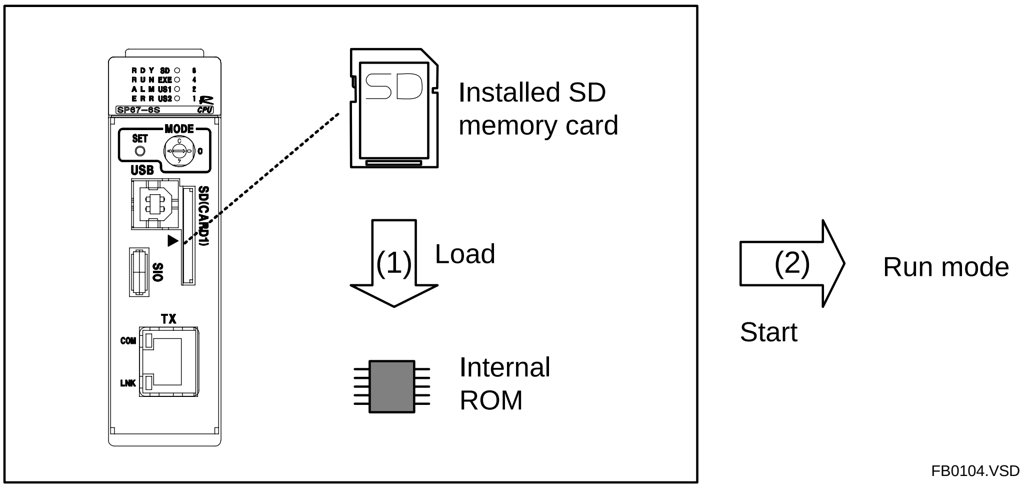


Figure B1.3.1 Card Boot (Run Mode) Function

If there are multiple files in the source directory
If there are multiple projects of card load format or CPU property files in the source directory, the project or CPU property file that has the latest time stamp is loaded.

Loading either a project or CPU property file only

To load either a project file or a CPU property file only, store only the required file in the source directory. In this case, existing project file or CPU property file in the internal ROM not overwritten by loading remains as is. If neither project nor CPU property file is found in the source directory, the function exits with an error.

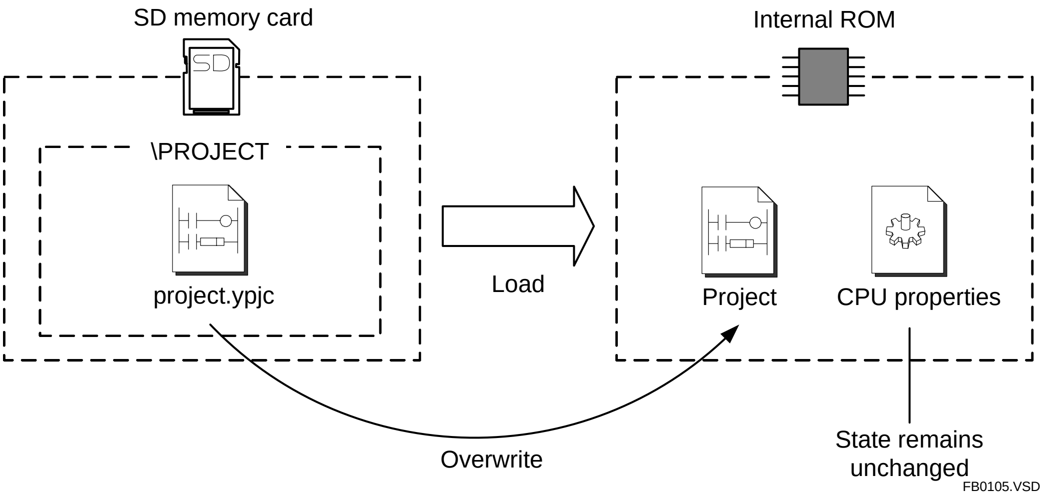


Figure B1.3.2 Loading Either Project or CPU Property Only (Card Boot (Run Mode) Function)

Protection

If executable program protection is enabled for the project stored in the internal ROM, the project to be loaded from memory card CARD1 must be protected by the same password for loading to succeed. Block protection alone has no effect on project loading.

If CPU properties data stored in the internal ROM is protected with a keyword, the property file to be loaded from memory card CARD1 must be set with the same keyword for loading to succeed.

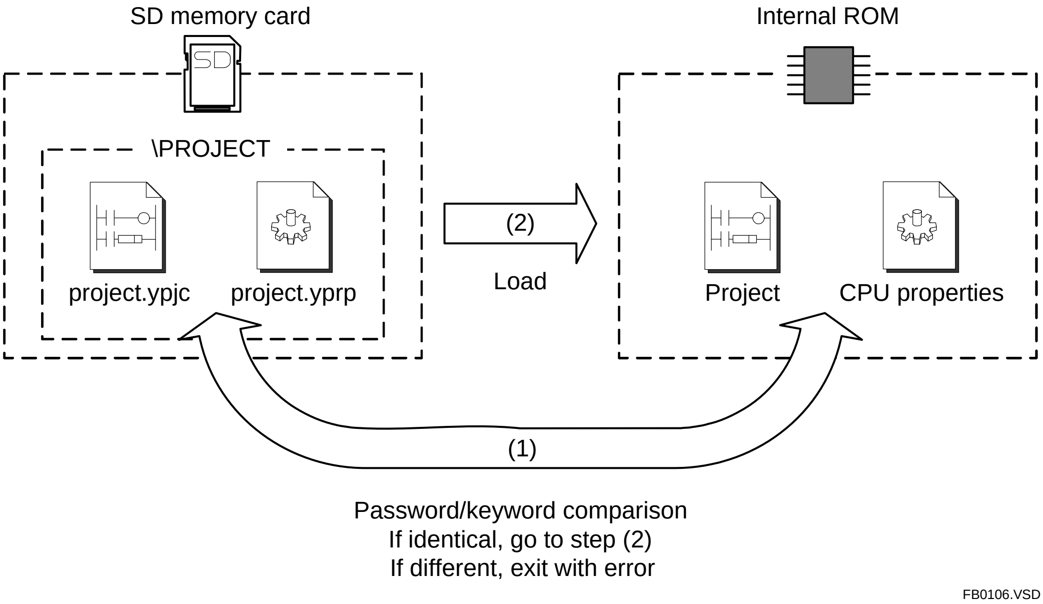


Figure B1.3.3 Protection (Card Boot (Run Mode) Function)

■ If an Error Occurs

The module stands by in Stop mode.

The **EXE** LED blinks.

The **ERR** LED is lit.

The table below shows the state of the data in the internal ROM in the event of an error.

Table B1.3.10 ROM Contents after an Error

Error	ROM Contents
Password/keyword mismatch	The data before execution is retained.
Operating mode-related error	
File reading error	
CPU property file error	The CPU property data before execution is retained.
Project file error	Project programs are cleared.
File system error	Indefinite

A message is output to the system log as shown in the table below.

Table B1.3.11 Error Message

Error Message	Code	Description
Boot mode error	10-12	Error related to boot mode. MODE switch value = 2

B1.3.4 Card Boot (Stop Mode)

The module starts up in Stop mode using project file of card load format and CPU property file loaded (copied) from source directory "\CARD1\PROJECT\" on the memory card (CARD1) to the internal ROM.

■ Operation

Table B1.3.12 Operation Specifications

MODE Switch Value	3
Operation	Boot up

■ LED Indication

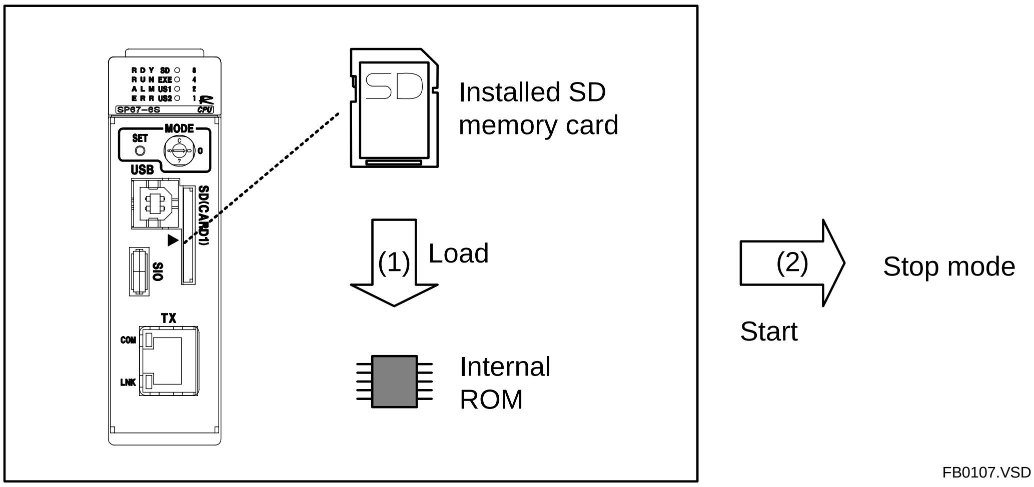
Table B1.3.13 LED Status

LED Meaning	LED Status
Boot in progress	The EXE and SD LEDs blink. No other LED is lit.
Normal exit*1	The RDY LED is lit. The EXE LED is not lit. The other LEDs reflect the present module status.
Error exit	The RDY and ERR LEDs are lit. The EXE LED blinks. The other LEDs reflect the present module status.

*1: Indicates the status of the module right after boot up. The LED statuses may change due to subsequent events (I/O module error, etc.).

■ Function

The module starts up in Stop mode using project file of card load format and CPU property file loaded (copied) from source directory "\CARD1\PROJECT\" on the memory card (CARD1) to the internal ROM. You can use this function to write an application to internal ROM before shipment.



FB0107.VSD

Figure B1.3.4 Card Boot (Stop Mode) Function

If there are multiple files in the source directory
If there are multiple projects of card load format or CPU property files in the source directory, the project or CPU property file that has the latest time stamp is loaded.

Loading either a project or CPU property file only

To load either a project file or a CPU property file only, store only the required file in the source directory. In this case, existing project file or CPU property file in the internal ROM not overwritten by loading remains as is. If neither project nor CPU property file is found in the source directory, the function exits with an error.

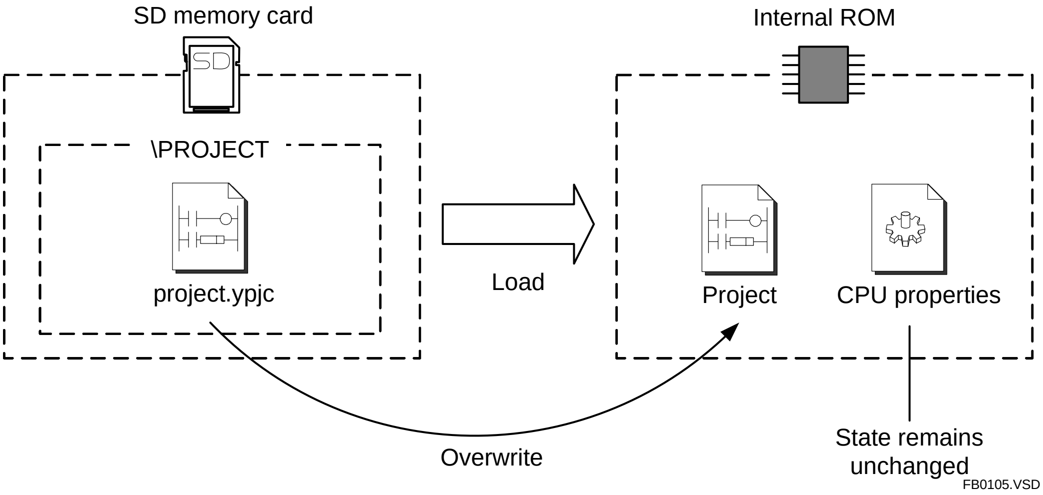


Figure B1.3.5 Loading Either Project or CPU Property Only (Card Boot (Stop Mode) Function)

Protection

If executable program protection is enabled for the project stored in the internal ROM, the project to be loaded from memory card CARD1 must be protected by the same password for loading to succeed. Block protection alone has no effect on project loading.

If CPU properties data stored in the internal ROM is protected with a keyword, the property file to be loaded from memory card CARD1 must be set with the same keyword for loading to succeed.

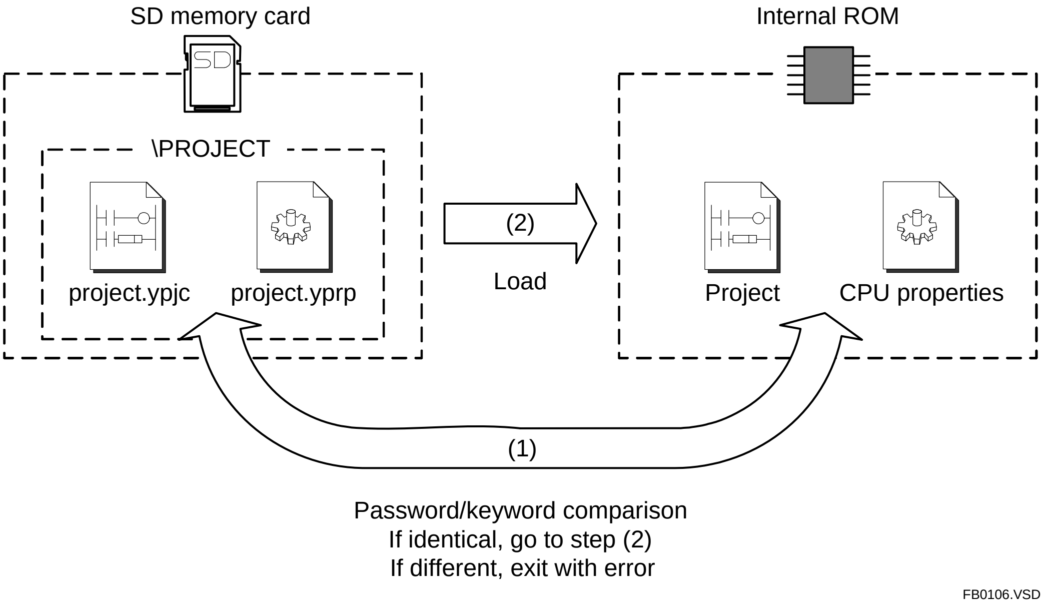


Figure B1.3.6 Protection (Card Boot (Stop Mode) Function)

■ If an Error Occurs

The module stands by in Stop mode.

The **EXE** LED blinks.

The **ERR** LED is lit.

The table below shows the state of the data in the internal ROM in the event of an error.

Table B1.3.14 ROM Contents after an Error

Error	ROM Contents
Password/keyword mismatch	The data before execution is retained.
Operating mode-related error	
File reading error	
CPU property file error	The CPU property data before execution is retained.
Project file error	Project programs are cleared.
File system error	Indefinite

A message is output to the system log as shown in the table below.

Table B1.3.15 Error Message

Error Message	Code	Description
Boot mode error	10-13	Error related to boot mode. MODE switch value = 3

B1.4 Press Rotary Switch Functions

When you set the MODE switch to the desired value and then press and release the SET switch within 1 second, the assigned press function is executed.

TIP

- All press rotary switch functions are disabled in boot mode or when the **EXE** LED is lit solid or blinking.
- The **EXE** LED does not light up if you press the SET switch but the MODE switch value has no assigned function.

CAUTION

Return the MODE switch to its original position after executing a press rotary switch function to prevent inadvertent selection of an unwanted boot mode at the next power up or reset.

The table below lists the assigned press rotary switch functions.

Table B1.4.1 List of Assigned Press Rotary Switch Functions

MODE Switch Value	Press Rotary Switch Function	Source/Destination Directory
0	—	—
1	Switch Operating Mode	—
2	—	—
3	—	—
4	Unmount CARD1	—
5	Clear Alarms	Destination: \CARD1\INFO
6	Save Project to CARD1	Destination: \CARD1\PROJECT
7	Module Info	Destination: \CARD1\INFO
8	Load Project from CARD1	Source: \CARD1\PROJECT
9	Copy RAM Disk	Source: \RAMDISK Destination: \CARD1_RAMDISK
A	—	—
B	Copy Memory Card	Source: \CARD1_RAMDISK Destination: \RAMDISK
C	Yokogawa Maintenance	—
D	—	—
E	User Event Press 1	—
F	User Event Press 2	—

B1.4.1 Switch Operating Mode

Switches the operating mode of the module.

■ Operation

Table B1.4.2 Operation Specifications

MODE Switch Value	1
Operation	Press

■ LED Indication

Table B1.4.3 LED Status

LED Meaning	LED Status
Normal exit	- RUN LED lights up when the CPU is switched to Run mode. - RUN LED goes out when the CPU is switched to Stop mode.

■ Function

Switches the operating mode of the module.

If the module is in Run or Debug mode, this function switches it to Stop mode. If the module is in Stop mode, this function switches it to Run mode.

Switching to Run mode is not allowed while edited changes are being written to the CPU module in online edit mode.

■ If an Error Occurs

Nothing happens.

B1.4.2 Unmount CARD1

Unmounts memory card CARD1 so that it can be safely removed.

■ Operation

Table B1.4.4 Operation Specifications

MODE Switch Value	4
Operation	Press

■ LED Indication

Table B1.4.5 LED Status

LED Meaning	LED Status
Execution is in progress	- The EXE LED is lit. - The SD LED blinks at irregular intervals.
Normal exit	- The EXE and SD LEDs go out.
Error exit	- The EXE LED blinks. - The SD LED is lit solid or blinks.

■ Function

Unmounts the memory card CARD1 so that it can be safely removed.

■ If an Error Occurs

The **EXE** LED blinks.

A message is output to the system log as shown in the table below.

Table B1.4.6 Error Message

Error Message	Code	Description
Smart access press function error	31-04	An error was detected by a press rotary switch function (Unmount CARD1 function)

B1.4.3 Clear Alarms

Clears all alarms and saves a cleared alarm log file named "alarmclear.txt" on memory card CARD1 in the \CARD1\INFO directory.

■ Operation

Table B1.4.7 Operation Specifications

MODE Switch Value	5
Operation	Press

■ LED Indication

Table B1.4.8 LED Status

LED Meaning	LED Status
Execution is in progress	- The EXE LED is lit. - The SD LED blinks.
Normal exit	- The EXE LED goes out. - The SD LED reflects the present module status.
Error exit	- The EXE LED blinks. - The SD LED reflects the present module status.

■ Function

Clears all alarms and saves a cleared alarm log file named "alarmclear.txt" on memory card CARD1 in the \CARD1\INFO directory. The cleared alarm log file contains the following information:

- Error code
- Error message

Any existing file named "alarmclear.txt" will be overwritten.

If any of the following conditions are true, the function generates an error and clears alarms but does not create the alarm log file.

- No memory card is mounted.
- A file system error is detected.
- An existing file named "alarmclear.txt" is read-only.

Examples of "alarmclear.txt":

- If there was a momentary power failure:

```
Momentary power failure,02-0000
```

- If there was no active alarm:

```
No error.
```

SEE ALSO

For details on the meaning of error messages, see Section B3.3, "Cleared Alarm Log Messages".

■ If an Error Occurs

The operating mode right before execution remains in effect.

The **EXE** LED blinks.

No change is made to any file or directory on memory card CARD1.

A message is output to the system log as shown in the table below.

Table B1.4.9 Error Message

Error Message	Code	Description
Smart access press function error	31-05	An error was detected by a press rotary switch function (Clear Alarms function)

TIP

- If the cause of an alarm persists, it would appear as if this function has failed to clear the alarm. In this case, remove the cause of the alarm and re-execute the function.
- This function does not clear alarms related to I/O comparison error.

B1.4.4 Save Project to CARD1

Saves project and CPU properties stored in the internal ROM to the \CARD1\PROJECT directory of memory card CARD1.

■ Operation

Table B1.4.10 Operation Specifications

MODE Switch Value	6
Operation	Press

■ LED Indication

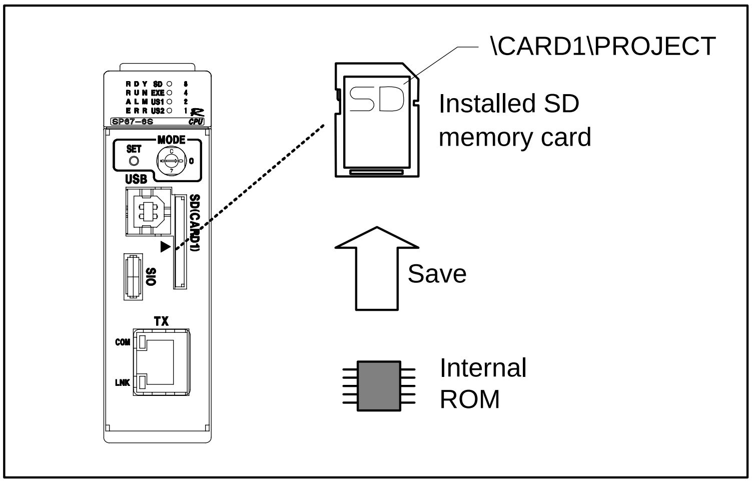
Table B1.4.11 LED Status

LED Meaning	LED Status
Execution is in progress	- The EXE LED is lit. - The SD LED blinks.
Normal exit	- The EXE LED goes out. - The SD LED reflects the present module status.
Error exit	- The EXE LED blinks. - The SD LED reflects the present module status.

■ Function

Saves project and CPU properties stored in the internal ROM to memory card CARD1 in the \CARD1\PROJECT directory with file extensions ‘.ypjc’ and ‘.yprp’ respectively. Any file of the same name in the destination directory is overwritten unless it is read-only. The files are saved in CARD1 with the following file names:

- For project file of card load format:
  Project name + .ypjc (file name extension)
- For CPU properties:
  CPU property filename at the time of loading + .yprp (file name extension)



No change in operating mode

FB0114.VSD

Figure B1.4.1 Save Project to CARD1 Function

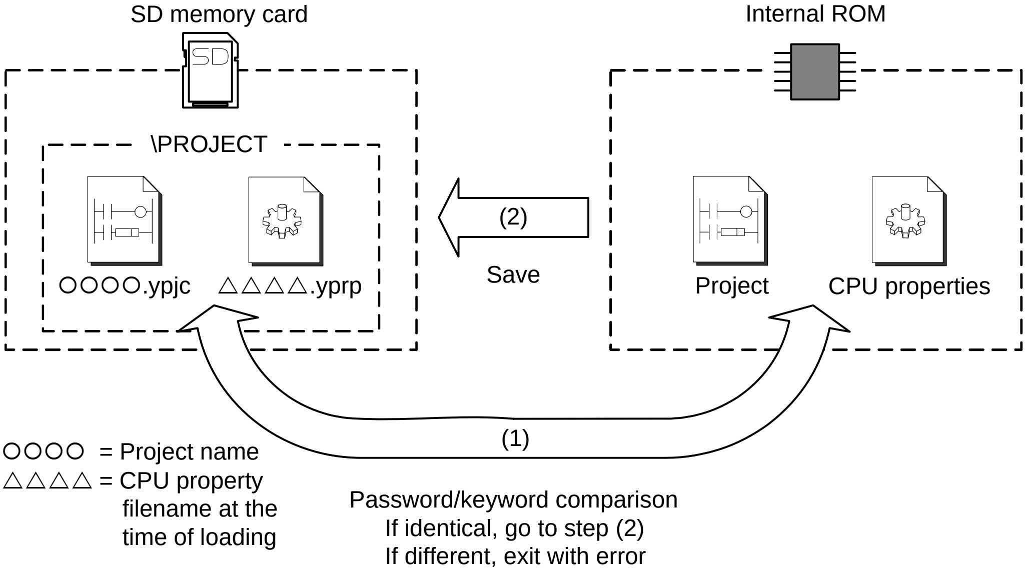
Protection

If executable program protection is enabled for the project stored in the internal ROM, a project of the same name and protected with the same password must be present on memory card CARD1 for saving to succeed. As the project on the memory card need not have the same content, you can simply create a dummy project with the same name and protect it using the same password beforehand.

Block protection alone has no effect on project saving.

If CPU property data stored in the internal ROM is protected with a keyword, a CPU property file protected with the same keyword must reside on memory card CARD1 for saving to succeed. If the keywords are different, the function exits with an error.

For security reasons, no keyword is output to the saved CPU property file.


SD memory card
Internal ROM
\PROJECT
○○○○.ypjc
△△△△.yprp
(2)
Save
Project
CPU properties
(1)
○○○○ = Project name
△△△△ = CPU property filename at the time of loading
Password/keyword comparison
If identical, go to step (2)
If different, exit with error

FB0115.VSD

Figure B1.4.2 Protection (Save Project to CARD1)

■ If an Error Occurs

The operating mode right before execution remains in effect.

The **EXE** LED blinks.

The state of the files on memory card CARD1 is indefinite as it depends on the state of processing before the error.

A message is output to the system log as shown in the table below.

Table B1.4.12 Error Message

Error Message	Code	Description
Smart access press function error	31-06	An error was detected by a press rotary switch function (Save Project to CARD1 function).

B1.4.5 Module Info

Gets and saves module information into the \CARD1\INFO directory of memory card CARD1 as text files.

■ Operation

Table B1.4.13 Operation Specifications

MODE Switch Value	7
Operation	Press

■ LED Indication

Table B1.4.14 LED Status

LED Meaning	LED Status
Execution is in progress	- The EXE LED is lit. - The SD LED blinks.
Normal exit	- The EXE LED goes out. - The SD LED reflects the present module status.
Error exit	- The EXE LED blinks. - The SD LED reflects the present module status.

■ Function

Gets and saves the following module information into the \CARD1\INFO directory of memory card CARD1 as text files:

- Log information (system logs and FTP server logs)
- CPU information
- Application information

The module information text files are given names as shown below.

Table B1.4.15 Names of Module Information Text Files

Text File	Name
System log	\CARD1\INFO\syslog.txt
FTP server log	\CARD1\INFO\ftpslog.txt
CPU information	\CARD1\INFO\cpuinfo.txt
Application information	\CARD1\INFO\apinfo.txt

Any file of the same name in the \CARD1\INFO directory is overwritten.

The function exits with error if:

- No memory card is mounted;
- A file system error is detected;
- The destination directory contains a read-only file having the same name; or
- The executable program is protected.

Protection

If executable program protection is enabled for the project stored in the module, application information cannot be returned. In this case, the function saves the other information and exits normally.

● Details of saved information

Log information:
Two types of log information are returned:

- System log
- FTP server log

CPU information:
CPU information is saved as a text file containing the following information.

Table B1.4.16 CPU Information

CPU Information	Data[*1]	Data Range
Nameplate information	MODEL = *CPU type* SERIAL NO. = *Serial number* DATE = *Date of manufacture* MAC ID = *MAC address* FIRMWARE REV. = *Revision no.*	*CPU type* [F3SP66-4S, F3SP67-6S]. *Serial number* [3 alphanumeric characters and 6 numeric characters] *Date of manufacture* [YY/MM/DD] *MAC address* [12-digit hexadecimal number] *Revision no.* [starts with R00]
Operating mode	PROGRAM MODE= *Operating mode*	*Operating mode* [0 = Stop mode 1 = Run mode or 2 = Debug mode]
LED status	RDY LED = *LED status* RUN LED = *LED status* ALM LED = *LED status* ERR LED = *LED status* SD LED = *LED status* EXE LED = *LED status* US1 LED = *LED status* US2 LED = *LED status*	*LED status* [0 = Not lit 1 = Lit 2 = Blinking]
MODE switch status	MODE SW = *MODE switch value*	*MODE switch value* [0 to F]
CARD1 mount status	CARD1 MOUNT STATUS = *Mount status*	*Mount status* [0 = Unmounted 1 = Mounted]
CARD1 free space	CARD1 FREE SPACE = *Free space*	*Free space* [bytes]
CARD1 capacity	CARD1 TOTAL SIZE = *Capacity*	*Capacity* [bytes]
RAM disk free space	RAMDISK FREE SPACE = *Free space*	*Free space* [bytes]
RAM disk capacity	RAMDISK TOTAL SIZE = *Capacity*	*Capacity* [bytes]
Alarm status	*Alarm name*	*Alarm name* (the same as that displayed by the alarm monitor)
Block activation status	*Block name 1 = Activation status* : *Block name n = Activation status*	*Block name* [up to 8 ASCII characters] *Activation status* [0 = Inactive, 1 = Active]

*1: The data range of each *italicized* item is given in the "Data Range" column.

Application information: Application information is saved as a text file containing project, configuration, and I/O setup information. Application information cannot be returned if executable program protection is enabled for a project.

(1) Project information

The following project information is output in text format.

Table B1.4.17 Application Information – Project Information

Project Information	Data[*1]	Data Range
CPU model	CPU TYPE = *CPU type*	*CPU type* [up to 9 ASCII characters]
Name of stored project	PROJECT NAME = *Project name*	*Project name* [up to 8 ASCII characters]
Name of stored CPU property data	CPU PROPERTY NAME = *CPU property name*	*CPU property name* [up to 255 ASCII characters]
Number of stored program steps	PROGRAM STEP = *Number of steps*	*Number of steps* [0 to maximum limit for the module]
Number of component blocks	BLOCK NUM = *Number of blocks*	*Number of blocks* [1 to 1024]
Names of component blocks	BLOCK NAME 1 = *Name of block 1* : BLOCK NAME n = *Name of block n*	*Name of block n* [up to 8 ASCII characters] (n is between 1 and 1024]
Names of registered macros[*2]	MACRO NAME 1 = *Name of macro 1* : MACRO NAME n = *Name of macro n*	*Name of macro n* [up to 8 ASCII characters] (n is between 1 and 256)

*1: The data range of each *italicized* item is given in the "Data Range" column.

*2: Information is not stored for unregistered macros.

(2) Configuration information

Configuration information is output in text format.

(3) I/O setup information

I/O setup information for each slot of a unit is output as one set of comma-delimited elements consisting of type, number of X points, number of Y points and number of registers as shown below. I/O setup information is available for up to 8 units each consisting of up to 16 slots. Information for each unit is delimited by the newline code.

Table B1.4.18 Application Information – I/O Setup Information

Unit	Sequence	Element	Format
UNIT0	1	Module type of SLOT1	4 ASCII characters
		Number of X points of SLOT1	2-digit decimal number
		Number of Y points of SLOT1	2-digit decimal number
		Number of registers of SLOT1	5-digit decimal number
		:	:
		Module type of SLOT16	4 ASCII characters
		Number of X points of SLOT16	2-digit decimal number
		Number of Y points of SLOT16	2-digit decimal number
		Number of registers of SLOT16	5-digit decimal number
UNIT1	2	Refer to the description for UNIT0.	Refer to the description for UNIT0.
UNIT2	3	Refer to the description for UNIT0.	Refer to the description for UNIT0.
UNIT3	4	Refer to the description for UNIT0.	Refer to the description for UNIT0.
UNIT4	5	Refer to the description for UNIT0.	Refer to the description for UNIT0.
UNIT5	6	Refer to the description for UNIT0.	Refer to the description for UNIT0.
UNIT6	7	Refer to the description for UNIT0.	Refer to the description for UNIT0.
UNIT7	8	Refer to the description for UNIT0.	Refer to the description for UNIT0.

■ If an Error Occurs

The operating mode right before execution remains in effect.

The **EXE** LED blinks.

No file or directory on the memory card is modified.

A message is output to the system log as shown in the table below.

Table B1.4.19 Error Message

Error Message	Code	Description
Smart access press function error	31-07	An error was detected by a press rotary switch function (Module Info function).

B1.4.6 Load Project from CARD1

Reads a project file of card load format and a CPU property file from the \CARD1\PROJECT directory of memory card CARD1 into the internal ROM.

■ Operation

Table B1.4.20 Operation Specifications

MODE Switch Value	8
Operation	Press

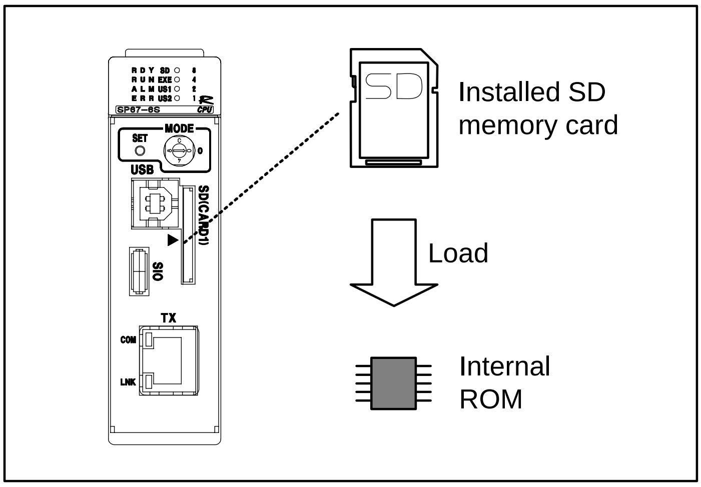
■ LED Indication

Table B1.4.21 LED Status

LED Meaning	LED Status
Execution is in progress	- The EXE LED is lit. - The SD LED blinks.
Normal exit	- The EXE LED goes out. - The SD LED reflects the present module status.
Error exit	- The EXE LED blinks. - The SD LED reflects the present module status.

■ Function

Reads a project file of card load format and a CPU property file from the \CARD1\PROJECT directory of memory card CARD1 into the internal ROM.



FB0110.VSD

Figure B1.4.3 Load Project from CARD1 Function

If there are multiple files in the source directory
If there are multiple projects of card load format or CPU property files in the source directory, the project or CPU property file that has the latest time stamp is loaded.

Loading either a project or CPU property file only

To load either a project file or a CPU property file only, store only the required file in the source directory. In this case, existing project file or CPU property file in the internal ROM not overwritten by loading remains as is. If neither project nor CPU property file is found in the source directory, the function exits with an error.

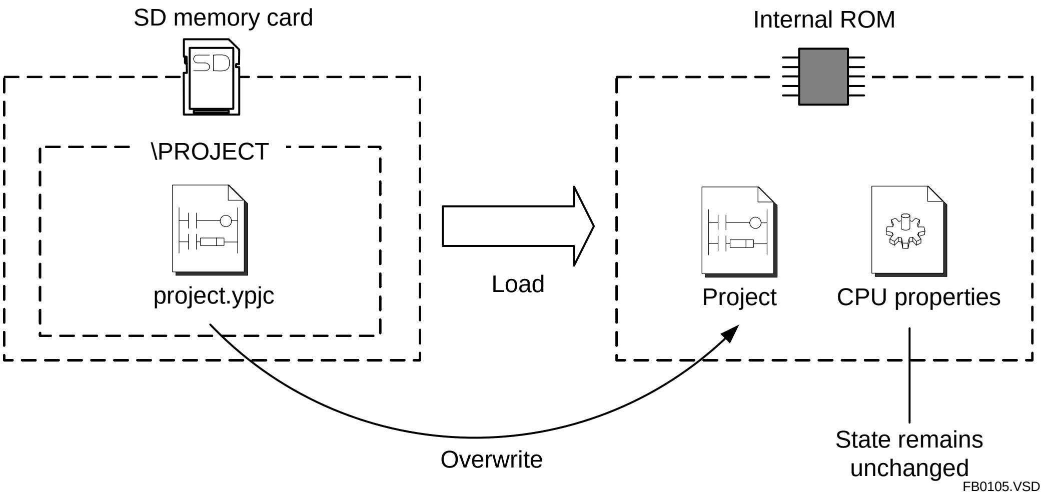


Figure B1.4.4 Loading Either Project or CPU Property Only (Load Project from CARD1 Function)

Operating mode dependency

Project loading is available only in Stop mode. CPU property loading is available in any operating mode. If you try to load both project and CPU properties in Run or Debug mode, the function exits with error without loading any file.

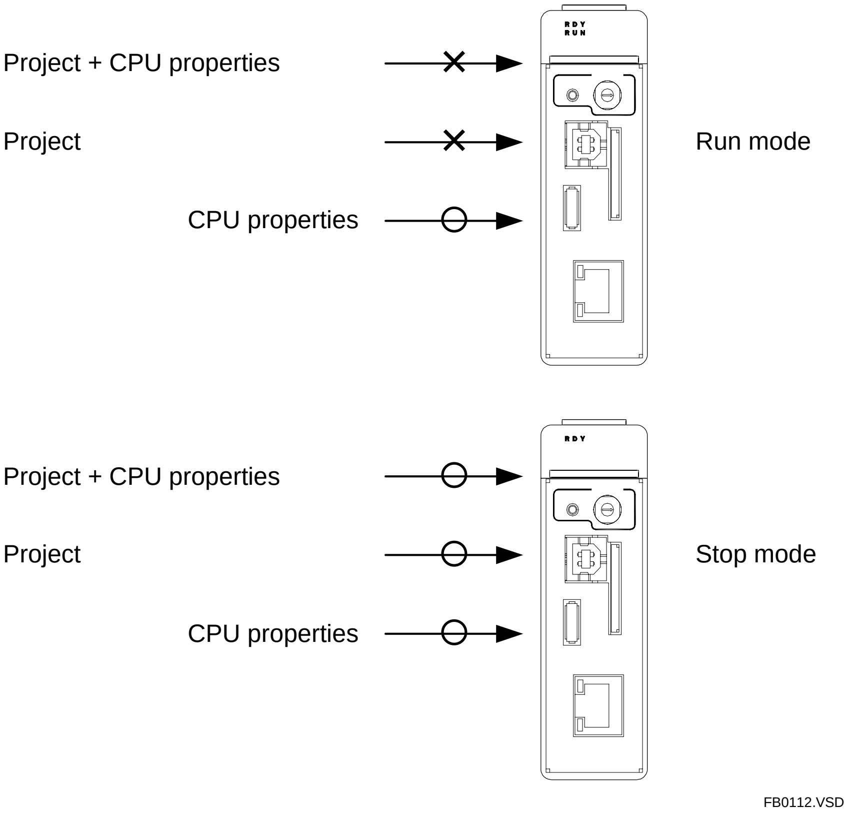
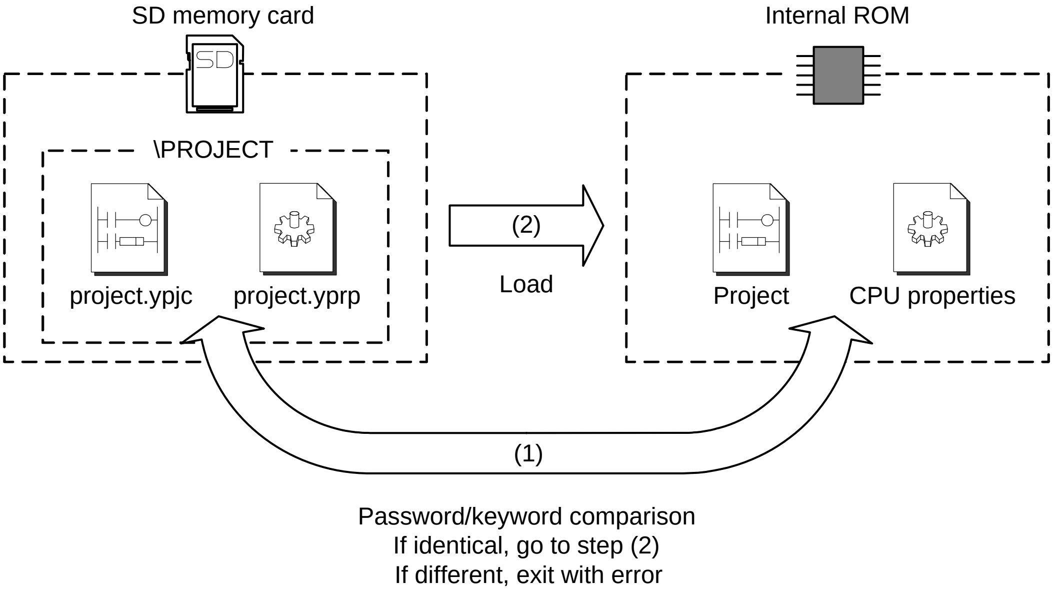


Figure B1.4.5 Operating Mode Dependency (Load Project from CARD1 Function)

Protection

If executable program protection is enabled for the project stored in the internal ROM, the project to be loaded from memory card CARD1 must be protected by the same password for loading to succeed. Block protection alone has no effect on project loading.

If CPU properties data stored in the internal ROM is protected with a keyword, the property file to be loaded from memory card CARD1 must be set with the same keyword for loading to succeed. If the keywords are different, the function exits with an error.



FB0106.VSD

Figure B1.4.6 Protection (Load Project from CARD1 Function)

■ If an Error Occurs

The operating mode right before execution remains in effect.

The **EXE** LED blinks.

The table below shows the state of the data in the internal ROM in the event of an error.

Table B1.4.22 ROM Contents after an Error

Error	ROM Contents
Password/keyword mismatch	The data before execution is retained.
Operating mode-related error	
File reading error	
CPU property file error	The CPU property data before execution is retained.
Project file error	Project programs are cleared.
File system error	Indefinite

A message is output to the system log as shown in the table below.

Table B1.4.23 Error Message

Error Message	Code	Description
Smart access press function error	31-08	An error was detected by a press rotary switch function (Load Project from CARD1 function).

B1.4.7 Copy RAM Disk

Copies the contents of the RAM disk to the \CARD1_RAMDISK directory of memory card CARD1 in uncompressed format, retaining the directory structure.

■ Operation

Table B1.4.24 Operation Specifications

MODE Switch Value	9
Operation	Press

■ LED Indication

Table B1.4.25 LED Status

LED Meaning	LED Status
Execution is in progress	- The EXE LED is lit. - The SD LED blinks.
Normal exit	- The EXE LED goes out. - The SD LED reflects the present module status.
Error exit	- The EXE LED blinks. - The SD LED reflects the present module status.

■ Function

Copies the contents of the RAM disk to the \CARD1_RAMDISK directory of memory card CARD1 in uncompressed format, retaining the directory structure.

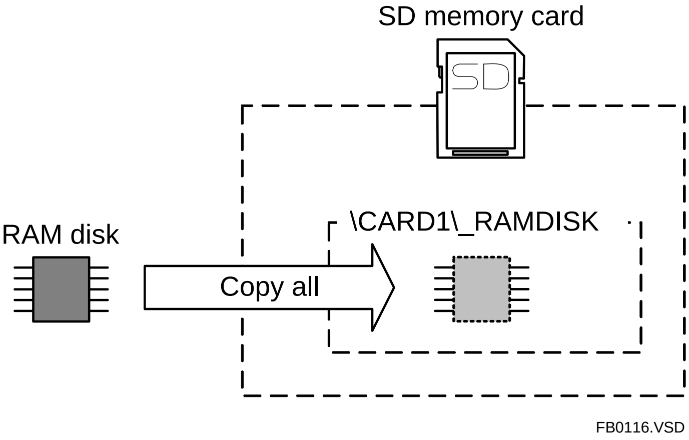


Figure B1.4.7 Copy RAM Disk Function

The function retains the source directory structure up to 8 levels where \RAMDISK counts as the first level directory. It exits with error if you attempt to copy a RAM disk with a directory structure deeper than 8 levels.

It also exits with error if a CARD1, RAM disk, or file system access error is detected.

It also exits with error if the destination directory contains a read-only file of the same name.

TIP

The contents of the RAM disk and its copy on the memory card may disagree due to file operations executed by a program or otherwise during the copying process.

■ If an Error Occurs

The operating mode right before execution remains in effect.

The **EXE** LED blinks.

The state of the files and directories on memory card CARD1 is indefinite as it depends on the state of processing before the error.

A message is output to the system log as shown in the table below.

Table B1.4.26 Error Message

Error Message	Code	Description
Smart access press function error	31-09	An error was detected by a press rotary switch function (Copy RAM Disk function).

B1.4.8 Copy Memory Card

Copies the contents of the \CARD1_RAMDISK directory on memory card CARD1 to the \RAMDISK directory of the RAM disk, preserving the source directory structure.

■ Operation

Table B1.4.27 Operation Specifications

MODE Switch Value	B
Operation	Press

■ LED Indication

Table B1.4.28 LED Status

LED Meaning	LED Status
Execution is in progress	- The EXE LED is lit. - The SD LED blinks.
Normal exit	- The EXE LED goes out. - The SD LED reflects the present module status.
Error exit	- The EXE LED blinks. - The SD LED reflects the present module status.

■ Function

Copies the contents of the \CARD1_RAMDISK directory on memory card CARD1 to the \RAMDISK directory of the RAM disk, preserving the source directory structure.

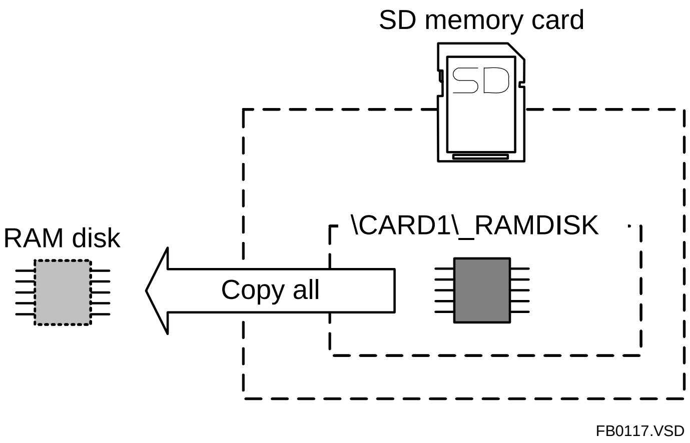


Figure B1.4.8 Copying Memory Card to RAM Disk

The function preserves the source directory structure up to 8 levels where \CARD1_RAMDISK counts as the first level directory. It exits with error if you attempt to copy a memory card with a directory structure deeper than 8 levels.

It also exits with error if a CARD1, RAM disk, or file system access error occurs.

It also exits with error if the destination directory contains a read-only file of the same name.

TIP

The contents of the memory card CARD1 and its copy on the RAM disk may disagree due to file operations executed by a program or otherwise during the copying process.

■ If an Error Occurs

The operating mode right before execution remains in effect.

The **EXE** LED blinks.

The state of the files or directories on the RAM disk is indefinite as it depends on the state of processing before the error.

A message is output to the system log as shown in the table below.

Table B1.4.29 Error Message

Error Message	Code	Description
Smart access press function error	31-0B	An error was detected by a press rotary switch function (copy Memory Card function).

B1.4.9 User Event Press 1, User Event Press 2

These rotary switch functions allow a ladder program to read the current rotary switch value and process the specified user event accordingly.

■ Operation

Table B1.4.30 Operation Specifications

MODE Switch Value	E or F
Operation	Press

■ LED Indication

Undefined

■ Function

These functions allow a ladder program to read the current rotary switch value and process the specified user event accordingly. When these functions are executed, the current MODE switch value is stored in special register Z0117 and the Press Event special relay (M0079) is turned on for one scan cycle.

You can program a ladder program to monitor the status of the Press Event special relay (M0079) and process a user event press 1 (or 2) if special register Z0117 contains a value of $E (or $F) when the special relay turns on. As these functions do not control the LEDs, a ladder program may freely control them as required.

■ If an Error Occurs

Nothing happens.

B1.4.10 Yokogawa Maintenance

This function is to be used exclusively by Yokogawa service personnel.

■ Operation

Table B1.4.31 Operation Specifications

MODE Switch Value	C
Operation	Press

■ LED Indication

Table B1.4.32 LED Status

LED Meaning	LED Status
Execution is in progress	- The EXE LED blinks. - The SD LED blinks at irregular intervals.
Normal exit	- The EXE LED goes out. - The SD LED reflects the present module status.
Error exit	- The EXE LED blinks. - The SD LED reflects the present module status.

■ Function

This function is to be used exclusively by Yokogawa service personnel.

If a user executes this function, a Smart access press function error is generated which, however, does not affect the user application or device data.

■ If an Error Occurs

The **EXE** LED blinks.

A message is output to the system log as shown in the table below.

Table B1.4.33 Error Message

Error Message	Code	Description
Smart access press function error	31-0C	An error was detected by a press rotary switch function (Yokogawa Maintenance function).

B1.5 Press & Hold Rotary Switch Functions

When you set the MODE switch to the desired value and then press and hold the SET switch for 3 seconds or longer, the assigned press & hold function is executed.

TIP

- All press & hold rotary switch functions are disabled in boot mode or when the **EXE** LED is lit solid or blinking.
- The **EXE** LED does not light up if you press the SET switch but the MODE switch value has no assigned function.

CAUTION

Return the MODE switch to its original position after executing a press & hold rotary switch function to prevent inadvertent selection of an unwanted boot mode at the next power up or reset.

The table below lists the assigned press & hold rotary switch functions:

Table B1.5.1 List of Assigned Press & Hold Rotary Switch Functions

MODE Switch Value	Press & Hold Rotary Switch Function	Source/Destination Directory
0	Reset CPU	—
1	Reset CPU	—
2	Reset CPU	—
3	Reset CPU	—
4	—	—
5	—	—
6	—	—
7	Technical Support Info	Destination: \CARD1\INFO
8	—	—
9	—	—
A	Format CARD1	—
B	—	—
C	Restore Factory Settings	—
D	—	—
E	User Event Press & Hold 1	—
F	User Event Press & Hold 2	—

B1.5.1 Reset CPU

Resets the CPU.

■ Operation

Table B1.5.2 Operation Specifications

MODE Switch Value	0 to 3
Operation	Press & hold

■ LED Indication

Table B1.5.3 LED Status

LED Meaning	LED Status
—	—

■ Function

Resets the CPU.

CAUTION

Pay attention to the boot mode, which is determined by the MODE switch value, before resetting the CPU.

■ If an Error Occurs

The **EXE** LED blinks.

A message is output to the system log as shown in the table below.

Table B1.5.4 Error Message

Error Message	Code	Description
Smart access press and hold function error	32-0n	An error was detected by a press & hold rotary switch function (n = MODE switch number within the range 0-3).

B1.5.2 Technical Support Info

Gets information for technical support.

■ Operation

Table B1.5.5 Operation Specifications

MODE Switch Value	7
Operation	Press & hold

■ LED Indication

Table B1.5.6 LED Status

LED Meaning	LED Status
Execution is in progress	- The EXE LED is lit. - The SD LED blinks.
Normal exit	- The EXE LED goes out. - The SD LED reflects the present module status.
Error exit	- The EXE LED blinks. - The SD LED reflects the present module status.

■ Function

Gets information for technical support.

You can execute this function when asked to do so by Yokogawa technical support personnel. When executed, the function outputs a binary file named "TECHINFO.BIN" to the \CARD1\INFO directory of memory card CARD1. The binary file does not include user’s ladder programs.

This function is only available in Stop mode.

■ If an Error Occurs

The **EXE** LED blinks.

A message is output to the system log as shown in the table below.

Table B1.5.7 Error Message

Error Message	Code	Description
Smart access press and hold error	32-07	An error was detected by a press & hold rotary switch function (Technical Support Info function).

B1.5.3 Format CARD1

Formats the SD memory card inserted and mounted in the CARD1 slot in FAT16 format.

■ Operation

Table B1.5.8 Operation Specifications

MODE Switch Value	A
Operation	Press & hold

■ LED Indication

Table B1.5.9 LED Status

LED Meaning	LED Status
Execution is in progress	- The EXE LED is lit. - The SD LED blinks.
Normal exit	- The EXE LED goes out. - The SD LED reflects the present module status.
Error exit	- The EXE LED blinks. - The SD LED reflects the present module status.

■ Function

Formats the SD memory card inserted and mounted in the CARD1 slot in FAT16 format.

If you insert an SD memory card not formatted in FAT16 format, the memory card will not be mounted but you can still format it in FAT16 format using this function.

This function is available only in Stop mode.

■ If an Error Occurs

The **EXE** LED blinks.

A message is output to the system log as shown in the table below.

Table B1.5.10 Error Message

Error Message	Code	Description
Smart access press and hold error	32-0A	An error was detected by a press & hold rotary switch function (Format CARD1 function).

B1.5.4 Restore Factory Settings

Restores the module to factory settings.

■ Operation

Table B1.5.11 Operation Specifications

MODE Switch Value	C
Operation	Press & hold

■ LED Indication

Table B1.5.12 LED Status

LED Meaning	LED Status
Execution is in progress	- The EXE LED is lit. - 1 LED, 2 LED, 4 LED and 8 LED light up and go out sequentially and cyclically. - The RDY LED is lit.
Normal exit	- All LEDs go out and then the RDY LED lights up.

■ Function

Restores the module to factory settings. When function execution is completed, all LEDs go out and then the RDY LED lights up. Never switch off the module before execution is completed. In the worst case, the module may no longer start up. For safety reasons, it is recommended that you remove all units except for the power supply unit and the module itself before executing this function.

As project and CPU properties in the internal ROM are also reset to factory settings, you should backup all required information before executing this function.

The function is available only in Stop mode.

TIP

If all rotary switch functions are disabled by function removal of configuration, you can still restore the module to factory settings using the procedure given below. Observe the same precautions described for the Restore Factory Settings function.

1. Set the MODE switch to the C position and install the module in slot 5 or a higher slot.
2. Switch on the power.
3. The reset is completed when the RDY LED lights up.

If the MODE switch is not set to the C position in step 1 above, all data except for project and CPU properties stored in the internal ROM will be cleared by this procedure.

■ If an Error Occurs

The **EXE** LED blinks.

A message is output to the system log as shown in the table below.

Table B1.5.13 Error Message

Error Message	Code	Description
Smart access press and hold error	32-0C	An error was detected by a press & hold rotary switch function (Restore Factory Settings function).

B1.5.5 User Event Press & Hold 1, User Event Press & Hold 2

These rotary switch functions allow a ladder program to read the current rotary switch value and process the specified user event accordingly.

■ Operation

Table B1.5.14 Operation Specifications

MODE Switch Value	E or F
Operation	Press & hold

■ LED Indication

Undefined

■ Function

These functions allow a ladder program to read the current rotary switch value and process the specified user event accordingly. When these functions are executed, the current MODE switch value is stored in special register Z0117 and the Press and Hold Event special relay (M0080) is turned on for one scan cycle.

You can program a ladder program to monitor the status of the Press and Hold Event special relay (M0080) and process a user event press & hold 1 (or 2) if special register Z0117 contains a value of $E (or $F) when the special relay turns on. As these functions do not control the LEDs, a ladder program may freely control them as required.

■ If an Error Occurs

Nothing happens.

B1.6 Resetting Rotary Switch Function Error

If a rotary switch function detects an error, the EXE LED blinks. In this state, the press or press & hold rotary switch functions are no longer available. To reset the error condition, press and release the SET switch within one second.

B1.7 Special Relays and Special Registers

■ Special Relays

Table B1.7.1 Special Relays Related to Rotary Switch

Category	Sequence Operation Status Relays		
No.	Name	Function	Description
M0079	Press Event	Reports a press event	The User Event Press 1 (or 2) function causes this relay to turn on for one scan cycle when a user presses and releases the SET switch with the MODE switch set to $E (or $F). A program may monitor this relay together with special register Z0117 to detect and process a user-defined event. This is a read-only relay.
M0080	Press & Hold Event	Reports a press & hold event	The User Event Press & Hold 1 (or 2) function causes this relay to turn on for one scan cycle when a user presses and holds the SET switch with the MODE switch set to $E (or $F). A program may monitor this relay together with special register Z0117 to detect and process a user-defined event. This is a read-only relay.
M0123	EXE LED (1)	ON : Lit OFF : Not lit	Turns on if the EXE LED is lit. Turns off if the EXE LED is not lit. If the EXE LED is blinking, the EXE LED (2) relay (M0124) turns on. In this case, the M0123 relay status has no significance. This is a read-only relay.
M0124	EXE LED (2)	ON : Blinking OFF : Not blinking	Turns on if the EXE LED is blinking. Turns off if the EXE LED is lit solid or not lit. To determine whether the EXE LED is lit solid or not lit, check the status of the EXE LED (1) relay (M0123). This is a read-only relay.

■ Special Registers

Table B1.7.2 Special Registers Related to Rotary Switch

Category	Sequence Operation Status Registers		
No.	Name	Function	Description
Z0117	MODE Switch	Indicates the MODE switch value	This register indicates the current MODE switch value. It is updated when a user presses & releases or presses & holds the SET switch and any updated value remains unchanged for at least one scan cycle. This is a read-only register.

B2. Card Batch File Function

The card batch file function enables card batch commands coded in a card batch file stored on a memory card to be automatically read, interpreted and executed by the module when an execution trigger occurs. As a card batch file can be coded with many commands, it allows complex processing to be performed.
Card batch commands are provided for project loading, file operation and other functions. Various execution triggers are also supported for initiating automatic execution of a card batch file upon the mounting of a memory card, an error or other events.
In addition to execution by a trigger, you can also manually perform execution using a command by specifying a card batch filename in a virtual directory command.

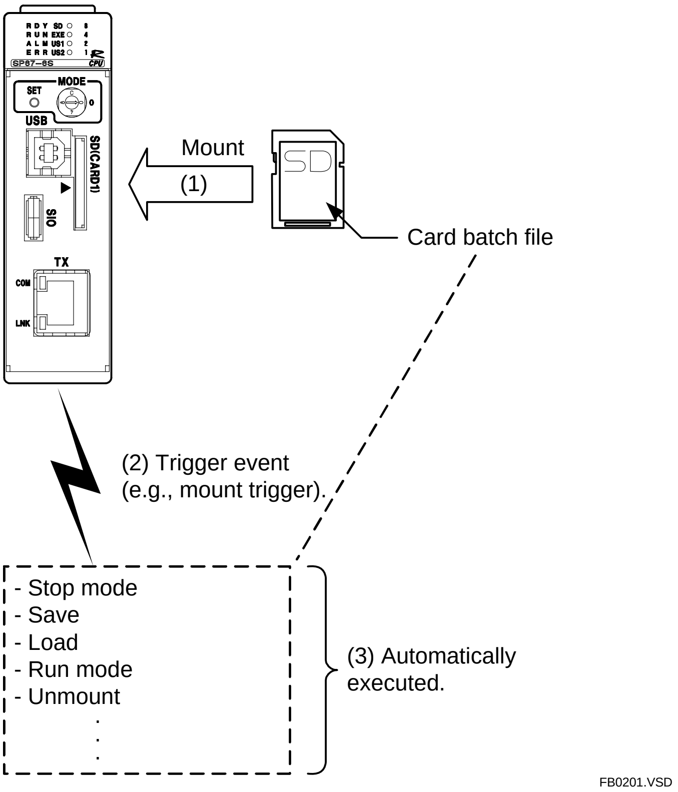


Figure B2.1 Execution of a Card Batch File (initiated by a MOUNT trigger)

Uses of the Card Batch File Function

Some typical uses of the card batch file function are described below.

Automating daily routines

With the card batch file function, a daily routine of transferring recipe files and device parameters to the module in the morning and obtaining log data in the evening can be simplified: simply insert a memory card pre-stored with a batch file of card batch commands in the morning and retrieve it in the evening.

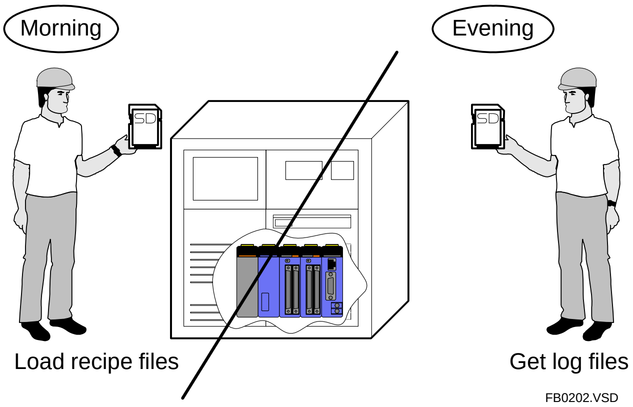


Figure B2.2 Automating Daily Routines

Troubleshooting from a remote location

Consider a situation where a system delivered to an end user reports an error, and the developer requires system log and device information to troubleshoot the problem but the end user does not know how to run the WideField2 software.

In this case, the developer can E-mail a batch file to the end user for loading into the memory card, or send a memory card containing a batch file to the end user. When the end user inserts the memory card, the batch file runs automatically to save the information required for troubleshooting on the memory card.

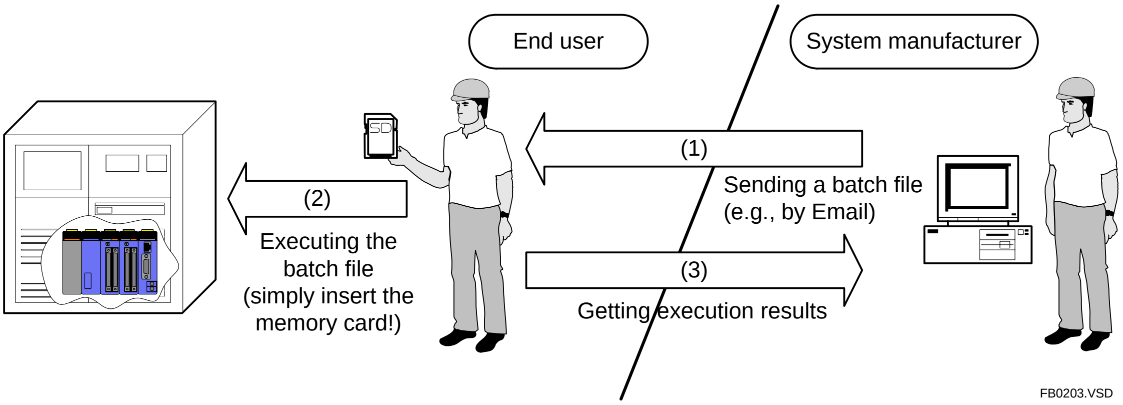


Figure B2.3 Troubleshooting from a Remote Location

■ List of Card Batch File Commands

A list of card batch file commands is given below.

SEE ALSO

For details on card batch file commands, see Section B2.8, "Specifications of Card Batch File Commands".

Table B2.1 Card Batch File Commands (File/Device Conversion Commands)

Command Name	Command Mnemonic	Function	Availability in Run Mode
Convert CSV File to Device	F2DCSV	Converts a CSV formatted file into device data.	Executable
Convert Device to CSV File	D2FCSV	Converts device data into a CSV formatted file.	Executable
Convert Binary File to Device	F2DBIN	Converts a binary file into device data.	Executable
Convert Device to Binary File	D2FBIN	Converts device data into a binary file.	Executable

Table B2.2 Card Batch File Commands (Maintenance Commands)

Command Name	Command Mnemonic	Function	Availability in Run Mode
Load Project	LOAD	Loads project and CPU properties to the internal ROM.	Not executable for project. Executable for CPU property.
Save Project	SAVE	Saves project and CPU properties residing on the internal ROM.	Executable
Get Log	LOG	Gets system log and FTP server log in text format.	Executable
CPU Info	CPUINFO	Gets CPU status (Run/Stop mode, alarm statuses, rotary switch value and card mount status).	Executable
Application Info	APINFO	Gets system information (project information and I/O setup information)	Executable
Technical Support Info	TECHINFO	Gets technical support information as a binary file.	Not executable
Run Mode	RUN	Switches operating mode to Run mode.	Executable
Stop Mode	STOP	Switches operating mode to Stop mode.	Executable
Activate Block	ACT	Activates a specified block.	Executable
Inactivate Block	INACT	Inactivates a specified block.	Executable
Reset CPU	CPURESET	Resets CPU.	Executable
Clear Alarms	ALMCLEAR	Clears all alarms.	Executable
Help	HELP	Gets help information on card batch file commands.	Executable

Table B2.3 Card Batch File Commands (File Operation and Disk Operation Commands)

Command Name	Command Mnemonic	Function	Availability in Run Mode
Copy File	COPY	Copies one or more files.	Executable
Move File	MOVE	Moves one or more files.	Executable
Delete File	DEL	Deletes one or more files.	Executable
Make Directory	MKDIR	Creates a directory.	Executable
Remove Directory	RMDIR	Removes a directory.	Executable
Rename File	REN	Renames a file.	Executable
File Status	STAT	Gets file status information (present or absent, etc.).	Executable
Change Directory	CD	Changes the current directory.	Executable
Concatenate File	CAT	Concatenates two files.	Executable
Change File Attribute	ATRW	Changes the attribute of a file.	Executable
Unmount	UNMOUNT	Unmounts a memory card.	Executable

■ List of Card Batch File Execution Methods

A card batch file may be automatically executed by an execution trigger or manually executed using a virtual directory command (see Table B2.4 below). Table B2.5 lists available execution triggers.

Table B2.4 Methods for Running a Card Batch File

Execution Method	Description
Execution by a trigger	A card batch file may be automatically executed by a trigger event such as the mounting of a memory card or a module startup.
Execution using a command	You can also run a card batch file manually using a virtual directory command (BATGO).

Table B2.5 List of Execution Triggers

Execution Trigger	Description
START (Startup event)	A card batch file associated with the START trigger is automatically executed when the module is powered on or reset. The module completes execution of the card batch file before switching its operating mode (Run mode or Stop mode) according to the selected boot mode.
RUN (Run Program event)	A card batch file associated with the RUN trigger is automatically executed when the operating mode is switched from Stop mode to Run or Debug mode. Execution of the card batch file is completed before mode switching is completed (and the RUN LED lights up). Switching from Debug mode to Run mode does not trigger execution of the card batch file. The tool service and the higher-level link service are suspended during execution of the card batch file.
STOP (Stop Program event)	A card batch file associated with the STOP trigger is automatically executed when the operating mode is switched from Run or Debug mode to Stop mode after the RUN LED goes out. If the switch to Stop mode is caused by an error, 1) A card batch file associated with the ERROR trigger is executed before 2) A card batch file associated with the STOP trigger.
MOUNT (Mount Memory Card event)	A card batch file associated with the MOUNT trigger is automatically executed when a memory card is inserted or when a ladder program executes a Mount instruction after the memory card is unmounted. The card batch file will not be executed at power on or module reset.
ERROR (Moderate failure)	A card batch file associated with the ERROR trigger is automatically executed when a moderate failure is generated. Moderate and minor failures are defined in the configuration under operation control, which defines whether to stop execution (for a moderate failure) or continue execution (for a minor failure) after each type of error. If a program is switched from Run mode to Stop mode due to an error, 1) A card batch file associated with the ERROR trigger is executed before 2) A card batch file associated with the STOP trigger.
ALARM (Minor failure)	A card batch file associated with the ALARM trigger is automatically executed when a minor failure (or alarm) is generated. Alarms due to a momentary power failure, however, will not trigger its execution. Moderate and minor failures are defined in the configuration under operation control, which defines whether to stop execution (for a moderate failure) or continue execution (for a minor failure) after each type of error.

CAUTION

The tool service and higher-level link service are suspended during execution of a card batch file executed by a RUN trigger. In this case, the tool service or higher-level link service may generate a timeout error if the execution of the card batch file takes too long.

B2.1 Card Batch File Function Setup

This section describes how to configure the card batch file function.

■ Basic Setup

The card batch file function requires no basic setup.

■ Optional Setup

The card batch file function may be configured as required before use.

Table B2.1.1 Optional Setup for Card Batch File Function

Name of Setup	Type of Setup	SEE ALSO
FTP Server Setup	CPU properties	A9.5.6, "FTP Server Setup"
Function removal	Configuration	A9.2.12, "Function Removal"

● FTP server setup

You must perform FTP server setup before you can run a card batch file with a virtual directory command. Other related setup (e.g. setup on the client end) may also be required.

The FTP server may generate a timeout error (SE05) if the execution of a card batch file takes too long. In this case, you may want to increase the value of the FTP Server Network Timeout setting (FTPS_NETACK_TOUT).

SEE ALSO

For details on FTP server setup, see Subsection A9.5.6, "FTP Server Setup".

● Function removal

To disable the card batch file function, remove the card batch file function using function removal of configuration.

B2.2 Using Card Batch File Function

A card batch file may be executed automatically by an execution trigger or manually using a virtual directory command.

Table B2.2.1 Methods for Running a Card Batch File

Execution Method	Description
Execution by a trigger	A card batch file may be automatically executed by a trigger event such as the mounting of a memory card or a module startup.
Execution using a command	You can also run a card batch file manually using a virtual directory command (BATGO).

