OMRON

Vision Sensor

FH Series

Vision System

Macro Customize Functions Programming Manual

FH-2□□□/FH-2□□□-□□
FH-5□□□/FH-5□□□-□□
FH-L□□□/FH-L□□□-□□

SYSMAC
always in control

Z367-E1-16

NOTE

- All rights reserved.
- No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means, mechanical, electronic, photocopying, recording, or otherwise, without the prior written permission of OMRON.
- No patent liability is assumed with respect to the use of the information contained herein. Moreover, because OMRON is constantly striving to improve its high-quality products, the information contained in this manual is subject to change without notice. Every precaution has been taken in the preparation of this manual. Nevertheless, OMRON assumes no responsibility for errors or omissions.
  Neither is any liability assumed for damages resulting from the use of the information contained in this publication.

Trademarks

- Sysmac and SYSMAC are trademarks or registered trademarks of OMRON Corporation in Japan and other countries for OMRON factory automation products.
- This software is based in part on the work of the Independent JPEG Group.
- This software includes 7-Zip.
  7-Zip is licensed under the terms of the GNU Lesser General Public License. 7-Zip is a trademark of Igor Pavlov. For the GNU Lesser General Public License, see the following website: (www.gnu.org)
- Microsoft, Windows, Windows Vista, Excel, and Visual Basic are either registered trademarks or trademarks of Microsoft Corporation in the United States and other countries.
- Intel, Core and Pentium are trademarks of Intel Corporation in the U.S. and/or other countries.
- EtherCAT® is trademarks or registered trademarks, licensed by Beckhoff Automation GmbH, Germany.
- ODVA, CIP, CompoNet, DeviceNet, and EtherNet/IP are trademarks of ODVA.
- The SD, SDHC, microSD, and microSDHC logos are trademarks of SD-3C, LLC.
- QR Code is a registered trademark of DENSO WAVE INCORPORATED.
- MELSEC is a registered trademarks of Mitsubishi Electric Corporation.

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

Copyrights

Microsoft product screen shots reprinted with permission from Microsoft Corporation.

Introduction

Thank you for purchasing the FH Series.
This manual contains information that is necessary to use the FH Series.
Please read this manual and make sure you understand the functionality and performance of the FH Series before you attempt to use it in a control system.
Keep this manual in a safe place where it will be available for reference during operation.

Intended Audience

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

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

Applicable Products

This manual covers the following products.

- FH-2□□□
- FH-2□□□-□□
- FH-5□□□
- FH-5□□□-□□
- FH-L□□□
- FH-L□□□-□□

Part of the specifications and restrictions are given in other manuals. Refer to Relevant Manuals on *Relevant Manuals* on page 2 and *Related Manuals* on page 18.

Relevant Manuals

The following table provides the relevant manuals for the FH Series.
Read all of the manuals that are relevant to your system configuration and application before you use the FH Series.

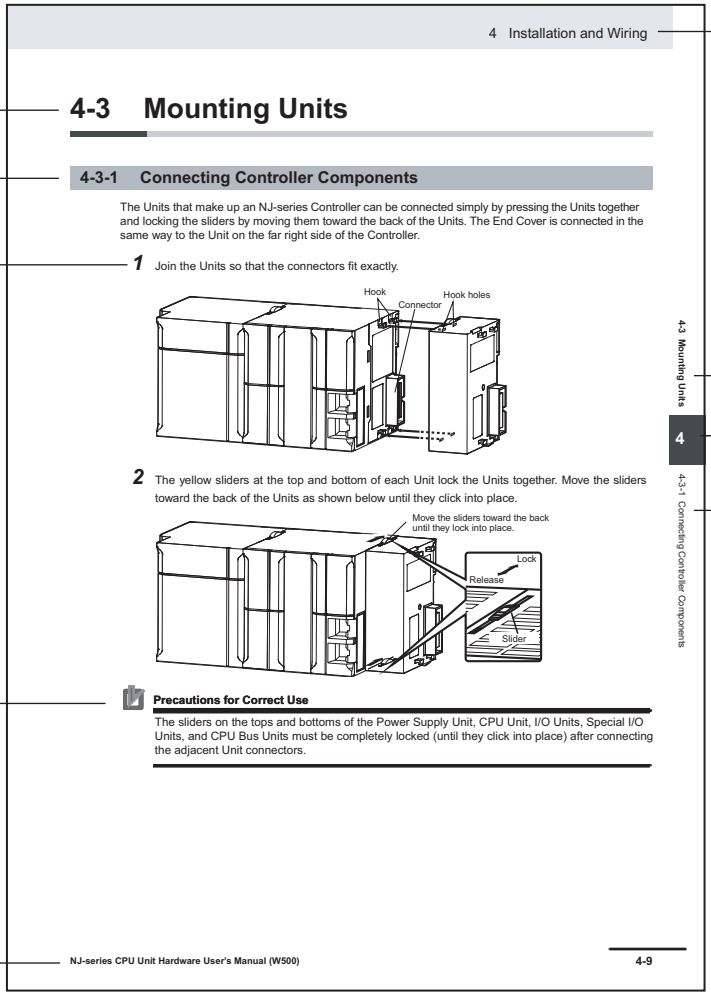
Purpose of use	Manual					
	Basic information					
	FH/FHV Series Vision System User's Manual	FH Series Vision System Hardware Setup Manual	FH/FHV Series Vision System Processing Item Function Reference Manual	FH Series Vision System Macro Customize Functions Programming Manual	FH/FHV Series Vision System User's Manual for Communications Settings	FH/FHV Series Vision System Operation Manual for Sysmac Studio
Overview of FH series	●	●				
Setup and Wiring						
EtherCAT						
EtherNet/IP						
PROFINET		●				
Ethernet						
RS-232C						
Parallel interface						
Setup the communication setting of sensor controller						●
EtherCAT						
EtherNet/IP						
PROFINET	●	●			●	
Ethernet						
RS-232C						
Parallel interface						
Setup the sensor controller						●
EtherCAT						
EtherNet/IP						
PROFINET	●				●	
Ethernet						
RS-232C						
Parallel interface						

Purpose of use	Manual					
	Basic information					
	FH/FHV Series Vision System User's Manual	FH Series Vision System Hardware Setup Manual	FH/FHV Series Vision System Processing Item Function Reference Manual	FH Series Vision System Macro Customize Functions Programming Manual	FH/FHV Series Vision System User's Manual for Communications Settings	FH/FHV Series Vision System Operation Manual for Sysmac Studio
Create and Set the Scene						●
EtherCAT						
EtherNet/IP						
PROFINET	●		●			
Ethernet						
RS-232C						
Parallel interface						
Optimizing the Scene Flow						
EtherCAT						
EtherNet/IP						
PROFINET			●	●		
Ethernet						
RS-232C						
Parallel interface						
Connecting the Controller						●
EtherCAT						
EtherNet/IP						
PROFINET	●	●			●	
Ethernet						
RS-232C						
Parallel interface						
Using Helpful Functions						●
EtherCAT						
EtherNet/IP						
PROFINET	●					
Ethernet						
RS-232C						
Parallel interface						
Troubleshooting and Problem Solving	●					

Manual Structure

Page Structure

The following page structure is used in this manual.



Level 1 heading

Level 2 heading

Level 3 heading

Gives the current headings.

Level 2 heading

Level 3 heading

A step in a procedure

Indicates a procedure.

Page tab

Gives the number of the main section.

Special information

Icons indicate precautions, additional information, or reference information.

Manual name

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

Special Information

Special information in this manual is classified as follows:

Precautions for Safe Use

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

Precautions for Correct Use

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

Additional Information

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

Conventions Used in This Manual

Use of Quotation Marks and Brackets

In this manual, menus and other items are indicated as follows.

Bold	Menu	Indicates the menu names or processing items shown in the menu bar.
Italic	Item name	Indicates the item names displayed on the screen.

Sections in This Manual

1 Overview of Macro Customize Functions

I Index

2 Using Macro Customize Functions

3 Screen Component and Setting Configuration

4 Basics of Programming

5 Macro Programming

6 Debug Function

7 Troubleshooting

8 Macro Functions

9 Macro Reference

CONTENTS

Introduction **1**
Intended Audience 1
Applicable Products 1

Relevant Manuals **2**

Manual Structure **4**
Page Structure 4
Special Information 5
Conventions Used in This Manual 5

Sections in This Manual **7**

Terms and Conditions Agreement **11**
Warranty, Limitations of Liability 11
Application Considerations 12
Disclaimers 12

Safety Precautions **14**

Precautions for Safe Use **15**

Precautions for Correct Use **16**

Regulations and Standards **17**

Related Manuals **18**

Terminology **19**

Revision History **23**

Section 1 Overview of Macro Customize Functions

1-1 Macro Customize Functions **1-2**
1-1-1 List of Macro Customize Functions 1-3
1-1-2 Structure of This Manual 1-5

Section 2 Using Macro Customize Functions

2-1 Instructions on Using Macro Customize Functions **2-2**
2-1-1 Components of the Macro Customize Functions 2-2
2-1-2 Procedures for Using the Macro Customize Functions 2-3

Section 3 Screen Component and Setting Configuration

3-1 Components of the Screens and How to Configure Settings **3-2**
3-1-1 Components of the Program Editing Screen 3-2
3-1-2 Description of the System Status Console Window 3-7

3-1-3 Description of the Setting Screen for the Unit Calculation Macro Processing Item and How to Configure Settings 3-8
3-1-4 Description of the Setting Screen of the Scene Control Macro Tool and How to Configure Settings 3-13
3-1-5 Components of the Setting Screen of the Communication Command Macro Tool and How to Configure Settings 3-15
3-1-6 Description of the Setting Screen of the Unit Macro Processing Item and How to Configure Settings 3-23
3-1-7 Saving and Loading Programs 3-27

Section 4 Basics of Programming

4-1 Basic Idea of Programming 4-2
4-1-1 Basic Syntax 4-2
4-1-2 Constant 4-5
4-1-3 Variable 4-6
4-1-4 Scene variables and System variables 4-11
4-1-5 Operator 4-12
4-1-6 Expression 4-14

Section 5 Macro Programming

5-1 How to Write Macro Programs 5-2
5-1-1 Data Types Related to Processing Units 5-2
5-1-2 Data Types Related to the System 5-7
5-1-3 Scope of Data and Save Area 5-10
5-1-4 State Transitions and Execution Timing 5-13
5-1-5 Exclusive Control in a Process 5-20

Section 6 Debug Function

6-1 How to Use the Debug Function 6-2
6-1-1 Debug Preparations 6-2
6-1-2 Debug Procedure 6-4
6-1-3 Checking Why an Error Occurred 6-5
6-1-4 Starting Debug 6-6
6-1-5 Identifying the Cause of an Error 6-7
6-1-6 Removing the Error 6-10
6-1-7 Exiting Debug 6-10

Section 7 Troubleshooting

7-1 Troubleshooting 7-2
7-1-1 Troubleshooting for Programming 7-2
7-1-2 Troubleshooting When Checking Operation 7-4
7-1-3 Troubleshooting during debugging 7-9
7-1-4 Troubleshooting during regular operation 7-10

Section 8 Macro Functions

8-1 Macro Function List 8-2
8-1-1 Alphabetical Order 8-2
8-1-2 Function-based Index 8-15
8-2 Macro Command Reference 8-24

Section 9 Macro Reference List

9-1 Error List **9-2**

9-2 List of Reserved Words **9-5**
9-2-1 List of Reserved Words 9-5
9-2-2 List of Reserved Global Data Words 9-9

9-3 System Data List **9-10**

9-4 List of I/O Modules **9-27**
9-4-1 EtherCAT 9-28
9-4-2 EtherNetIP 9-31
9-4-3 PROFINET 9-34
9-4-4 ParallelIo 9-37
9-4-5 SerialNormal 9-39
9-4-6 SerialPlcLinkM 9-41
9-4-7 SerialPlcLink 9-43
9-4-8 TcpClient 9-46
9-4-9 TcpNormal 9-48
9-4-10 UdpNormal 9-50
9-4-11 UdpPlcLinkM 9-52
9-4-12 UdpPlcLinkY 9-55
9-4-13 UdpPlcLink 9-58
9-4-14 TcpPlcLinkM 9-61
9-4-15 TcpPlcLink 9-64

9-5 List for Processing Item Identifier **9-67**
9-5-1 Measurement 9-67
9-5-2 Input image 9-67
9-5-3 Compensate image 9-68
9-5-4 Support measurement 9-68
9-5-5 Branch 9-69
9-5-6 Output result 9-69
9-5-7 Display result 9-69

9-6 Figure Data List **9-70**
9-6-1 Figure Data Structure List 9-70
9-6-2 List of Figure Data Array Elements 9-71

9-7 List of Figure Numbers **9-73**

9-8 Model Number List **9-76**

9-9 Image Number List **9-78**

9-10 List of Sub-Image Numbers **9-82**

Index

Terms and Conditions Agreement

Warranty, Limitations of Liability

Warranties

● Exclusive Warranty

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

● Limitations

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

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

● Buyer Remedy

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

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

Limitation on Liability; Etc

OMRON COMPANIES SHALL NOT BE LIABLE FOR SPECIAL, INDIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, LOSS OF PROFITS OR PRODUCTION OR COMMERCIAL LOSS IN ANY

WAY CONNECTED WITH THE PRODUCTS, WHETHER SUCH CLAIM IS BASED IN CONTRACT, WARRANTY, NEGLIGENCE OR STRICT LIABILITY.

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

Application Considerations

Suitability of Use

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

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

Programmable Products

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

Disclaimers

Performance Data

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

Change in Specifications

Product specifications and accessories may be changed at any time based on improvements and other reasons. It is our practice to change part numbers when published ratings or features are changed, or when significant construction changes are made. However, some specifications of the Product may

be changed without any notice. When in doubt, special part numbers may be assigned to fix or establish key specifications for your application. Please consult with your Omron's representative at any time to confirm actual specifications of purchased Product.

Errors and Omissions

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

Safety Precautions

For details of Safety Precautions, refer to *Safety Precautions* in the *Vision System FH/FHV Series User's Manual (Cat. No. Z365)*.

Precautions for Safe Use

For details of Precautions for Safe Use, refer to *Precautions for Safe Use* in the *Vision System FH/FHV Series User's Manual (Cat. No. Z365)*.

Precautions for Correct Use

For details of Precautions for Correct Use, refer to *Precautions for Correct Use* in the *Vision System FH/FHV Series User's Manual (Cat. No. Z365)*.

Regulations and Standards

For details of Regulations and Standards, refer to *Regulations and Standards* in the *Vision System FH/FHV Series User's Manual (Cat. No. Z365)*.

Related Manuals

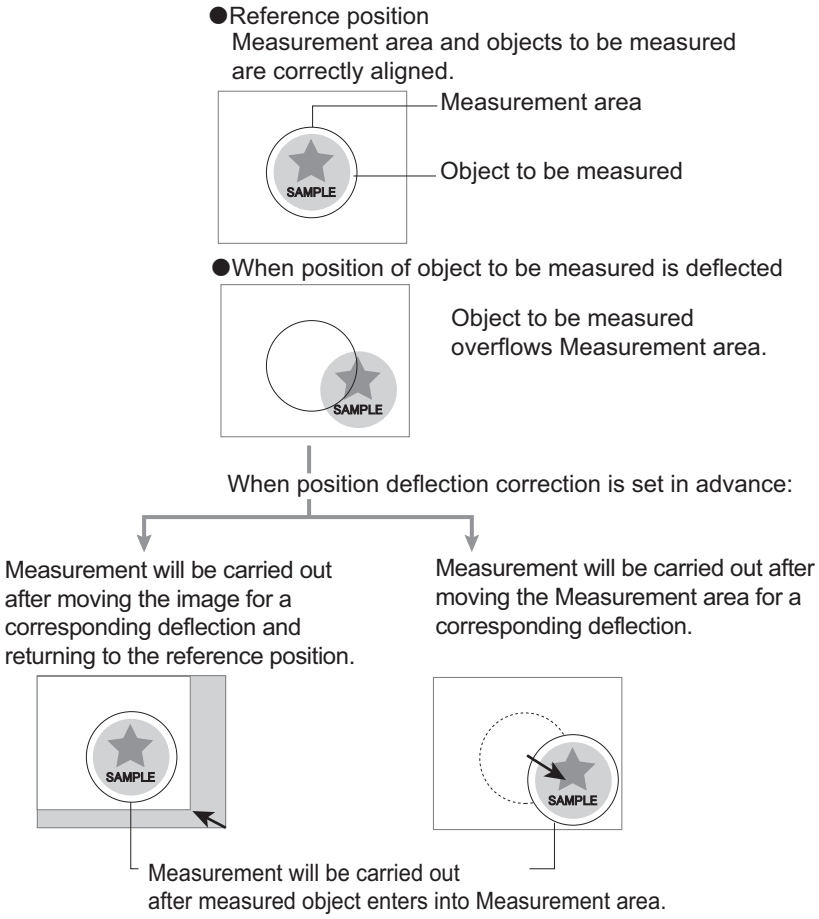
The followings are the manuals related to this manual. Use these manuals for reference.

Name of Manual	Cat. No.	Model	Purpose	Contents
Vision System FH Instruction Sheet	3648743-1	FH-2□□□ FH-2□□□-□□ FH-5□□□ FH-5□□□-□□	To confirm the safety and usage precautions of the Vision System FH series sensor controller.	Describes the definitions of basic terms, meaning of signal words, and precautions for correct use of FH series in the manual.
Vision System FH-L Instruction Sheet	3615792-0	FH-L□□□ FH-L□□□-□□	To confirm the safety and usage precautions of the Vision System FH-Lite series sensor controller.	Describes the definitions of basic terms, meaning of signal words, and precautions for correct use of FH-L series in the manual.
Vision System FH/FHV Series User's Manual	Z365	FH-2□□□ FH-2□□□-□□ FH-5□□□ FH-5□□□-□□ FH-L□□□ FH-L□□□-□□	When User want to know about the FH/FHV series.	Describes the soft functions, setup, and operations to use FH/FHV series/
Vision System FH/FHV series Processing Item Function Reference Manual	Z341		When User confirm the details of each processing items at the create the measurement flow or operate it.	Describes the software functions, settings, and operations for using FH/FHV series.
Vision System FH/FHV Series User's manual for Communications Settings	Z342		When User confirm the setting of communication functions.	Describes the functions, settings, and communications methods for communication between FH/FHV series and PLCs. The following communications protocol are described. Parallel, PLC Link, EtherNet/IP, EtherCAT, and Non-procedure.
Vision System FH series Hardware Setup Manual	Z366		When User want to know about the Hard-ware specifications or to setup the sensor controller of the Vision System FH series.	Describes FH series specifications, dimensions, part names, I/O information, installation information, and wiring information.
Vision System FH series Macro Customize Functions Programming Manual	Z367		When User operate or programming using Macro Customize functions.	Describes the functions, settings, and operations for using Macro Customize function of the FH series.
Vision System FH/FHV Series Operation Manual for Sysmac Studio	Z343	FH-2□□□ FH-2□□□-□□ FH-5□□□ FH-5□□□-□□	When User connect to NJ/NX series via EtherCAT communication.	Describes the operating procedures for setting up and operating FH/FHV series Vision Sensors from the Sysmac Studio FH/FHV Tools.

Terminology

Term	Definition
FH Series	All FH series model names as follows: FH-2□□□, FH-2□□□-□□, FH-5□□□, FH-5□□□-□□, FH-L□□□, FH-L□□□-□□
FH-2000 series	All FH-2□□□ series model names as follows: FH-2□□□, FH-2□□□-□□
FH-5000 series	All FH-5□□□ series model names as follows: FH-5□□□, FH-5□□□-□□
FH-L series	All FH-L□□□ series model names as follows: FH-L□□□, FH-L□□□-□□
FHV Series	All FHV series model names.
FZ5 series	All FZ series name shows the following: FZ5-6□□, FZ5-6□□-□□, FZ5-8□□, FZ5-8□□-□□, FZ5-11□□, FZ5-11□□-□□, FZ5-12□□, FZ5-12□□-□□, FZ5-L35□, FZ5-L35□-□□
FZ5-600 series	All FZ5-6□□ series name the following: FZ5-6□□, FZ5-6□□-□□
FZ5-800 series	All FZ5-8□□ series name the following: FZ5-8□□, FZ5-8□□-□□
FZ5-1100 series	All FZ5-11□□ series name the following: FZ5-11□□, FZ5-11□□-□□
FZ5-1200 series	All FZ5-12□□ series name the following: FZ5-12□□, FZ5-12□□-□□
FZ5-L series	All FZ5-L35□ series name the following: FZ5-L35□, FZ5-L35□-□□
Sensor controller	It is a generic name of FH/FZ5 series. For FHV series, it has the same meaning as *Smart Camera*.
Measurement flow (abbreviated as *flow*)	A continuous flow of measurement processing. A measurement flow consists of a scene created from a combination of processing items.
Measurement processing	Executing processing items for inspections and measurements.
Measurement ID	Measurement ID is the line number combined with the date and time information of the sensor controller at the time of measurement. Format of measurement ID: YYYY-MM-DD_HH-MM-SS-XXXN (YYYY: Year, MM: Month, DD: Date, HH: Hour, MM: Minute, SS: Second, XXX: Millisecond, N: Line number) • Example: When the measurement time is 11:10:25.500 AM, December 24, 2007 and the line is 0, the measurement ID is "2007-12-24_11-10-25-5000".
Processing item	Any of the individual items for vision inspections that are partitioned and packaged so that they can be flexibly combined. These include the Search, Position Compensation, and Fine Matching items. Processing items can be classified for image input ([Input image]), inspection/measurement ([Measurement]), image correction ([Compensate image]), inspection/measurement support ([Support measurement]), process branching ([Branch]), results external output ([Output result]), resulting image display ([Display result]), etc. You can freely classify processing items to handle a wide range of applications. A scene (i.e., a unit for changing the measurement flow) is created by registering the processing items as units.