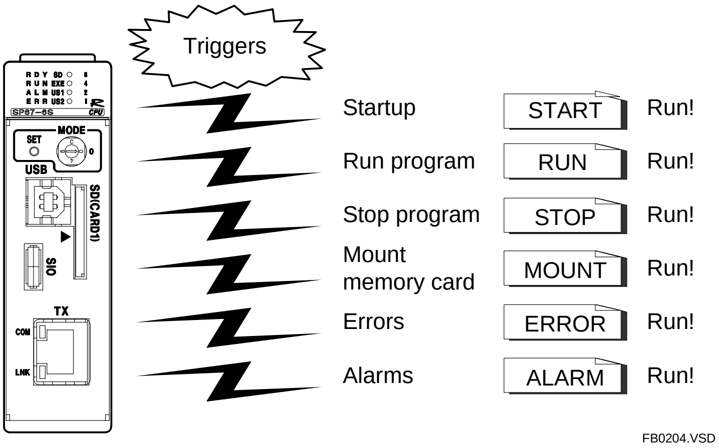
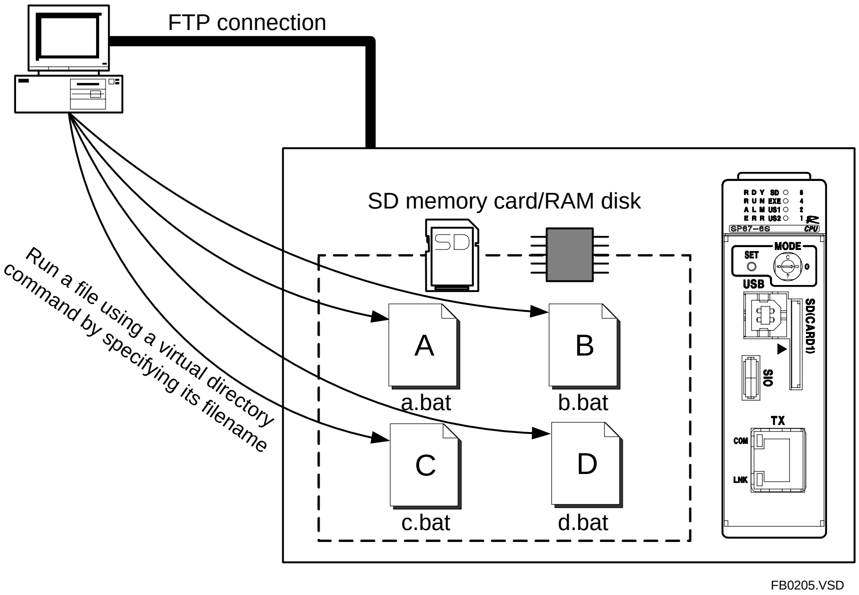


Figure B2.2.1 Card Batch File Execution by a Trigger



FTP connection
SD memory card/RAM disk
Run a file using a virtual directory command by specifying its filename
A
a.bat
B
b.bat
C
c.bat
D
d.bat
FB0205.VSD

Figure B2.2.2 Card Batch File Execution using a Command

SEE ALSO

- For details on how to run a card batch file, see Section B2.5, "Running a Card Batch File and Checking Execution Status" of this manual.
- For details on virtual directory commands, see Section 3.7, "Virtual Directory Commands" and Subsection 3.7.9.1, "Run Card Batch File (BATGO)" of "Sequence CPU – Network Functions User's Manual (for F3SP66-4S, F3SP67-6S)" (IM34M6P14-02E).

■ Procedure for Card Batch File Execution by a Trigger

Follow the procedure below to create and run a card batch file using an execution trigger.

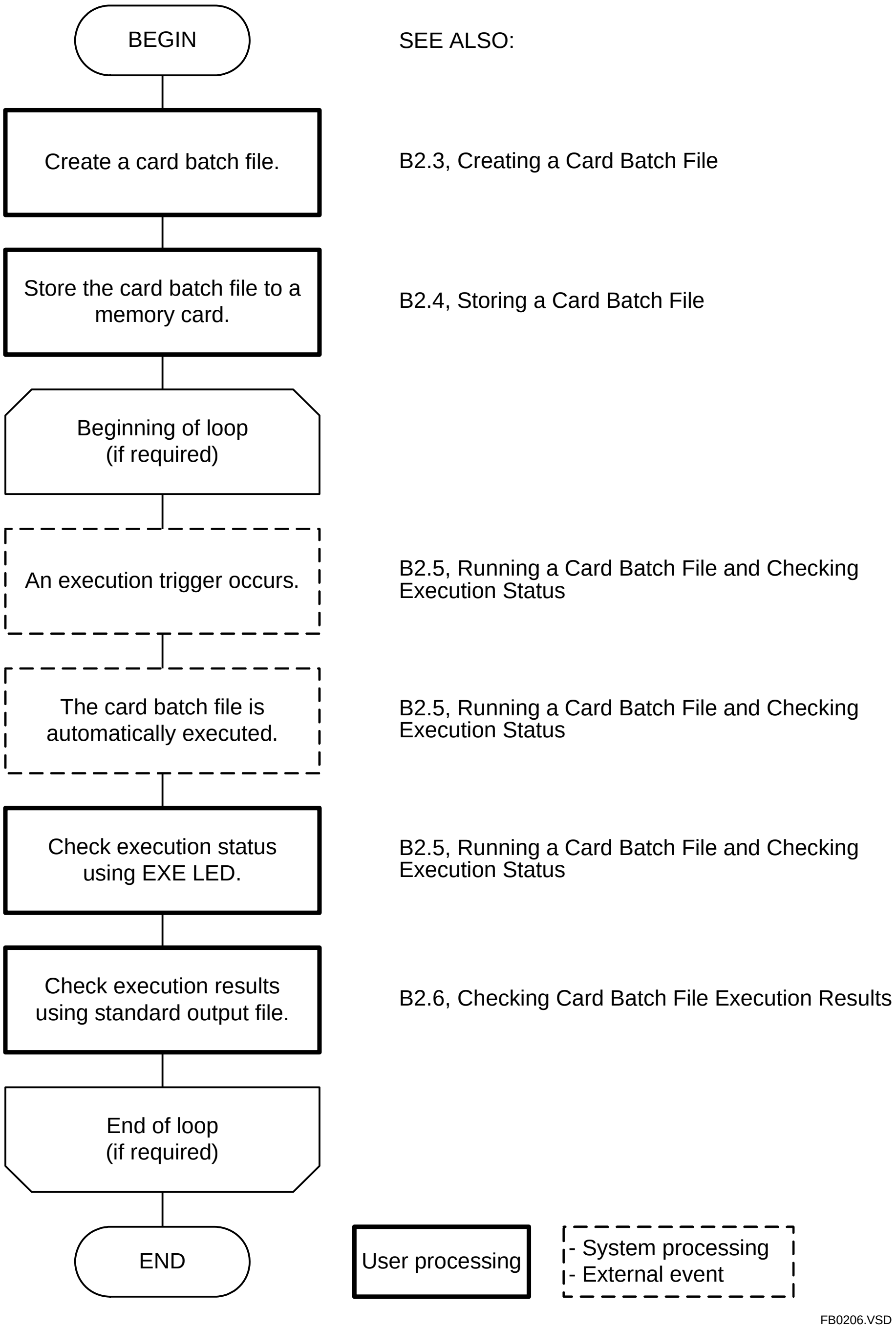

BEGIN
Create a card batch file.
Store the card batch file to a memory card.
Beginning of loop (if required)
An execution trigger occurs.
The card batch file is automatically executed.
Check execution status using EXE LED.
Check execution results using standard output file.
End of loop (if required)
END
SEE ALSO:
B2.3, Creating a Card Batch File
B2.4, Storing a Card Batch File
B2.5, Running a Card Batch File and Checking Execution Status
B2.5, Running a Card Batch File and Checking Execution Status
B2.5, Running a Card Batch File and Checking Execution Status
B2.6, Checking Card Batch File Execution Results
User processing
- System processing
- External event
FB0206.VSD

Figure B2.2.3 Running a Card Batch File using a Trigger

■ Procedure for Card Batch File Execution Using a Command

Follow the procedure below to create and run a card batch file using a command.

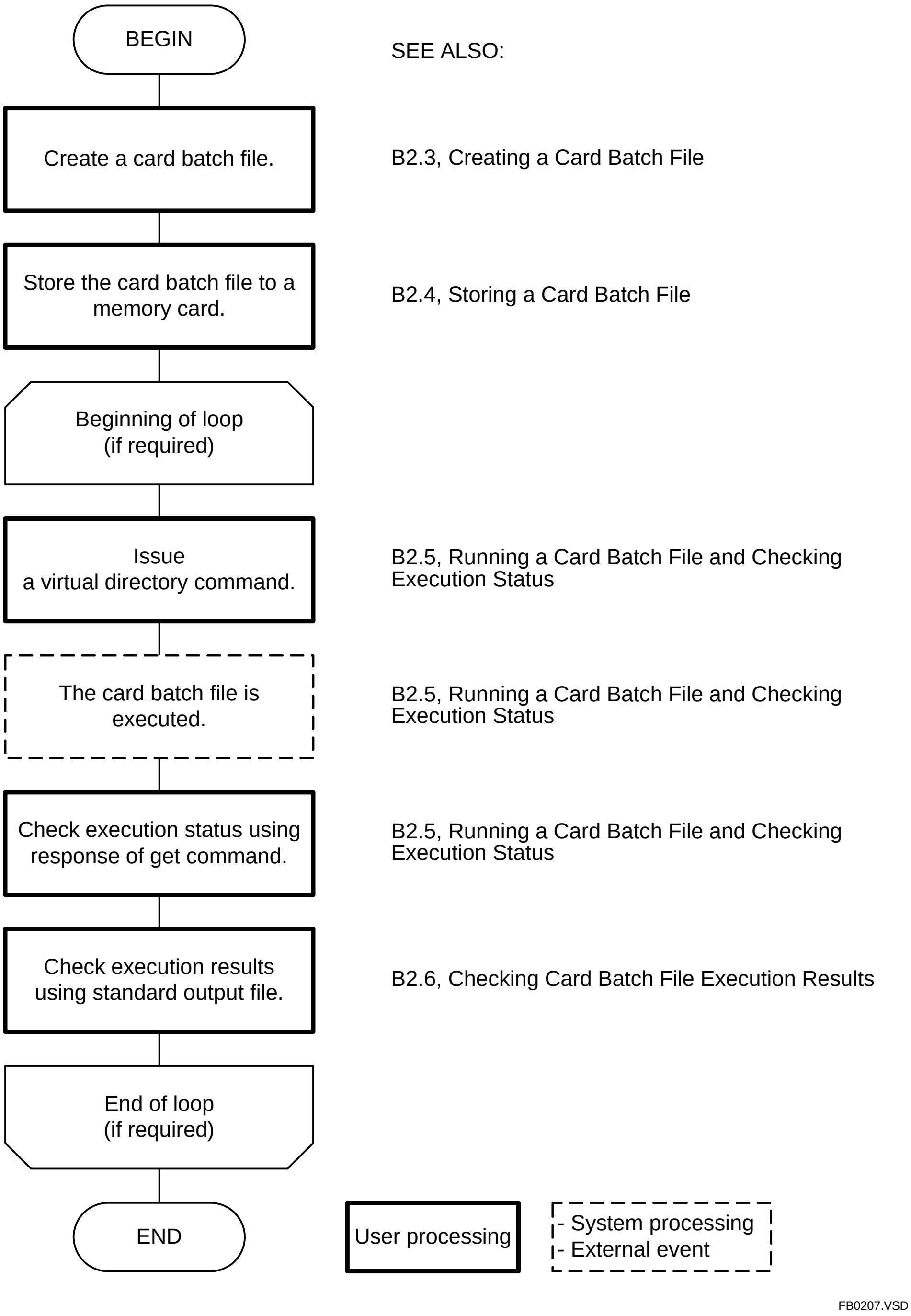

BEGIN
Create a card batch file.
Store the card batch file to a memory card.
Beginning of loop (if required)
Issue a virtual directory command.
The card batch file is executed.
Check execution status using response of get command.
Check execution results using standard output file.
End of loop (if required)
END
SEE ALSO:
B2.3, Creating a Card Batch File
B2.4, Storing a Card Batch File
B2.5, Running a Card Batch File and Checking Execution Status
B2.5, Running a Card Batch File and Checking Execution Status
B2.5, Running a Card Batch File and Checking Execution Status
B2.6, Checking Card Batch File Execution Results
User processing
- System processing
- External event
FB0207.VSD

Figure B2.2.4 Running a Card Batch File using a Command

B2.3 Creating a Card Batch File

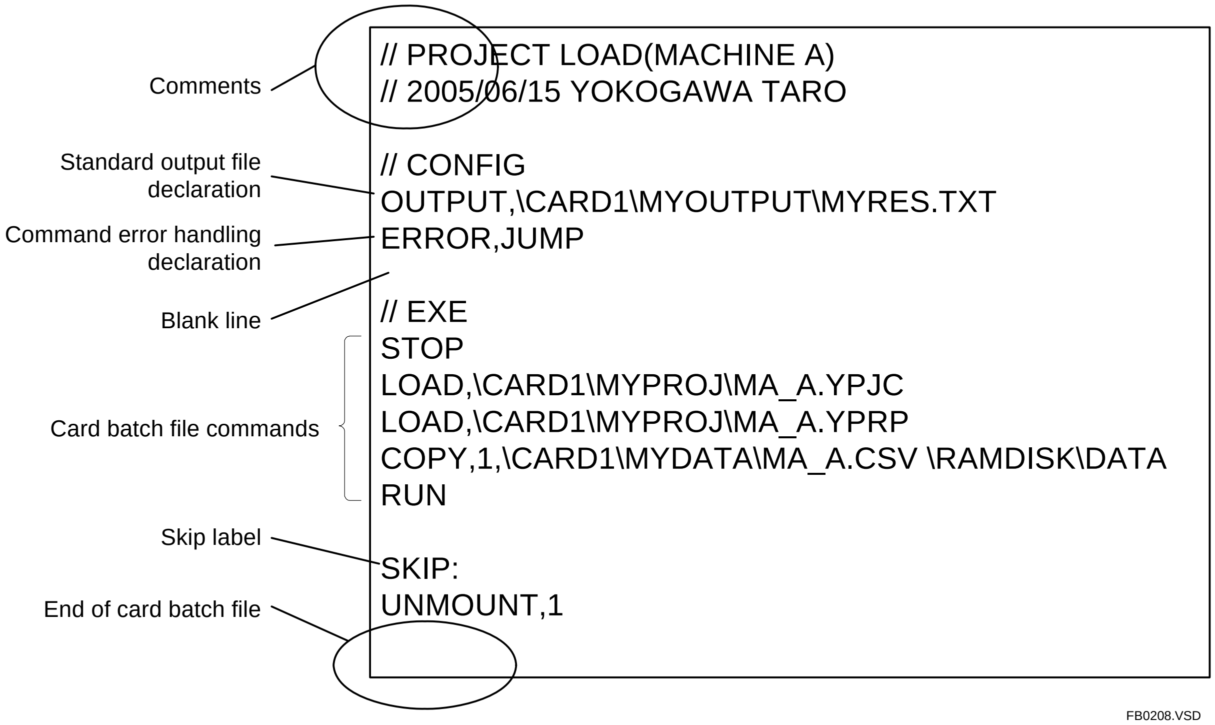
This section describes how to create a card batch file.

■ Writing a Card Batch File

You can specify the following elements in a card batch file:

- Standard output file declaration
- Command error handling declaration
- Card batch file commands
- Comments
- Skip labels
- Blank lines
- End of card batch file
- File pathnames

An example card batch file is given below.



```
// PROJECT LOAD(MACHINE A)
// 2005/06/15 YOKOGAWA TARO

// CONFIG
OUTPUT,\CARD1\MYOUTPUT\MYRES.TXT
ERROR,JUMP

// EXE
STOP
LOAD,\CARD1\MYPROJ\MA_A.YPJC
LOAD,\CARD1\MYPROJ\MA_A.YPRP
COPY,1,\CARD1\MYDATA\MA_A.CSV \RAMDISK\DATA
RUN

SKIP:
UNMOUNT,1
```

FB0208.VSD

Figure B2.3.1 Sample Card Batch File

● Standard output file declaration

The card batch file function automatically creates a specified standard output file and outputs execution results to the file. Specify a file pathname for the standard output file before coding card batch file commands using the following syntax:

```
OUTPUT, file pathname
  (file pathname = the pathname of the standard output file to
   be created)
```

If this declaration is omitted, the function creates a default standard output file in the same directory as the executed card batch file with the following filename:

```
Card batch file name_timestamp.RES
  (card batch file name = the name of the executed card batch
   file)
  (timestamp = YYYYMMDDHHMMSS)
```

Any existing file with the same pathname as the standard output file is overwritten.

● Command error handling declaration

You can specify whether control should proceed to the next line or jump to a skip label when a card batch file command exits with error. By default, control proceeds to the next line. The command error handling declaration must be made before card batch file commands using the following syntax:

```
ERROR, error handling
  (error handling = RUN (proceed to the next line) or JUMP (jump
   to a skip label))
```

If two or more skip labels are specified, control jumps to the first matching skip label in a forward search towards the file end. If no skip label is specified, execution of the card batch file ends if a command exits with error.

● Card batch file commands

Enter one card batch file command per line.

If a line is longer than 255 bytes, a syntax error is reported and execution of the card batch file is aborted.

SEE ALSO

For details on card batch file commands, see Section B2.8, "Specifications of Card Batch File Commands".

● Comments

Any text beginning with two slash characters ("//") until end-of-line is interpreted as comment text.

● Skip labels

You can define one or more skip labels in a card batch file. Once defined, a skip label can be specified in a command error handling declaration as a jump destination in the event a card batch file command exits with error. A skip label may be followed by other command lines. To define a skip label, use the following syntax:

```
SKIP:
```

● Blank lines

Blank lines are ignored.

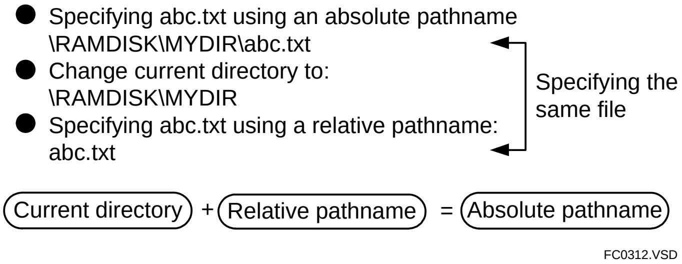
● End of card batch file

The execution of a card batch file ends when it reaches the file end.

● File pathnames

Relative pathname and absolute pathname

Both relative pathnames and absolute pathnames can be used with the card batch file function. Relative pathnames are pathnames relative to the current directory.



- Specifying abc.txt using an absolute pathname
  \RAMDISK\MYDIR\abc.txt
- Change current directory to:
  \RAMDISK\MYDIR
- Specifying abc.txt using a relative pathname:
  abc.txt

Figure B2.3.2 Relative Pathname and Absolute Pathname

Current directory

Each card batch file uses a current directory value, which applies only within the card batch file during its execution, and is independent of the current directory of FTP, file system instructions or other card batch files. The current directory defaults to "\RAMDISK" when the card batch file begins execution. To change the current directory, use the Change Directory (CD) command. You may not change the current directory to any subdirectory of the \VIRTUAL directory.

TIP

Deleting or moving a directory designated as the current directory does not generate an error. However, you must redefine the current directory in this case.

B2.4 Storing a Card Batch File

■ Naming a Card Batch File

● Execution by a trigger

The card batch file function distinguishes card batch files and their associated execution triggers by their file names as shown in the table below.

Table B2.4.1 Card Batch File Names and Associated Execution Triggers

File Names	Execution Trigger
CBAT_STA.BAT	START (Startup event)
CBAT_RUN.BAT	RUN (Run Program event)
CBAT_STOP.BAT	STOP (Stop Program event)
CBAT_MOUNT.BAT	MOUNT (Mount Memory Card event)
CBAT_ERR.BAT	ERROR (Moderate failure)
CBAT_ALARM.BAT	ALARM (Minor failure)

SEE ALSO

For details on triggers, see Section B2.5, "Running a Card Batch File and Checking Execution Status".

● Execution using a command

You can assign any name to a card batch file, and then execute the file by specifying its filename in a virtual directory command.

■ Location for Card Batch Files

● Execution by a trigger

You must store the card batch file in the root directory of memory card CARD1. Otherwise, it will not be executed. The root directory is “\CARD1” for FA-M3 and “x:\” for a PC where x denotes a drive name of that PC.

You can store card batch files with different file names together in the root directory so that each file can be automatically executed by its associated execution trigger.

Table B2.4.2 Location for Card Batch Files (to be executed by a trigger)

Directory	Remarks
\CARD1	You must store the file in the directory itself but not in its subdirectory.

● Execution using a command

You must store the card batch file in the root directory of memory card CARD1 or the RAM disk. Otherwise, it will not be executed. In FA-M3, the root directory is “\CARD1” for a memory card and “\RAMDISK” for the RAM disk. For a PC, the root directory is “x:\” where x denotes a drive name of that PC.

Table B2.4.3 Location for Card Batch Files (to be executed using a command)

Directory	Remarks
\CARD1	You must store the file in the directory itself but not in its subdirectory.
\RAMDISK	You must store the file in the directory itself but not in its subdirectory

B2.5 Running a Card Batch File and Checking Execution Status

This section describes how to run a card batch file and check the execution status.

■ Running a Card Batch File

● Execution by a trigger

Store the card batch file in the root directory (\CARD1) of memory card CARD1 and it will be automatically executed when the trigger associated with its filename occurs.

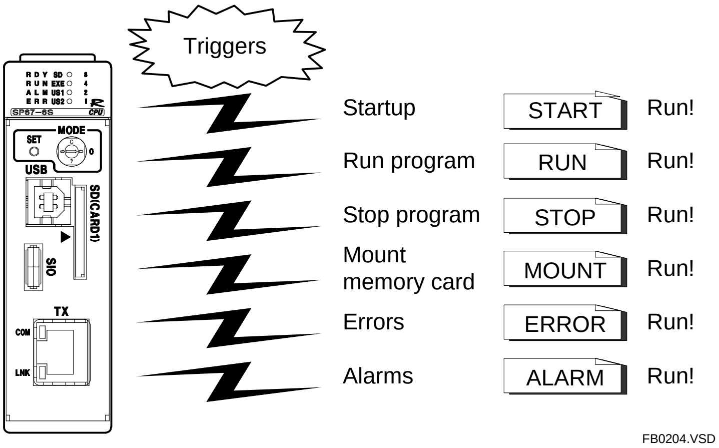


Figure B2.5.1 Execution by a Trigger

The table below lists and describes the execution triggers with their associated card batch file names.

Table B2.5.1 List of Execution Triggers for Card Batch Files

Execution Trigger	Card Batch File Name	Description
START (Startup event)	CBAT_STA.BAT	A card batch file associated with the START trigger is automatically executed when the module is switched on or reset. The module completes execution of the card batch file before switching its operating mode (Run mode or Stop mode) according to the selected boot mode.
RUN (Run Program event)	CBAT_RUN.BAT	A card batch file associated with the RUN trigger is automatically executed when the operating mode is switched from Stop mode to Run or Debug mode. Execution of the card batch file is completed before mode switching is completed (and the RUN LED lights up). Switching from Debug mode to Run mode does not trigger execution of the card batch file. The tool service and the higher-level link service are suspended during execution of the card batch file.
STOP (Stop Program event)	CBAT_STOP.BAT	A card batch file associated with the STOP trigger is automatically executed when the operating mode is switched from Run or Debug mode to Stop mode after the RUN LED goes out. If the switch to Stop mode is caused by an error, 1) A card batch file associated with the ERROR trigger is executed before 2) A card batch file associated with the STOP trigger.
MOUNT (Mount Memory Card event)	CBAT_MOUNT.BAT	A card batch file associated with the MOUNT trigger is automatically executed when a memory card is inserted or when a ladder program executes a Mount instruction after the memory card is unmounted. The card batch file will not be executed at power on or module reset.
ERROR (Moderate failure)	CBAT_ERR.BAT	A card batch file associated with the ERROR trigger is automatically executed when a moderate failure is generated. Moderate and minor failures are defined in the configuration under operation control, which defines whether to stop execution (for a moderate failure) or continue execution (for a minor failure) after each type of error. If a program is switched from Run mode to Stop mode due to an error, 1) A card batch file associated with the ERROR trigger is executed before 2) A card batch file associated with the STOP trigger.
ALARM (Minor failure)	CBAT_ALARM.BAT	A card batch file associated with the ALARM trigger is automatically executed when a minor failure (or alarm) is generated. Alarms due to a momentary power failure, however, will not trigger its execution. Moderate and minor failures are defined in the configuration under operation control, which defines whether to stop execution (for a moderate failure) or continue execution (for a minor failure) after each type of error.

CAUTION

The tool service and higher-level link service are suspended during execution of a card batch file executed by a RUN trigger. In this case, the tool service or higher-level link service may generate a timeout error if the execution of the card batch file takes too long.

● Execution using a command

You can use a virtual directory command to run a card batch file stored on memory card CARD1 or the RAM disk by specifying its file name. For this purpose, the card batch file must be stored directly under \CARD1 on a memory card or directly under \RAMDISK on the RAM disk.

The virtual directory command to be used for this purpose is the Run Card Batch File (BATGO) command.

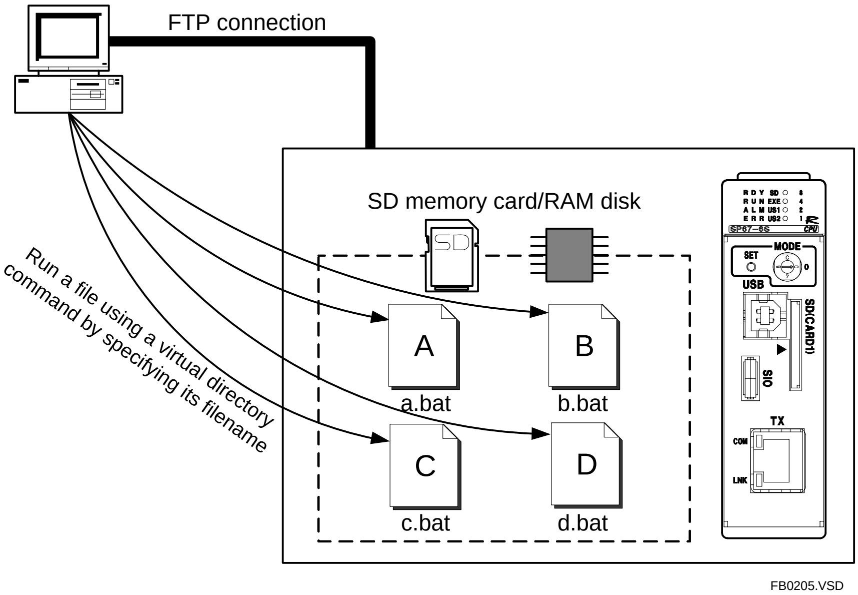


Figure B2.5.2 Execution using a Command

SEE ALSO

For details on the Run Card Batch File (BATGO) command, see Subsection 3.7.9.1, "Run Card Batch File (BATGO)" of "Sequence CPU – Network Functions User's Manual (for F3SP66-4S, F3SP67-6S)" (IM34M6P14-02E).

■ Restrictions on Card Batch File Execution

● Priority of card batch files

If a trigger occurs during the execution of a card batch file, in principle, the card batch file continues to run until it comes to an end before the card batch file that is associated with the new trigger begins running.

The only exception to this rule is when a running card batch file caused the operating mode to be switched from Stop mode to Run mode. In this case the execution of the card batch file is suspended and the card batch file associated with the RUN trigger starts running. When the execution of the second card batch file is completed, the first card batch file resumes its execution.

Table B2.5.2 Priority of Card Batch Files

		Second Trigger						
		START	RUN	STOP	MOUNT	ERROR	ALARM	NAME
First Trigger	START	–	–	–	–	–	–	–
	RUN	–	–	–	–	–	–	△
	STOP	–	–	–	–	△	△	△
	MOUNT	–	○	△	–	△	△	△
	ERROR	–	○	△	–	–	△	△
	ALARM	–	○	△	–	△	–	△
	NAME	–	○	△	△	△	△	△

Note:- ○= The file executed by the first trigger is suspended and the file associated with the second trigger starts running.
- △=The file executed by the first trigger completes execution before the file associated with the second trigger starts running.
- – =Not applicable
- START : Startup event trigger
- RUN : Run program event trigger
- STOP : Stop program event trigger
- MOUNT : Mount memory card event trigger
- ERROR : Error event trigger
- ALARM : Alarm event trigger
- NAME : Run batch file trigger

● Prohibited commands for each trigger

Some commands cannot be included in card batch files associated with certain triggers. The table below lists these restrictions.

Table B2.5.3 List of Prohibited Commands for Each Trigger

Trigger	Prohibited Commands
START	RUN, STOP, ACT, INACT
RUN	LOAD, SAVE, RUN, STOP, ACT, INACT
STOP	STOP, ACT, INACT
MOUNT	None
ERROR	None
ALARM	None
NAME	None

Note:- START : Startup event trigger
- RUN : Run program event trigger
- STOP : Stop program event trigger
- MOUNT : Mount memory card event trigger
- ERROR : Error event trigger
- ALARM : Alarm event trigger
- NAME : Run batch file trigger

■ Checking Execution Status of a Card Batch File

The execution status of a card batch file is indicated by the combined statuses of the **EXE** and **SD** LEDs as shown below.

Table B2.5.4 Execution Status Indicated by Statuses of EXE and SD LEDs

Execution Status	LED Status
Execution is in progress	- The EXE LED is lit. - The SD LED blinks at irregular intervals.
Normal exit	- The EXE LED goes out. - The SD LED goes out.*1
Error exit	- The EXE LED blinks.*2 - The SD LED goes out.*1

*1: If the SD memory card is being accessed not by the card batch file but by FTP or other functions, the SD LED blinks at irregular intervals.

*2: The EXE LED stops blinking and goes out when a card batch file execution exits normally or when you press and release the SET switch located on the front panel of the module within one second.

B2.6 Checking Card Batch File Execution Results

This section describes how to check for the completion of a card batch file execution and the execution results.

■ Checking for Completion of Card Batch File Execution

● Execution by a trigger

During execution, the **EXE** LED is lit. When the card batch file exits normally, the **EXE** LED goes out. If the card batch file exits with error, the **EXE** LED blinks.

TIP

When a card batch file exits with error, the **EXE** LED blinks and goes off when:

- another card batch file is executed; or
- a user presses and releases the SET switch within 1 second.

● Execution using a command

When a get command, which has issued a Run Batch File (BATGO) command, completes execution, the card batch file completes execution too. You can check for completion of the card batch file execution using the **EXE** LED in the same way as a card batch file executed by a trigger.

If the get command exits normally:

The card batch file exits normally.

If the get command exits with error:

The card batch file exits with error.

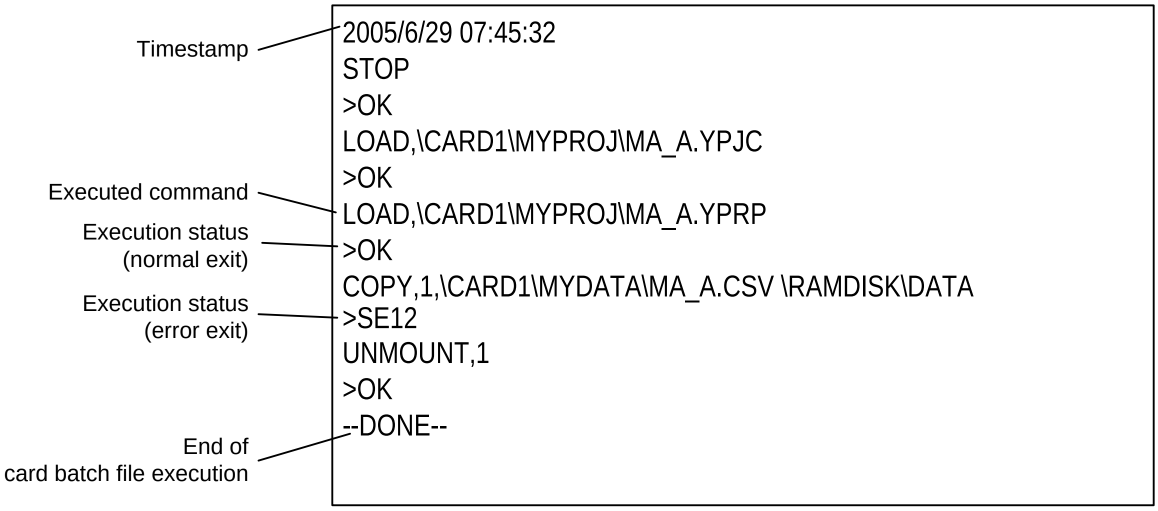
■ Checking Card Batch File Execution Results

● Execution by a trigger

To check the results of a card batch file execution, read the contents of the standard output file after a card batch file has completed execution. The standard output file contains a timestamp at the beginning of execution, followed by a list of executed card batch commands together with their execution statuses (OK for successful execution or an error code in the case of an execution error), with one command on each line.

You can find the standard output file using the file pathname specified in a standard output file declaration in the card batch file if a declaration is present. If the output file declaration is omitted, the standard output file will be stored in the same directory as the card batch file with the file name given by “<card batch file name>_timestamp.RES".

A sample standard output file is shown below.



```
2005/6/29 07:45:32
STOP
>OK
LOAD,\CARD1\MYPROJ\MA_A.YPJC
>OK
LOAD,\CARD1\MYPROJ\MA_A.YPRP
>OK
COPY,1,\CARD1\MYDATA\MA_A.CSV \RAMDISK\DATA
>SE12
UNMOUNT,1
>OK
--DONE--
```

Timestamp
Executed command
Execution status
(normal exit)
Execution status
(error exit)
End of
card batch file execution

FB0211.VSD

Figure B2.6.1   Sample Standard Output File

● Execution using a command

To check the results of a card batch file execution, read the contents of the standard output file after a card batch file has completed execution. The standard output file contains a timestamp at the beginning of execution, followed by a list of executed card batch commands together with their execution statuses (OK for successful execution or an error code in the case of an execution error), with one command on each line.

You can find the standard output file using the file pathname specified in a standard output file declaration in the card batch file if a declaration is present.

The get command, which has issued the Run Card Batch File (BATGO) command, also gets a standard output file if the card batch file exits normally. If the card batch file exits with error, the get command exits with error without getting the standard output file.

■ Card Batch File Command Response Messages

SEE ALSO

For details on card batch file command response messages, read "■ Errors and Events of Smart Access Functions Logged to Function Output” in Section B3.1, "Errors and Events of Smart Access Functions".

B2.7 Special Relays and Special Registers

There are no special relays or special registers related to the card batch file function.

B2.8 Specifications of Card Batch File Commands

This section describes the specifications of card batch file commands.

B2.8.1 File/Device Conversion Commands

Table B2.8.1 List of File/Device Conversion Commands

Command Name	Command Mnemonic	Function
Convert CSV File to Device	F2DCSV	Converts a CSV formatted file into device data.
Convert Device to CSV File	D2FCSV	Converts device data into a CSV formatted file.
Convert Binary File to Device	F2DBIN	Converts a binary file into device data.
Convert Device to Binary File	D2FBIN	Converts device data into a binary file.

Table B2.8.2 Terminology Description for File/Device Conversion Commands

Term	Description
Binary file	A file containing binary data, with no delimiters.
CSV formatted file	A text file in which ASCII coded data elements are delimited by comma (,) characters or TAB characters. A CSV file can be displayed directly in Excel. Conversely, an Excel file can be converted to a CSV formatted file with some limitations. A newline is also considered as a delimiter. Beware that a newline code and a delimiter character are combined and treated as one field.
Field	A field is one data element in a CSV formatted file.
Record	A record (one line) in a CSV formatted file is delimited by a newline code. One record contains 1 to n fields.

B2.8.1.1 Convert CSV File to Device (F2DCSV)

Converts data in CSV formatted file to binary data and writes the data to contiguous devices.

■ Syntax

Table B2.8.3 Command Specifications

Command Part	Syntax
Command	F2DCSV
Parameters [*1]	First device for writing (ASCII) [device name]
	No. of fields to be read (ASCII) [-1 = Until file end 0- 4194304 (if device unit = bit) 0- 524288 (if device unit = byte) 0- 262144 (if device unit = word) 0- 131072 (if device unit = long word)]
	Field representation type (ASCII) [0 = Decimal 1 = Hexadecimal 2 = Floating-point representation A ([-]d.dddd e[+/-]ddd form) 3 = Floating-point representation B ([-]dddd.dddd form)]
	Device unit (ASCII) [0 = Bit 1 = Byte 2 = Word 3 = Long word]
	Sign extension (ASCII) [0 = Pad with zeros 1 = Extend sign]
	Delimiter option (ASCII) [0 = Comma (,) 1 = TAB]
	Newline option (ASCII) [0 = CRLF 1 = LF]
	Write limit in words (ASCII) [1-262144 (words)]
	Input file pathname (ASCII) [126 bytes max.]

*1 : Delimit the command name and each parameter using a comma character (,).

Table B2.8.4 Valid Devices for the First Device for Writing Parameter

X	Y	I	E	L	M	T□	C□	D	B	W	Z	R	V
	✓	✓	✓	✓		✓	✓	✓	✓	✓		✓	✓

Note: - TP = timer current value
- CP = counter current value

Command Line:

```
command,parameter(1),parameter(2),...,parameter(n)
```

■ Example

This sample command reads the CSV formatted file named "\CARD1\MYDATA\data012.csv", which contains field data stored in decimal representation, and writes the number of required fields (128 words max.) as word data to devices starting from the specified first device (B2001) for writing. It assumes that the file uses CRLF as newline.

```
F2DCSV,B2001,-1,0,2,1,0,0,128,\CARD1\MYDATA\data012.csv
```

■ Reply

Table B2.8.5 Reply Messages

Reply Message	Code	Description
OK FIELD NUM. = xxxx	SE00	Normal exit "xxxx" indicates the number of fields processed.
Other messages	SE01,…	Error reply message

Note: Reply message is written to the standard output file.

SEE ALSO

For more details on reply messages of card batch file commands, see "■ Errors and Events of Smart Access Functions Logged to Function Output" of Section B3.1, "Errors and Events of Smart Access Functions".

■ Function

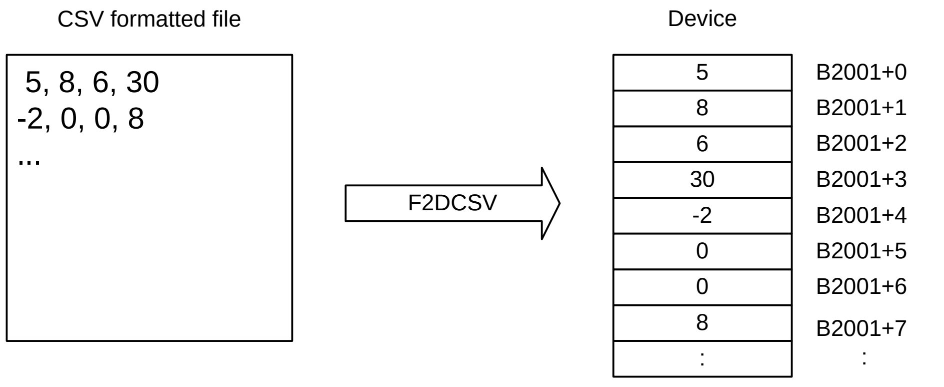
Converts data in CSV formatted file to binary data and writes the data to contiguous devices.

- Text in decimal, hexadecimal or floating-point representation can be converted to device data. Floating-point representation is converted to IEEE single-precision floating-point representation.

  Decimal ("-128" to "255", "-32768" to "65535", "-2147483648" to "4294967295")

  Hexadecimal ("0x0" to "0xFFFFFFFF", "0" to "FFFFFFFF")

  Floating-point ([-]d.dddd e[+/-]ddd, [-]dddd.dddd, Infinite "-INF"/"+INF")
- Available device unit options are bit, byte, word and long word. You can also specify whether to perform sign extension.
- Available field delimiter options are the comma (,) and Tab characters.
- Comments can be included in the file. If a field begins with a double-quote ("), single-quote ('), two slashes (//), or a slash and an asterisk (/*), the instruction skips over all characters up to a delimiter character or newline.
- Newline can be specified as CRLF (standard for Windows) or LF.



Note: Device numbers and conversion method shown are examples.

FB0212.VSD

Figure B2.8.1 CSV Formatted File to Device Conversion

TIP

Reading of File

- If end-of-file is encountered before the required number of fields is read, execution ends without error.
- A newline ends a record, and thus always ends a field.
- Within a field, any and all space characters preceding the data string are ignored, but any space character following the data string results in a field conversion error.
- '//' and other comment mark characters must always be coded at the beginning of a field. Otherwise, a conversion error will be generated.
- If NULL or other invalid binary code is encountered, execution ends with a file interpretation error.

Conversion Error and Interpretation Error

- If a conversion error is detected, 0 is written to the device. If a conversion error is detected in a field during conversion, an error is generated but processing continues.
- When the converted numeric value of a field exceeds the range of the device unit, a conversion error is generated.
- Non-numeric representation "NaN" of D2FCSV generates a conversion error.

Data Conversion and Writing to Device

- You can specify to pad with '0's or extend the sign when the converted value of a field is smaller than the size of the device unit.
- If the device unit is specified as bit, 0 is stored for a zero value while 1 is stored for any other value.
- If you specify the field representation type as floating-point representation, you must specify the device unit as long word.
- Writing to device spans multiple scan cycles.

B2.8.1.2 Convert Device to CSV File (D2FCSV)

Converts device data to text and outputs a CSV formatted file.

■ Syntax

Table B2.8.6 Command Specifications

Command Part	Syntax
Command	D2FCSV
Parameters *1	First device for reading (ASCII) [device name]
	Device unit (ASCII) [0 = bit 1 = byte 2 = word 3 = long word]
	No. of data units to be read (ASCII) [0 − 4194304 (if device unit = bit) 0 − 524288 (if device unit = byte) 0 − 262144 (if device unit = word) 0 − 131072 (if device unit = long word)]
	Field representation type (ASCII) [0 = Decimal 1 = Hexadecimal 2 = Floating-point representation A ([-]d.dddd e[+/-]ddd form) 3 = Floating−point representation B ([-]dddd.dddd form)]
	Field length (ASCII) [0 = automatic (as required after conversion) 1-13 = fixed field length in characters]
	Field space handling (ASCII) [*2 0 = Pad with spaces 1 = Pad with zeros]
	Delimiter option (ASCII) [0 = comma (,) 1 =TAB]
	Newline option (ASCII) [0 = CRLF 1 = LF]
	Newline insertion position (ASCII) [0 = Do not insert newline 1 − 32767 = Insert newline after n fields]
	Output file pathname (ASCII) [126 bytes max.]

*1: Delimit the command name and each parameter using a comma character (,).

*2: If the field length is specified as 0 (automatic), this parameter is ignored but a valid dummy value (say, 0) must still be specified.

Table B2.8.7 Valid Devices for the First Device for Reading Parameter

X	Y	I	E	L	M	T□	C□	D	B	W	Z	R	V
✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Note: - TP = timer current value; TS = timer preset value
- CP = counter current value; CS = counter preset value

Command Line:

```
command,parameter(1),parameter(2),...,parameter(n)
```

■ Example

This sample command writes device data starting from B2001 to a CSV formatted file in the current directory according to the conditions given below.

-	Device unit	Word
-	No. of data units to be read	128
-	Field representation type	Decimal
-	Field length	Automatic
-	Field space handling	Pad with spaces
-	Delimiter option	Comma
-	Newline option	CRLF
-	Newline insertion position	4

```
D2FCSV,B2001,2,128,0,0,0,0,0,4,data012.csv
```

■ Reply

Table B2.8.8 Reply Messages

Reply Message	Code	Description
OK DATA NUM. = xxxx	SE00	Normal exit "xxxx" indicates the number of data units processed.
Other messages	SE01,…	Error reply message

Note: Reply message is written to the standard output file.

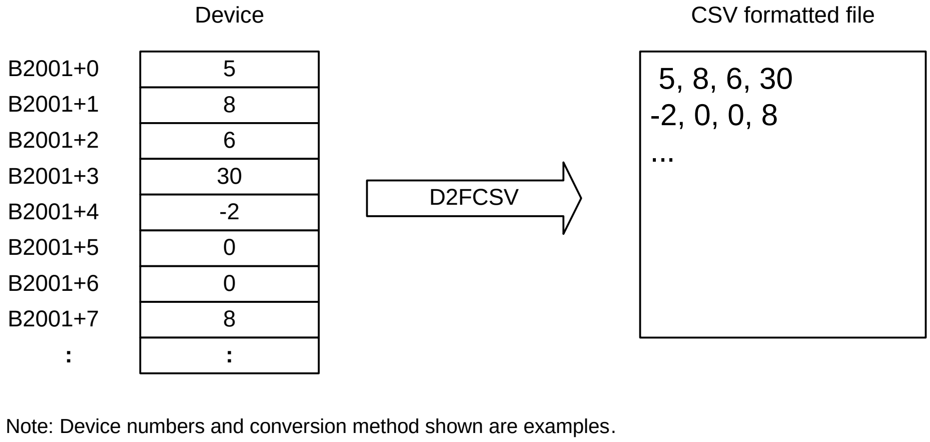
SEE ALSO

For more details on reply messages of card batch file commands, see "■ Errors and Events of Smart Access Functions Logged to Function Output" of Section B3.1, "Errors and Events of Smart Access Functions".

■ Function

Converts device data to text and outputs a CSV formatted file.

- Available device unit options for reading are bit, byte, word and long word.
- Device data can be converted to text in decimal, hexadecimal or floating-point representation after reading.
  Decimal ("0" to "1", "-128" to "127", "-32768" to "32767", "-2147483648" to "2147483647")
  Hexadecimal ("0" to "FFFFFFFF")
  Floating-point ([-]d.dddd e[+/-]ddd, [-]dddd.dddd, infinity "-INF" or "+INF", non-numeric "NaN")
- You can specify the field length in characters for text conversion.
- You can specify whether to pad with space characters or pad with zeros if the converted text is shorter than the specified field length.
- Available field delimiter options are the comma (,) and Tab characters.
- Newline can be specified as CRLF (standard for Windows) or LF.
- The number of fields in one record (from line beginning to line end) can be specified.



FB0213.VSD

Figure B2.8.2 Device to CSV Formatted File Conversion

TIP

Reading of Device Data

- Reading of data from devices spans multiple scan cycles.
- If you specify the field representation type as floating-point representation, you must specify the device unit as long word for devices storing IEEE single-precision floating-point numbers.

Conversion Error and Interpretation Error

- If a conversion error occurs, "ERR" is written to the field. If a conversion error is detected for a field during conversion, an error is generated but processing continues.
- If the converted text string is longer than the specified field length, a conversion error is generated.

Data Conversion

- If the field representation type is specified as decimal, the sign is included in the output digit count.
- In floating-point representation B, there are always 6 digits after the decimal point. Numeric values smaller than 0.000001 are rounded to 0.

B2.8.1.3 Convert Binary File to Device（F2DBIN）

Converts data in binary file and writes the data to contiguous devices using the specified data unit.

■ Syntax

Table B2.8.9 Command Specifications

Command Part	Syntax
Command	F2DBIN
Parameters [*1]	First device for writing (ASCII) [device name]
	No. of data units to be read (ASCII) [-1 = Until file end 0 - 4194304 (if device unit = bit) 0 - 524288 (if device unit = byte) 0 - 262144 (if device unit = word) 0 - 131072 (if device unit = long word)]
	Data unit (ASCII) [1 = byte 2 = word 3 = long word]
	Device unit (ASCII) [0 = bit 1 = byte 2 = word 3 = long word]
	Sign extension (ASCII) [0 = Pad with zeros 1 = Extend sign]
	Write limit in words (ASCII) [1－ 262144 (words)]
	Input file pathname (ASCII) [126 bytes max.]

*1 : Delimit the command name and each parameter using a comma character (,).

Table B2.8.10 Valid Devices for the First Device for Writing parameter

X	Y	I	E	L	M	T□	C□	D	B	W	Z	R	V
	✓	✓	✓	✓		✓	✓	✓	✓	✓		✓	✓

Note: - TP = timer current value
- CP = counter current value

Command Line:

```
command,parameter(1),parameter(2),...,parameter(n)
```

■ Example

This sample command reads all data (128 words max.) contained in the binary file named "\CARD1\NEWPRODUCT\data012.bin" in word units and writes the word data to devices starting from B2001.The sign extension parameter has no significance in this example.

```
F2DBIN,B2001,-1,2,2,1,128,¥CARD1¥NEWPRODUCT¥data012.bin
```

■ Reply

Table B2.8.11 Reply Messages

Reply Message	Code	Description
OK DATA NUM. = xxxx	SE00	Normal exit "xxxx" indicates the number of data units processed.
Other messages	SE01,…	Error reply message

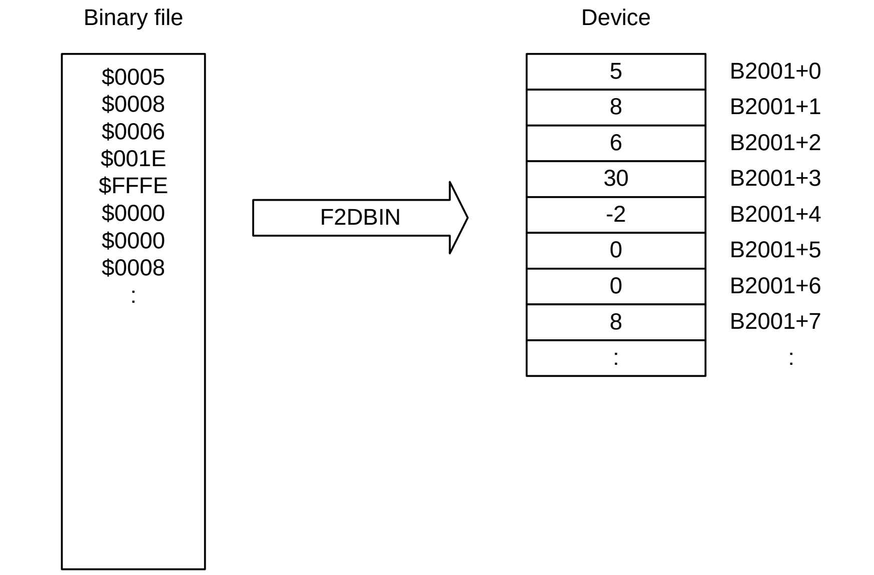
Note: Reply message is written to the standard output file.

SEE ALSO

For more details on reply messages of card batch file commands, see "■ Errors and Events of Smart Access Functions Logged to Function Output" of Section B3.1, "Errors and Events of Smart Access Functions".

■ Function

Converts data in binary file and writes the data to contiguous devices using the specified data unit.



Note: Device numbers and conversion method shown are examples.

FB0214.VSD

Figure B2.8.3 Binary File to Device Conversion

TIP

- If a conversion error occurs, 0 is written to the device. If a conversion error is detected during conversion, an error is generated but processing continues.
- If the specified data unit is larger than the specified device unit, a conversion error is generated.
- If the device unit is specified as bit, 0 is stored for a zero value while 1 is stored for any other value.
- If the specified data unit is smaller than the specified device unit, you can specify to pad with '0's or extend the sign.
- If end-of-file is encountered before the required number of data units are read, execution ends without error.
- Writing to device spans multiple scan cycles.

B2.8.1.4 Convert Device to Binary File（D2FBIN）

Converts device data to a binary file.

■ Syntax

Table B2.8.12 Command Specifications

Command Part	Syntax
Command	D2FBIN
Parameters[*1]	First device for reading (ASCII) [device name]
	No. of data units to be read (ASCII) [0 −4194304 (if device data unit = bit) 0 − 524288 (if device data unit = byte) 0 − 262144 (if device data unit = word) 0 − 131072 (if device data unit = long word)]
	Device data unit (ASCII) [0 = bit 1 = byte 2 = word 3 = long word]
	File data unit (ASCII) [1 = byte 2 = word 3 = long word]
	Sign extension (ASCII) [0 = Pad with zeros 1 = Extend sign]
	Output file pathname (ASCII) [126 bytes max.]

*1 : Delimit the command name and each parameter using a comma character (,).

Table B2.8.13 Valid Devices for the First Device for Reading Parameter

X	Y	I	E	L	M	T□	C□	D	B	W	Z	R	V
✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Note:
- TP = timer current value; TS = timer preset value
- CP = counter current value; CS = counter preset value

Command Line:

```
command,parameter(1),parameter(2),...,parameter(n)
```

■ Example

This sample command writes device data starting from B2001 to a binary file named "data012.bin" in the current directory according to the conditions given below.

- No. of data units to be read	128
- Device data unit	Word
- File data unit	Word
- Sign extension	Extend sign (has no significance in this example)

```
D2FBIN,B2001,128,2,2,1,data012.bin
```

■ Reply

Table B2.8.14 Reply Messages

Reply Message	Code	Description
OK DATA NUM. = xxxx	SE00	Normal exit "xxxx" indicates the number of data units processed.
Other messages	SE01,…	Error reply message

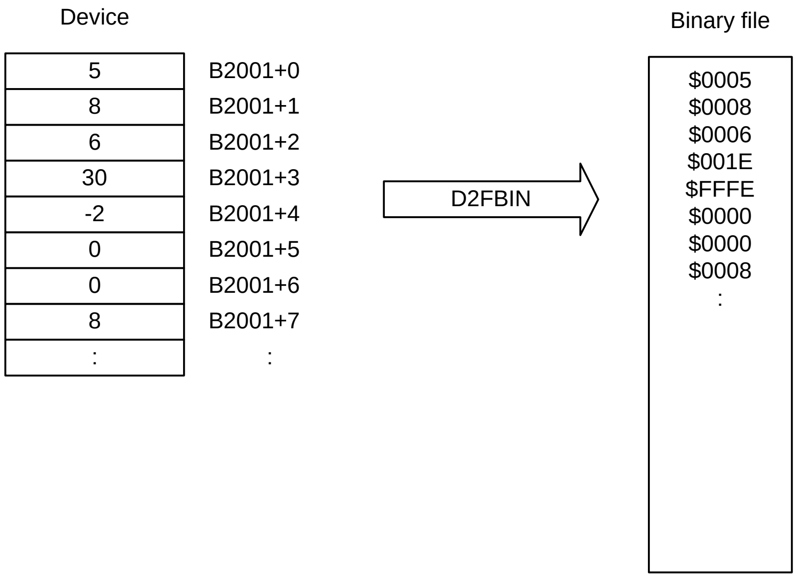
Note: Reply message is written to the standard output file.

SEE ALSO

For more details on reply messages of card batch file commands, see "■ Errors and Events of Smart Access Functions Logged to Function Output" of Section B3.1, "Errors and Events of Smart Access Functions".

■ Function

Converts device data to a binary file.



Note: Device numbers and conversion method shown are examples.

FB0215.VSD

Figure B2.8.4 Device Data to Binary File Conversion

TIP

- If a conversion error occurs, 0 is written to the file. If a conversion error is detected during conversion, an error is generated but processing continues.
- If the specified device unit is larger than the specified file unit, a conversion error is generated.
- If the specified device unit is smaller than the specified file unit, you can specify to pad with '0's or extend the sign.
- Reading of data from devices spans multiple scan cycles.

B2.8.2 Maintenance Commands

Table B2.8.15 List of Maintenance Commands

Command Name	Command Mnemonic	Description
Load Project	LOAD	Loads project and CPU properties.
Save Project	SAVE	Saves project and CPU properties.
Get Log	LOG	Gets various log files.
CPU Info	CPUINFO	Gets CPU module information.
Application Info	APINFO	Gets user application information.
Technical Support Info	TECHINFO	Gets information for technical support.
Run Mode	RUN	Switches operating mode to Run mode.
Stop Mode	STOP	Switches operating mode to Stop mode.
Activate Block	ACT	Activates a specified block. You may also activate a sensor control block.
Inactivate Block	INACT	Inactivates a specified block. You may also inactivate a sensor control block.
Reset CPU	CPURESET	Resets the CPU.
Clear Alarms	ALMCLEAR	Clears all alarms, and returns a log of cleared alarms.
Help	HELP	Gets help information on card batch file commands.

B2.8.2.1 Load Project (LOAD)

Loads a specified project file in card load format or a specified CPU property file into the internal ROM.

■ Syntax

Table B2.8.16 Command Specifications

Command Part	Syntax
Command	LOAD
Parameter	Pathname of project file in card load format or pathname of a CPU property file.

Command Line:

```
command,parameter
```

■ Example

The first sample command loads a project file in card load format named "myproj.ypjc" from the current directory; The second sample command loads a CPU property file named "myprop.yprp" from the current directory.

```
LOAD,myproj.ypjc
LOAD,myprop.yprp
```

■ Reply

Table B2.8.17 Reply Messages

Reply Message	Code	Description
OK	SE00	Normal exit
Other messages	SE01,...	Error reply message

Note: Reply message is written to the standard output file.

SEE ALSO

For more details on reply messages of card batch file commands, see "■ Errors and Events of Smart Access Functions Logged to Function Output" of Section B3.1, "Errors and Events of Smart Access Functions".

■ Function

Loads a specified project file in card load format or a specified CPU property file into the internal ROM.

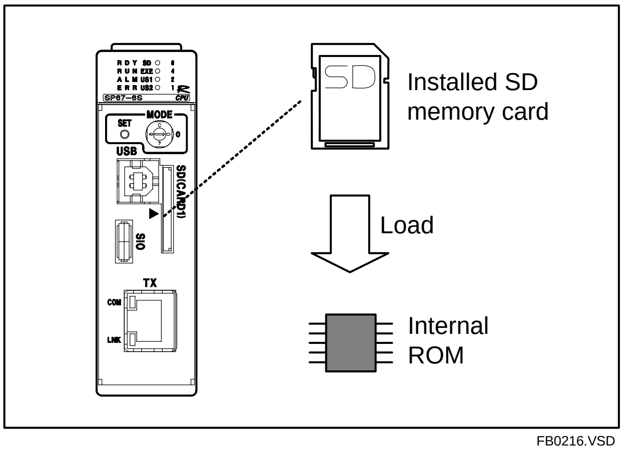


Figure B2.8.5 Load Project

Loading either a project or CPU Property file only

You can load either a project file or load a CPU property file only. In this case, existing project file or CPU property file in the internal ROM not overwritten by loading remains as is. The command identifies a project file or property file using its filename extension ("ypjc" for a project file; "yprp" for a CPU property file).

Operating mode dependency

Project loading is available only in Stop mode. CPU property loading is available in any operating mode.

The operating mode before command execution is retained after command execution.

Protection

If executable program protection is enabled for the project stored in the internal ROM, the project to be loaded from memory card CARD1 must be protected by the same password for loading to succeed. Block protection alone has no effect on project loading.

If CPU properties data stored in the internal ROM is protected with a keyword, the property file to be loaded from memory card CARD1 must be set with the same keyword for loading to succeed.

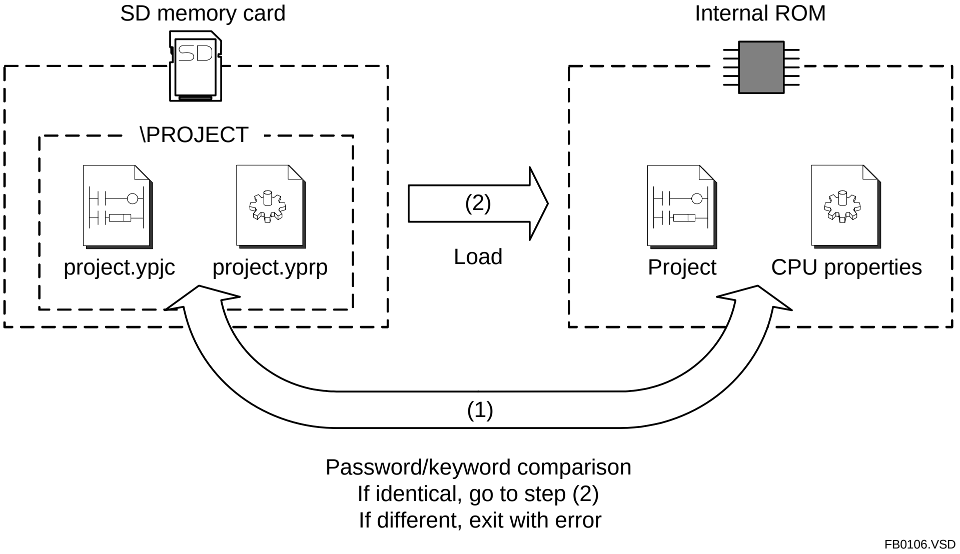


Figure B2.8.6 Protection (Load Project)

This command is prohibited by the following triggers:

RUN (Run Program) trigger

If an error occurs

The operating mode right before execution remains in effect.

The table below shows the state of the data in the internal ROM in the event of an error.

Table B2.8.18 Internal ROM Contents after an Error

Error	Data of Internal ROM
PARAMETER ERROR	The data before execution is retained.
RUN MODE ERROR	
SECURITY ERROR	
INVALID FILE (CPU property file)	The CPU property data before execution is retained.
INVALID FILE (project file)	Project programs are cleared.
FILE SYSTEM ERROR	Unpredictable

B2.8.2.2 Save Project (SAVE)

Saves project or CPU property data stored in the internal ROM.

■ Syntax

Table B2.8.19 Command Specifications

Command Part	Syntax
Command	SAVE
Parameter	Output file pathname (ASCII) [126 bytes max. File extension "ypjc" = save project File extension "yprp" = save CPU properties]

Command Line:

```
command,parameter
```

■ Example

The first sample command saves a project file in card load format to the current directory. The second command saves a CPU property file to the current directory.

```
SAVE,myproject.ypjc
SAVE,no02.yprp
```

■ Reply

Table B2.8.20 Reply Messages

Reply Message	Code	Description
OK	SE00	Normal exit
Other messages	SE01,...	Error reply message

Note: Reply message is written to the standard output file.

SEE ALSO

For more details on reply messages of card batch file commands, see "■ Errors and Events of Smart Access Functions Logged to Function Output" of Section B3.1, "Errors and Events of Smart Access Functions".

■ Function

This function saves project or CPU property data stored in the internal ROM as a file in card load format or a CPU property file. Any file of the same name in the destination directory is overwritten unless it is read-only.

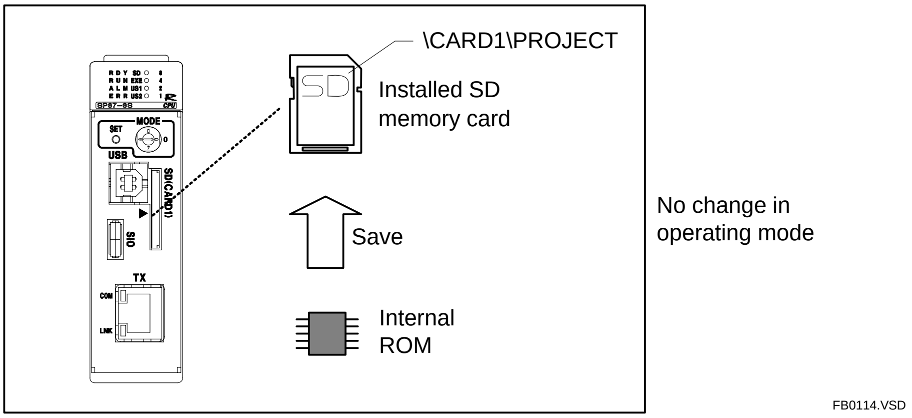


Figure B2.8.7 Save Project

The command distinguishes between saving project data and saving CPU properties data using the extension of the specified output file pathname. To get a project file, specify a filename with extension ".ypjc". To get a CPU property file, specify a filename with extension ".yprp".

TIP

You may save the data using any filename but the project name remains the same as at the time of loading.

Protection

If executable program protection is enabled for the project stored in the internal ROM, a project of the same name and protected with the same password must be present on memory card CARD1 for saving to succeed. If the passwords are different, an error is generated and the project is not saved. As the project on the memory card need not have the same content, you can simply create a dummy project with the same name and protect it using the same password beforehand.

Block protection, even if enabled, is ignored during saving.

If the CPU property data in the internal ROM is protected with a keyword, a CPU property file protected with the same keyword must reside on memory card CARD1 for saving to succeed.

For security reasons, no keyword is output to the saved CPU property file.

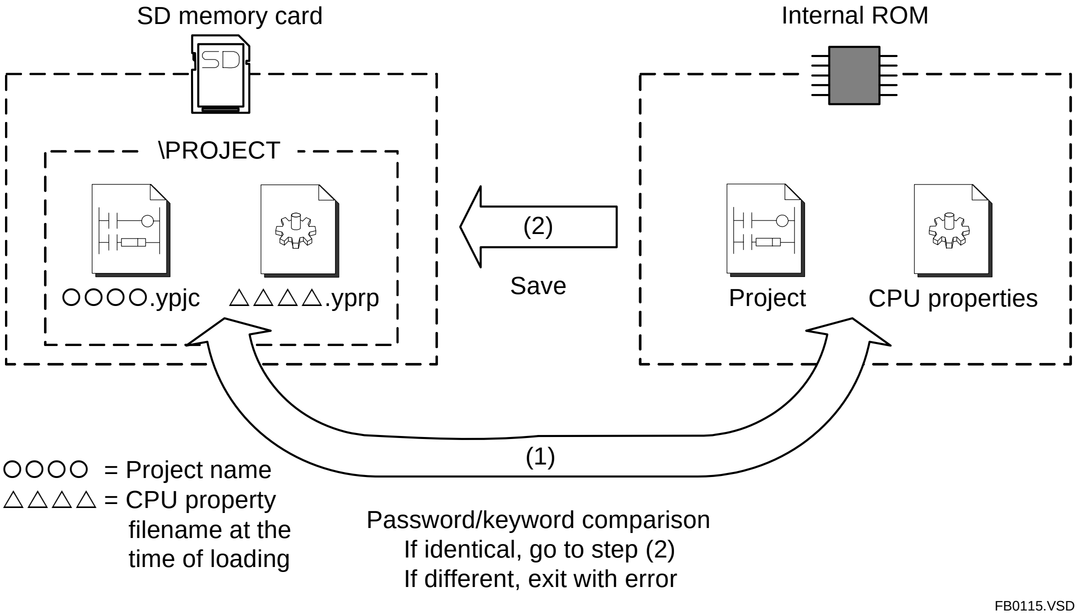

SD memory card
Internal ROM
\PROJECT
○○○○.ypjc
△△△△.yprp
(2)
Save
Project
CPU properties
(1)
○○○○ = Project name
△△△△ = CPU property filename at the time of loading
Password/keyword comparison
If identical, go to step (2)
If different, exit with error
FB0115.VSD

Figure B2.8.8 Protection (Save Project)

This command is prohibited by the following triggers:
RUN (Run Program) trigger

B2.8.2.3 Get Log (LOG)

Gets and stores log information as a text file.

■ Syntax

Table B2.8.21 Command Specifications

Command Part	Syntax
Command	LOG
Parameters	Log type [0 = System log 1 = FTP server log]
	Output file pathname

Command Line:

```
command,parameter(1),parameter(2)
```

■ Example

This sample command gets and saves the system log in a file named "syslog.txt" in the current directory.

```
LOG,0,syslog.txt
```

■ Reply

Table B2.8.22 Reply Messages

Reply Message	Code	Description
OK	SE00	Normal exit
Other messages	SE01,...	Error reply message

Note: Reply message is written to the standard output file.

SEE ALSO

For more details on reply messages of card batch file commands, see "■ Errors and Events of Smart Access Functions Logged to Function Output" of Section B3.1, "Errors and Events of Smart Access Functions".

■ Function

Gets and saves log data in a text file. You can get the following types of log data:

- System log (messages logged for error events, power on/off events, etc.)
- FTP server log (execution log of the FTP server)

B2.8.2.4 CPU Info (CPUINFO)

Gets and stores CPU information as a text file.

■ Syntax

Table B2.8.23 Command Specifications

Command Part	Syntax
Command	CPUINFO
Parameter	Output file pathname

Command Line:

```
command,parameter
```

■ Example

This sample command gets and saves CPU information in a file named "\CARD1\MYINFO\mycpuinfo.txt".

```
CPUINFO,\CARD1\MYINFO\mycpuinfo.txt
```

■ Reply

Table B2.8.24 Reply Messages

Reply Message	Code	Description
OK	SE00	Normal exit
Other messages	SE01,...	Error reply message

Note: Reply message is written to the standard output file.

SEE ALSO

For more details on reply messages of card batch file commands, see "■ Errors and Events of Smart Access Functions Logged to Function Output" of Section B3.1, "Errors and Events of Smart Access Functions".

■ Function

Gets CPU information and saves it in a text file. The table below lists the returned CPU information.

Table B2.8.25 Overview of CPU Information

CPU Information	Data[*1]	Data Range
Nameplate information	MODEL = *CPU type* SERIAL NO. = *Serial number* DATE = *Date of manufacture* MAC ID = *MAC address* FIRMWARE REV. = *Revision no.*	*CPU type* [F3SP66-4S, F3SP67-6S] *Serial number* [3 alphanumeric characters and 6 numeric characters] *Date of manufacture* [YY/MM/DD] *MAC address* [12-digit hexadecimal number] *Revision no.* [starts with R00]
Operating mode	PROGRAM MODE= *Operating mode*	*Operating mode* [0 = Stop mode 1 = Run mode 2 = Debug mode]
LED status	RDY LED = *LED status* RUN LED = *LED status* ALM LED = *LED status* ERR LED = *LED status* SD LED = *LED status* EXE LED = *LED status* US1 LED = *LED status* US2 LED = *LED status*	*LED status* [0 = Not lit 1 = Lit 2 = Blinking]
MODE switch status	MODE SW = *MODE switch value*	*MODE switch value* [0 to F]
CARD1 mount status	CARD1 MOUNT STATUS = *Mount status*	*Mount status* [0 = Unmounted 1 = Mounted]
CARD1 free space	CARD1 FREE SPACE = *Free space*	*Free space* [bytes]
CARD1 capacity	CARD1 TOTAL SIZE = *Capacity*	*Capacity* [bytes]
RAM disk free space	RAMDISK FREE SPACE = *Free space*	*Free space* [bytes]
RAM disk capacity	RAMDISK TOTAL SIZE = *Capacity*	*Capacity* [bytes]
Alarm status	*Alarm name*	*Alarm name* (the same as that displayed by the alarm monitor)
Block activation status	*Block name 1 = Activation status* : *Block name n = Activation status*	*Block name* [up to 8 ASCII characters] *Activation status* [0 = Inactive, 1 = Active]

*1: The data range of each italicized item is given in the "Data Range" column.

B2.8.2.5 Application Info (APINFO)

Returns application information (project information, configuration, I/O setup) in text format.

■ Syntax

Table B2.8.26 Command Specifications

Command Part	Syntax
Command	APINFO
Parameters	Output file pathname (ASCII) [126 bytes max.]
	Security password of executable program (ASCII) [8 bytes max.] *1

*1: This parameter is required even if the executable program is not protected. You may specify any valid dummy string in this case.

Command Line:

```
command,parameter(1),parameter(2)
```

■ Example

This sample program gets and saves application information of a project whose executable program is not protected in a file named "myapr.txt" in the current directory. It specifies a dummy password of "aaa" as the executable program is not protected.

```
APINFO,myapr.txt,aaa
```

■ Reply

Table B2.8.27 Reply Messages

Reply Message	Code	Description
OK	SE00	Normal exit
Other messages	SE01,...	Error reply message

Note: Reply message is written to the standard output file.

SEE ALSO

For more details on reply messages of card batch file commands, see "■ Errors and Events of Smart Access Functions Logged to Function Output" of Section B3.1, "Errors and Events of Smart Access Functions".

■ Function

Returns application information (project information, configuration, I/O setup) in text format. If executable program protection is enabled for the project, you must specify a valid password as a parameter.

(1) Project information

The following project information is output in text format.

Table B2.8.28 System Information – Project Information

Project Information	Data[*1]	Data Range
CPU model	CPU TYPE = *CPU type*	*CPU type* [up to 9 ASCII characters]
Name of stored project	PROJECT NAME = *Project name*	*Project name* [up to 8 ASCII characters]
Name of stored CPU property data	CPU PROPERTY NAME = *CPU property name*	*CPU property name* [up to 255 ASCII characters]
Number of stored program steps	PROGRAM STEP = *Number of steps*	*Number of steps* [0 to maximum limit for the module]
Number of component blocks	BLOCK NUM = *Number of blocks*	*Number of blocks* [1 to 1024]
Names of component blocks	BLOCK NAME 1 = *Name of block 1* : BLOCK NAME n = *Name of block n*	*Name of block n* [up to 8 ASCII characters] (n is between 1 and 1024)
Names of registered macros[*2]	MACRO NAME 1 = *Name of macro 1* : MACRO NAME n = *Name of macro n*	*Name of macro n* [up to 8 ASCII characters] (n is between 1 and 256)

*1: The data range of each italicized item is given in the "Data Range" column.
*2: Information is not stored for unregistered macros.

(2) Configuration information

Configuration information is output in text format.

(3) I/O setup information

I/O setup information for each slot of a unit is output as one set of comma-delimited elements consisting of type, number of X points, number of Y points and number of registers as shown below. I/O setup information is available for up to 8 units each consisting of up to 16 slots. Information for each unit is delimited by the newline code.

Table B2.8.29 System Information – I/O Setup Information

Unit	Sequence	Element	Format
UNIT0	1	Module type of SLOT1	4 ASCII characters
		Number of X points of SLOT1	2-digit decimal number
		Number of Y points of SLOT1	2-digit decimal number
		Number of registers of SLOT1	5-digit decimal number
		:	:
		Module type of SLOT16	4 ASCII characters
		Number of X points of SLOT16	2-digit decimal number
		Number of Y points of SLOT16	2-digit decimal number
		Number of registers of SLOT16	5-digit decimal number
UNIT1	2	Refer to the description for UNIT0.	Refer to the description for UNIT0.
UNIT2	3	Refer to the description for UNIT0.	Refer to the description for UNIT0.
UNIT3	4	Refer to the description for UNIT0.	Refer to the description for UNIT0.
UNIT4	5	Refer to the description for UNIT0.	Refer to the description for UNIT0.
UNIT5	6	Refer to the description for UNIT0.	Refer to the description for UNIT0.
UNIT6	7	Refer to the description for UNIT0.	Refer to the description for UNIT0.
UNIT7	8	Refer to the description for UNIT0.	Refer to the description for UNIT0.

B2.8.2.6 Technical Support Info (TECHINFO)

Gets information for technical support.

■ Syntax

Table B2.8.30 Command Specifications

Command Part	Syntax
Command	TECHINFO
Parameter	Output file pathname

Command Line:

```
command,parameter
```

■ Example

This sample command saves technical support information in a file named "CARD1\support.bin".

```
TECHINFO,\CARD1\support.bin
```

■ Reply

Table B2.8.31 Reply Messages

Reply Message	Code	Description
OK	SE00	Normal exit
Other messages	SE01,...	Error reply message

Note: Reply message is written to the standard output file.

SEE ALSO

For more details on reply messages of card batch file commands, see "■ Errors and Events of Smart Access Functions Logged to Function Output" of Section B3.1, "Errors and Events of Smart Access Functions".

■ Function

Gets information for technical support. The returned file is in binary format and not readable by a user. You can execute this function when asked to do so by Yokogawa technical support personnel.

This command is only available in Stop mode.

This command is prohibited by the following triggers:
RUN (Run Program) trigger

B2.8.2.7 Run Mode (RUN)

Switches the operating mode to Run mode.

■ Syntax

Table B2.8.32 Command Specifications

Command Part	Syntax
Command	RUN
Parameters	–

Command Line:

```
command
```

■ Example

This sample command switches the operating mode to Run mode.

```
RUN
```

■ Reply

Table B2.8.33 Reply Messages

Reply Message	Code	Description
OK	SE00	Normal exit
Other messages	SE01,...	Error reply message

Note: Reply message is written to the standard output file.

SEE ALSO

For more details on reply messages of card batch file commands, see "■ Errors and Events of Smart Access Functions Logged to Function Output" of Section B3.1, "Errors and Events of Smart Access Functions".

■ Function

Switches the operating mode to Run mode. No error is generated even if the module is already in Run mode.

Switching to Run mode is not allowed while edited changes are being written to the CPU module in online edit mode.

This command is prohibited by the following triggers:
START (Startup) trigger, RUN (Run Program) trigger

B2.8.2.8 Stop Mode (STOP)

Switches the operating mode to Stop mode.

■ Syntax

Table B2.8.34 Command Specifications

Command Part	Syntax
Command	STOP
Parameters	–

Command Line:

```
command
```

■ Example

This sample command switches the operating mode to Stop mode.

```
STOP
```

■ Reply

Table B2.8.35 Reply Messages

Reply Message	Code	Description
OK	SE00	Normal exit
Other messages	SE01,...	Error reply message

Note: Reply message is written to the standard output file.

SEE ALSO

For more details on reply messages of card batch file commands, see "■ Errors and Events of Smart Access Functions Logged to Function Output" of Section B3.1, "Errors and Events of Smart Access Functions".

■ Function

Switches the operating mode to Stop mode. No error is generated even if the module is already in Stop mode.

This command is prohibited by the following triggers:
START (Startup) trigger, RUN (Run Program) trigger, STOP (Stop Program) trigger

B2.8.2.9 Activate Block (ACT)

Activates a specified block.

■ Syntax

Table B2.8.36 Command Specifications

Command Part	Syntax
Command	ACT
Parameter	Block name (ASCII) [up to 8 bytes]

Command Line:

```
command,parameter
```

■ Example

This sample command activates the block named "MAINBLK".

```
ACT,MAINBLK
```

■ Reply

Table B2.8.37 Reply Messages

Reply Message	Code	Description
OK	SE00	Normal exit
Other messages	SE01,...	Error reply message

Note: Reply message is written to the standard output file.

SEE ALSO

For more details on reply messages of card batch file commands, see "■ Errors and Events of Smart Access Functions Logged to Function Output" of Section B3.1, "Errors and Events of Smart Access Functions".

■ Function

Activates a specified block. You may also activate a sensor control block. No error is generated even if the block is already running.

Block activation is not allowed in Stop mode and execute-all-blocks mode.

This command is prohibited by the following triggers:
START (Startup) trigger, RUN (Run Program) trigger, STOP (Stop Program) trigger

B2.8.2.10 Inactivate Block (INACT)

Inactivates a specified block.

■ Syntax

Table B2.8.38 Command Specifications

Command Part	Syntax
Command	INACT
Parameter	Block name (ASCII) [up to 8 bytes]

Command Line:

```
command,parameter
```

■ Example

This sample command inactivates the block named "MOTION1".

```
INACT,MOTION1
```

■ Reply

Table B2.8.39 Reply Messages

Reply Message	Code	Description
OK	SE00	Normal exit
Other messages	SE01,...	Error reply message

Note: Reply message is written to the standard output file.

SEE ALSO

For more details on reply messages of card batch file commands, see "■ Errors and Events of Smart Access Functions Logged to Function Output" of Section B3.1, "Errors and Events of Smart Access Functions".

■ Function

Inactivates a specified block. You may also inactivate a sensor control block.

No error is generated even if the block is not running.

Block inactivation is not allowed in Stop mode and execute-all-blocks mode.

This command is prohibited by the following triggers:
START (Startup) trigger, RUN (Run Program) trigger, STOP (Stop Program) trigger

B2.8.2.11 Reset CPU (CPURESET)

Resets the CPU.

■ Syntax

Table B2.8.40 Command Specifications

Command Part	Syntax
Command	CPURESET
Parameters	–

Command Line:

```
command
```

■ Example

This sample command resets the CPU.

```
CPURESET
```

■ Reply

Table B2.8.41 Reply Messages

Reply Message	Code	Description
OK yyyy/mm/dd hh:mm:ss,SW=x	SE00	Normal exit The date output indicates the time at reset. yyyy/mm/dd denotes the year, month and date. hh:mm:ss denotes the hour, minute and second. 'x' indicates the MODE switch value at the time of reset, and ranges from hexadecimal value 0 to F.
Other messages	SE01,...	Error reply message

Note: Reply message is written to the standard output file.

SEE ALSO

For more details on reply messages of card batch file commands, see "■ Errors and Events of Smart Access Functions Logged to Function Output" of Section B3.1, "Errors and Events of Smart Access Functions".

■ Function

Resets the CPU.

The MODE switch value and timestamp at the time of reset are logged to the standard output file.

CAUTION

Pay attention to the boot mode, which is determined by the MODE switch value, before resetting the CPU.

B2.8.2.12 Clear Alarms (ALMCLEAR)

Clears all alarms.

■ Syntax

Table B2.8.42 Command Specifications

Command Part	Syntax
Command	ALMCLEAR
Parameters	–

Command Line:

```
command
```

■ Example

This sample command clears all alarms.

```
ALMCLEAR
```

■ Reply

Table B2.8.43 Reply Messages

Reply Message	Code	Description
OK Error message, error code	SE00	Normal exit
Other messages	SE01,...	Error reply message

Note: Reply message is written to the standard output file.

SEE ALSO

For more details on reply messages of card batch file commands, see "■ Errors and Events of Smart Access Functions Logged to Function Output" of Section B3.1, "Errors and Events of Smart Access Functions".

■ Function

Clears all alarms. No error is generated even if no error is active when this command is executed.

The following information is returned as a reply message when the command exits normally.

- Error code
- Error message

Examples of Reply Messages

- If there was a momentary power failure:

```
OK
Momentary power failure,02-0000
```

- If there was no active alarm:

```
OK
No error.
```

SEE ALSO

For details on the meaning of error messages, see Section B3.3, "Cleared Alarm Log Messages".

To check alarm information without clearing alarms, use the CPU Info (CPUINFO) command instead.

TIP

- If the cause of an alarm persists, it would appear as if this command has failed to clear the alarm. In this case, remove the cause of the alarm and re-execute the function.
- This command does not clear alarms related to I/O comparison error.

B2.8.2.13 Help (HELP)

Returns help information on card batch file commands in a text file.

■ Syntax

Table B2.8.44 Command Specifications

Command Part	Syntax
Command	HELP
Parameter	Output file pathname (ASCII) [126 bytes max.]

Command Line:

```
command,parameter
```

■ Example

This sample command returns help information about card batch commands in a file named "cbathelp.txt".

```
HELP,cbathelp.txt
```

■ Reply

Table B2.8.45 Reply Messages

Reply Message	Code	Description
OK	SE00	Normal exit
Other messages	SE01,...	Error reply message

Note: Reply message is written to the standard output file.

SEE ALSO

For more details on reply messages of card batch file commands, see "■ Errors and Events of Smart Access Functions Logged to Function Output" of Section B3.1, "Errors and Events of Smart Access Functions".

■ Function

Returns help information on card batch file commands in a text file.

The help information includes a list of commands with their functional overview.

B2.8.3 File Operation and Disk Operation Commands

Table B2.8.46 File Operation and Disk Operation Commands

Command Name	Command Mnemonic	Function	Availability in Run Mode
Copy File	COPY	Copies one or more files.	Executable
Move File	MOVE	Moves one or more files.	Executable
Delete File	DEL	Deletes one or more files.	Executable
Make Directory	MKDIR	Creates a directory.	Executable
Remove Directory	RMDIR	Deletes a directory.	Executable
Rename File	REN	Renames a file or directory.	Executable
File Status	STAT	Gets file status information (present or absent, etc.).	Executable
Change Directory	CD	Changes the current directory.	Executable
Concatenate File	CAT	Concatenates two files.	Executable
Change File Attribute	ATRW	Changes the attribute of a file or directory.	Executable
Unmount	UNMOUNT	Unmounts a memory card.	Executable

SEE ALSO

For details on the specifications and restrictions of the file system, see Chapter C3, "File System".

B2.8.3.1 Copy File (COPY)

Copies one or more files.

■ Syntax

Table B2.8.47 Command Specifications

Command Part	Syntax
Command	COPY
Parameters	Overwrite option (ASCII) [0 = Exit with error without overwriting 1 = Overwrite file of the same name 2 = Overwrite read-only file of the same name]
	Source pathname (ASCII) [126 bytes max.]
	Destination pathname (ASCII) [126 bytes max.]

Command Line:

```
command,parameter(1),parameter(2),parameter(3)
```

■ Example

This sample command copies the file named "data1.csv" in the current directory to a file named "\CARD1\OLDDATA\data1.bak," overwriting the destination file if it exists.

```
COPY,1,data1.csv,\CARD1\OLDDATA\data1.bak
```

■ Reply

Table B2.8.48 Reply Messages

Reply Message	Code	Description
OK	SE00	Normal exit
Other messages	SE01,...	Error reply message

Note: Reply message is written to the standard output file.

SEE ALSO

For more details on reply messages of card batch file commands, see "■ Errors and Events of Smart Access Functions Logged to Function Output" of Section B3.1, "Errors and Events of Smart Access Functions".

■ Function

Copies one or more files.

Multiple files can be specified using wildcard characters.

If a wildcard character is used to copy multiple files and an error is detected during copying, execution terminates with error.

If the specified source pathname is a directory, the effect is equivalent to specifying a wildcard character ('*') to include all files stored immediately below the directory.

If a file having the same name already exists at the specified destination pathname, whether the file will be overwritten depends on the specified overwrite option.

B2.8.3.2 Move File (MOVE)

Moves one or more files.

■ Syntax

Table B2.8.49 Command Specifications

Command Part	Syntax
Command	MOVE
Parameters	Overwrite option (ASCII) [0 = Exit with error without overwriting 1 = Overwrite file of the same name 2 = Overwrite read-only file of the same name]
	Source pathname (ASCII) [126 bytes max.]
	Destination pathname (ASCII) [126 bytes max.]

Command Line:

command,parameter(1),parameter(2),parameter(3)

■ Example

This sample command moves the file named "data1.csv" in the current directory to the destination "\CARD1\OLDDATA\data1.bak". The command exits with error if a file with the same file exists at the specified destination.

```
MOVE,0,data1.csv,\CARD1\OLDDATA\data1.bak
```

■ Reply

Table B2.8.50 Reply Messages

Reply Message	Code	Description
OK	SE00	Normal exit
Other messages	SE01,...	Error reply message

Note: Reply message is written to the standard output file.

SEE ALSO

For more details on reply messages of card batch file commands, see "■ Errors and Events of Smart Access Functions Logged to Function Output" of Section B3.1, "Errors and Events of Smart Access Functions".

■ Function

Moves one or more files.

Multiple files can be specified using wildcard characters.

If a wildcard character is used to move multiple files and an error is detected during moving, execution terminates with error.

If the specified source pathname is a directory, the effect is equivalent to specifying a wildcard character ('*') to include all files stored immediately below the directory.

If a file having the same name already exists at the specified destination pathname, whether the file will be overwritten depends on the specified overwrite option.

B2.8.3.3 Delete File (DEL)

Deletes one or more files.

■ Syntax

Table B2.8.51 Command Specifications

Command Part	Syntax
Command	DEL
Parameters	Forced delete option (ASCII) [0 = Exit with error without deleting read-only files 1 = Delete read-only files]
	Target pathname (ASCII) [126 bytes max.]

Command Line:

```
command,parameter(1),parameter(2)
```

■ Example

This sample command deletes all files in the current directory with names matching wildcard pattern "data*.csv".

```
DEL,0,data*.csv
```

■ Reply

Table B2.8.52 Reply Messages

Reply Message	Code	Description
OK	SE00	Normal exit
Other messages	SE01,...	Error reply message

Note: Reply message is written to the standard output file.

SEE ALSO

For more details on reply messages of card batch file commands, see "■ Errors and Events of Smart Access Functions Logged to Function Output" of Section B3.1, "Errors and Events of Smart Access Functions".

■ Function

Deletes one or more files.

Multiple files can be specified using wildcard characters.

If a wildcard character is used to deletes multiple files and an error is detected during deletion, execution terminates with error.

If the specified target pathname is a directory, the effect is equivalent to specifying a wildcard character ('*') to include all files stored immediately below the directory.

You can specify whether to delete files with read-only file attribute using the forced delete option.

CAUTION

The wildcard character can be used to delete many files in one go. Beware of inadvertently deleting required files when using the wildcard character.

B2.8.3.4 Make Directory (MKDIR)

Creates a directory.

■ Syntax

Table B2.8.53 Command Specifications

Command Part	Syntax
Command	MKDIR
Parameter	Directory to be created (ASCII) [126 bytes max.]

Command Line:

```
command,parameter
```

■ Example

This sample command creates a directory designated by the directory pathname "\RAMDISK\OLDDATA".

```
MKDIR,\RAMDISK\OLDDATA
```

■ Reply

Table B2.8.54 Reply Messages

Reply Message	Code	Description
OK	SE00	Normal exit
Other messages	SE01,...	Error reply message

Note: Reply message is written to the standard output file.

SEE ALSO

For more details on reply messages of card batch file commands, see "■ Errors and Events of Smart Access Functions Logged to Function Output" of Section B3.1, "Errors and Events of Smart Access Functions".

■ Function

Creates a directory.

B2.8.3.5 Remove Directory (RMDIR)

Deletes a directory.

■ Syntax

Table B2.8.55 Command Specifications

Command Part	Syntax
Command	RMDIR
Parameters	Delete all option (W) [0 = No 1 = Yes]
	Pathname of directory to be deleted (ASCII) [126 bytes max.]

Command Line:

```
command,parameter(1),parameter(2)
```

■ Example

This sample command deletes directory "\CARD1\OLDDATA", including all files and subdirectories contained in the directory.

```
RMDIR,1,\CARD1\OLDDATA
```

■ Reply

Table B2.8.56 Reply Messages

Reply Message	Code	Description
OK	SE00	Normal exit
Other messages	SE01,...	Error reply message

Note: Reply message is written to the standard output file.

SEE ALSO

For more details on reply messages of card batch file commands, see "■ Errors and Events of Smart Access Functions Logged to Function Output" of Section B3.1, "Errors and Events of Smart Access Functions".

■ Function

Deletes a directory.

If the directory to be deleted is not empty, whether the files and subdirectories contained in the directory are deleted depends on the specified "Delete all option". If the specified "Delete all option" is 1, all contents of the directory are deleted. If a read-only file is encountered during deletion, execution terminates with error. If the specified "Delete all option" is 0 and the directory to be deleted is not empty, execution terminates with error without deleting the directory and its content.

B2.8.3.6 Rename (REN)

Renames a file or directory.

■ Syntax

Table B2.8.57 Command Specifications

Command Part	Syntax
Command	REN
Parameters	Old file pathname or directory pathname (ASCII) [126 bytes max.]
	New file name or directory name (ASCII) [126 bytes max.]

Command Line:

```
command,parameter(1),parameter(2)
```

■ Example

This sample command renames file "data006.dat" residing in the current directory to "data006.bin".

```
REN,data006.dat,data006.bin
```

■ Reply

Table B2.8.58 Reply Messages

Reply Message	Code	Description
OK	SE00	Normal exit
Other messages	SE01,...	Error reply message

Note: Reply message is written to the standard output file.

SEE ALSO

For more details on reply messages of card batch file commands, see "■ Errors and Events of Smart Access Functions Logged to Function Output" of Section B3.1, "Errors and Events of Smart Access Functions".

■ Function

Renames a file or directory.

For the "New filename or directory name" parameter, you must specify only a filename or directory name, without the file path.

B2.8.3.7 File Status (STAT)

Returns file status information for a specified file or specified directory.

■ Syntax

Table B2.8.59 Command Specifications

Command Part	Syntax
Command	STAT
Parameter	Target file pathname or directory pathname

Command Line:

```
command,parameter
```

■ Example

This sample command checks whether a file exists at pathname "\RAMDISK\MYFILE\data002.csv".

```
STAT,\RAMDISK\MYFILE\data002.csv
```

■ Reply

Table B2.8.60 Reply Messages

Reply Message	Code	Description
OK ATR xxxxxxxxxxxxxxxx DATE yy/dd/mm hh:mm:ss SIZE xxxxxxxxxx	SE00	Normal exit - ATR denotes the file attribute. For details on how to interpret the following numeric value, see Table B2.8.61, "File (Directory) Attribute Data". - DATE denotes the last modified time, expressed in year, month, day, hour, minute and second. - SIZE denotes the file size in bytes.
Other messages	SE01,...	Error reply message

Note: Reply message is written to the standard output file.

SEE ALSO

For more details on reply messages of card batch file commands, see "■ Errors and Events of Smart Access Functions Logged to Function Output" of Section B3.1, "Errors and Events of Smart Access Functions".

■ Function

Returns file status information for a specified file or specified directory.

File status information that can be returned includes information on whether file/directory exists, file/directory attribute, file size and last modified time.

The file attribute is returned as one data word in binary representation.

Table B2.8.61 File (Directory) Attribute Data

Bit Position	Description
0 (LSB)	0 =Read and write 1=Read-only
1	0=Non-system file 1=System file
2	0=Non-hidden file 1=Hidden file
3	Reserved
4	Reserved
5	Reserved
6	Reserved
7	Reserved
8	Reserved
9	Reserved
10	Reserved
11	Reserved
12	Reserved
13	Reserved
14	Reserved
15 (MSB)	0 =File 1 =Directory

B2.8.3.8 Change Directory (CD)

Changes the current directory.

■ Syntax

Table B2.8.62 Command Specifications

Command Part	Syntax
Command	CD
Parameter	New directory pathname (ASCII) [126 bytes max.]

Command Line:

```
command,parameter
```

■ Example

This sample command changes the current directory to "\RAMDISK\OLDDATA".

```
CD,\RAMDISK\OLDDATA
```

■ Reply

Table B2.8.63 Reply Messages

Reply Message	Code	Description
OK	SE00	Normal exit
Other messages	SE01,...	Error reply message

Note: Reply message is written to the standard output file.

SEE ALSO

For more details on reply messages of card batch file commands, see "■ Errors and Events of Smart Access Functions Logged to Function Output" of Section B3.1, "Errors and Events of Smart Access Functions".

■ Function

Changes the current directory.

This command changes the current directory of the card batch file function, which applies only to the current card batch file execution and is independent of the current directory of another batch file execution or the FTP client function.

When execution of the card batch file ends, the current directory reverts to its initial state.

The default initial current directory is "\RAMDISK".

An error is generated if the parameter is omitted.

B2.8.3.9 Concatenate File (CAT)

Concatenates two files.

■ Syntax

Table B2.8.64 Command Specifications

Command Part	Syntax
Command	CAT
Parameters	File pathname 1 (source file 1 and destination) (ASCII) [126 bytes max.]
	File pathname 2 (source file 2) (ASCII) [126 bytes max.]

Command Line:

```
command,parameter(1),parameter(2)
```

■ Example

This sample command concatenates the file named "data001.bin" and the file named "data002.bin" in the current directory, and stores the result as a file named "data001.bin".

```
CAT,data001.bin,data002.bin
```

■ Reply

Table B2.8.65 Reply Messages

Reply Message	Code	Description
OK	SE00	Normal exit
Other messages	SE01,...	Error reply message

Note: Reply message is written to the standard output file.

SEE ALSO

For more details on reply messages of card batch file commands, see "■ Errors and Events of Smart Access Functions Logged to Function Output" of Section B3.1, "Errors and Events of Smart Access Functions".

■ Function

Concatenates two files.

This command appends the file designated by file pathname 2 to the file designated by file pathname 1, and stores the result as a file designated by file pathname 1.

The wildcard character may not be used with this command.

B2.8.3.10 Change File Attribute (ATRW)

Changes the attribute of a specified file or directory.

■ Syntax

Table B2.8.66 Command Specifications

Command Part	Syntax
Command	ATRW
Parameters	Read-only attribute (ASCII) [0 = Read and write 1 = Read only]
	System file attribute (ASCII) [0 = Non-system file 1 = System file]
	Hidden file attribute (ASCII) [0 = Non-hidden file 1 = Hidden file]
	Target file pathname or directory pathname (ASCII) [126 bytes max.]

Command Line:

```
command,parameter(1),parameter(2),parameter(3),parameter(4)
```

■ Example

This sample command changes the file attribute of the file named "\RAMDISK\MYTEXT\data004.txt" to "read-only, non-system file and non-hidden file".

```
ATRW,1,0,0,\RAMDISK\MYTEXT\data004.txt
```

■ Reply

Table B2.8.67 Reply Messages

Reply Message	Code	Description
OK	SE00	Normal exit
Other messages	SE01,...	Error reply message

Note: Reply message is written to the standard output file.

SEE ALSO

For more details on reply messages of card batch file commands, see "■ Errors and Events of Smart Access Functions Logged to Function Output" of Section B3.1, "Errors and Events of Smart Access Functions".

■ Function

Changes the attribute of a specified file or directory.

The wildcard character may not be used with this command.

B2.8.3.11 Unmount (UNMOUNT)

Unmounts the memory card, which is inserted and mounted in the card slot.

■ Syntax

Table B2.8.68 Command Specifications

Command Part	Syntax
Command	UNMOUNT
Parameter	Memory card slot number (ASCII) [always 1]

Command Line:

```
command,parameter
```

■ Example

This sample command unmounts the memory card .

```
UNMOUNT,1
```

■ Reply

Table B2.8.69 Reply Messages

Reply Message	Code	Description
OK	SE00	Normal exit
Other messages	SE01,...	Error reply message

Note: Reply message is written to the standard output file.

SEE ALSO

For more details on reply messages of card batch file commands, see "■ Errors and Events of Smart Access Functions Logged to Function Output" of Section B3.1, "Errors and Events of Smart Access Functions".

■ Function

Unmounts the memory card, which is inserted and mounted in the card slot. A memory card in unmounted state can be safely removed, but does not allow access by programs or via FTP.

If the memory card is successfully unmounted with normal exit, the **SD** LED located on the front panel of the module turns off. Conversely, the **SD** LED is lit if the memory card is mounted.

Executing this command terminates card batch processing so this command must be executed as the last command of a card batch file.

The memory card slot number is fixed to 1.

B3. Common Specifications of Smart Access Functions

This chapter describes common specifications of smart access functions.

B3.1 Errors and Events of Smart Access Functions

This section describes errors and events, as well as log output generated by smart access functions.

■ Error and Event Log Output of Smart Access Functions

All smart access functions output error and event messages to the system log.

Individual functions may also use an additional function-specific output as shown in the table below.

Table B3.1.1 Function Output of Individual Smart Access Functions

Function	Function Output
Rotary switch function	None
Card batch file function	Standard output file
Virtual directory commands	Response file

■ Errors and Events of Smart Access Functions Logged to System Log

The table below lists the errors that may be generated by smart access functions and logged to system log.

When any of these errors is detected, the **EXE** LED blinks, but neither the **ERR** LED nor the **ALM** LED is lit.

Table B3.1.2 List of Smart Access Function Errors Logged to the System Log

Error Message	Error Code	Description
Boot mode error	10-1n	Error related to boot mode. n=MODE switch value
Smart access press function error	31-0n	An error was detected by a press rotary switch function. n=MODE switch value
Smart access press and hold function error	32-0n	An error was detected by a press & hold rotary switch function. n=MODE switch value

Card batch file error	33-0n	An error was detected by the card batch file function.
		This is the representative error code for a card batch file function error.
		A detailed error code is output to the standard output file.
		n=1: startup event trigger
		n=2: error event trigger
		n=3: run program trigger
		n=4: stop program trigger
		n=5: mount memory card event trigger
		n=6: run batch file trigger
		n=7: alarm event trigger

Virtual command error	34-00	An error was detected by virtual directory function. This is the representative error code for a virtual directory function error. A detailed error code is output to the response file.

■ Errors and Events of Smart Access Functions Logged to Function Output

The table below lists the errors and events that may be generated by smart access functions and logged to function output. Usually, a higher level error code is output to the system log while a more detailed error code is logged to the function output.

Table B3.1.3 List of Smart Access Function Errors Logged to Function Output

Reply Message	Error Code	Description
OK	SE00	Processing has completed successfully.
PARAMETER ERROR	SE01	Invalid parameter
DATA CONVERT ERROR	SE02	Input data could not be converted to the specified format.
DEVICE BOUNDARY VALUE EXCEEDED.	SE03	An attempt was made to access a write-prohibit area.
MULTI CPU ERROR	SE04	CPU number is invalid or no response was received from the target CPU.
TIMEOUT ERROR	SE05	Internal timeout has occurred.
FILE SYSTEM ERROR	SE10	Processing could not continue because a file system failure was detected. Reformat the disk in FAT16 format, or replace the memory card.
INVALID FILE	SE11	File data is invalid or could not be interpreted.
NO FILE ERROR	SE12	File or directory was not found. Or, no match was found for the specified wildcard pattern.
FILE OPEN ERROR	SE13	An attempt was made to open a file which is already opened in Write or Append mode.
FILE EXIST ERROR	SE14	Specified destination file already exists. Or, a directory could not be deleted because there are files in it.
FILE PERMISSION ERROR	SE15	A write attempt to a destination was unsuccessful because: - the destination was being accessed; - the destination is a directory; or - the destination is read-only
NOEMPTY ERROR	SE16	No free space is available on the disk. Or, the number of files or directories exceeded the system limit.
NO CARD ERROR	SE17	Processing is not allowed because no memory card is inserted.
CARD UNMOUNT ERROR	SE18	Processing is not allowed because no memory card is mounted.
CARD PROTECT ERROR	SE19	Processing is not allowed because the protection switch is ON.
CARD ERROR	SE20	Processing could not continue because a memory card failure was detected. Replace the memory card.
SECURITY ERROR	SE21	Security password mismatch
RUN MODE ERROR	SE22	Processing is not allowed in Run or Debug mode.
STOP MODE ERROR	SE23	Processing is not allowed in Stop mode.
CHANGE MODE ERROR	SE24	Operating mode change is not allowed. This may be because online edited changes are being written to the CPU module or because of some other reason.
PROGRAM EXECUTION MODE ERROR	SE25	Block activation is not allowed in execute-all-blocks mode.
INVALID BLOCK NAME	SE26	The specified block was not found.
FUNCTION DELETION	SE27	The function is removed in the configuration.
FTPSERVER ERROR	SE30	Processing could not continue due to an FTP server error.

B3.2 System Log File Text Messages

Error messages for the same error code output to log files by smart access functions and displayed in the WideField2 software may differ somewhat. The table below lists smart access function messages alongside WideField2 messages for comparison purposes.

SEE ALSO

For details on errors and troubleshooting, see Section A8.1, "Self Diagnosis".

Table B3.1.4 System Log Messages

Error Message (Logged by Smart Access)	Error Message (Displayed in WideField2)	Error Code	Description
Startup completed	Startup completed	01-0n	Initialization after power on has completed successfully.
Momentary power failure	Momentary power failure	02-00 (13-02)	Momentary power failure
Power Off	Power Off	03-00	Power is turned off.
Startup error	Startup error	10-nn	Error detected during initialization after power on.
SPU error	SPU error	11-nn	Failure of the internal processor of the sequence CPU module
Memory error	Memory error	12-01 12-02 12-03	Memory error
Battery error	Battery error /Memory check error	18-01	Backup battery failure.
Scan timeout	Scan timeout	14-01	Scan timeout interval exceeded.
Invalid instruction found	Invalid instruction found	17-01	An invalid instruction word was detected.
FA link (1-8) error	FA link (1-8) error	15-0n 16-0n 19-0n 1A-0n 1B-0n 1C-0n 1D-0n 1E-0n	Invalid FA link settings
Program Error	Program error	17-02 20-01	Invalid program
Instruction processing error	Instruction error	21-01 21-02 21-03 21-04 21-05 21-06 21-07	Error was detected during execution of instruction.
Subroutine Error	Subroutine error	22-01 22-02	Subroutine mismatch
Interrupt Error	Interrupt error	23-01 23-02	- Non-existent interrupt instruction return location - More than 8 interrupt wait events
I/O Comparison Error	I/O comparison error	24-01 24-02 24-03	- I/O module installation and program mismatch. - A READ/WRITE instruction is used for DIO. - A HRD/HWR instruction is used for DIO.
Macro Instruction Error	Macro instruction error	25-01 25-02	No macro instruction return location
The press function Error	Smart access press function error	31-0n	An error was detected in rotary switch press function.
The press and hold function Error	Smart access press and hold function error	32-0n	An error was detected in rotary switch press and hold function.
Batch file Error	Card batch file error	33-0n	An error was detected by card batch file function.
Virtual command Error	Virtual command error	34-00	An error was detected by virtual directory function.
Inter-CPU Communication Error	Inter-CPU communication error	40-0n	Hardware failure
I/O Error	I/O error	80-00	Cannot read from or write to I/O module.
Sub Unit Transmitter Error	Sub unit transmitter error	83-01	Cannot read from or write to the module attached to the sub-unit.
Sub Unit Transmitter Switching has occurred	Sub unit transmitter switching has occurred	84-01	Line discontinuity detected in the remote I/O system connected in a loop.

Note: For details on the values of 'n' in the error code, see Section A8.1, "Self Diagnosis" or "■ Errors and Events of Smart Access Functions Logged to System Log" of Section B3.1

B3.3 Cleared Alarm Log Messages

Error messages for the same error code output to the cleared alarm log file by smart access functions and displayed in the WideField2 software may differ. The table below lists smart access function messages alongside WideField2 messages for comparison.

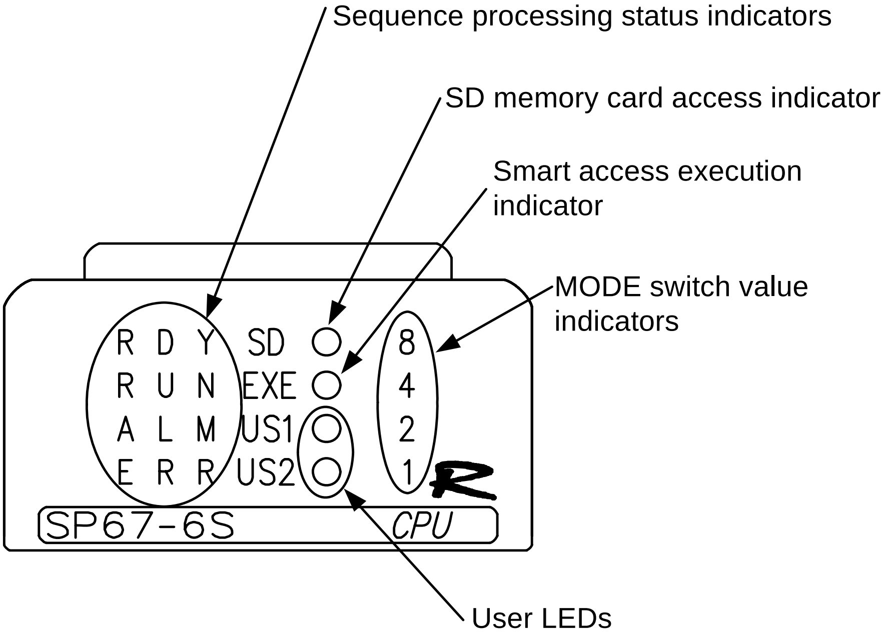
Table B3.1.5 List of Cleared Alarm Log Messages

Error Message (Logged by Smart Access)	Error Message (Displayed in WideField2)	Error Code	Description
Self-diagnostics error	Self-diagnosis error	01-1002	Number of modules exceeded CPU capacity.
		01-1003	Module mapping failure
		01-1004	Module access failure
		01-1005	Other failures during CPU initialization
		01-11XX	SPU error
		01-1201	Program memory failure
		01-1202	Device memory failure
		01-1203	System memory failure
		01-1701	An invalid instruction word was detected.
		01-1702	No END instruction
		01-2001	Label mismatch
		01-2002	I/O points exceeded maximum limit.
Momentary power failure	Momentary power failure	02-0000	Momentary power failure
Inter-CPU communication error	Inter-CPU communication error	03-0000	Hardware failure
Instruction processing error	Instruction processing error	04-2101	Invalid instruction parameter range
		04-2102	Incorrect operation calculation
		04-2103	BIN/BCD conversion error.
		04-2104	FIFO table pointer failure
		04-2105	Device boundary value exceeded.
		04-2106	FOR-NEXT mismatch.
		04-2107	Instruction processing error (IL - ILC mismatch)
		04-2201	Non-existent subroutine return location
		04-2202	Subroutine nesting exceeded 8 levels
		04-2301	Non-existent interrupt instruction return location
		04-2302	More than 8 interrupt wait events
		04-2501	No macro instruction return location
I/O comparison error	I/O comparison error	05-0000	I/O module installation and program mismatch.
I/O error	I/O error	06-0000	Cannot read from or write to I/O module.
Scan timeout	Scan timeout	07-0000	Scan timeout interval exceeded.
FA link (1-8) error	FA link (1-8) error	09-0000 0A-0000 0B-0000 0C-0000 0D-0000 0E-0000 0F-0000 10-0000	Invalid FA link settings
Battery error	Battery error /Memory check error	11-0000	Battery error
Sub unit transmitter error	Sub unit transmitter error	12-0000	Cannot read to or write from the module attached to the sub-unit.
Sub unit transmitter switching has occurred	Sub unit transmitter switching has occurred	13-0000	Line discontinuity detected in the remote I/O system connected in a loop.
Sensor CB scan timeout	Sensor CB scan timeout	14-0000	Sensor control block scan timeout interval exceeded.
***********	***********	XX-XXXX	An undefined alarm was detected.

B4. LED Specifications

This chapter describes the twelve LEDs located on the upper front panel of the module. We recommend that you read this chapter before using smart access functions. Out of the twelve LEDs, five LEDs, namely, the EXE LED, 1 LED, 2 LED, 4 LED and 8 LED are dedicated for use by smart access functions.

■ List of LEDs Located on Upper Front Panel of Module


Sequence processing status indicators
SD memory card access indicator
Smart access execution indicator
MODE switch value indicators
RDY
RUN
ALM
ERR
SD
EXE
US1
US2
8
4
2
1
R
SP67-6S
CPU
User LEDs

FA0103.VSD

Figure B4.1 Overview of LEDs Located on Upper Front Panel of Module

Table B4.1 List of LEDs Located on Upper Front Panel of Module

LED Name	Color	Description	User Control	System Control
RDY LED	Green	Normal: Lit Major failure: Off	No	Yes
RUN LED	Green	Program in Run mode: Lit Program in Debug mode: Lit Program in Stop mode: Off Shutdown in progress: Blinks	No	Yes
ALM LED	Orange	Lit when an alarm is detected.	No	Yes
ERR LED	Red	Lit when an error is detected.	No	Yes
SD LED	Green	Card is unmounted: Off Card is mounted: Lit Card is being accessed: Blinks	No	Yes
EXE LED	Green	Smart access function running: Lit [*1] No smart access function running: Off Smart access function error: Blinks	No	Yes
US1 LED	Green	User-controlled	Yes	No
US2 LED	Green	User-controlled	Yes	No
8 LED 4 LED 2 LED 1 LED	Green	Indicates MODE switch value. The current MODE switch value can be calculated by summing the values of lit LEDs in hexadecimal.	No	Yes

*1: This indicates the basic operation. For exceptions, see the description of individual smart access functions in Part B, "Smart Access Functions".

B4.1 Relationship between Functions and LED Statuses

● Relationship between Program Operating Mode and RUN LED

The table below shows how the **RUN** LED changes according to the program operating mode.

Table B4.1.1 Relationship between Program Operating Mode and RUN LED

Operating Mode	State of RUN LED
Run mode	Lit
Debug mode	Lit
Stop mode	Off
Shutdown in progress (Transition to Stop mode)	Blinks

● Relationship between Error Events and ERR LED, ALM LED

The table below shows how the **ERR** LED and **ALM** LED changes according to the severity of an error event.

Table B4.1.2 Relationship between Error Events and ERR LED, ALM LED

Error Severity	State of ERR LED	State of ALM LED
Major failure	Lit	Off
Moderate failure	Lit[*1]	Off[*1]
Minor failure	Off	Lit
Smart access error	Off	Off

*1: Some moderate failures can be customized as minor failures in the configuration, and will then affect the LEDs in the same way as minor failures.

● Relationship between SD Memory Card Access and SD LED

The table below shows how the **SD** LED changes with the state of the SD memory card.

Table B4.1.3 Relationship between SD Memory Card Access and SD LED

State of SD Memory Card	State of SD LED
Mounted	Lit
Unmounted	Off
Being accessed	Blinks

● Relationship between MODE Switch Value and 8,4,2,1 LEDs

These indicators together indicate the current MODE switch value. The current MODE switch value can be calculated by summing the values of lit LEDs in hexadecimal.

Table B4.1.4 Relationship between MODE Switch Value and 8,4,2,1 LEDs

LED Name	Description
8 LED	Binary, '8' place
4 LED	Binary, '4' place
2 LED	Binary, '2' place
1 LED	Binary, '1' place

Table B4.1.5 Mode Switch Value and Lit Status of 8,4,2,1 LEDs

	MODE Switch Value															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
State of 8 LED	—	—	—	—	—	—	—	—	**8**	**8**	**8**	**8**	**8**	**8**	**8**	**8**
State of 4 LED	—	—	—	—	**4**	**4**	**4**	**4**	—	—	—	—	**4**	**4**	**4**	**4**
State of 2 LED	—	—	**2**	**2**	—	—	**2**	**2**	—	—	**2**	**2**	—	—	**2**	**2**
State of 1 LED	—	**1**	—	**1**	—	**1**	—	**1**	—	**1**	—	**1**	—	**1**	—	**1**

Note: - Each LED indicates value **8**, **4**, **2** or **1** when lit.
- '-' denotes Off

● Relationship between Smart Access Functions and EXE LED

The table below shows how the **EXE** LED changes according to the execution status of smart access functions. Beware that the behavior may differ for some requested functions.

Table B4.1.6 Relationship between Smart Access Functions and EXE LED

State of Rotary Switch Function	State of EXE LED
No operation	Off
Running	Lit
Normal exit	Lit->Off
Error exit	Lit->Blinking[*1]

*1: Goes off if the SET switch is pressed and released.

SEE ALSO

For details, see the description of individual smart access functions in Part B, "Smart Access Functions".

B4.2 Accessing LEDs from a Program

The states of the LEDs located on the front panel of the module are reflected on special relays. The state of some LEDs can also be controlled by writing to special relays.

The table below shows the mapping between LEDs and special relays. A special relay indicating a blinking status has higher precedence than a relay indicating a lit/off status. In other words, if a relay indicating blinking status is ON, you can ignore the state of the corresponding relay indicating lit/off status and safely conclude that the LED is blinking.

Table B4.2.1 Mapping between Upper Front Panel LEDs and Special Relays

Name of LED	State of LED	Relay No.	Name of Special Relay	Read/Write
RDY LED	Lit/Off	M0113	RDY LED special relay	Read-only
	Blinking	None	None	—
RUN LED	Lit/Off	M0115	RUN LED special relay	Read-only
	Blinking	None	None	—
ALM LED	Lit/Off	M0117	ALM LED (1) special relay	Read-only
	Blinking	M0118	ALM LED (2) special relay	Read-only
ERR LED	Lit/Off	M0119	ERR LED (1) special relay	Read-only
	Blinking	M0120	ERR LED (2) special relay	Read-only
SD LED	Lit/Off	M0121	SD LED (1) special relay	Read-only
	Blinking	M0122	SD LED (2) special relay	Read-only
EXE LED	Lit/Off	M0123	EXE LED (1) special relay	Read-only
	Blinking	M0124	EXE LED (2) special relay	Read-only
US1 LED	Lit/Off	M0125	US1 LED (1) special relay	Read/write
	Blinking	M0126	US1 LED (2) special relay	Read/write
US2 LED	Lit/Off	M0127	US2 LED (1) special relay	Read/write
	Blinking	M0128	US2 LED (2) special relay	Read/write
8 LED, 4 LED 2 LED, 1 LED	Lit/Off	None	None	—
	Blinking	None	None	—

Note: The current MODE switch value (summation of 1 LED to 8 LED) is stored in special register Z0117.

FA-M3

Sequence CPU – Functions (for F3SP66-4S, F3SP67-6S)

PART B Smart Access Functions

IM 34M6P14-01E 1st Edition

INDEX

FA-M3

Sequence CPU – Functions (for F3SP66-4S, F3SP67-6S)

PART C Storage Functions

IM 34M6P14-01E 1st Edition

C1. Memory Card

This chapter describes the memory card for use with the sequence CPU module.

CAUTION

Before using an SD memory card with the module for the first time, ensure that it is formatted in FAT16. If the format of an SD memory card is unknown, format it using the sequence CPU module before use.

SEE ALSO

For more details on how to format an SD memory card, see Subsection B1.5.3, "Format CARD1", or Subsection C3.5.5.3, "Format Disk (FORMAT)".

C1.1 Overview of Memory Card

This section describes the uses, types and functions of the memory card.

C1.1.1 Uses of Memory Card

When mounted in a CPU module, a memory card serves as a large capacity non-volatile memory disk for file storage and retrieval. The memory card can be used not only as a conventional ROM pack but also to provide unique memory card functions. This subsection describes some typical uses of the memory card.

■ Storage and Retrieval of Recipe Files and Log Files

A memory card can be used to store recipe or equipment parameter files in CSV or any other format for retrieval by a program.

Device log data can be saved as files in CSV or any other format on a memory card.

■ Storage of Manuals and Design Specifications

A memory card has a large capacity, and can be used to store user manuals for equipment and FA-M3 modules, as well as equipment design specifications. By connecting a PC to a CPU module installed with a memory card, you can read these manuals and specifications in the field. This is especially convenient if you are maintaining a range of modules in the field as you would not need to carry a lot of documents with you.

■ Loading and Saving of Projects

If project files are pre-stored on a memory card in card load format, you can load the projects into the CPU module using smart access functions without using a PC.

You can also save any project on the built-in ROM to a memory card in card load format.

Project data can be converted between card load format and WideField2 format.

■ Card Boot

The card boot function allows the CPU module to automatically load a project of card load format from the memory card into the built-in ROM and then enter Run mode or Stop mode.

With the RUN mode option, the module executes the program after startup. With the Stop mode option, the module enters standby after startup. Card boot with the Stop mode option is especially useful for automatic loading of software application at a user site following delivery of the module from the factory.

■ File Exchange with a PC

The memory card adopts the FAT16 format, which is the standard file system for PCs, so the card can be used for file exchange between the FA-M3 module and a PC without the need for any special application software. For example, by removing a memory card from the module and inserting it into a card slot on your PC, you can utilize the data of the module on your PC. Likewise, by removing a memory card from your PC and inserting it into a card slot in the module, you can read data that have been created on the PC using CPU programs.

■ Other Uses

The module has other functions, which make use of the memory card. The following is a brief description of these functions. For details, see the description of individual function in other parts of this manual.

● Rotary switch functions

You can perform simple maintenance using the rotary switch (MODE switch) located on the front panel of the CPU module without using a PC. For instance, you can load a WideField2 project from a memory card into the module.

SEE ALSO

For details on rotary switch functions, see Chapter B1, "Rotary Switch Functions".

● Card batch file functions

The card batch file function executes a series of batch file commands contained in a batch file (auto-execute file) on a memory card. For instance, the module can be made to automatically execute the commands in a batch file to copy specified files or perform other tasks without the use of a PC when a memory card is inserted into the module or when an error is detected.

SEE ALSO

For details on card batch file functions, see Chapter B2, "Card Batch File Function".

● FTP functions

You can transfer files between a memory card and a PC via a network.

SEE ALSO

For details on FTP functions, see Chapter 3, "FTP Function" of "Sequence CPU – Network Functions User's Manual (for F3SP66-4S, F3SP67-6S)" (IM34M6P14-02E).

C1.1.2 Types of Memory Card

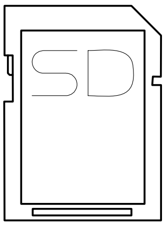
This subsection describes the types of memory card available for use with the module.

SEE SLSO

For recommended memory cards, see "■ Recommended Memory Cards" in Subsection C1.2.1, "Memory Card Specifications".

■ SD Memory Card

An SD memory card is a thin, small flash-ROM memory card about the size of a postal stamp. The capacity of commercial SD memory cards today ranges from several megabytes to 1 gigabyte. As a non-volatile memory device, an SD memory card can retain data without any power supply from the module.



FC0101.VSD

Figure C1.1.1 SD Memory Card

CAUTION

An SD memory card contains blocks of flash ROM memory and each memory block may be overwritten up to 100,000 times. Do not write to the memory card frequently as if it were an ordinary RAM memory.

C1.2 Specifications of Memory Card and Card Slot

This section describes the specifications of the memory card and card slot.

C1.2.1 Memory Card Specifications

■ Memory Card Specifications

Table C1.2.1 SD Memory Card Specifications

Items	Specifications
Card Type	SD Memory Card
Capacity	Up to 1G bytes (before formatting)
Memory Type	Flash ROM
Disk Type	Removable
Volatility	Non-volatile
Maximum Number of Write Operations	About 100,000 times

■ Recommended Memory Cards

A list of recommended memory cards is given below. Yokogawa does not guarantee proper operation with non-recommended memory cards.

The list of recommended memory cards may grow as new memory cards are introduced in the market or may change depending on Yokogawa test results. For the latest information on recommended memory cards, visit Yokogawa's FA-M3 website (http://www.fa-m3.com).

Table C1.2.2 Recommended Memory Cards

Card Slot	Media	Manufacturer	Models
CARD1	SD memory card	Matsushita Electric Industrial Co., Ltd. (Panasonic)	Visit http://www.fa-m3.com.

CAUTION

Yokogawa does not guarantee proper operation with non-recommended memory cards.

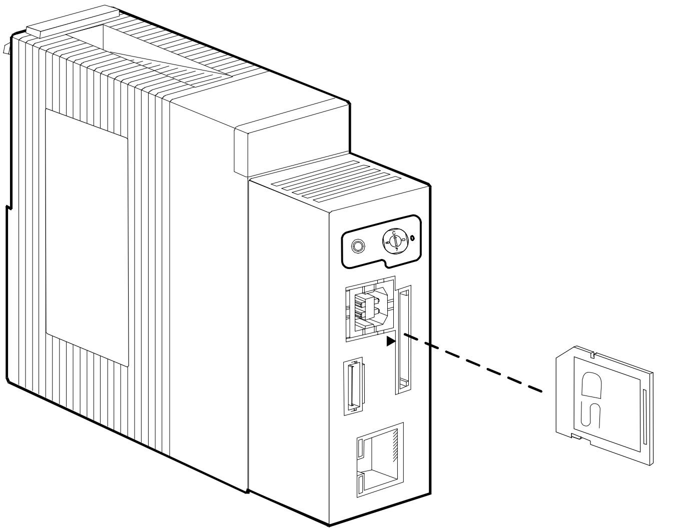
C1.2.2 Card Slot Specifications

This subsection describes the specifications of the SD memory card slot (CARD1) located on the front panel of the module.

■ Specifications of SD Memory Card Slot (CARD1)

● Location of the SD memory card slot (CARD1)

The SD memory card slot (CARD1) is located in the front panel of the module. Insert an SD memory card into the card slot as shown in the figure below. Do not insert a card by force with a wrong orientation as this may damage the SD memory card, the module or both devices.



FC0102.VSD

Figure C1.2.1 Location of the SD Memory Card Slot (CARD1)

● Compatible memory card types

The SD memory card slot (CARD1) only accepts SD memory cards.

Use recommended SD memory cards only.

SEE ALSO

For details on recommended memory cards, see "■ Recommended Memory Cards" in Subsection C1.2.1, "Memory Card Specifications".

● Assigned directory name

The SD memory card slot (CARD1) is assigned the directory name of "\CARD1" which can be used by a ladder instruction or FTP function to access the memory card.

C1.2.3 File and Directory Restrictions

The file and directory restrictions for the memory card are the same as those for the FAT16 file system. The file system of the memory card supports the FAT16 format.

SEE ALSO

For details on the file system, see Subsection C3.2.1, "File System Specifications".

C1.2.4 System Directory Structure

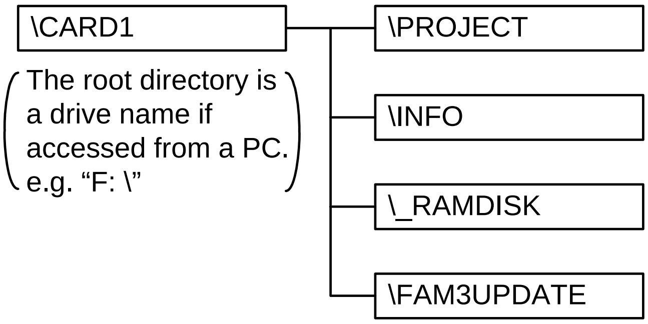
System directories are automatically created on a memory card when the module functions run. They are used for file input or output by a function. Always use a system directory according to its intended purpose. Otherwise, it may result in an unexpected operation.

■ Generation of System Directories

A function that requires any system directory will normally create it automatically if it is not present. You may also create system directories manually.

■ System Directories of Memory Card CARD1

The figure below shows the tree structure of the system directories of the memory card CARD1.


\CARD1
The root directory is a drive name if accessed from a PC. e.g. "F: \"
\PROJECT
\INFO
_RAMDISK
\FAM3UPDATE

FC0103.VSD

Figure C1.2.2 System Directories of the Memory Card CARD1

The table below lists the system directories of the memory card CARD1 together with the functions using them.

Table C1.2.3 System Directories of the Memory Card CARD1

Directory Paths	Related Functions	Description
\CARD1\PROJECT	Rotary switch functions	Projects and CPU properties are saved to or loaded from the directory.
\CARD1\INFO	Rotary switch functions	Logs, CPU information, system information, or technical support information is saved under this directory.
\CARD1_RAMDISK	Rotary switch functions	A rotary switch function can be used to copy the contents of the RAM disk to this directory, or to copy the contents under this directory to the RAM disk.
\CARD1\FAM3UPDATE	Rotary switch functions	This is a directory reserved by Yokogawa for maintenance purposes.

C1.2.5 Special Relays and Special Registers

■ Special Relays

Table C1.2.4 Special Relays Related to Memory Card

Category	System Operation Status Relays		
No.	Name	Function	Description
M0250	CARD1 mounted	Indicates whether memory card CARD1 is mounted.	This relay is ON if the memory card CARD1 is mounted; It is OFF if no memory card is mounted.

■ Special Registers

There are no special registers related to the memory card.

C1.3 Memory Card Setup

This section describes how to configure the memory card before use.

C1.3.1 Basic Setup

All new memory cards must be formatted before use using the module. If the module is not available, you can also use a PC running Windows to format a new memory card but you must specify the FAT16 format.

Beware that a new memory card having a capacity of 128M bytes or less is most likely pre-formatted in FAT12. A FAT12 memory card will be re-formatted in FAT12 format by a PC running Windows if you specify FAT format during formatting. You can physically insert a FAT12 memory card into the module card slot but it will not be mounted.

Beware also that memory cards that come with digital cameras may be pre-formatted in a proprietary format.

CAUTION

When using an SD memory card with the module for the first time, ensure that it is formatted in FAT16 format. If the format of an SD memory card is unknown, always format it using the sequence CPU module before use.

SEE ALSO

For details on how to format an SD memory card, see Subsection B1.5.3, "Format CARD1", or Subsection C3.5.5.3, "Format Disk (FORMAT)".

C1.3.2 Optional Setup

The memory card may be configured as required before use.

Table C1.3.1 Optional Setup for Memory Card

Name of Setup	Type of Setup	SEE ALSO
FTP client setup	CPU properties	A9.5.5, "FTP Client Setup"
FTP server setup	CPU properties	A9.5.6, "FTP Server Setup"

● FTP client setup

FTP client setup is required if the memory card is to be accessed by an FTP client. Otherwise, this setup is not required.

● FTP server setup

FTP server setup is required if the memory card is to be accessed by an FTP server. Otherwise, this setup is not required.

C1.4 Using a Memory Card

This section describes how to use a memory card.

C1.4.1 Accessing a Memory Card

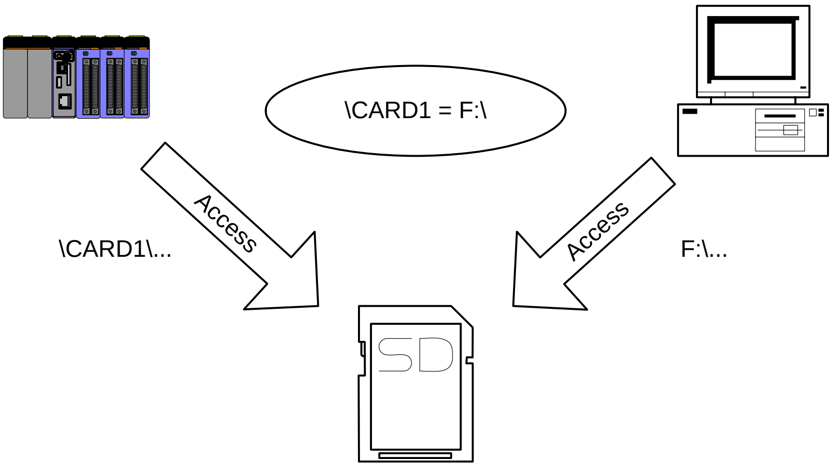
This subsection outlines various means of accessing the memory card.

■ Assigned Directory Name

The SD memory card slot (CARD1) of the module is assigned the directory name "\CARD1".

This directory name is used to access the memory card from the module. For example, a ladder program may access a file on the memory card by specifying "\CARD1" followed by its file pathname on the memory card.

To access a file on the same memory card from a PC, insert the memory card in the PC and specify its file path starting with the root directory for the drive such as “F:\” (the root directory of the PC maps to "\CARD1" of the module).



Note: The drive identifier “F” here is just an example.
Its actual value depends on the configuration of your PC.

FC0104.VSD

Figure C1.4.1 Root Directory of the Memory Card

■ Means of Access

● Ladder program

A ladder program can be written to access a memory card using file system instructions. This is the most flexible means of memory card access. Using a ladder program, you can freely create and manipulate files on a memory card or even transfer files between the memory card and an FTP server or some other external equipment via a network using FTP client instructions.

SEE ALSO

- For details on memory card access by program, see Section C3.5, "File System Instructions".
- For details on FTP client, see Chapter 3, "FTP Function" of “Sequence CPU – Network Functions User's Manual (for F3SP66-4S, F3SP67-6S)” (IM34M6P14-02E).

● FTP server

Any external equipment acting as an FTP client can exchange files with a memory card over a network.

SEE ALSO

For details on memory card access by FTP server, see Chapter 3, "FTP Function" of “Sequence CPU – Network Functions User's Manual (for F3SP66-4S, F3SP67-6S)” (IM34M6P14-02E).

● Card batch file functions

A batch file (auto execution file) of commands stored on a memory card can be executed automatically to manipulate memory card files or perform maintenance operations upon the mounting of a memory card or some other events.

SEE ALSO

For details on memory card access by card batch file functions, see Chapter B2, "Card Batch File Function".

● Rotary switch functions

You can initiate memory card related maintenance operations using the rotary switch.

SEE ALSO

For details on memory card access by rotary switch functions, see Chapter B1, "Rotary Switch Functions".

● PC

Memory cards used with the module are formatted in standard FAT16 format. You can access the files on the memory card by simply removing it from the module and inserting it in a USB card reader/writer or a PCMCIA slot on your PC without the need for any special software.

SEE ALSO

For details on accessing the memory card from a PC using a USB card reader/writer, see the respective user manuals of the PC and USB card reader/writer.

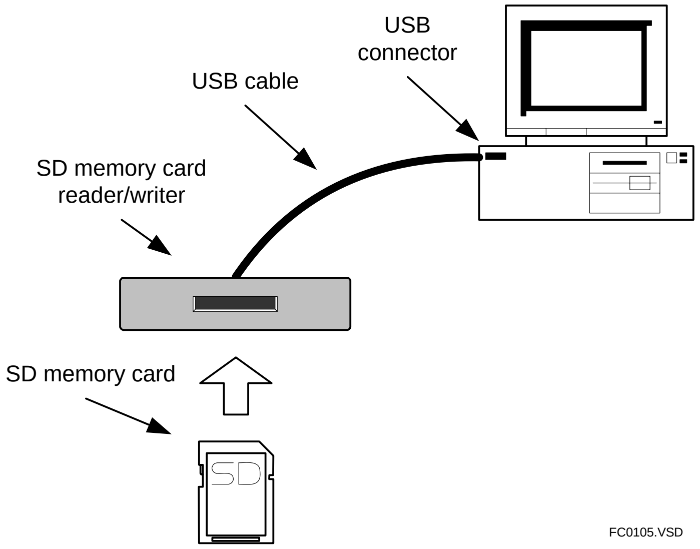


Figure C1.4.2   Accessing Memory Card from PC using USB Card Reader/Writer

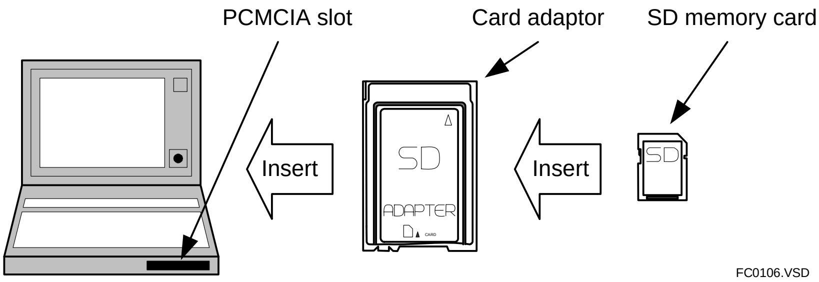


Figure C1.4.3   Accessing Memory Card from PC using PCMCIA Slot and Card Adaptor

C1.4.2 Precautions When Using SD Memory Card Slot (CARD1)

This subsection describes some precautions when using the SD memory card slot (CARD1).

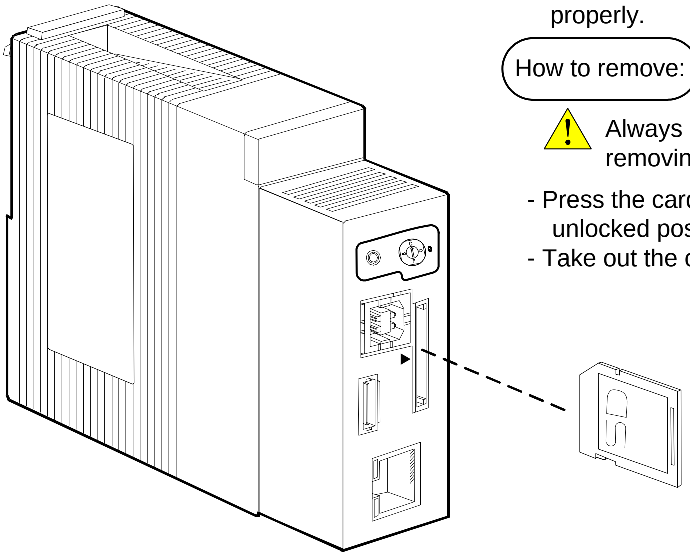
■ Inserting and Removing SD Memory Card

● Mounting an SD memory card

The SD memory card must be “mounted” by the module to be ready for use. When an SD memory card is properly inserted in the SD memory card slot (CARD1), it is automatically recognized and mounted by the module and becomes ready for ladder program access or FTP access. If an SD memory card is improperly inserted in the slot due to incorrect orientation, invalid card format or some other reason, it will not be mounted successfully. You can check whether an SD memory card is mounted by checking the CARD1 Mounted relay (M0250) (on if mounted) or the **SD** LED (lit if mounted) located on the front panel of the module.

● Inserting and removing an SD memory card

1. Insert an SD memory card into the SD memory card slot (CARD1) located on the front panel of the module with the correct orientation (see figure below) and push it in until it clicks into place.
2. Confirm that the **SD** LED located on the front face of the module is lit or confirm that the Card1 Mounted relay (M0250) is ON.



How to insert:

- Insert the card with its face toward the ▾ mark on the slot.
- It clicks into a locked position when inserted properly.

How to remove:

⚠ Always unmount the card before removing it.

- Press the card once so that it clicks into an unlocked position.
- Take out the card.

FA0101.VSD

Figure C1.4.4 Inserting and Removing an SD Memory Card

TIP

If the **SD** LED is not lit or the CARD1 Mounted relay (M0250) is OFF with an SD memory card inserted, remove and re-insert the card. If it still fails to be mounted, try inserting it in a different device which you know is working normally.

● Precautions when removing an SD memory card

Always unmount an SD memory card before removing it from the module. Removing a mounted SD memory card without first unmounting it may damage its files or file system.

An SD memory card in the slot is unmounted if the **SD** LED is not lit and the CARD1 Mounted special relay M0250 is turned off. If it is unmounted, you can safely remove it even if the module is switched on because the CPU module cannot access it.

● Unmounting and removing an SD memory card

1. Use one of the methods given in the table below to unmount an SD memory card which is mounted in the memory card slot (CARD1).
2. Confirm that the **SD** LED located on the front panel of the module is not lit or the Card1 Mounted relay (M0250) is OFF.
3. Press the SD memory card to release it and then remove it from the module card slot.

TIP

If an unmount request is initiated while an SD memory card is being accessed, the card is unmounted only after access is completed. This may take a few minutes depending on the nature of the access.
Accessing a memory card that is already unmounted generates an access error.

Table C1.4.1 Methods for Unmounting Memory Card

Unmounting Memory Card Using:	SEE ALSO
Rotary switch function	B1.4.2, "Unmount CARD1"
Card batch file function	B2.8.3.11,"Unmount (UNMOUNT)"
Ladder instruction	C3.5.5.2, "Unmount Memory Card (UNMOUNT)"

CAUTION

Always unmount an SD memory card before removing it from the module. Removing a mounted SD memory card without first unmounting it may damage its files or file system.

■ SD Memory Card Slot (CARD1) Statuses

The two tables below list the possible statuses of the SD memory card slot (CARD1) and methods for checking its statuses.

● SD memory card slot statuses (CARD1)

Table C1.4.2 SD Memory Card Slot (CARD1) Statuses

Status	Description
CARD1 Mount Status	Indicates whether memory card CARD1 is mounted.
CARD1 Free Space	Indicates the available space on memory card CARD1 in bytes.
CARD1 Capacity	Indicates the capacity of memory card CARD1 in bytes.

● Methods for checking SD memory card slot (CARD1) status

Table C1.4.3 Methods for Checking SD Memory Card Slot (CARD1) Status

Checking Status Using:	Accessible Statuses	SEE ALSO
Rotary switch function	CARD1 Mount Status CARD1 Free Space CARD1 Capacity	B1.4.5, "Module Info"
Card batch file function		B2.8.2.4, "CPU Info (CPUINFO)"
Virtual directory command		3.7.7.4, "CPU Info (CPUINFO)" of “Sequence CPU – Network Functions User's Manual (for F3SP66-4S, F3SP67-6S)” (IM34M6P14-02E)"
Ladder instruction		C3.5.5.4, "Disk Info (DISKINFO)"
Special relay or special register	CARD1 Mount Status	C1.2.5, "Special Relays and Special Registers"

C2. RAM Disk

This chapter describes the RAM disk of the module.

C2.1 Overview of RAM Disk

This section describes the uses and features of the RAM disk.

C2.1.1 Uses of RAM Disk

The RAM disk of the module features both fast access of a RAM memory and easy access like a disk. It is faster than a memory card for reading and writing files and can be thought as a large capacity, volatile memory disk. This subsection describes some typical uses of the RAM disk.

■ Work Area for High-speed File Processing

Although the memory card is handy for loading recipe files and saving log files, it uses flash ROM and thus has slower access than RAM memory. In particular, if data is to be written intermittently, say, during a tracing task, the module may not be able collect the required number of samples due to the bottleneck caused by slow access to the memory card.

You can use the RAM disk as work area for fast file processing to overcome such bottlenecks and improve application performance. For instance, you can write data to the RAM disk by taking advantage of its high access speed, and then transfer data sequentially from the RAM disk to a memory card later when time permits. As the RAM disk has a large capacity of 4M bytes, using it as work area allows you to reduce the load of file registers (B) and other internal devices.

■ Other Uses

The module provides other functions, which make use of the RAM disk. These functions are briefly described below. For details, see the description of individual functions in other parts of this manual.

● Rotary switch functions

You can copy all data on the RAM disk to a memory card using the rotary switch (MODE switch) located on the front panel of the CPU module without using a PC. You can also copy data on a memory card to the RAM disk.

SEE ALSO

For details, see Subsection B1.4.7, "Copy RAM Disk" and Subsection B1.4.8, "Copy Memory Card".

● FTP functions

You can transfer files between the RAM disk and a PC over a network. The time required is less than transferring files to the memory card using FTP.

SEE ALSO

For details on FTP functions, see Chapter 3, "FTP Function" of "Sequence CPU – Network Functions User's Manual (for F3SP66-4S, F3SP67-6S)" (IM34M6P14-02E).