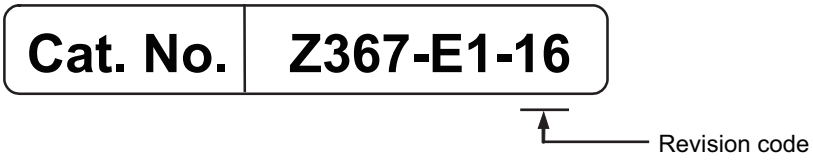
Term	Definition
Scene	A unit for changing the measurement flow that consists of a combination of processing items. Scene is used because of the correspondence to the scene (i.e., type of measurement object and inspection contents) where measurements are performed. A scene is created for each measurement or measurement contents. You can easily achieve a changeover simply by changing the scene when the measurement object or inspection content changes. Normally you can set up to 128 scenes. If you need more than 128 scenes, you can separate them into different groups or use the Conversion Scene Group Data Tool to create a scene group that contains over 128 scenes.
Processing unit (abbreviated as *unit*)	A processing item that is registered in a scene. Numbers are assigned to processing units in order from the top and they are executed in that order. Processing items are registered for the processing units to create a scene (i.e., a unit for changing the measurement flow).
Measurement trigger	A trigger for executing measurements. (STEP signal, etc.) (In addition to measurement trigger, measurement can also be executed by measurement command.)
Test measurement	A measurement that is performed to manually test (check) measurements under the conditions that are set in the currently displayed scene. Test measurements can be executed on an Adjustment Window. Processing is completed inside the Controller and the measurement results are not normally output on an external interface. However, you can select **Output** in **Test measurement** to output the measurement results after executing measurements.
Single measurement	A measurement that is executed only once in synchronization with the trigger input.
Continuous measurement	Measurements are executed repeatedly and automatically without a trigger input.
Operation mode	• Double Speed Multi-input: A mode that processes the measurement flow for the first trigger and then processes the measurement flow in parallel for the second trigger to achieve a high-speed trigger input interval. It is used together with the multi-input function. • Multi-line Random-trigger: A trigger mode that allows you to independently processing multiple measurement flows. With traditional image processing, two or more triggers cannot be acknowledged at the same time. In Multi-line Random-trigger Mode, you can randomly input multiple triggers into one Controller to independently process multiple scenes in parallel. • Non-stop adjustment mode: A mode that allows you to adjust the flow and set parameters while performing measurements. The enables adjustments without stopping the line or stopping inspections. • Standard: A logging mode that allows complete parallel processing of measurements and logging. Traditionally, logging was not possible while processing measurements. Either measurements or logging had to be given priority and the other one had to wait. With this mode, you can save the measurement images in external storage without affecting the transaction time.

Term	Definition
Parallel processing (an option for any of the above operation modes)	Parallel processing splits part of the measurement flow into two or more tasks, and processes each task in parallel to shorten the transaction time. Processing items for parallel processing are used so that the user can specify the required parallel processing.
Multi-input function	A function that is used to consecutively and quickly input images. It allows the next STEP signal to be acknowledged as soon as the image input processing is completed. There is no need to wait for measurement processing to be completed. You can check whether image input processing has been completed with the status of the READY signal. Even if the READY signal is ON when measurement processing is being executed, the next STEP signal can be acknowledged.
Position compensation	When the location and direction of measured objects are not fixed, the positional deviation between reference position and current position is calculated and measurement is performed after correcting. Please select processing items that are appropriate to the measurement object from processing items that are related to position compensation. 
Reference position	The point that is always the reference. If the location of the registered model is different from the reference position, the setting should be changed in **Ref. setting**.
Model	The image pattern that serves as the inspection target. Characteristics portions are extracted from images of the object and registered as model registration.

Term	Definition
2's complement	Binary numbers are generally used to represent negative numbers. Negative numbers are expressed by *Inverting all bits of a positive number and adding 1 to the result*. Ex. *-1* is expressed as 2's complement. *-1* can be calculated by *0-1*. (In the case of 1, minus 1) 00000000 (= 0) -) 00000001 (= 1) 11111111 (=-1) ⇦ −“1” expresses with 2’s Complement (for 8 bits) There are methods for simple calculation without performing this kind of computation. For instance, *Negative number = inverting all bits of a positive number and then adding 1 to the result*. 00000001 (= 1) ↓ Invert all bits 11111110 ↓ Plus 1 11111111 (=-1) The *first digit* is used to judge whether the number is positive or negative. • When 0: Positive number (or 0) • When 1: Negative number The advantage of two's complement numbers is that positive and negative numbers can be used as is in calculations. Ex. When -1+10=9 11111111 (= -1) +)00001010 (= 10) 00001001 (= 9)

Revision History

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



Rev. Code	Rev. Date	Revision Contents	Software Version
01	Apr. 2016	Original production	Ver.5.6
02	Mar. 2017	Corrected mistakes.	Ver.5.71
03	Jun. 2017	Additions for software version upgrade.	Ver.5.72
04	Jul. 2018	Additions for software version upgrade.	Ver.6.1
05	Jul. 2019	Additions for software version upgrade. Removed the description of FZ5 series.	Ver.6.3
06	Nov. 2019	Layout adjustment. Corrected mistakes.	Ver.6.3
07	Jun. 2020	Additions for software version upgrade.	Ver.6.4
08	Nov. 2020	Corrected mistakes.	Ver.6.4
09	Jun. 2021	Corrected mistakes.	Ver.6.4
10	May 2022	Removed FH-1050, FH-1050-10, FH-1050-20, FH-3050, FH-3050-10, and FH-3050-20. Additions for software version upgrade. Corrected mistakes.	Ver.6.5
11	Dec. 2022	Revisions for update , *Related Manuals*. Corrected mistakes.	Ver.6.5
12	Mar. 2023	Corrected mistakes.	Ver.6.5
13	Mar. 2023	Additions for software version upgrade (8-2 Macro Command Reference, 9-1-3 System Data List, 9-1-9 Image Number List). Revisions for update , *Related Manuals*. Corrected mistakes.	Ver.6.55 Ver.6.6

Rev. Code	Rev. Date	Revision Contents	Software Version
14	Jun. 2025	Removed FH-2050, FH-2050-10, FH-2050-20, FH-5050, FH-5050-10, FH-5050-20, FH-5550, FH-5550-10, FH-5550-20, FH-L550 and FH-L550-10. Additions for software version upgrade (3-1-1 Components of the Program Editing Screen, 3-1-3 Description of the Setting Screen for the Unit Calculation Macro Processing Item and How to Configure Settings, 3-1-4 Description of the Setting Screen of the Scene Control Macro Tool and How to Configure Settings, ExecuteImage-Logging, Measure, SaveMeasureImage, 9-1-2 List of Reserved Words, 9-1-3 System Data List, 9-1-5 List for Processing Item Identifier, 9-1-7 List of Figure Numbers, 9-1-9 Image Number List, 9-1-10 List of Sub-Image Numbers). Corrected mistakes.	Ver.6.7
15	Nov. 2025	Layout adjustment (Section 9). Corrected mistakes.	Ver.6.7
16	Sep. 2026	Additions for software version upgrade (System-Info, 9-1-3 System Data List) Corrected mistakes.	Ver.6.8

1

Overview of Macro Customize Functions

1-1 Macro Customize Functions 1-2
1-1-1 List of Macro Customize Functions 1-3
1-1-2 Structure of This Manual 1-5

1-1 Macro Customize Functions

In the FH series, the macro customize functions can be used to realize finely adjusted and expandable image processing.
The macro customize functions enable you to perform various types of calculations that are more advanced than normal *Calculation* processing items, as well as functions such as measurement flow/scene control, creation of communication commands, various types of display control, and result output control.
On the FH series, the following types of macro functions can be used. The four types below can be used.

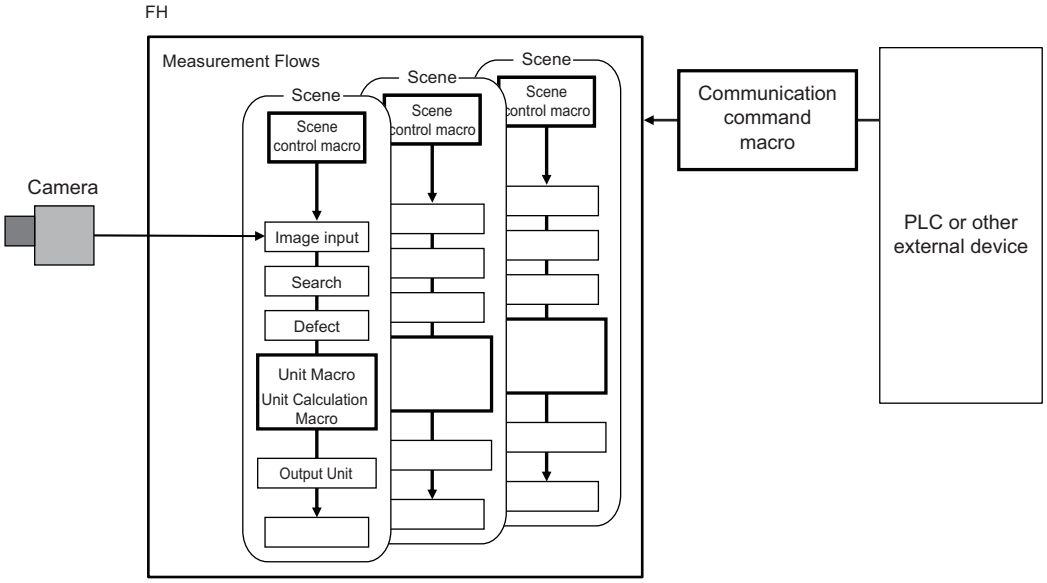
- *Unit Calculation Macro* processing item
- *Scene Control Macro* tool
- *Communication Command Macro* tool
- *Unit Macro* processing item

1-1-1 List of Macro Customize Functions

A table of the macro customize functions and their approximate level of difficulty is shown below. The level of difficulty varies by function. Check the approximate level of difficulty in the table when considering the expansion you want to create.

Approximate level of difficulty	Function	Description	Target function			
			Measurement processing		Communication commands	Scene control
			Calculation	Other than calculation*1		
High ↑	Unit Macro processing item	Use this function to supplement and expand measurement processing performed by processing units. In addition to measurement processing, you can create your own custom processing such as result display processes and measurement initialization processes for scene control.	●	●	-	-
	Communication Command Macro tool	Use this function to supplement and expand communication commands. You can create custom communication commands that implement functions that do not exist in the standard communication commands, and combine multiple communication commands into a signal command.	-	-	●	-
	Scene Control Macro tool	Use this function to supplement and expand measurement flow and Scene Control. For example, you can add and set processing units to the measurement flow.	-	-	-	●
↓ Low	Unit Calculation Macro processing item	Use this function to supplement and expand calculation processes during measurement. A *Calculation* processing item allows you to implement complex operations with difficult settings, and calculations that include a logical expression or repeated process.	●	-	-	-

*1. Result display (graphic display, detailed text display, etc.), result output, and creation of initial processing



In the following cases, a macro customize function is used.

In this case	Macro customize function that is used
Use to perform a calculation process that is difficult or cannot be expressed using a *Calculation* processing item, such as those below. • Use to execute a calculation process that extends over multiple lines or contains a logical expression, conditional branch, loop process, or data setting process • Use to execute a customized judgement process for calculation results	Unit Calculation Macro processing item
Use to customize or include the following processes that that are difficult or cannot be expressed with a *Calculation* processing item or a *Unit Calculation Macro* processing item. • Display result (graphic display, detailed text display, etc.) • Output result • Measurement initialization processing (scene switching, etc.)	Unit Macro processing item
Use to control the Scene such as the followings. • Changing the settings of multiple processing units at once • Managing the common data of multiple processing units • Adding/deleting a processing unit	Scene Control Macro tool
Use to create expansions that are difficult or cannot be expressed with the standard communication commands, such as those below. • Not to exist in the standard communication commands. • To combine multiple communication command functions into a single communication command	Communication Command Macro tool

1-1-2 Structure of This Manual

The relation between the contents of this manual and the macro customize functions is shown below. Refer to function items that you need to use.

Item name	Macro customize function			
	Unit Calculation Macro	Scene Control Macro	Communication Command Macro	Unit Macro
What are the Macro Customize Functions? Reference: *Section 1 Overview of Macro Customize Functions* on page 1-1	Required			
How to use Macro Customize Functions Reference: *Section 2 Using Macro Customize Functions* on page 2-1	Required			
Preparations for use of the Unit Calculation Macro processing item Reference: *Procedure for Using the unit calculation macro processing item* on page 2-4	Required	Not required		
Preparations for use of the scene control macro Reference: *Procedure for Using the Scene Control Macro Tool* on page 2-4	Not required	Required	Not required	
Preparations for use of the communication command macro tool Reference: *Procedure for Using the Communication Command Macro Tool* on page 2-5	Not required		Required	Not required
Preparations for use of the Unit Macro processing item Reference: *Procedure for Using the unit macro processing item* on page 2-6	Not required			Required
Components of the Screens and How to Configure Settings Reference: *Section 3 Screen Component and Setting Configuration* on page 3-1	Required			
Basic Method for Writing Programs Reference: *Section 4 Basics of Programming* on page 4-1	Required			
How to Write Advanced Programs Reference: *Section 5 Macro Programming* on page 5-1	As needed			
Data Types Related to Processing Units Reference: *5-1-1 Data Types Related to Processing Units* on page 5-2	As needed			
Data Types Related to the System Reference: *5-1-2 Data Types Related to the System* on page 5-7	As needed			
Scope of Data and Save Area Reference: *5-1-3 Scope of Data and Save Area* on page 5-10	As needed			
Status Transitions and Execution Timing Reference: *5-1-4 State Transitions and Execution Timing* on page 5-13	Not required	As needed	Not required	As needed

Item name		Macro customize function			
		Unit Calculation Macro	Scene Control Macro	Communication Command Macro	Unit Macro
	Exclusive Control in a Process Reference: *5-1-5 Exclusive Control in a Process* on page 5-20	As needed			
How to Use the Debug Function Reference: *Section 6 Debug Function* on page 6-1		Required			
Troubleshooting Reference: *Section 7 Troubleshooting* on page 7-1		As needed			
Macro Function List Reference: *Section 8 Macro Functions* on page 8-1		As needed			
Macro Reference List Reference: *Section 9 Macro Reference List* on page 9-1		As needed			

2

Using Macro Customize Functions

2-1 Instructions on Using Macro Customize Functions 2-2
2-1-1 Components of the Macro Customize Functions 2-2
2-1-2 Procedures for Using the Macro Customize Functions 2-3

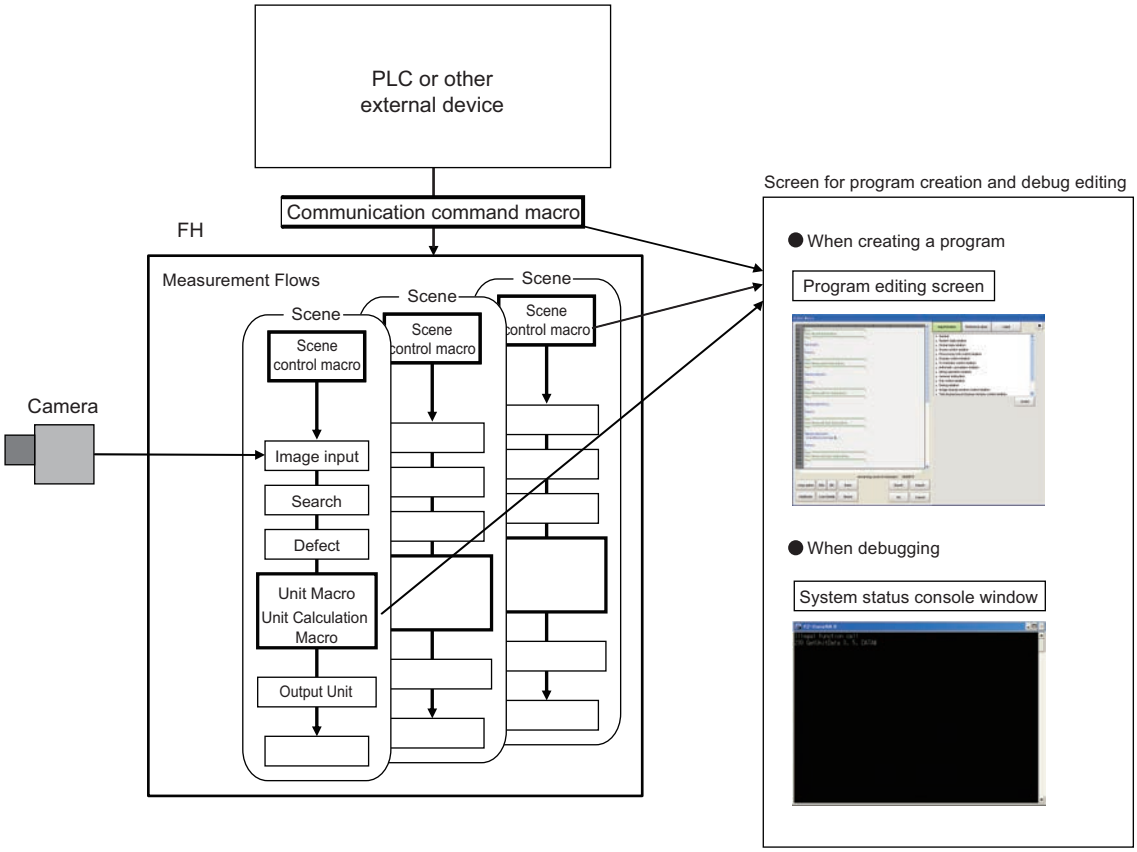
2-1 Instructions on Using Macro Customize Functions

Necessity of referring to this manual	Unit Macro	Communication Command Macro	Scene Control Macro	Unit Calculation Macro
	Required			

You can use the macro customize functions to program processes that you want to add or expand. On a sensor controller, you can execute the macro customize functions and debug the programs of the macro customize functions.

2-1-1 Components of the Macro Customize Functions

The macro customize functions consist of program editing screens that enable the creation and editing of programs in the processing items, tool setting screens, and setting screens of each function, and a system status console window that allow errors to be checked when a program operates abnormally.



Function	Description
Program editing screen	Setting screen of each function in the macro customize functions. The contents of the program editing screen vary by function. Use the program editing screen to create programs. You can create and edit programs, and use input auxiliary functions. Reference: *3-1-1 Components of the Program Editing Screen* on page 3-2

Function	Description
System status console window	Console window that shows the system status as text. When a program created with a macro customize function operates abnormally, a description of the error appears in text in the system status console window. Use this to debug the program. Reference: *3-1-2 Description of the System Status Console Window* on page 3-7

Precautions for Correct Use

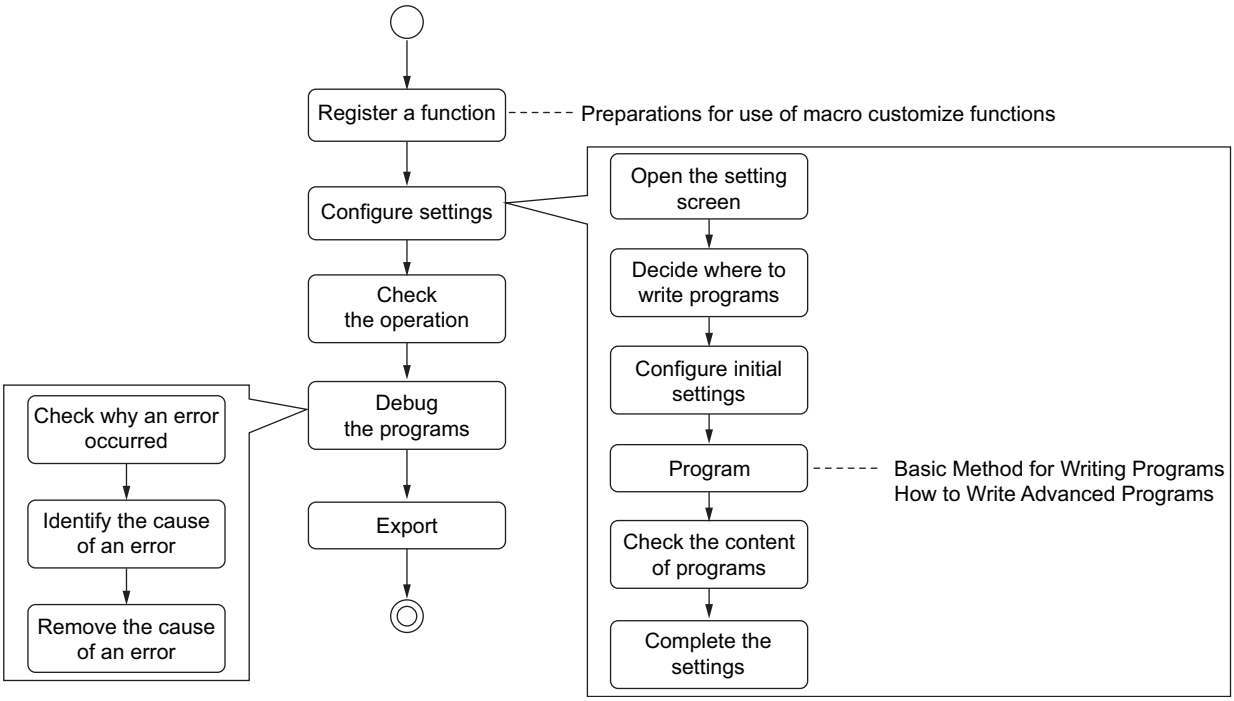
The setting screens of the macro customize functions cannot be displayed by remote operation. To change settings, directly open the setting screen of the function on the sensor controller.