C2.1.2 Features of RAM Disk

The RAM disk is a volatile 4M-byte area of the SDRAM of the module, which is automatically allocated and formatted at module startup. This initialization is transparent to a user and the RAM disk is ready for access by ladder programs, FTP servers or FTP clients immediately after module startup.

As the RAM disk is part of the SDRAM, it allows faster access than the memory card, which uses flash ROM. For this reason, it is designed to be used as temporary work space during log file generation or recipe file analysis.

The RAM disk loses its data when power is shut off. To retain the data on the RAM disk, you must copy the data to a memory card before switching off the module.

C2.2 RAM Disk Specifications

This section describes the specifications of the RAM disk.

C2.2.1 RAM Disk Specifications

■ RAM Disk Specifications

Table C2.2.1 RAM Disk Specifications

Items	Specifications
Capacity	4M bytes (before formatting)
Memory Type	SDRAM
Disk Type	Fixed disk
Volatility	Volatile
Maximum Number of Write Operations	Unlimited
Sector Size	512 bytes
Cluster Size	4096 bytes (8 sectors)

■ Assigned Directory Name

The RAM disk is assigned the directory name of "\RAMDISK". A ladder program instruction or an FTP function must specify this path when accessing data on the RAM disk.

C2.2.2 File and Directory Restrictions

The file and directory restrictions on the RAM disk are determined by the FAT16 file system. The maximum allowable number of files or directories is about 1000, the actual limit depending on actual conditions.

SEE ALSO

For details on the file system, see Subsection C3.2.1, "File System Specifications".

C2.2.3 System Directory Structure

The RAM disk has no directory, which is generated by the system.

C2.2.4 Special Relays and Special Registers

There are no special relays and special registers related to the RAM disk.

C2.3 RAM Disk Setup

This section describes how to configure the RAM disk before use.

C2.3.1 Basic Setup

The RAM disk requires no basic setup.

C2.3.2 Optional Setup

The RAM disk may be configured as required before use.

Table C2.3.1 Optional Setup for RAM Disk

Name of Setup	Type of Setup	SEE ALSO
FTP client setup	CPU properties	A9.5.5, "FTP Client Setup"
FTP server setup	CPU properties	A9.5.6, "FTP Server Setup"

● FTP client setup

FTP client setup is required if the RAM disk is to be accessed by an FTP client. Otherwise, this setup is not required.

● FTP server setup

FTP server setup is required if the RAM disk is to be accessed by an FTP server. Otherwise, this setup is not required.

C2.4 Using RAM Disk

This section describes how to use the RAM disk.

C2.4.1 Accessing the RAM Disk

This subsection describes how you can access the RAM disk. You do not have to format a RAM disk before use because the module automatically formats it at a startup or restart.

■ Assigned Directory Name

The RAM disk is assigned the directory name "\RAMDISK".

To access a file on the RAM disk, specify the directory name "\RAMDISK" followed by the file path on the RAM disk.

■ Means of Access

● Ladder program

A ladder program can be written to access the RAM disk using file system instructions. This is the most flexible means of memory card access. Using a ladder program, you can freely create and manipulate files on the RAM disk or even transfer files between the RAM disk and an FTP server or some other external equipment via a network using FTP client instructions.

SEE ALSO

- For details on RAM disk access by program, see Section C3.5, "File System Instructions".
- For details on FTP client, see Chapter 3, "FTP Function" of the “Sequence CPU – Network Functions User's Manual (for F3SP66-4S, F3SP67-6S)” (IM34M6P14-02E).

● FTP server

Any external equipment acting as an FTP client can exchange files with the RAM disk over a network.

SEE ALSO

For details on RAM disk access by FTP server, see Chapter 3, "FTP Function" of the “Sequence CPU – Network Functions User's Manual (for F3SP66-4S, F3SP67-6S)” (IM34M6P14-02E).

● Card batch file functions

A batch file (auto execution file) of commands stored on a memory card can be executed automatically to manipulate files on the RAM disk upon the mounting of a memory card or some other events.

SEE ALSO

For details on RAM disk access by card batch file functions, see Chapter B2, "Card Batch File Function".

● Rotary switch functions

You can copy all data on the RAM disk to a memory card using the rotary switch.

SEE ALSO

For details on RAM disk access by rotary switch functions, see Chapter B1, "Rotary Switch Functions".

C2.4.2 Simulating Non-volatility for the RAM Disk

The RAM disk is actually RAM memory and thus features fast access but loses all its data when its power supply is lost. To overcome this limitation, write a program to copy all data on the RAM disk to a memory card before power off and restore the data from the memory card to the RAM disk after the next power on.

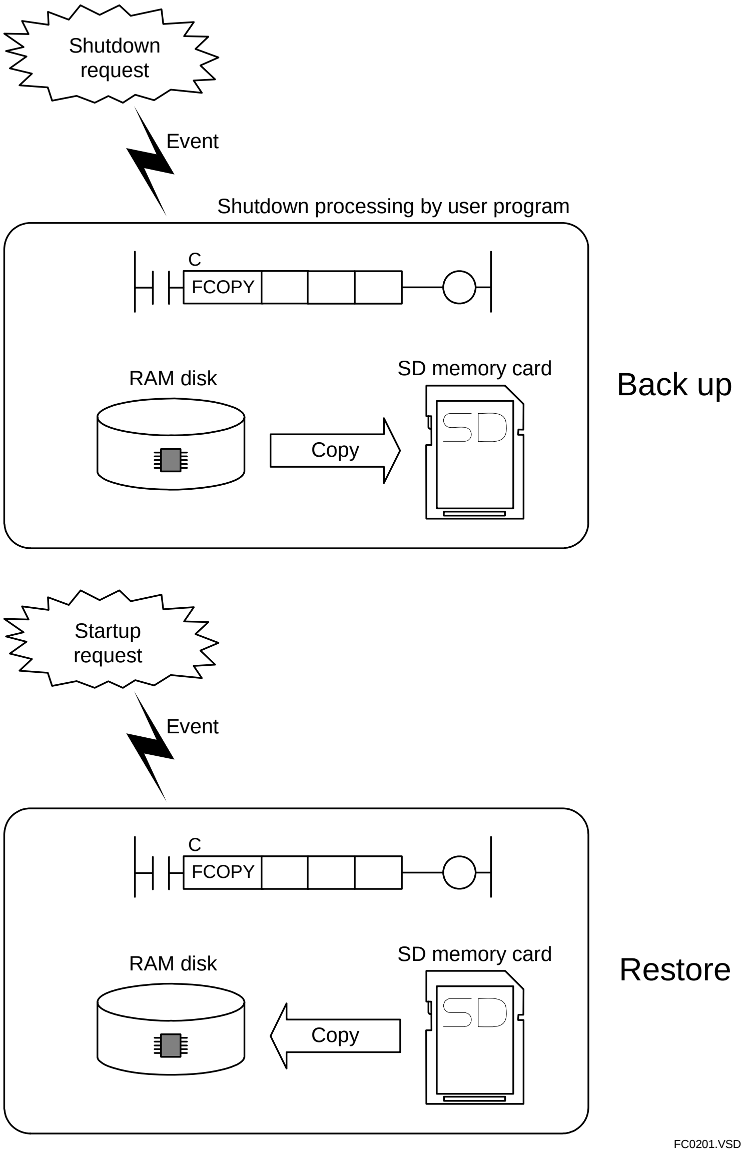


Figure C2.4.1 Simulating Non-volatility for the RAM Disk

C2.4.3 RAM Disk Statuses

This subsection discusses the various statuses of the RAM disk and how to check the RAM disk statuses.

■ RAM Disk Statuses

Table C2.4.1 RAM Disk Statuses

Status	Description
RAM Disk Free Space	Indicates the free space on the RAM disk in bytes.
RAM Disk Capacity	Indicates the capacity of the RAM disk in bytes.

■ Reading RAM Disk Statuses

Table C2.4.2 Reading RAM Disk Statuses

Checking Ram Disk Status Using:	Accessible Statuses	SEE ALSO
Rotary switch function	RAM Disk Free Space RAM Disk Capacity	B1.4.5, "Module Info"
Card batch file function		B2.8.2.4, "CPU Info (CPUINFO)"
Virtual directory command		3.7.7.4, "CPU Info (CPUINFO)" of “Sequence CPU – Network Functions User's Manual (for F3SP66-4S, F3SP67-6S)” (IM34M6P14-02E）
Ladder instruction		C3.5.5.4, "Disk Info (DISKINFO)"

C3. File System

This chapter consists of the following sections:

C3.1 Overview of File System

This section outlines the file system of the module.

C3.1.1 What Is a File System?

A file system is a mechanism for managing files and directories of a memory card or RAM disk on a PC. A file system allows a user to manipulate data on a memory card or RAM disk on a file-and-directory basis.

When manipulating files using a ladder program or FTP, it is important to understand the features, specifications and restrictions of the file system services provided by the file system.

C3.1.2 Uses of File System

Of the various file systems available today, the module adopts FAT16. As the FAT16 file system is also supported by Windows, you can directly manipulate files saved by the module on the memory card from a PC by inserting the card into a card reader/writer connected to the PC. Conversely, you can remove a memory card from your PC and insert it into the memory card slot of the module so that module functions can read files on the memory card, which were saved from the PC.

SEE ALSO

- For details on memory card functions, see Chapter C1, "Memory Card."
- For details on RAM disk functions, see Chapter C2, "RAM Disk."

C3.2 File System Specifications

This section describes the specifications of the file system of the module.

C3.2.1 File System Specifications

This subsection describes the specifications of the file system of the module and usage restrictions.

■ File System Specifications

Table C3.2.1 File System Specifications

Item	Specifications
File System Format	FAT16
Long File Name	Supported (VFAT)
Wildcard Characters	Supported (*, ?)
Maximum Number of Open Files	16 files
Directory Structure	Supported

■ File and Directory Naming Rules

A file or directory must be named according to the rules given below.

Table C3.2.2 File and Directory Naming Rules

Items	Rules	
Valid Characters	ASCII characters	
Characters Prohibited in a File or Directory name	\ (backslash), | (vertical bar), ? (question mark), < (less-than sign), " (double quote)	/ (slash), * (asterisk), : (colon), > (greater-than sign),
Maximum Length of File or Directory Name	255 characters	
Maximum Length of File Pathname	255 characters	
Directory Delimiter	\ (backslash)	

■ File and Directory Restrictions

The table below lists file and directory restrictions other than naming rules.

System reserved directory names are reserved for use by the module, and should not be used by a user.

Table C3.2.3 File and Directory Restrictions

Item	Restrictions
Maximum Number of Directories and Files under the Mounted Directory of a Drive	256 max.*3
Directory Hierarchy	8 layers*1*2
System Reserved Directory Names or File Names (not to be used by a user)	CARD1
	RAMDISK
	CPUx (x = 1 to 4)
	ESLOTxx (xx = 01 to 16)
	FSLOTxx (xx = 01 to 16)
	FNODExxx (xxx = 001 to 254)
	INUNIT
	INFLNET

*1: Some instructions report no errors for directory hierarchy deeper than 8 layers but normal operation is not guaranteed.

*2: A deep directory hierarchy slows down file access.

*3: Actual limit depends on the lengths of file and directory names.

CAUTION

- Having too many files under one directory may significantly reduce access speed. A directory should normally contain no more than 50 files but it is advisable to confirm this limit against the performance of your application.
- Long file names and directory names slow down file access. Shorter file names and directory names are recommended for better performance.
- Do not use any system reserved directory name or filename for a user directory or file. Otherwise, some module functions may fail to work properly.

C3.2.2 Allowed File Types

All file types that satisfy the file system specifications are allowed. The module imposes no special restrictions on file types.

C3.2.3 FA-M3 File Types

Some file types are proprietary to FA-M3. These file types are identified by their file extensions.

Table C3.2.4 FA-M3 Proprietary File Types

File Types	Extension	Description
Project file of card load format	ypjc	These project files are loaded or saved using smart access functions.
CPU property file	yprp	These files are CPU module configuration files.

● Project file of card load format (file extension: ypjc):

A WideField2 project file of card load format is loaded or saved using smart access functions (such as rotary switch functions). Beware that such a file has a different format from standard WideField2 project files downloaded or uploaded using the WideField2 software. Project files of card load format can be converted to standard WideField2 project files and vice versa.

SEE ALSO

- For details on smart access functions, see Part B, "Smart Access Functions" of this manual.
- For details on how to create and convert project files of card load format, see "FA-M3 Programming Tool WideField2 User's Manual" (IM34M6Q15-01E).

● CPU property file (file extension: yprp):

A CPU property file is used for configuring a CPU module. It is a collection of setup items that can be modified by a user during module operation. CPU property files handled by the WideField2 software and smart access functions have the same format.

SEE ALSO

For details on CPU property files, see Chapter A9, "Setup Description."

C3.2.4 Special Relays and Special Registers

■ Special Relays

Table C3.2.5 Special Relays Related to File System

Category	File System Status Relays		
No.	Name	Function	Description
M1026	No Unused File ID	No unused file ID is available.	Turns on when all file IDs are in use. This is a read-only relay. Do not write to it.
M1025	File/Disk Operation Group Busy	File operation instruction group or disk operation instruction group is running.	Turns on during execution of any file operation instruction or disk operation instruction (FCOPY, DISKCOPY, etc.). Execution of any other file operation instruction or disk operation instruction is not allowed while this relay is ON. This relay is not affected by file access instructions. This is a read-only relay. Do not write to it.
M1041 to M1056	File ID Open	File ID is open.	Each file ID is associated with one special relay. The relay for a file ID turns on while the file ID is open. When the relay for a file ID is OFF, no instruction using the file ID can be executed. This is a read-only relay. Do not write to it.
M1057 to M1072	File ID Busy	File ID is busy.	Each file ID is associated with one special relay. The relay for a file ID turns on during execution of any file system instruction using the file ID. When the relay for a file ID is ON, no other file system instruction using the same file ID can be executed. This is a read-only relay. Do not write to it.

■ Special Registers

There are no special registers related to the file system.

C3.3 File System Setup

This section describes how to configure the file system before use.

SEE ALSO

- For details on memory card setup, see Section C1.3, "Memory Card Setup."
- For details on RAM disk setup, see Section C2.3, "RAM Disk Setup."

C3.3.1 Basic Setup

The file system requires no basic setup.

C3.3.2 Optional Setup

The file system has no optional setup.

C3.4 Using the File System

This section describes how to use the file system and its related instructions. You should read this section before using file access instructions.

SEE ALSO

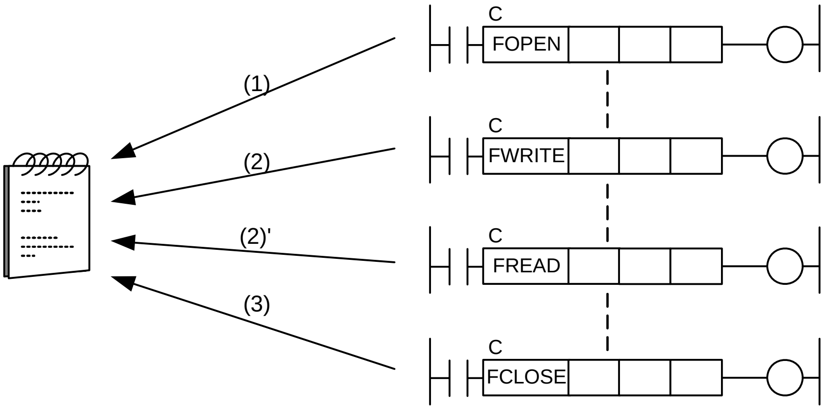
For details on the use of memory card related to the file system, see Chapter C1, "Memory card."
For details on the use of RAM disk related to the file system, see Chapter C2, "RAM Disk."

C3.4.1 File ID

You must gain the access right to a file before using it. To get access right to a file, you "open" it. If the operation is successful, the system returns the access right in the form of its "file ID." In other words, you must open a file and received its file ID before you can use the file.

When you have finished using a file, you must release (close) the file ID so that other processes may the access right to the file.

The figure below shows, in the form of a ladder diagram, the basic flow of file operation: opening a file → accessing (reading from and writing to) the file → closing the file.



(1) Get a file ID (gain access right to a file)
(2)(2)' Access the file
(3) Release the file ID (release access right to the file)

FC0303.VSD

Figure C3.4.1 File ID

■ Modes of File IDs

File IDs are associated with one of three modes as shown in the table below. They are used according to the intended file processing.

Table C3.4.1 File ID Modes and Access Rights

File ID Mode	Access Right
Read-only mode	Represents the right only to read a file. A file can have a number of different open file IDs. A Read-only mode file ID may be opened for any file including files for which a write or append file ID is open.
Write mode	Represents the right to both read and write a file. Opening an existing file in Write mode deletes the file and creates a new file with the same file name. When a Write mode file ID is open for a file, no other Write or Append mode file ID can be opened for that file due to exclusive control.
Append mode	Represents the right to append to an existing file. Opening a non-existent file in Append mode creates a new file with the specified file name. When an Append mode file ID is open for a file, no other Append or Write mode file ID can be opened for that file due to exclusive control.

■ Getting and Releasing a File ID

There are two methods by which file IDs are obtained or released: manual and automatic. The former is initiated by a user while that latter is initiated by the module.

The table below shows the functions for getting file IDs manually and automatically.

Table C3.4.2 Getting File IDs Manually and Automatically

Method for Getting File ID	Related Function
Manual	File access instructions
Automatic	File operation instructions
	Disk operation instructions
	FTP server functions
	FTP client functions
	Rotary switch functions
	Card batch file functions
	Virtual directory commands

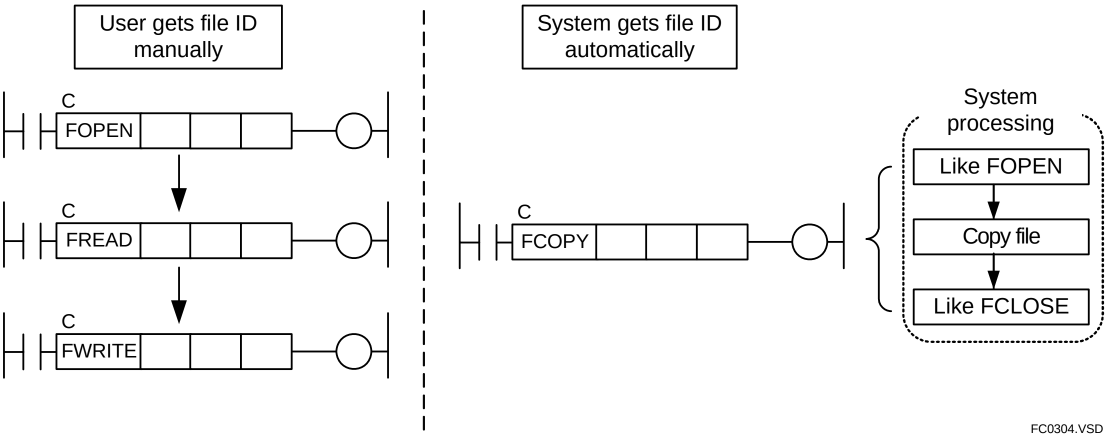


Figure C3.4.2 Getting a File ID

■ Getting and Releasing a File ID Manually

You must open a file and receive its file ID before you can use the file with file access instructions. Execute the Open File (FOPEN) instruction to open a file.

Execute the Close File (FCLOSE) instruction to release an open file ID.

SEE ALSO

- For details on the Open File (FOPEN) instruction, see C3.5.3.1, "Open File (FOPEN)."
- For details on the Close File (FCLOSE) instruction, see C3.5.3.2, "Close File (FCLOSE)."

C3.4.2 Exclusive Control Feature of the File System

The exclusive control feature of the file system prevents loss of data integrity of a file arising from concurrent modification or deletion of the file by different processes. The file system implements exclusive control by controlling the allocation of file IDs.

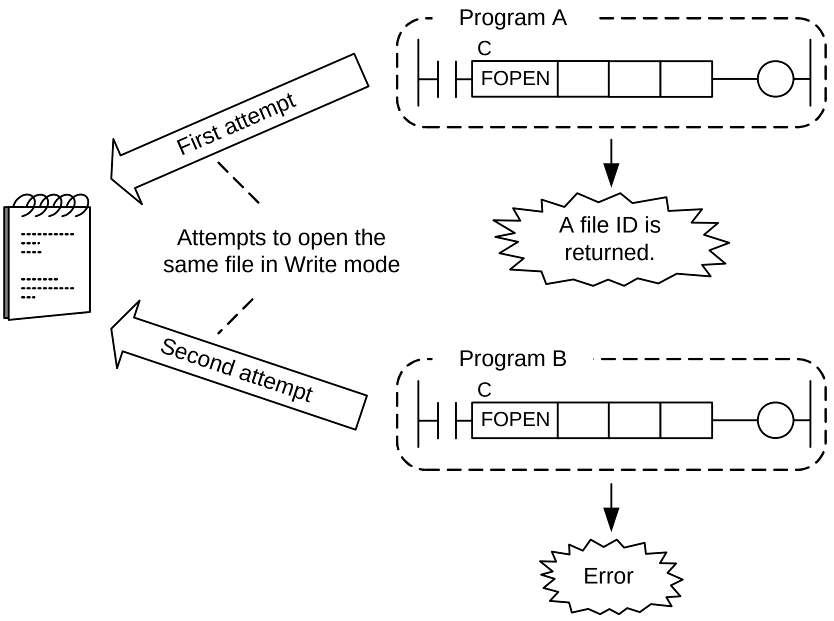
■ Exclusive Control of a File

Two or more file IDs may or may not be open for the same file at the same time depending on the combination of the modes of the file IDs involved.

The table below shows what happens when an attempt is made to open a file in Read-only, Write or Append mode when it is already opened in a specific mode. The table applies equally to manual and automatic means of getting file IDs by all functions (such as file access instructions or FTP functions).

Table C3.4.3 Getting File IDs Manually or Automatically

File is Already Opened in:	Attempt to Open a File in:		
	Read-only mode	Write mode	Append mode
Read-only mode	OK	Error	OK
Write mode	OK	Error	Error
Append mode	OK	Error	Error



FC0301.VSD

Figure C3.4.3 Exclusive Write Control of Files

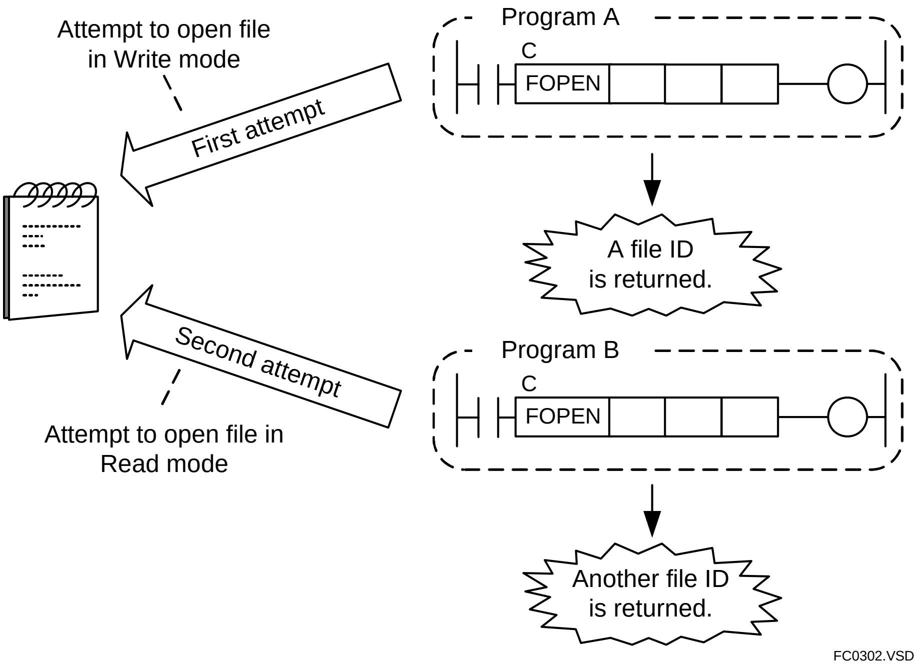


Figure C3.4.4 Exclusive Read Control of Files

C3.4.3 Fail Safe Feature of the File System

This subsection describes the fail safe feature and how it protects data against file access abort events.

■ Fail Safe Feature

The file system of the module has a built-in fails safe feature for protection against file access errors. The fail safe function backs up disk control information before a file write operation to guard against power interruption or improper removal of a memory card. If a file access error occurs, the file system tries to recover any lost data using the backup information.

■ File Access Abort Events

● If a memory card is removed during file access

In the worst case, the file system is destroyed and the memory card is no longer usable.

Thanks to the fail safe feature of the file system, even if you have removed a memory card while a file is being accessed, you may be able to restore the memory card to its state right after the most recent successful write operation by re-mounting the memory card on the module. Beware that the fail safe function discards any backup information if a different memory card is mounted.

● If the module is switched off or reset during file access

In the worst case, the file system is destroyed and the memory card is no longer usable.

Thanks to the fail safe feature of the file system, even if you have switched off or reset the module while a file is being accessed, you may be able to restore the memory card to its state right after the most recent successful write operation by re-mounting the memory card on the module. Beware that the fail safe function discards any backup information if a different memory card is mounted.

C3.5 File System Instructions

This section describes file system instructions.

File system instructions are continuous type application instructions, whose instruction processing spans multiple scan cycles. In addition, some file system instructions use text parameters, which should be set up before the instructions are executed.

This section first describes how to use continuous type application instructions and text parameters. It then describes the three categories of file system instructions, namely, file access instructions, file operation instructions and disk operation instructions.

C3.5.1 Using File System Instructions

■ Continuous Type Application Instructions

Execution of many file system instructions cannot be completed within one scan period. To avoid affecting control processing, a processing request is issued at the time of instruction execution but the time-consuming actual processing is carried out in the background. Such instructions are known as "continuous type application instructions".

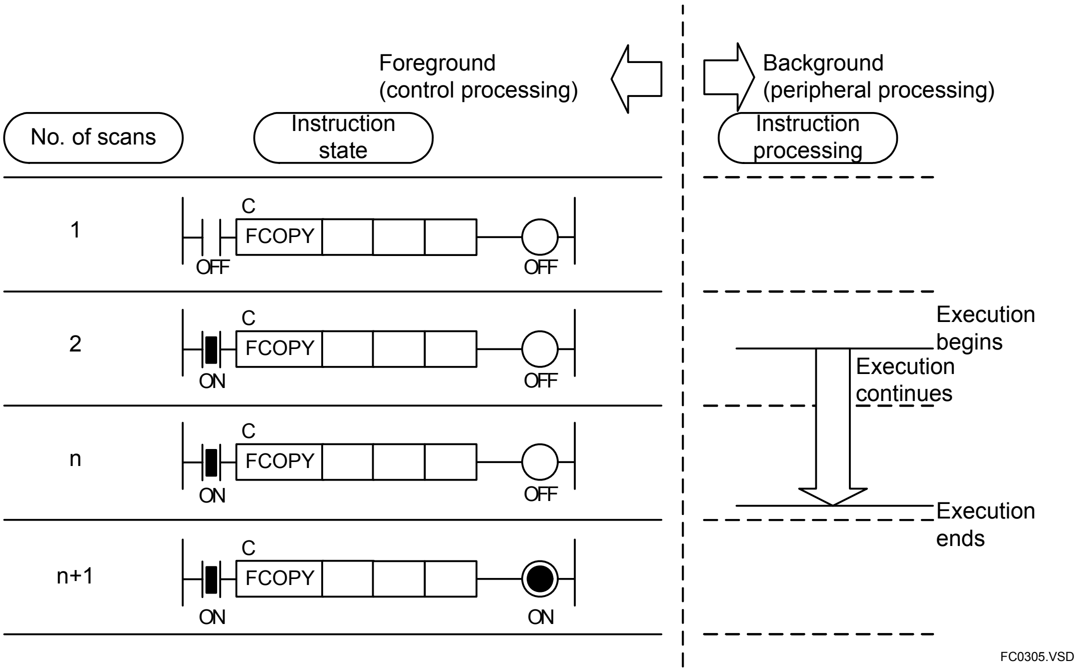


Figure C3.5.1 Concept of Continuous Type Application Instruction

● Operation of Continuous Type Application Instructions

This subsection describes the operation of a continuous type application instruction. In the description, the term "input condition" refers to the ON/OFF state of the circuit connection line immediately preceding the continuous type application instruction.

- To execute the instruction:
  Change its input condition from OFF to ON.
- To continue instruction execution:
  Hold its input condition in ON state.
- When instruction execution completes:
  The result signal (on the circuit line connected to the output (right) end of the instruction) is held to ON for one scan period. A user program can check the completion of a continuous type application instruction by monitoring an OUT instruction or some other output-type instruction placed on the output end of the instruction.
- To re-execute the instruction after it has completed execution:
  Turn off and again turn on its input condition. The condition must be held in OFF state for at least 1 scan period.
- To cancel (abort) instruction execution:
  Turn off its input condition during instruction execution. The result signal is held to ON for one scan period. However, the background instruction processing does not end immediately. For more details, see "● Canceling Execution of Continuous Type Application Instructions" later in this subsection.

Table C3.5.1 Operation of Continuous Type Application Instructions

Instruction State of Preceding Scan	Input Condition of Preceding Scan	Input Condition of Current Scan	Transition of Instruction State in Current Scan	Result Signal of Current Scan
Stopped	OFF	ON	Execute	OFF
		OFF	Stopped	OFF
Execute	ON	ON	Continue Execution	OFF
			Execution Completed[*1]	ON for 1 scan
		OFF	Cancelled	ON for 1 scan
Execution Completed[*1]	ON	ON	Execution Completed	OFF
		OFF	Stopped	OFF
Cancelled	OFF	ON	Start execution	OFF
		OFF	Stopped	OFF

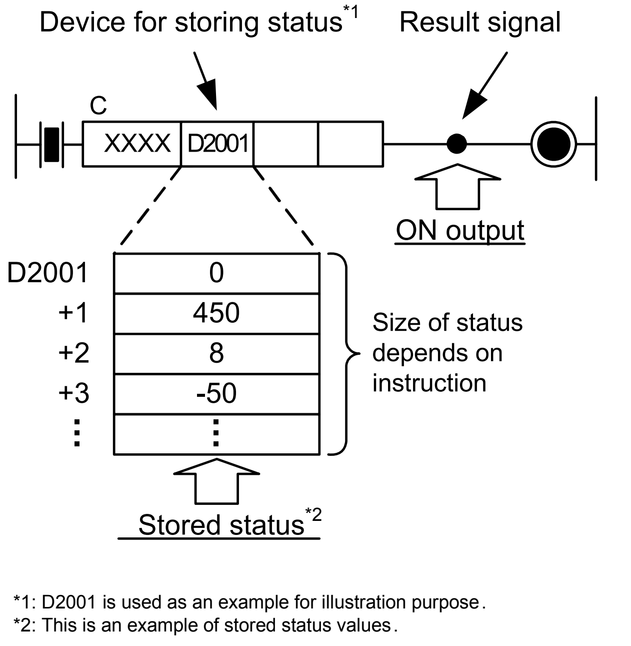
*1: The transition to 'Execution Completed' state is independent of the input condition, and is triggered by completion of background instruction processing.

● Operation Result of Continuous Type Application Instructions

Continuous type application instructions output two types of operation result at the end of instruction execution. A user program determines the completion of instruction execution using the result signal, and checks whether execution is successful using the status.

Table C3.5.2 Operation Result of Continuous Type Application Instructions

Operation Result	Description
Result signal	At the end of instruction execution, the result signal is held to ON for one scan. The result signal is OFF at other times. A user program determines whether instruction execution has completed by checking the ON/OFF state of the result signal.
Status	Regardless of whether instruction execution is successful, a status value is stored in a user-specified device. Some devices may store other return values in addition to the status so the status has a multi-word table structure. If an error status is returned, a user program should perform application error processing such as retry processing.


Device for storing status*1
Result signal
C
XXXX
D2001
ON output
D2001
+1
+2
+3
0
450
8
-50
Size of status depends on instruction
Stored status*2
*1: D2001 is used as an example for illustration purpose.
*2: This is an example of stored status values.

FC0306.VSD

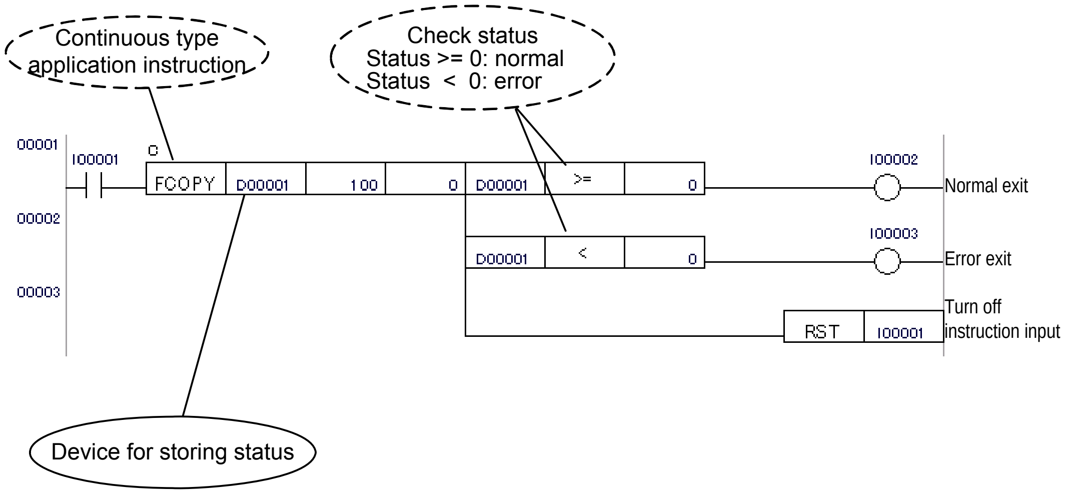
Figure C3.5.2 Operation Result Output of Continuous Type Application Instruction

● Error Processing of Continuous Type Application Instructions

If instruction execution ends normally, a zero or positive integer is stored in status. If execution ends in error, a negative integer is stored in status.

A user program should read the execution result status and perform whatever error processing (e.g. retry) as appropriate if an error status is returned.

Even if an error status is returned, the module does not store an error code in a special register, write to the system log (error log), turn on the ALM LED or ERR LED, or switch the program operating mode.



FC0307.VSD

Figure C3.5.3 Error Processing of Continuous Type Application Instructions

● Error Status of Continuous Type Application Instructions

The table below shows the error status codes of continuous type application instructions.

Table C3.5.3 Continuous Type Application Instruction Status (timeout-related (-1xxx), non-error-related (-2xxx), exclusive-control-related (-3xxx))

Category	Continuous Type Application Instruction Status		
	Value	Name	Description
Timeout	-1000	Instruction Timeout	Processing failed to end within the timeout interval specified by an instruction parameter.
	-1001	Internal Communication Timeout	No response was received within the internal communication timeout interval. The following timeout interval can be defined by a user as a CPU property. - FTP Client Network Timeout
Non-error	-2000	End of File Detected	End of file was detected during processing.
	-2001	No Match Found	No match was found.
	-2002	Disconnected by Remote Node	Connection was terminated by the remote node. Check the status of the remote node. This status is also returned if high network load causes data loss.
	-2003	Specified Size/Times Processed	Processing has been completed for the specified data size or iterations. - The size of data received by a TCP/IP Receive Instruction (TCPRCV instruction) reaches the specified receive area size.
	-2004	Block Size Error	Data size is smaller than the specified block size.
Exclusive control	-3001	Repeated Use of Function	A function or resource that disallows repeated use was used repeatedly. - Repeated execution of FTP client instruction - Repeated execution of file operation instruction or disk operation instruction - Repeated use of file ID or socket ID
	-3003	Write-prohibit Destination	A write attempt to a destination was unsuccessful because: - the destination was being accessed - the destination is a directory - the destination is read-only
	-3004	Repeated Write Mode	An attempt was made to open a file, which is already open in Write (Append) mode.
	-3005	Internal Resource Depleted	Internal resource is temporarily depleted. To resolve the problem, retry later. If the problem persists, consider reducing processing load. - FA-M3 internal resource - Protocol stack internal resource

Table C3.5.4 Continuous Type Application Instruction Status (network-related (-5xxx))

Category	Continuous Type Application Instruction Status		
	Value	Name	Description
Network	-5000	Connection Error	Error was detected during connection.
	-5001	Unknown Destination	The destination was not found.
	-5002	Buffer Overflow	Send/receive buffer used by socket instructions has overflowed.
	-5030	FTP User Authentication Failure	Access was denied by FTP server's user authentication process.
	-5031	FTP Password Authentication Failure	Access was denied by FTP server's password authentication process.
	-5032	FTP Command Sequence Error	FTP client processing could not continue because a reply received from the FTP server was out of sequence. This error may be due to repeated cancel operations or bad line quality.
	-5421	FTP Negative Reply 421	FTP server returns a negative reply. The last three digits of this error code (positive value) represent the reply code received from the FTP server.*1
	-5425	FTP Negative Reply 425	
	-5426	FTP Negative Reply 426	
	-5450	FTP Negative Reply 450	
	-5451	FTP Negative Reply 451	
	-5452	FTP Negative Reply 452	
	-5500	FTP Negative Reply 500	
	-5501	FTP Negative Reply 501	
	-5502	FTP Negative Reply 502	
	-5503	FTP Negative Reply 503	
	-5504	FTP Negative Reply 504	
	-5530	FTP Negative Reply 530	
	-5532	FTP Negative Reply 532	
	-5550	FTP Negative Reply 550	
	-5551	FTP Negative Reply 551	
	-5552	FTP Negative Reply 552	
	-5553	FTP Negative Reply 553	

*1: For details on the meaning of each reply code, see the official FTP specification (RFC959). Note that the causes and meanings of reply codes may vary with individual FTP server implementations.

Table C3.5.5 Continuous Type Application Instruction Status (file system related (-6xxx))

Category	Continuous Type Application Instruction Status		
	Value	Name	Description
File system	-6000	Duplicate Filename	Specified destination filename already exists
	-6002	Insufficient Space	There is insufficient space on the storage media. Or, number of files or directories exceeded maximum limit.
	-6004	Memory Card Not Installed	Processing is not allowed because no memory card is installed.
	-6005	Memory Card Not Mounted	Processing is not allowed because no memory card is mounted.
	-6006	Protection Switch is ON	Processing is not allowed because the card protection switch is enabled.
	-6007	File System Failure	Processing could not continue because a file system failure was detected or the file system is not in FAT16 format. Reformat the disk in FAT16 format, or replace the memory card. This status may be returned occasionally when there is insufficient space on the storage media.
	-6008	Memory Card Failure	Processing could not continue because a memory card failure was detected. Replace the memory card.
	-6009	Unknown Write Error	An error of unknown cause was detected during write processing. Reformat disk to FAT16 format, or replace the memory card.
	-6010	FLS Processing Sequence Error	Executions of FLSFIRST, FLS and FLSFIN instructions were out of sequence.
	-6011	File Interpretation Error	The NULL byte was detected during interpretation of a text file.

Table C3.5.6 Continuous Type Application Instruction Status (General Instruction (-9xxx))

Category	Continuous Type Application Instruction Status		
	Value	Name	Description
General Instructions	-9000	Cancel Request Issued	A cancel request was issued. Check the resource relay to determine when cancellation is completed.
	-9010	Resource Not Opened	The specified file ID or socket ID is not open. Execute an Open instruction for the ID.
	-9011	Resource Depleted	- No more unused socket ID or file ID is available. Check the resource relay. - An attempt was made to run multiple FTP clients. Concurrent execution of FTP clients is not allowed.
	-9012	Resource Released by External Factor	Processing could not continue because a user has caused the resource relay to be turned off so writing to the resource relay is prohibited. This error may occur if the SD memory card is unmounted when a file is open.
	-9013	Function Not Started	- A function required for processing is not running. - FTP client is not running. Execute an FTPOPEN instruction.
	-9014	Invalid Device Access	An attempt was made to access an invalid device number. Check index modification, indirect designation, data size and status size.
	-9015	Data Processing Error	The requested processing could not continue because of invalid data.
	-9020	Security Error	The specified password or keyword is incorrect.
	-9021	CPU Property ROM Write Error	An attempt to write CPU property data to the internal ROM failed.
	-9999	Internal Error	Internal error was detected.

Table C3.5.7 Continuous Type Application Instruction Status (parameter error related (-1xxxx))

Category	Continuous Type Application Instruction Status		
	Value	Name	Description
Parameter Error	-10xxx	Parameter Error	The specified parameter is invalid. The last 3 digits of the error code indicate the position of the invalid instruction parameter and its offset from the beginning of the table in words if the parameter is a table. Status: -10 △□□ △: Parameter number (1 to 3) □□: Offset in table (00 to 99)
	-12xxx	Invalid Pathname	The specified pathname is invalid. This error is generated if path interpretation failed because the specified file pathname violated a syntax rule. The third digit of the error code indicates the location of the invalid parameter. Status: -12 △□□ △: 1 to 3 : Text parameter number 4 : CPU property 9 : Unknown type □□: System reserved (currently 00)
	-13xxx	Pathname Object Not Found	The object designated by the pathname is not found. This error is generated if the specified pathname contains an invalid file or directory. For instance, "\RAMDISK\MYDIR" is specified but there is no directory named "MYDIR" on the RAM disk. This error may also be generated if a wildcard is specified but no match is found. The third digit of the error code indicates the location of the invalid parameter. Status: -13 △□□ △: 1 to 3 : Text parameter number 4 : CPU property 9 : Unknown type □□: System reserved (currently 00)
	-15xxx	Invalid String Length	The string length parameter is invalid. This error is generated if the string length exceeds the maximum limit, or if NULL is specified for a parameter that does not allow a NULL value. The third digit of the error code indicates the location of the invalid parameter. Status: -15 △□□ △: 1 to 3 : Text parameter number 4 : CPU property 9 : Unknown type □□: System reserved (currently 00)

● Canceling Execution of Continuous Type Application Instructions

Execution of a continuous type application instruction can be cancelled by turning off its input condition during execution. When a falling edge is detected in the input condition, the result signal is immediately held to ON to notify termination of execution, and a Cancel Request Issued status code (-9000) is stored in the instruction status.

However, note that despite notification of instruction termination, background instruction processing is not yet terminated. Instead, a cancellation request is issued to background processing, and a few seconds may be required to complete the termination.

If the same continuous type application instruction is executed before background processing cancellation is completed, resource competition occurs and an exclusive control related error will be generated. To avoid this, you should include the resource relay in the input condition of a continuous type application instruction.

TIP

Just as with instruction cancellation, in the event of an instruction timeout (error code -1000), background instruction processing continues to run for a short while. Therefore, it is also necessary in this case to incorporate exclusive control in the program using resource relays.

SEE ALSO

For details on resource relays, see "■ Resource Relays" later in this subsection.

CAUTION

When the input of a continuous type application instruction is turned off, the instruction immediately returns a Cancel Request Issued status and terminates execution. However, actual background processing such as background communications is not terminated immediately. To check for termination of actual processing, check that the associated resource relay is turned off.

● Precautions When Executing Continuous Type Application Instructions

By nature, continuous type application instructions require multiple scans to complete processing, and thus should not be executed only once but should be executed repeatedly until execution completes.

The table below shows the precautions when executing continuous type application instructions from different program types.

Table C3.5.8 Precautions when Executing Continuous Type Application Instructions

Program Type	Precaution
Ladder block (execute-all-blocks mode)	None
Ladder block (execute-specified-blocks mode)	Executing an Inactivate Block (INACT) instruction during execution of a continuous type application instruction forces cancellation of instruction processing.
Sensor control block	Continue execution of a sensor control block until the execution of a continuous type application instruction ends. If you stop a sensor control block before instruction execution ends, instruction processing cannot be completed.
I/O interrupt routine	Use of continuous type application instructions in I/O interrupt routines is not allowed.
Subroutine	Repeat a subroutine call until the execution of a continuous type application instruction ends. If you stop subroutine call before instruction execution ends, instruction processing cannot be completed.
Macro and input macro	Repeat a macro call until the execution of a continuous type application instruction ends. If you stop macro call before execution of continuous type application instruction ends, instruction processing cannot be completed. Calling a macro containing an executing continuous type application instruction from a different location in the program is not allowed.

CAUTION

- A continuous type application instruction will not execute correctly if it is executed in only one scan.
- Do not execute the same continuous type application instruction more than once within the same scan using macros. Repeat execution using FOR-NEXT instruction or JMP instruction is also disallowed.

● Restrictions for Inserting Continuous Type Application Instructions

There are some restrictions for inserting continuous type application instructions in a ladder diagram. Placing a continuous type application instruction in an invalid location generates a program syntax error in WideField2.

The figure below illustrates some locations where continuous type application instructions cannot be inserted.

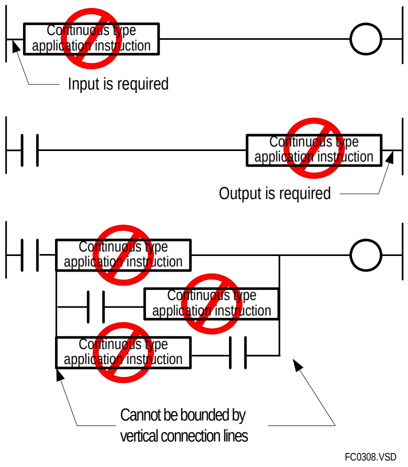

Continuous type application instruction
Input is required
Continuous type application instruction
Output is required
Continuous type application instruction
Continuous type application instruction
Continuous type application instruction
Cannot be bounded by vertical connection lines
FC0308.VSD

Figure C3.5.4 Restrictions for Inserting Continuous Type Application Instructions

● Online Edit of Continuous Type Application Instructions

Do not online edit a circuit containing an executing continuous type application instruction. If an executing continuous type application instruction is edited online, instruction processing will be forcedly terminated and re-executed using the modified parameters (including text parameters). In this case, the result signal of the continuous type application instruction will not be held to ON for 1 scan to indicate end of instruction processing.

Even if parameter values are not modified during online edit, a Repeated Use of Function error (status code -3001) may still be generated during re-execution depending on the status of the resource.

CAUTION

Before performing online edit of a circuit containing continuous type application instructions, check to ensure that no continuous type application instruction is running.

■ Resource Relays

Resource relays are special relays for preventing competition between continuous type application instructions. A resource relay indicates the status of a resource, which is subject to exclusive control. Resources include file IDs socket IDs, functions and instructions.

By inserting a resource relay in the input condition of a continuous type application instruction, you can prevent errors due to resource competition. In particular, resource relays are required for checking for completion of cancellation processing or instruction timeout processing in user applications where cancellation request for a continuous type application instruction, or timeout (-1000) may occur.

● Resource Relays (related to file system instructions)

Table C3.5.9 Resource Relays (related to file system instructions)

Category	File System Continuous Type Application Instruction Resource Relays		
No.	Name	Function	Description
M1026	No Unused File ID	No unused file ID is available.	Turns on when all file IDs are in use. This is a read-only relay. Do not write to it.
M1025	File/Disk Operation Group Busy	File operation instruction group or disk operation instruction group is running.	Turns on during execution of any file operation instruction or disk operation instruction. Execution of any other file operation instruction or disk operation instruction is not allowed while this relay is ON. This relay is not affected by file access instructions. This is a read-only relay. Do not write to it.
M1041 to M1056	File ID Open	File ID is open.	Each file ID is associated with one special relay. The relay for a file ID turns on while the file ID is open. When the relay for a file ID is OFF, no instruction using the file ID can be executed. This is a read-only relay. Do not write to it.
M1057 to M1072	File ID Busy	File ID is busy.	Each file ID is associated with one special relay. The relay for a file ID turns on during execution of any file system instruction using the file ID. When the relay for a file ID is ON, no other file system instruction using the same file ID can be executed. This is a read-only relay. Do not write to it.

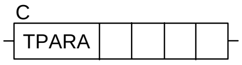
■ Text Parameter

Some file system instructions use text parameters as instruction parameters. A text parameter value can be stored using the Text Parameter (TPARA) instruction. The Text Parameter (TPARA) instruction must be executed before an instruction that requires text parameters.

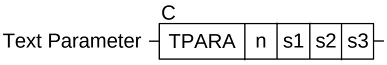
● Text Parameter (TPARA)

This instruction is used to specify a text parameter required by some continuous type application instructions.

Table C3.5.10 Text Parameter

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro-cessing Unit	Carry
					Yes	No			
Application Instruction	–	Text Parameter	TPARA		✓	–	5	8 bit	–

Parameter



n : Text parameter number (W) (1-4)

s1 : Device storing character string 1 (W)
(Up to 255 characters, terminated by a NULL character)

s2 : Device storing character string 2 (W)
(Up to 255 characters, terminated by a NULL character)

s3 : Device storing character string 3 (W)
(Up to 255 characters, terminated by a NULL character)

Available Devices

Table C3.5.11 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	W	Z	R	V	Con-stant	Index Modification	Indirect Designation, Pointer P
n									✓	✓	✓		✓	✓	✓	Yes	Yes
s1									✓	✓	✓		✓	✓	✓	Yes	Yes
s2									✓	✓	✓		✓	✓	✓	Yes	Yes
s3									✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.15, "Restrictions on Devices Used as Instruction Parameters" of "Sequence CPU – Instructions User’s Manual" (IM34M6P12-03E).

Function

This instruction is used to specify text parameters required by some continuous type application instructions. You should specify the text parameter number according to the text parameter number of the instruction requiring the text parameter.

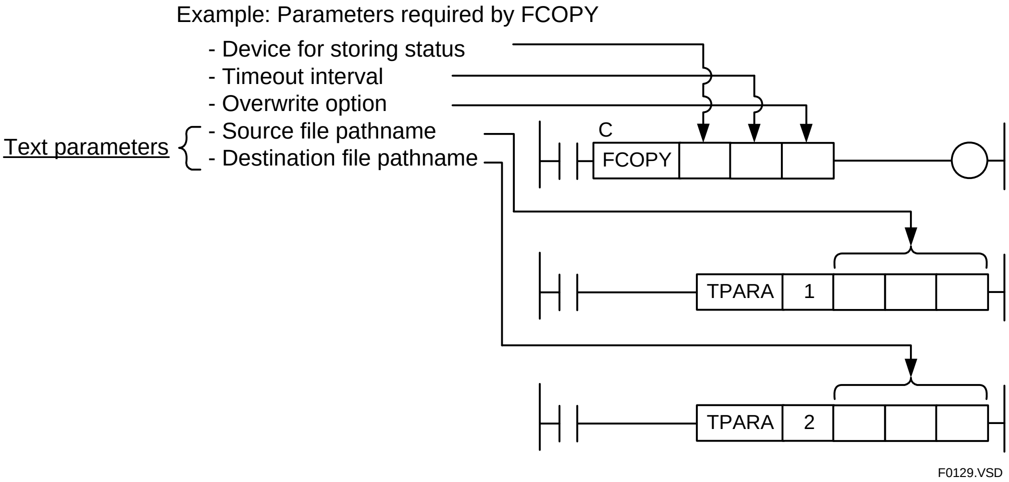


Figure C3.5.5 Text Parameter Number (TPARA instruction must be executed before continuous type application instruction)

The Text Parameter instruction must be executed to set up a text parameter before an instruction requiring the text parameter. Text parameters are stored in the system text parameter area. An instruction requiring a text parameter reads the text parameter from the text parameter area when it begins execution (at the rising edge of the input).

One text parameter area is provided for all continuous type application instructions executing in the normal scan, and another area is provided for all continuous type application instructions executing in the sensor control block. Therefore, normal blocks and the sensor control block do not compete for the text parameter area but continuous type application instructions sharing each area do compete. You should store text parameter value before each instruction execution to ensure proper execution.

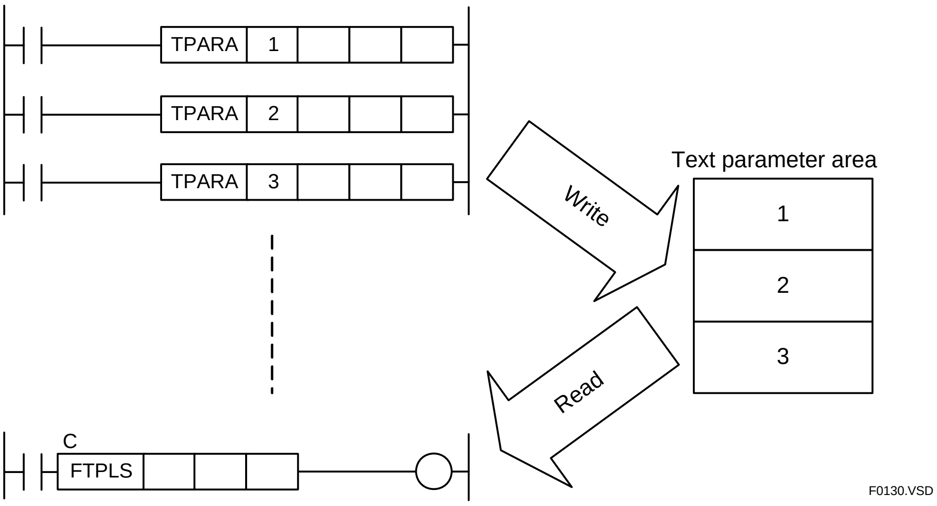


Figure C3.5.6 Text Parameter Area

This instruction performs character string concatenation. It concatenates strings A to C (see next figure) into one text parameter n (n=1 to 3). This string concatenation feature enables user programs to define smaller string units and increase reuse of defined string constants.

Specify NULL for all non-required strings A to C. Using a zero constant value in an instruction parameter is equivalent to specifying a NULL value.

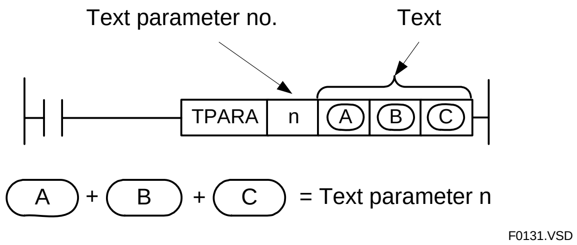


Figure C3.5.7 One Text Parameter

Each text parameter can contain up to 255 characters. The individual lengths and combined length of strings A to C must not exceed 255 characters.

TIP

- Using string concatenation, you can specify as text parameter various string combinations such as string A, string B, string C, string (A+C), string (A+B), etc. Note that all unused parameters must be specified as NULL.

Programming Example

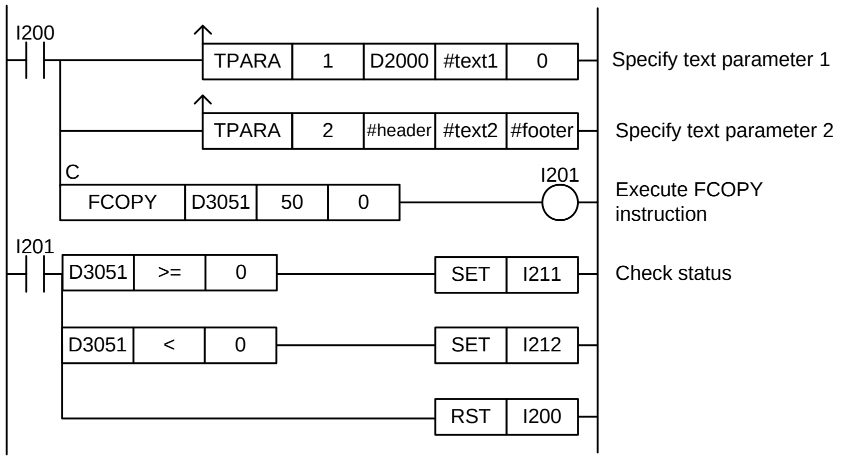


Figure C3.5.8 Example of a Text Parameter Program

This sample code sets up text parameter 1 and text parameter 2, which are to be passed to a Copy File instruction (FCOPY).

The string stored in devices starting with D2000 and the string defined by constant name #text1 are concatenated to become text parameter 1. The three strings defined by constant names #header, #text2 and #footer are concatenated to become text parameter 2.

■ Handling of File Pathname

● Drive Name

A drive name is a disk identifier. On a Windows PC, a disk identifier is typically represented in the form of "C:\" or "D:\". The CPU module has two types of disks, namely, RAM disk and SD memory card, which are assigned the following directory names.

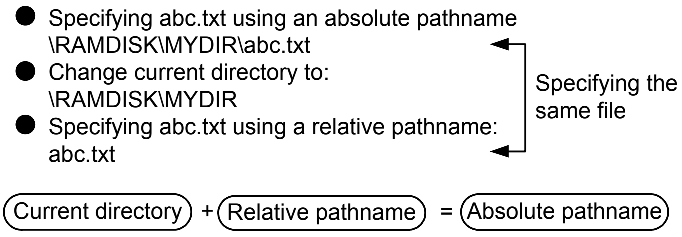
RAM disk	:	\RAMDISK
SD memory card	:	\CARD1

TIP

- The '\' prefix in a drive name indicates the "root directory".
- Each directory or file in the root directory is coded after the drive name, separated by a backslash character (\).

● Relative Pathname and Absolute Pathname

Both relative pathnames and absolute pathnames can be used in file system instructions. Relative pathnames are pathnames relative to the current directory.



- Specifying abc.txt using an absolute pathname
  \RAMDISK\MYDIR\abc.txt
- Change current directory to:
  \RAMDISK\MYDIR
- Specifying abc.txt using a relative pathname:
  abc.txt

Specifying the same file

Current directory + Relative pathname = Absolute pathname

FC0312.VSD

Figure C3.5.9 Relative Pathname and Absolute Pathname

● Current Directory

File system instructions use a common current directory value, which applies only to file system instructions, and is independent of the current directory of FTP and card batch file functions.

The current directory defaults to "\RAMDISK" after module startup.

To change the current directory, use the Change Directory (FCD) instruction. Any change in current directory using the FCD instruction applies thereafter unless and until it is reset to the default value of "RAMDISK" by a power on, reset or Stop mode to Run mode switchover event.

You may not specify a directory below pathname "\VIRTUAL".

TIP

Deleting or moving a directory designated as the current directory does not generate an error.
However, you must redefine the current directory in this case.

● Root Directory

The root directory is the directory at the top of the file hierarchy. It is represented by pathname "\". No file system instruction operations can be applied to the root directory. Doing so generates an error.

C3.5.2 List of File Access Instructions

Table C3.5.12 List of File Access Instructions

Instruction Name	Mnemonic	Function
Open File	FOPEN	Gets access right to a specified file for a specified mode.
Close File	FCLOSE	Releases access right for a specified file.
Read File Line	FGETS	Reads one line from a file.
Write File Line	FPUTS	Writes one line to a file.
Read File Block	FREAD	Reads data of specified size from a file.
Write File Block	FWRITE	Writes data of specified size to a file.
File Seek	FSEEK	Moves the file pointer of a file.
File Text Search	FSEARCHT	Finds a specified string in a file, and moves the file pointer.
File Binary Search	FSEARCHB	Finds specified binary data in a file, and moves the file pointer.
Convert CSV File to Device	F2DCSV	Converts a CSV formatted file into device data.
Convert Device to CSV File	D2FCSV	Converts device data into a CSV formatted file.
Convert Binary File to Device	F2DBIN	Converts a binary file into device data.
Convert Device to Binary File	D2FBIN	Converts device data into a binary file.

Table C3.5.13 List of File Operation Instructions

Instruction Name	Mnemonic	Function
Copy File	FCOPY	Copies one or more files.
Move File	FMOVE	Moves one or more files.
Delete File	FDEL	Deletes one or more files.
Make Directory	FMKDIR	Creates a directory.
Remove Directory	FRMDIR	Deletes a directory.
Rename File	FREN	Renames a file or directory.
File Status	FSTAT	Gets status information of a file or directory.
File List Start	FLSFIRST	Declares a file list operation for getting status information of successive files or directories. FLSFIRST, FLS and FLSFIN instructions are used as a group in a file list operation.
File List Next	FLS	Gets status information of the next file or directory in a pre-declared file list operation. FLSFIRST, FLS and FLSFIN instructions are used as a group in a file list operation.
File List End	FLSFIN	Declares the end of a file list operation. FLSFIRST, FLS and FLSFIN instructions are used as a group in a file list operation.
Change Directory	FCD	Changes the current directory.
Concatenate File	FCAT	Concatenates two files.
Change File Attribute	FATRW	Changes the attribute of a specified file or directory.

Table C3.5.14 List of Disk Operation Instructions

Instruction Name	Mnemonic	Function
Mount Memory Card	MOUNT	Mounts the memory card so that it is ready for use.
Unmount Memory Card	UNMOUNT	Unmounts a memory card so that it can be physically removed.
Format Disk	FORMAT	Formats the SD memory card or RAM disk.
Disk Info	DISKINFO	Gets information on the free space and capacity of a disk.

C3.5.3 File Access Instructions

You use an FOPEN instruction to get access right to a file in the form of a file ID, which can then be used to read from and write to the file using FREAD, FWRITE and other file access instructions. Finally, you use an FCLOSE instruction to release the file ID.

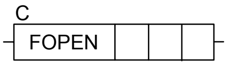
Table C3.5.15 Terminology Description for File Access Instructions

Term	Description
Binary file	A file containing binary data, with no delimiters.
CSV formatted file	A text file in which ASCII coded data elements are delimited by comma (,) characters or TAB characters. A CSV file can be displayed directly in Excel. Conversely, an Excel file can be converted to a CSV formatted file with some limitations. A newline is also considered as a delimiter. Beware that if a newline and a contiguous delimiter character is treated as one field.
Field	A field is one data element in a CSV formatted file.
Record	A record (one line) in a CSV formatted file is delimited by a newline code. One record contains 1 to n fields.

C3.5.3.1 Open File (FOPEN)

Opens a specified file according to the specified open mode so that it is ready for use. At the same time, secures exclusive access right to the file against access by other instructions.

Table C3.5.16 Open File

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro-cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	Open File	FOPEN		✓	–	6	16 bit	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see "■ Continuous Type Application Instructions" in Section C3.5.1, "Using File System Instructions".

■ Parameter

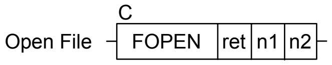


Table C3.5.17 Parameter

Parameter	Description
ret[*1]	Device for storing return status (W)
n1	Timeout interval (W) [0=indefinite, 1-32767 (x 100 ms)]
n2	Open mode (W) [0 = Read-only mode 1 = Write mode (read and write) 2 = Append mode (read and append)]

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

Table C3.5.18 Text Parameter

Parameter		Description
1	n3	Pathname of file to be opened

SEE ALSO

Specify text parameters using the Text Parameter (TPARA) instruction. For details on text parameters and the Text Parameter (TPARA) instruction, see "■ Text Parameter" in Section C3.5.1, "Using File System Instructions".

■ Status (Return Value)

Table C3.5.19 Status (Return Value)

Offset (word)		Description	
ret	ret+0	≥ 0	File ID (W) [0-15] (File is opened successfully)
		< 0	Error status

SEE ALSO

For more details on error status, see "● Error Status of Continuous Type Application Instructions" of "■ Continuous Type Application Instructions" in Section C3.5.1, "Using File System Instructions".

■ Available Devices

Table C3.5.20 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	W	Z	R	V	Con-stant	Index Modification	Indirect Designation, Pointer P
ret									✓	✓	✓		✓	✓		Yes	Yes
n1									✓	✓	✓		✓	✓	✓	Yes	Yes
n2									✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.15, "Restrictions on Devices Used as Instruction Parameters" of "Sequence CPU – Instructions User's Manual" (IM34M6P12-03E).

■ Resource Relays

Table C3.5.21 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
✓	M1026	No Unused File ID	Execute FOPEN instruction if "No Unused File ID" is OFF.
	M1041 to M1056	File ID Open	
	M1057 to M1072	File ID Busy	

■ Function

Opens a specified file according to the specified open mode so that it is ready for use. At the same time, secures exclusive access right to the file against access by other instructions.

If the file is opened successfully, the instruction returns a file ID as return value. This file ID can be used subsequently in file access instructions to access the open file.

There are three types of file open mode, which are described in the table below. Specify the appropriate open mode as required. Files are always opened in binary mode.

Table C3.5.22 Open Modes

Open Mode	Mode No.	Description	Initial File Pointer Position
Read-only mode	0	The read-only mode grants access right for reading a file. Multiple file IDs of read-only mode can be allocated for the same file. File ID of read-only mode can be allocated even for a file that has been opened in write or append mode.	0
Write mode	1	The write mode grants access right for reading and writing a file. When an existing file is opened in write mode, the module deletes the file and creates a new file. For purpose of exclusive control, only one file ID of write mode is allowed for a file at any one time.	0
Append mode	2	The append mode grants access right for appending data to an existing file. If the file does not exist, a new file is created. For purpose of exclusive control, only one file ID of append mode is allowed for a file at any one time.	End of file

CAUTION

- Up to 16 files can be opened concurrently. This maximum limit may be reduced by file operations executed via FTP or other interfaces.
- Do not access the status (return value, offset: +0) during instruction execution processing as it is used by the system.

■ Programming Example

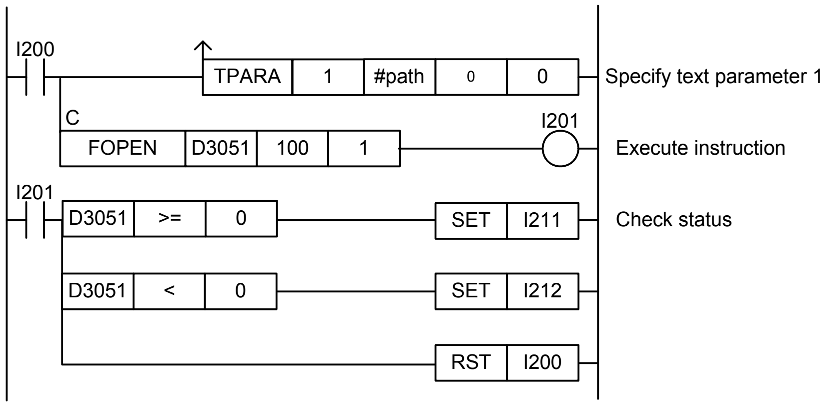


Figure C3.5.10   Example of an Open File Program

This sample code opens "\RAMDISK\myfile.txt" (=#path) in write mode. It specifies D3051 as the device for storing the return status, and specifies the timeout interval as 100 (10 s).

The table below shows the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter Name
ret=D3051	2	Status (file ID)

C3.5.3.2 Close File (FCLOSE)

Closes a file opened with a specified file ID, and releases access right to the file.

Table C3.5.23 Close File

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro-cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	Close File	FCLOSE	C ┤FCLOSE│ │ │ ├	✓	–	6	16 bit	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see "■ Continuous Type Application Instructions" in Section C3.5.1, "Using File System Instructions".

■ Parameter

```
                C
Close File  ┤FCLOSE│ret│n1│n2├
```

Table C3.5.24 Parameter

Parameter	Description
ret*1	Device for storing return status (W)
n1	Timeout interval (W) [0=indefinite, 1-32767 (x 100 ms)]
n2	File ID (W) [0-15]

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

■ Status (Return Value)

Table C3.5.25 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see "● Error Status of Continuous Type Application Instructions" of "■ Continuous Type Application Instructions" in Section C3.5.1, "Using File System Instructions".

■ Available Devices

Table C3.5.26 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	W	Z	R	V	Con-stant	Index Modification	Indirect Designation, Pointer P
ret									✓	✓	✓		✓	✓		Yes	Yes
n1									✓	✓	✓		✓	✓	✓	Yes	Yes
n2									✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.15, "Restrictions on Devices Used as Instruction Parameters" of "Sequence CPU – Instructions User's Manual" (IM34M6P12-03E).

■ Resource Relays

Table C3.5.27 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
	M1026	No Unused File ID	Execute instruction if: "File ID Open" is ON and "File ID Busy" is OFF for the specified file ID.
✓	M1041 to M1056	File ID Open	
✓	M1057 to M1072	File ID Busy	

■ Function

Closes a file opened with a specified file ID, and releases access right to the file.

■ Programming Example

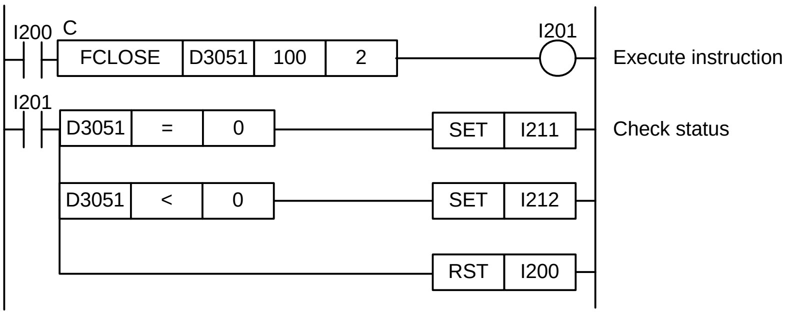


Figure C3.5.11 Example of a Close File Program

This sample code closes the open file designated by file ID 2. It specifies D3051 as the device for storing the return status, and the timeout interval as 10s.

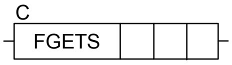
The table below shows the returned status data, assuming normal exit.

Device	Value	Table Parameter Name
ret=D3051	0	Status

C3.5.3.3 Read File Line (FGETS)

Reads one line (up to a newline) from the file associated with a specified file ID.

Table C3.5.28 Read File Line

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Processing Unit	Carry
					Yes	No			
Continuous type application instruction	–	Read File Line	FGETS		✓	–	6	8 bit	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see "■ Continuous Type Application Instructions" in Section C3.5.1, "Using File System Instructions".