2-1-2 Procedures for Using the Macro Customize Functions

The procedures for using macro customize functions are two types, one is a common procedure used for all functions and second is the specific procedures for each function.

Common procedures for Using the Macro Customize Functions

The flow of the common procedure for using the macro customize functions is shown below.



Item name	Step	Description
Function registration and preparation	Register a function and display the setting screen	Register processing items that enable use of macro customize functions in the measurement flow, and open tool screens.The specific procedure depends on the each functions.
	↓	
Function settings	Default function settings	Prepare initial settings and variables.The specific procedure depends on the each functions.
	↓	
	Creation of processing content	Write the program.
	↓	

Item name	Step	Description
Program debug	Check operation	Check if your settings operate as expected.
	↓	
	Debug	Debug the program. Debug helps you identify the cause of the unexpected operation and correct the program to run as expected.
	↓	
Save	Save settings	Save your changes.

Procedure for Using the unit calculation macro processing item

The usage flow for the unit calculation macro processing item and the basic usage procedures are described below.

Item name	Step	Description
Registration and preparation	Registration of a unit calculation macro processing unit	Add a macro calculation processing unit to the measurement flow.
	↓	
Settings of the unit calculation macro processing unit (Reference: *3-1-3 Description of the Setting Screen for the Unit Calculation Macro Processing Item and How to Configure Settings* on page 3-8)	Selection of operators	Select the checkboxes of operators to be used in the unit calculation macro.
	↓	
	Reference Variable Settings	Set reference values if reference values will be used. Set reference values in order to use data other than that of the macro calculation processing unit, such as external reference data of other processing units and system data.
	↓	
	Program input	Write the program.
	↓	
	Setting of judgement conditions	Set the conditions used to judge calculation results.
	↓	
Program debug (Reference: *6-1 How to Use the Debug Function* on page 6-2)	Check operation	Check if your settings operate as expected.
	↓	
	Debug	Debug the program. Debug helps you identify the cause of the unexpected operation and correct the program to make the unit calculation macro processing unit operate as expected.
	↓	
Save	Save settings	Save your changes.

Procedure for Using the Scene Control Macro Tool

The usage flow for the scene control macro tool and the basic usage procedures are described below.

Item name	Step	Description
Registration and preparation	Starting the Scene Control Macro Tool	Start the scene control macro tool from the external tools.
	↓	

Item name	Step	Description
Scene control macro tool settings (Reference: *3-1-4 Description of the Setting Screen of the Scene Control Macro Tool and How to Configure Settings* on page 3-13)	Reference Variable Settings	Set reference values if reference values will be used. Set reference variables in order to use data such as external reference data of processing units and system data.
	↓	
	Unit label settings	Set unit labels if unit labels will be used. Make preparations to use unit labels, rather than processing unit numbers, to reference processing units in the measurement flow.
	↓	
	Program input	Write in the program.
	↓	
Program debug (Reference: *6-1 How to Use the Debug Function* on page 6-2)	Check operation	Check if your settings operate as expected.
	↓	
	Debug	Debug the program. Debug helps you identify the cause of the unexpected operation and correct the program to make the scene control macro operate correctly.
	↓	
Save	Save settings	Save your changes.

Procedure for Using the Communication Command Macro Tool

The usage flow for the communication command macro tool and the basic usage procedure are described below.

Item name	Step	Description
Registration and preparation	Starting the communication command macro tool	Start the communication command macro tool from the external tools.
	↓	
Communication command macro settings (Reference: *3-1-5 Components of the Setting Screen of the Communication Command Macro Tool and How to Configure Settings* on page 3-15)	Selection of the communication command macro to be used	Select the checkbox of the communication command macro to be used.
	↓	
	Communication command macro name setting	Assign a name to the communication command macro.
	↓	
	Flow signal output setting	Select whether flow signals such as the BUSY signal are turned ON or left OFF during processing. To turn on, select the checkbox.
	↓	
	Program input	Write the program.
	↓	
Program debug (Reference: *6-1 How to Use the Debug Function* on page 6-2)	Check operation	Check if your settings operate as expected.
	↓	
	Debug	Debug the program. Debug helps you identify the cause of the unexpected operation and correct the program to make the communication command macro operate as expected.
	↓	

Item name	Step	Description
Save	Save settings	Save your changes.

Procedure for Using the unit macro processing item

The usage flow for the unit macro processing item and the basic usage procedure are described below.

Item name	Step	Description
Registration and preparation	Registration of unit macro processing unit	Add a unit macro processing unit to the measurement flow.
	↓	
Settings of unit macro processing unit (Reference: *3-1-6 Description of the Setting Screen of the Unit Macro Processing Item and How to Configure Settings* on page 3-23)	Reference Variable Settings	Set reference values if reference values will be used. Set reference values to use data other than that of the unit macro processing unit, such as external reference data of other processing units and system data.
	↓	
	Program input	Write the program.
	↓	
Program debug (Reference: *6-1 How to Use the Debug Function* on page 6-2)	Check operation	Check if your settings operate as expected.
	↓	
	Debug	Debug the program. Debug helps you identify the cause of the unexpected operation and correct the program to make the unit macro processing unit operate as expected.
	↓	
Save	Save settings	Save your changes.

3

Screen Component and Setting Configuration

3-1 Components of the Screens and How to Configure Settings 3-2
3-1-1 Components of the Program Editing Screen 3-2
3-1-2 Description of the System Status Console Window..... 3-7
3-1-3 Description of the Setting Screen for the Unit Calculation Macro Processing Item and How to Configure Settings..... 3-8
3-1-4 Description of the Setting Screen of the Scene Control Macro Tool and How to Configure Settings..... 3-13
3-1-5 Components of the Setting Screen of the Communication Command Macro Tool and How to Configure Settings 3-15
3-1-6 Description of the Setting Screen of the Unit Macro Processing Item and How to Configure Settings..... 3-23
3-1-7 Saving and Loading Programs 3-27

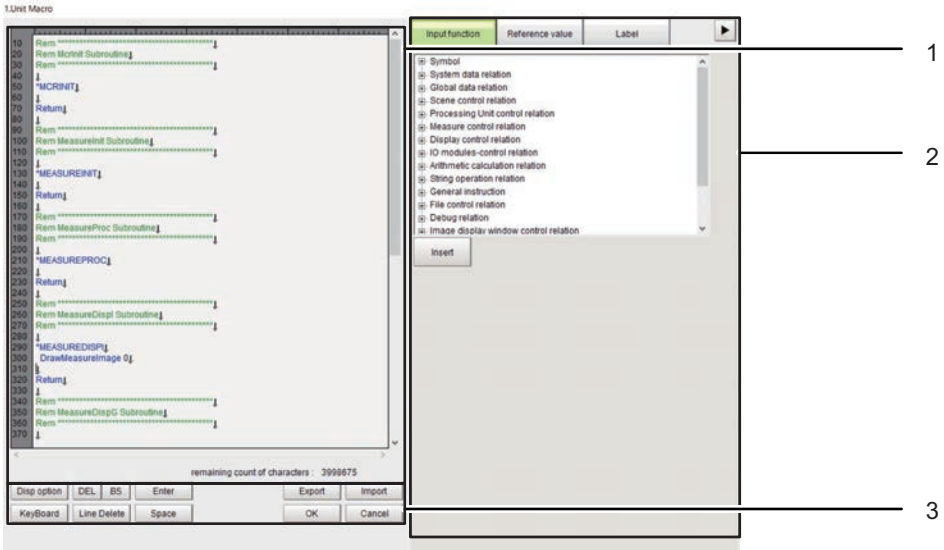
3-1 Components of the Screens and How to Configure Settings

Necessity of referring to this manual	Unit Macro	Communication Command Macro	Scene Control Macro	Unit Calculation Macro
	Required			

The setting screens of the macro customize function consist of a program editing screen that is shared by all macro customize functions, and individual setting screens for each function. The setting methods vary by setting screen.

3-1-1 Components of the Program Editing Screen

Use the program editing screen to edit a program in the macro customize function.
The program editing screen consists of the areas below.



1. **Program area**
   The program appears in this area. You can select whether some of the display items are displayed. Refer to *Display Option* on page 3-6.
   - Program input area
     This area is used to enter programs. Create and edit programs in this area.
     When creating and editing programs in the program input area, use the keyboard to write the program in the program input area.
     The screen keyboard can be displayed from the operation button area. Refer to *3. Operation button area* on page 3-5.

```
10  Rem ****************************************↓
20  Rem McrInit Subroutine↓
30  Rem ****************************************↓
40  ↓
50  *MCRINIT↓
60  ↓
70  Return↓
80  ↓
90  Rem ****************************************↓
100 Rem MeasureInit Subroutine↓
```

The control characters are displayed in a visible form in the program area. Control characters are included in the remaining count of characters, and thus are convenient for checking the number of inputtable characters. Refer to *Remaining count of characters* on page 3-4.

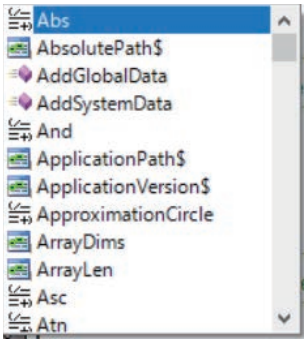
Display (color)	Description
↓ (gray)	Indicates a line break.
□ (gray)	Indicates a double-byte space.
→ (gray)	Indicates the tab character.

Precautions for Correct Use

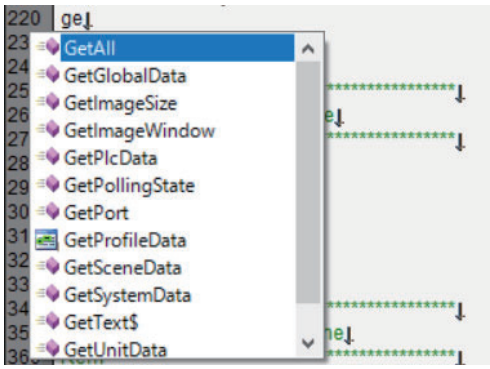
- You can add comments in the program area.
  Note that allowable character strings differ depending on the type of the sensor controller as shown below.
  English characters and characters for the language selected in *Language setting* are allowed.
  Refer to *Comment* on page 4-4.

Additional Information

- If a function entered in a program cannot be used, the function name is shown in red.
  Whether or not a function can be used depends on the macro customize function that is used. For details, refer to *Section 9 Macro Reference List* on page 9-1.
- Enter the *Ctrl* + *Space* or *Alt* + → keys to display a list of candidates. You can also enter it in the program area by selecting from the list of candidates and double-clicking or entering the *Enter* key.



- If more than two characters are entered, a list of candidates will be automatically displayed. You can also enter it in the program area by selecting from the list of candidates and double-clicking or entering the *Enter* key.



- Line number display area
  The program line number appears in this area. The line number is used in debugging. For lines and line numbers, refer to the *4-1-1 Basic Syntax* on page 4-2.

Additional Information

- Line numbers are assigned as unique numbers in the program. When multiple calculation expressions are set for one unit calculation macro processing unit, unique line numbers that are not redundantly used in the multiple calculation expressions in the processing unit are assigned.

- Remaining count of characters
  This area shows how many characters can still be input in the program. Control characters such as line breaks and tabs are also included in the count. Create/edit the program so that the remaining inputtable character count is 0 or more. If a program has more than the inputtable character count, it may not operate correctly.

2. **Supplemental Program Input Area**
   This area can be used to input supplemental settings and perform supplemental input operations for programs.
   To hide the supplemental program input area and enlarge the program area, click the ▶ button. Click the button again to return to the previous screen.
   - Input function tab
     This area shows a list of the macro functions.

Button	Description
Insert	Inserts the selected macro function immediately behind the cursor position in the program input area.

- Reference variable tab
  This area allows to register and configure settings for reference variables. This area only appears in the program editing screen for unit calculation macros, scene control macros, and unit macros. This area is not shown in the communication command macro.
  Refer to *4-1-3 Variable* on page 4-6.

Button	Description
Insert	Inserts the selected reference variable immediately behind the cursor position in the program input area.

Button	Description
Add	Displays the reference variable setting screen. You can add a reference variable in the setting screen.
Delete	Deletes a reference variable that has been selected in the reference variable list.
Edit	Displays the setting screen of a reference variable selected in the reference variable list. You can change the settings of the reference variable in the setting screen.
Export	Displays the save file screen. Specify the save location and file name to save the settings of the current reference variable list as a file in XML format.
Import	Displays the import file screen. Imports reference variables saved in a file in XML format.

- Label tab
  This area can be used to set and reference unit labels.

Button	Description
Insert	Inserts the selected unit label immediately behind the cursor position in the program input area.
Edit	Displays the setting screen of unit labels. For details on unit labels, refer to *3-1-4 Description of the Setting Screen of the Scene Control Macro Tool and How to Configure Settings* on page 3-13.

3. **Operation button area**
   This area contains buttons for editing programs by button operation, and buttons for exporting and importing programs.
   The buttons that can be used depend on the macro customize function that is used.
   Buttons that can be used in the operation button area are shown below.

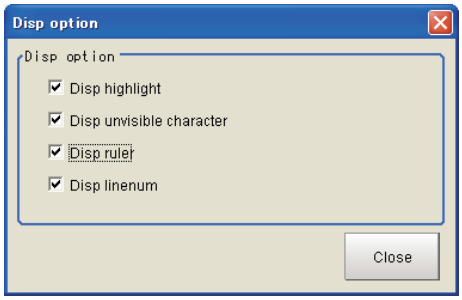
Button	Description
Display Settings	Shows the display settings screen.
Keyboard	Displays the screen keyboard. Use the keyboard to create and edit programs.
DEL	Deletes one character immediately after the cursor position in the program input area.
BS	Deletes one character immediately in front of the cursor position in the program input area.
Enter	Breaks the line at the cursor position in the program input area.
Line Delete	Deletes the line where the cursor is located in the program input area.
Space	Inserts a space at the cursor position in the program input area.
Clear	Initializes the program shown in the program area. Only appears for the scene control macro.
Export	Exports the program file to a file. Only appears for the unit macro.
Import	Imports a program file from a file. Only appears for the unit macro.
OK	Finalizes the edited contents of the program editing screen and closes the screen. If there is a error in the program, the error dialog box appears. If the error dialog box appears, remove the error and click OK. A description of the error appears in the system status console window.
Cancel	Discards editing changes in the program editing screen and closes the screen.

Additional Information

- In the FH series, the layout of the keyboard on the sensor controller is the same as an English keyboard.To enter Japanese, you can change the input mode with **Alt** + **~**.
- If the program has an error, the error dialog appears when [OK] is clicked. If the error dialog box appears, remove the error and click OK to complete the settings. You can view a description of the error in the system status console window.
  For details, refer to *3-1-2 Description of the System Status Console Window* on page 3-7.

Display Option

To add the contents of the program area and make the program easier to view, change the program area display settings.
The settings in the display option screen are described below.



Setting item	Setting value [Factory default]	Description
Disp highlight	• [Checked] • Unchecked	Highlights the program in the program input area with distinguishing colors. Each keyword type is highlighted in a different color, making the program easier to view.
Disp unvisible character	• [Checked] • Unchecked	Shows line breaks, tabs, and other control characters in the program input area in a visible form. For details on hidden characters, refer to *1. Program Area* on page 3-2.
Disp ruler	• [Checked] • Unchecked	Shows the ruler in the program input area. Showing the ruler makes it easier to check the number of characters on one line.
Disp linenum	• [Checked] • Unchecked	Shows line numbers on the left side of the program input area. Showing line numbers makes it easier to identify lines where errors occur during debug.

Additional Information

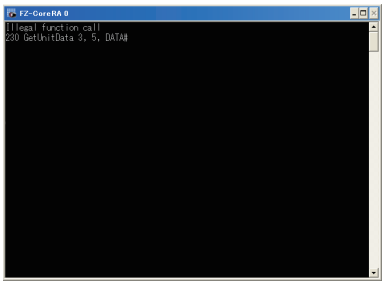
- You can increase the size of the program area and make the program easier to view by removing the checkmarks from the display settings. This will also improve the program editing response.
- The display settings are not saved. When the program editing screen is closed, the display settings revert to the factory default settings.
- When "Disp highlight" is selected and a function entered in a program cannot be used, the function name is shown in red.
  Whether or not a function can be used depends on the macro customize function that is used. For details, refer to *Section 9 Macro Reference List* on page 9-1.

3-1-2 Description of the System Status Console Window

Use the system status console window to debug macro customize programs and check error descriptions.

Structure of the System Status Console Window

The information shown in the system status console window is described below.



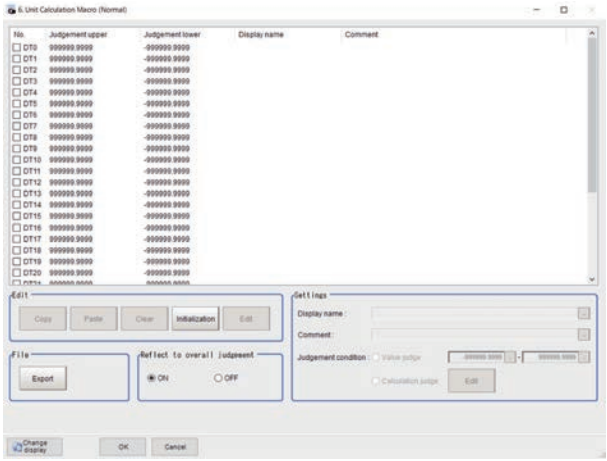
Display area	Description
System status display area	Shows the system status as text. When a program created with a macro customize function operates abnormally, a description of the error appears in the system status display area. For the information displayed, refer to *Checking the System Status Console Window* on page 6-5.

Precautions for Correct Use

- Do not close the system status console window by a method such as clicking the **x** button in the upper right corner of the system status console window. The system may not operate correctly. If the system status console window is accidentally closed, save your settings and restart the sensor controller.
- When displaying text in the System status monitoring console window, if the System status monitoring console window is minimized, processing will take more time.

3-1-3 Description of the Setting Screen for the Unit Calculation Macro Processing Item and How to Configure Settings

The components of the properties screen of the *Unit Calculation Macro* processing item are described below.



Selection of Operators (Unit Calculation Macro)

32 calculation processes from *DT0* to *DT31* can be set per unit.
The processing is executed in ascending order.

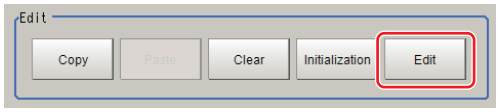
Additional Information

Calculation results cannot be output to external devices when you only set up macro calculations. When calculation results are output to external devices, set processing items related to results output in units after *Unit Calculation Macro* with flow editing. For details, refer to *Output Result* in the *Vision System FH/FHV Series Processing Item Function Reference Manual (Cat. No. Z341)*.

1 From the list, click the operator of the calculation processing to be set.

2 Place a check at the operator to use to perform the calculation processing.

3 In the Edit area, click **Edit**.



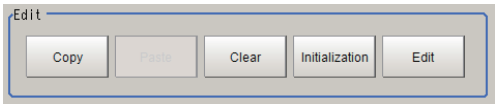
The unit calculation macro setting window is displayed.

Editing Operator (Unit Calculation Macro)

The calculation processing you have set can be copied or cleared.

1 From the list, click the operator of the calculation processing whose setting is to be edited.

2 Select each operation in the *Edit* area.



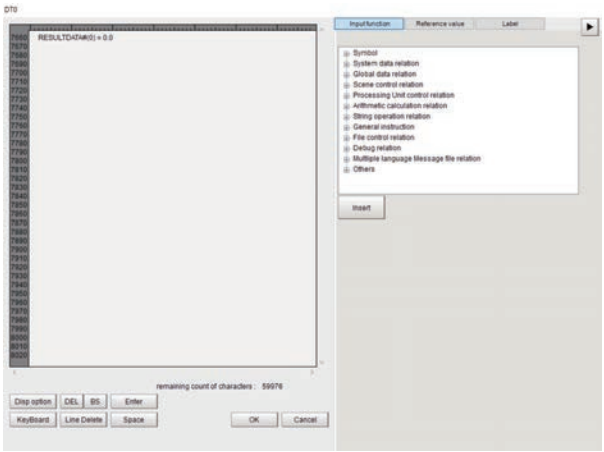
The unit calculation macro setting window is displayed.

Item name	Description
Copy	Copy the setting of the operator you have selected from the list. The copied setting can be pasted to other operator via Paste.
Paste	Paste the copied set value to the operator selected from the list. Data that can be pasted includes valid/invalid flags, calculation macro codes, display name, comment, judgement flag, upper/lower limits of figure judgement and judgement macro codes.
Clear	Initialize the setting of the operator you have selected from the list. Data to be initialized includes valid/invalid flags, calculation macro codes, display name, comment, judgement flag, upper/lower limits of figure judgement and judgement macro codes.
Initialization	Initialize the settings of all operators.
Edit	Edit the setting of the operator you have selected from the list.

Reference Variable Settings (Unit Calculation Macro)

Set the reference variables used in the program. The reference variables setting method is the same method as for the *Unit Macro* processing item.
For details, refer to *Reference Variable Settings (Unit Macro)* on page 3-23.

Program Input (Unit Calculation Macro)

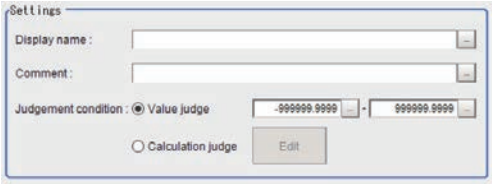


Note: RESULTDATA#() is the data identifier of the operation result.

The program input method is the same method as for the *Unit Macro* processing item.
For details, refer to *Program Input (Unit Macro)* on page 3-24.

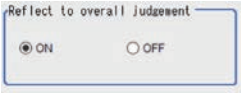
Judgement Condition Settings (Unit Calculation Macro)

1 Set judgement conditions in the Settings area.



Setting item	Setting value [Factory default]	Description
Display name	-	Enter the display name on the calculation processing selected in the list. The display name is also displayed in the Detail result area. If the display name is blank, it is displayed as *CalculationN*(N=0 to 31) in the Detail result area. Multilingual support is also possible. For multilingual support, refer to *Inputting text* in the *Vision System FH/FHV Series User's Manual (Cat. No. Z365)*.
Comment	-	Enter the comment on the calculation processing selected in the list.
Judgement condition	• [Value judge] • Calculation judge	Select whether to use a figure or macro judgement for the calculation result. If figure judgement selected, set the upper/lower limits of OK judgement. If macro judgement is selected, click **Edit** and define the calculation processing to be performed on the calculated value.

2 Select whether or not to reflect the judgement result in the scene overall judgement in Reflect to overall judgement area.



Setting item	Setting value [Factory default]	Description
Reflect to overall judgement	• [ON] • OFF	Specify whether or not the judgement results of this processing unit is reflected in the scene overall judgement.

Key Points for Test Measurement and Adjustment (Unit Calculation Macro)

To increase the speed and precision of measurement, you can adjust the parameters by performing and checking the results of test measurements.
The following content is displayed in the Detail result area as text.

Displayed items	Description
Judge	Judgement result
Calculation 0 display name	Calculation 0 value
Calculation 1 display name	Calculation 1 value
:	:
Calculation 31 display name	Calculation 31 value

The image specified in the sub image in image display setting is displayed in the Image display area.

Sub image.	Explanation of image to be displayed
0	Measurement image

Key Points for Adjustment (Unit Calculation Macro)

Select the adjustment method referring to the following points.

● An error message appears on the Console Window

Parameter to be adjusted	Troubleshooting
-	Refer to the *9-1 Error List* on page 9-2.

● Nothing happens when *DEL*, *BS*, *Enter*, etc., is clicked

Parameter to be adjusted	Troubleshooting
-	Nothing happens while the focus is not on the code window (key entry cursor is not displayed). Click the position you want to operate, and then click the button.

● Want to include a line feed code in a string

Parameter to be adjusted	Troubleshooting
Macro code	Add (+) CR → Chr$(13) LF → Chr$(10) to the string.

● Calculation result is indicated as "Unmeasured"

Parameter to be adjusted	Troubleshooting
Enabled/disabled	Place a check to enable the operator.
Calculation judgement	The judgement result may not be set correctly in calculation judgement.

Check the measurement results that can be displayed and output in *Measurement Results That Can be Output (Unit Calculation Macro)* on page 3-12.

Measurement Results That Can be Output (Unit Calculation Macro)

The measurement results provided by the unit calculation macro are shown below. The measurement results appear in the detailed result area, and you can also use a result output processing item to output the measurement results to an external device.
Refer to the *External Reference Tables (Unit Calculation Macro)* on page 3-12 for the parameters that can be referenced, including measurement results.

Measurement items	Character string	Description
Judge	JG	Judgement result
Calculation result 0	DT00	Calculation result 0
Judgement result 0	JG00	Judgement result 0
:	:	:
Calculation result 31	DT31	Calculation result 31
Judgement result 31	JG31	Judgement result 31

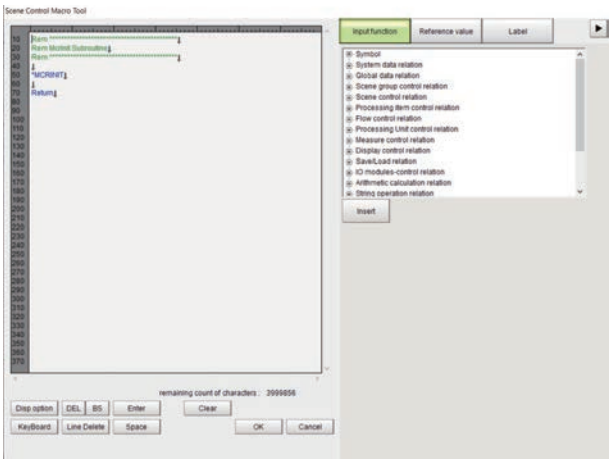
External Reference Tables (Unit Calculation Macro)

By specifying a number, you can access the following data from processing items that support processing unit data setting/acquisition, and from control commands.

No.	Data name	Set/Get	Data range
0	Judge	Get only	0: No judgement (unmeasured) 1: Judgement result OK -1: Judgement result NG
5 + N x 1 (N = 0 to 31)	Calculation result N (N = 0 to 31)	Set/Get	-99999.9999 to 99999.9999
37 + N x 1 (N = 0 to 31)	Judgement result N (N = 0 to 31)	Get only	0: No judgement (unmeasured) 1: Judgement result OK -1: Judgement result NG

3-1-4 Description of the Setting Screen of the Scene Control Macro Tool and How to Configure Settings

The setting screen for the scene control macro tool is the same as program editing screen. For details on the setting screen, refer to *3-1 Components of the Screens and How to Configure Settings* on page 3-2.



Reference Variable Settings (Scene Control Macro)

Set the reference variables used in the program. The reference variables setting method is the same method as for the *Unit Macro* processing item.
For details, refer to *Reference Variable Settings (Unit Macro)* on page 3-23.

Unit Label Settings (Scene Control Macro)

Setting a unit label allows you to specify a processing unit in a program by the label rather than the processing unit number. The unit label and Ut function is used to specify the processing unit. Using unit labels eliminates the need to change the program when processing unit numbers change due to a change of measurement flow.
Other macro customize functions can be used to reference unit labels.
For details, refer to Ut function in Macro Command Reference *Ut* on page 8-351.

1 Click the **Label** tab of the setting screen for the scene control macro tool.
A list of the unit labels set in the current scene appears.

2 Click **Edit**.
The Unit Label screen appears.

3 Select the processing unit for which you want to set a unit label.

4 Click ... and set the unit label name.

Setting item	Setting value [Factory default]	Description
Unit Label	1 to 32 characters	Set the unit label name. The unit label set here is used by the Ut function. You can use letters, numbers, ".", and "_" in the unit label name.

5 Click **OK**.

Precautions for Correct Use

- The unit label setting is saved in the scene data of the scene. If you want to use the unit label setting in another scene, repeat the setting in that scene, or use the scene maintenance function to copy the scene. For details, refer to *Editing Scenes* in the *Vision System FH/FHV Series User's Manual (Cat. No. Z365)*.
- The same unit label cannot be set twice in the same scene.

Program Input (Scene Control Macro)

The program input method is the same method as for the *Unit Macro* processing item.
For details, refer to *Program Input (Unit Macro)* on page 3-24.

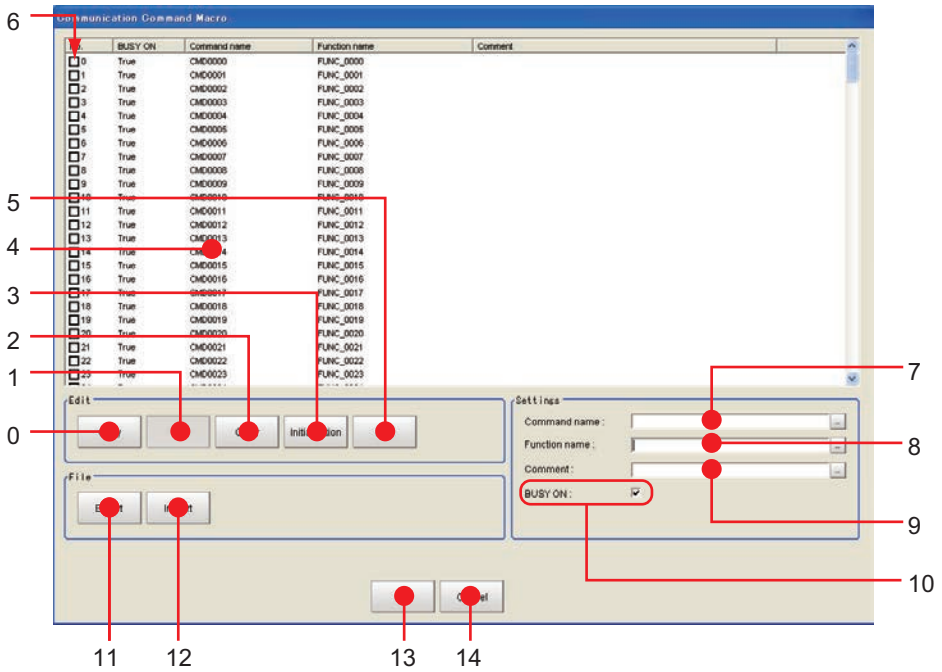
3-1-5 Components of the Setting Screen of the Communication Command Macro Tool and How to Configure Settings

The components of the setting screen of the communication command macro tool are indicated below.

Setting Procedure for Communication Command Macro

You can create and edit communication command macros.

You can create up to 256 (from 0 to 255) communication command macros.



● Descriptions of Dialog Box Objects

No.	Name	Description
0	Copy	Copy the selected command in the command list.
1	Paste	The selected command is overwritten by the copied information. Paste targets are "comments" and "processing details." If nothing is copied, this is disabled.
2	Clear	The information of selected command, such as "Command name", "Function Name", "Comment" and "Program" is initialized.
3	Initialization	The information of all of the commands is initialized.
4	Command list	Display the list of the commands.
5	Edit	Launch the Macro program editor for selected command.
6	Enable/Disable	Set/display whether custom command is enabled. If defined but not checked, it is not executed.
7	Command name	Display and edit command name for selected command.
8	Function name	Display and edit function name for selected command.
9	Comment	Display and edit comment for selected command. Multilingual support is also possible. For details, refer to *Inputting text* in the *Vision System FH/FHV Series User's Manual (Cat. No. Z365)*.

No.	Name	Description
10	Busy On	Set/display whether to change to measurement stop state (MeasureStop) before executing command. If checked, BUSY is turned ON while command is executing, and then after execution of command has finished, measurement stop state is released (MeasureStart). Afterwards, a MeasureInit event is raised.
11	Export	Export the macro program to file.
12	Import	Import the macro program from file. The existing data will be overwritten.
13	OK	Save change and return to Main window.
14	Cancel	Return to Main window without saving.

Selection of the Communication Command Macro to be Used (Communication Command Macro)

Select the command to define in the "No." column. Only the command numbers that are selected are enabled.

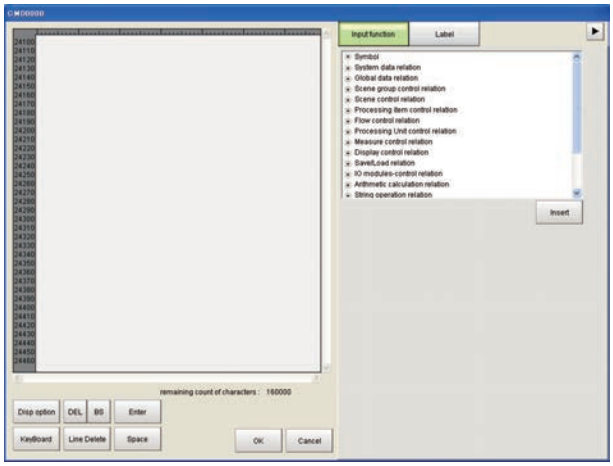
Communication Command Macro Name Setting (Communication Command Macro)

In the *Command name* field in the Settings Area, enter the *Command name* to use with the non-procedure communications protocol.

Flow Signal Output Setting (Communication Command Macro)

Use the *BUSY ON* check box in Settings Area in the lower right corner to specify whether the BUSY flag should be turned ON (TRUE) or OFF (FALSE) when the communication command macro that is being set is in execution.

Program Input (Communication Command Macro)



The program input method is the same method as for the *Unit Macro* processing item.
For details, refer to *Program Input (Unit Macro)* on page 3-24.

Specifying Communication Command Macro

The specified custom communications command is sent from an external device to the sensor controller. The method that you use to specify a command depends on the communications protocol.

● Non-procedure Protocol

Specify the name of the command as an ASCII character string.
Send the ASCII character string as *Command_name* + (space) + *Numeric_value*.
If you intend to use standard commands, do not use an existing command name.
Example: Specifying the Command Name *mycommand* with a Numeric Value of 18 as the Argument.
Send *mycommand 18*.

● Using the Parallel Interface

Specify the command number from 0 to 255 as a binary number. Specify the command number with the seven bits from DI0 to DI6. Then, turn ON the Command Execution Bit, DI7.
Because the command must be specified with seven bits, the method differs as shown below for command numbers *0 to 127* and command numbers 128 to 255.

- **For 0 to 127: Specify the number as is with the seven bits from DI0 to DI6.**
  Example: For command number 120, the binary notation for decimal 120 is 111 1000. Set each bit as follows: DI6: 1, DI5:1, DI4: 1, DI3: 1, DI2: 0, DI1:0, DI0: 0. Then, after 1 ms, change DI7 from OFF to ON to execute the command.
- **For 128 to 255: Use the terminal offset command *DIOFFSET* to add the difference from 0 to 127 with the seven bits from DI0 to DI6.**
  Example: For command number 150, the binary notation for 150 is 1001 0110, which requires eight bits.
  In this case, you use the terminal offset command DIOFFSET to add, for instance, half of 150, or 75. Then you can use DI0 to DI6 to specify the remaining 75. This procedure is given below.

1 Send DIOFFSET 75.

2 The binary notation for decimal 75 is 100 1011. Set each bit as follows: DI6: 1, DI5: 0, DI4: 0, DI3: 1, DI2: 0, DI1: 1, DI0: 1. Then, after 1 ms, change DI7 from OFF to ON to send the command.

3 Send DIOFFSET 0.
(0: OFF, 1: ON)

● Except for the Non-procedure Protocol and Parallel Interface (Using PLC Link, EtherCAT, EtherNet/IP, or PROFINET)

Specify the command number from 0 to 255 as a hexadecimal number from 00 to FF hex to represent the command code (CMD-CODE). Specify the command code in order from the upper digits to the lower digits of the hexadecimal number. This corresponds to the upper word address and the lower (smaller) word address in the I/O memory in the PLC. In this case, the highest number is FF hex, so we specify a number from 00 to FF hex to the first word in the Command Area + 2, and 0000 hex to the first word in the Command Area + 3.

Example: For command number 120, the hexadecimal notation is 0078 hex. Specify 0078 hex in the first word in the Command Area + 2, and 0000 hex in the first word in the Command Area + 3.

Common Behavior of Custom Commands

Basic sequence

Normally, each of IO commands are processed in the sequence as below:

1. Check input command and parameters are valid (range or type)
2. Body of the procedure
3. Output the result or response

The way of input/output command, parameters and response depends on the type of IO modules.

For detail, please refer the pages shown below.

- *Creating serial command* on page 3-20
- *Creating Parallel Command* on page 3-21
- *Creating PLC Link Command* on page 3-21
- *Creating Fieldbus command* on page 3-22

Control BUSY signal

Basically the BUSY flag on custom I/O command list window should be ON.

Precautions for Correct Use

Executing measurement (Measure command) with BUSY flag set to ON causes error.
If you need to combine measure command with the command which is necessary to set BUSY ON (For example, switching scene and executing measurement), please set BUSY flag OFF and write the program like this way:

MeasureStop	' Set BUSY ON (Forbid measurement)
CangeSceneArgumentValue#(0)	' Execute the command which can be used with BUSY ON condition
MeasureStart	' Set BUSY OFF (Permit measurement) before measurement
Measure	' Execute measurement

If *BUSY ON* is unchecked, the timing of BUSY signal at the execution of the communication command may differ from the timing described in User's Manual for Communications Settings.

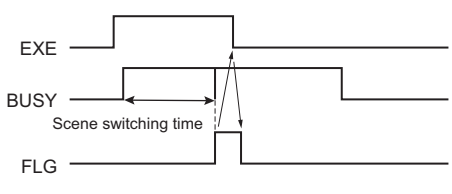
The example below shows the timing when scene switching and measurement are executed in series.

The timing changes depending on the MeasureStart and MeasureStop commands.

To set the BUSY signal ON during processing, send MeasureStop and MeasureStart commands.

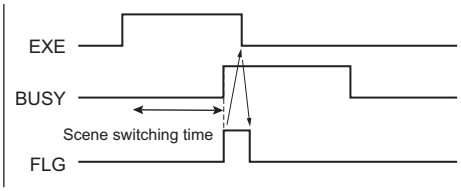
1. To prohibit measurement trigger input, allow scene switching, allow measurement trigger input and execute measurement.

```
Measure Stop↓
ChangeScene ArgumentValue#(0)↓
Measure Start↓
Measure↓
```



2. To execute scene switching by communication command macro and execute measurement.

```
ChangeScene ArgumentValue#(0)↓
Measure↓
```



Standard I/O commands

When the same command name / command id of standard I/O commands are used for custom I/O commands, custom I/O command has the priority and the standard I/O commands are not executed. If you execute the standard I/O commands after execution of custom I/O commands, please add the line as below.

```
CommandExecute&=False
```

In this case, standard I/O commands is executed just after executing custom I/O command.

Calling the procedure defined on the other commands

It is possible to call the procedure defined on the other commands during the command execution. Each command has *Function name*, and it is used to call the procedure.

Example: When command is defined as the table below and we intend to call procedure of CMD0 from CMD1.

Command No.	Command name	Busy	Function name
0	CMD0	False	FUNC0
1	CMD1	True	FUNC1

The codes for CMD1 should be like this:

```
Gosub *FUNC0
```

This case, the behavior of the command like BUSY depends on the caller (CMD1), and BUSY stays ON until the end of the procedure.

Precautions for Correct Use

Please be careful not to make the commands calling each other (In the case above, CMD0 also calls FUNC1), because it makes infinite loop.

Define the different procedure according to I/O module

The variable IoIdent$ stores the IO module identification name, which received the current I/O command.
When you define the different procedure for each I/O modules, please make branch by the value of IoIdent$.

Example: The command which receives *Serial* for serial command, and *Ethernet* for UDP normal

```
If IoIdent$ = "SerialNormal" Then
   Response$ = "Serial"
Elseif IoIdent$ = "UdpNormal" Then
   Response$ = "Ethernet"
Endif
```

Creating serial command

- **Command parameters**

  Received text string is split by space character(" ") into command and parameters, and stored in the predefined variables shown below:

Variable name	Type	Description
ArgumentsLength&	Integer	Number of parameters (0 to 32)
ArgumentString$()	Array of text string	Array of parameters (string)
ArgumentValue#()	Array of real numbers	Array of parameters converted to number Note: If conversion fails, set to 0.

```
"AAA param0 param1 param2"
```

When the system received the string as above, parameters are set like this :

ArgumentsLength&	: 3 (number of parameters)
ArgumentString$(0)	: param0 (String type)
ArgumentString$(1)	: param1 (String type)
ArgumentString$(2)	: param2 (String type)
ArgumentValue#(0)	: numeric value converted from param0 (0 when conversion failed)
ArgumentValue#(1)	: numeric value converted from param1 (0 when conversion failed)
ArgumentValue#(2)	: numeric value converted from param2 (0 when conversion failed)

Example: The command *SC1* that switches scene 1

```
SceneChange ArgumentValue#(0)
```

- **Response output**

  Result of the command procedure can be returned to the system by setting the value on these variables.

Variable name	Type	Description
ResponseString$	Text string	Output data
ResponseCode&	Integer	Result of command • 0: Success (Returns *OK*.) • -1: Command processing failed (Returns *ER*.) • -2: No response (No returns *OK or ER*.)

Precautions for Correct Use

When ResponseCode& is not set: Returns 0 as the result of command. Returns -1 if an error occurs.

Example: The command *TEST*

```
ResponseString$ = "TestString"
```

Command and response will be like this :

```
-> TEST
<- TestString
<- OK
```

Creating Parallel Command

- **Response output**

  Result of the command procedure can be returned to the system by setting the value on these variables.

Variable name	Type	Description
ResponseCode&	Integer	Result of command • 0: Command processing successful • -1: Command processing failed (The ERROR signal turns on.) • -2: No response

Precautions for Correct Use

When ResponseCode& is not set: Returns 0 as the result of command. Returns -1 if an error occurs.

Creating PLC Link Command

- **Command parameters**

  The command parameters are stored on the predefined variables as below.

Variable name	Type	Description
ArgumentsLength&	Integer	Number of parameters (0 to 6)
ArgumentValue#()	Array of real numbers	Array of parameters Note: integer type of data for 2 channels

- **Response output**

  Result of the command procedure can be returned to the system by setting the value on these variables.