■ Parameter

Read File Line

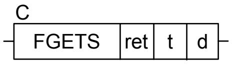


Table C3.5.29 Parameter

Parameter		Description
ret[*1]		Device for storing return status (W)
t	t+0	Timeout interval (W) [0=indefinite, 1-32767 (x 100 ms)]
	t+1	File ID (W) [0-15]
	t+2	Newline option (W) [0=CRLF 1=LF]
	t+3	Delete Newline option (W) [0=No 1=Yes]
	t+4	Append NULL option (W) [0 = No 1 = Yes]
	t+5	Read size limit (W) [1-128 words][*2]
d		First device for storing string (W)

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".
*2: Limit includes appended NULL and newline bytes.

■ Status (Return Value)

Table C3.5.30 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status
	ret+1	No. of bytes read (W) [1-256][*1]	
	ret+2	No. of words written to device (W) [1-128][*1*2]	

*1: Excluding appended NULL, if any.
*2: Rounded up for odd number of bytes.

SEE ALSO

For more details on error status, see "● Error Status of Continuous Type Application Instructions" of "■ Continuous Type Application Instructions" in Section C3.5.1, "Using File System Instructions".

■ Available Devices

Table C3.5.31 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	W	Z	R	V	Con-stant	Index Modification	Indirect Designation, Pointer P
ret									✓	✓	✓		✓	✓		Yes	Yes
t									✓	✓	✓		✓	✓		Yes	Yes
d									✓	✓	✓		✓	✓		Yes	Yes

Note: See Section 1.15, "Restrictions on Devices Used as Instruction Parameters" of "Sequence CPU – Instructions User's Manual" (IM34M6P12-03E).

■ Resource Relays

Table C3.5.32 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
	M1026	No Unused File ID	Execute instruction if: "File ID Open" is ON and "File ID Busy" is OFF for the specified file ID.
✓	M1041 to M1056	File ID Open	
✓	M1057 to M1072	File ID Busy	

■ Function

Reads one line (up to a newline) from the file designated by a specified file ID. It includes an option to delete the newline code from read data.

It also includes an option to append a NULL byte at the end of the read string. When the option is selected, a NULL byte is appended and stored to device.

Up to 256 bytes (128 words) are allowed in one line. This length limit includes appended NULL and newline bytes.

TIP

The file pointer movement is as follows:

- Start position for reading = current file pointer position
- File pointer position after instruction execution = the byte following the last byte that was read

■ Programming Example

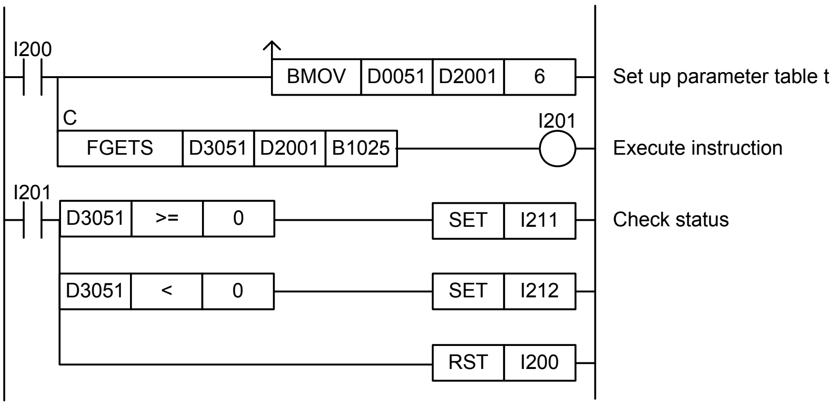


Figure C3.5.12 Example of a Read File Line Program

This sample code reads one line of data from the open file associated with file ID 2.

It specifies ret(=D3051), t(=D2001) and d(=B1025), with t set up as follows.

Device	Value	Table Parameter
t=D2001	10	Timeout interval (= 1 s)
D2002	2	File ID (= 2)
D2003	0	Newline option (= CRLF)
D2004	1	Delete newline option (= Yes)
D2005	1	Append NULL option (= Yes)
D2006	128	Read size limit (= 128 words)

Assuming normal exit and 50 bytes are read, the following data is stored in ret.

Device	Value	Table Parameter
ret=D3051	0	Status
D3052	50	No. of bytes read
D3053	25	No. of words written to device

C3.5.3.4 Write File Line (FPUTS)

Writes one line of text to the file associated with a specified file ID.

Table C3.5.33 Write File Line

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro-cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	Write File Line	FPUTS	C FPUTS	✓	–	5	8 bit	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see "■ Continuous Type Application Instructions" in Section C3.5.1, "Using File System Instructions".

■ Parameter

C

Write File Line FPUTS ret t

Table C3.5.34 Parameter

Parameter		Description
ret[*1]		Device for storing return status (W)
t	t+0	Timeout interval (W) [0=indefinite, 1-32767 (x 100 ms)]
	t+1	File ID (W) [0-15]
	t+2	Append newline option (W) [0 = None 1 = Append CRLF 2 = Append LF]

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

Table C3.5.35 Text Parameter

Parameter		Description
1	s	Write string

SEE ALSO

Specify text parameters using the Text Parameter (TPARA) instruction. For details on text parameters and the Text Parameter (TPARA) instruction, see "■ Text Parameter" in Section C3.5.1, "Using File System Instructions".

■ Status (Return Value)

Table C3.5.36 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status
	ret+1	No. of bytes written to file (W) [1-258][*1]	

*1: Including appended newline bytes.

SEE ALSO

For more details on error status, see "● Error Status of Continuous Type Application Instructions" of "■ Continuous Type Application Instructions" in Section C3.5.1, "Using File System Instructions".

■ Available Devices

Table C3.5.37 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	W	Z	R	V	Con-stant	Index Modification	Indirect Designation, Pointer P
ret									✓	✓	✓		✓	✓		Yes	Yes
t									✓	✓	✓		✓	✓		Yes	Yes

Note: See Section 1.15, "Restrictions on Devices Used as Instruction Parameters" of "Sequence CPU – Instructions User's Manual" (IM34M6P12-03E).

■ Resource Relays

Table C3.5.38 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
	M1026	No Unused File ID	Execute instruction if: "File ID Open" is ON and "File ID Busy" is OFF for the specified file ID.
✓	M1041 to M1056	File ID Open	
✓	M1057 to M1072	File ID Busy	

■ Function

Writes one line of text to the file associated with a specified file ID. Writes characters starting from the specified device up to a NULL character as one line of text to the file. The NULL at the end is not written to the file.

It includes options for appending a newline.

Up to 256 bytes can be written as a line.

TIP

The file pointer movement is as follows:

- Start position for writing = current file pointer position
- File pointer position after instruction execution = the byte following the last written byte

■ Programming Example

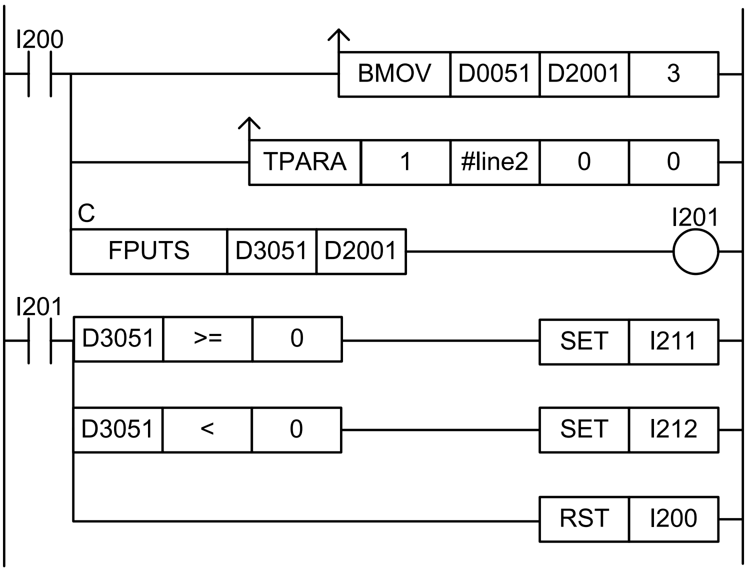


Figure C3.5.13 Example of a Write File Line Program

This sample code writes the string defined by constant #line2 into a file opened as file ID 5.

It specifies ret(=D3051) and t(=D2001), with t set up as shown in the table below.

Device	Value	Table Parameter
t=D2001	10	Timeout interval (= 1 s)
D2002	5	File ID (= 5)
D2003	1	Append newline option (= Append CRLF)

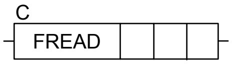
The table below shows the content of ret, assuming normal exit.

Device	Value	Table Parameter
ret=D3051	0	Status
D3052	34	No. of bytes written to file

C3.5.3.5 Read File Block (FREAD)

Reads [block size] x [No. of blocks] bytes from the file associated with a specified file ID.

Table C3.5.39 Read File Block

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Processing Unit	Carry
					Yes	No			
Continuous type application instruction	–	Read File Block	FREAD		✓	–	6	8 bit	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see "■ Continuous Type Application Instructions" in Section C3.5.1, "Using File System Instructions".

■ Parameter

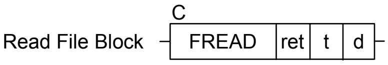


Table C3.5.40 Parameter

Parameter		Description
ret[*1]		Device for storing return status (W)
t	t+0	Timeout interval (W) [0=indefinite, 1-32767 (x 100 ms)]
	t+1	File ID (W) [0-15]
	t+2	Block size low word (L) [1-524288 (byte)]
	t+3	(Block size high word)
	t+4	No. of blocks low word (L) [0-262144]
	t+5	(No. of blocks high word)
	t+6	Read size limit low word (L) [1-262144 (words)]
	t+7	(Read size limit high word)
d		First device for storing read string (W)

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

■ Status (Return Value)

Table C3.5.41 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status
	ret+1	No. of blocks read low word (L) [1-262144]	
	ret+2	(No. of blocks read high word)	

SEE ALSO

For more details on error status, see "● Error Status of Continuous Type Application Instructions" of "■ Continuous Type Application Instructions" in Section C3.5.1, "Using File System Instructions".

■ Available Devices

Table C3.5.42 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	W	Z	R	V	Con-stant	Index Modification	Indirect Designation, Pointer P
ret									✓	✓	✓		✓	✓		Yes	Yes
t									✓	✓	✓		✓	✓		Yes	Yes
d									✓	✓	✓		✓	✓		Yes	Yes

Note: See Section 1.15, "Restrictions on Devices Used as Instruction Parameters" of "Sequence CPU – Instructions User’s Manual" (IM34M6P12-03E).

■ Resource Relays

Table C3.5.43 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
	M1026	No Unused File ID	Execute instruction if: "File ID Open" is ON and "File ID Busy" is OFF for the specified file ID.
✓	M1041 to M1056	File ID Open	
✓	M1057 to M1072	File ID Busy	

■ Function

Reads [block size] x [No. of blocks] bytes from the file associated with a specified file ID. You can specify the number of text blocks to be read.

If an odd number is specified for block size and multiple blocks are read from the file, the blocks will be stored to device with a one-byte gap between blocks. Each block is stored to device, beginning on a word boundary.

If the data read from the file is less than one block size, a block size error (error code -2004) will be generated, and that block will be not included in the number of blocks read stored to status.

TIP

The file pointer movement is as follows:

- Start position for reading = current file pointer position
- File pointer position after instruction execution = the byte following the last byte that was read

CAUTION

Pay attention to device boundary as this instruction may transfer a large amount of data.

■ Programming Example

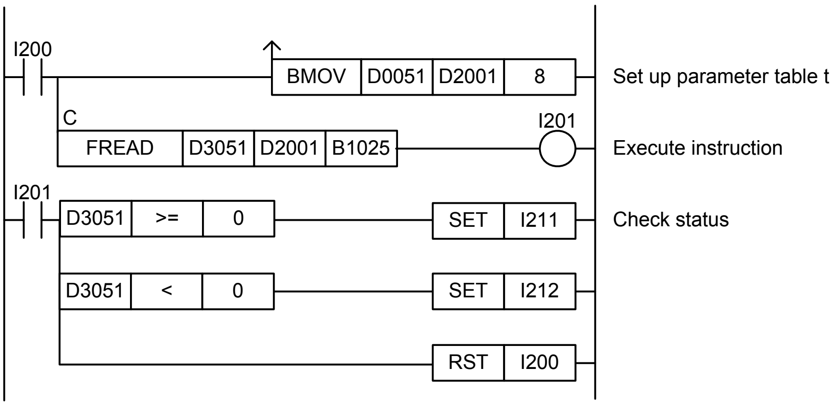


Figure C3.5.14 Example of a Read File Block Program

This sample code reads data blocks from the open file associated with file ID 3.

It specifies ret(=D3051), t(=D2001) and d(=B1025), with t set up as shown in the table below.

Device	Value	Table Parameter
t=D2001	10	Timeout interval (= 1 s)
D2002	3	File ID (=3)
D2003	512	Block size (= 512 bytes)
D2004		
D2005	10	No. of blocks (= 10)
D2006		
D2007	10000	Read size limit (= 10000 words)
D2008		

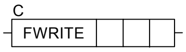
The table below shows an example of the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret=D3051	0	Status
D3052	10	No. of blocks read
D3053		

C3.5.3.6 Write File Block (FWRITE)

Write [block size] x [No. of blocks] bytes into the file associated with a specified file ID.

Table C3.5.44 Write File Block

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Processing Unit	Carry
					Yes	No			
Continuous type application instruction	–	Write File Block	FWRITE		✓	–	6	8 bit	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see "■ Continuous Type Application Instructions" in Section C3.5.1, "Using File System Instructions".

■ Parameter

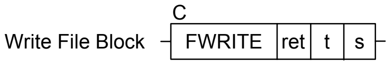


Table C3.5.45 Parameter

Parameter		Description
ret[*1]		Device for storing return status (W)
t	t+0	Timeout interval (W) [0=indefinite, 1-32767 (x 100 ms)]
	t+1	File ID (W) [0-15]
	t+2	Block size low word (L) [1-524288(bytes)]
	t+3	(Block size high word)
	t+4	No. of blocks low word (L) [0-262144]
	t+5	(No. of blocks high word)
s		First device storing write data

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

■ Status (Return Value)

Table C3.5.46 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status
	ret+1	No. of written blocks low word (L) [1 to 262144]	
	ret+2	(No. of written blocks high word)	

SEE ALSO

For more details on error status, see "● Error Status of Continuous Type Application Instructions" of "■ Continuous Type Application Instructions" in Section C3.5.1, "Using File System Instructions".

Available Devices

Table C3.5.47 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	W	Z	R	V	Con-stant	Index Modification	Indirect Designation, Pointer P
ret									✓	✓	✓		✓	✓		Yes	Yes
t									✓	✓	✓		✓	✓		Yes	Yes
s									✓	✓	✓		✓	✓	#	Yes	Yes

Note: See Section 1.15, "Restrictions on Devices Used as Instruction Parameters" of "Sequence CPU – Instructions User’s Manual" (IM34M6P12-03E).
#: Only a defined constant can be specified. Specifying a normal constant is not allowed.

Resource Relays

Table C3.5.48 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
	M1026	No Unused File ID	Execute instruction if: "File ID Open" is ON and "File ID Busy" is OFF for the specified file ID.
✓	M1041 to M1056	File ID Open	
✓	M1057 to M1072	File ID Busy	

Function

Writes [block size] x [No. of blocks] bytes to the file associated with a specified file ID. You can specify the number of text blocks to be written.

If an odd number is specified for block size and multiple blocks are written to the file, each data block will begin on a word boundary.

TIP

The file pointer movement is as follows:

- Start position for writing = current file pointer position
- File pointer position after instruction execution = the byte following the last written byte

CAUTION

Pay attention to device boundary as this instruction may transfer a large amount of data.

■ Programming Example

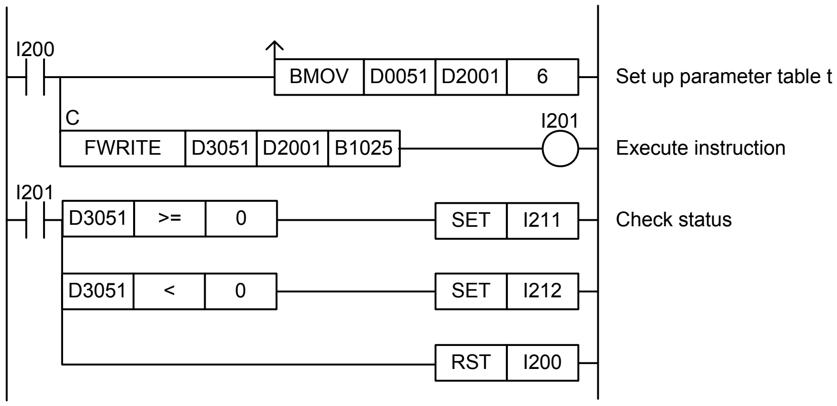


Figure C3.5.15 Example of a Write File Block Program

This sample code writes data blocks stored in device, beginning at B1025 to the file associated with file ID 2.

It specifies ret(=D3051), t(=D2001) and d(=B1025), with t set up as shown in the table below.

Device	Value	Table Parameter
t=D2001	10	Timeout interval (= 1 s)
D2002	2	File ID (=2)
D2003	128	Block size (= 128 bytes)
D2004		
D2005	5	No. of blocks (= 5)
D2006		

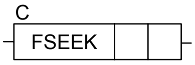
The table below shows an example of the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret=D3051	0	Status
D3052	5	No. of written blocks
D3053		

C3.5.3.7 File Seek (FSEEK)

Moves the file pointer of the file associated with a specified file ID.

Table C3.5.49 File Seek

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro-cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	File Seek	FSEEK		✓	–	5	8 bit	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see "■ Continuous Type Application Instructions" in Section C3.5.1, "Using File System Instructions".

■ Parameter

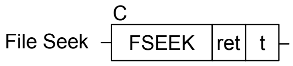


Table C3.5.50 Parameter

Parameter		Description
ret[*1]		Device for storing return status (W)
t	t+0	Timeout interval (W) [0=indefinite, 1-32767 (x 100 ms)]
	t+1	File ID (W) [0-15]
	t+2	Origin (W) [0-2]
	t+3	Offset low word (L) [-2147483648 to 2147483647 (bytes)]
	t+4	(Offset high word)

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

■ Status (Return Value)

Table C3.5.51 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status
	ret+1	File pointer offset from file start after execution (low word) (L)[1-2147483647]	
	ret+2	(File pointer offset from file start after execution (high word))	

SEE ALSO

For more details on error status, see "● Error Status of Continuous Type Application Instructions" of "■ Continuous Type Application Instructions" in Section C3.5.1, "Using File System Instructions".

■ Available Devices

Table C3.5.52 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	W	Z	R	V	Con-stant	Index Modification	Indirect Designation, Pointer P
ret									✓	✓	✓		✓	✓		Yes	Yes
t									✓	✓	✓		✓	✓		Yes	Yes

Note: See Section 1.15, "Restrictions on Devices Used as Instruction Parameters" of "Sequence CPU – Instructions User's Manual" (IM34M6P12-03E).

■ Resource Relays

Table C3.5.53 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
	M1026	No Unused File ID	Execute instruction if: "File ID Open" is ON and "File ID Busy" is OFF for the specified file ID.
✓	M1041 to M1056	File ID Open	
✓	M1057 to M1072	File ID Busy	

■ Function

Moves the file pointer of the file associated with a specified file ID. A file pointer marks the beginning position for reading and writing.

The file pointer destination is specified as an origin and an offset from the origin. The offset is specified as a signed number of bytes, while the origin is specified as one of the following three values.

Table C3.5.54 Origin

Description	Value of Origin
File start	0
Current position	1
File end	2

To determine the current file pointer position, execute the instruction with origin=1 and offset=0 and check the return value.

To determine the file size (file end position), execute the instruction with origin=2 and offset=0 and check the return value.

An attempt to move the file pointer before file start will move the file pointer to file start. An attempt to move the file pointer after file end will move the file pointer to file end. Neither of the above two cases generate an error.

■ Programming Example

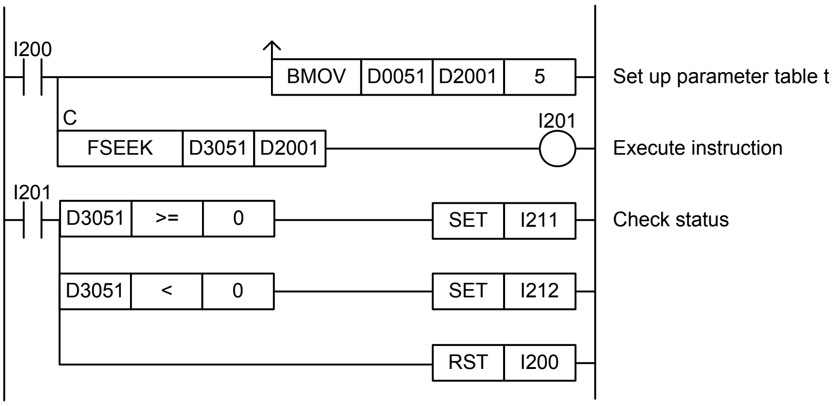


Figure C3.5.16 Example of a File Seek Program

This sample code moves the file pointer of the file opened as file ID 6 to a position, which is 500 bytes from the beginning of the file.

It specifies ret(=D3051) and t(=D2001), with t set up as shown in the table below.

Device	Value	Table Parameter
t=D2001	10	Timeout interval (= 1 s)
D2002	6	File ID (=6)
D2003	0	Origin (= file start)
D2004	500	Offset (= 500 bytes)
D2005		

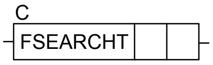
The table below shows an example of the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret=D3051	0	Status
D3052	500	File pointer offset from the specified origin after execution.
D3053		

C3.5.3.8 File Text Search (FSEARCHT)

Searches for a user-specified string within the file associated with a specified file ID.

Table C3.5.55 File Text Search

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Processing Unit	Carry
					Yes	No			
Continuous type application instruction	–	File Text Search	FSEARCHT		✓	–	5	8 bit	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see "■ Continuous Type Application Instructions" in Section C3.5.1, "Using File System Instructions".

■ Parameter

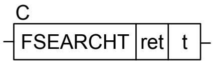
File Text Search 

Table C3.5.56 Parameter

Parameter		Description
ret[*1]		Device for storing return status (W)
t	t+0	Timeout interval (W) [0=indefinite, 1-32767 (x 100 ms)]
	t+1	File ID (W) [0-15]

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

Table C3.5.57 Text Parameter

Parameter		Description
1	s	Search string

SEE ALSO

Specify text parameters using the Text Parameter (TPARA) instruction. For details on text parameters and the Text Parameter (TPARA) instruction, see "■ Text Parameter" in Section C3.5.1, "Using File System Instructions".

■ Status (Return Value)

Table C3.5.58 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status
	ret+1, ret+2	Match location [>0 : Byte offset of match from file start (L) -1 : No match found]	

SEE ALSO

For more details on error status, see "● Error Status of Continuous Type Application Instructions" of "■ Continuous Type Application Instructions" in Section C3.5.1, "Using File System Instructions".

■ Available Devices

Table C3.5.59 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	W	Z	R	V	Con-stant	Index Modification	Indirect Designation, Pointer P
ret									✓	✓	✓		✓	✓		Yes	Yes
t									✓	✓	✓		✓	✓		Yes	Yes

Note: See Section 1.15, "Restrictions on Devices Used as Instruction Parameters" of "Sequence CPU – Instructions User's Manual" (IM34M6P12-03E).

■ Resource Relays

Table C3.5.60 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
	M1026	No Unused File ID	Execute instruction if: "File ID Open" is ON and "File ID Busy" is OFF for the specified file ID.
✓	M1041 to M1056	File ID Open	
✓	M1057 to M1072	File ID Busy	

■ Function

Searches for a user-specified string within the file associated with a specified file ID.

TIP

The file pointer movement is as follows:

- Starting position for search = current file pointer position
- File pointer position after instruction execution (if a match is found) = the byte following matched data
- File pointer position after instruction execution (if no match is found) = file pointer position before the search

CAUTION

If the instruction is cancelled or transition is made to Stop mode during execution and file size is large (megabytes to gigabytes), it may take some time (from several seconds to several minutes) for processing to be completed.

■ Programming Example

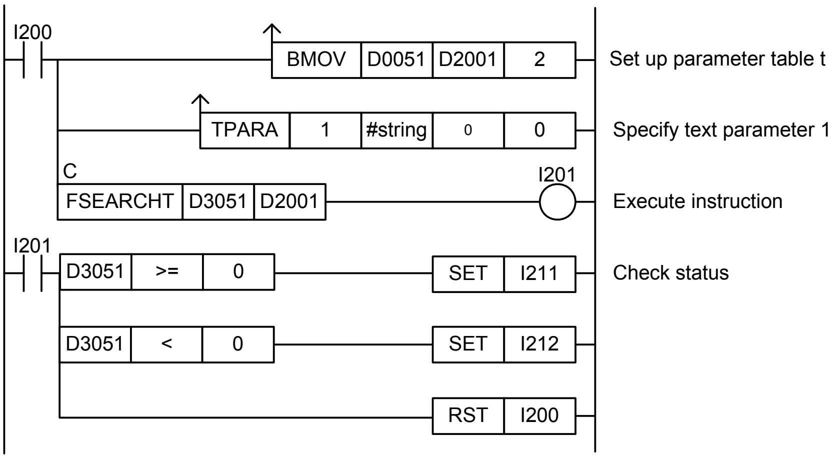


Figure C3.5.17 Example of a File Text Search Program

This sample code searches for the string defined by constant name #string within a file opened as file ID 2.

It specifies ret(=D3051) and t(=D2001), with t set up as shown in the table below.

Device	Value	Table Parameter
t=D2001	100	Timeout interval (= 10 s)
D2002	2	File ID (= 2)

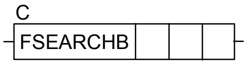
The table below shows an example of the returned status (ret), assuming a match is found at byte offset 3044 from the beginning of the file.

Device	Value	Table Parameter
ret=D3051	0	Status
D3052	3044	Match location
D3053		

C3.5.3.9 File Binary Search (FSEARCHB)

Searches for user-specified binary data within the file associated with a specified file ID.

Table C3.5.61 File Binary Search

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro-cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	File Binary Search	FSEARCHB		✓	–	6	8 bit	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see "■ Continuous Type Application Instructions" in Section C3.5.1, "Using File System Instructions".

■ Parameter

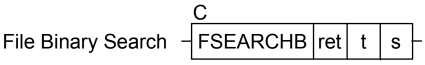


Table C3.5.62 Parameter

Parameter		Description
ret[*1]		Device for storing return status (W)
t	t+0	Timeout interval (W) [0=indefinite, 1-32767 (x 100 ms)]
	t+1	File ID (W) [0-15]
	t+2	Binary search data size (W) [0-32767 (bytes)]
s		First device of binary search data (W)

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

■ Status (Return Value)

Table C3.5.63 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status
	ret+1, ret+2	Match location [>0 : Byte location of match (L) -1 : No match found]	

SEE ALSO

For more details on error status, see "● Error Status of Continuous Type Application Instructions" of "■ Continuous Type Application Instructions" in Section C3.5.1, "Using File System Instructions".

■ Available Devices

Table C3.5.64 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	W	Z	R	V	Con-stant	Index Modification	Indirect Designation, Pointer P
ret									✓	✓	✓		✓	✓		Yes	Yes
t									✓	✓	✓		✓	✓		Yes	Yes
s									✓	✓	✓		✓	✓	#	Yes	Yes

Note: See Section 1.15, "Restrictions on Devices Used as Instruction Parameters" of "Sequence CPU – Instructions User’s Manual" (IM34M6P12-03E).
#: Only a defined constant can be specified. Specifying a normal constant is not allowed.

■ Resource Relays

Table C3.5.65 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
	M1026	No Unused File ID	Execute instruction if: "File ID Open" is ON and "File ID Busy" is OFF for the specified file ID.
✓	M1041 to M1056	File ID Open	
✓	M1057 to M1072	File ID Busy	

■ Function

Searches for user-specified binary data within the file associated with a specified file ID.

TIP

The file pointer movement is as follows:

- Starting position for search = current file pointer position
- File pointer position after instruction execution (if a match is found) = the byte following matched data
- File pointer position after instruction execution (if no match is found) = file pointer position before the search

CAUTION

If the instruction is cancelled during execution and file size is large (megabytes to gigabytes), it may take some time (from several seconds to several minutes) for the transition to stop mode to be completed.

■ Programming Example

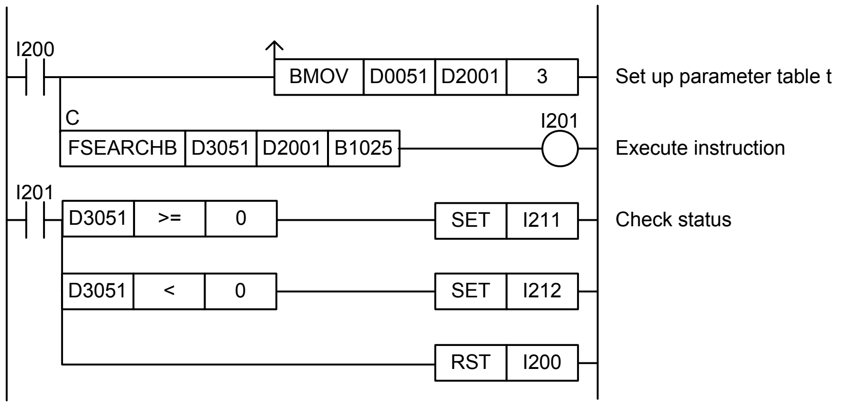


Figure C3.5.18 Example of a File Binary Search Program

This sample code searches for 20 bytes of device data starting at B1025 within the file opened as file ID 2.

It specifies ret(=D3051), t(=D2001) and s(=B1025), with t set up as follows.

Device	Value	Table Parameter
t=D2001	100	Timeout interval (= 10 s)
D2002	2	File ID (=2)
D2003	20	Size of search data (= 20)

The table below shows an example of the returned status (ret), assuming a match is found at byte offset 3044 from the beginning of the file.

Device	Value	Table Parameter
ret=D3051	0	Status
D3052	3044	Match location
D3053		

C3.5.3.10 Convert CSV File to Device (F2DCSV)

Converts data in CSV formatted file to binary data and writes the data to contiguous devices.

Table C3.5.66 Convert CSV File to Device

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Processing Unit	Carry
					Yes	No			
Continuous type application instruction	–	Convert CSV File to Device	F2DCSV	C F2DCSV	✓	–	6	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see "■ Continuous Type Application Instructions" in Section C3.5.1, "Using File System Instructions".

■ Parameter

Convert CSV File to Device C F2DCSV | ret | t | d

Table C3.5.67 Parameter

Parameter		Description
ret[*1]		Device for storing return status (W)
t	t+0	Timeout interval (W) [0=indefinite, 1-32767 (x 100 ms)]
	t+1	File ID (W) [0-15]
	t+2, t+3	No. of fields to be read (L) [-1 = until file end 0 - 4194304 (if device unit = bit) 0 - 524288 (if device unit = byte) 0 - 262144 (if device unit = word) 0 - 131072 (if device unit = long word)]
	t+4	Field representation type (W) [0 = Decimal 1 = Hexadecimal 2 = Floating-point representation A ([-]d.dddd e[+/-]ddd form) 3 = Floating-point representation B ([-]dddd.dddd form)]
	t+5	Device unit (W) [0 = Bit 1 = Byte 2 = Word 3 = Long word]
	t+6	Sign extension (W) [0 = Pad with zeros 1 = Extend sign]
	t+7	Delimiter option (W) [0 = Comma (,) 1 = TAB]
	t+8	Newline option [0 = CRLF 1 = LF]
	t+9, t+10	Write limit in words (L) [1-262144 (words)]
d		First device for writing (W)

[*1] ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

■ Status (Return Value)

Table C3.5.68 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status
	ret+1	No. of fields read (low word) (L) [0-2147483647 (fields)]	
	ret+2	(No. of fields read high word)	
	ret+3	No. of written words low word (L) [0-2147483647 (words)]	
	ret+4	(No. of written words high word)	

SEE ALSO

For more details on error status, see "● Error Status of Continuous Type Application Instructions" of "■ Continuous Type Application Instructions" in Section C3.5.1, "Using File System Instructions".

■ Available Devices

Table C3.5.69 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	W	Z	R	V	Con-stant	Index Modification	Indirect Designation, Pointer P
ret									✓	✓	✓		✓	✓		Yes	Yes
t									✓	✓	✓		✓	✓		Yes	Yes
d		✓	✓	✓	✓				✓	✓	✓		✓	✓		Yes	Yes

Note: See Section 1.15, "Restrictions on Devices Used as Instruction Parameters" of "Sequence CPU – Instructions User's Manual" (IM34M6P12-03E).

■ Resource Relays

Table C3.5.70 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
	M1026	No Unused File ID	Execute instruction if: "File ID Open" is ON and "File ID Busy" is OFF for the specified file ID.
✓	M1041 to M1056	File ID Open	
✓	M1057 to M1072	File ID Busy	

■ Function

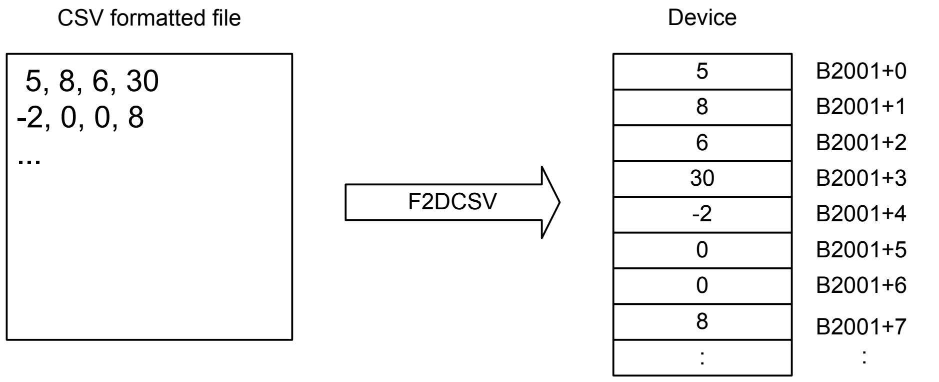
Converts data in CSV formatted file to binary data and writes the data to contiguous devices.

- Text in decimal, hexadecimal or floating-point representation can be converted to device data. Floating-point representation is converted to IEEE single-precision floating-point representation.

  Decimal ("-128" to "255", "-32768" to "65535", "-2147483648" to "4294967295")

  Hexadecimal ("0x0" to "0xFFFFFFFF", "0" to "FFFFFFFF")

  Floating-point ([-]d.dddd e[+/-]ddd, [-]dddd.dddd, Infinite "-INF"/"+INF")
- Available device unit options are bit, byte, word and long word. You can also specify whether to perform sign extension.
- Available field delimiter options are the comma (,) and Tab characters.
- Comments can be included in the file. If a field begins with a double-quote ("), single-quote ('), two slashes (//), or a slash and an asterisk (/*), the instruction skips over all characters until it encounters a delimiter character or newline.
- Newline can be specified as CRLF (standard for Windows) or LF.



Note: Device numbers and conversion method shown are examples.

FB0212.VSD

Figure C3.5.19 CSV Formatted File to Device Conversion

TIP

Reading of File

- If end-of-file is encountered before the required number of fields is read, execution ends without error.
- A newline ends a record, and thus always ends a field.
- Within a field, any and all space characters preceding the data string are ignored but any space character following the data string results in a field conversion error.
- Double slashes (""//") and other comment mark characters must always be coded at the beginning of a field. Otherwise, a conversion error will be generated.
- If NULL or other invalid binary code is encountered, execution ends with a file interpretation error.

Conversion Error and Interpretation Error

- If a conversion error is detected, 0 is written to the device. If a conversion error is detected in a field during conversion, an error is generated but processing continues.
- When the converted numeric value of a field exceeds the range of the device unit, a conversion error is generated.
- Non-numeric representation "NaN" of D2FCSV generates a conversion error.

Data Conversion and Writing to Device

- You can specify to pad with '0's or extend the sign when the converted value of a field is smaller than the size of the device unit.
- If the device unit is specified as bit, 0 is stored for a zero value while 1 is stored for any other value.
- If you specify the field representation type as floating-point representation, you must specify the device unit as long word.
- Writing to device spans multiple scan cycles.

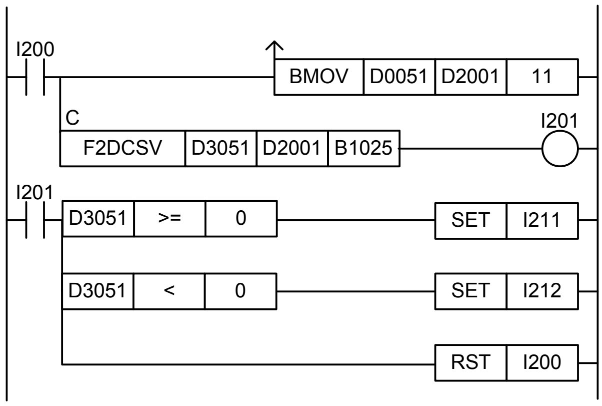
File Pointer Movement

- Start position for reading = current file pointer position
- File pointer position after instruction execution = the byte following the last byte that was read

CAUTION

Pay attention to the size of data read when specifying "until file end" as the number of fields to be read.

■ Programming Example



Set up parameter table t

Execute instruction

Check status

Figure C3.5.20 Example of a Convert CSV File to Device Program

This sample code reads the data contained in a CSV formatted file opened as file ID 4 according to the conditions set up in parameter table t.

It specifies ret(=D3051), t(=D2001) and d(=B1025), with t set up as shown in the table below.

Device	Value	Table Parameter
t=D2001	100	Timeout interval (= 10 s)
D2002	4	File ID (= 4)
D2003	2048	No. of fields to be read (= 2048 fields)
D2004		
D2005	0	Field representation type (= decimal)
D2006	2	Device unit (= word)
D2007	1	Sign extension (= Extend sign)
D2008	0	Delimiter option (= comma)
D2009	0	Newline option (= CRLF)
D2010	10000	Write limit in words (= 10000 words)
D2011		

The table below shows an example of the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret=D3051	0	Status
D3052	2048	No. of fields read
D3053		
D3054	2048	No. of written words
D3055		

C3.5.3.11 Convert Device to CSV File (D2FCSV)

Converts device data to text and outputs a CSV formatted file.

Table C3.5.71 Convert Device to CSV File

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro-cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	Convert Device to CSV File	D2FCSV	C D2FCSV	✓	–	6	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see "■ Continuous Type Application Instructions" in Section C3.5.1, "Using File System Instructions".

■ Parameter

Convert Device to CSV File
C
D2FCSV | ret | t | s

Table C3.5.72 Parameter

Parameter		Description
ret[*1]		Device for storing return status (W)
t	t+0	Timeout interval (W) [0=indefinite, 1-32767 (x 100 ms)]
	t+1	File ID (W) [0-15]
	t+2	Device unit (W) [0 = bit 1 = byte 2 = word 3 = long word]
	t+3, t+4	No. of data units to be read (L) [0 - 4194304 (if device unit = bit) 0 - 524288 (if device unit = byte) 0 - 262144 (if device unit = word) 0 - 131072 (if device unit = long word)]
	t+5	Field representation type (W) [0 = Decimal 1 = Hexadecimal 2 = Floating-point representation A ([-]d.dddd e[+/-]ddd form) 3 = Floating-point representation B ([-]dddd.dddd form)]
	t+6	Field length (W) [0 = automatic (as required after conversion) 1-13 = fixed field length in characters]
	t+7	Field space handling (W) [[*2] 0 = Pad with spaces 1 = Pad with zeros]
	t+8	Delimiter option (W) [0 = comma (,) 1 =TAB]
	t+9	Newline option (W) [0 = CRLF 1 = LF]
	t+10	Newline insertion position (W) [0 = Do not insert newline 1 - 32767 = Insert newline after n fields]
s		First device for reading (W)

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

*2: If the field length is specified as 0 (automatic), this parameter is ignored but a valid dummy value (say, 0) must still be specified.

■ Status (Return Value)

Table C3.5.73 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status
	ret+1	No. of data units written to file (low word) (L) [0-4194304 (data count)]	
	ret+2	(No. of data units written to file (high word))	
	ret+3	No. of bytes written to file (low word) (L) [0 to 2147483647 (bytes)]	
	ret+4	(No. of bytes written to file (high word))	

SEE ALSO

For more details on error status, see "● Error Status of Continuous Type Application Instructions" of "■ Continuous Type Application Instructions" in Section C3.5.1, "Using File System Instructions".

■ Available Devices

Table C3.5.74 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	W	Z	R	V	Con-stant	Index Modification	Indirect Designation, Pointer P
ret									✓	✓	✓		✓	✓		Yes	Yes
t									✓	✓	✓		✓	✓		Yes	Yes
s	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		Yes	Yes

Note: See Section 1.15, "Restrictions on Devices Used as Instruction Parameters" of "Sequence CPU – Instructions User's Manual" (IM34M6P12-03E).

■ Resource Relays

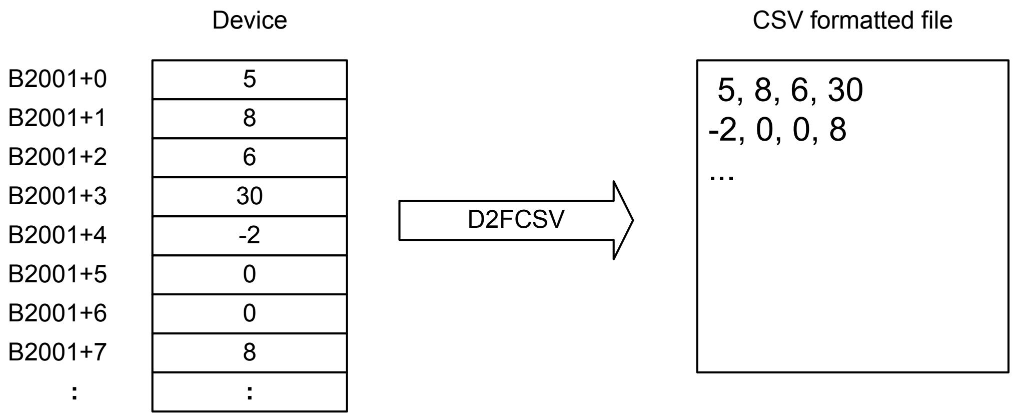
Table C3.5.75 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
	M1026	No Unused File ID	Execute instruction if: "File ID Open" is ON and "File ID Busy" is OFF for the specified file ID.
✓	M1041 to M1056	File ID Open	
✓	M1057 to M1072	File ID Busy	

■ Function

Converts device data to text and outputs a CSV formatted file.

- Available device unit options for reading are bit, byte, word and long word.
- Device data can be converted to text in decimal, hexadecimal or floating-point representation after reading.
  Decimal ("0" to "1", "-128" to "127", "-32768" to "32767", "-2147483648" to "2147483647")
  Hexadecimal ("0" to "FFFFFFFF")
  Floating-point ([-]d.dddd e[+/-]ddd, [-]dddd.dddd, infinity "-INF" or "+INF", non-numeric "NaN")
- You can specify the field length in characters for text conversion.
- You can specify whether to pad with space characters or pad with zeros when the converted text is shorter than the specified field length.
- Available field delimiter options are the comma (,) and Tab characters.
- Newline can be specified as CRLF (standard for Windows) or LF.
- The number of fields in one record (from line beginning to line end) can be specified.



Note: Device numbers and conversion method shown are examples.

FB0213.VSD

Figure C3.5.21 Device to CSV Formatted File Conversion

TIP

Reading of Device Data

- Reading of data from devices spans multiple scan cycles.
- If you specify the field representation type as floating-point representation, you must specify the device unit as long word for devices storing IEEE single-precision floating-point numbers.

Conversion Error and Interpretation Error

- If a conversion error occurs, "ERR" is written to the field. If a conversion error is detected for a field during conversion, an error is generated but processing continues.
- If the converted text string is longer than the specified field length, a conversion error is generated.

Data Conversion

- If the field representation type is specified as decimal, the sign is included in the output digit count.
- In floating-point representation B, there are always 6 digits after the decimal point. Numeric values smaller than 0.000001 are rounded to 0.

File Pointer Movement

- Start position for writing = current file pointer position
- File pointer position after instruction execution = the byte following the last written byte

■ Programming Example

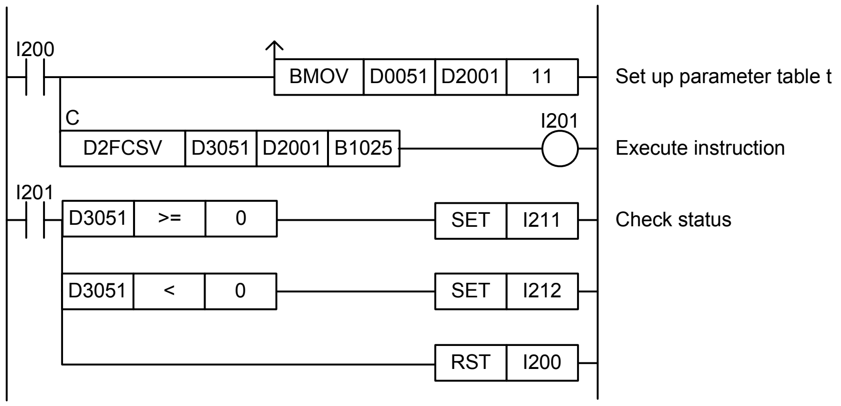


Figure C3.5.22 Example of a Convert Device to CSV File Program

This sample code writes device data starting from B1025 to a file opened as file ID 7 according to the conditions set up in parameter table t.

It specifies ret(=D3051), t(=D2001) and s(=B1025), with t set up as follows.

Device	Value	Table Parameter
t=D2001	100	Timeout interval (= 10 s)
D2002	7	File ID (= 7)
D2003	2	Device unit (= word)
D2004	300	No. of data units to be read (= 300)
D2005		
D2006	1	Field representation type (= hexadecimal)
D2007	4	Field length (= 4 characters)
D2008	0	Field space handling (= pad with spaces)
D2009	0	Delimiter option (= comma)
D2010	0	Newline option (= CRLF)
D2011	30	Newline insertion position (= every 30 fields)

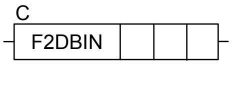
The table below shows an example of the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret=D3051	0	Status
D3052	300	No. of data units written to file
D3053		
D3054	1510	No. of bytes written to file
D3055		

C3.5.3.12 Convert Binary File to Device (F2DBIN)

Converts data in binary file and writes the data to contiguous devices using the specified data unit.

Table C3.5.76 Convert Binary File to Device

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro-cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	Convert Binary File to Device	F2DBIN		✓	–	6	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see "■ Continuous Type Application Instructions" in Section C3.5.1, "Using File System Instructions".

■ Parameter

Convert Binary File to Device

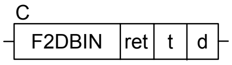


Table C3.5.77 Parameter

Parameter		Description
ret[*1]		Device for storing return status (W)
t	t+0	Timeout interval (W) [0=indefinite, 1-32767 (x 100 ms)]
	t+1	File ID (W) [0-15]
	t+2, t+3	No. of data units to be read (L) [-1 = Until file end 0 - 4194304 (if device unit = bit) 0 - 524288 (if device unit = byte) 0 - 262144 (if device unit = word) 0 - 131072 (if device unit = long word)]
	t+4	Data unit (W) [1 = byte 2 = word 3 = long word]
	t+5	Device unit (W) [0 = bit 1 = byte 2 = word 3 = long word]
	t+6	Sign extension (W) [0 = Pad with zeros 1 = Extend sign]
	t+7, t+8	Write limit in words (L) [1 - 262144 (words)]
d		First device for writing (W)

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

■ Status (Return Value)

Table C3.5.78 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status
	ret+1	No. of data units read (low word) (L) [0-2147483647 (data count)]	
	ret+2	(No. of data units read high word)	
	ret+3	No. of written words low word (L) [0-2147483647 (words)]	
	ret+4	(No. of written words high word)	

SEE ALSO

For more details on error status, see "● Error Status of Continuous Type Application Instructions" of "■ Continuous Type Application Instructions" in Section C3.5.1, "Using File System Instructions".

■ Available Devices

Table C3.5.79 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	W	Z	R	V	Con-stant	Index Modification	Indirect Designation, Pointer P
ret									✓	✓	✓		✓	✓		Yes	Yes
t									✓	✓	✓		✓	✓		Yes	Yes
d		✓	✓	✓	✓				✓	✓	✓		✓	✓		Yes	Yes

Note: See Section 1.15, "Restrictions on Devices Used as Instruction Parameters" of "Sequence CPU – Instructions User's Manual" (IM34M6P12-03E).

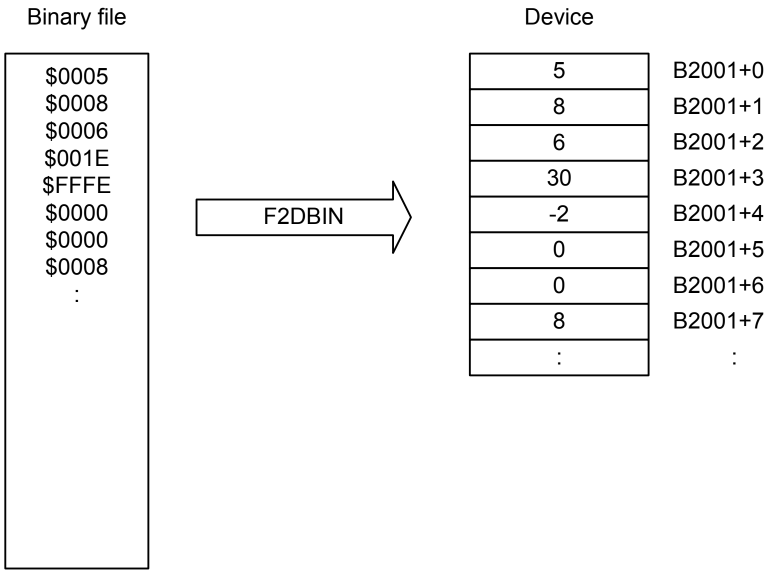
■ Resource Relays

Table C3.5.80 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
	M1026	No Unused File ID	Execute instruction if: "File ID Open" is ON and "File ID Busy" is OFF for the specified file ID.
✓	M1041 to M1056	File ID Open	
✓	M1057 to M1072	File ID Busy	

■ Function

Converts data in binary file and writes the data to contiguous devices using the specified data unit.



Note: Device numbers and conversion method shown are examples.

FB0214.VSD

Figure C3.5.23 Binary File to Device Conversion

TIP

- If a conversion error occurs, 0 is written to the device. If a conversion error is detected during conversion, an error is generated but processing continues.
- If the specified data unit is larger than the specified device unit, a conversion error is generated.
- If the device unit is specified as bit, 0 is stored for a zero value while 1 is stored for any other value.
- If the specified data unit is smaller than the specified device unit, you can specify to pad with '0's or extend the sign.
- If end-of-file is encountered before the required number of data units are read, execution ends without error.
- Writing to device spans multiple scan cycles.

TIP

The file pointer movement is as follows:

- Start position for reading = current file pointer position
- File pointer position after instruction execution = the byte following the last byte that was read

CAUTION

Pay attention to the size of data read when specifying "until file end" as the number of data units to be read.

■ Programming Example

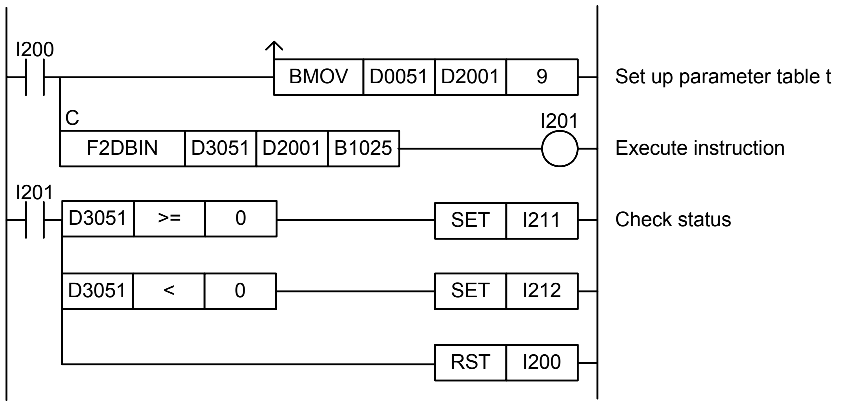


Figure C3.5.24 Example of a Convert Binary File to Device Program

This sample code reads the data contained in a binary file opened as file ID 4 according to the conditions set up in parameter table t.

It specifies ret(=D3051), t(=D2001) and d(=B1025), with t set up as shown in the table below.

Device	Value	Table Parameter Name
t=D2001	100	Timeout interval (= 10 s)
D2002	4	File ID (= 4)
D2003	-1	No. of data units to be read (= until file end)
D2004		
D2005	2	Data unit for reading (= word)
D2006	2	Device unit (= word)
D2007	0	Sign extension (irrelevant for this sample program)
D2008	10000	Write limit in words (= 10000 words)
D2009		

The table below shows an example of the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret=D3051	0	Status
D3052	4066	No. of data units read
D3053		
D3054	4066	No. of written words
D3055		

C3.5.3.13 Convert Device to Binary File (D2FBIN)

Converts device data to a binary file.

Table C3.5.81 Convert Device to Binary File

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Processing Unit	Carry
					Yes	No			
Continuous type application instruction	–	Convert Device to Binary File	D2FBIN	C D2FBIN	✓	–	6	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see "■ Continuous Type Application Instructions" in Section C3.5.1, "Using File System Instructions".

■ Parameter

Convert Device to Binary File

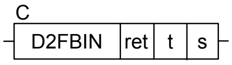


Table C3.5.82 Parameter

Parameter		Description
ret[*1]		Device for storing return status (W)
t	t+0	Timeout interval (W) [0=indefinite, 1-32767 (x 100 ms)]
	t+1	File ID (W) [0-15]
	t+2, t+3	No. of data units to be read (L) [0 - 4194304 (if device data unit = bit) 0 - 524288 (if device data unit = byte) 0 - 262144 (if device data unit = word) 0 - 131072 (if device data unit = long word)]
	t+4	Device data unit (W) [0 = bit 1 = byte 2 = word 3 = long word]
	t+5	File data unit (W) [1 = byte 2 = word 3 = long word]
	t+6	Sign extension (W) [0 = Pad with zeros 1 = Extend sign]
s		First device for reading (W)

[*1] ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

■ Status (Return Value)

Table C3.5.83 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status
	ret+1	No. of data units written to file (low word) (L) [0-4194304 (data count)]	
	ret+2	(No. of data units written to file (high word))	
	ret+3	No. of bytes written to file (low word) (L) [0-2147483647 (bytes)]	
	ret+4	(No. of bytes written to file (high word))	

SEE ALSO

For more details on error status, see "● Error Status of Continuous Type Application Instructions" of "■ Continuous Type Application Instructions" in Section C3.5.1, "Using File System Instructions".

■ Available Devices

Table C3.5.84 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	W	Z	R	V	Con-stant	Index Modification	Indirect Designation, Pointer P
ret									✓	✓	✓		✓	✓		Yes	Yes
t									✓	✓	✓		✓	✓		Yes	Yes
s	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		Yes	Yes

Note: See Section 1.15, "Restrictions on Devices Used as Instruction Parameters" of "Sequence CPU – Instructions User's Manual" (IM34M6P12-03E).

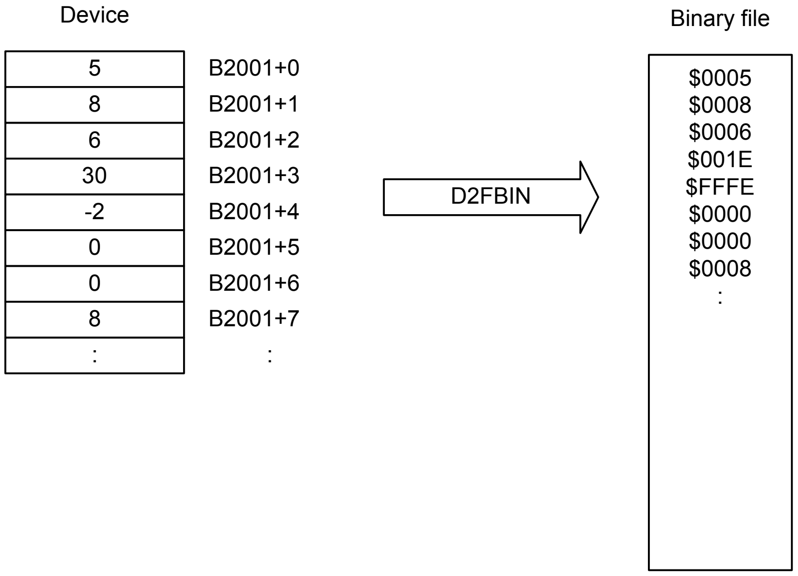
■ Resource Relays

Table C3.5.85 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
	M1026	No Unused File ID	Execute instruction if: "File ID Open" is ON and "File ID Busy" is OFF for the specified file ID.
✓	M1041 to M1056	File ID Open	
✓	M1057 to M1072	File ID Busy	

■ Function

Converts device data to a binary file.



Note: Device numbers and conversion method shown are examples.

FB0215.VSD

Figure C3.5.25 Device Data to Binary File Conversion

TIP

- If a conversion error occurs, 0 is written to the file. If a conversion error is detected during conversion, an error is generated but processing continues.
- If the specified device unit is larger than the specified file unit, a conversion error is generated.
- If the specified device unit is smaller than the specified file unit, you can specify to pad with '0's or extend the sign.
- Reading of data from devices spans multiple scan cycles.

TIP

The file pointer movement is as follows:

- Start position for writing = current file pointer position
- File pointer position after instruction execution = the byte following the last written byte

■ Programming Example

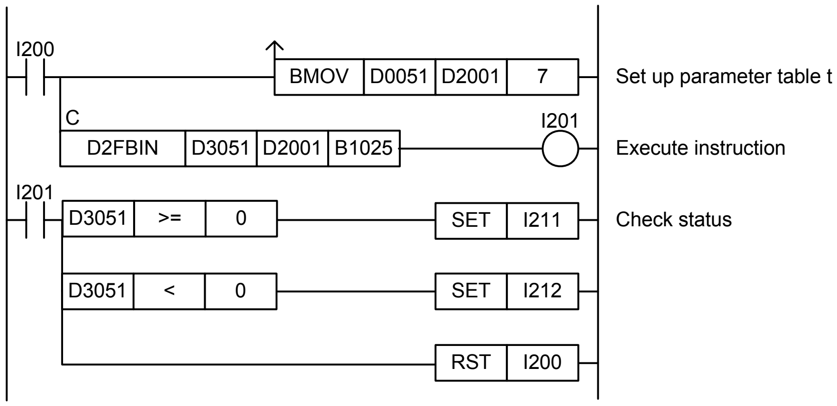


Figure C3.5.26 An Example of a Convert Device to Binary File Program

This sample code writes device data starting from B1025 to a file opened as file ID 7 according to the conditions set up in parameter table t.

It specifies ret(=D3051), t(=D2001) and s(=B1025), with t set up as follows.

Device	Value	Table Parameter
t=D2001	100	Timeout interval (= 10 s)
D2002	7	File ID (= 7)
D2003	300	No. of data units to be read (= 300)
D2004		
D2005	2	Device data unit (= word)
D2006	3	File data unit (= long word)
D2007	1	Sign extension (= Extend sign)

The table below shows an example of the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret=D3051	0	Status
D3052	300	No. of data units written to file
D3053		
D3054	1200	No. of bytes written to file
D3055		

C3.5.4 File Operation Instructions

File operation instructions performs file based processing such as copying or moving a file.

When using a file operation instruction, you directly specify a file using its filename. Unlike file access instructions, there is no need to open a file using the FOPEN instruction. The system automatically gets the required file ID internally.

Of the file operation instruction group and disk operation instruction group, only instructions from one group can be executed at any one time. If you attempt to execute multiple instructions, the instruction that is executed later will be terminated with a repeat use of function error (error code -3001). Before executing a file operation instruction, check to ensure that the File/Disk Operation Group Busy relay (M1025) is OFF.

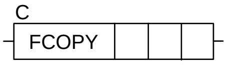
TIP

File operation instructions automatically get the required file IDs, sharing a common pool of available file IDs with other file system processes. If many files are opened by file access instructions or other file system processes so that no more unused file IDs are available, file operations instructions cannot be executed successfully.

C3.5.4.1 Copy File (FCOPY)

Copies one or more files.

Table C3.5.86 Copy File

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro-cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	Copy File	FCOPY		✓	–	6	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see "■ Continuous Type Application Instructions" in Section C3.5.1, "Using File System Instructions".

■ Parameter

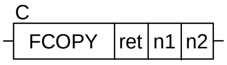
Copy File 

Table C3.5.87 Parameter

Parameter	Description
ret[*1]	Device for storing return status (W)
n1	Timeout interval (W) [0=indefinite, 1-32767 (x 100 ms)]
n2	Overwrite option (W) [0 = Exit with error without overwriting 1 = Overwrite file of the same name 2 = Overwrite read-only file of the same name]

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

Table C3.5.88 Text Parameter

Parameter		Description
1	s	Source pathname
2	d	Destination pathname

SEE ALSO

Specify text parameters using the Text Parameter (TPARA) instruction. For details on text parameters and the Text Parameter (TPARA) instruction, see "■ Text Parameter" in Section C3.5.1, "Using File System Instructions".

■ Status (Return Value)

Table C3.5.89 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see "● Error Status of Continuous Type Application Instructions" of "■ Continuous Type Application Instructions" in Section C3.5.1, "Using File System Instructions".

Available Devices

Table C3.5.90 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	W	Z	R	V	Con-stant	Index Modification	Indirect Designation, Pointer P
ret									✓	✓	✓		✓	✓		Yes	Yes
n1									✓	✓	✓		✓	✓	✓	Yes	Yes
n2									✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.15, "Restrictions on Devices Used as Instruction Parameters" of "Sequence CPU – Instructions User's Manual" (IM34M6P12-03E).

Resource Relay

Table C3.5.91 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
✓	M1026	No Unused File ID	Execute instruction only if both "No Unused File ID" and "File/Disk Operation Group Busy" relays are OFF.
✓	M1025	File/Disk Operation Group Busy	

Function

Copies one or more files.

Multiple files can be specified using wildcard characters.

If a wildcard character is used to copy multiple files and an error is detected during copying, execution terminates with error.

If the specified source pathname (s) is a directory, the effect is equivalent to specifying a wildcard character ('*') to include all files stored immediately below the directory.

If a file having the same name or a file having the same name and a read-only file attribute already exists at the specified destination pathname, whether the file will be overwritten depends on the specified overwrite option.

Programming Example

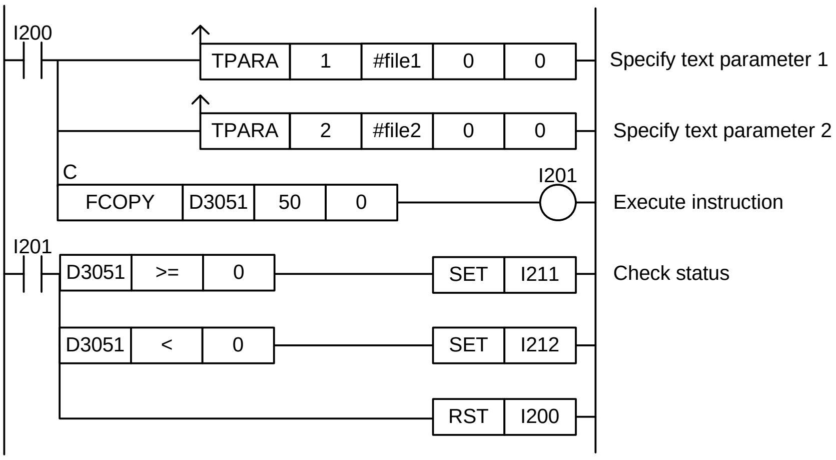


Figure C3.5.27 Example of a Copy File Program

This sample code copies files from the pathname defined by constant name #file1 to the pathname defined by constant name #file2.

It specifies timeout interval as 50 (5 s) and overwrite option as 0 (Exit with error without overwriting).

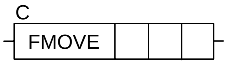
The table below shows an example of the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret=D3051	0	Status

C3.5.4.2 Move File (FMOVE)

Moves one or more files.

Table C3.5.92 Move File

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro-cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	Move File	FMOVE		✓	–	6	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see "■ Continuous Type Application Instructions" in Section C3.5.1, "Using File System Instructions".

■ Parameter

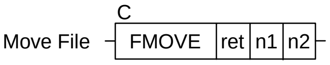


Table C3.5.93 Parameter

Parameter	Description
ret[*1]	Device for storing return status (W)
n1	Timeout interval (W) [0=indefinite, 1-32767 (x 100 ms)]
n2	Overwrite option (W) [0 = Exit with error without overwriting 1 = Overwrite file of the same name 2 = Overwrite read-only file of the same name]

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

Table C3.5.94 Text Parameter

Parameter		Description
1	s	Source pathname
2	d	Destination pathname

SEE ALSO

Specify text parameters using the Text Parameter (TPARA) instruction. For details on text parameters and the Text Parameter (TPARA) instruction, see "■ Text Parameter" in Section C3.5.1, "Using File System Instructions".

■ Status (Return Value)

Table C3.5.95 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see "● Error Status of Continuous Type Application Instructions" of "■ Continuous Type Application Instructions" in Section C3.5.1, "Using File System Instructions".

■ Available Devices

Table C3.5.96 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	W	Z	R	V	Con-stant	Index Modification	Indirect Designation, Pointer P
ret									✓	✓	✓		✓	✓		Yes	Yes
n1									✓	✓	✓		✓	✓	✓	Yes	Yes
n2									✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.15, "Restrictions on Devices Used as Instruction Parameters" of "Sequence CPU – Instructions User's Manual" (IM34M6P12-03E).

■ Resource Relay

Table C3.5.97 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
✓	M1026	No Unused File ID	Execute instruction only if both "No Unused File ID" and "File/Disk Operation Group Busy" relays are OFF.
✓	M1025	File/Disk Operation Group Busy	

■ Function

Moves one or more files.

Multiple files can be specified using wildcard characters.

If a wildcard character is used to move multiple files and an error is detected during moving, execution terminates with error.

If the specified source pathname (s) is a directory, the effect is equivalent to specifying a wildcard character ('*') to include all files stored immediately below the directory.

If a file having the same name or a read-only file having the same name already exists at the specified destination pathname, whether the file will be overwritten depends on the specified overwrite option.

■ Programming Example

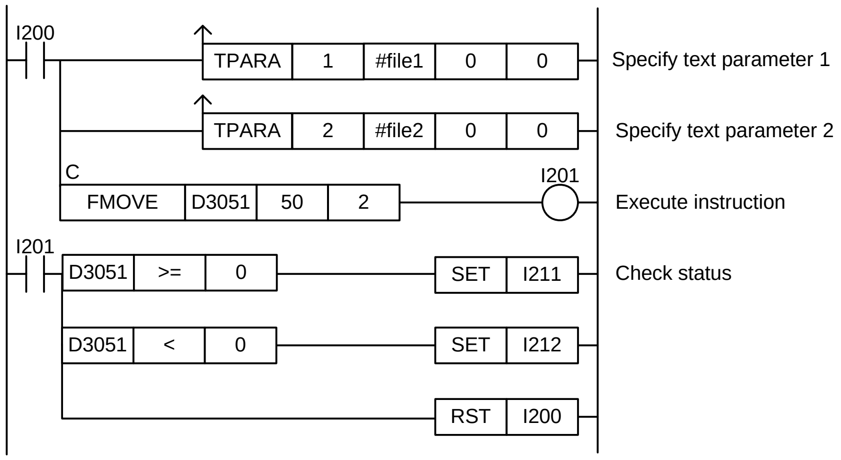


Figure C3.5.28 Example of a Move File Program

This sample code moves copies files from the pathname defined by constant name #file1 to the pathname defined by constant name #file2.

It specifies timeout interval as 50 (5s) and overwrite option as 2 (Overwrite read-only file of the same name).

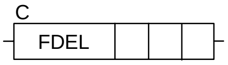
The table below shows an example of the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret=D3051	0	Status

C3.5.4.3 Delete File (FDEL)

Deletes one or more files.

Table C3.5.98 Delete File

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro-cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	Delete File	FDEL		✓	–	6	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see "■ Continuous Type Application Instructions" in Section C3.5.1, "Using File System Instructions".

■ Parameter

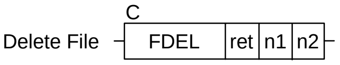


Table C3.5.99 Parameter

Parameter	Description
ret[*1]	Device for storing return status (W)
n1	Timeout interval (W) [0=indefinite, 1-32767 (x 100 ms)]
n2	Forced delete option (W) [0 = Exit with error without deleting read-only files 1 = Delete read-only files]

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

Table C3.5.100 Text Parameter

Parameter		Description
1	d	Target pathname

SEE ALSO

Specify text parameters using the Text Parameter (TPARA) instruction. For details on text parameters and the Text Parameter (TPARA) instruction, see "■ Text Parameter" in Section C3.5.1, "Using File System Instructions".

■ Status (Return Value)

Table C3.5.101 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see "● Error Status of Continuous Type Application Instructions" of "■ Continuous Type Application Instructions" in Section C3.5.1, "Using File System Instructions".

■ Available Devices

Table C3.5.102 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	W	Z	R	V	Con-stant	Index Modification	Indirect Designation, Pointer P
ret									✓	✓	✓		✓	✓		Yes	Yes
n1									✓	✓	✓		✓	✓	✓	Yes	Yes
n2									✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.15, "Restrictions on Devices Used as Instruction Parameters" of "Sequence CPU – Instructions User's Manual" (IM34M6P12-03E).

■ Resource Relay

Table C3.5.103 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
	M1026	No Unused File ID	Execute instruction only if "File/Disk Operation Group Busy" is OFF.
✓	M1025	File/Disk Operation Group Busy	

■ Function

Deletes one or more files.

Multiple files can be specified using wildcard characters.

If a wildcard character is used to delete multiple files and an error is detected during deletion, execution terminates with error.

If the specified target pathname (d) is a directory, the effect is equivalent to specifying a wildcard character ('*') to include all files stored immediately below the directory.

You can specify whether to delete files with read-only file attribute using the forced delete option.

CAUTION

The wildcard character can be used to delete many files in one go. Beware of inadvertently deleting required files when using the wildcard character.

■ Programming Example

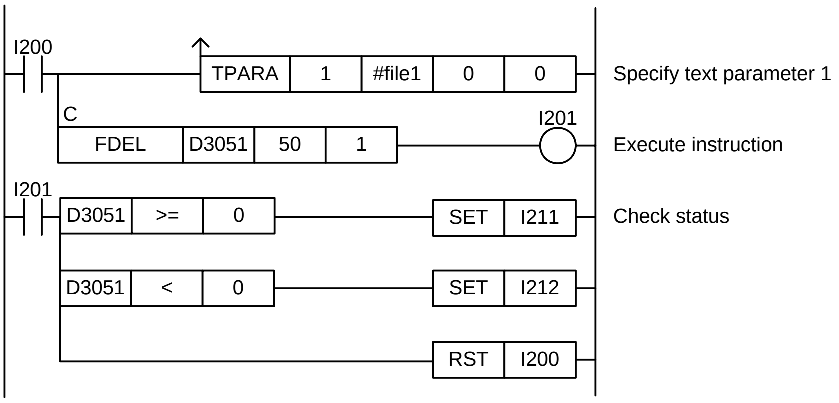


Figure C3.5.29 Example of a Delete File Program

This sample code deletes the file designated by the pathname defined by constant name #file1.

It specifies timeout interval as 50 (5 s) and forced delete option as 1 (Delete read-only files).

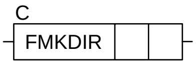
The table below shows an example of the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret=D3051	0	Status

C3.5.4.4 Make Directory (FMKDIR)

Creates a directory.

Table C3.5.104 Make Directory

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro-cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	Make Directory	FMKDIR		✓	–	5	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see "■ Continuous Type Application Instructions" in Section C3.5.1, "Using File System Instructions".

■ Parameter

Make Directory

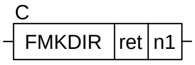


Table C3.5.105 Parameter

Parameter	Description
ret[*1]	Device for storing return status (W)
n1	Timeout interval (W) [0=indefinite, 1-32767 (x 100 ms)]

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

Table C3.5.106 Text Parameter

Parameter		Description
1	d	Pathname of directory to be created

SEE ALSO

Specify text parameters using the Text Parameter (TPARA) instruction. For details on text parameters and the Text Parameter (TPARA) instruction, see "■ Text Parameter" in Section C3.5.1, "Using File System Instructions".

■ Status (Return Value)

Table C3.5.107 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see "● Error Status of Continuous Type Application Instructions" of "■ Continuous Type Application Instructions" in Section C3.5.1, "Using File System Instructions".

■ Available Devices

Table C3.5.108 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	W	Z	R	V	Con-stant	Index Modification	Indirect Designation, Pointer P
ret									✓	✓	✓		✓	✓		Yes	Yes
n1									✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.15, "Restrictions on Devices Used as Instruction Parameters" of "Sequence CPU – Instructions User's Manual" (IM34M6P12-03E).

■ Resource Relay

Table C3.5.109 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
	M1026	No Unused File ID	Execute instruction only if "File/Disk Operation Group Busy" is OFF.
✓	M1025	File/Disk Operation Group Busy	

■ Function

Creates a directory.

■ Programming Example

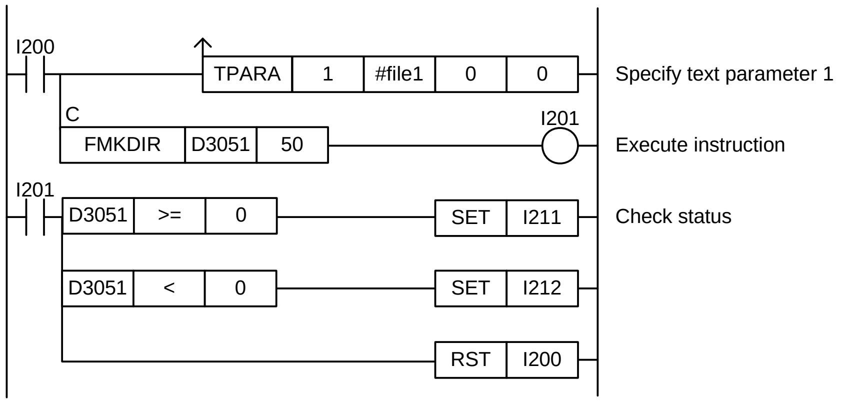


Figure C3.5.30 Example of a Make Directory Program

This sample code creates a directory designated by the directory pathname defined by constant name #file1. It specifies timeout interval as 50 (5 s).

The table below shows an example of the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret=D3051	0	Status

C3.5.4.5 Remove Directory (FRMDIR)

Deletes a directory.

Table C3.5.110 Remove Directory

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro-cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	Remove Directory	FRMDIR	C –FRMDIR | | | –|	✓	–	6	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see "■ Continuous Type Application Instructions" in Section C3.5.1, "Using File System Instructions".

■ Parameter

C

Remove Directory –FRMDIR | ret | n1 | n2–

Table C3.5.111 Parameter

Parameter	Description
ret*1	Device for storing return status (W)
n1	Timeout interval (W) [0=indefinite, 1-32767 (x 100 ms)]
n2	Delete all option (W) [0 = No 1 = Yes]

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

Table C3.5.112 Text Parameter

Parameter		Description
1	d	Pathname of directory to be deleted

SEE ALSO

Specify text parameters using the Text Parameter (TPARA) instruction. For details on text parameters and the Text Parameter (TPARA) instruction, see "■ Text Parameter" in Section C3.5.1, "Using File System Instructions".

■ Status (Return Value)

Table C3.5.113 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see "● Error Status of Continuous Type Application Instructions" of "■ Continuous Type Application Instructions" in Section C3.5.1, "Using File System Instructions".

Available Devices

Table C3.5.114 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	W	Z	R	V	Con-stant	Index Modification	Indirect Designation, Pointer P
ret									✓	✓	✓		✓	✓		Yes	Yes
n1									✓	✓	✓		✓	✓	✓	Yes	Yes
n2									✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.15, "Restrictions on Devices Used as Instruction Parameters" of "Sequence CPU – Instructions User's Manual" (IM34M6P12-03E).

Resource Relays

Table C3.5.115 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
	M1026	No Unused File ID	Execute instruction only if "File/Disk Operation Group Busy" is OFF.
✓	M1025	File/Disk Operation Group Busy	

Function

Deletes a directory.

If the directory to be deleted is not empty, whether the files and subdirectories contained in the directory are deleted depends on the specified "Delete all" option. If the specified "Delete all" option is 1, all contents of the directory are deleted. If a read-only file is encountered during deletion, execution terminates with error. If the specified "Delete all option" is 0 and the directory to be deleted is not empty, execution terminates with error without deleting the directory.

Programming Example

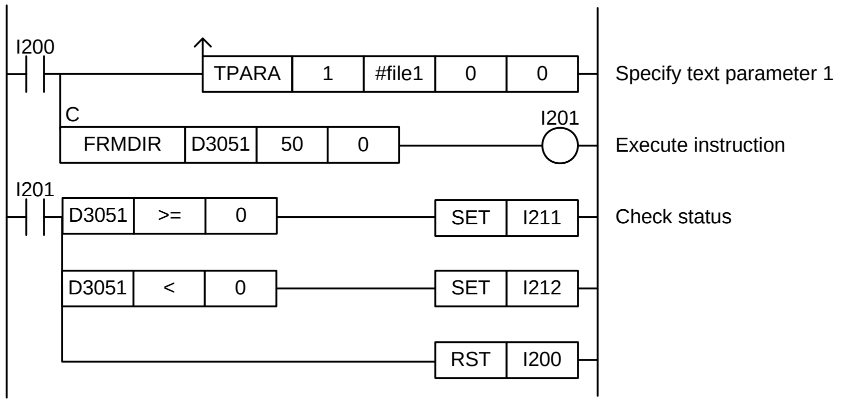


Figure C3.5.31 Remove Directory Programming Example

This sample code deletes the directory designated by the directory pathname defined by constant name #file1. It specifies timeout interval as 50 (5 s).

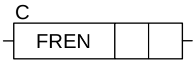
The table below shows an example of the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret=D3051	0	Status

C3.5.4.6 Rename File (FREN)

Renames a file or directory.

Table C3.5.116 Rename File

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Processing Unit	Carry
					Yes	No			
Continuous type application instruction	–	Rename File	FREN		✓	–	5	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see "■ Continuous Type Application Instructions" in Section C3.5.1, "Using File System Instructions".

■ Parameter

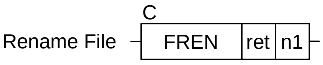


Table C3.5.117 Parameter

Parameter	Description
ret[*1]	Device for storing return status (W)
n1	Timeout interval (W) [0=indefinite, 1-32767 (x 100 ms)]

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

Table C3.5.118 Text Parameter

Parameter		Description
1	s	Old file pathname or directory pathname
2	d	New file name or directory name

SEE ALSO

Specify text parameters using the Text Parameter (TPARA) instruction. For details on text parameters and the Text Parameter (TPARA) instruction, see "■ Text Parameter" in Section C3.5.1, "Using File System Instructions".

■ Status (Return Value)

Table C3.5.119 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see "● Error Status of Continuous Type Application Instructions" of "■ Continuous Type Application Instructions" in Section C3.5.1, "Using File System Instructions".

■ Available Devices

Table C3.5.120 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	W	Z	R	V	Con-stant	Index Modification	Indirect Designation, Pointer P
ret									✓	✓	✓		✓	✓		Yes	Yes
n1									✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.15, "Restrictions on Devices Used as Instruction Parameters" of "Sequence CPU – Instructions User's Manual" (IM34M6P12-03E).

■ Resource Relays

Table C3.5.121 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
	M1026	No Unused File ID	Execute instruction only if "File/Disk Operation Group Busy" is OFF.
✓	M1025	File/Disk Operation Group Busy	

■ Function

Renames a file or directory.

For the "New filename" parameter, you should specify only a filename or directory name. If you specify a file pathname including a directory for the parameter, the instruction returns an Invalid Pathname error (status code -12200).

■ Programming Example

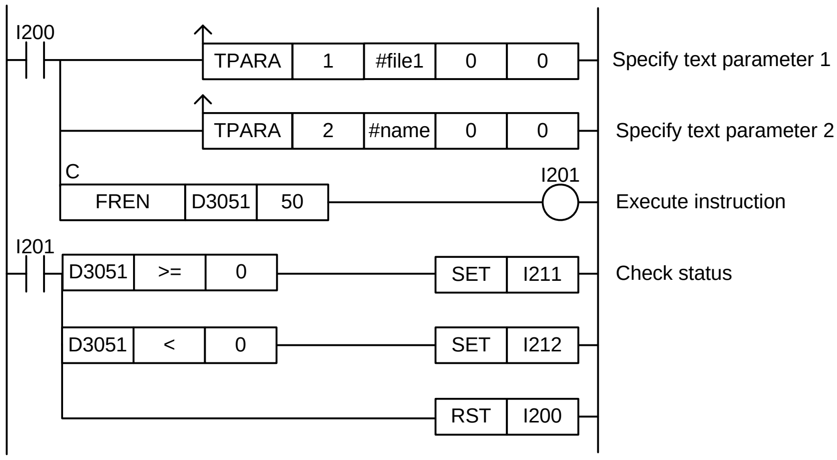


Figure C3.5.32 Example of a Rename File Program

This sample code renames the file designated by the pathname defined by constant name #file1 to the filename defined by constant name #name. It specifies timeout interval as 50 (5 s).

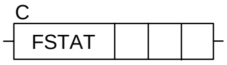
The table below shows an example of the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret=D3051	0	Status

C3.5.4.7 File Status (FSTAT)

Returns file status information for a specified file or specified directory.

Table C3.5.122 File Status

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro-cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	File Status	FSTAT		✓	–	6	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see "■ Continuous Type Application Instructions" in Section C3.5.1, "Using File System Instructions".

■ Parameter

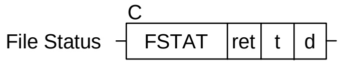


Table C3.5.123 Parameter

Parameter		Description
ret[*1]		Device for storing return status (W)
t	t+0	Timeout interval (W) [0=indefinite, 1-32767 (x 100 ms)]
	t+1	Output option (W) [0 = File presence only Non-zero = All information]
d		File status output device (W)

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

Table C3.5.124 Text Parameter

Parameter		Description
1	s	Target file/directory pathname

SEE ALSO

Specify text parameters using the Text Parameter (TPARA) instruction. For details on text parameters and the Text Parameter (TPARA) instruction, see "■ Text Parameter" in Section C3.5.1, "Using File System Instructions".

■ Status (Return Value)

Table C3.5.125 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit (specified file or directory exists)
		< 0	Error status

SEE ALSO

For more details on error status, see "● Error Status of Continuous Type Application Instructions" of "■ Continuous Type Application Instructions" in Section C3.5.1, "Using File System Instructions".

■ Available Devices

Table C3.5.126 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	W	Z	R	V	Con-stant	Index Modification	Indirect Designation, Pointer P
ret									✓	✓	✓		✓	✓		Yes	Yes
t									✓	✓	✓		✓	✓		Yes	Yes
d									✓	✓	✓		✓	✓		Yes	Yes

Note: See Section 1.15, "Restrictions on Devices Used as Instruction Parameters" of "Sequence CPU – Instructions User's Manual" (IM34M6P12-03E).

■ Resource Relays

Table C3.5.127 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
	M1026	No Unused File ID	Execute instruction only if "File/Disk Operation Group Busy" is OFF.
✓	M1025	File/Disk Operation Group Busy	

■ Function

Returns file status information for a specified file or specified directory.

File status information that can be returned include whether file/directory exists, file/directory attribute, file size and last modified time.

Output Option:
Output option controls the file status information to be returned.

If 0 (=file presence only) is specified, nothing is stored to the file status output device. However, whether the file/directory is present is indicated by the returned status (ret).

If 1 (=all information) is specified, the file attribute, file size and last modified time are stored to the specified file status output device.

Table C3.5.128 Information Stored to File Status Output Device

Appended Information		Offset (word)	Size (word)	Description
File attribute	-	+0	1	*1
File size	Low word	+1	2	The file size in bytes is stored. For a directory, 0 is stored.
	High word	+2		
Last modified time*2	Year	+3	1	Data is stored in BCD representation. Example: 1999 as $0099, 2000 as $0000
	Month	+4	1	Data is stored in BCD representation. Example: January as $0001
	Date	+5	1	Data is stored in BCD representation. Example: 28th as $0028
	Hour	+6	1	Data is stored in BCD representation. Example: 18:00 hours as $0018
	Minute	+7	1	Data is stored in BCD representation. Example: 15 minutes as $0015
	Second	+8	1	Data is stored in BCD representation. Example: 30 seconds as $0030

Note: Nothing is stored to File Status Output Device if the specified output option is 0 (=File presence only).
*1: For details on the format of the stored file attribute data, see the following table.
*2: The stored information has the same format as the date/time special registers (Z049-Z054).

Table C3.5.129 File (Directory) Attribute Data

Bit Position	Description
0 (LSB)	0 =Read and write 1=Read-only
1	0=Non-system file 1=System file
2	0=Non-hidden file 1=Hidden file
3	Reserved
4	Reserved
5	Reserved
6	Reserved
7	Reserved
8	Reserved
9	Reserved
10	Reserved
11	Reserved
12	Reserved
13	Reserved
14	Reserved
15 (MSB)	0 =File 1 =Directory

■ Programming Example

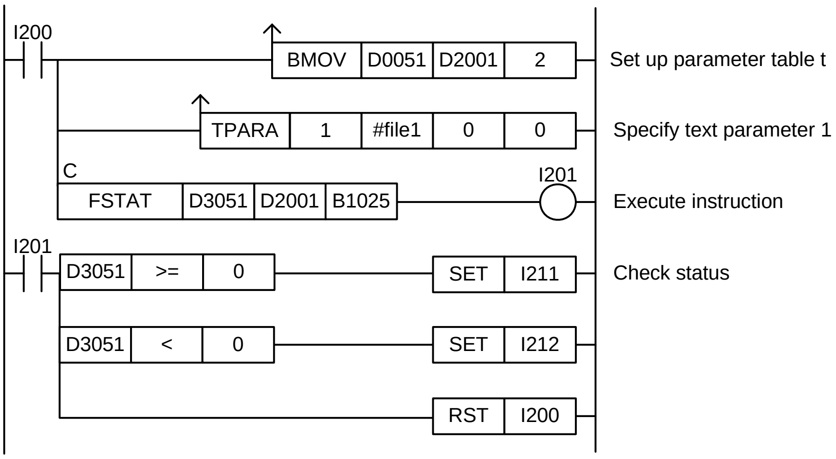


Figure C3.5.33 Example of a File Status Program

This sample code stores status information for the file designated by the pathname defined by constant name #file1 to the device, starting from B1025. It specifies ret(=D3051) and t(=D2001), with t set up as shown in the table below.

Device	Value	Table Parameter
t=D2001	50	Timeout interval (= 5 s)
D2002	1	Output option (=All information)

The table below shows the content of ret, assuming normal exit.

Device	Value	Table Parameter
ret=D3051	0	Status

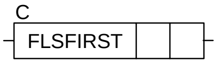
The table below shows the content of d, assuming normal exit.

Device	Value	Table Parameter
B1025	$0000	File attribute
B1026	4620	File size
B1027		
B1028	$0005	Year
B1029	$0012	Month
B1030	$0031	Date
B1031	$0011	Hour
B1032	$0003	Minute
B1033	$0050	Second

C3.5.4.8 File List Start (FLSFIRST)

Declares a file list operation for getting status information of successive files or directories.

Table C3.5.130 File List Start

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro-cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	File List Start	FLSFIRST		✓	–	5	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see "■ Continuous Type Application Instructions" in Section C3.5.1, "Using File System Instructions".

■ Parameter

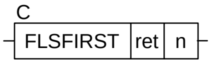
File List Start 

Table C3.5.131 Parameter

Parameter	Description
ret[*1]	Device for storing return status (W)
n	Timeout interval (W) [0=indefinite, 1-32767 (x 100 ms)]

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

Table C3.5.132 Text Parameter

Parameter		Description
1	s	Target pathname (including use of wildcard)

SEE ALSO

Specify text parameters using the Text Parameter (TPARA) instruction. For details on text parameters and the Text Parameter (TPARA) instruction, see "■ Text Parameter" in Section C3.5.1, "Using File System Instructions".

■ Status (Return Value)

Table C3.5.133 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see "● Error Status of Continuous Type Application Instructions" of "■ Continuous Type Application Instructions" in Section C3.5.1, "Using File System Instructions".

Available Devices

Table C3.5.134 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	W	Z	R	V	Con-stant	Index Modification	Indirect Designation, Pointer P
ret									✓	✓	✓		✓	✓		Yes	Yes
n									✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.15, "Restrictions on Devices Used as Instruction Parameters" of "Sequence CPU – Instructions User’s Manual" (IM34M6P12-03E).

Resource Relays

Table C3.5.135 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
	M1026	No Unused File ID	Execute instruction only if "File/Disk Operation Group Busy" is OFF.
✓	M1025	File/Disk Operation Group Busy	

Function

TIP

FLSFIRST (File List Start), FLS (File List Next) and FLSFIN (File List End) instructions are used as a group in a file list operation.

Declares a file list operation for getting status information of successive files.

After this declaration, the File List Next (FLS) instruction can be executed repeatedly to get status information for multiple files designated by the use of a wildcard character within the specified directory, one at a time. If the specified source pathname ("s" parameter) is a directory, the effect is equivalent to specifying a wildcard character ('*') to include all files stored immediately below the directory.

The reference pathname declared by this instruction for the list operation remains in force until the File List End (FLSFIN) instruction is executed. Therefore, if the File List End (FLSFIN) instruction is not executed, no new reference pathname can be specified using a File List Start (FLSFIRST) instruction.

The reference pathname can be specified using the wildcard character.

Example: To target all files and directories on the RAM disk, specify:

```
\RAMDISK\*
```

Example: To target all files with file extension ".txt" on the RAM disk, specify:

```
\RAMDISK\*.txt
```

Example: To target all files with filename "abc" and any file extension on the RAM disk, specify:

```
\RAMDISK\abc.*
```

If you execute this instruction while a reference pathname declared by an earlier execution of this instruction is in force, an error will be generated. To specify a new reference pathname for file list operation, you must first execute the File List End (FLSFIN) instruction.

SEE ALSO

For details on how to get file status information successively, see Subsection C3.5.4.9, "File List Next (FLS)".

■ Programming Example

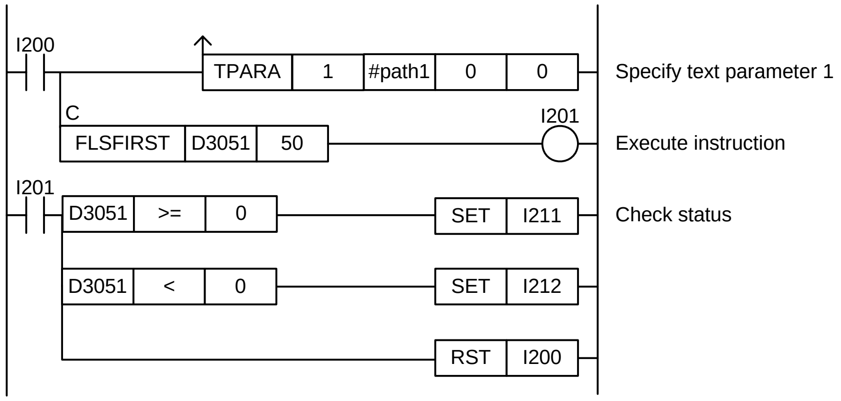


Figure C3.5.34 Example of a File List Start Program

This sample code declares a file list operation for the directory defined by constant name #path1. It specifies timeout interval as 50 (5 s).

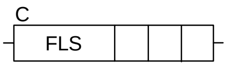
The table below shows an example of the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret=D3051	0	Status

C3.5.4.9 File List Next (FLS)

Gets status information of the next file in a file list operation pre-declared with a reference pathname.

Table C3.5.136 File List Next

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro-cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	File List Next	FLS		✓	–	6	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see "■ Continuous Type Application Instructions" in Section C3.5.1, "Using File System Instructions".

■ Parameter

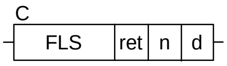
File List Next 

Table C3.5.137 Parameter

Parameter	Description
ret[*1]	Device for storing return status (W)
n	Timeout interval (W) [0=indefinite, 1-32767 (x 100 ms)]
d	File Status Output Device (W)

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

■ Status (Return Value)

Table C3.5.138 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see "● Error Status of Continuous Type Application Instructions" of "■ Continuous Type Application Instructions" in Section C3.5.1, "Using File System Instructions".

■ Available Devices

Table C3.5.139 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	W	Z	R	V	Con-stant	Index Modification	Indirect Designation, Pointer P
ret									✓	✓	✓		✓	✓		Yes	Yes
n									✓	✓	✓		✓	✓	✓	Yes	Yes
d									✓	✓	✓		✓	✓		Yes	Yes

Note: See Section 1.15, "Restrictions on Devices Used as Instruction Parameters" of "Sequence CPU – Instructions User's Manual" (IM34M6P12-03E).

■ Resource Relays

Table C3.5.140 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
	M1026	No Unused File ID	Execute instruction only if "File/Disk Operation Group Busy" is OFF.
✓	M1025	File/Disk Operation Group Busy	

■ Function

TIP

FLSFIRST (File List Start), FLS (File List Next) and FLSFIN (File List End) instructions are used as a group in a file list operation.

Gets status information of the next file or directory in a file list operation pre-declared with a reference pathname.

When the instruction encounters the last file matching the wildcard character in the reference pathname, it returns a status of "No match found" (status code -2001).

The table below shows the file status information stored to the specified file status output device by the instruction.

Table C3.5.141 Information Stored to File Status Output Device

Appended Information		Offset (word)	Size (word)	Description
File attribute	–	+0	1	*1
File size	Low word	+1	2	The file size in bytes is stored. For a directory, 0 is stored.
	High word	+2		
Last modified time*2	Year	+3	1	Data is stored in BCD representation. Example: 1999 as $0099, 2000 as $0000
	Month	+4	1	Data is stored in BCD representation. Example: January as $0001
	Date	+5	1	Data is stored in BCD representation. Example: 28th as $0028
	Hour	+6	1	Data is stored in BCD representation. Example: 18:00 hours as $0018
	Minute	+7	1	Data is stored in BCD representation. Example: 15 minutes as $0015
	Second	+8	1	Data is stored in BCD representation. Example: 30 seconds as $0030
Filename	–	+9 to +136	128	The filename is stored with a variable length.

*1: For details on the format of the stored file attribute data, see the following table.
*2: The stored information has the same format as the date/time special registers (Z049-Z054).

Table C3.5.142 File (Directory) Attribute Data

Bit Position	Description
0 (LSB)	0 =Read and write 1=Read-only
1	0=Non-system file 1=System file
2	0=Non-hidden file 1=Hidden file
3	Reserved
4	Reserved
5	Reserved
6	Reserved
7	Reserved
8	Reserved
9	Reserved
10	Reserved
11	Reserved
12	Reserved
13	Reserved
14	Reserved
15 (MSB)	0 =File 1 =Directory

How to Perform a File List Operation:

1. Declare a file list operation by executing a File List Start (FLSFIRST) instruction, specifying a target pathname including a wildcard character.
2. Execute the File List Next (FLS) instruction to get the status information of the next file. The order of the retrieval will be according to the order of files on the file system.
3. Repeat step 2 until "No match found" (-2001) is returned in the instruction status.
4. Execute the File List End (FLSFIN) instruction to end the declared file list operation.

■ Programming Example

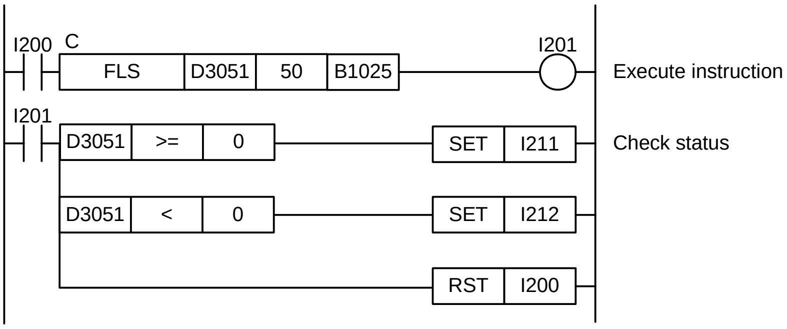


Figure C3.5.35 Example of a File List Next Program

This sample code stores status information for files within the directory previously declared by a File List Start (FLSFIRST) instruction to device, starting from B1025. It specifies timeout interval as 50 (5 s).

The table below shows an example of the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret=D3051	0	Status

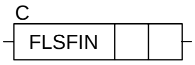
The table below shows an example of the data stored to file status output device d, assuming normal exit.

Device	Value	Table Parameter
B1025	$0000	File attribute
B1026	4620	File size
B1027		
B1028	$0005	Year
B1029	$0012	Month
B1030	$0031	Date
B1031	$0011	Hour
B1032	$0003	Minute
B1033	$0050	Second
B1034	“AB”	Filename
B1035	“C.”	
B1036	“CS”	
B1037	“V”+NULL	

C3.5.4.10 File List End (FLSFIN)

Declares the end of a file list operation.

Table C3.5.143 File List End

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro-cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	File List End	FLSFIN		✓	–	5	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see "■ Continuous Type Application Instructions" in Section C3.5.1, "Using File System Instructions".

■ Parameter

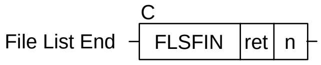


Table C3.5.144 Parameter

Parameter	Description
ret[*1]	Device for storing return status (W)
n	Timeout interval (W) [0=indefinite, 1-32767 (x 100 ms)]

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

■ Status (Return Value)

Table C3.5.145 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see "● Error Status of Continuous Type Application Instructions" of "■ Continuous Type Application Instructions" in Section C3.5.1, "Using File System Instructions".

■ Available Devices

Table C3.5.146 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	W	Z	R	V	Con-stant	Index Modification	Indirect Designation, Pointer P
ret									✓	✓	✓		✓	✓		Yes	Yes
n									✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.15, "Restrictions on Devices Used as Instruction Parameters" of "Sequence CPU – Instructions User's Manual" (IM34M6P12-03E).

■ Resource Relays

Table C3.5.147 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
	M1026	No Unused File ID	Execute instruction only if "File/Disk Operation Group Busy" is OFF.
✓	M1025	File/Disk Operation Group Busy	

■ Function

TIP

FLSFIRST (File List Start), FLS (File List Next) and FLSFIN (File List End) instructions are used as a group in a file list operation.

Declares the end of a file list operation.

After instruction execution, the reference pathname specified in the File List Start (FLSFIRST) instruction executed previously no longer holds and a new reference pathname can be specified by executing another File List Start (FLSFIRST) instruction.

■ Programming Example

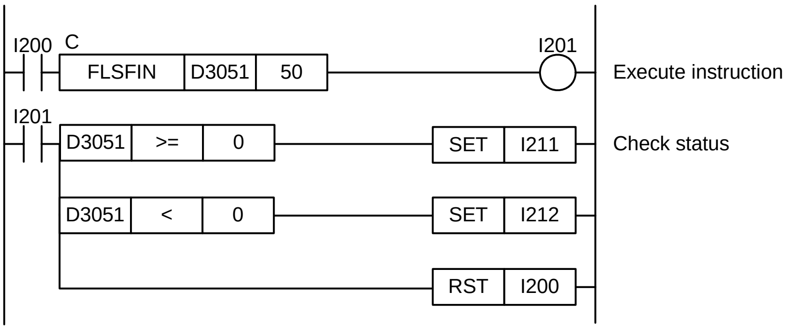


Figure C3.5.36 Example of a File List End Program

This sample code ends a file list operation. It specifies timeout interval as 50 (5 s).

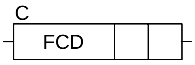
The table below shows an example of the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret=D3051	0	Status

C3.5.4.11 Change Directory (FCD)

Changes the current directory.

Table C3.5.148 Change Directory

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro-cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	Change Directory	FCD		✓	–	5	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see "■ Continuous Type Application Instructions" in Section C3.5.1, "Using File System Instructions".

■ Parameter

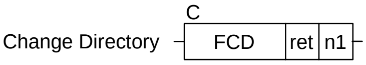


Table C3.5.149 Parameter

Parameter	Description
ret[*1]	Device for storing return status (W)
n1	Timeout interval (W) [0=indefinite, 1-32767 (x 100 ms)]

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

Table C3.5.150 Text Parameter

Parameter		Description
1	n2	New current directory pathname

SEE ALSO

Specify text parameters using the Text Parameter (TPARA) instruction. For details on text parameters and the Text Parameter (TPARA) instruction, see "■ Text Parameter" in Section C3.5.1, "Using File System Instructions".

■ Status (Return Value)

Table C3.5.151 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see "● Error Status of Continuous Type Application Instructions" of "■ Continuous Type Application Instructions" in Section C3.5.1, "Using File System Instructions".

Available Devices

Table C3.5.152 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	W	Z	R	V	Con-stant	Index Modification	Indirect Designation, Pointer P
ret									✓	✓	✓		✓	✓		Yes	Yes
n1									✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.15, "Restrictions on Devices Used as Instruction Parameters" of "Sequence CPU – Instructions User's Manual" (IM34M6P12-03E).

Resource Relays

Table C3.5.153 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
	M1026	No Unused File ID	Execute instruction only if "File/Disk Operation Group Busy" is OFF.
✓	M1025	File/Disk Operation Group Busy	

Function

Changes the current directory.

This instruction changes the current directory, which is common to file access instructions and file operation instructions, but unrelated to the current directory of the FTP client.

The default initial current directory is "\RAMDISK".

A parameter error is generated if the destination directory pathname is omitted.

Programming Example

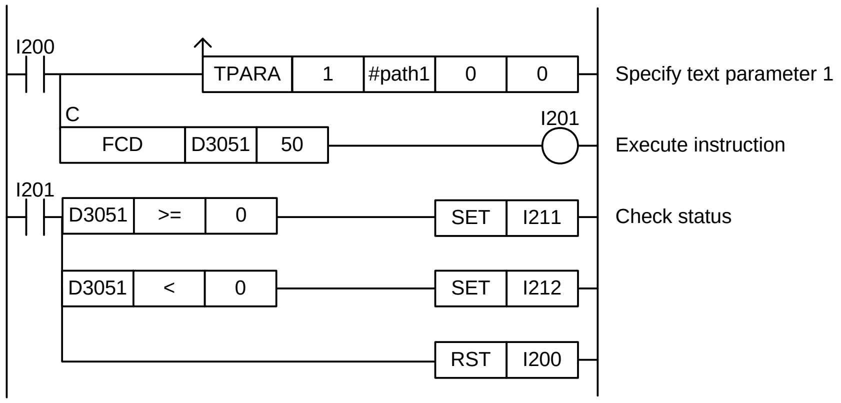


Figure C3.5.37 Example of a Change Directory Program

This sample code changes the current directory to the directory defined by constant name #path1. It specifies timeout interval as 50 (5 s).

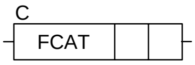
The table below shows an example of the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret=D3051	0	Status

C3.5.4.12 Concatenate File (FCAT)

Concatenates two files.

Table C3.5.154 Concatenate File

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro-cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	Concatenate File	FCAT		✓	–	5	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see "■ Continuous Type Application Instructions" in Section C3.5.1, "Using File System Instructions".

■ Parameter

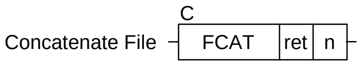


Table C3.5.155 Parameter

Parameter	Description
ret*1	Device for storing return status (W)
n	Timeout interval (W) [0=indefinite, 1-32767 (x 100 ms)]

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

Table C3.5.156 Text Parameter

Parameter		Description
1	s1	File pathname 1 (source file 1 and destination)
2	s2	File pathname 2 (source file 2)

SEE ALSO

Specify text parameters using the Text Parameter (TPARA) instruction. For details on text parameters and the Text Parameter (TPARA) instruction, see "■ Text Parameter" in Section C3.5.1, "Using File System Instructions".

■ Status (Return Value)

Table C3.5.157 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see "● Error Status of Continuous Type Application Instructions" of "■ Continuous Type Application Instructions" in Section C3.5.1, "Using File System Instructions".

Available Devices

Table C3.5.158 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	W	Z	R	V	Con- stant	Index Modification	Indirect Designation, Pointer P
ret									✓	✓	✓		✓	✓		Yes	Yes
n									✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.15, "Restrictions on Devices Used as Instruction Parameters" of "Sequence CPU – Instructions User's Manual" (IM34M6P12-03E).

Resource Relays

Table C3.5.159 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
✓	M1026	No Unused File ID	Execute instruction only if both "No Unused File ID" and "File/Disk Operation Group Busy" relays are OFF.
✓	M1025	File/Disk Operation Group Busy	

Function

Concatenates two files.

This instruction appends the file designated by file pathname 2 to the file designated by file pathname 1, and stores the result as a file designated by file pathname 1.

Wildcard characters may not be used with this command.

Programming Example

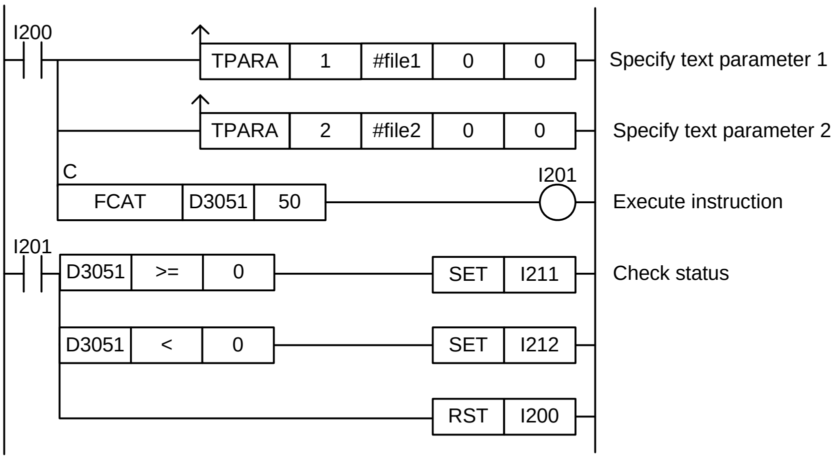


Figure C3.5.38 Example of a Concatenate File Program

This sample code appends the file designated by the file pathname defined by constant name #file2 to the file designated by the file pathname defined by constant name #file1. It specifies timeout interval as 50 (5 s).

The table below shows an example of the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret=D3051	0	Status

C3.5.4.13 Change File Attribute (FATRW)

Changes the attribute of a specified file or directory.

Table C3.5.160 Change File Attribute

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Processing Unit	Carry
					Yes	No			
Continuous type application instruction	–	Change File Attribute	FATRW	C FATRW	✓	–	5	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see "■ Continuous Type Application Instructions" in Section C3.5.1, "Using File System Instructions".

■ Parameter

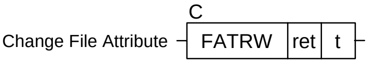


Table C3.5.161 Parameter

Parameter		Description
ret[*1]		Device for storing return status (W)
t	t+0	Timeout interval (W) [0=indefinite, 1-32767 (x 100 ms)]
	t+1	File attribute (W) [*2]
	t+2	File attribute mask (W) [bit mask] [*2] Specifies which bits of the file attribute are to be changed. [0=hold, 1=overwrite]

[*1] ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

[*2] For details on the bitmap of the file attribute, see Table C3.5.166, "File (Directory) Attribute Data".

Table C3.5.162 Text Parameter

Parameter		Description
1	d	Target file/directory pathname

SEE ALSO

Specify text parameters using the Text Parameter (TPARA) instruction. For details on text parameters and the Text Parameter (TPARA) instruction, see "■ Text Parameter" in Section C3.5.1, "Using File System Instructions".

■ Status (Return Value)

Table C3.5.163 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see "● Error Status of Continuous Type Application Instructions" of "■ Continuous Type Application Instructions" in Section C3.5.1, "Using File System Instructions".

■ Available Devices

Table C3.5.164 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	W	Z	R	V	Con-stant	Index Modification	Indirect Designation, Pointer P
ret									✓	✓	✓		✓	✓		Yes	Yes
t									✓	✓	✓		✓	✓		Yes	Yes

Note: See Section 1.15, "Restrictions on Devices Used as Instruction Parameters" of "Sequence CPU – Instructions User's Manual" (IM34M6P12-03E).

■ Resource Relays

Table C3.5.165 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
	M1026	No Unused File ID	Execute instruction only if "File/Disk Operation Group Busy" is OFF.
✓	M1025	File/Disk Operation Group Busy	

■ Function

Changes the attribute of a specified file or directory.

Specify the attribute value as one word in binary representation as shown in the table below.

Multiple files can be specified using wildcard characters. If a wildcard character is used to change the attribute of multiple files and an error is detected during execution, execution terminates with error.

File attributes can be changed even for files that are open in write mode.

The file attribute mask can be used to specify the attributes to be changed. A '1' in a bit position of the mask indicates to change the corresponding file attribute while a '0' in a bit position indicates to leave the corresponding file attribute unchanged.

Table C3.5.166 File (Directory) Attribute Data

Bit Position	Description
0 (LSB)	0 = Read and write 1 = Read-only
1	0 = Non-system file 1 = System file
2	0 = Non-hidden file 1 = Hidden file
3	Reserved
4	Reserved
5	Reserved
6	Reserved
7	Reserved
8	Reserved
9	Reserved
10	Reserved
11	Reserved
12	Reserved
13	Reserved
14	Reserved
15 (MSB)	0 = File 1 = Directory

■ Programming Example

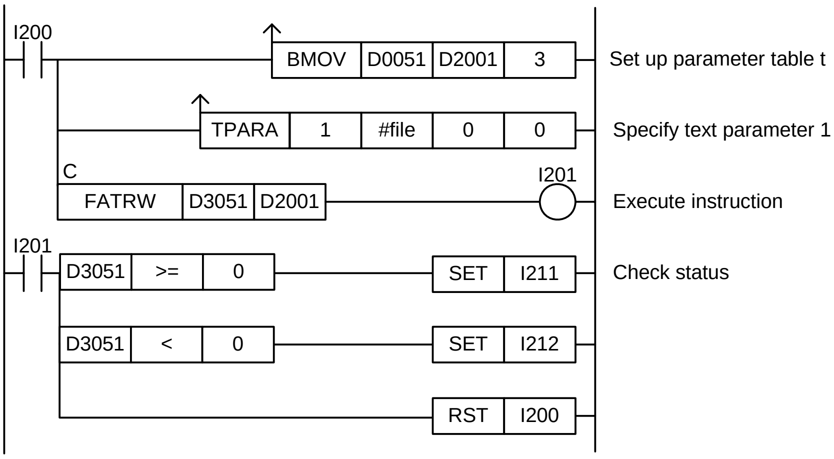


Figure C3.5.39 Change File Attribute Programming Example

This sample code sets the read-only attribute of all CSV formatted files in the directory designated by the directory pathname defined by constant name #file.

```
#file = "\RAMDISK\NEWDATA\*.CSV"
```

It specifies ret(=D3051) and t(=D2001), with t set up as shown in the table below.

Device	Value	Table Parameter
t=D2001	50	Timeout interval (= 5 s)
D2002	$0001	File attribute (= set read-only attribute)
D2003	$0001	File attribute mask (= change read-only attribute and nothing else)

The table below shows an example of the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret=D3051	0	Status

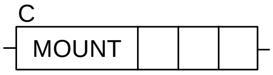
C3.5.5 Disk Operation Instructions

Of the file operation instruction group and disk operation instruction group, only instructions from one group can be executed at any one time. If you attempt to execute multiple instructions, the instruction that is executed later will be terminated with a repeat use of function error (error code -3001). Before executing a disk operation instruction, check to ensure that the File/Disk Operation Group Busy relay (M1025) is OFF.

C3.5.5.1 Mount Memory Card (MOUNT)

Mounts the memory card inserted in the card slot so that it is ready for use by programs and various services.

Table C3.5.167 Mount Memory Card

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro-cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	Mount Memory Card	MOUNT		✓	–	6	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see "■ Continuous Type Application Instructions" in Section C3.5.1, "Using File System Instructions".

■ Parameter

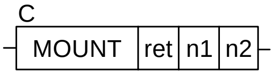
Mount Memory Card 

Table C3.5.168 Parameter

Parameter	Description
ret[*1]	Device for storing return status (W)
n1	Timeout interval (W) [0=indefinite, 1 to 32767 (x 100 ms)]
n2	Card slot number (W) [always 1]

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

■ Status (Return Value)

Table C3.5.169 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see "● Error Status of Continuous Type Application Instructions" of "■ Continuous Type Application Instructions" in Section C3.5.1, "Using File System Instructions".

Available Devices

Table C3.5.170 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	W	Z	R	V	Con-stant	Index Modification	Indirect Designation, Pointer P
ret									✓	✓	✓		✓	✓		Yes	Yes
n1									✓	✓	✓		✓	✓	✓	Yes	Yes
n2									✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.15, "Restrictions on Devices Used as Instruction Parameters" of "Sequence CPU – Instructions User's Manual" (IM34M6P12-03E).

Resource Relays

Table C3.5.171 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
✓	M1025	File/Disk Operation Group Busy	Execute instruction only if the "File/Disk Operation Group Busy" relay is OFF.

Function

Mounts the memory card inserted in the card slot so that it is ready for use by programs and various services. It is usually not necessary to execute this instruction as the CPU module automatically recognizes and mounts a memory card when it is inserted into the memory card slot. This instruction can be used however in situations where there is a need to mount and unmount a memory card which remains inserted in the memory card slot. A possible scenario would be to unmount the memory card in the day but to mount and access the memory card in the night.

If the memory card is successfully mounted with normal exit, the **SD** LED located on the front panel of the module lights up. Conversely, the **SD** LED is not lit if the memory card is unmounted.

Always unmount the memory card either by executing the Unmount Memory card (UNMOUNT) instruction or using the rotary switch before removing the memory card from the memory card slot.

The card slot number is fixed to 1.

CAUTION

- Inserting a memory card automatically mounts it without the need to execute this instruction.
- Do not remove a memory card without first unmounting it. Otherwise, data may be damaged or lost.
- Execution of this instruction is highly likely to complete even in the presence of a timeout or cancel event.

■ Programming Example

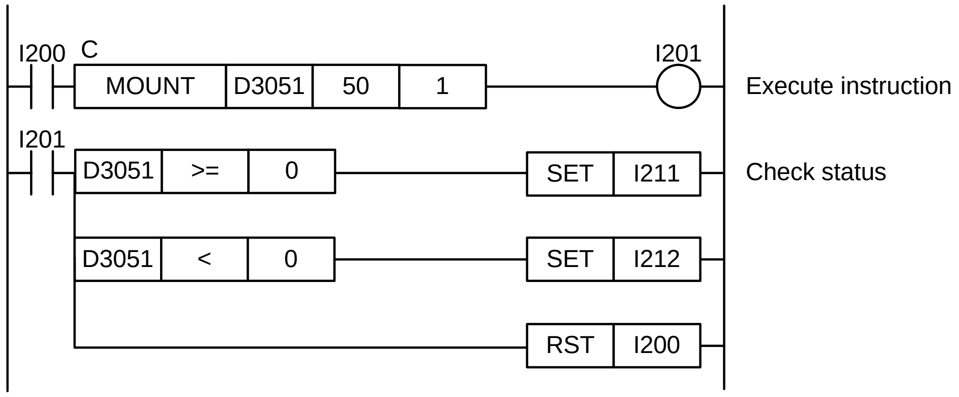


Figure C3.5.40 Example of a Mount Memory Card Program

This sample code mounts memory card CARD1. It specifies timeout interval as 50 (5 s).

The table below shows an example of the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret=D3051	0	Status

C3.5.5.2 Unmount Memory Card (UNMOUNT)

Unmounts the memory card, which is inserted and mounted in the card slot.

Table C3.5.172 Unmount Memory Card

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro-cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	Unmount Memory Card	UNMOUNT	C UNMOUNT	✓	–	6	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see "■ Continuous Type Application Instructions" in Section C3.5.1, "Using File System Instructions".

■ Parameter

C
Unmount Memory Card UNMOUNT ret n1 n2

Table C3.5.173 Parameter

Parameter	Description
ret*1	Device for storing return status (W)
n1	Timeout interval (W) [0=indefinite, 1 to 32767 (x 100 ms)]
n2	Card slot number (W) [always 1]

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

■ Status (Return Value)

Table C3.5.174 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see "● Error Status of Continuous Type Application Instructions" of "■ Continuous Type Application Instructions" in Section C3.5.1, "Using File System Instructions".

■ Available Devices

Table C3.5.175 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	W	Z	R	V	Con-stant	Index Modification	Indirect Designation, Pointer P
ret									✓	✓	✓		✓	✓		Yes	Yes
n1									✓	✓	✓		✓	✓	✓	Yes	Yes
n2									✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.15, "Restrictions on Devices Used as Instruction Parameters" of "Sequence CPU – Instructions User's Manual" (IM34M6P12-03E).

Resource Relays

Table C3.5.176 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
✓	M1025	File/Disk Operation Group Busy	Execute instruction only if the "File/Disk Operation Group Busy" relay is OFF.

Function

Unmounts the memory card, which is inserted and mounted in the card slot. A memory card in unmounted state can be safely removed, but does not allow access by programs or via FTP.

If the memory card is successfully unmounted with normal exit, the **SD** LED located on the front panel of the module turns off. Conversely, the **SD** LED is lit if the memory card is mounted.

The card slot number is fixed to 1.

CAUTION

- Do not remove a memory card without first unmounting it. Otherwise, data may be damaged or lost.
- Execution of this instruction is highly likely to complete even in the presence of a timeout or cancel event.

Programming Example

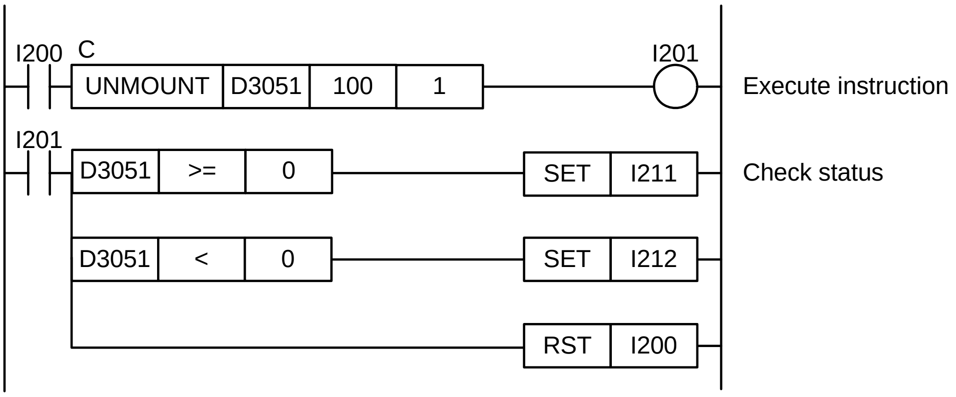


Figure C3.5.41 Example of an Unmount Memory Card Program

This sample code unmounts memory card CARD1. It specifies timeout interval as 100 (10 s).

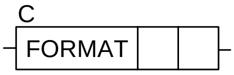
The table below shows an example of the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret=D3051	0	Status

C3.5.5.3 Format Disk (FORMAT)

Formats a specified disk in FAT16 format.

Table C3.5.177 Format Disk

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro-cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	Format Disk	FORMAT		✓	–	5	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see "■ Continuous Type Application Instructions" in Section C3.5.1, "Using File System Instructions".

■ Parameter

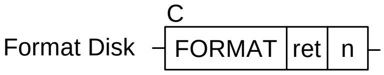


Table C3.5.178 Parameter

Parameter	Description
ret[*1]	Device for storing return status (W)
n	Disk selection (W) [1=\RAMDISK 2=\CARD1]

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

■ Status (Return Value)

Table C3.5.179 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see "● Error Status of Continuous Type Application Instructions" of "■ Continuous Type Application Instructions" in Section C3.5.1, "Using File System Instructions".

■ Available Devices

Table C3.5.180 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	W	Z	R	V	Con-stant	Index Modification	Indirect Designation, Pointer P
ret									✓	✓	✓		✓	✓		Yes	Yes
n									✓	✓	✓		✓	✓	✓	Yes	Yes

Note: See Section 1.15, "Restrictions on Devices Used as Instruction Parameters" of "Sequence CPU – Instructions User's Manual" (IM34M6P12-03E).

■ Resource Relays

Table C3.5.181 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
✓	M1025	File/Disk Operation Group Busy	Execute instruction only if the "File/Disk Operation Group Busy" relay is OFF.

■ Function

Formats a specified disk in FAT16 format.

If the disk to be formatted is in use, formatting cannot be done. Close all files before executing this instruction.

An SD memory card, which is not in FAT16 format, cannot be mounted even if it is inserted. In this case, this instruction can be executed to format the memory card to FAT16 format.

CAUTION

- This instruction removes all information on the specified disk so be careful when executing the instruction.
- A disk cannot be accessed while formatting in progress.
- This instruction does not have a timeout interval parameter because formatting takes quite a while. Once executed, the instruction ignores cancel requests, if any.

■ Programming Example

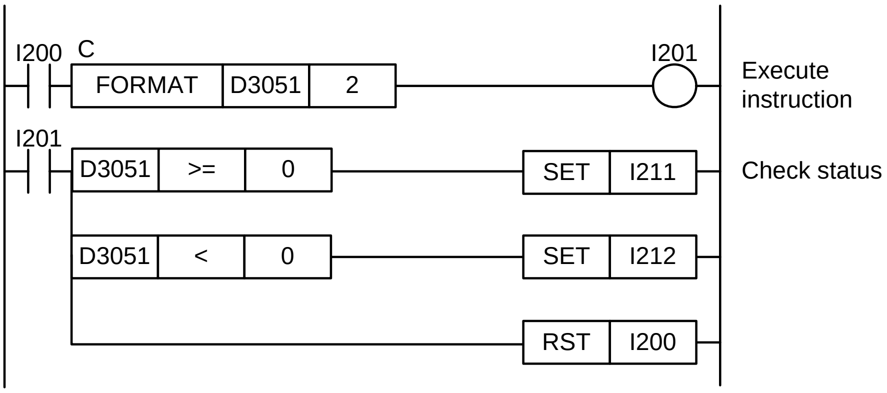


Figure C3.5.42 Example of a Format Disk Program

This sample code formats memory card CARD1.

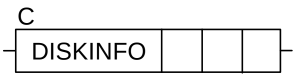
The table below shows an example of the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret=D3051	0	Status

C3.5.5.4 Disk Info (DISKINFO)

Gets information about a specified disk.

Table C3.5.182 Disk Info

Classification	FUNC No.	Instruction	Mnemonic	Symbol	Input Condition Required?		Step Count	Pro-cessing Unit	Carry
					Yes	No			
Continuous type application instruction	–	Disk Info	DISKINFO		✓	–	6	–	–

SEE ALSO

Unlike normal application instructions, the execution of a continuous type application instruction spans multiple scan cycles. For details on the execution of continuous type application instructions, see "■ Continuous Type Application Instructions" in Section C3.5.1, "Using File System Instructions".

■ Parameter

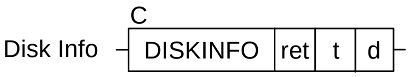


Table C3.5.183 Parameter

Parameter		Description
ret[*1]		Device for storing return status (W)
t	t+0	Timeout interval (W) [0=indefinite, 1-32767 (x 100 ms)]
	t+1	Disk selection (W) [1=\RAMDISK 2=\CARD1]
d		Device for storing disk information (W)

*1: ret (status) is table data. For details on the return status (ret), see "■ Status (Return Value)".

■ Status (Return Value)

Table C3.5.184 Status (Return Value)

Offset (word)		Description	
ret	ret+0	0	Normal exit
		< 0	Error status

SEE ALSO

For more details on error status, see "● Error Status of Continuous Type Application Instructions" of "■ Continuous Type Application Instructions" in Section C3.5.1, "Using File System Instructions".

■ Available Devices

Table C3.5.185 Available Devices

Device Parameter	X	Y	I	E	L	M	T	C	D	B	W	Z	R	V	Con-stant	Index Modification	Indirect Designation, Pointer P
ret									✓	✓	✓		✓	✓		Yes	Yes
t									✓	✓	✓		✓	✓		Yes	Yes
d									✓	✓	✓		✓	✓		Yes	Yes

Note: See Section 1.15, "Restrictions on Devices Used as Instruction Parameters" of "Sequence CPU – Instructions User's Manual" (IM34M6P12-03E).

Resource Relays

Table C3.5.186 Resource Relays Recommended for Insertion into Input Condition of Instruction to Avoid Competition

Add to Input Condition	Number	Name	Usage
✓	M1025	File/Disk Operation Group Busy	Execute instruction only if the "File/Disk Operation Group Busy" relay is OFF.

Function

Gets the following information about a specified disk.

Table C3.5.187 Returned Information

Information		Offset (word)	Size (word)	Description
Free space	Low word	+0	2	Free space [0 to 2147483647 (bytes)]
	High word	+1		
Capacity	Low word	+2	2	Capacity [0 to 2147483647 (bytes)]
	High word	+3		

Programming Example

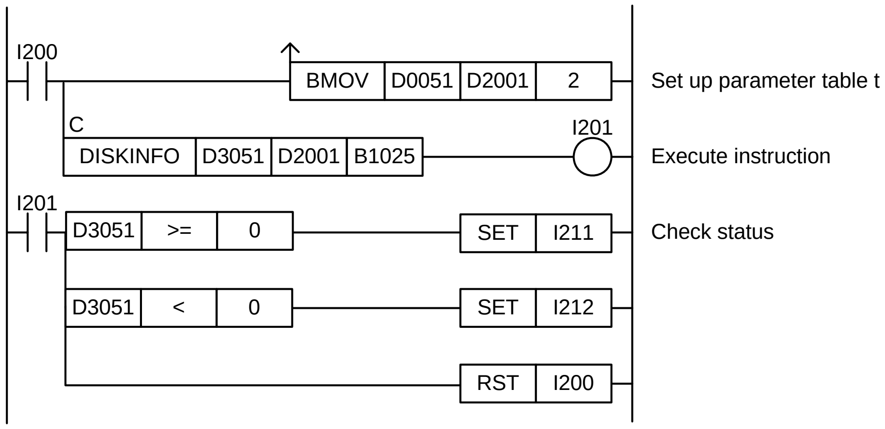


Figure C3.5.43 Example of a Disk Info Program

This sample code gets and stores disk information about memory card CARD1 to device, starting from B1025.

It specifies ret(=D3051), t(=D2001) and d(=B1025), with t set up as shown in the table below.

Device	Value	Table Parameter
t=D2001	50	Timeout interval (= 5 s)
D2002	2	Disk selection (=CARD1)

The table below shows an example of the returned status data (ret), assuming normal exit.

Device	Value	Table Parameter
ret=D3051	0	Status

The table below shows an example of the data stored to d (B1025) for a 1 gigabyte memory card.

Device	Value	Table Parameter
d=B1025	990019584	Free space
B1026		
B1027	990642176	Capacity
B1028		

C3.6 File System Sample Programs

This chapter describes some sample programs for file system instructions. These sample programs are intended to help a user better understand the instruction specifications and are not intended to be used directly in user applications.

■ List of Sample Programs

The table below shows the file structure of the sample programs provided for file system instructions. Files of these sample programs are automatically copied to their respective folders shown below when WideField2 is installed.

Table C3.6.1 Sample Program Components and Location

Sample Program Name	Component	Location
File operation and disk operation	Project	~ \Fam3pjt\CPUSample\F3SP66\FILEOPE\FILEOPE.YPJT
	CPU properties	Undefined
	Files	Undefined
ASCII file access	Project	~ \Fam3pjt\CPUSample\F3SP66\FASCII\FASCII.YPJT
	CPU properties	Undefined
	Files	Undefined
Binary file access	Project	~ \Fam3pjt\CPUSample\F3SP66\FBINARY\FBINARY.YPJT
	CPU properties	Undefined
	Files	Undefined
CSV file conversion from and to device	Project	~ \Fam3pjt\CPUSample\F3SP66\F2D2FCSV\F2D2FCSV.YPJT
	CPU properties	Undefined
	Files	~ \Fam3pjt\CPUSample\F3SP66\F2D2FCSV\DEC.CSV
Binary file conversion from and to device	Project	~ \Fam3pjt\CPUSample\F3SP66\F2D2FBIN\F2D2FBIN.YPJT
	CPU properties	Undefined
	Files	~ \Fam3pjt\CPUSample\F3SP66\F2D2FBIN\BYTE.BIN

Note: " ~ " in the "Location" column denotes the folder where Wildefiled2 is installed.

C3.6.1 File Operation and Disk Operation

■ Function and Usage

This is a sample program for file operation instructions and disk operation instructions. A F3SP66-4S and an SD memory card are required to run the sample program.

The sample program performs the following processing:

1. Initializes devices, and creates a seed file for concatenation use.
2. Checks for the presence of the work directory.
3. Creates the work directory or deletes files in the work directory.
4. Copies files from the SD memory card to the work directory.
5. Prepares to search for files with file extension "TXT".
6. Searches for a file with file extension "TXT".
7. If a file with file extension "TXT" is found, concatenates it to the seed file for concatenation.
8. If no file with file extension "TXT" is found, ends the search. Otherwise, repeats steps 6 and 7.
9. Unmounts the SD memory card.

■ Structure of Sample Program

● List of Instructions Used

The table below lists the main ladder instructions used in the sample program.

Table C3.6.2 List of File Operation and Disk Operation Instructions Used

Ladder Instruction Mnemonic	Purpose
FSTAT	Checks for presence of work directory
FMKDIR	Creates work directory
FDEL	Deletes files within work directory
FCOPY	Copies files from SD memory card to work directory
FLSFIRST	Prepares to search for files with file extension "TXT"
FLS	Finds file with file extension "TXT"
FLSFIN	Ends search for files with file extension "TXT"
FCAT	Concatenates file with file extension "TXT"
UNMOUNT	Unmounts SD memory card

● List of Special Relays Used

The table below lists the main special relays used in the sample program.

Table C3.6.3 List of Special Relays Used

Name of Special Relay	No. of Special Relay	Function
File/Disk Operation Group Busy	M1025	Checks whether file operation instruction group or disk operation instruction group is in use.
Always ON	M0033	Used for Always on circuit
1 Scan ON at Program Start	M0035	Turns on for one scan after program starts execution
US1 LED Lit	M0125	Manipulates US1 LED

● Project

The table below shows the content of the WideField2 project containing the sample program.

Table C3.6.4 Project Content

Name	Component	Description	
FILEOPE	Configuration	SP66 configuration with default setup. You can also F3SP67-6S provided you change the CPU type in the configuration.	
	Blocks	Total no. of blocks	1
		Block 1	MAIN
	Macros	Total no. of macros	1
		Macro 1	SEED (create seed file for concatenation)
	Constant definition	#INDIR	Source directory for copying
		#INFILE	Source file(s) to be copied
		#OUTDIR	Output directory for concatenated file
		#OUTFILE	Concatenated file
		#WORKDIR	Work directory
		#CATFILE	File to be concatenated
		Others, 12 definitions in total	

● CPU Properties

This sample program does not define any CPU property. Any CPU property file can be used.

● Files

The table below lists the files used in the sample program.

Table C3.6.5 List of Files Used

File Name	Input/Output	Description
ALLTEXT.FCAT	Output	Concatenated file.
*.TXT	Input	Files to be concatenated. Before running the program, you should store a few files with file extension "TXT" in the root directory of the SD memory card.

■ Ladder Program Listing

The figure on the following pages shows the ladder program listing. For details on the purpose of individual devices used in the ladder program, see the I/O comments of the block tag names of individual blocks. The macro listing has been omitted as the macro is not directly related to file and disk operation instructions.

● Project (FILEOPE) Block (MAIN)

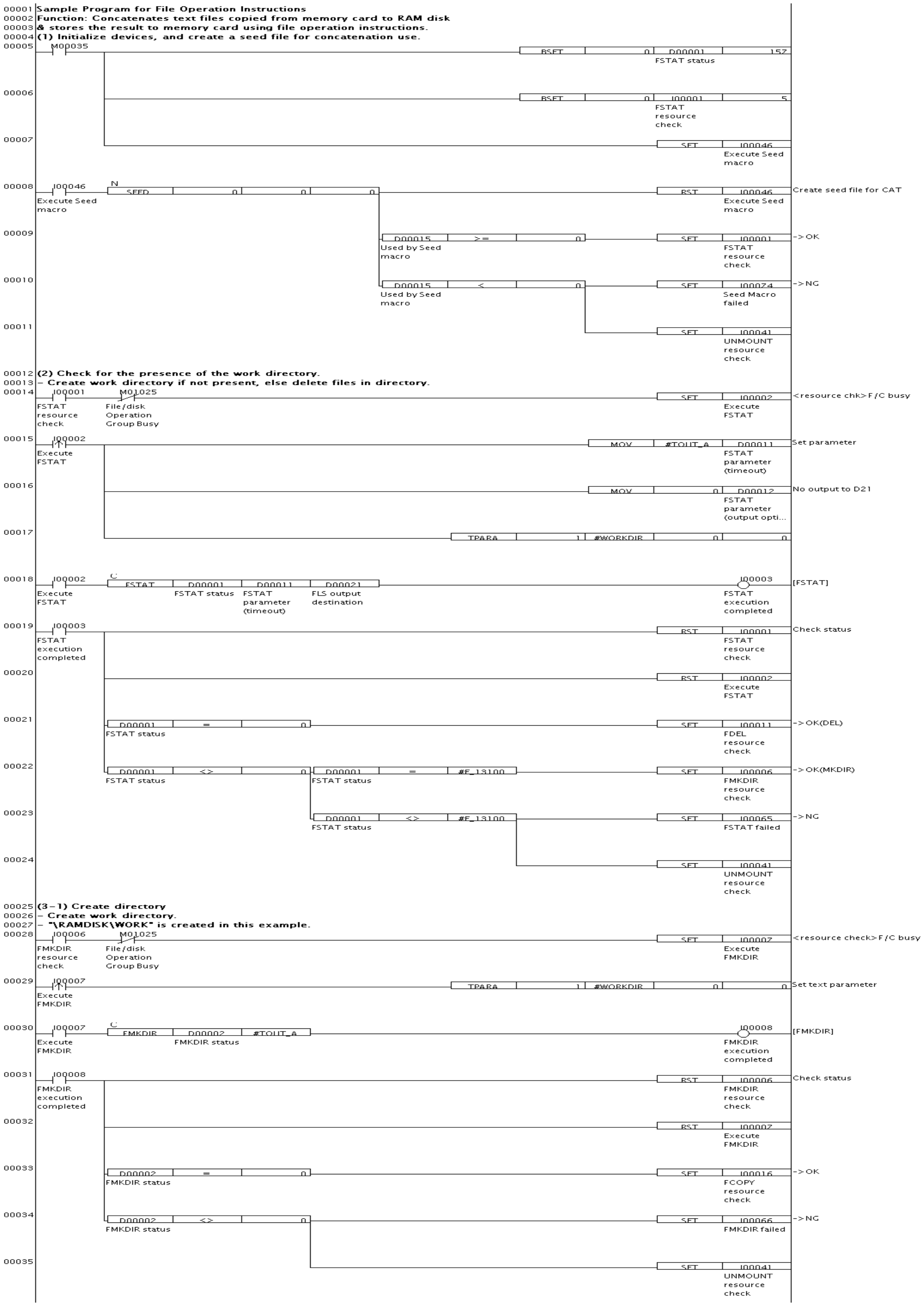
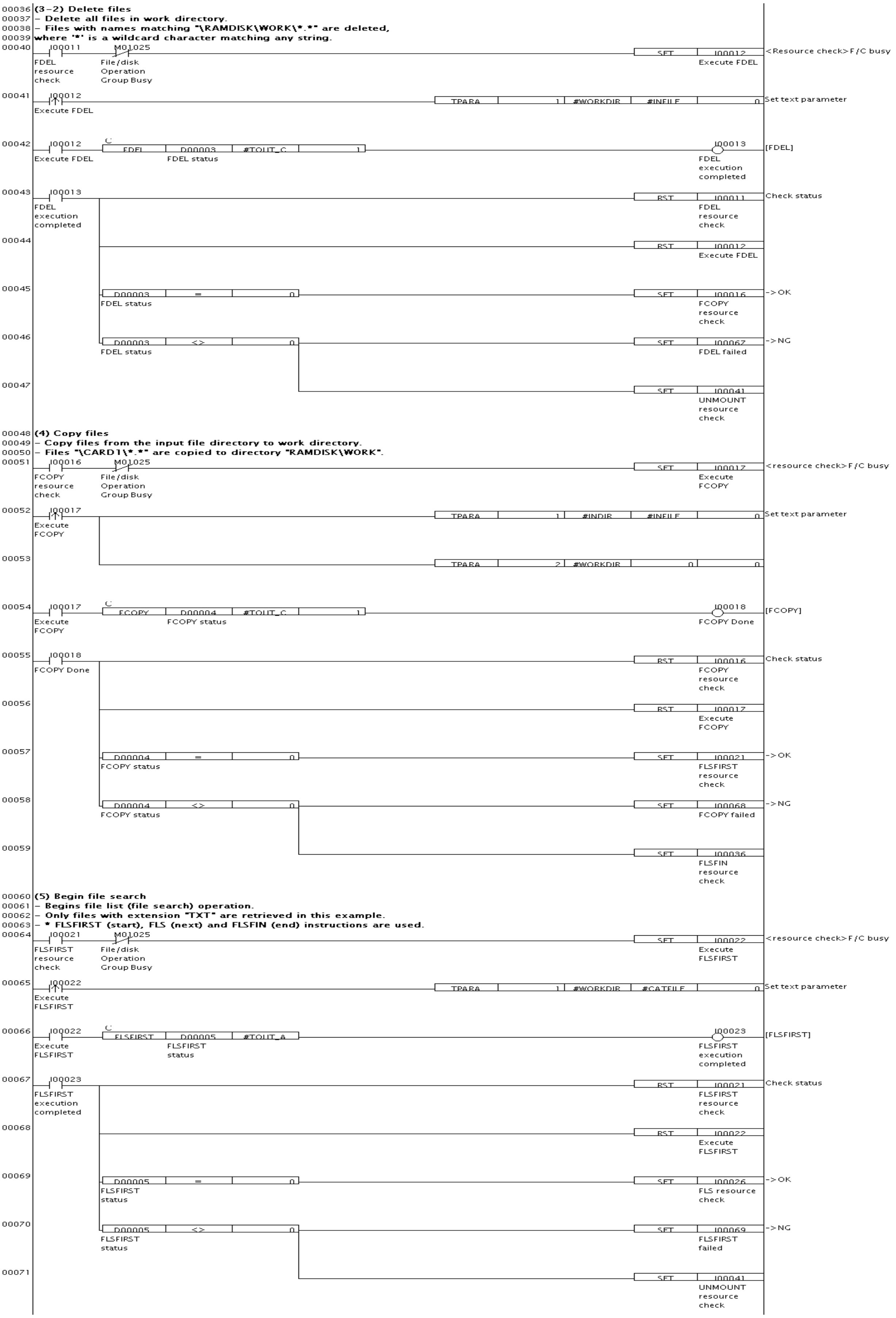

00001 Sample Program for File Operation Instructions
00002 Function: Concatenates text files copied from memory card to RAM disk
00003 & stores the result to memory card using file operation instructions.
00004 (1) Initialize devices, and create a seed file for concatenation use.
00005 M00035 BSET 0 D00001 157 FSTAT status
00006 BSET 0 I00001 5 FSTAT resource check
00007 SET I00046 Execute Seed macro
00008 I00046 Execute Seed macro N SEED 0 0 0 RST I00046 Execute Seed macro Create seed file for CAT
00009 D00015 >= 0 Used by Seed macro SET I00001 FSTAT resource check ->OK
00010 D00015 < 0 Used by Seed macro SET I00074 Seed Macro failed ->NG
00011 SET I00041 UNMOUNT resource check
00012 (2) Check for the presence of the work directory.
00013 – Create work directory if not present, else delete files in directory.
00014 I00001 FSTAT resource check M01025 File/disk Operation Group Busy SET I00002 Execute FSTAT <resource chk>F/C busy
00015 I00002 Execute FSTAT MOV #TOUT_A D00011 FSTAT parameter (timeout) Set parameter
00016 MOV 0 D00012 FSTAT parameter (output opti... No output to D21
00017 TPARA 1 #WORKDIR 0 0
00018 I00002 Execute FSTAT C FSTAT D00001 FSTAT status D00011 FSTAT parameter (timeout) D00021 FLS output destination I00003 FSTAT execution completed [FSTAT]
00019 I00003 FSTAT execution completed RST I00001 FSTAT resource check Check status
00020 RST I00002 Execute FSTAT
00021 D00001 = 0 FSTAT status SET I00011 FDEL resource check ->OK(DEL)
00022 D00001 <> 0 FSTAT status D00001 = #E_13100 FSTAT status SET I00006 FMKDIR resource check ->OK(MKDIR)
00023 D00001 <> #E_13100 FSTAT status SET I00065 FSTAT failed ->NG
00024 SET I00041 UNMOUNT resource check
00025 (3-1) Create directory
00026 – Create work directory.
00027 – "\RAMDISK\WORK" is created in this example.
00028 I00006 FMKDIR resource check M01025 File/disk Operation Group Busy SET I00007 Execute FMKDIR <resource check>F/C busy
00029 I00007 Execute FMKDIR TPARA 1 #WORKDIR 0 0 Set text parameter
00030 I00007 Execute FMKDIR C FMKDIR D00002 FMKDIR status #TOUT_A I00008 FMKDIR execution completed [FMKDIR]
00031 I00008 FMKDIR execution completed RST I00006 FMKDIR resource check Check status
00032 RST I00007 Execute FMKDIR
00033 D00002 = 0 FMKDIR status SET I00016 FCOPY resource check ->OK
00034 D00002 <> 0 FMKDIR status SET I00066 FMKDIR failed ->NG
00035 SET I00041 UNMOUNT resource check
FC0313.VSD

Figure C3.6.1 File and Disk Operation Sample Program Listing: MAIN (1/4)



(3-2) Delete files
- Delete all files in work directory.
- Files with names matching "\RAMDISK\WORK\*.*" are deleted,
where '*' is a wildcard character matching any string.