Variable name	Type	Description
ResponseValue&()	Array of Integers	Response data Outputs the response data, i.e. Scene No.
ResponseCode&	Integer	Result of command • 0: Command processing successful • -1: Command processing failed • -2: No response

Precautions for Correct Use

When ResponseCode& is not set: Returns 0 as the result of command. Returns -1 if an error occurs.

Creating Fieldbus command

- **Command parameters**

  The command parameters are stored on the predefined variables as below.

Variable name	Type	Description
ArgumentsLength&	Integer	Number of parameters (0 to 3)
ArgumentValue#()	Array of real numbers	Array of parameters Note: integer type of data for 2 channels

- **Response output**

  Result of the command procedure can be returned to the system by setting the value on these variables.

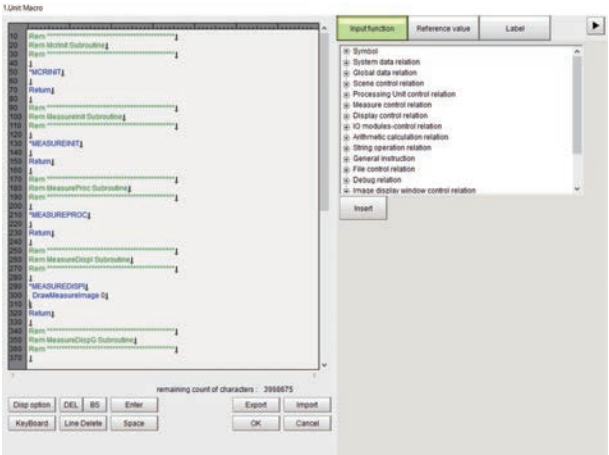
Variable name	Type	Description
ResponseValue&()	Array of Integers	Response data Outputs the response data, i.e. Scene No.
ResponseCode&	Integer	Result of command • 0: Command processing successful • -1: Command processing failed • -2: No response

Precautions for Correct Use

When ResponseCode& is not set: Returns 0 as the result of command. Returns -1 if an error occurs.

3-1-6 Description of the Setting Screen of the Unit Macro Processing Item and How to Configure Settings

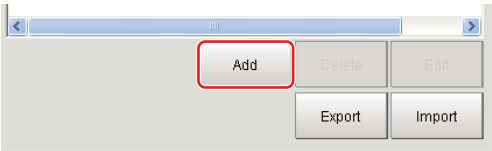
The properties screen of the *Unit Macro* processing item is the same as the program editing screen. For details on the setting screen, refer to*3-1-1 Components of the Program Editing Screen* on page 3-2.



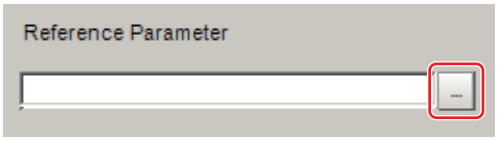
Reference Variable Settings (Unit Macro)

Set up the reference variables used for function.

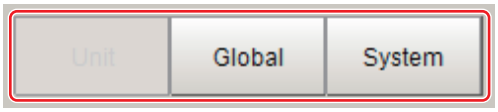
1 Click **Add** in the reference variable list on the macro setting screen.
A reference variable window is displayed.



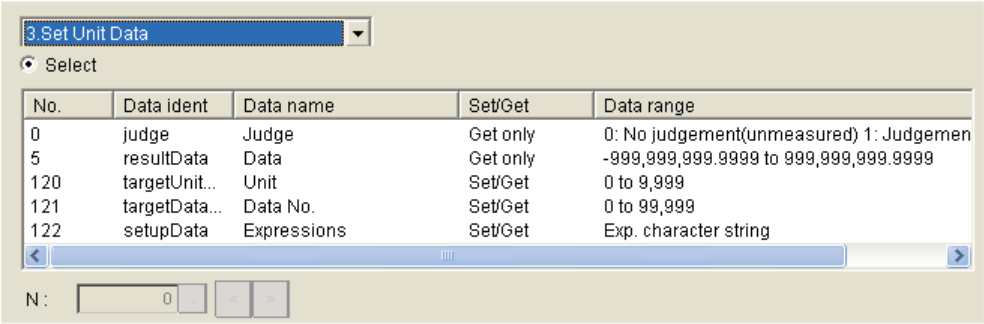
2 Click ... to set the variable name.
The variable name must consist of alphanumeric characters beginning with a capital letter.



3 Set the variable to be referenced.



- If *Unit* is selected:
  Select the processing item to be referenced, and then select the data to be referenced from the list.



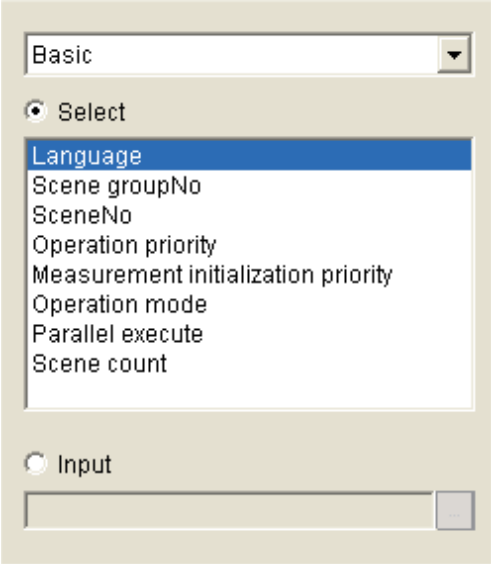
- If *Global* is selected:

  Click ... to set the Data ident of Global data. Data ident of Global data is defined with AddGlobalData function.
  Refer to *AddGlobalData* on page 8-26.

- If *System* is selected:

  Place a check at applicable **Select** to select the type of system variable, and then select the variable to be referenced from the list.

  To set a variable value, place a check at **Enter** and then click ... to set the variable value.



4 Click **OK**.

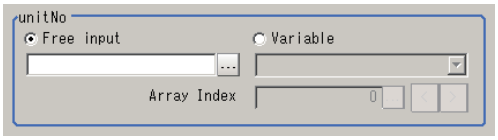
Unit Label Settings (Unit Macro)

Set the unit labels used in the program. The unit labels setting method is the same method as for the *Scene Control Macro*.
For details, refer to *Unit Label Settings (Scene Control Macro)* on page 3-13.

Program Input (Unit Macro)

1 From the function list, select the function to be inserted.
When the function is selected, an operand list appears below the function list.

2 Set the operand.



Setting item	Setting value [Factory default]	Description
Operand input method	• [Free input] • Variable	• Free input: Select this option if you want to enter an operand freely. Click ... to set the operand. • Variable: Select this option if you want to select an operand from variables. Click ▼ to select the variable. You can select any variable or reference variable currently defined in the macro code.
Array Index	0 to number of arrangements [0]	If the selected variable is an arrangement variable, set the arrangement number to be used as an operand.

3 Click **Insert**.

The set function is registered and appears in the code edit window in the top left-hand corner.

4 Repeat steps 1 to 3, and set the calculation processing.

5 When the setting of calculation processing is complete, click **OK**.

Key Points for Test Measurement and Adjustment (Unit Macro)

To increase the speed and precision of measurement, you can adjust the parameters by performing and checking the results of test measurements.
The following content is displayed in the Detail result area as text.

Displayed items	Description
Judge	Judgement result

The image specified in the sub image in image display setting is displayed in the Image display area.

Sub image.	Explanation of image to be displayed
0 to 100	Can be defined arbitrarily.

Key Points for Adjustment (Unit Macro)

Select the adjustment method referring to the following points.

● An error message appears on the Console Window

Parameter to be adjusted	Troubleshooting
-	Refer to the *9-1 Error List* on page 9-2.

● Nothing happens when *DEL*, *BS*, *Enter*, etc., is clicked

Parameter to be adjusted	Troubleshooting
-	Nothing happens while the focus is not on the code window (key entry cursor is not displayed). Click the position you want to operate, and then click the button.

● Want to include a line feed code in a string

Parameter to be adjusted	Troubleshooting
Macro code	Add (+) CR → Chr$(13) LF → Chr$(10) to the string.

● The "Positions" display is not as desired

Parameter to be adjusted	Troubleshooting
Macro code	When "Positions" is OFF, the display processing written in "*MEASUREDISPT" and "*MEASUREDISPG" is executed. When "Positions" is ON, only what is written in "*MEASUREDISPG" is displayed.

● Data acquisition sometimes fails during measurement

Parameter to be adjusted	Troubleshooting
Macro code	Data may be stolen by the applicable communication processing unless the communication processing is stopped using the SetPollingState function. Example: Receive data without TCP procedure: Write the processing in the following sequence. • SetPollingState "TcpNormal", false • Data receive processing • SetPollingState "TcpNormal", true

Check the measurement results that can be displayed and output in *Measurement Results For Which Output is Possible (Unit Macro)* on page 3-27.

Measurement Results For Which Output is Possible (Unit Macro)

To output the measurement result of the unit macro, assign the value of the unit macro calculation result to data output or the calculation processing item and output externally.

External Reference Table (Unit Macro)

The external reference table of the unit macro does not contain any data that can be referenced.

3-1-7 Saving and Loading Programs

Programs created using macro customize functions can be saved and loaded as scene data in the *Unit Macro* processing item and the *Unit Calculation Macro* processing item, and the settings can be saved in the sensor controller with **Data save**. With the exception of some functions, standalone programs can be saved and loaded. A standalone program can be handled as a file, and thus programs can be managed and analyzed on a computer using a text editor.

Saving and Loading Programs in the Unit Calculation Macro

In the *Unit Calculation Macro* processing item, programs can be saved and loaded to the scene data as a processing unit, and in the properties screen of the processing item, programs can be saved as standalone programs. A program saved in the *Unit Calculation Macro* processing item can be loaded using the properties screen of the *Unit Macro* processing item. Follow the procedure below to output the set calculation process to a file.

1. Click **Export** in the file area.
2. Specify the file save location and the file name, and click **OK**.

Precautions for Correct Use

- A standalone program cannot be loaded in the *Unit Calculation Macro* processing item. If loading is necessary, load scene data.
- The programs saved in the *Unit Calculation Macro* processing item include created programs and programs automatically generated by the processing item. Do not change an automatically generated program. If changed, the program may not load normally.

Saving and Loading Programs in the Scene Control Macro

In the scene control macro tool, a program can be saved when **Data save** is executed.

Saving and Loading Programs in the Communication Command Macro

In the communication command macro tool, a program can be saved when **Data save** is executed, and a standalone program can be saved and loaded in the tool setting screen.

● Saving a program

1. Click **Export** in the file area.
2. Specify the file save location and the file name, and click **OK**.

● Loading a program

1. Click **Import** in the file area.
2. Select the saved file (.mcr) that you want to load and click **OK**.

Precautions for Correct Use

The programs saved in the communication command macro tool include created programs and programs automatically generated by the tool. Do not change an automatically generated program. If changed, the program may not load normally.

Saving and Loading Programs in the Unit Macro

In the *Unit Macro* processing item, programs can be saved and loaded to the scene data as a processing unit, and in the properties screen of the processing item, standalone programs can be saved and loaded.

● Saving a program

1. Click **Export** in the file area.
2. Specify the file save location and the file name, and click **OK**.

● Loading a program

1. Click **Import** in the file area.
2. Select the saved file (.mcr) that you want to load and click **OK**.

4

Basics of Programming

4-1 Basic Idea of Programming 4-2
4-1-1 Basic Syntax 4-2
4-1-2 Constant 4-5
4-1-3 Variable 4-6
4-1-4 Scene variables and System variables 4-11
4-1-5 Operator 4-12
4-1-6 Expression 4-14

4-1 Basic Idea of Programming

Necessity of referring to this manual	Unit Macro	Communication Command Macro	Scene Control Macro	Unit Calculation Macro
	Required			

The macro customize functions of the FH series use an interpreter-type programming language. Unlike a compiler-type language, programs can be created in an interpreter programming language without the need for a special development environment to compile the program. In contrast to programming languages such as C, you can create and run programs on the FH without a development environment.
The programming language that is used for the macro customize functions is based on BASIC, which is also an interpreter-type programming language. The programming syntax of the macro customize functions is based on the syntax of BASIC, with the addition of some specialized syntax.
The syntax required to create programs for the macro customize functions is explained in the following, starting from basic syntax.

4-1-1 Basic Syntax

There are some rules which must be known as a minimum in order to create programs.
The examples below include the basic program syntax that is used in the macro customize functions.

Basic syntax examples:

```
*MEASUREDISPT

   Rem Displays text in the Detail result pane.
   DrawText "Judgment OK", 1, 0

Return
```

Types of syntax	Description
Character	Alphabetical characters, numbers, symbols, and special marks can be used as characters. For details, refer to *Character* on page 4-3.
Line	A unit that is composed of a line number and a statement is called a line. For details, refer to *Line* on page 4-3.
Line number	The number that is automatically assigned to each line when a program is loaded in the FH system is called a line number. For details, refer to *Line number* on page 4-3.
Statement	The program processing text written on each line is called a statement. For details, refer to *Statement* on page 4-3.
Label	A name assigned to a line in the program to enable identification is called a label. For details, refer to *Label* on page 4-4.
Subroutine	A part of a program that is enclosed by <label> - Return is called a subroutine. For details, refer to *Subroutine* on page 4-4.
Comment	Text that follows a Rem function or a single quotation is called a comment. For details, refer to *Comment* on page 4-4.

Character

Alphabetical characters, numbers, symbols, and the special marks below can be used as characters. Lower case alphabetical characters are only recognized as lower case when enclosed by double quotation marks ("). Lower case and upper case are not distinguished anywhere else. With the exception of text enclosed by double quotations (") and comments described below, use only half-width alphanumeric characters and halfwidth symbols.

Special marks that can be used	Description
Colon (:)	Use as a separator when writing two or more lines as a single line.
Comma (,)	Use as a separator when listing parameters.
Semicolon (;)	Use as a separator when listing parameters in output text.
Apostrophe (‘)	Add in front of a comment. A Rem statement can also be used to indicate a comment.
Asterisk (*)	Add in front of a label name.
Space ( )	Always insert a space between a macro function and its arguments. Spaces can also be inserted wherever desired. However, a space must not be inserted inside a macro function name, variable name, or value.
Double Quotation Marks (")	Use to enclose a character string value.
Ampersand (&)	Use as a type identifier for integer type variables. Always add after an integer type variable name or array name. For details on type identifiers, refer to the *Naming Rules for Variables* on page 4-7.
Pound Sign (#)	Use as a type identifier for real number variables. Always add after a real number variable name or array name. For details on type identifiers, refer to the *Naming Rules for Variables* on page 4-7.
Dollar Mark ($)	Use as a type identifier for character string type variables. Always add after a character string type variable name or array name. For details on type identifiers, refer to the *Naming Rules for Variables* on page 4-7.
"At" mark (@)	Use as a type identifier for reference variables. Always add at the end of a reference variable name. For details on type identifiers, refer to the *Naming Rules for Variables* on page 4-7.

Line

A unit that is composed of a line number and a statement is called a line. Blank lines and lines that are only comments are acceptable in a program. Multiple statements (multi-statement) can also be written on one line. When writing multiple statements, separate each statement with a colon (:).

Line number

The number that is automatically assigned to each line when a program is loaded in the FH system is called a line number. Line numbers are mainly used during program debugging to identify the locations of errors. Do not write line numbers when creating a program.

Statement

The program processing text written on each line is called a statement. A statement is mainly composed of expressions of minimum processing units. A statement can be up to 245 characters in

length. If a statement is longer than 245 characters, an error will occur and program execution will stop.
Three types of statements exist, as shown below.

Statement type	Description
Execution statement	A statement that describes and executes processing of a macro function.
Non-execution statement	A statement that indicates comments but does not execute processing.
Label	A statement that defines a program branch destination.

Label

A name assigned to a line in the program to enable identification is called a label. Among the basic syntax examples, the following is a label.

```
*MEASUREDISPT
```

By specifying a label, the "Goto <label>" macro function can be used to jump to the specified label position.

Subroutine

A part of a program that is enclosed by <label> - Return is called a subroutine. Write the process that you want to execute in the subroutine. Among the basic syntax examples, the following is a subroutine.

```
*MEASUREDISPT

Return
```

Subroutines can be executed by calling the "GoSub <label>" function. By writing each standalone process as a subroutine, the visibility of the program is improved.
The processing component of a macro customize function consists of subroutines. By writing "GoSub*(function name)", the processing component of the corresponding macro customize function can be called and executed from another subroutine.
Some of the subroutines of macro customize functions are predefined in the system. The timing at which system defined subroutines are called is fixed. Write the process in a subroutine appropriately for its purpose. For details on calling these subroutines, refer to *5-1-4 State Transitions and Execution Timing* on page 5-13.

Comment

Text that follows a Rem function or a single quotation is called a comment. You can insert any comment line in a program. When a Rem function or single quotation (') is written in a program, the line is treated as a comment.
Among the basic syntax examples, the following is a comment.

```
Rem Displays text in the text result field.
```

Precautions for Correct Use

- Do not write comments and statements on the same line. The comments will not be recognized correctly and the program may malfunction.
- You can add comments in the program area.
  Note that allowable character strings differ depending on the type of the sensor controller as shown below.
  English characters and characters for the language selected in *Language setting* are allowed.

4-1-2 Constant

Among the values and character strings used in programming, a constant is a value that never changes and has a unique assigned name. Use constants for fixed numeric values and character strings that you want to use repeatedly in a program.

How to Use Constants

Constants are used as shown below.

Example:

```
A& = 255
AA& = &h7f

B# = 3.14

C$ = "TEST STRING"
```

Constant Data Types

Constants that can be used in macro customize functions are shown below.

Data type	Description	Data range	Number of bytes per data item
Integer	Used for signed integer values.	-2147483648 to 2147483647	4 bytes
Real number	Used for double precision type real numbers.	-1.0E30 to 1.0E30	8 bytes
Character string	Used for character strings.	Up to 255 characters	-

Integer constants can be written in several bases, including decimal. Base notations that can be used in macro customize functions are shown below.

Base	Notation method	Example	Mathematical notation
Decimal	Not required	100, 3456	100, 3456
Hexadecimal	&h	&hff, &h7fff	$(ff)_{16}$, $(7fff)_{16}$
Octal	&o	&o77, &o3447	$(77)_8$, $(3447)_8$
Binary	&b	&b1111, &b01100111	$(1111)_2$, $(01100111)_2$

Additional Information

When a program that uses hexadecimal constants is displayed using the List function, the hexadecimal notation is converted to decimal notation.
Example: When a program with *A& = &hff* is displayed using the List function, this is shown as *A& = 255*.

4-1-3 Variable

Among the values and character strings used in programming, a variable has a unique assigned name and is used for data that changes. Use a variable for a numeric value or character string that you want to use repeatedly in a program, but whose specific value or character string changes.

How to Use Variables

Variables are used as shown below.

Example:

```
Rem 1.Declare the variable
Dim POSITION#(1)

Rem 2.Assign a value to the variable
A# = 320.0
B# = 310.0

Rem 3.Reference the variable
POSITION#(0) = A#
POSITION#(1) = B#
C# = POSITION#(0) + POSITION#(1)
```

1. Declare the variable
   By declaring a variable, you allocate the data area required for the variable and enable use of the variable in the remainder of the program. Declaration is only required for array variables, and the Dim function is used for this purpose. For variables other than array variables, the data area is automatically allocated when the program is executed, and thus declaration is not necessary.
   In addition, you can check undefined or duplicate variables using the Macro variable check function. Refer to the *Macro Variable Check Function* on page 4-10.
2. Assign a value to the variable
   Assign a value to a variable. Only values of the same data type as the variable can be assigned. In the case of a variable other than an array variable, the variable can be used without being declared. Add the type identifier to the end of the name, and use a variable name that is not used by any other variables.
   For details on type identifiers, refer to the *Naming Rules for Variables* on page 4-7.
3. Reference the variable
   Reference the value that is assigned to a variable. You can also pass the variable to a macro function as an argument.

Variable Data Types

Before a variable can be used, the data area that will hold the data must be allocated. A data type defines the size of the data area to be allocated and how the data is handled.
Data types and data ranges of variables that can be used in macro customize functions are described below. Select the data type based on the use and objective of the variable.

Data type	Description	Data range	Number of bytes per data item
Integer	Used for signed integer values.	-2147483648 to 2147483647	4 bytes
Real number	Used for double precision type real numbers.	-1.0E30 to 1.0E30	8 bytes
Character string	Used for character strings.	Up to 255 characters	-
Variant type	Infinite length binary data can be handled.	-	-

Variable area size: Variable length

Precautions for Correct Use

In versions earlier than 5.20, variant data types cannot be used due to it causing a Syntax error.

Naming Rules for Variables

Rules exist for variable names. Decide variable names based on the rules below.

Variable name		
Start	**Middle**	**End**
Alphabet ('A' to 'Z', 'a' to 'z')	Alphabet, numbers, Symbols ('A' to 'Z', 'a' to 'z', '0' to '9', underline '_', period '.'))	Type identifier ('&', '#', '$', '@')*1

*1. "@" is a type identifier for reference variables, and cannot be used for variables that the user directly declares in the program.

An identifier that identifies the type of data used as a variable is referred to as a "type identifier". Always add a type identifier to the end of the variable name of each variable. For the type identifiers that can be used, refer to *Variable Types and Type Identifiers* on page 4-8.