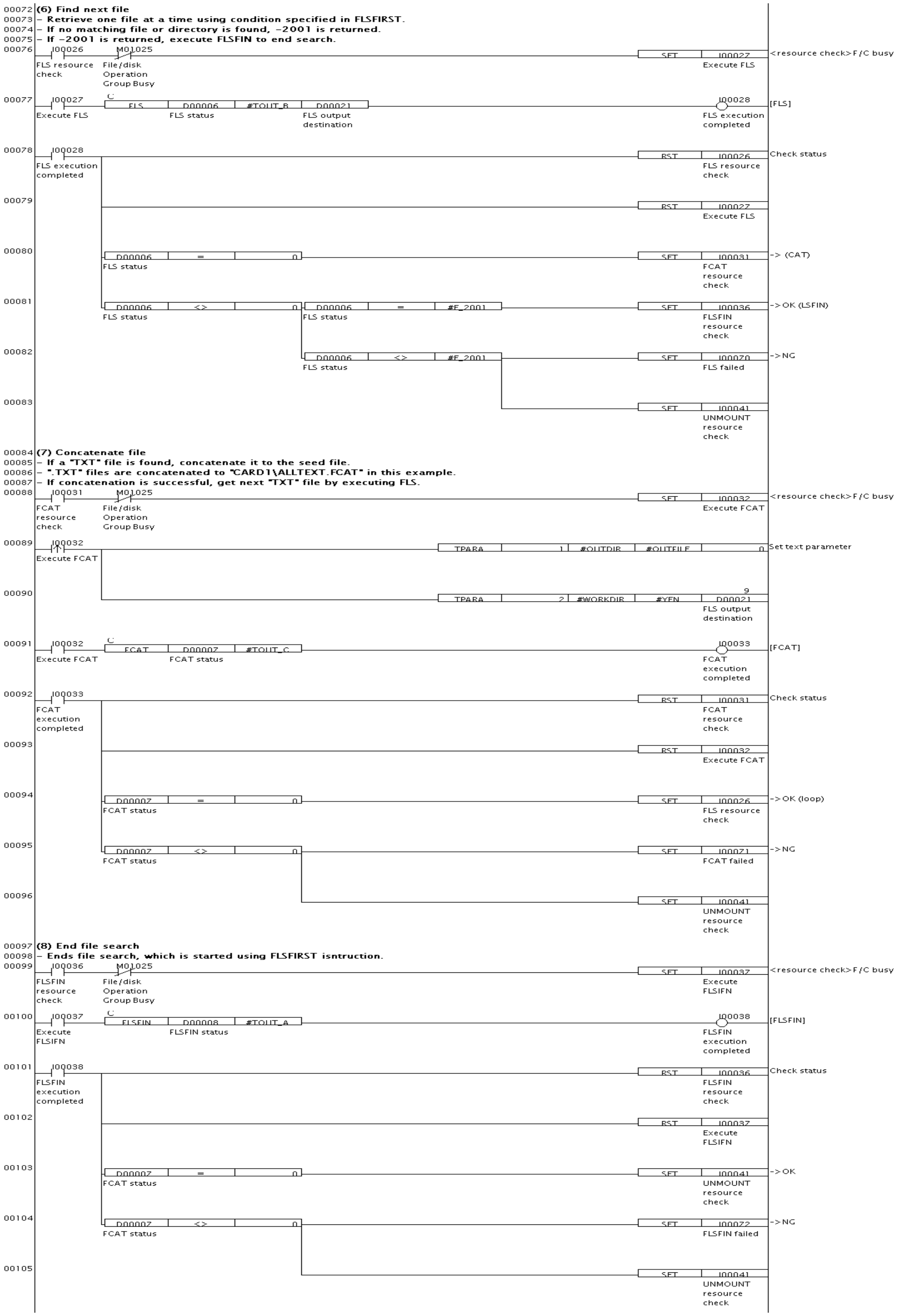
(4) Copy files
- Copy files from the input file directory to work directory.
- Files "\CARD1*.*" are copied to directory "RAMDISK\WORK".

(5) Begin file search
- Begins file list (file search) operation.
- Only files with extension "TXT" are retrieved in this example.
- * FLSFIRST (start), FLS (next) and FLSFIN (end) instructions are used.

I00011 FDEL resource check
M01025 File/disk Operation Group Busy
SET I00012 Execute FDEL
<Resource check>F/C busy
I00012 Execute FDEL
TPARA 1 #WORKDIR #INFILE 0
Set text parameter
C FDEL D00003 FDEL status #TOUT_C 1
I00013 FDEL execution completed
[FDEL]
RST I00011 FDEL resource check
Check status
RST I00012 Execute FDEL
D00003 = 0 FDEL status
SET I00016 FCOPY resource check
->OK
D00003 <> 0 FDEL status
SET I00067 FDEL failed
->NG
SET I00041 UNMOUNT resource check
I00016 FCOPY resource check
M01025 File/disk Operation Group Busy
SET I00017 Execute FCOPY
<resource check>F/C busy
I00017 Execute FCOPY
TPARA 1 #INDIR #INFILE 0
Set text parameter
TPARA 2 #WORKDIR 0 0
C FCOPY D00004 FCOPY status #TOUT_C 1
I00018 FCOPY Done
[FCOPY]
RST I00016 FCOPY resource check
Check status
RST I00017 Execute FCOPY
D00004 = 0 FCOPY status
SET I00021 FLSFIRST resource check
->OK
D00004 <> 0 FCOPY status
SET I00068 FCOPY failed
->NG
SET I00036 FLSFIN resource check
I00021 FLSFIRST resource check
M01025 File/disk Operation Group Busy
SET I00022 Execute FLSFIRST
<resource check>F/C busy
I00022 Execute FLSFIRST
TPARA 1 #WORKDIR #CATFILE 0
Set text parameter
C FLSFIRST D00005 FLSFIRST status #TOUT_A
I00023 FLSFIRST execution completed
[FLSFIRST]
RST I00021 FLSFIRST resource check
Check status
RST I00022 Execute FLSFIRST
D00005 = 0 FLSFIRST status
SET I00026 FLS resource check
->OK
D00005 <> 0 FLSFIRST status
SET I00069 FLSFIRST failed
->NG
SET I00041 UNMOUNT resource check

FC0314.VSD

Figure C3.6.2 File and Disk Operation Sample Program Listing: MAIN (2/4)



00072 (6) Find next file
00073 - Retrieve one file at a time using condition specified in FLSFIRST.
00074 - If no matching file or directory is found, -2001 is returned.
00075 - If -2001 is returned, execute FLSFIN to end search.

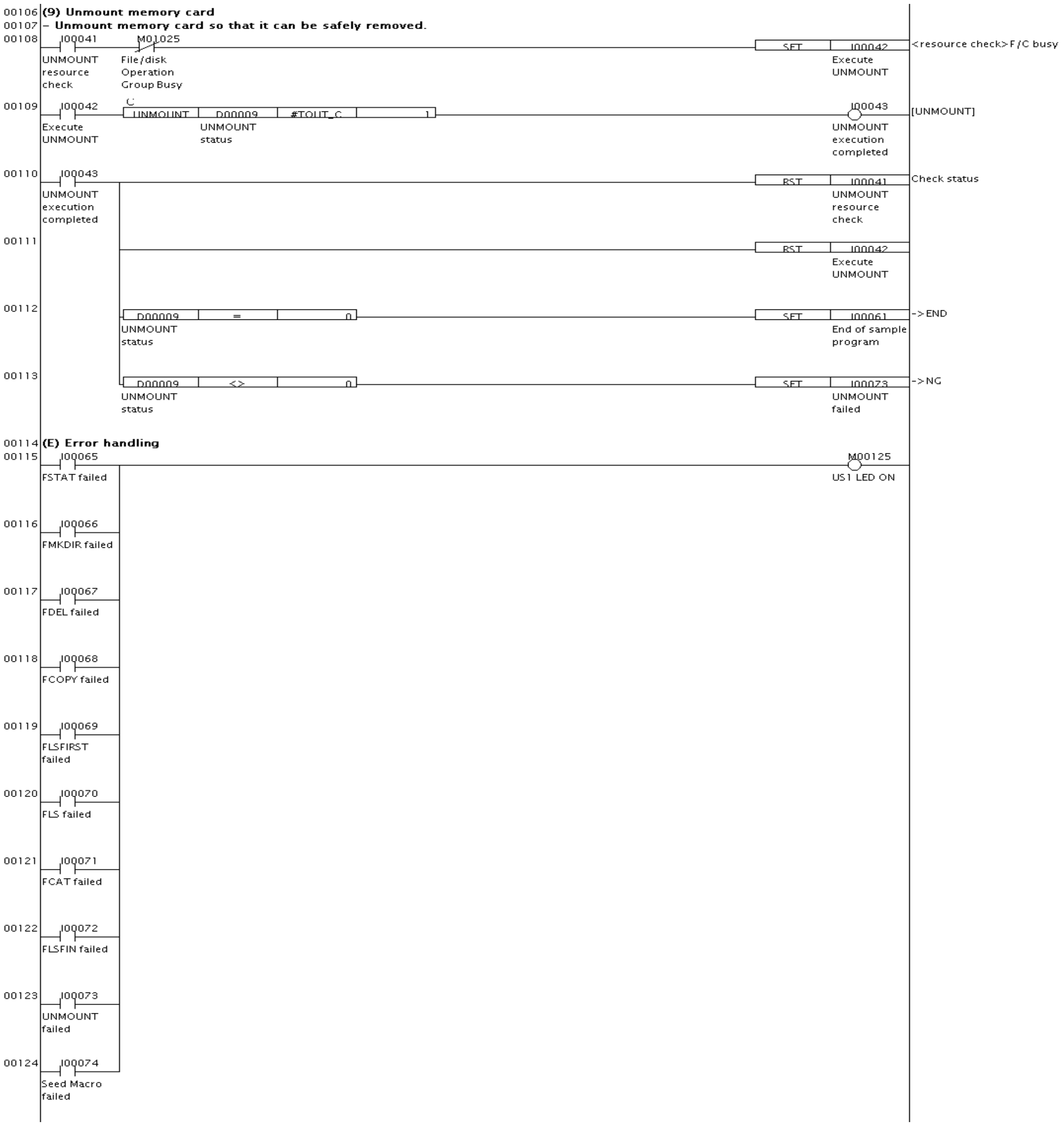
00084 (7) Concatenate file
00085 - If a "TXT" file is found, concatenate it to the seed file.
00086 - ".TXT" files are concatenated to "CARD1\ALLTEXT.FCAT" in this example.
00087 - If concatenation is successful, get next "TXT" file by executing FLS.

00097 (8) End file search
00098 - Ends file search, which is started using FLSFIRST isntruction.

00076 I00026 FLS resource check M01025 File/disk Operation Group Busy SET I00027 Execute FLS <resource check>F/C busy
00077 I00027 Execute FLS C FLS D00006 FLS status #TOUT_B D00021 FLS output destination I00028 FLS execution completed [FLS]
00078 I00028 FLS execution completed RST I00026 FLS resource check Check status
00079 RST I00027 Execute FLS
00080 D00006 FLS status = 0 SET I00031 FCAT resource check -> (CAT)
00081 D00006 FLS status <> 0 D00006 FLS status = #E_2001 SET I00036 FLSFIN resource check -> OK (LSFIN)
00082 D00006 FLS status <> #E_2001 SET I00070 FLS failed ->NG
00083 SET I00041 UNMOUNT resource check
00088 I00031 FCAT resource check M01025 File/disk Operation Group Busy SET I00032 Execute FCAT <resource check>F/C busy
00089 I00032 Execute FCAT TPARA 1 #OUTDIR #OUTFILE 0 Set text parameter
00090 TPARA 2 #WORKDIR #YEN 9 D00021 FLS output destination
00091 I00032 Execute FCAT C FCAT D00007 FCAT status #TOUT_C I00033 FCAT execution completed [FCAT]
00092 I00033 FCAT execution completed RST I00031 FCAT resource check Check status
00093 RST I00032 Execute FCAT
00094 D00007 FCAT status = 0 SET I00026 FLS resource check ->OK (loop)
00095 D00007 FCAT status <> 0 SET I00071 FCAT failed ->NG
00096 SET I00041 UNMOUNT resource check
00099 I00036 FLSFIN resource check M01025 File/disk Operation Group Busy SET I00037 Execute FLSIFN <resource check>F/C busy
00100 I00037 Execute FLSIFN C FLSFIN D00008 FLSFIN status #TOUT_A I00038 FLSFIN execution completed [FLSFIN]
00101 I00038 FLSFIN execution completed RST I00036 FLSFIN resource check Check status
00102 RST I00037 Execute FLSIFN
00103 D00007 FCAT status = 0 SET I00041 UNMOUNT resource check ->OK
00104 D00007 FCAT status <> 0 SET I00072 FLSFIN failed ->NG
00105 SET I00041 UNMOUNT resource check

FC0315.VSD

Figure C3.6.3 File and Disk Operation Sample Program Listing: MAIN (3/4)


00106 (9) Unmount memory card
00107 – Unmount memory card so that it can be safely removed.
00108 I00041 UNMOUNT resource check
M01025 File/disk Operation Group Busy
SET I00042 Execute UNMOUNT
<resource check>F/C busy
00109 I00042 Execute UNMOUNT
C UNMOUNT D00009 #TOUT_C 1
D00009 UNMOUNT status
I00043 UNMOUNT execution completed
[UNMOUNT]
00110 I00043 UNMOUNT execution completed
RST I00041 UNMOUNT resource check
Check status
00111
RST I00042 Execute UNMOUNT
00112 D00009 = 0
UNMOUNT status
SET I00061 End of sample program
->END
00113 D00009 <> 0
UNMOUNT status
SET I00073 UNMOUNT failed
->NG
00114 (E) Error handling
00115 I00065 FSTAT failed
M00125 US1 LED ON
00116 I00066 FMKDIR failed
00117 I00067 FDEL failed
00118 I00068 FCOPY failed
00119 I00069 FLSFIRST failed
00120 I00070 FLS failed
00121 I00071 FCAT failed
00122 I00072 FLSFIN failed
00123 I00073 UNMOUNT failed
00124 I00074 Seed Macro failed

FC0316.VSD

Figure C3.6.4 File and Disk Operation Sample Program Listing: MAIN (4/4)

C3.6.2 ASCII File Access

■ Function and Usage

This sample program processes an ASCII file using file access instructions. A F3SP66-4S and an SD memory card are required to run the sample program.

The sample program performs the following processing:

1. Initializes devices.
2. Checks for the presence of the file (referred to as the target file below) to be subject to text search.
3. Copies the target file to the work directory or creates the target file in the work directory.
4. Opens the target file in read-only mode.
5. Performs text search for a keyword within the target file.
6. If a match is found, stores the keyword to device, starting from B1.
7. Closes the target file.

■ Structure of Sample Program

● List of Instructions Used

The table below lists the main instructions used in the sample program.

Table C3.6.6 List of File Access Instructions Used

Ladder Instruction Mnemonic	Purpose
FOPEN	Opens target file for creation and for text search.
FCLOSE	Closes the target file.
FPUTS	Writes text to target file.
FGETS	Reads text from target file.
FSEARCHT	Performs text search within the target file.
FSEEK	Adjusts the position within the target file for reading.

● List of Special Relays Used

The table below lists the main special relays used in the sample program.

Table C3.6.7 List of Special Relays Used

Name of Special Relay	No. of Special Relay	Function
No Unused File ID	M1026	Checks for unused field ID.
File ID Open	M1041 to M1056	Checks whether a file ID to be used is already open.
File ID Busy	M1057 to M1072	Checks whether a file ID to be used is being used by another instruction.
File/Disk Operation Group Busy	M1025	Checks whether file operation instruction group or disk operation instruction group is in use.
Always ON	M0033	Used for Always on circuit
1 Scan ON at Program Start	M0035	Turns on for one scan after program starts execution
US1 LED Lit	M0125	Turns on US1 LED
US2 LED Lit	M0127	Turns on US2 LED

Project

The table below shows the content of the WideField2 project containing the sample program.

Table C3.6.8 Project Content

Name	Component	Description	
FASCII	Configuration	SP66 configuration with default setup. You can also F3SP67-6S provided you change the CPU type in the configuration.	
	Blocks	Total No. of blocks	1
		Block 1	MAIN
	Macros	Total no. of macros	0
	Constant definition	#INDIR	Source directory for copying the target file.
		#INFILE	Target file
		#WORKDIR	Work directory
		#KEYWD1	Text to be written when creating the target file
		#KEYWD2	Text to be written when creating the target file. This is also the search string.
		#KEYWD3	Text to be written when creating the target file.
		Others, 10 definitions in total.	

CPU Properties

This sample program does not define any CPU property. Any CPU property file can be used.

Files

The table below lists the files used in the sample program.

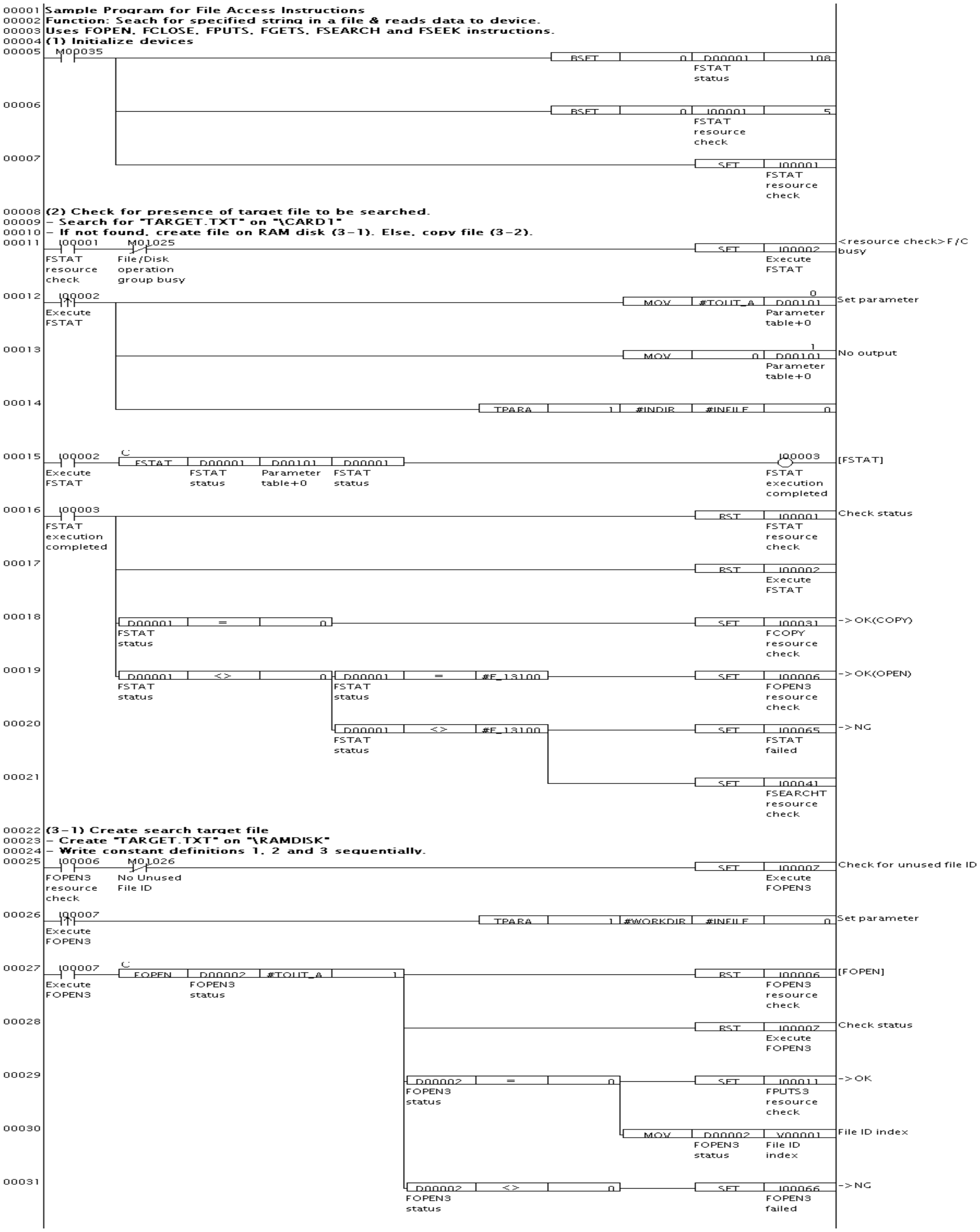
Table C3.6.9 List of Files Used

File Name	Input/Output	Description
TARGET.TXT	Input	Target file for keyword search. You can save the target file in the root directory of the SD memory card before running the program. If no file is found, the sample program automatically creates the target file.

■ Ladder Program Listing

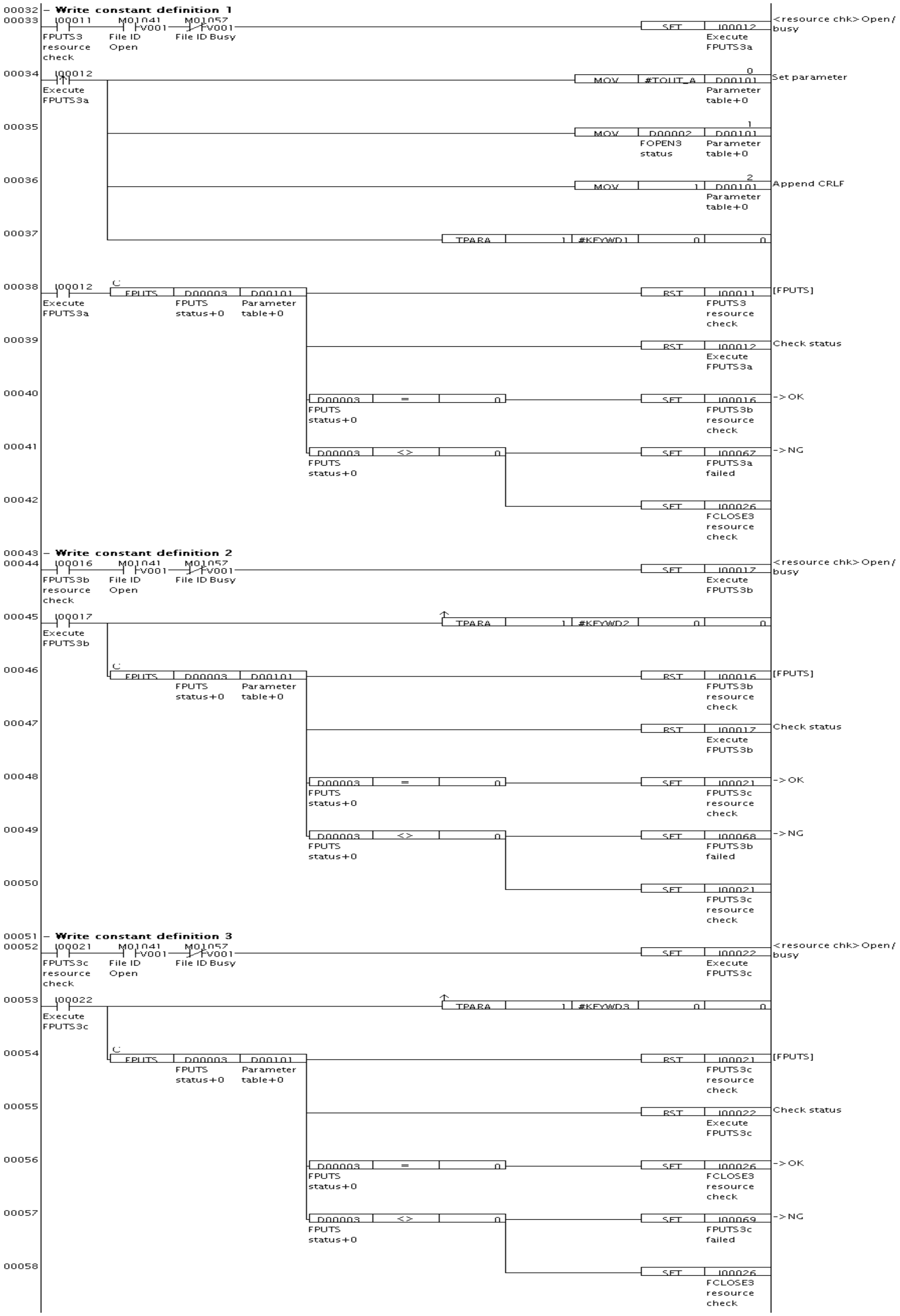
The figure on the following pages shows the ladder program listing. For details on the purpose of individual devices used in the ladder program, see the block tag names and I/O comments of each block.

● Project (FASCII) Block (MAIN)


00001 Sample Program for File Access Instructions
00002 Function: Seach for specified string in a file & reads data to device.
00003 Uses FOPEN, FCLOSE, FPUTS, FGETS, FSEARCH and FSEEK instructions.
00004 (1) Initialize devices
00005 M00035 BSET 0 D00001 108 FSTAT status
00006 BSET 0 I00001 5 FSTAT resource check
00007 SET I00001 FSTAT resource check
00008 (2) Check for presence of target file to be searched.
00009 – Search for "TARGET.TXT" on "\CARD1"
00010 – If not found, create file on RAM disk (3-1). Else, copy file (3-2).
00011 I00001 FSTAT resource check M01025 File/Disk operation group busy SET I00002 Execute FSTAT <resource check>F/C busy
00012 I00002 Execute FSTAT MOV #TOUT_A D00101 0 Parameter table+0 Set parameter
00013 MOV 0 D00101 1 Parameter table+0 No output
00014 TPARA 1 #INDIR #INFILE 0
00015 I00002 Execute FSTAT C FSTAT D00001 FSTAT status D00101 Parameter table+0 D00001 FSTAT status I00003 FSTAT execution completed [FSTAT]
00016 I00003 FSTAT execution completed RST I00001 FSTAT resource check Check status
00017 RST I00002 Execute FSTAT
00018 D00001 = 0 FSTAT status SET I00031 FCOPY resource check ->OK(COPY)
00019 D00001 <> 0 FSTAT status D00001 = #F_13100 FSTAT status SET I00006 FOPEN3 resource check ->OK(OPEN)
00020 D00001 <> #F_13100 FSTAT status SET I00065 FSTAT failed ->NG
00021 SET I00041 FSEARCHT resource check
00022 (3-1) Create search target file
00023 – Create "TARGET.TXT" on "\RAMDISK"
00024 – Write constant definitions 1, 2 and 3 sequentially.
00025 I00006 FOPEN3 resource check M01026 No Unused File ID SET I00007 Execute FOPEN3 Check for unused file ID
00026 I00007 Execute FOPEN3 TPARA 1 #WORKDIR #INFILE 0 Set parameter
00027 I00007 Execute FOPEN3 C FOPEN D00002 FOPEN3 status #TOUT_A 1 RST I00006 FOPEN3 resource check [FOPEN]
00028 RST I00007 Execute FOPEN3 Check status
00029 D00002 = 0 FOPEN3 status SET I00011 FPUTS3 resource check ->OK
00030 MOV D00002 FOPEN3 status V00001 File ID index File ID index
00031 D00002 <> 0 FOPEN3 status SET I00066 FOPEN3 failed ->NG

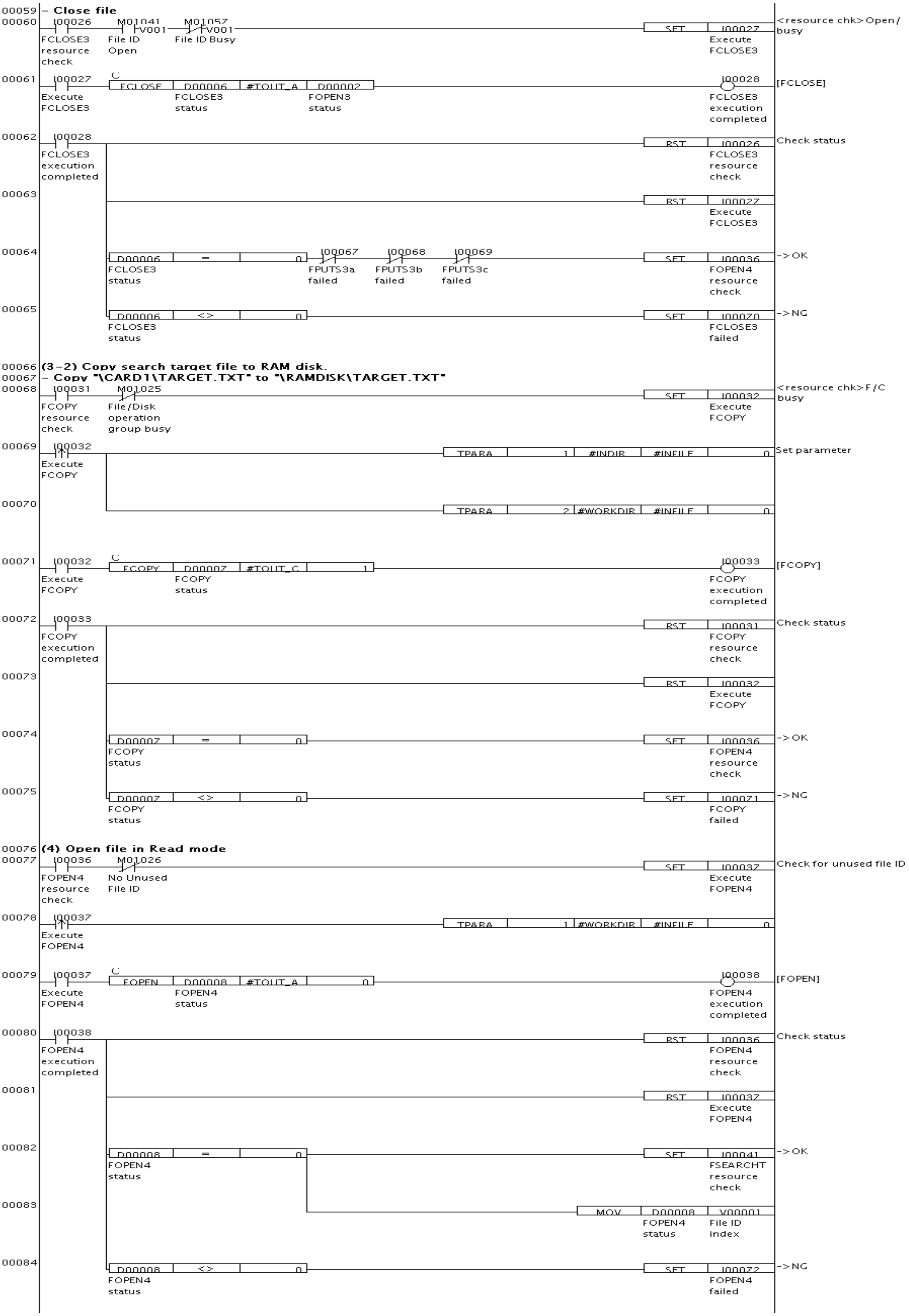
FC0317.VSD

Figure C3.6.5 ASCII File Access Sample Program Listing: MAIN (1/6)


00032 – Write constant definition 1
00033 I00011 FPUTS3 resource check
M01041 V001 File ID Open
M01057 V001 File ID Busy
SET I00012 Execute FPUTS3a
<resource chk> Open/ busy
00034 I00012 Execute FPUTS3a
MOV #TOUT_A D00101 Parameter table+0 0
Set parameter
00035 MOV D00002 FOPEN3 status D00101 Parameter table+0 1
00036 MOV 1 D00101 Parameter table+0 2
Append CRLF
00037 TPARA 1 #KEYWD1 0 0
00038 I00012 Execute FPUTS3a
C FPUTS D00003 FPUTS status+0 D00101 Parameter table+0
RST I00011 FPUTS3 resource check
[FPUTS]
00039 RST I00012 Execute FPUTS3a
Check status
00040 D00003 FPUTS status+0 = 0
SET I00016 FPUTS3b resource check
->OK
00041 D00003 FPUTS status+0 <> 0
SET I00067 FPUTS3a failed
->NG
00042 SET I00026 FCLOSE3 resource check
00043 – Write constant definition 2
00044 I00016 FPUTS3b resource check
M01041 V001 File ID Open
M01057 V001 File ID Busy
SET I00017 Execute FPUTS3b
<resource chk> Open/ busy
00045 I00017 Execute FPUTS3b
TPARA 1 #KEYWD2 0 0
00046 C FPUTS D00003 FPUTS status+0 D00101 Parameter table+0
RST I00016 FPUTS3b resource check
[FPUTS]
00047 RST I00017 Execute FPUTS3b
Check status
00048 D00003 FPUTS status+0 = 0
SET I00021 FPUTS3c resource check
->OK
00049 D00003 FPUTS status+0 <> 0
SET I00068 FPUTS3b failed
->NG
00050 SET I00021 FPUTS3c resource check
00051 – Write constant definition 3
00052 I00021 FPUTS3c resource check
M01041 V001 File ID Open
M01057 V001 File ID Busy
SET I00022 Execute FPUTS3c
<resource chk> Open/ busy
00053 I00022 Execute FPUTS3c
TPARA 1 #KEYWD3 0 0
00054 C FPUTS D00003 FPUTS status+0 D00101 Parameter table+0
RST I00021 FPUTS3c resource check
[FPUTS]
00055 RST I00022 Execute FPUTS3c
Check status
00056 D00003 FPUTS status+0 = 0
SET I00026 FCLOSE3 resource check
->OK
00057 D00003 FPUTS status+0 <> 0
SET I00069 FPUTS3c failed
->NG
00058 SET I00026 FCLOSE3 resource check

FC0318.VSD

Figure C3.6.6 ASCII File Access Sample Program Listing: MAIN (2/6)



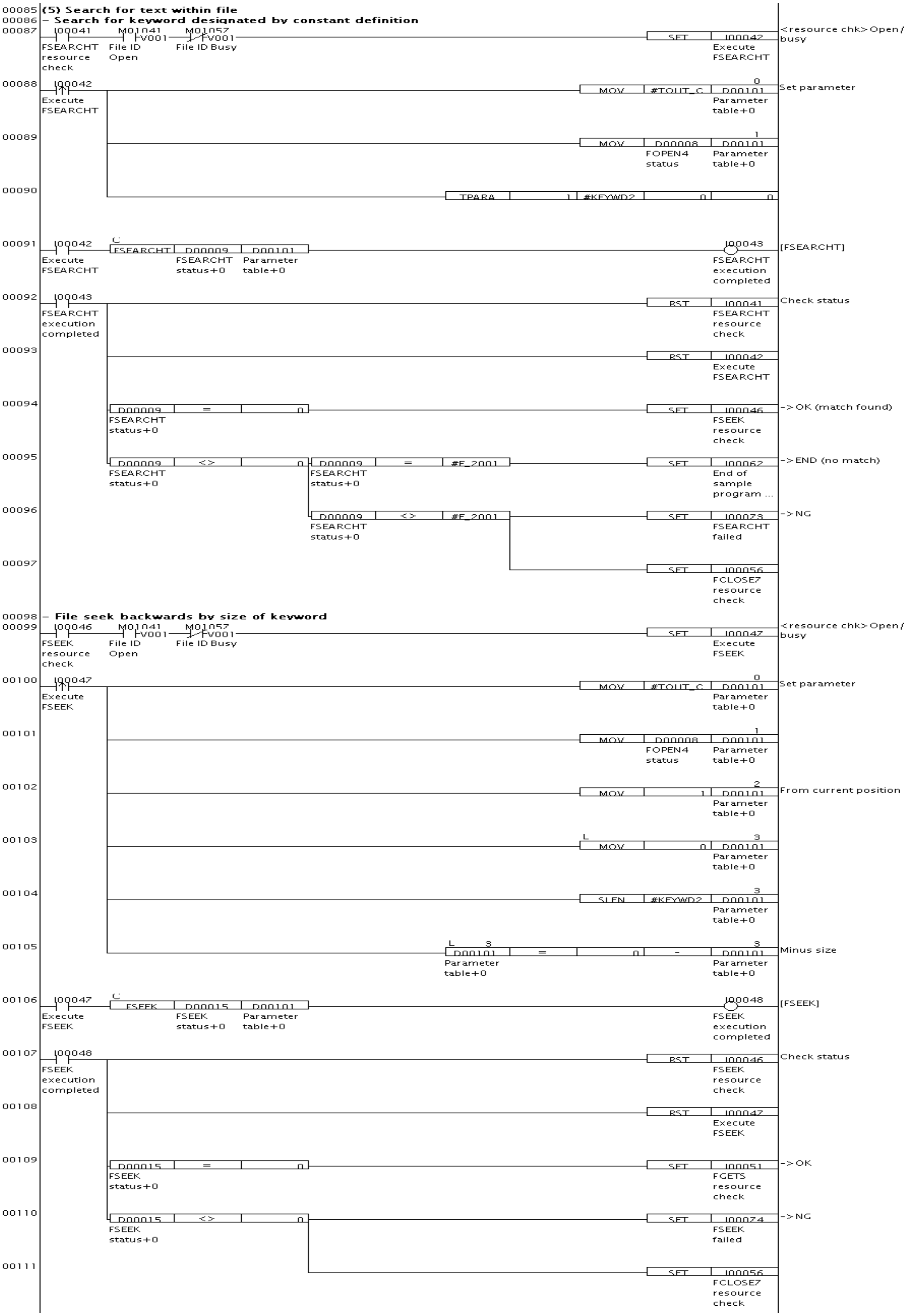
```
00059 - Close file
00060  I00026            M01041 V001      M01057 V001 (NC)              SET  I00027                  <resource chk>Open/busy
       FCLOSE3 resource  File ID Open     File ID Busy                       Execute FCLOSE3
       check
00061  I00027   C FCLOSE  D00006  #TOUT_A  D00002                            I00028                  [FCLOSE]
       Execute FCLOSE3    FCLOSE3 status   FOPEN3 status                     FCLOSE3 execution completed
00062  I00028                                                         RST  I00026                  Check status
       FCLOSE3 execution completed                                        FCLOSE3 resource check
00063                                                                 RST  I00027
                                                                           Execute FCLOSE3
00064        D00006 = 0   I00067 (NC)  I00068 (NC)  I00069 (NC)        SET  I00036                  ->OK
             FCLOSE3 status  FPUTS3a failed  FPUTS3b failed  FPUTS3c failed  FOPEN4 resource check
00065        D00006 <> 0                                              SET  I00070                  ->NG
             FCLOSE3 status                                               FCLOSE3 failed

00066 (3-2) Copy search target file to RAM disk.
00067 - Copy "\CARD1\TARGET.TXT" to "\RAMDISK\TARGET.TXT"
00068  I00031            M01025 (NC)                                  SET  I00032                  <resource chk>F/C busy
       FCOPY resource    File/Disk operation                               Execute FCOPY
       check             group busy
00069  I00032 (↑)                          TPARA  1  #INDIR   #INFILE   0                          Set parameter
       Execute FCOPY
00070                                      TPARA  2  #WORKDIR #INFILE   0
00071  I00032   C FCOPY  D00007  #TOUT_C  1                               I00033                  [FCOPY]
       Execute FCOPY      FCOPY status                                     FCOPY execution completed
00072  I00033                                                         RST  I00031                  Check status
       FCOPY execution completed                                          FCOPY resource check
00073                                                                 RST  I00032
                                                                           Execute FCOPY
00074        D00007 = 0                                               SET  I00036                  ->OK
             FCOPY status                                                 FOPEN4 resource check
00075        D00007 <> 0                                              SET  I00071                  ->NG
             FCOPY status                                                 FCOPY failed

00076 (4) Open file in Read mode
00077  I00036            M01026 (NC)                                  SET  I00037                  Check for unused file ID
       FOPEN4 resource   No Unused File ID                                 Execute FOPEN4
       check
00078  I00037 (↑)                          TPARA  1  #WORKDIR #INFILE   0
       Execute FOPEN4
00079  I00037   C FOPEN  D00008  #TOUT_A  0                               I00038                  [FOPEN]
       Execute FOPEN4     FOPEN4 status                                    FOPEN4 execution completed
00080  I00038                                                         RST  I00036                  Check status
       FOPEN4 execution completed                                         FOPEN4 resource check
00081                                                                 RST  I00037
                                                                           Execute FOPEN4
00082        D00008 = 0                                               SET  I00041                  ->OK
             FOPEN4 status                                                FSEARCHT resource check
00083                                                   MOV  D00008  V00001
                                                             FOPEN4 status  File ID index
00084        D00008 <> 0                                              SET  I00072                  ->NG
             FOPEN4 status                                                FOPEN4 failed
```

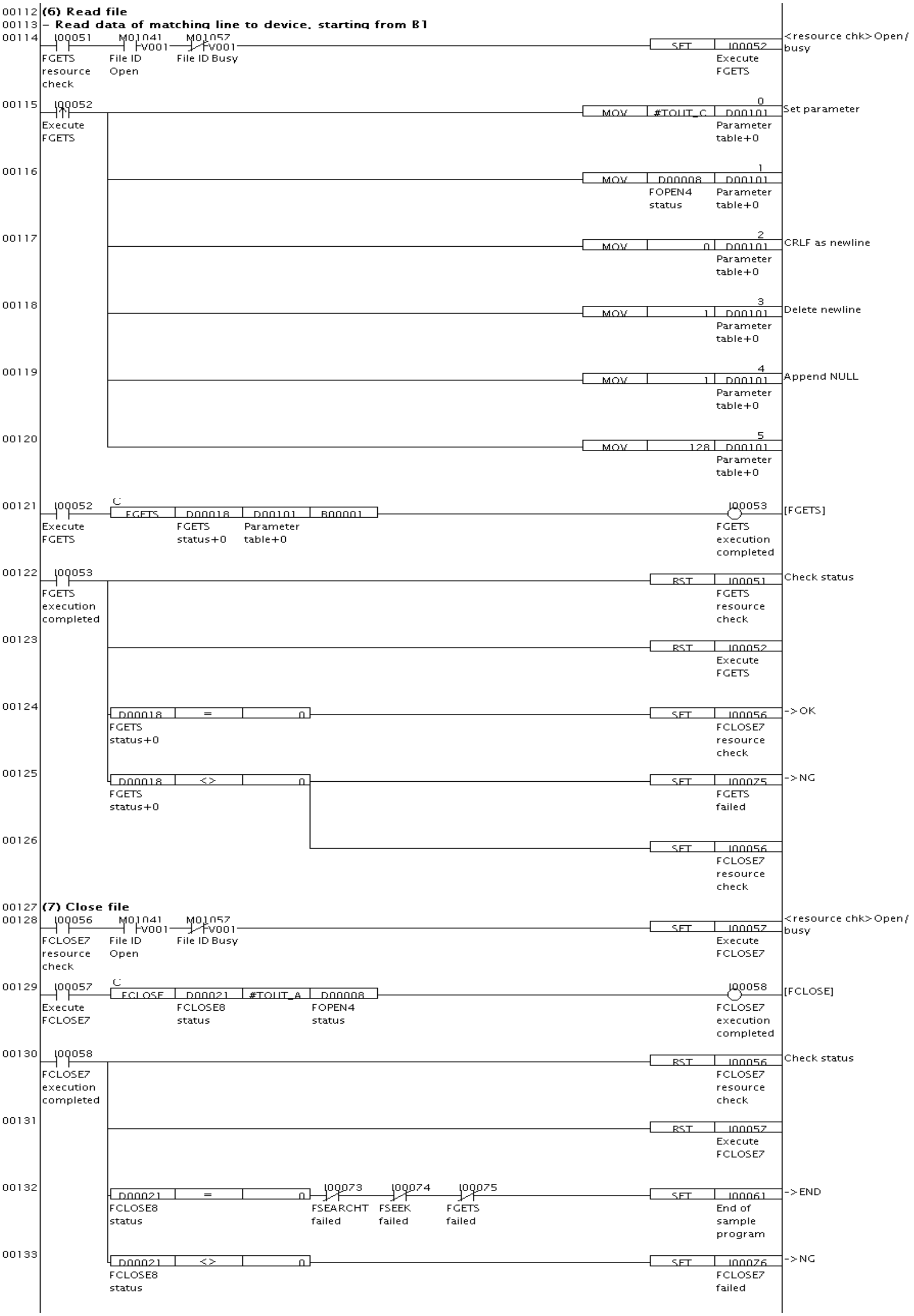
FC0319.VSD

Figure C3.6.7 ASCII File Access Sample Program Listing: MAIN (3/6)


00085 (5) Search for text within file
00086 – Search for keyword designated by constant definition
00087 I00041 FSEARCHT resource check
M01041 FV001 File ID Open
M01057 FV001 File ID Busy
SET I00042 Execute FSEARCHT
<resource chk> Open/ busy
00088 I00042 Execute FSEARCHT
MOV #TOUT_C D00101 0 Parameter table+0
Set parameter
00089 MOV D00008 FOPEN4 status D00101 1 Parameter table+0
00090 TPARA 1 #KEYWD2 0 0
00091 I00042 Execute FSEARCHT
C FSEARCHT D00009 FSEARCHT status+0 D00101 Parameter table+0
I00043 FSEARCHT execution completed
[FSEARCHT]
00092 I00043 FSEARCHT execution completed
RST I00041 FSEARCHT resource check
Check status
00093 RST I00042 Execute FSEARCHT
00094 D00009 FSEARCHT status+0 = 0
SET I00046 FSEEK resource check
-> OK (match found)
00095 D00009 FSEARCHT status+0 <> 0
D00009 FSEARCHT status+0 = #E_2001
SET I00062 End of sample program ...
-> END (no match)
00096 D00009 FSEARCHT status+0 <> #E_2001
SET I00073 FSEARCHT failed
-> NG
00097 SET I00056 FCLOSE7 resource check
00098 – File seek backwards by size of keyword
00099 I00046 FSEEK resource check
M01041 FV001 File ID Open
M01057 FV001 File ID Busy
SET I00047 Execute FSEEK
<resource chk> Open/ busy
00100 I00047 Execute FSEEK
MOV #TOUT_C D00101 0 Parameter table+0
Set parameter
00101 MOV D00008 FOPEN4 status D00101 1 Parameter table+0
00102 MOV 1 D00101 2 Parameter table+0
From current position
00103 L MOV 0 D00101 3 Parameter table+0
00104 SLEN #KEYWD2 D00101 3 Parameter table+0
00105 L 3 D00101 Parameter table+0 = 0 - D00101 3 Parameter table+0
Minus size
00106 I00047 Execute FSEEK
C FSEEK D00015 FSEEK status+0 D00101 Parameter table+0
I00048 FSEEK execution completed
[FSEEK]
00107 I00048 FSEEK execution completed
RST I00046 FSEEK resource check
Check status
00108 RST I00047 Execute FSEEK
00109 D00015 FSEEK status+0 = 0
SET I00051 FGETS resource check
-> OK
00110 D00015 FSEEK status+0 <> 0
SET I00074 FSEEK failed
-> NG
00111 SET I00056 FCLOSE7 resource check

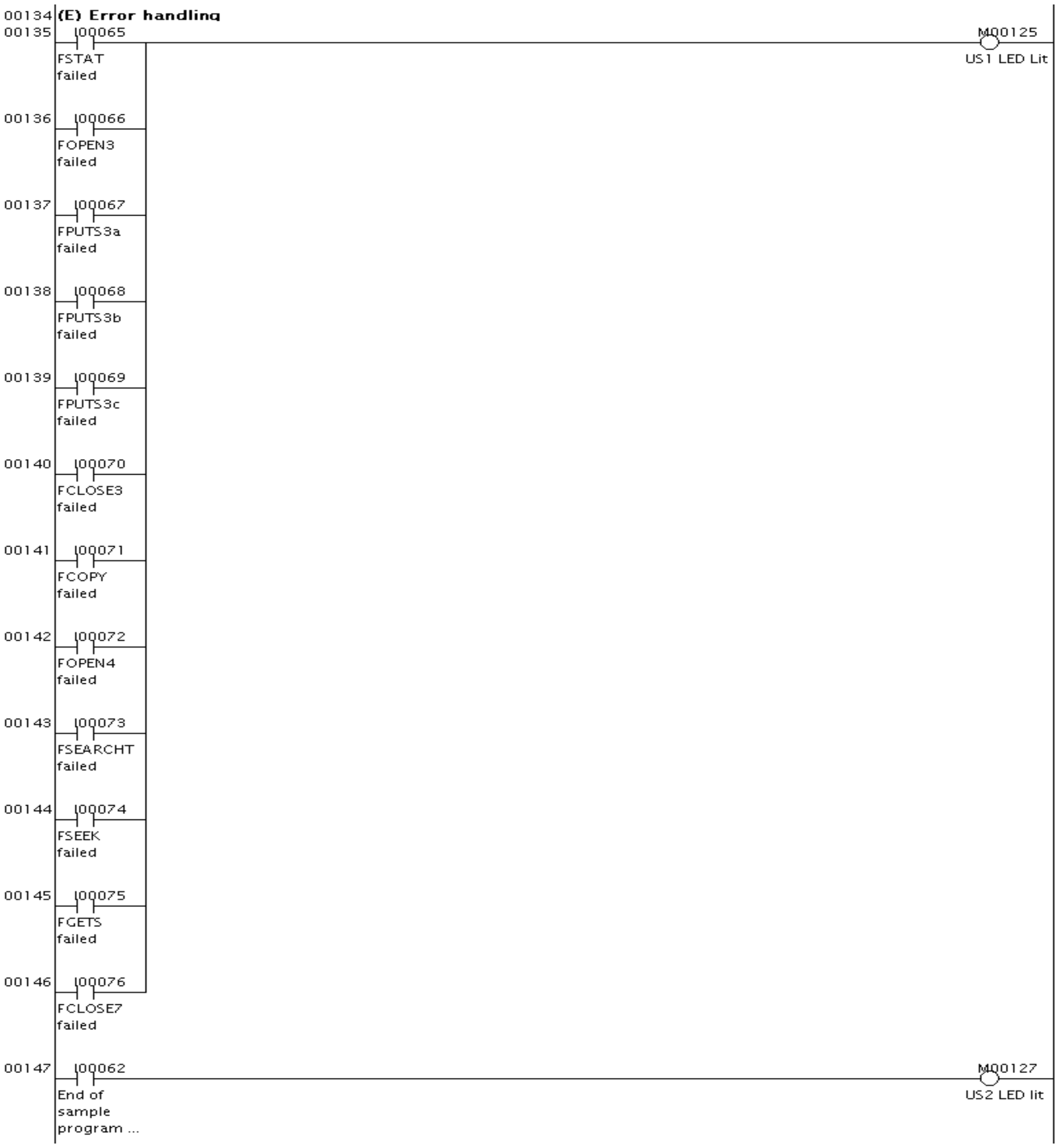
FC0320.VSD

Figure C3.6.8 ASCII File Access Sample Program Listing: MAIN (4/6)


00112 (6) Read file
00113 – Read data of matching line to device, starting from B1
00114 I00051 FGETS resource check
M01041 V001 File ID Open
M01057 V001 File ID Busy
SET I00052 Execute FGETS
<resource chk> Open/busy
00115 I00052 Execute FGETS
MOV #TOUT_C D00101 0 Parameter table+0
Set parameter
00116 MOV D00008 FOPEN4 status D00101 1 Parameter table+0
00117 MOV 0 D00101 2 Parameter table+0
CRLF as newline
00118 MOV 1 D00101 3 Parameter table+0
Delete newline
00119 MOV 1 D00101 4 Parameter table+0
Append NULL
00120 MOV 128 D00101 5 Parameter table+0
00121 I00052 Execute FGETS
C FGETS D00018 FGETS status+0 D00101 Parameter table+0 B00001
I00053 FGETS execution completed
[FGETS]
00122 I00053 FGETS execution completed
RST I00051 FGETS resource check
Check status
00123 RST I00052 Execute FGETS
00124 D00018 FGETS status+0 = 0
SET I00056 FCLOSE7 resource check
->OK
00125 D00018 FGETS status+0 <> 0
SET I00075 FGETS failed
->NG
00126 SET I00056 FCLOSE7 resource check
00127 (7) Close file
00128 I00056 FCLOSE7 resource check
M01041 V001 File ID Open
M01057 V001 File ID Busy
SET I00057 Execute FCLOSE7
<resource chk> Open/busy
00129 I00057 Execute FCLOSE7
C FCLOSE D00021 FCLOSE8 status #TOUT_A D00008 FOPEN4 status
I00058 FCLOSE7 execution completed
[FCLOSE]
00130 I00058 FCLOSE7 execution completed
RST I00056 FCLOSE7 resource check
Check status
00131 RST I00057 Execute FCLOSE7
00132 D00021 FCLOSE8 status = 0
I00073 FSEARCHT failed
I00074 FSEEK failed
I00075 FGETS failed
SET I00061 End of sample program
->END
00133 D00021 FCLOSE8 status <> 0
SET I00076 FCLOSE7 failed
->NG

FC0321.VSD

Figure C3.6.9 ASCII File Access Sample Program Listing: MAIN (5/6)


00134 (E) Error handling
00135 I00065 FSTAT failed
M00125 US1 LED Lit
00136 I00066 FOPEN3 failed
00137 I00067 FPUTS3a failed
00138 I00068 FPUTS3b failed
00139 I00069 FPUTS3c failed
00140 I00070 FCLOSE3 failed
00141 I00071 FCOPY failed
00142 I00072 FOPEN4 failed
00143 I00073 FSEARCHT failed
00144 I00074 FSEEK failed
00145 I00075 FGETS failed
00146 I00076 FCLOSE7 failed
00147 I00062 End of sample program ...
M00127 US2 LED lit

FC0322.VSD

Figure C3.6.10 ASCII File Access Sample Program Listing: MAIN (6/6)

C3.6.3 Binary File Access

■ Function and Usage

This sample program processes a binary file using file access instructions. A F3SP66-4S and an SD memory card are required to run the sample program.

The sample program performs the following processing:

1. Initializes devices.
2. Checks for the presence of the file (referred to as the target file below) to be subject to binary search.
3. Copies the target file to the work directory or creates the target file in the work directory.
4. Opens the target file in read-only mode.
5. Performs binary search for a keyword within the target file.
6. If a match is found, reads the keyword to device, starting from B1.
7. Closes the target file.

■ Structure of Sample Program

● List of Instructions Used

The table below shows the main instructions used in the sample program.

Table C3.6.10 List of File Access Instructions Used

Ladder Instruction Mnemonic	Usage
FOPEN	Opens target file for creation and for text search.
FCLOSE	Closes the target file.
FWRITE	Writes data blocks to target file.
FREAD	Reads data blocks from target file.
FSEARCHB	Performs binary search within the target file.
FSEEK	Adjusts the position within the target file for reading.

● List of Special Relays Used

The table below lists the main special relays used in the sample program.

Table C3.6.11 List of Special Relays Used

Name of Special Relay	No. of Special Relay	Function
No Unused File ID	M1026	Checks for unused field ID.
File ID Open	M1041 to M1056	Checks whether file ID to be used is already open.
File ID Busy	M1057 to M1072	Checks whether a file ID to be used is being used by another instruction.
File/Disk Operation Group Busy	M1025	Checks whether file operation instruction group or disk operation instruction group is in use.
Always ON	M0033	Used for Always on circuit
1 Scan ON at Program Start	M0035	Turns on for one scan after program starts execution
US1 LED	M0125	Turns on US1 LED
US2 LED Lit	M0127	Turns on US2 LED

● Project

The table below shows the content of the WideField2 project containing the sample program.

Table C3.6.12 Project Content

Name	Component	Description	
FBINARY	Configuration	SP66 configuration with default setup. You can also F3SP67-6S provided you change the CPU type in the configuration.	
	Blocks	Total No. of blocks	1
		Block 1	MAIN
	Macros	Total no. of macros	0
	Constant definition	#INDIR	Source directory for copying the target file.
		#INFILE	Target file
		#WORKDIR	Work directory
		#KEYWD1	Data to be written when creating target file. This is also the keyword for binary search within the file.
		#KWSIZE1	Keyword size
		#KWSIZ1L	Long word type definition of keyword size
		Others, 10 definitions in total.	

● CPU Properties

This sample program does not define any CPU property. Any CPU property file can be used.

● Files

The table below lists the files used in the sample program.

Table C3.6.13 List of Files Used

File Name	Input/Output	Description
TARGET.BIN	Input	Target file for keyword search. You can save the target file in the root directory of the SD memory card before running the program. If no file is found, the sample program automatically creates the target file.

■ Ladder Program Listing

The figure on the following pages shows the ladder program listing. For details on the purpose of individual devices used in the ladder program, see the block tag names and I/O comments of each block.

● Project (FBINARY) Block (MAIN)

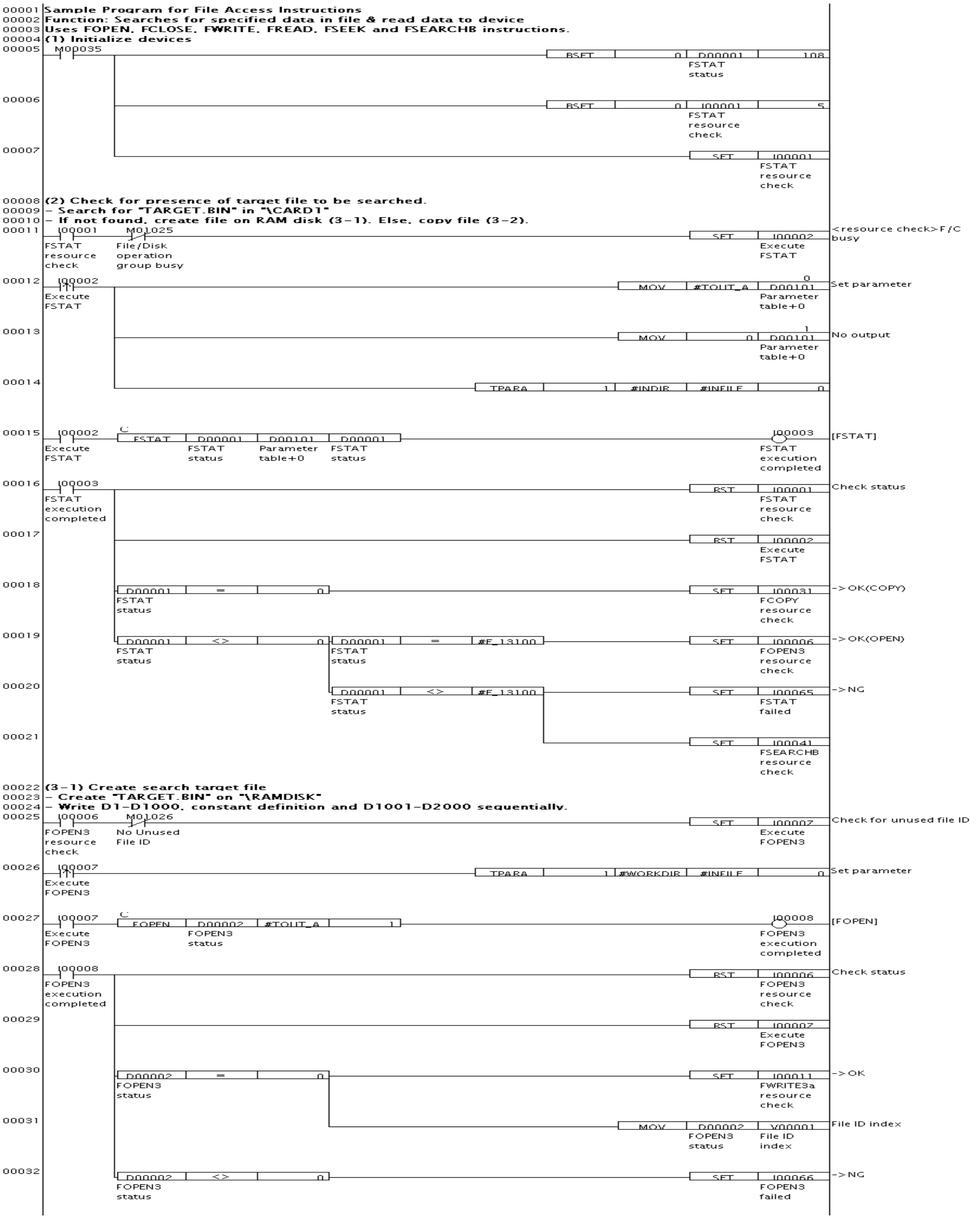
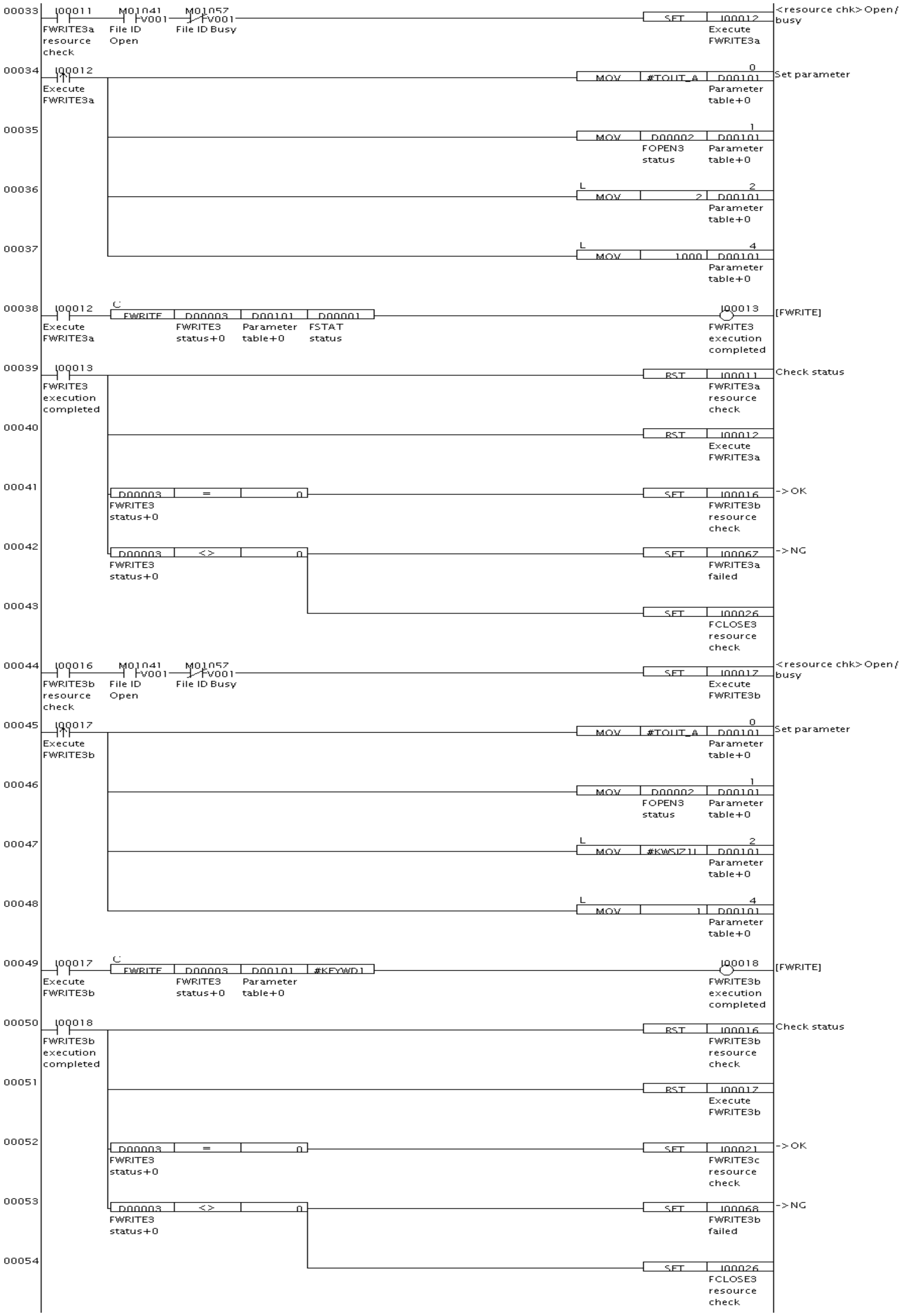
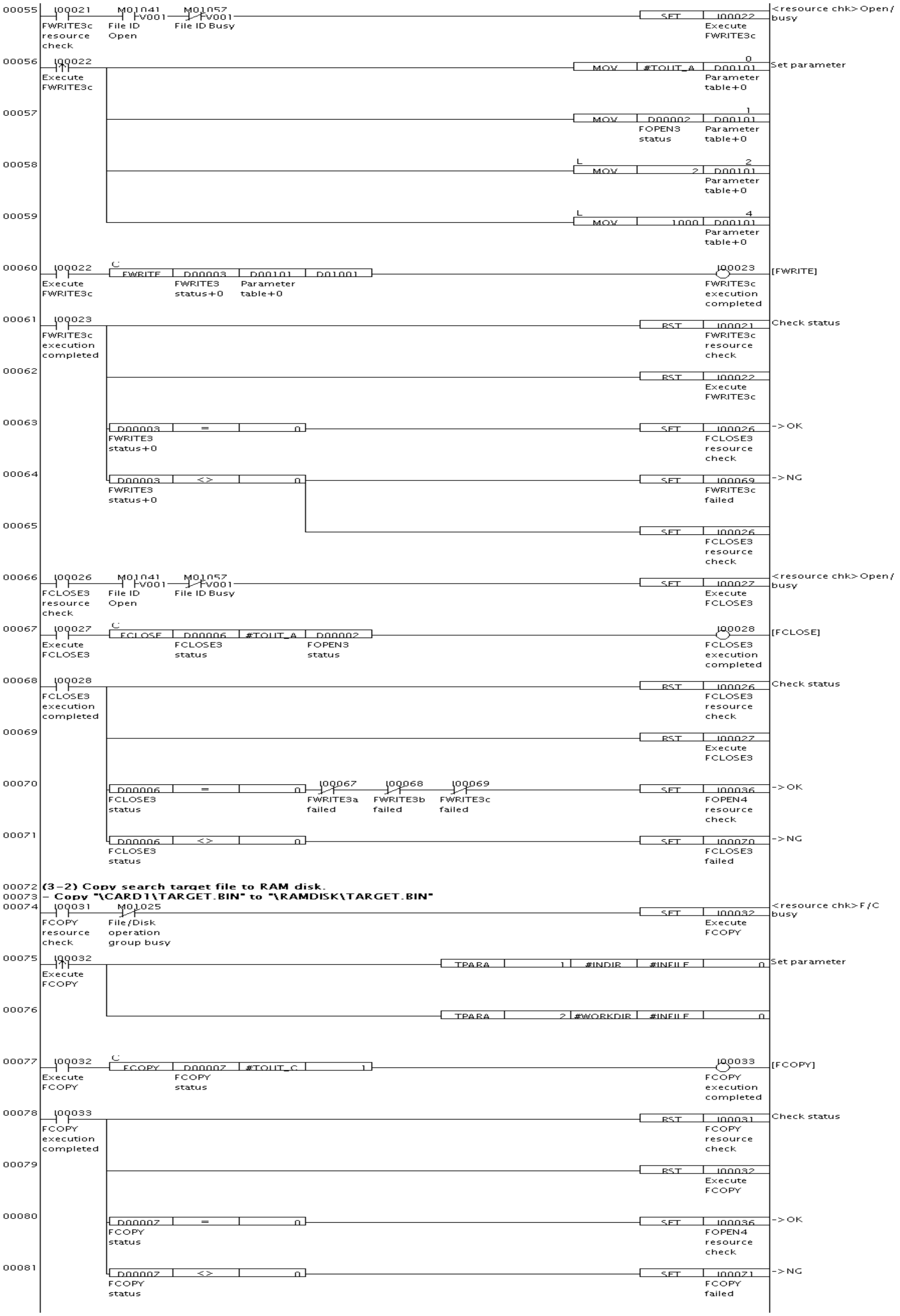

00001 Sample Program for File Access Instructions
00002 Function: Searches for specified data in file & read data to device
00003 Uses FOPEN, FCLOSE, FWRITE, FREAD, FSEEK and FSEARCHB instructions.
00004 (1) Initialize devices
00005 M00035 BSET 0 D00001 108 FSTAT status
00006 BSET 0 I00001 5 FSTAT resource check
00007 SET I00001 FSTAT resource check
00008 (2) Check for presence of target file to be searched.
00009 - Search for "TARGET.BIN" in "\CARD1"
00010 - If not found, create file on RAM disk (3-1). Else, copy file (3-2).
00011 I00001 FSTAT resource check M01025 File/Disk operation group busy SET I00002 Execute FSTAT <resource check>F/C busy
00012 I00002 Execute FSTAT MOV #TOUT_A D00101 0 Parameter table+0 Set parameter
00013 MOV 0 D00101 1 Parameter table+0 No output
00014 TPARA 1 #INDIR #INFILE 0
00015 I00002 Execute FSTAT C FSTAT D00001 FSTAT status D00101 Parameter table+0 D00001 FSTAT status I00003 FSTAT execution completed [FSTAT]
00016 I00003 FSTAT execution completed RST I00001 FSTAT resource check Check status
00017 RST I00002 Execute FSTAT
00018 D00001 FSTAT status = 0 SET I00031 FCOPY resource check ->OK(COPY)
00019 D00001 FSTAT status <> 0 D00001 FSTAT status = #E_13100 SET I00006 FOPEN3 resource check ->OK(OPEN)
00020 D00001 FSTAT status <> #E_13100 SET I00065 FSTAT failed ->NG
00021 SET I00041 FSEARCHB resource check
00022 (3-1) Create search target file
00023 - Create "TARGET.BIN" on "\RAMDISK"
00024 - Write D1-D1000, constant definition and D1001-D2000 sequentially.
00025 I00006 FOPEN3 resource check M01026 No Unused File ID SET I00007 Execute FOPEN3 Check for unused file ID
00026 I00007 Execute FOPEN3 TPARA 1 #WORKDIR #INFILE 0 Set parameter
00027 I00007 Execute FOPEN3 C FOPEN D00002 FOPEN3 status #TOUT_A 1 I00008 FOPEN3 execution completed [FOPEN]
00028 I00008 FOPEN3 execution completed RST I00006 FOPEN3 resource check Check status
00029 RST I00007 Execute FOPEN3
00030 D00002 FOPEN3 status = 0 SET I00011 FWRITE3a resource check ->OK
00031 MOV D00002 FOPEN3 status V00001 File ID index File ID index
00032 D00002 FOPEN3 status <> 0 SET I00066 FOPEN3 failed ->NG
FC0323.VSD

Figure C3.6.11 Binary File Access Sample Program Listing: MAIN (1/6)


00033
I00011 FWRITE3a resource check
M01041 FV001 File ID Open
M01057 FV001 File ID Busy
SET I00012 Execute FWRITE3a
<resource chk>Open/busy
00034
I00012 Execute FWRITE3a
MOV #TOUT_A D00101 0 Parameter table+0
Set parameter
00035
MOV D00002 FOPEN3 status D00101 1 Parameter table+0
00036
L MOV 2 D00101 2 Parameter table+0
00037
L MOV 1000 D00101 4 Parameter table+0
00038
I00012 Execute FWRITE3a
C FWRITE D00003 FWRITE3 status+0 D00101 Parameter table+0 D00001 FSTAT status
I00013 FWRITE3 execution completed
[FWRITE]
00039
I00013 FWRITE3 execution completed
RST I00011 FWRITE3a resource check
Check status
00040
RST I00012 Execute FWRITE3a
00041
D00003 FWRITE3 status+0 = 0
SET I00016 FWRITE3b resource check
->OK
00042
D00003 FWRITE3 status+0 <> 0
SET I00067 FWRITE3a failed
->NG
00043
SET I00026 FCLOSE3 resource check
00044
I00016 FWRITE3b resource check
M01041 FV001 File ID Open
M01057 FV001 File ID Busy
SET I00017 Execute FWRITE3b
<resource chk>Open/busy
00045
I00017 Execute FWRITE3b
MOV #TOUT_A D00101 0 Parameter table+0
Set parameter
00046
MOV D00002 FOPEN3 status D00101 1 Parameter table+0
00047
L MOV #KWSIZ1L D00101 2 Parameter table+0
00048
L MOV 1 D00101 4 Parameter table+0
00049
I00017 Execute FWRITE3b
C FWRITE D00003 FWRITE3 status+0 D00101 Parameter table+0 #KEYWD1
I00018 FWRITE3b execution completed
[FWRITE]
00050
I00018 FWRITE3b execution completed
RST I00016 FWRITE3b resource check
Check status
00051
RST I00017 Execute FWRITE3b
00052
D00003 FWRITE3 status+0 = 0
SET I00021 FWRITE3c resource check
->OK
00053
D00003 FWRITE3 status+0 <> 0
SET I00068 FWRITE3b failed
->NG
00054
SET I00026 FCLOSE3 resource check

FC0324.VSD

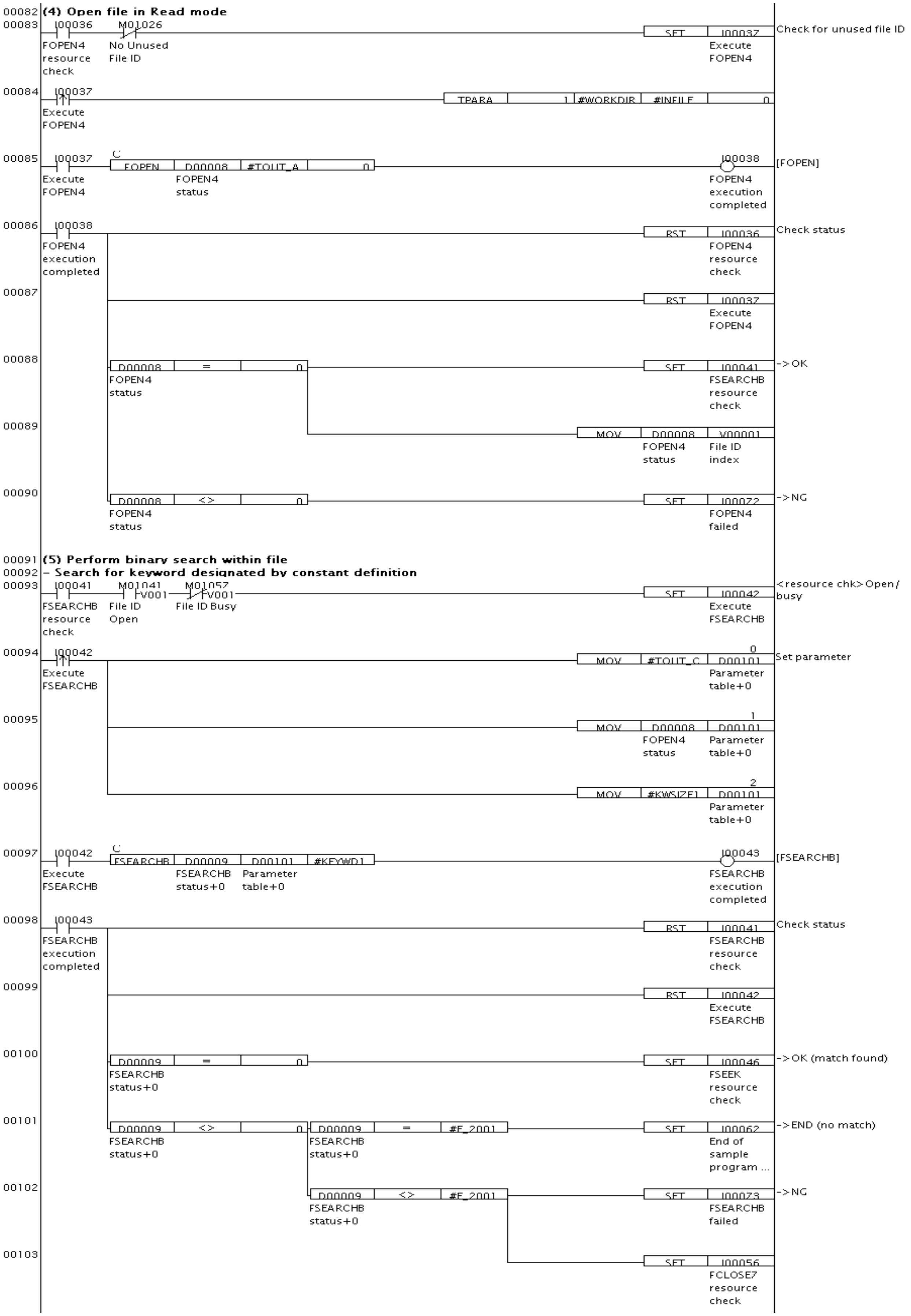
Figure C3.6.12 Binary File Access Sample Program Listing: MAIN (2/6)



00055 I00021 FWRITE3c resource check; M01041 FV001 File ID Open; M01057 FV001 File ID Busy; SET I00022 Execute FWRITE3c; <resource chk> Open/ busy
00056 I00022 Execute FWRITE3c; MOV #TOUT_A D00101 Parameter table+0 (0); Set parameter
00057 MOV D00002 FOPEN3 status D00101 Parameter table+0 (1)
00058 L MOV 2 D00101 Parameter table+0 (2)
00059 L MOV 1000 D00101 Parameter table+0 (4)
00060 I00022 Execute FWRITE3c; C FWRITE D00003 FWRITE3 status+0 D00101 Parameter table+0 D01001; I00023 FWRITE3c execution completed; [FWRITE]
00061 I00023 FWRITE3c execution completed; RST I00021 FWRITE3c resource check; Check status
00062 RST I00022 Execute FWRITE3c
00063 D00003 = 0 FWRITE3 status+0; SET I00026 FCLOSE3 resource check; ->OK
00064 D00003 <> 0 FWRITE3 status+0; SET I00069 FWRITE3c failed; ->NG
00065 SET I00026 FCLOSE3 resource check
00066 I00026 FCLOSE3 resource check; M01041 FV001 File ID Open; M01057 FV001 File ID Busy; SET I00027 Execute FCLOSE3; <resource chk> Open/ busy
00067 I00027 Execute FCLOSE3; C FCLOSE D00006 FCLOSE3 status #TOUT_A D00002 FOPEN3 status; I00028 FCLOSE3 execution completed; [FCLOSE]
00068 I00028 FCLOSE3 execution completed; RST I00026 FCLOSE3 resource check; Check status
00069 RST I00027 Execute FCLOSE3
00070 D00006 = 0 FCLOSE3 status; I00067 FWRITE3a failed; I00068 FWRITE3b failed; I00069 FWRITE3c failed; SET I00036 FOPEN4 resource check; ->OK
00071 D00006 <> 0 FCLOSE3 status; SET I00070 FCLOSE3 failed; ->NG
00072 (3-2) Copy search target file to RAM disk.
00073 - Copy "\CARD1\TARGET.BIN" to "\RAMDISK\TARGET.BIN"
00074 I00031 FCOPY resource check; M01025 File/Disk operation group busy; SET I00032 Execute FCOPY; <resource chk> F/C busy
00075 I00032 Execute FCOPY; TPARA 1 #INDIR #INFILE 0; Set parameter
00076 TPARA 2 #WORKDIR #INFILE 0
00077 I00032 Execute FCOPY; C FCOPY D00007 FCOPY status #TOUT_C 1; I00033 FCOPY execution completed; [FCOPY]
00078 I00033 FCOPY execution completed; RST I00031 FCOPY resource check; Check status
00079 RST I00032 Execute FCOPY
00080 D00007 = 0 FCOPY status; SET I00036 FOPEN4 resource check; ->OK
00081 D00007 <> 0 FCOPY status; SET I00071 FCOPY failed; ->NG

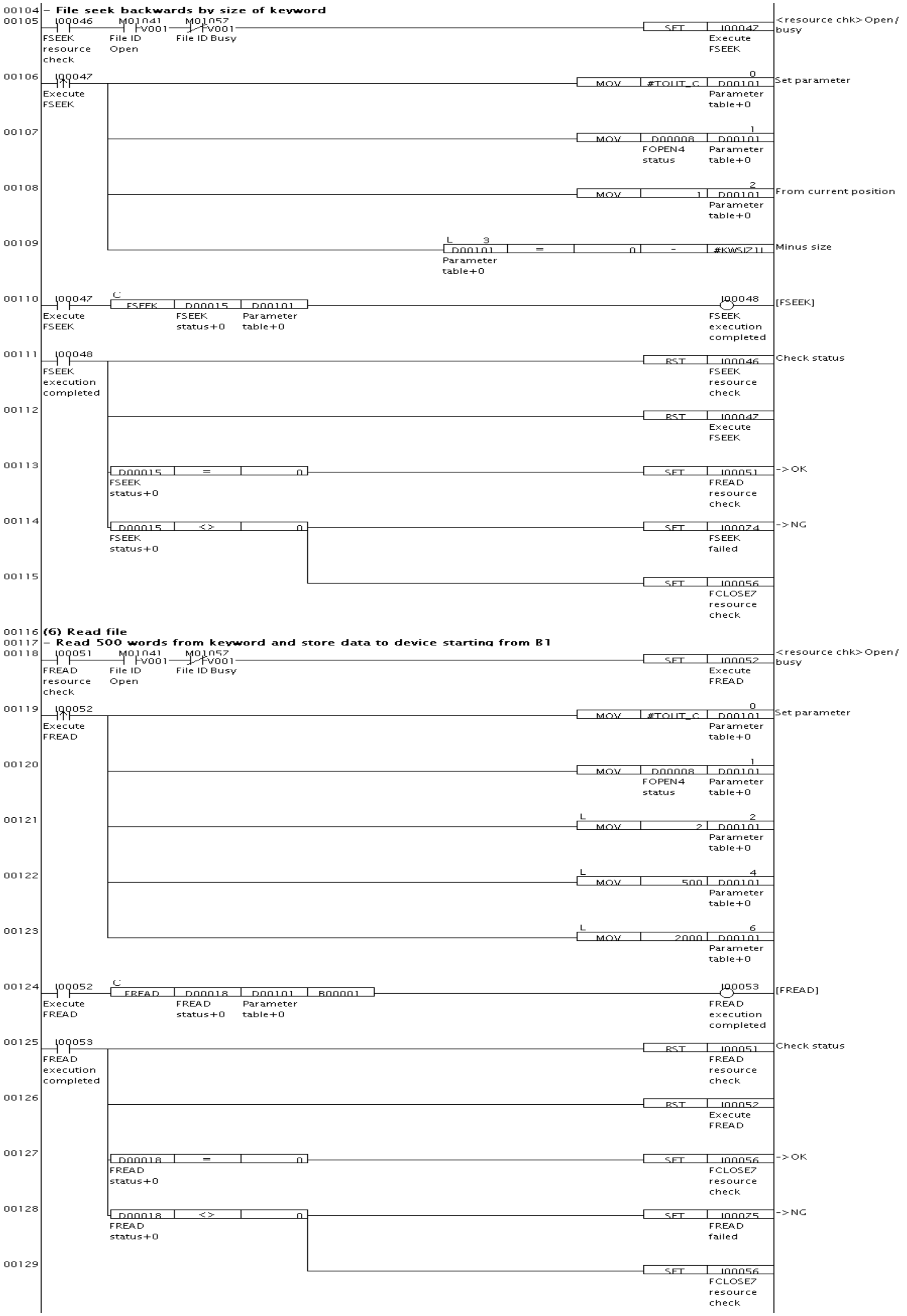
FC0325.VSD

Figure C3.6.13 Binary File Access Sample Program Listing: MAIN (3/6)


00082 (4) Open file in Read mode
00083 I00036 M01026 SET I00037 Check for unused file ID
FOPEN4 resource check
No Unused File ID
Execute FOPEN4
00084 I00037 TPARA 1 #WORKDIR #INFILE 0
Execute FOPEN4
00085 I00037 C FOPEN D00008 #TOUT_A 0 I00038 [FOPEN]
Execute FOPEN4
FOPEN4 status
FOPEN4 execution completed
00086 I00038 RST I00036 Check status
FOPEN4 execution completed
FOPEN4 resource check
00087 RST I00037
Execute FOPEN4
00088 D00008 = 0 SET I00041 ->OK
FOPEN4 status
FSEARCHB resource check
00089 MOV D00008 V00001
FOPEN4 status
File ID index
00090 D00008 <> 0 SET I00072 ->NG
FOPEN4 status
FOPEN4 failed
00091 (5) Perform binary search within file
00092 - Search for keyword designated by constant definition
00093 I00041 M01041 FV001 M01057 FV001 SET I00042 <resource chk>Open/busy
FSEARCHB resource check
File ID Open
File ID Busy
Execute FSEARCHB
00094 I00042 MOV #TOUT_C D00101 0 Set parameter
Execute FSEARCHB
Parameter table+0
00095 MOV D00008 D00101 1
FOPEN4 status
Parameter table+0
00096 MOV #KWSIZE1 D00101 2
Parameter table+0
00097 I00042 C FSEARCHB D00009 D00101 #KEYWD1 I00043 [FSEARCHB]
Execute FSEARCHB
FSEARCHB status+0
Parameter table+0
FSEARCHB execution completed
00098 I00043 RST I00041 Check status
FSEARCHB execution completed
FSEARCHB resource check
00099 RST I00042
Execute FSEARCHB
00100 D00009 = 0 SET I00046 ->OK (match found)
FSEARCHB status+0
FSEEK resource check
00101 D00009 <> 0 D00009 = #E_2001 SET I00062 ->END (no match)
FSEARCHB status+0
FSEARCHB status+0
End of sample program ...
00102 D00009 <> #E_2001 SET I00073 ->NG
FSEARCHB status+0
FSEARCHB failed
00103 SET I00056
FCLOSE7 resource check

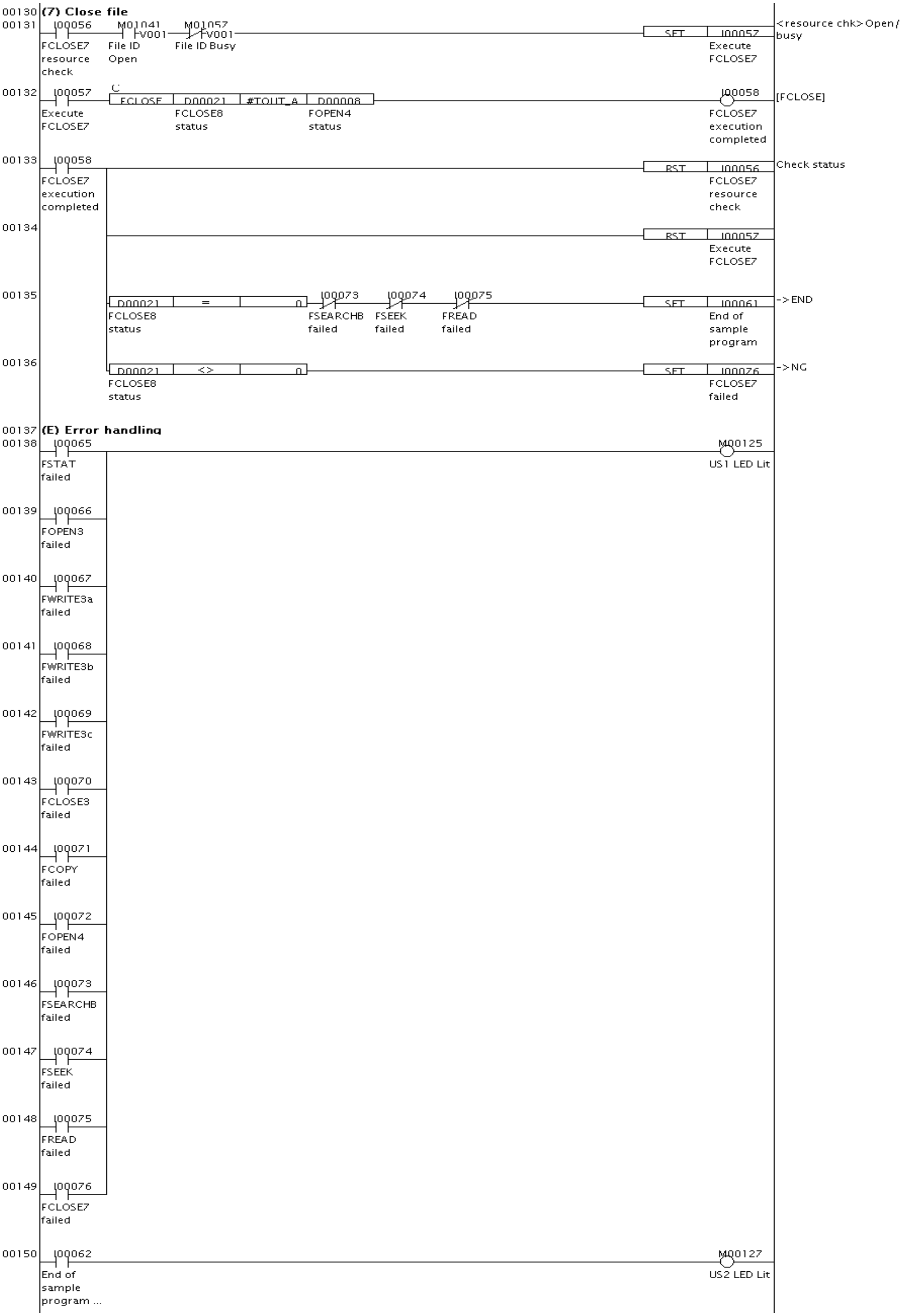
FC0326.VSD

Figure C3.6.14 Binary File Access Sample Program Listing: MAIN (4/6)


00104 – File seek backwards by size of keyword
00105 I00046 FSEEK resource check
M01041 FV001 File ID Open
M01057 FV001 File ID Busy
SET I00047 Execute FSEEK
<resource chk> Open/ busy
00106 I00047 Execute FSEEK
MOV #TOUT_C D00101 0 Parameter table+0
Set parameter
00107 MOV D00008 FOPEN4 status D00101 1 Parameter table+0
00108 MOV 1 D00101 2 Parameter table+0
From current position
00109 L D00101 3 Parameter table+0 = 0 - #KWSIZ1L
Minus size
00110 I00047 Execute FSEEK
C FSEEK D00015 FSEEK status+0 D00101 Parameter table+0
I00048 FSEEK execution completed
[FSEEK]
00111 I00048 FSEEK execution completed
RST I00046 FSEEK resource check
Check status
00112 RST I00047 Execute FSEEK
00113 D00015 FSEEK status+0 = 0
SET I00051 FREAD resource check
->OK
00114 D00015 FSEEK status+0 <> 0
SET I00074 FSEEK failed
->NG
00115 SET I00056 FCLOSE7 resource check
00116 (6) Read file
00117 – Read 500 words from keyword and store data to device starting from B1
00118 I00051 FREAD resource check
M01041 FV001 File ID Open
M01057 FV001 File ID Busy
SET I00052 Execute FREAD
<resource chk> Open/ busy
00119 I00052 Execute FREAD
MOV #TOUT_C D00101 0 Parameter table+0
Set parameter
00120 MOV D00008 FOPEN4 status D00101 1 Parameter table+0
00121 L MOV 2 D00101 2 Parameter table+0
00122 L MOV 500 D00101 4 Parameter table+0
00123 L MOV 2000 D00101 6 Parameter table+0
00124 I00052 Execute FREAD
C FREAD D00018 FREAD status+0 D00101 Parameter table+0 B00001
I00053 FREAD execution completed
[FREAD]
00125 I00053 FREAD execution completed
RST I00051 FREAD resource check
Check status
00126 RST I00052 Execute FREAD
00127 D00018 FREAD status+0 = 0
SET I00056 FCLOSE7 resource check
->OK
00128 D00018 FREAD status+0 <> 0
SET I00075 FREAD failed
->NG
00129 SET I00056 FCLOSE7 resource check

FC0327.VSD

Figure C3.6.15 Binary File Access Sample Program Listing: MAIN (5/6)


00130 (7) Close file
00131 I00056 FCLOSE7 resource check
M01041 FV001 File ID Open
M01057 FV001 File ID Busy
SET I00057 Execute FCLOSE7
<resource chk> Open/ busy
00132 I00057 Execute FCLOSE7
C FCLOSE D00021 FCLOSE8 status #TOUT_A D00008 FOPEN4 status
I00058 FCLOSE7 execution completed
[FCLOSE]
00133 I00058 FCLOSE7 execution completed
RST I00056 FCLOSE7 resource check
Check status
00134 RST I00057 Execute FCLOSE7
00135 D00021 FCLOSE8 status = 0
I00073 FSEARCHB failed
I00074 FSEEK failed
I00075 FREAD failed
SET I00061 End of sample program
->END
00136 D00021 FCLOSE8 status <> 0
SET I00076 FCLOSE7 failed
->NG
00137 (E) Error handling
00138 I00065 FSTAT failed
M00125 US1 LED Lit
00139 I00066 FOPEN3 failed
00140 I00067 FWRITE3a failed
00141 I00068 FWRITE3b failed
00142 I00069 FWRITE3c failed
00143 I00070 FCLOSE3 failed
00144 I00071 FCOPY failed
00145 I00072 FOPEN4 failed
00146 I00073 FSEARCHB failed
00147 I00074 FSEEK failed
00148 I00075 FREAD failed
00149 I00076 FCLOSE7 failed
00150 I00062 End of sample program ...
M00127 US2 LED Lit

FC0328.VSD

Figure C3.6.16 Binary File Access Sample Program Listing: MAIN (6/6)

C3.6.4 CSV File Conversion from and to Device

■ Function and Usage

This is the sample program for the Convert CSV File to Device instruction and the Convert Device to CSV File instruction. A F3SP66-4S and an SD memory card are required to run the sample program. Before running the sample program, you should store the "DEC.CSV" file in the root directory of the SD memory card.

The sample program performs the following processing:

1. Initializes devices.
2. Opens the CSV formatted file named "DEC.CSV" (referred to as the input file below) in read-only mode.
3. Converts all CSV data fields stored in the input file in decimal representation to word data, and stores the converted data to device, starting from B1.
4. Closes the input file.
5. Opens the output file named "HEX.CSV" in write mode.
6. Converts word data stored in device starting from B1 to CSV data fields in hexadecimal representation and writes the result to the output file. The number of converted fields is determined by the number of fields read in step 2.
7. Closes the output file.

■ Structure of Sample Program

● List of Instructions Used

The table below shows the main instructions used in the sample program.

Table C3.6.14 List of File Access Instructions Used

Ladder Instruction Mnemonic	Usage
FOPEN	Opens the input file and output file.
FCLOSE	Closes the input file and output file.
F2DCSV	Reads the CSV formatted input file, and converts the data to word data.
D2FCSV	Converts word data stored in device to CSV format and writes the result to the output file.

● List of Special Relays Used

The table below lists the main special relays used in the sample program.

Table C3.6.15 List of Special Relays Used

Name of Special Relay	No. of Special Relay	Function
No Unused File ID	M1026	Checks for unused field ID.
File ID Open	M1041 to M1056	Checks whether file ID to be used is already open.
File ID Busy	M1057 to M1072	Checks whether a file ID to be used is being used by another instruction.
Always ON	M0033	Used for Always on circuit
1 Scan ON at Program Start	M0035	Used in circuit to turn on for one scan after program starts execution
US1 LED Lit	M0125	Turns on US1 LED

Project

The table below shows the content of the WideField2 project containing the sample program.

Table C3.6.16 Project Content

Name	Component	Description	
F2D2FCSV	Configuration	SP66 configuration with default setup. You can also F3SP67-6S provided you change the CPU type in the configuration.	
	Blocks	Total No. of blocks	1
		Block 1	MAIN
	Macros	Total no. of macros	0
	Constant definition	#INDIR	Directory where the input file is stored.
		#INFILE	Input file
		#OUTDIR	Directory for storing the output file
		#OUTFILE	Output file
		Others, 7 definitions in total.	

CPU Properties

This sample program does not define any CPU property. Any CPU property file can be used.

Files

The table below lists the files used in the sample program.

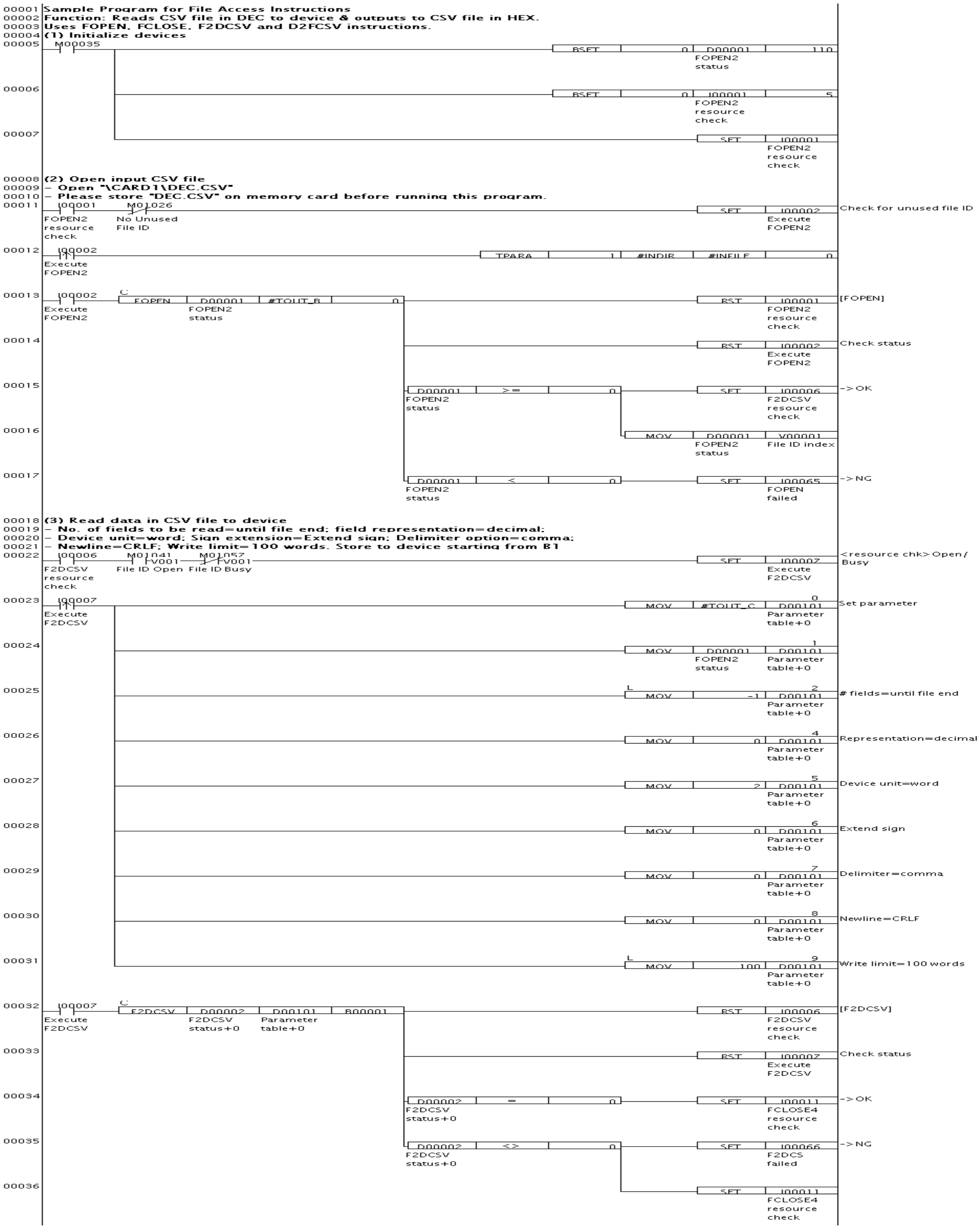
Table C3.6.17 List of Files Used

File Name	Input/Output	Description
DEC.CSV	Input	This is a CSV formatted file containing data in decimal representation. Before running the sample program, store this file in the root directory of the SD memory card.
HEX.CSV	Output	This is a CSV formatted file containing data in hexadecimal representation, to be created by the sample program.

■ Ladder Program Listing

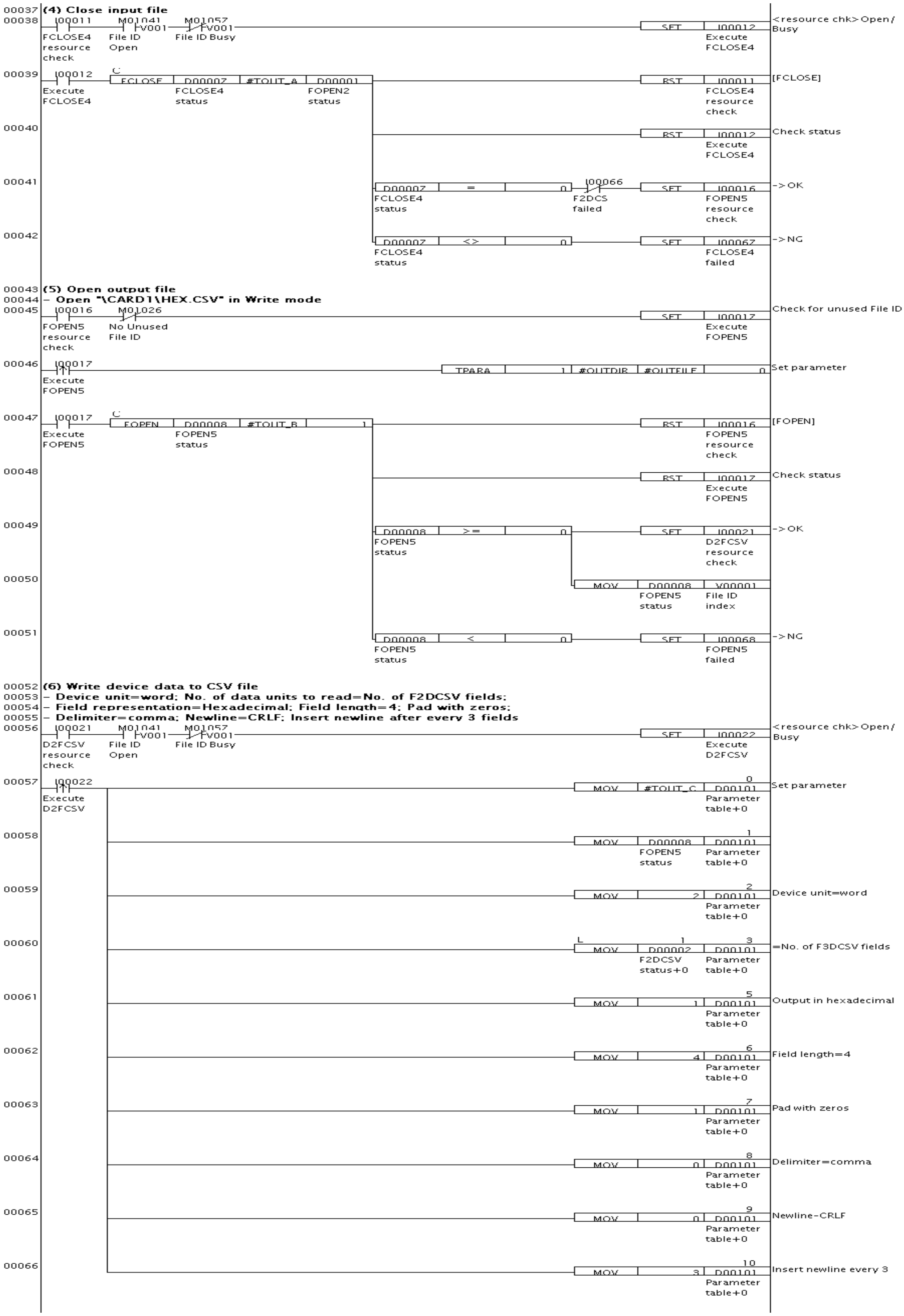
The figure on the following pages shows the ladder program listing. For details on the purpose of individual devices used in the ladder program, see the block tag names and I/O comments of each block.

● Project (F2D2FCSV) Block (MAIN)


00001 Sample Program for File Access Instructions
00002 Function: Reads CSV file in DEC to device & outputs to CSV file in HEX.
00003 Uses FOPEN, FCLOSE, F2DCSV and D2FCSV instructions.
00004 (1) Initialize devices
00005 M00035 BSET 0 D00001 110 FOPEN2 status
00006 BSET 0 I00001 5 FOPEN2 resource check
00007 SET I00001 FOPEN2 resource check
00008 (2) Open input CSV file
00009 – Open "\CARD1\DEC.CSV"
00010 – Please store "DEC.CSV" on memory card before running this program.
00011 I00001 FOPEN2 resource check M01026 No Unused File ID SET I00002 Execute FOPEN2 Check for unused file ID
00012 I00002 Execute FOPEN2 TPARA 1 #INDIR #INFILE 0
00013 I00002 Execute FOPEN2 C FOPEN D00001 FOPEN2 status #TOUT_B 0 RST I00001 FOPEN2 resource check [FOPEN]
00014 RST I00002 Execute FOPEN2 Check status
00015 D00001 FOPEN2 status >= 0 SET I00006 F2DCSV resource check ->OK
00016 MOV D00001 FOPEN2 status V00001 File ID index
00017 D00001 FOPEN2 status < 0 SET I00065 FOPEN failed ->NG
00018 (3) Read data in CSV file to device
00019 – No. of fields to be read=until file end; field representation=decimal;
00020 – Device unit=word; Sign extension=Extend sign; Delimiter option=comma;
00021 – Newline=CRLF; Write limit=100 words. Store to device starting from B1
00022 I00006 F2DCSV resource check M01041 V001 File ID Open M01057 V001 File ID Busy SET I00007 Execute F2DCSV <resource chk> Open/ Busy
00023 I00007 Execute F2DCSV MOV #TOUT_C 0 D00101 Parameter table+0 Set parameter
00024 MOV D00001 FOPEN2 status 1 D00101 Parameter table+0
00025 L MOV -1 2 D00101 Parameter table+0 # fields=until file end
00026 MOV 0 4 D00101 Parameter table+0 Representation=decimal
00027 MOV 2 5 D00101 Parameter table+0 Device unit=word
00028 MOV 0 6 D00101 Parameter table+0 Extend sign
00029 MOV 0 7 D00101 Parameter table+0 Delimiter=comma
00030 MOV 0 8 D00101 Parameter table+0 Newline=CRLF
00031 L MOV 100 9 D00101 Parameter table+0 Write limit=100 words
00032 I00007 Execute F2DCSV C F2DCSV D00002 F2DCSV status+0 D00101 Parameter table+0 B00001 RST I00006 F2DCSV resource check [F2DCSV]
00033 RST I00007 Execute F2DCSV Check status
00034 D00002 F2DCSV status+0 = 0 SET I00011 FCLOSE4 resource check ->OK
00035 D00002 F2DCSV status+0 <> 0 SET I00066 F2DCS failed ->NG
00036 SET I00011 FCLOSE4 resource check

FC0329.VSD

Figure C3.6.17 CSV File Conversion Sample Program Listing: MAIN (1/3)


00037 (4) Close input file
00038 I00011 FCLOSE4 resource check; M01041 V001 File ID Open; M01057 V001 File ID Busy; SET I00012 Execute FCLOSE4
<resource chk> Open/ Busy
00039 I00012 Execute FCLOSE4; C FCLOSE D00007 FCLOSE4 status #TOUT_A D00001 FOPEN2 status; RST I00011 FCLOSE4 resource check
[FCLOSE]
00040 RST I00012 Execute FCLOSE4
Check status
00041 D00007 FCLOSE4 status = 0; I00066 F2DCS failed; SET I00016 FOPEN5 resource check
-> OK
00042 D00007 FCLOSE4 status <> 0; SET I00067 FCLOSE4 failed
-> NG
00043 (5) Open output file
00044 – Open "\CARD1\HEX.CSV" in Write mode
00045 I00016 FOPEN5 resource check; M01026 No Unused File ID; SET I00017 Execute FOPEN5
Check for unused File ID
00046 I00017 Execute FOPEN5; TPARA 1 #OUTDIR #OUTFILE 0
Set parameter
00047 I00017 Execute FOPEN5; C FOPEN D00008 FOPEN5 status #TOUT_B 1; RST I00016 FOPEN5 resource check
[FOPEN]
00048 RST I00017 Execute FOPEN5
Check status
00049 D00008 FOPEN5 status >= 0; SET I00021 D2FCSV resource check
-> OK
00050 MOV D00008 FOPEN5 status V00001 File ID index
00051 D00008 FOPEN5 status < 0; SET I00068 FOPEN5 failed
-> NG
00052 (6) Write device data to CSV file
00053 – Device unit=word; No. of data units to read=No. of F2DCSV fields;
00054 – Field representation=Hexadecimal; Field length=4; Pad with zeros;
00055 – Delimiter=comma; Newline=CRLF; Insert newline after every 3 fields
00056 I00021 D2FCSV resource check; M01041 V001 File ID Open; M01057 V001 File ID Busy; SET I00022 Execute D2FCSV
<resource chk> Open/ Busy
00057 I00022 Execute D2FCSV; MOV #TOUT_C 0 D00101 Parameter table+0
Set parameter
00058 MOV D00008 FOPEN5 status 1 D00101 Parameter table+0
00059 MOV 2 2 D00101 Parameter table+0
Device unit=word
00060 L MOV 1 D00002 F2DCSV status+0 3 D00101 Parameter table+0
=No. of F3DCSV fields
00061 MOV 1 5 D00101 Parameter table+0
Output in hexadecimal
00062 MOV 4 6 D00101 Parameter table+0
Field length=4
00063 MOV 1 7 D00101 Parameter table+0
Pad with zeros
00064 MOV 0 8 D00101 Parameter table+0
Delimiter=comma
00065 MOV 0 9 D00101 Parameter table+0
Newline-CRLF
00066 MOV 3 10 D00101 Parameter table+0
Insert newline every 3

FC0330.VSD

Figure C3.6.18 CSV File Conversion Sample Program Listing: MAIN (2/3)

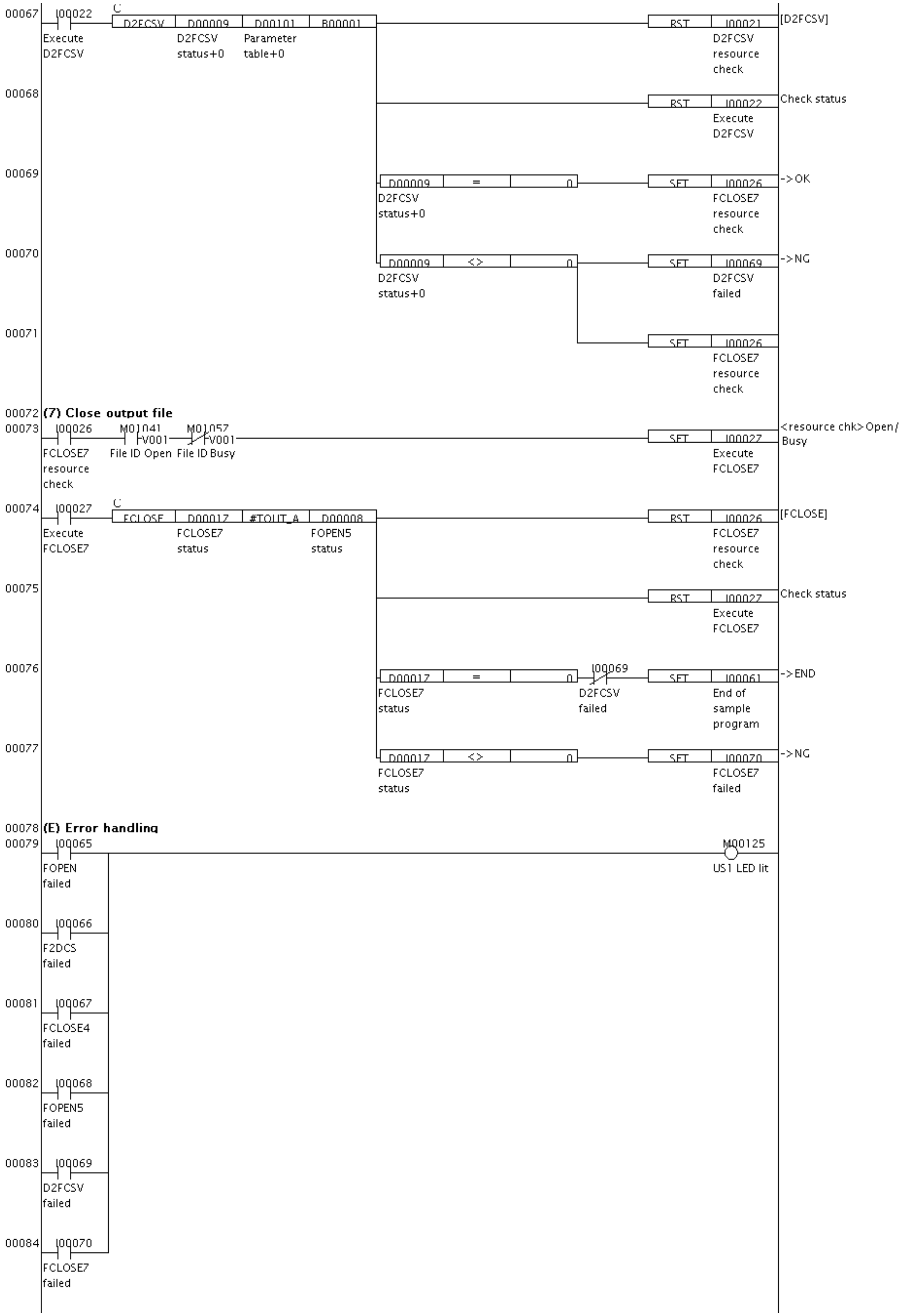


Figure C3.6.19 CSV File Conversion Sample Program Listing: MAIN (3/3)

C3.6.5 Binary File Conversion from and to Device

■ Function and Usage

This is the sample program for the Convert Binary File to Device instruction and the Convert Device to Binary File instruction. A F3SP66-4S and an SD memory card are required to run the sample program. Before running the sample program, you should store the "BYTE.BIN" file in the root directory of the SD memory card.

The sample program performs the following processing:

1. Initializes devices.
2. Opens the binary file named "BYTE.BIN" (referred to as input file below) in read-only mode.
3. Reads the input file in byte units, and stores the data as sign extended words to devices, starting from B1.
4. Closes the input file.
5. Opens the output file named "LONG.BIN" in write mode.
6. Converts word data stored in device starting from B1 to sign extended long word data and writes the result to the output file. The number of written long words is determined by the number of bytes read in step 2.
7. Closes the output file.

■ Structure of Sample Program

● List of Instructions Used

The table below shows the main instructions used in the sample program.

Table C3.6.18 List of File Access Instructions Used

Ladder Instruction Mnemonic	Usage
FOPEN	Opens the input file and output file.
FCLOSE	Closes the input file and output file.
F2DBIN	Reads the input file in byte units, converts the data to sign extended word data and stores the data to device.
D2FBIN	Reads word data stored in device, converts the data to sign extended long word data and writes the data to the output file.

● List of Special Relays Used

The table below lists the main special relays used in the sample program.

Table C3.6.19 List of Special Relays Used

Name of Special Relay	No. of Special Relay	Function
No Unused File ID	M1026	Checks for unused field ID.
File ID Open	M1041 to M1056	Checks whether file ID to be used is already open.
File ID Busy	M1057 to M1072	Checks whether a file ID to be used is being used by another instruction.
Always ON	M0033	Used for Always on circuit
1 Scan ON at Program Start	M0035	Used in circuit to turn on for one scan after program starts execution
US1 LED Lit	M0125	Turns on US1 LED

Project

The table below shows the content of the WideField2 project containing the sample program.

Table C3.6.20 Project Content

Name	Component	Description	
F2D2FBIN	Configuration	SP66 configuration with default setup. You can also F3SP67-6S provided you change the CPU type in the configuration.	
	Blocks	Total No. of blocks	1
		Block 1	MAIN
	Macros	Total no. of macros	0
	Constant definition	#INDIR	Directory where the input file is stored.
		#INFILE	Input file
		#OUTDIR	Directory for storing the output file
		#OUTFILE	Output file
		Others, 7 definitions in total.	

CPU Properties

This sample program does not define any CPU property. Any CPU property file can be used.

Files

The table below lists the files used in the sample program.

Table C3.6.21 List of Files Used

File Name	Input/Output	Description
BYTE.BIN	Input	This is a binary file. Before running the sample program, store this file in the root directory of the SD memory card.
LONG.BIN	Output	This is a binary file, to be created by the sample program.

■ Ladder Program Listing

The figure on the following pages shows the ladder program listing. For details on the purpose of individual devices used in the ladder program, see the block tag names and I/O comments of each block.

● Project (F2D2FBIN) Block (MAIN)

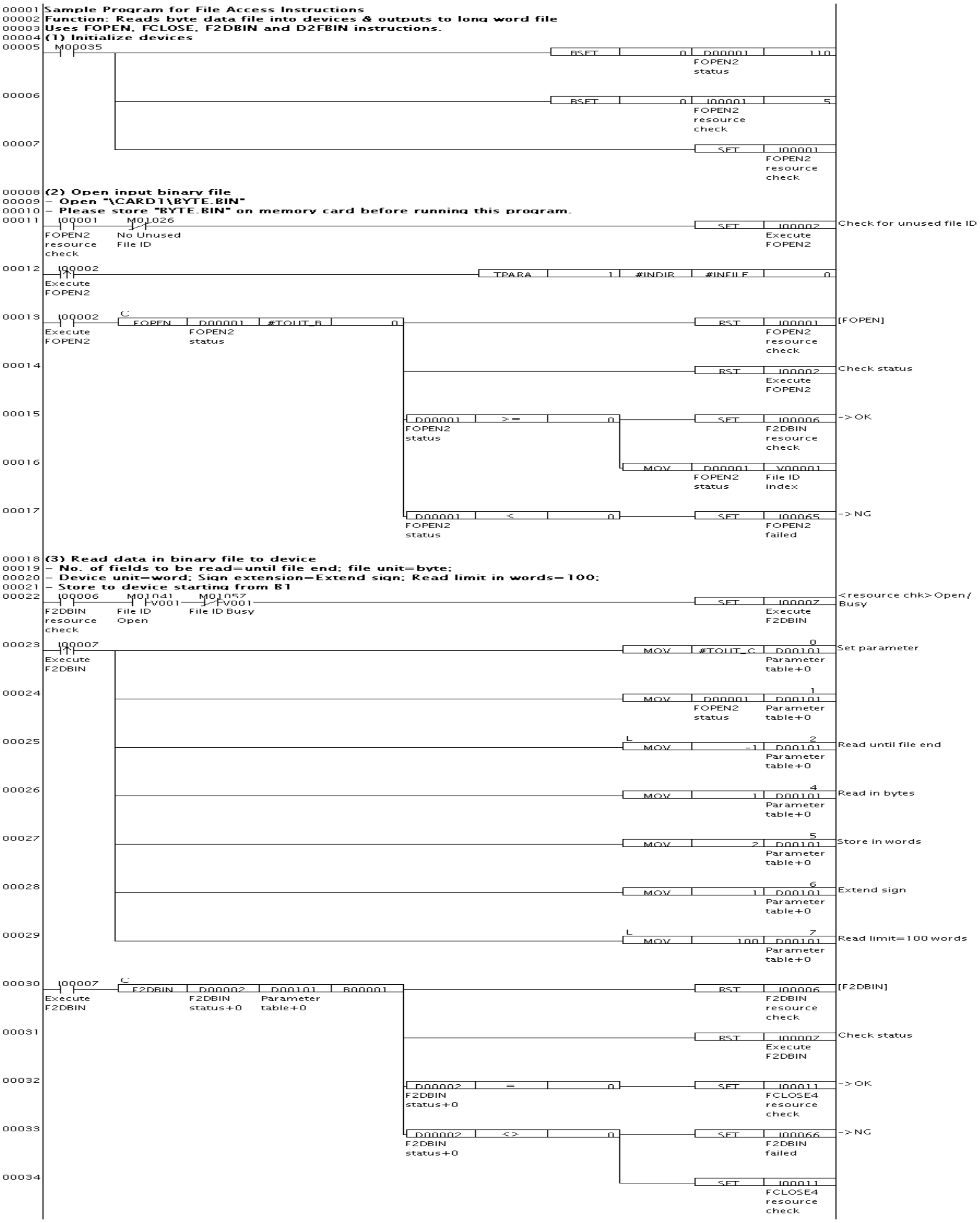
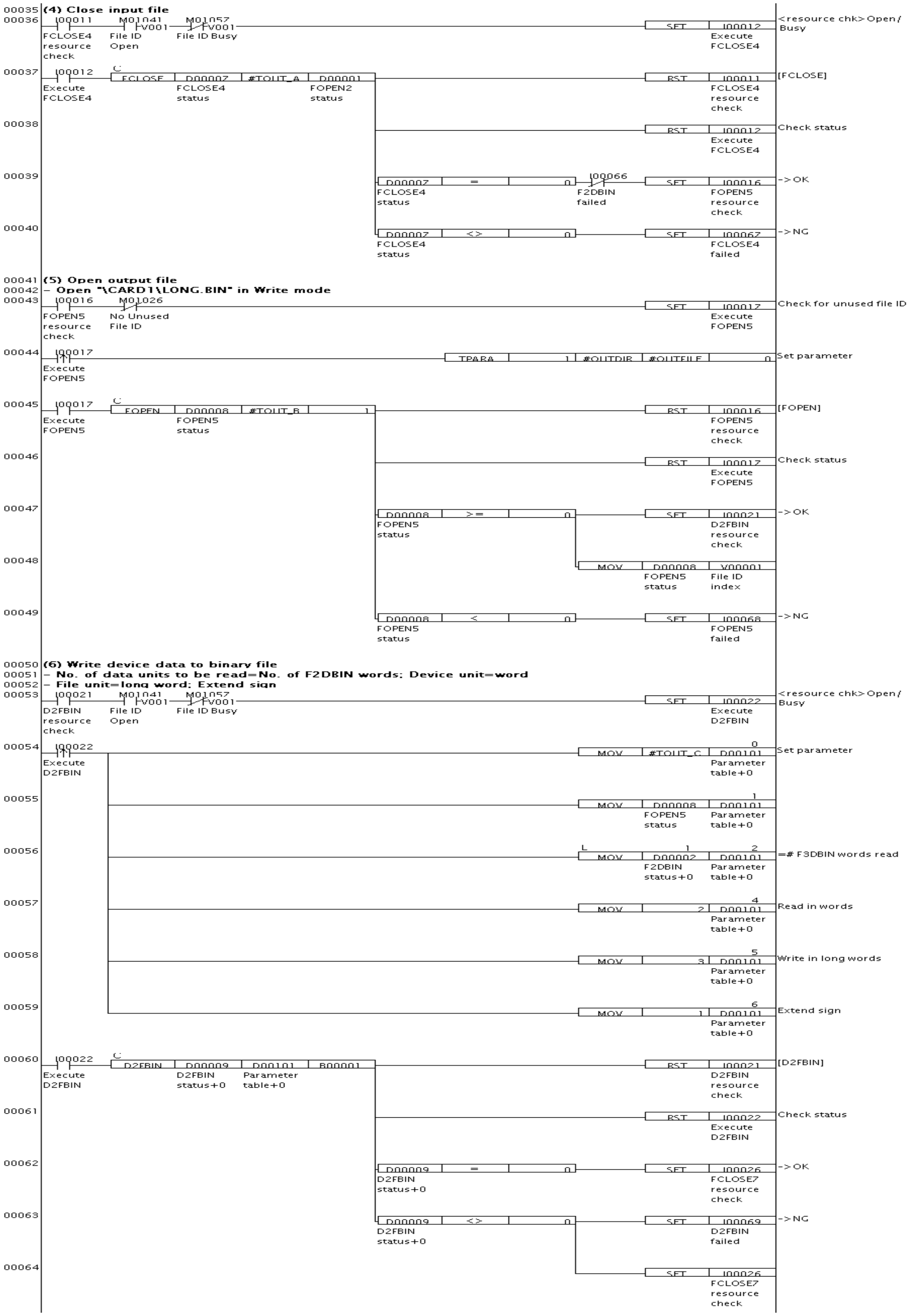

00001 Sample Program for File Access Instructions
00002 Function: Reads byte data file into devices & outputs to long word file
00003 Uses FOPEN, FCLOSE, F2DBIN and D2FBIN instructions.
00004 (1) Initialize devices
00005 M00035 BSET 0 D00001 110 FOPEN2 status
00006 BSET 0 I00001 5 FOPEN2 resource check
00007 SET I00001 FOPEN2 resource check
00008 (2) Open input binary file
00009 – Open "\CARD1\BYTE.BIN"
00010 – Please store "BYTE.BIN" on memory card before running this program.
00011 I00001 FOPEN2 resource check M01026 No Unused File ID SET I00002 Execute FOPEN2 Check for unused file ID
00012 I00002 Execute FOPEN2 TPARA 1 #INDIR #INFILE 0
00013 I00002 Execute FOPEN2 C FOPEN D00001 FOPEN2 status #TOUT_B 0 RST I00001 FOPEN2 resource check [FOPEN]
00014 RST I00002 Execute FOPEN2 Check status
00015 D00001 FOPEN2 status >= 0 SET I00006 F2DBIN resource check ->OK
00016 MOV D00001 FOPEN2 status V00001 File ID index
00017 D00001 FOPEN2 status < 0 SET I00065 FOPEN2 failed ->NG
00018 (3) Read data in binary file to device
00019 – No. of fields to be read=until file end; file unit=byte;
00020 – Device unit=word; Sign extension=Extend sign; Read limit in words=100;
00021 – Store to device starting from B1
00022 I00006 F2DBIN resource check M01041 V001 File ID Open M01057 V001 File ID Busy SET I00007 Execute F2DBIN <resource chk>Open/ Busy
00023 I00007 Execute F2DBIN MOV #TOUT_C 0 D00101 Parameter table+0 Set parameter
00024 MOV D00001 FOPEN2 status 1 D00101 Parameter table+0
00025 L MOV -1 2 D00101 Parameter table+0 Read until file end
00026 MOV 1 4 D00101 Parameter table+0 Read in bytes
00027 MOV 2 5 D00101 Parameter table+0 Store in words
00028 MOV 1 6 D00101 Parameter table+0 Extend sign
00029 L MOV 100 7 D00101 Parameter table+0 Read limit=100 words
00030 I00007 Execute F2DBIN C F2DBIN D00002 F2DBIN status+0 D00101 Parameter table+0 B00001 RST I00006 F2DBIN resource check [F2DBIN]
00031 RST I00007 Execute F2DBIN Check status
00032 D00002 F2DBIN status+0 = 0 SET I00011 FCLOSE4 resource check ->OK
00033 D00002 F2DBIN status+0 <> 0 SET I00066 F2DBIN failed ->NG
00034 SET I00011 FCLOSE4 resource check
FC0332.VSD

Figure C3.6.20 Binary File Conversion Sample Program Listing: MAIN (1/3)



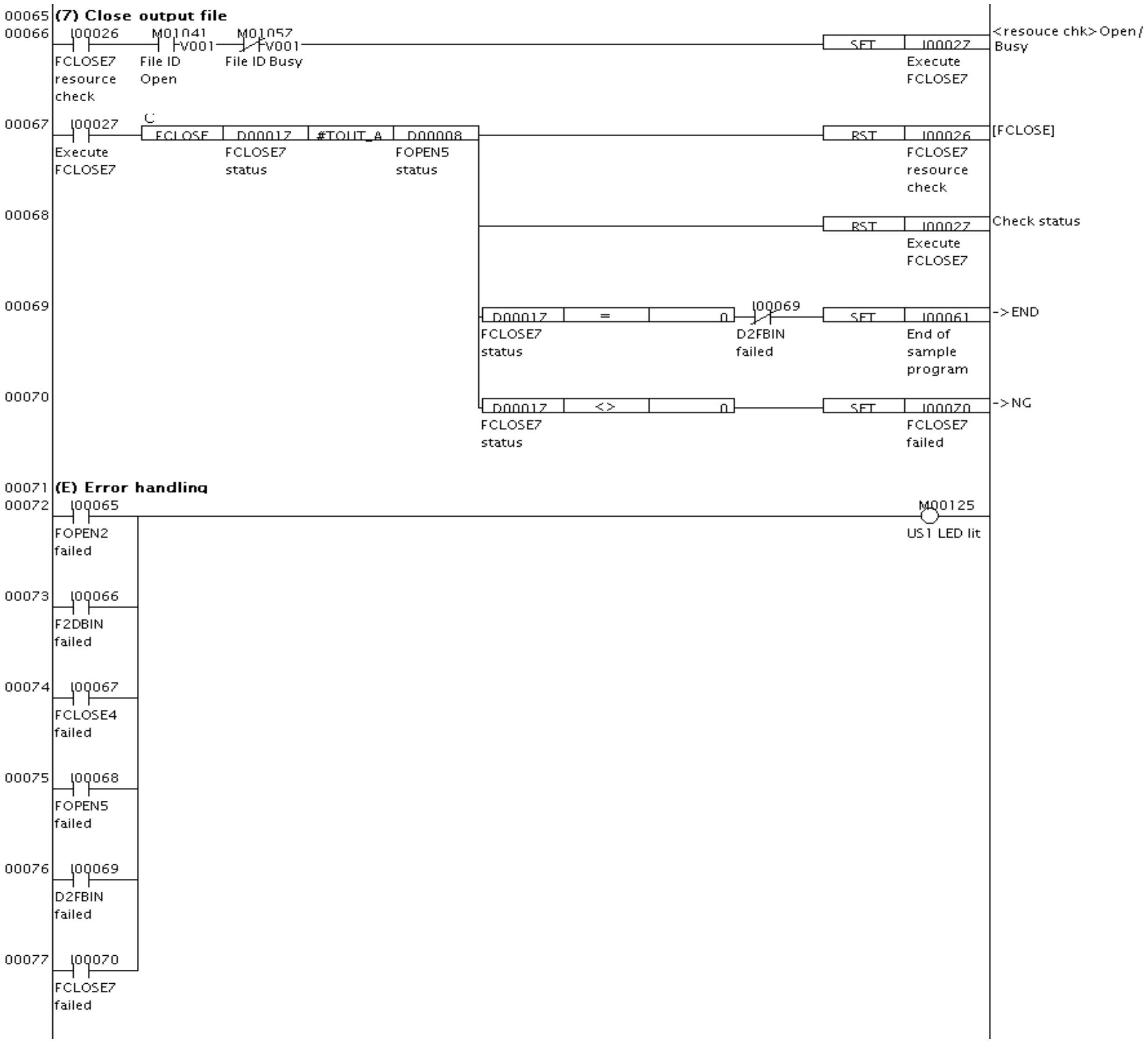
```
00035 (4) Close input file
00036  I00011          M01041          M01057                          SET  I00012        <resource chk>Open/
       FCLOSE4 resource check  File ID Open  V001  File ID Busy  V001       Execute FCLOSE4    Busy
00037  I00012  C FCLOSE  D00007  #TOUT_A  D00001                      RST  I00011        [FCLOSE]
       Execute FCLOSE4   FCLOSE4 status   FOPEN2 status                     FCLOSE4 resource check
00038                                                                  RST  I00012        Check status
                                                                            Execute FCLOSE4
00039                  D00007  =  0   I00066                           SET  I00016        ->OK
                       FCLOSE4 status  F2DBIN failed                        FOPEN5 resource check
00040                  D00007  <>  0                                   SET  I00067        ->NG
                       FCLOSE4 status                                       FCLOSE4 failed

00041 (5) Open output file
00042 - Open "\CARD1\LONG.BIN" in Write mode
00043  I00016          M01026                                          SET  I00017        Check for unused file ID
       FOPEN5 resource check  No Unused File ID                             Execute FOPEN5
00044  I00017          TPARA  1  #OUTDIR  #OUTFILE  0                                     Set parameter
       Execute FOPEN5
00045  I00017  C FOPEN  D00008  #TOUT_B  1                             RST  I00016        [FOPEN]
       Execute FOPEN5   FOPEN5 status                                       FOPEN5 resource check
00046                                                                  RST  I00017        Check status
                                                                            Execute FOPEN5
00047                  D00008  >=  0                                   SET  I00021        ->OK
                       FOPEN5 status                                        D2FBIN resource check
00048                                                   MOV  D00008  V00001
                                                             FOPEN5 status  File ID index
00049                  D00008  <  0                                    SET  I00068        ->NG
                       FOPEN5 status                                        FOPEN5 failed

00050 (6) Write device data to binary file
00051 - No. of data units to be read=No. of F2DBIN words; Device unit=word
00052 - File unit=long word; Extend sign
00053  I00021          M01041          M01057                          SET  I00022        <resource chk>Open/
       D2FBIN resource check  File ID Open  V001  File ID Busy  V001        Execute D2FBIN     Busy
00054  I00022                                   MOV  #TOUT_C  D00101 0                       Set parameter
       Execute D2FBIN                                     Parameter table+0
00055                                           MOV  D00008  D00101 1
                                                     FOPEN5 status  Parameter table+0
00056                                         L MOV  D00002 1  D00101 2                      =# F3DBIN words read
                                                     F2DBIN status+0  Parameter table+0
00057                                           MOV  2  D00101 4                              Read in words
                                                        Parameter table+0
00058                                           MOV  3  D00101 5                              Write in long words
                                                        Parameter table+0
00059                                           MOV  1  D00101 6                              Extend sign
                                                        Parameter table+0
00060  I00022  C D2FBIN  D00009  D00101  B00001                        RST  I00021        [D2FBIN]
       Execute D2FBIN    D2FBIN status+0  Parameter table+0                 D2FBIN resource check
00061                                                                  RST  I00022        Check status
                                                                            Execute D2FBIN
00062                  D00009  =  0                                    SET  I00026        ->OK
                       D2FBIN status+0                                      FCLOSE7 resource check
00063                  D00009  <>  0                                   SET  I00069        ->NG
                       D2FBIN status+0                                      D2FBIN failed
00064                                                                  SET  I00026
                                                                            FCLOSE7 resource check
```

FC0333.VSD

Figure C3.6.21   Binary File Conversion Sample Program Listing: MAIN (2/3)


00065 (7) Close output file
00066 I00026 FCLOSE7 resource check
M01041 V001 File ID Open
M01057 V001 File ID Busy
SET I00027 Execute FCLOSE7
<resouce chk> Open/ Busy
00067 I00027 Execute FCLOSE7
C FCLOSE D00017 FCLOSE7 status #TOUT_A D00008 FOPEN5 status
RST I00026 FCLOSE7 resource check
[FCLOSE]
00068 RST I00027 Execute FCLOSE7
Check status
00069 D00017 FCLOSE7 status = 0
I00069 D2FBIN failed
SET I00061 End of sample program
->END
00070 D00017 FCLOSE7 status <> 0
SET I00070 FCLOSE7 failed
->NG
00071 (E) Error handling
00072 I00065 FOPEN2 failed
M00125 US1 LED lit
00073 I00066 F2DBIN failed
00074 I00067 FCLOSE4 failed
00075 I00068 FOPEN5 failed
00076 I00069 D2FBIN failed
00077 I00070 FCLOSE7 failed

FC0334.VSD

Figure C3.6.22   Binary File Conversion Sample Program Listing: MAIN (3/3)

FA-M3
Sequence CPU – Functions (for F3SP66-4S, F3SP67-6S)
PART-C　Storage Functions

IM 34M6P14-01E　1st Edition

INDEX

Blank Page

Revision Information

Document Name: Sequence CPU – Functions (for F3SP66-4S, F3SP67-6S)
Document No.: IM 34M6P14-01E

Edition	Date	Revised Item
1st	Jun 2007	New publication

Written by PLC International Sales & Marketing Gr.
PLC Product Marketing Dept.
Industrial Automation Systems Business
Yokogawa Electric Corporation
Published by Yokogawa Electric Corporation
2-9-32 Nakacho, Musashino-shi, Tokyo, 180-8750, JAPAN
Printed by Kohoku Publishing & Printing Inc.

Blank Page