Precautions for Correct Use

- A variable name that starts with a number cannot be used.
- A special mark cannot be used in a variable name.
- A variable name that is reserved cannot be used. However, a variable name that includes a reserved variable name can be used.
  Example: (Cannot be used) WAIT& → (Can be used) WAITTIME&
- Letters are not case sensitive.
  Example: AA& and aa& are treated as the same variable.

Variable Types and Type Identifiers

Variable types that can be used in macro customize functions are shown below. Use a variable type that matches your use and purpose.

Temporary Variable

This can handle one datum and is the most basic variable.
Type identifiers that can be used with temporary variables are shown below.

Data type	Type identifier	Example
Integer	&	A& = 1
Real number	#	A# = 12.34
Character string	$	A$ = "OMRON"
Variant type	!	A! = 1 A! = 12.34 A! = "OMRON"

Precautions for Correct Use

The type identifier cannot be omitted. Always add a type identifier to the end of the variable name.

Array Variable

When you want to handle multiple data items of the same data type as a group, you can assign numbers to temporary variables. Such temporary variables are called array variables. Macro customize functions enable the use of up to four dimensions of array variables. The element number of an array starts from 0, and the number of elements is "element number + 1". You can also change the number of elements in array variables using the ReDim function.

Example:

```
Rem One-dimensional array with four elements A&(0), A&(1), A&(2), A&(3)
Dim A&(3)

Rem Two-dimensional array with 11 x 11 = 121 elements
Dim B&(10, 10)

Rem Three-dimensional array with 101 x 101 x 101 = 1030301 elements
Dim C&(100, 100, 100)

Rem Array definition with undefined number of elements.
Dim AA&()

Rem Change the number of elements for defined array variables.
Dim AA&()
ReDim AA&(10)
Dim BB&(3)
ReDim BB&(10)
```

Type identifiers that can be used with array variables are shown below.

Data type	Type identifier	Example
Integer	&()	A&(1) = 1
Real number	#()	A#(2) = 12.34
Character string	$()	A$(3) = "OMRON"
Variant type	!()	A!(1) = 1 A!(2) = 12.34 A!(3) = "OMRON"

Precautions for Correct Use

- Even if the name is the same, an array variable name and a regular temporary variable name are distinguished.
  Example: The variables below are all distinguished.
  A&, A&(10), A#(10), A$, A$(10)
- Even if the number of dimensions is different, two arrays with the same variable name are treated as the same array. In this case, the array defined last is effective.
  Example: The last defined A&(10.10.10.10) is enable.
  A&(10), A&(10,10), A&(10,10,10), A&(10,10,10,10)
- Using the ReDim function without defining the number of elements will cause a Syntax error.
- Defining arrays without defining the number of elements, or using the ReDim function will cause a Syntax error on the FH sensor controller versions earlier than ver.5.40.

Reference Variables

Reference variables are defined by the user and can be used to reference processing unit data, global data, and system data. (Note: When using the macro customize functions, a macro function can be used to set and acquire data such as processing unit data and system data. Reference variables allow you to handle processing unit data and system data without using the macro function.)
Reference variables must be set in advance in the program editing screen of the Unit Macro processing item, Unit Calculation Macro processing item, or the scene control macro tool. For the setting method, refer to the setting screen of each processing item.
Type identifiers that can be used with reference variables are shown below.

Data type	Type identifier	Example
Integer	@	A@ = 1
Real number	@	A@ = 12.34
Character string	@	A@ = "OMRON"

Precautions for Correct Use

- The data type (integer, real number, etc.) can be identified from the type identifier of a regular variable, however, identification of the data type from the type identifier is not possible with a reference variable. When using a reference variable, check the data type of the allocated data before using the variable.
- A reference variable setting is saved in scene data, however, the value itself that the reference variable references is not saved. If you want to save the value, use user data.
- When processing unit data or get only system data is used as a reference variable, if you assign it to the reference variable, an Illegal function calll error will occur.
- When you set the Global data to the Variable whose type is integer or double precision real number variable, use Val function (*Val* on page 8-352).

Additional Information

- If you set the name of a reference variable in the properties screen the type identifier of the reference variable is automatically added to the reference variable name.
- There is no limit on the number of reference variable settings that can be added. However, there is a limit on the total number of variable name characters per processing unit of the unit macro and unit calculation macro, and the number of variable name characters per scene of the scene control macro.
  When the reference variable name is 10 characters, about 100 reference variable settings can be added.

Macro Variable Check Function

The Macro Variable Check function is a capability of the macro customization function to check undefined or duplicate variables.
You can define a variable as a Dim variable, and execute the Option Explicit command to be able to find undefined or duplicate variables afterwards. The **Undefined variable** error message is shown for undefined variables, and the**Duplicated variable** error message is shown for duplicate variables in the System Status Console Window.
The Macro Variable Check function can be used with the Unit Macro processing item, Communication Command Macro tool, and Scene Control Macro tool.
Support version of the FH sensor controller is ver.5.40 or later.
(Refer to *Dim* on page 8-73, *Option Explicit* on page 8-217, *ReDim* on page 8-233.)

● Defining the Dim variable

When you execute the Option Explicit command, only defined variables will be available to use, and duplicate of the defined variables will be checked.
Therefore, you need to define temporary variable, array variable, and reference variable in the Dim variable format.
Note that you cannot assign value to the variables at the same time when you define them.

Example:

```
Rem Check of macro variables
Option Explicit
Rem Dim Definition of variables by the Dim variable
Dim A&
Dim B&
Dim C&
A& = 0
B& = 1
C& = A& + B&
Print Str$(C&)
```

Precautions for Correct Use

Using the Dim variable on the FH sensor controller versions earlier than ver.5.40 causes the Syntax error. Note that using the Dim array variables does not cause error since it has already been used on the earlier versions.

● Executing the Option Explicit command

When you execute the Option Explicit command, the Macro Variable Check function will be effective for lines after the line where the Option Explicit command is executed.
Undefinition and duplicates of variables in preceding lines will not be checked.
In other word, they are not the subject to the Undefined variable error or Duplicated variable error.

Precautions for Correct Use

- The Option Explicit command is unavailable when the power is on or when macros are loaded. It is available from the time a macro and the Option Explicit command are both executed until the next macro is loaded or the power is turned off.
- Using the Option Explicit command on the FH sensor controller versions earlier than ver.5.40 causes the Syntax error.

4-1-4 Scene variables and System variables

Scene variables and system variables defined in **TDM Editor** can be used in the program.
For more information on how to use TDM Editor, Scene variables and System variables, see the *Using Variables to Edit the Flow [TDM Editor]* in the *FH/FHV Series Vision System User Manual (Z365)*.

How to use Scene variables and System variables

Scene variables and System variables can be used as shown in the example below.
You can use the variable name defined in TDM editor. It is not neccessary to declare variables in the program.
The variable name displayed in TDM editor is displayed as (**. Variable name ***) with the type of variable and type identifier appended.
For Variable type, there are two types defined, SC (Scene variable) and SY (System variable).
For Identifier type, see the *Using Variables* in the *FH/FHV Series Vision System User Manual (Z365)*.

Example:

```
SC.AA& = 12
SY.BB& = 34

Print SC.AA&
Print SY.BB&
```

Precautions for Correct Use

Scene variables and system variables cannot be set directly in the arguments to store the results of the macro commands.
When storing the result of a macro command in a scene variable or a system variable, store the value in another variable once and store that variable in the scene variable or the system variable as shown below.

```
GetUnitData 1, 5, TEMP&
SC.AA& = TEMP&
```

4-1-5 Operator

A symbol that indicates an operation in a program is referred to as an operator. Operators are used to add a process to a variable, as well as to calculate and compare variables and constants.

How to Use Operators

Operators are used as shown below.
Example:

```
A& = 0                          (1)
B& = 2
C& = 4
D& = 35
E& = 0

A& = 1 + 2 + 4 / 2              (2)

F& = (A& + B&) * C&             (5)
                                (3)
If F& < D& Then
   E& = D& AND 31               (4)
Else
   E& = F& AND 31
EndIf
```

1. Assignment Operator
   An assignment operator is used to assign a value to a variable.
   Assignment operators that can be used in macro customize functions are shown below.

Operator	Description of operation	Example
=	Assigns the value on the right side to the left side	A& = B& + C&

Additional Information

- When "=" is used in a conditional comparison such as an "If - Then" or "Select" statement, "=" is treated as a relational operator that determines whether the left side and right side are equal.
- When a real number variable value is assigned to an integer variable, the digits to the right of the decimal point are rounded off.
- If the value assigned to an integer variable is other than -2147483648 to 2147483647, an overflow error will occur when the assignment takes place.
- If the value assigned to a real number variable is other than -1.0E30 to 1.0E30, an overflow error will occur when the assignment takes place.

2. Arithmetic Operator
   An arithmetic operator performs an arithmetic operation, exponent operation, or remainder operation on numerical value data, Division by 0 results in an error. If the interim result of an arithmetic operation such as addition, subtraction, or multiplication is outside the range -1.0e30 to 1.0e30, an error will result.
   Arithmetic operators that can be used in macro customize functions are shown below.

Operator	Description of operation	Example	Mathematical notation
+	Addition	A& + B&	A+B
-	Subtraction	A& - B&	A-B
*	Multiplication	A& * B&	AxB or AB
/	Division	A& / B&	A/B
^	Exponent operation	A& ^ B&	A^B
mod	Remainder	A& MOD B&	A-[A/B]xB [] is the Gauss symbol

3. Relational Operator
A relational operator compares two numerical data items or two character data items. If the result of the comparison is true, (-1) is returned. If false, (0) is returned. Normally this is used in an "If - Then" statement for such purposes as controlling the flow of the program.
Relational operators that can be used in macro customize functions are shown below.

Operator	Description of operation	Example
=	Equal	A& = B&
<>, ><	Not equal	A& <> B&, A& >< B&
<	Less than	A& < B&
>	Greater than	A& > B&
<=, =<	Less than or equal to	A& <= B&, A& =< B&
>=, =>	Greater than or equal to	A& >= B&, A& => B&

Additional Information

When "=" is used in other than a conditional comparison such as an "If - Then" or "Select" statement, "=" is treated as a assignment operator that assigns the value on the right side to the left side.

4. Logic Operator
A logic operator is used to investigate multiple conditions, and perform bit operations and binary operations on exponential values.
Logic operators that can be used in macro customize functions are shown below.

Operator	Description of operation	Example
NOT	Not	NOT A&
AND	Logical AND	A& AND B&
OR	Logical OR	A& OR B&
XOR	Exclusive OR	A& XOR B&

5. Operation Order of Operators
When multiple operators are included in one expression, operations are executed in the order of priority of the operators. If you want to control the order of the operations, enclose the operations you want to perform first in parentheses.
The order of priority of the operators is shown below.

Order of priority	Operator
1	Operation enclosed in parentheses
2	Macro function

Order of priority	Operator
3	Exponent operation (^)
4	Minus sign (-)
5	Multiplication and division (*, /)
6	Remainder (mod)
7	Addition and subtraction (+, -)
8	Relational operators (< >, =, etc.)
9	NOT
10	AND
11	OR
12	XOR

4-1-6 Expression

Constants, variables, and the operators that join them are referred to as an expression. Not only joined constants and variables, but the constants and variables themselves are expressions, and expressions and combinations of expressions form a statement.

Numerical expression

An expression that returns a numerical value is referred to as a numerical expression. This joins numerical constants, numerical variables, and macro functions that return numerical values with arithmetical operators and logic operators. Multiple numerical expressions can be joined by enclosing the expressions in parentheses.

Example of a numerical expression:

```
A& = 1 + 2 + 4 / 2
```

Character expression

An expression that returns a character string is referred to as a character expression. This joins character string constants, character string variables, and macro functions that return character strings with plus signs. Multiple character expressions can be joined by enclosing the expressions in parentheses.

Example of a character expression:

```
B$ = "OMRON" + "FH"
```

Relational expression

An expression that joins two numerical expressions by a relational operator is referred to as a relational expression.

Example of a relational expression:

```
If A& < 10 Then

EndIf
```

Logical Expression

An expression that joins multiple relational expressions by a logical operator is referred to as a logical expression. This is used to execute bit operations and binary operations, and to evaluate complex conditions. When the operation result of a logical expression is other than 0, the result is treated as true, and when 0, the result is treated as false. Because a logical expression returns a numerical value, it can also be treated as a numerical expression.

Example of a logical expression:

```
A& = D& AND &b110000
```

Function

An expression that executes a predefined command or operation, or an expression that executes a predefined operation on a specific specified value (argument) and returns the result of the operation, is generally referred to as a function. In particular, the functions that can be used in the macro customize functions are referred to as macro functions. Macro functions include functions that do not return a value, and functions that return the numerical value or character string that is the result of the operation. Macro functions that do not return a value are written with the macro function name and argument separated by a space. Macro functions that return a value add an argument enclosed by parentheses ( ) to the end of the macro function name.

Example of a macro function that does not return a value:

```
ChangeScene 1
```

Example of a macro function that returns a value:

```
C# = Abs(-10)
```

5

Macro Programming

5-1 How to Write Macro Programs 5-2
5-1-1 Data Types Related to Processing Units 5-2
5-1-2 Data Types Related to the System 5-7
5-1-3 Scope of Data and Save Area 5-10
5-1-4 State Transitions and Execution Timing 5-13
5-1-5 Exclusive Control in a Process 5-20

5-1 How to Write Macro Programs

Macro customize functions can be used for a variety of purposes depending on the content being programmed. This section explains programming techniques that can be used together with the basic program writing methods to widen the range of application of the macro customize functions.

5-1-1 Data Types Related to Processing Units

Necessity of referring to this manual	Unit Macro	Communication Command Macro	Scene Control Macro	Unit Calculation Macro
	As needed			

A macro customize function program can be used to change measurement flow settings by setting or acquiring processing unit data such as external reference data, figure data, and model data. The data access method varies depending on the data type.

Type	Description
External reference data	Data used to set or acquire settings and measurement values of a processing unit. For details, refer to *External reference data* on page 5-2.
Figure data	Data used to set and acquire region figures and model figures of a processing unit. For details, refer to *Figure data* on page 5-3.
Model data	Data used to set the model registration of a processing unit. For details, refer to *Model data* on page 5-5.
Image data	Image data held by the processing unit itself. For details, refer to *Image data* on page 5-6.

External reference data

External reference data is used to set and acquire settings and measurement values of a processing unit. Unique numbers are assigned to the external reference data within the processing item, and thus data can be set and acquired by specifying the processing unit number and external reference data number.
In addition to a macro customize program, external reference data can be set and acquired from a processing unit data setting item, processing unit data acquisition item, and a communication commands.
A processing unit data setting macro function or processing unit data acquisition macro function is used to set or acquire external reference data from a macro customize program. Numerical data and character string data can be set and acquired in a processing unit data setting macro function or processing unit data acquisition macro function. However, figure data such as measurement region data, model data such as search model and model edge data, and image data such as camera images and measurement filter images cannot be set or acquired by this method. Refer to the figure data, model data, and image data sections.

● Acquisition of Data

External reference data is acquired by the methods below.

Example: Acquiring External Reference Data No. 5 (correlation value) of the Unit No. 1 Search processing unit

```
Rem The correlation value is a real number, and thus the variable type identifi
er is #
GetUnitData 1, 5, DATA#
```

Example: Acquiring External Reference Data No. 7 (decode character string) of the Unit No. 2D code code processing unit

```
Rem The decode character string is a character string, and thus the variable ty
pe identifier is $
GetUnitData 2, 7, DATA$
```

● Data Settings

The method of setting external reference data is as follows.

Example: Setting 1 in External Reference Data No. 147 (search number) of the Unit No. 1 search processing unit

```
SetUnitData 1, 147, 1
```

Example: Setting "comparison" in External Reference Data No. 300 (index 0 comparison character string) of the Unit No. 2 two-dimensional code processing unit

```
SetUnitData 2, 300, "comparison"
```

In addition to the external reference data number, external reference data can also be set and acquired using the external reference data identification name.

Example: Setting "1" in the external reference data ID name "searchNo" (search number) of the Unit No. 1 search processing unit

```
SetUnitData, "searchNo", 1
```

The result is the same as when "1" is set in External Reference Data No. 147.

Figure data

Figure data is used to set and acquire region figures and model figures of a processing unit. When there are multiple figure data items in the processing item, a unique number is assigned to each figure data item. This makes it possible to set and acquire figure data by specifying the processing unit number and figure number.
In addition to a macro customize program, figure data can be set and acquired from a processing unit figure setting item and a processing unit figure acquisition item.
A processing unit figure setting macro function or processing unit figure acquisition macro function is used to set or acquire figure data from a macro customize program.
Refer to *9-7 List of Figure Numbers* on page 9-73.

● Format of Figure Data

Figure data is specified using an array variable. The elements of the array are described below.

Array element	Description	
figure(0)	Figure data header information	This is figure data header information. Includes the number of figures and figure data size information. Upper 16 bits: Number of figures Lower 16 bits: Number of bytes of figure data size (figure array length x 4) Figure data header information = Number of bytes of figure data size + Number of figure data x 65536 Number of figures: Sets the number of figures included in the figure data. Normally 1 should be set. If you are combining multiple figures, set the number of figures that are combined (2 or more). Number of bytes of figure data size: Set the figure data size converted into bytes. Set the value that is 4 times the number of array elements. Example: One rectangle Array length = 5, Number of figure data items = 1 Number of bytes of figure data size = 5 x 4 = 20 Figure data header information = 20 + 1 x 65536 = 65556
figure(1)	Figure 0 type information	Type information of figure 0 data. Includes drawing mode and figure type information. Upper 16 bits: Drawing mode Lower 16 bits: Figure type Figure type information = Figure type + Drawing mode x 65536 Drawing mode: Set whether the figure drawing mode is OR mode or NOTE mode. Normally 0 (OR mode) should be set. When multiple figures are used and you want to exclude some of the figures, set 1 (NOT mode) for the 2nd or later figures. Figure type: Set the figure type (line, rectangle, etc.). Example: One rectangle (drawing mode is OR) Figure type = 8, Drawing mode = 0 Figure type information = 8 + 0 x 65536 = 8
figure(2)	Figure 0 data	Figure data of figure 0. The size and content depends on the figure type.
figure(M)	Figure 1 type information	Type information of figure 1 data.
figure(M+1)	Figure 1 data	Figure data of figure 1. The size and content depends on the figure type.
:	:	:
figure(N*M)	Figure N type information	Type information of figure N data.
figure(N*M+1)	Figure N data	Figure data of figure N. The size and content depends on the figure type.

Acquisition of Data

Figure data is acquired by the method below.

Example: Acquiring Figure Data No. 1 (region figure) of the Unit No. 1 Search processing unit

```
Dim FIGURE&(5)
GetUnitFigure 1, 1, FIGURE&()
```

Additional Information

If the number of array elements is insufficient for the figure data to be set or acquired, an error will occur when setting or acquiring is attempted.

Data Settings

The method of setting figure data is as follows.

Example: Changing the upper left coordinates of Figure Data No. 1 (region figure) of the Unit No. 1 Search processing unit to (100, 50)

```
Dim FIGURE&(10)
GetUnitFigure 1, 1, FIGURE&()

FIGURE&(2) = 100
FIGURE&(3) = 50

SetUnitFigure 1, 1, FIGURE&()
```

Example: Changing the ellipse x direction radius of Figure 1 of Figure Data No. 0 (region figure (Figure 0: rectangle, Figure 1: Ellipse)) of the Unit No. 2 defect processing unit to 100

```
Dim FIGURE&(10)
GetUnitFigure 2, 0, FIGURE&()

Rem FIGURE&(0): Figure data header information
Rem FIGURE&(1): Figure 0 type information
Rem FIGURE&(2) to FIGURE&(5): Upper left XY coordinates and lower right XY coor
dinates of rectangle
Rem FIGURE&(6): Figure 1 type information
Rem FIGURE&(7) to FIGURE&(8): XY coordinates of center of ellipse
Rem FIGURE&(9) to FIGURE&(10): X direction radius and Y direction radius of ell
ipse
FIGURE&(9) = 100

SetUnitFigure 2, 0, FIGURE&()
```

Model data

Model data is used to set the model registration of a processing unit. When there are multiple model data items in the processing item, a unique number is assigned to each model data item. This makes it possible to set model data by specifying the processing unit number and model number.
In addition to a macro customize program, model data can also be set by communication commands.
A processing unit figure setting macro function or model copy macro function is used to set model data from a macro customize program.
Refer to *9-8 Model Number List* on page 9-76.

Precautions for Correct Use

Model data has a close association with the model figure and model parameters. Before re-registering a model, set the model figure and model parameters.

Acquisition of Data

Execution of only model data acquisition is not possible.

Data Settings

The data setting method is as follows.

Example: Re-registering the model of the Unit No. 1 Search processing unit (model figure: rectangle) in the current measurement image, and changing the reference position (0, 0) and detection point coordinates (320, 240)

```
Dim FIGURE&(5)
GetUnitFigure 1, 0, FIGURE&()

Rem Re-register the model by setting the model figure
SetUnitFigure 1, 0, FIGURE&()

Rem Update reference coordinates XY
SetUnitData 1, 129, 0
SetUnitData 1, 130, 0

Rem Update detection point coordinates XY
SetUnitData 1, 132, 320
SetUnitData 1, 133, 240
```

Additional Information

In the variant type, model data cannot be obtained or changed, but model data can be copied.

Example: When copying the model of the search processing unit with the processing unit number 1 to the model of the search processing unit with the unit number 7

```
Rem No variable definition is required
GetUnitFigure 1, 0, FIGURE!

Rem Set to figure data of another processing unit
SetUnitFigure 7, 0, FIGURE!
```

Image data

Image data is data that is held by the processing unit itself.
When there are multiple image data items in the processing item, a unique number is assigned to each image data item. This makes it possible to set image data by specifying the processing unit number and image number.
Image data can be set from a macro customize program.
An image data setting macro function is used to set image data from a macro customize program.

Image Data Types

There are two types of image data.

Type	Image number	Description
Measurement image data	0 to 3	Image data that the processing unit registered in the measurement flow processes at the time of measurement. Normally the image is set in measurement image 0 at the time of measurement for "image input related" processing items and "image conversion related" processing items such as camera image input and filtering, and image processing is performed using images set in other processing items. Normally 0 should be used.
Processing unit image data	0 to max. 31	Image data that is held by each processing item separately from the measurement image data. The number of processing unit image data items and content varies by processing item. Use when you want to retain image processing results such as binarized images and filtered images.

● Acquisition of Data

Execution of only image data acquisition is not possible.

● Data Settings

The image data setting method is as follows.

Example: Setting Image Data No. 0 (filtered image) of the Unit No. 2 filtering processing unit in Filtered Image No. 0, and enabling use as a measurement image by following processing units

```
SetMeasureImage 0, 2, 0
```

5-1-2 Data Types Related to the System

Necessity of referring to this manual	Unit Macro	Communication Command Macro	Scene Control Macro	Unit Calculation Macro
	As needed			

With a macro customize program, data that does not depend on the measurement flow can be set and acquired by setting or acquiring data held by the system such as user data, global data, and system data.
The data access method varies depending on the data type.

Type	Description
Global Data	Data that can be set or acquired from a different scene using a macro function. Use this when you want to set or acquire a temporary numerical value or character string that does not need to be saved, such as an interim result of an operation. For details, refer to *Global Data* on page 5-8.
User Data	Data that can be set or acquired from a different scene using a user data processing item. Use this when you want to set or acquire numerical values that you want to be retained even if a power interruption occurs or restart is performed. For details, refer to *User Data* on page 5-8.
System data	Data that can be set or acquired from a different scene using a macro function. Use this when you want to set or acquire numerical values or character strings that you want to be retained even if a power interruption occurs or restart is performed. For details, refer to *System data* on page 5-9.

Global Data

Global data is data that can be set or acquired from a different scene using a macro function.
Use this when you want to set or acquire a temporary numerical value or character string that does not need to be saved, such as an interim result of an operation, as global data. (Refer to *5-1-3 Scope of Data and Save Area* on page 5-10.)
Use system data when you need to save data that you want to set or acquire from a different scene. Refer to *System data* on page 5-9.)
When you newly use the Global data, in advance, you should specify the Data ident with AddGlobalData, and then register it as Global data. Half-width alphabetical characters and the half-width marks "." and "_" can be used for the global data identification name. Refer to *AddGlobalData* on page 8-26.)

● Acquisition of Data

Global data is acquired by the method below.

Example: Acquiring the data identification name "GData" value

```
Rem Register the default value 10 for the global data whose data identification
 name is "GData"
AddGlobalData "GData", 10

Rem Acquire global data with the data identification name "GData"
GetGlobalData "GData", GDATA&
```

● Data Settings

The global data setting method is as follows.

Example: Setting the data identification name "GData" value

```
Rem Register the default value 10 for the global data whose data identification
 name is "GData"
AddGlobalData "GData", 10

Rem Set 15 in the global data with the data identification name "GData"
SetGlobalData "GData", 15
```

User Data

User data is data that can be set or acquired from a different scene using a user data processing item.
Use user data to set or acquire numerical values that you want to be retained even if a power interruption occurs or restart is performed. (Refer to *5-1-3 Scope of Data and Save Area* on page 5-10.)
When there is no need to save data that you want to set or acquire from a different scene, use global data. When you want to set or acquire not only numerical values but character strings as well, use system data. (Refer to *Global Data* on page 5-8, *System data* on page 5-9.)
Before using user data, you can set the default values of the user data with the user data tool. (Refer to *User Data Tool* in the *Vision System FH/FHV Series User's Manual (Cat. No. Z365)*.)

● Acquisition of Data

The user data processing item is used to acquire user data. (Refer to *User Data Tool* in the *Vision System FH/FHV Series User's Manual (Cat. No. Z365)*.)

● Data Settings

A user data processing item is used to set user data. (Refer to *User Data Tool* in the *Vision System FH/FHV Series User's Manual (Cat. No. Z365)*.)

System data

System data is data that can be set or acquired from a different scene using a macro function. Use system data to set or acquire numerical values or character strings that you want to be retained even if a power interruption occurs or restart is performed. (Refer to *5-1-3 Scope of Data and Save Area* on page 5-10.)
Use global data when there is no need to save data that you want to set or acquire from a different scene. When you only want to set or acquire numerical values, use user data. (Refer to *Global Data* on page 5-8, *User Data* on page 5-8.)
To use new system data, first specify a data identification name and register the system data that you want to use. Data identification names for system data are Data ID Name 0 and Data ID Name 1. The data identification names that correspond to ID Information 0 and ID Information 1 must be specified as arguments of the macro function.
For details on identification information and data identification names, refer to the system data list. (Refer to *Section 9 Macro Reference List* on page 9-1, *9-3 System Data List* on page 9-10.)

● Acquisition of Data

System data is acquired by the method below.

Example: Acquiring the value of "Initial Scene No." in the system data

```
Rem Acquire the system data whose Data ID Name 0 is "Configuration" and whose D
ata ID Name 1 is
"initialSceneNo"
GetSystemData "Configuration", "initialSceneNo", SDATA&
```

● Data Settings

The system data setting method is as follows.

Example: Setting the values of Data ID Name 0 "PanDA" and Data ID Name 1 "SData"

```
Rem Register 0 as the default value of the system data whose Data ID Name 0 is
"PanDA" and whose
Data ID Name 1 is "SData"
AddSystemData "PanDA", "SData", 0

Rem Set 5 in the system data whose Data ID Name 0 is "PanDA" and whose Data ID
Name 1 is "SData"
SetSystemData "PanDA", "SData", 5
```

Precautions for Correct Use

When adding new data to the system data, specify "PanDA" in Data ID Name 0.

5-1-3 Scope of Data and Save Area

Necessity of referring to this manual	Unit Macro	Communication Command Macro	Scene Control Macro	Unit Calculation Macro
	As needed			

In macro customize functions, many types of data can be used as needed for the objective, and variables and macro functions can be used to set and acquire data.
The areas where data settings and acquisition can be executed and the areas where data are saved vary depending on the type of data.
By using data types appropriately for the objective of the program, macro customize functions enable the creation of programs that are easy to change and maintain.

Data Scope

The locations from which data can be set and acquired are limited; data cannot be set or acquired from any location. This limited location is called a "scope", and data can only be set and acquired from within the scope defined for that data type.
Data scopes that can be used with the macro customize functions are as follows.

Scope	Description
Within a processing unit Within a communication command macro	Only enabled within one processing unit. If multiple processing units of the Unit Macro processing item or the Unit Calculation Macro processing item are included in the measurement flow, variables with the same name that exist in the processing units will be treated as separate variables. If multiple communication command macros are defined in the communication command macro tool, variables with the same name that exist in the communication command macros will be treated as same variables. When a communication command macro is executed, the value of the variable will be the value set when executing the communication command macro executed immediately before.
Within a scene	Enabled within one scene. If multiple processing units of the Unit Macro processing item or the Unit Calculation Macro processing item are included in the measurement flow, the same data can be set and acquired from each processing unit. The data cannot be set or acquired from the processing unit of a different scene.
Within the system	Enabled with the system. If multiple processing units of the Unit Macro processing item or the Unit Calculation Macro processing item are included in the measurement flow, the same data can be set and acquired from each processing unit. If multiple communication command macros are defined in the communication command macro tool, the same data can be set and acquired from each communication command macro. The data can be set and acquired from all scene groups and scenes.

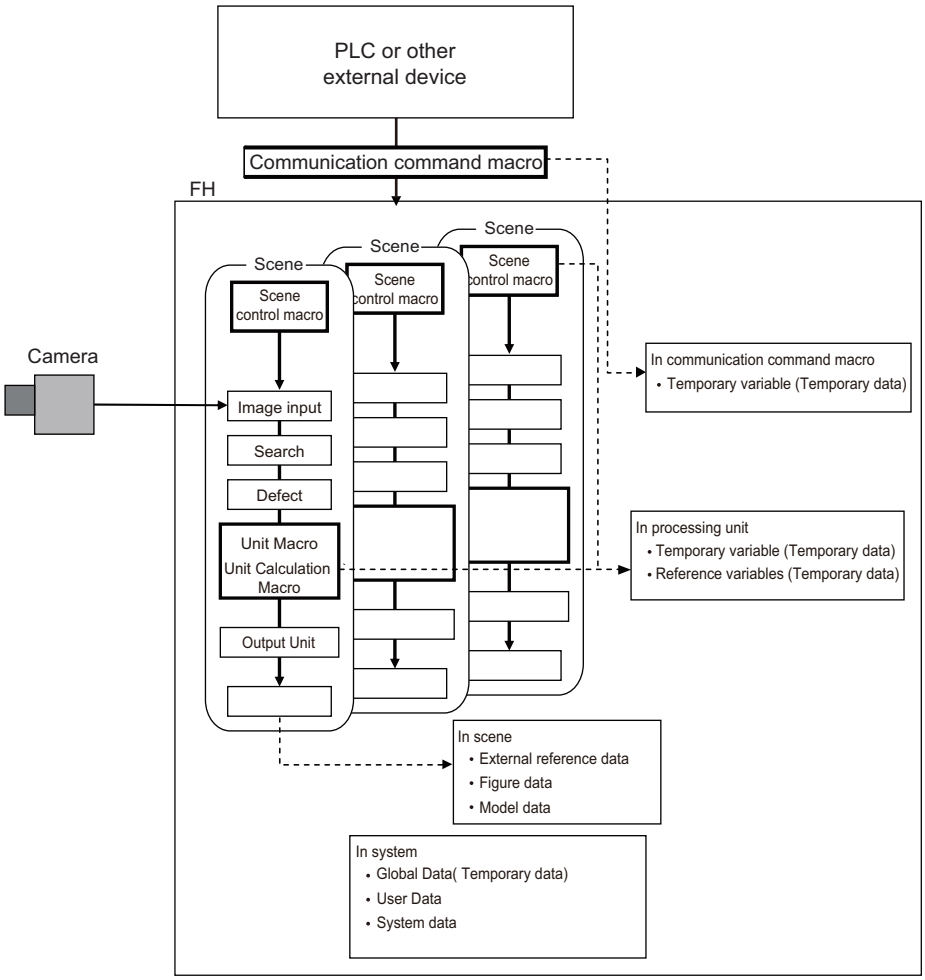
Data Save Area

Some data that can be used in the macro customize functions is only saved temporarily in memory, and some data is saved as scene or system data. The area where data is saved is called the "save area", and each data item has a specific save area.
Data save areas that can be used with the macro customize functions are as follows.

Save area	Can be saved?	Description
Temporary data	No	Data temporarily stored in memory. Cleared when the power is interrupted or the system is re-started. Use to handle data that does not need to be stored, such as temporary data used during an operation.
Scene data	Yes	Data stored as scene data. In addition to **Data save**, scene data can be saved to a file as a scene data file. For example, use this when handling data that must be saved for each scene, such as measurement parameters and other settings that vary by product type.
System data	Yes	Data stored as system data. In addition to **Data save**, system data can be saved to a file as a system data file. For example, use this to handle common scene data (not for only a single scene) and common system system data, such as reference settings and other inspection settings that do not depend on the product type.

Scope and Save Area by Variable and Data Item

The relation between the scope and save area of each variable and data item that can be used in a macro customize function is shown below.

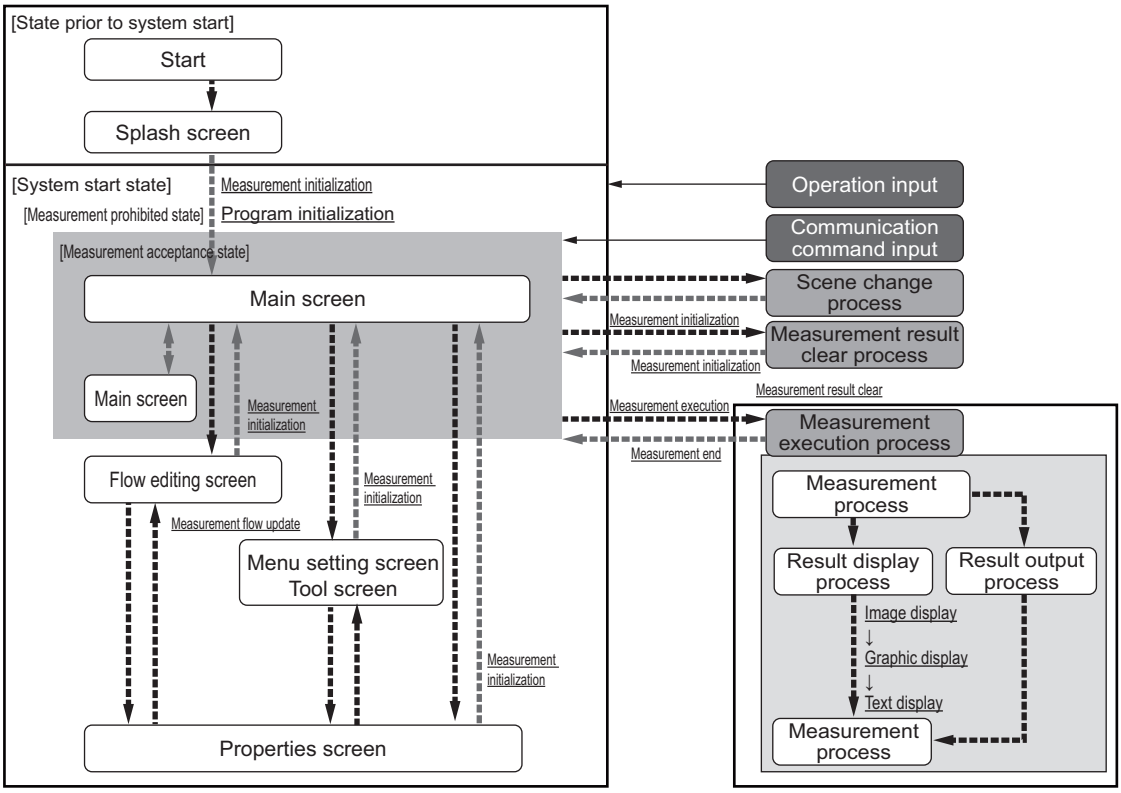


Variable and Data Types	Scope	Can be saved?	Data save area
Temporary variable (array variable) Refer to *Temporary Variable* on page 4-8.	Within a processing unit Within a communication command macro	No	Temporary data
Reference variables Refer to *Reference Variables* on page 4-9.			
External reference data Refer to *External reference data* on page 5-2.	Within a scene	Yes	Scene data
Figure data Refer to *Figure data* on page 5-3.			
Model data Refer to *Model data* on page 5-5.			
Image data Refer to *Image data* on page 5-6.		No	Temporary data
Global Data Refer to *Global Data* on page 5-8.	Within the system	No	Temporary data
Global Data Refer to *User Data* on page 5-8.		Yes	System data
System data Refer to *System data* on page 5-9.			

5-1-4 State Transitions and Execution Timing

Necessity of referring to this manual	Unit Macro	Communication Command Macro	Scene Control Macro	Unit Calculation Macro
	As needed			Not required

Macro customize programs consist of several subroutines. Subroutines are generally executed when there is a change of screen, setting, or data status. The subroutines that can be used vary by function of the macro customize functions. Decide which subroutine will be used based on the execution timing of the processes being programmed.



Execution timing and corresponding subroutines are shown below. For a description of each execution timing, refer to *Details of subroutine execution timing* on page 5-17.

Execution timing of subroutine	Pre-defined subroutine name	Unit Calculation Macro	Scene Control Macro	Communication Command Macro	Unit Macro	Description
Program initialization	*MCRINIT	-	●	-	●	Executed immediately after the program is loaded.
Measurement initialization	*MEASUREINIT	-	-	-	●	Executed before starting measurements.
Measurement execution	*MEASUREPROC	●	-	●	●	Executed when measurement processing is executed.
Display						
Image display	*MEASUREDISPI	-	-	-	●	Called when an image is displayed in the image window.

Execution timing of subroutine		Pre-defined subroutine name	Unit Calculation Macro	Scene Control Macro	Communication Command Macro	Unit Macro	Description
	Text display	*MEASUREDISPT	-	-	-	●	Called when text is displayed in the text window.
	Graphic display	*MEASUREDISPG	-	-	-	●	Called when a graphic display is displayed in the image window.
Measurement flow update		*RENUMPROC	-	-	-	●	Called when the processing unit reference number is updated.
Measurement result clear		*CLEARMEASUREDATA	-	-	-	●	Called when the processing unit measurement results are initialized.

System Status Transitions and Possibility of Execution

There are processes that cannot be executed in certain system statuses in a macro customize program.
The following types of system status exist.

Status type	Description
Measurement prohibited state	Measurement instructions and communication commands cannot be accepted in this state. The measurement prohibited state generally occurs during startup, as well as changes of measurement flow, processing unit and system settings, and when executing a process or operation. In the measurement prohibited state, the BUSY signal or similar status signal turns ON.
Measurement acceptance state	Measurement instructions and communication commands can be accepted in this state. In the measurement accepted state, the BUSY signal or similar status signal turns OFF.

The relation between sensor controller system statuses and execution timing is shown below.

Execution timing of subroutine		Pre-defined subroutine name	System status	
			Measurement prohibited state	Measurement acceptance state
Program initialization		*MCRINIT	Can be executed	Cannot be executed
Measurement initialization		*MEASUREINIT	Can be executed	Cannot be executed
Measurement execution		*MEASUREPROC	Cannot be executed	Can be executed
Display	Image display	*MEASUREDISPI	Can be executed	Can be executed
	Text display	*MEASUREDISPT		
	Graphic display	*MEASUREDISPG		
Measurement flow update		*RENUMPROC	Can be executed	Cannot be executed
Measurement result clear		*CLEARMEASUREDATA	Can be executed	Cannot be executed

In some cases it is possible to change the measurement prohibited state and measurement acceptance state while in a subroutine. For details, refer to *5-1-5 Exclusive Control in a Process* on page 5-20.

Execution During Screen Transitions

Some types of macro customize program execution timing occur in relation to screen transitions. The following types of screen transitions exist.

Original screen	New screen	Description
Splash screen	Main screen	Transition from splash screen at startup to main screen.
Main screen	Flow edit screen Menu settings screen Tool screen Properties screen	Transition from main screen to other screens.
Flow edit screen	Main screen	Transition from flow edit screen to main screen.
	Properties screen	Transition from flow edit screen to properties screen of processing unit.
Menu settings screen	Main screen	Transition from menu settings screen to main screen.
Tool screen	Main screen	Transition from tool screen to main screen.
	Properties screen	Transition from tool screen to properties screen of processing unit.
Properties screen	Main screen	Transition from properties screen of processing unit to main screen.
	Flow edit screen	Transition from properties screen of processing unit to flow edit screen.

The relation between sensor controller screen transitions and program execution timing is shown below.

Execution timing of subroutine		Pre-defined subroutine name	Screen transition	
			Original screen	New screen
Program initialization		*MCRINIT	Splash screen	Main screen
			Properties screen (Macro customize function)	Flow edit screen Main screen
			Tool screen (Macro customize function)	Main screen
Measurement initialization		*MEASUREINIT	Splash screen Flow edit screen Menu settings screen Tool screen Properties screen	Main screen
Measurement execution		*MEASUREPROC	-	-
Display	Image display	*MEASUREDISPI	-	-
	Text display	*MEASUREDISPT	-	-
	Graphic display	*MEASUREDISPG	-	-
Measurement flow update		*RENUMPROC	-	-
Measurement result clear		*CLEARMEASUREDATA	-	-

Process Transitions and Execution Timing

The timing of macro customize program execution is related both to screen transitions and to the type of process being executed.
The relation between process type and program execution timing is shown below.

Process	Timing of execution during process	Timing of execution after process	Description
Scene change	-	Measurement initialization	Scene change process. The program is executed when measurement initialization takes place after the process.
Measurement result clear	Measurement result clear	Measurement initialization	Measurement result clear process. The program is executed when the measurement result is cleared during the process, and when measurement initialization takes place after the process.
Measurement execution	Measurement execution	-	Measurement execution process. The program is executed when measurement is executed during the process.
	Image display	-	Image display process. The program is executed when the image is displayed during the process.
	Graphic display	-	Graphic display process. The program is executed when graphic display takes place during the process.
	Text display	-	Text display process. The program is executed when text display takes place during the process.
Flow edit	Measurement flow update	-	Measurement flow update process. The program is executed when the flow is updated during the process. Measurement flow update consists of the operation and process by which the unit number of the measurement flow is updated, including adding and deleting processing units to the measurement flow and changing the order.

Menu Operations and Execution Timing

In addition to screen transitions and process transitions, the timing of macro customize program execution is also related to the type of menu operation being executed.
The relation between process type and program execution timing is shown below.

Process	Timing of execution during process	Timing of execution after process	Description
Processing unit selection	Image display	-	Selection of a processing unit in the measurement flow when *Define displayed unit* is selected. The program is executed when the image is displayed.
	Graphic display	-	Selection of a processing unit in the measurement flow when *Define displayed unit* is selected. The program is executed when graphic display takes place.
	Text display	-	Selection of a processing unit in the measurement flow when *Define displayed unit* is selected. The program is executed when text display takes place.

Image Display Status Transitions and Execution Timing

The timing of execution of macro customize programs related to display is also related to image window display changes.
The relation between the image window and program execution timing is shown below.

Process	Timing of execution during process	Timing of execution after process	Description
Image mode change	Image display Graphic display	Measurement initialization	Image mode change process for the image window. The program is executed when image display and graphic display take place during the process, and when measurement initialization takes place after the process.
Image mode *Through camera image*	Image display Graphic display	Image display Graphic display	Process when the image mode of the image window is *Through camera image*. The program is executed repeatedly when image display or graphic display take place while the image mode is *Through camera image*.

Precautions for Correct Use

If the image mode of one or more of the image windows of the 24 image windows that can be used in the layout function is *Through camera image*, the Image mode *Through camera image* process is executed repeatedly.

Details of subroutine execution timing

Details of the execution timing of subroutines pre-defined in the system are described below.

Measurement Initialization (*MCRINIT)

The program initialization subroutine is executed immediately after the program is loaded. The process is executed at the timing below.

Details of subroutine execution timing	Main cases of execution
Immediately after macro editing	• When the program editing screen is opened • When the program editing screen is closed
When scene data is loaded	• At startup • When the scene group is changed • When any of the following file is loaded Scene data Scene group data System + Scene group 0 data Backup data

Measurement Initialization (*MEASUREINIT)

The measurement initialization subroutine is executed before starting measurements. Processing to access the processing unit and scene control macro does not operate correctly because it is executed during loading of scenes, etc. at startup. The process is executed at the timing below.

Details of subroutine execution timing	Main cases of execution
Screen transition to main screen	• When the flow edit screen is closed • When the system settings screen is closed • When any other screen that can be called from the main screen such as a scene change screen or saved screen is closed
Immediately after a layout change	• When the layout change is closed • When an image window setting or text window setting is changed
When a setting communication macro function or acquisition communication macro function is executed	When measurement is executed or a communication command other than continuous execution is executed
Immediately after the measurement start macro function is executed	When the *MeasureStart* macro function is executed

Precautions for Correct Use

When the image mode is *Freeze camera image*, the measurement image is not updated when measurement initialization takes place. However, if at least one of the 24 image windows that can be used in the layout function is set to *Through camera image*, the measurement image of the *Freeze camera image* image windows will be updated. Use the main window layout change function to change the image of all image windows to *Freeze camera image*, or use the *SetImageWindow* macro function to set the image mode of all image windows to *Freeze camera image* before the measurement initialization process is executed.
For details, refer to *Changing the Image Mode and Other Display Contents* in the *Vision System FH/FHV Series User's Manual (Cat. No. Z365)*.

Measurement Execution (*MEASUREPROC)

The measurement subroutine is executed when measurement is executed. The process is executed at the timing below.

Details of subroutine execution timing	Main cases of execution
When measurement is executed	• When measurement is executed or a communication command for continuous measurement execution is executed • When a measurement button such as the measurement button in the main screen or in the properties screen is clicked • When the *Measure* macro function is executed

● Image display (*MEASUREDISPI)

The image display subroutine is executed when an image is displayed. The process is executed at the timing below.

Details of subroutine execution timing	Main cases of execution
When the image display is updated	• When measurement is executed • When the image window settings are changed • When the image window is set to *Define displayed unit* and the processing unit selected in the measurement flow is changed

Precautions for Correct Use

When the position list display is ON, the image display subroutine and text display subroutine are not executed. If you want to execute the image display subroutine or text display subroutine, set the position list display to OFF.
For details, refer to *Changing the Image Mode and Other Display Contents* in the *Vision System FH/FHV Series User's Manual (Cat. No. Z365)*.

● Text display (*MEASUREDISPT)

The text display subroutine is executed when text display is executed. The process is executed at the timing below.

Details of subroutine execution timing	Main cases of execution
When the text display is updated	• When measurement is executed • When the text window settings are changed • When the text window is set to *Define displayed unit* and the processing unit selected in the measurement flow is changed

● Graphic display (*MEASUREDISPG)

The graphic display subroutine is executed when graphic display is executed. The process is executed at the timing below.

Details of subroutine execution timing	Main cases of execution
When the image display is updated	• When measurement is executed • When the image window settings are changed • When the image window is set to *Define displayed unit* and the processing unit selected in the measurement flow is changed

● Measurement Flow Update (*RENUMPROC)

The measurement flow update subroutine is executed when the measurement flow is changed. The process is executed at the timing below.

Details of subroutine execution timing	Main cases of execution
Immediately after flow editing	• When the order of the processing units in the measurement flow is changed, such as adding, deleting, or moving a processing unit. • When a macro function is executed that changes a processing unit number in the measurement flow or changes a processing unit, such as *AssignUnit*,*CopyUnit*, or *DeleteUnit*.

● Measurement result clear (*CLEARMEASUREDATA)

The measurement result clear subroutine is executed when measurement results are cleared. The process is executed at the timing below.

Details of subroutine execution timing	Main cases of execution
When measurement results are cleared	• When **OK** is clicked in the measurement result clear screen. • When the macro function that clears measurement results is executed

5-1-5 Exclusive Control in a Process

Necessity of referring to this manual	Unit Macro	Communication Command Macro	Scene Control Macro	Unit Calculation Macro
	Not required	As needed		Not required

This primarily uses the communication command macro and scene control macro.
Macro customize functions can be used to perform exclusive control in a process when a program is executed. Exclusive control must be performed during measurement and communication in order to prevent incorrect processing, such as a communication command or screen operation accidentally changing the scene during measurement, or subsequently received data overwriting the currently received data.

Exclusive Control in a Measurement Process

When executing the setting or acquisition of processing unit data, system data, or other data in a program, and when executing a process that changes scene data such as a scene change, you must first stop acceptance of measurement triggers during processing. If data setting, data acquisition, or a scene change is executed when measurement triggers can be accepted and a measurement trigger is then input during the setting process, acquisition process, or scene change, there is a risk that data inconsistencies will occur or an incorrect measurement result will be output.
To stop acceptance of measurement trigger input, use the MeasureStop function. To enable acceptance of measurement trigger input, use the MeasureStart function.

Example:

```
Rem Stop measurement
MeasureStop

Rem Change the scene or set data

Rem Resume measurement
MeasureStart
```

Precautions for Correct Use

- When executing a scene change or setting or acquiring processing unit data or other data in a program, always execute MeasureStop beforehand. If data setting or data acquisition is executed without executing MeasureStop, there is a risk that measurement may be executed during the setting process or acquisition process and cause inconsistencies in the data being set or acquired.
- If MeasureStart is not executed after MeasureStop, it will not be possible to accept measurement triggers. The BUSY signal will remain ON.
- If measurement is to be executed after MeasureStop in a program, execute MeasureStart before executing measurement. If MeasureStart is not executed after MeasureStop, measurement will not take place when executed.
- With the unit macro or unit calculation macro, create a program that does not require exclusive control. Switching between the measurement prohibited state and measurement acceptance state during measurement execution may cause unexpected operation.

Exclusive Control in a Communication Process

In communication with external devices, the FH series uses a polling process to monitor statuses and data transmission/reception.
When sending or receiving data, stop the communication polling process befor sending/receiving the data. If data transmission/reception is executed without stopping the polling process, there is a risk that inconsistencies may occur in the received data, such as data other than the intended data being received from the external device.
To change the state of the communication polling process, use the SetPollingState function.

Example: Receiving data in serial normal communication

```
Rem Stop serial normal communication
SetPollingState "SerialNormal", FALSE

Rem Receive the data

Rem Start serial normal communication
SetPollingState "SerialNormal", TRUE
```

6

Debug Function

6-1 How to Use the Debug Function 6-2

6-1-1 Debug Preparations 6-2
6-1-2 Debug Procedure 6-4
6-1-3 Checking Why an Error Occurred 6-5
6-1-4 Starting Debug 6-6
6-1-5 Identifying the Cause of an Error 6-7
6-1-6 Removing the Error 6-10
6-1-7 Exiting Debug 6-10

6-1 How to Use the Debug Function

Necessity of referring to this manual	Unit Macro	Communication Command Macro	Scene Control Macro	Unit Calculation Macro
	Required			

Invalid macro function calls and programming bugs in macro customize programs may cause errors in operation. Identifying the cause and correcting the program when an error occurs is called "debugging". Macro customize functions have a support function for program debugging, allowing errors to be efficiently removed.

Precautions for Correct Use

- Do not close the system status console window by a method such as clicking the **x** button in the upper right corner of the system status console window. The system may not operate correctly. If the system status console window is accidentally closed, save your settings and restart the sensor controller.
- When displaying text in the System status monitoring console window, if the System status monitoring console window is minimized, processing will take more time.

6-1-1 Debug Preparations

Debugging is performed in macro customize functions by changing the execution form of the program and using macro functions that are effective for debugging. Debugging can be performed by writing a macro function for debugging in the program and executing the program, and by directly entering a macro function from the system status console window and executing the program.

Program Execution Mode

Macro customize functions allow the execution mode of the program to be specified for each function. By varying the execution form of the program as appropriate for the execution conditions, program management and debugging is made easier.

Execution form	Description
Release mode	Execution form used for program execution and regular use. In release mode, the macro functions that are entered in the program for debugging are disabled.
Debugging mode	This is a convenient execution form used for program correction and debugging. Some debugging functions are only enabled in debug mode.

Precautions for Correct Use

Specify the program execution mode individually for each function and unit.
- Unit calculation macro, unit macro: specify by processing unit
- Scene control macro: specify by scene
- Communication command macro: specify by communication macro

Macro Functions Used for Debugging

Some of the macro functions are effective for debugging. Macro functions that are effective for debugging are described below.

Macro function	Description
Debug (Refer to *Debug* on page 8-68)	Select the program usage mode and the information output method used when the program is executed. • 0: Set the usage mode to release mode. When an error occurs, an error description is output. • 1: Set the usage mode to release mode. When an error occurs, an error description is output to the system status console window. This setting is recommended for times other than when debugging is performed. • 2: Set the usage mode to release mode. When the program is executed, the contents of each line are output to the system status console window. • 3: Set the usage mode to release mode. When an error occurs, an error description is output to the message box. • 16: Set the usage mode to debug mode. When an error occurs, an error description is output. • 17: Set the usage mode to debug mode. When an error occurs, an error description is output to the system status console window. • 18: Set the usage mode to debug mode. When the program is executed, the contents of each line are output to the system status console window. This setting is recommended when normal debugging is performed. • 19: Set the usage mode to debug mode. When an error occurs, an error description is output to the message box.
Stop (Refer to *Stop* on page 8-312)	This macro function is only enabled in debug mode. The function is used to stop execution of the program at a break point. You can also set conditions with the SetStop function. When the specified conditions are met, execution of the program stops.
SetStop (Refer to *SetStop* on page 8-293)	This macro function is only enabled in debug mode. Use this function to set conditions for stopping the program with the Stop function.
DebugPrint (Refer to *DebugPrint* on page 8-69)	This macro function is only enabled in debug mode. The function displays text in the system status console window.

When performing debugging, you can enter macro functions for debugging in the system console window and execute the program. This allows you to conveniently execute and stop the program in the system status console window without the need to edit the program.
Macro functions for debugging that are convenient to use in the system status console window are described below.

Macro function	Description
Cont (Refer to *Cont* on page 8-53)	Resumes execution of the program after it has been stopped using the Stop function. Parameters can be specified to execute the program in steps. • No parameters: Resumes execution of the program. The next program is executed until it ends or an error occurs. • 0: Executes the program by step-in execution. If the current program line calls a subroutine, the subroutine is entered and is executed in steps. Otherwise, the current statement is executed and the program is stopped at the next line. • 1: Executes the program by step-over execution. If the current program line calls a subroutine, the entire subroutine is executed and the program stops at the next line after the subroutine call. Otherwise, the current statement is executed and the program is stopped at the next line. • 2: Executes the program by step-out execution. If the current program line is a subroutine that was called from a subroutine, the entire subroutine after the current program line is executed, and the program stops at the next line of the subroutine that called the subroutine. Otherwise, the program is executed until it ends or an error occurs.
Varlist (Refer to *VarList* on page 8-353)	Displays information on the variables with the specified variable names in a list in the system status console window.

6-1-2 Debug Procedure

When an error occurs in the program, follow the steps below to correct the program.

Item name	Step	Description
Error occurs	Error occurs	An error occurs during program execution and the program execution process is forcibly stopped.
	↓	
Checking Why an Error Occurred (Refer to *6-1-3 Checking Why an Error Occurred* on page 6-5.)	Check the Console Window	A brief description of the error appears in the system status Console Window. Check the error description.
	↓	
	Identify the nature of the error	Refer to the error list to identify the nature of the error.
	↓	
Starting Debug (Refer to *6-1-4 Starting Debug* on page 6-6.)	Change to debug mode	Specify 18 for the debug function parameter at the start of the *MCRINIT subroutine or process of the program.
	↓	
Identifying the Cause of an Error (Refer to *6-1-5 Identifying the Cause of an Error* on page 6-7.)	Identifying the location of the error	Based on the identified nature of the error, determine which line and where on the line the error occurred.
	↓	
	Identifying the Cause of the Error	Identify why the error occurred.
	↓	

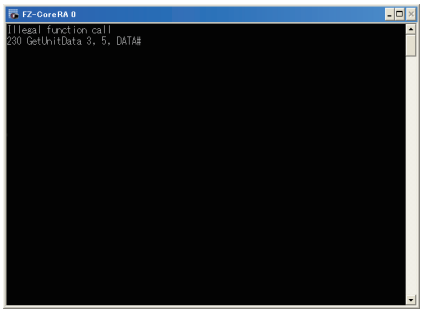
Item name	Step	Description
Removing the Error (Refer to *6-1-6 Removing the Error* on page 6-10.)	Correct and check operation	Correct the program to remove the cause of the error, and check operation to verify that the error has been eliminated.
	↓	
Exiting Debug	Change to release mode	Specify 1 for the debug function parameter at the start of the *MCRINIT subroutine or process of the program.
	↓	
Save	Save settings	Save your changes.

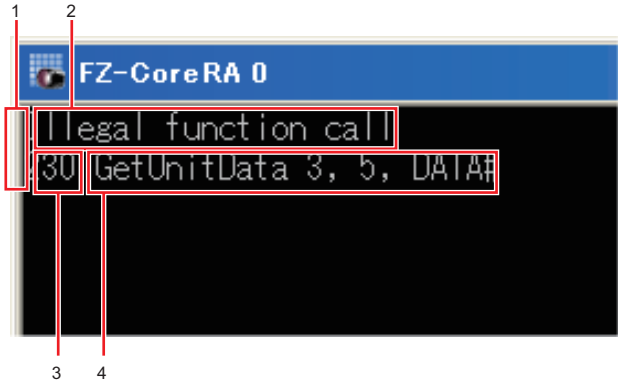
6-1-3 Checking Why an Error Occurred

When an error occurs during program execution, the subroutine process in which the error occurred is forcibly stopped. Processes other than that in which the sensor controller error occurred continue to run.
If an error occurs in the Unit Macro processing item or Unit Calculation Macro processing item, the measurement result of the processing unit in which the error occurred is NG. If an error occurs in a communication command macro, the macro function returns an error. Regardless of where the error occurred, error information appears in the system status console window.

Checking the System Status Console Window

When an error occurs in the program, error information appears in the system status console window. You can check this information to determine the type of error and the location of the error in the program.





1. Error module
   Module that outputs the error. This indicates the function of the macro customize function that output the error.

Content displayed for error modu	Description
Macro(U**)>	The unit macro or unit calculation macro outputted error information. "**" after "U" shows the processing unit number.
Macro(SC)>	The scene control macro outputted error information.
Macro(IO)>	The communication command macro outputted error information.

2. Error Message

This message indicates error factors. Based on the error message, you can check what type of error occurred.

3. Error line number
   The number of the line where the error occurred. Use this to determine the location of the error in the program.
4. Error statement
   Written program content on the line where the error occurred. The error statement allows you to check the written content of the program without opening the program editing screen.

Identifying the Error

Based on the error message that appears in the system status console window, check the error in the *9-1 Error List* on page 9-2.
If this gives you sufficient information to identify the cause of the error, remove the error.

Precautions for Correct Use

If you specified 0 or 16 for the debug function parameter, the error information will not appear in the system status console window. Specify a value other than 0 or 16 for the debug function parameter.
For details, refer to *Macro Functions Used for Debugging* on page 6-3.

6-1-4 Starting Debug

After checking why the error occurred, start Debug. The *MEASUREPROC subroutine of the Unit Macro processing item is used for debugging below as an example.
Open the program editing screen and write the debug function at the beginning of the *MCRINIT subroutine or the program.

Example: To start Unit Macro Debug, specify 18 in the parameter for the debug function in the *MCRINIT subroutine

```
*MCRINIT

    Rem Specify 18 for the debug function parameter and execute
    Debug 18

Return
```

Example: To start Communication Command Macro Debug, specify 18 in the parameter for the debug function at the beginning of the communication command program to be debugged

```
Rem Specify 18 for the debug function parameter and execute
Debug 18

Rem The actual communication command process is written from here
ChangeScene 1
SetUnitData 2, 101, 0
```

Additional Information

- In addition to 1 and 18, other values are available for the parameter for the debug function. Use these when appropriate.
  For details, refer to *Macro Functions Used for Debugging* on page 6-3.
- If you delete the debug function from the program, the usage mode will remain the usage mode that was set the last time the debug function was executed. Restart to return the usage mode to its default setting.

6-1-5 Identifying the Cause of an Error

After changing the usage mode to debug mode, identify the cause of the error.

Identifying the Location of the Error

To identify the cause of the error, determine where the error occurred.

1 Enter the Stop function in the program editing screen.

Example: Enter the Stop function immediately before the measurement process in the *MEASUREPROC subroutine of the Unit Macro

```
*MEASUERPROC

    Rem Execute the Stop function and stop the program at this line
    Stop

    Rem The actual measurement process is written from here
    POS.X#=(POS0.X@ + POS1.X@) / 2
    POS.Y#=(POS0.Y@ + POS1.Y@) / 2
    Print POS.Y# / POS.X#

Return
```

2 Close the program editing screen with **OK** and return to the main screen.
The main screen appears.

3 Execute measurement.
When the Stop function line is executed, the measurement process stops. If the process does not stop, check if debug mode is specified with the debug function parameter.

4 Check the system status console window.
The system status console window shows the following:

```
Macro(U3)210 *MEASUREPROC
Macro(U3)220 Stop
Macro(U3)>
```

5 Execute the program in steps of one line.
Enter "Cont 1" in the system status console window from your keyboard.

```
Macro(U3)210 *MEASUREPROC
Macro(U3)220 Stop
Macro(U3)>cont 1
```

6 Hit **Enter** on your keyboard.
One line of the program is executed. The program stops at the next line.

```
Macro(U3)210 *MEASUREPROC
Macro(U3)220 Stop
Macro(U3)>cont 1
Macro(U3)230 POS.X#=(POS0.X@ + POS1.X@) / 2
Macro(U3)>
```

7 Continue to step through the program unit until the line that contains the error is executed. When the line with the error is executed, error information appears in the system status console window. In this case, it can be seen that the error is caused by dividing by 0.

```
Macro(U3)210 *MEASUREPROC
Macro(U3)220 Stop
Macro(U3)>cont 1
Macro(U3)230 POS.X#=(POS0.X@ + POS1.X@) / 2
Macro(U3)>cont 1
Macro(U3)240 POS.Y#=(POS0.Y@ + POS1.Y@) / 2
Macro(U3)>cont 1
Macro(U3)250 Print POS.Y# / POS.X#
Macro(U3)Division by zero in 250
Macro(U3)>
```

Identifying the Cause of the Error

Once you have determined the location of the error, identify the cause.

1 Check the value of the variable.
Enter "VarList" in the system status console window from your keyboard.

2 Hit **Enter** on your keyboard.
The variables are listed in the system status console window.

```
Macro(U3)210 *MEASUREPROC
Macro(U3)220 Stop
Macro(U3)>cont 1
Macro(U3)230 POS.X#=(POS0.X@ + POS1.X@) / 2
Macro(U3)>cont 1
Macro(U3)240 POS.Y#=(POS0.Y@ + POS1.Y@) / 2
Macro(U3)>cont 1
Macro(U3)250 Print POS.Y# / POS.X#
Macro(U3)Division by zero in 250
Macro(U3)>varlist
POS.X#=0.000
POS0.X@=0.000
POS1.X@=0.000
POS.Y#=211.000
POS0.Y@=209.000
POS1.Y@=213.000
```

From the program contents and the variable list, it can be seen that the value of the POS.X# denominator variable is 0.

Stopping the Program When a Specific Condition is Met

The program can also be stopped when a specific condition is met. Setting an appropriate condition allows the location of an error to be identified more efficiently than with the Stop function.

```
*MCRINIT

   Debug 18

   Rem Use the SetStop function to set the program stop condition so that the prog
ram stops when the Stop function
   SetStop Str$(0)

Return

*MEASUERPROC

   Rem The actual measurement process is written from here
   POS.X#=(POS0.X@ + POS1.X@) / 2
   POS.Y#=(POS0.Y@ + POS1.Y@) / 2

   Rem Execute the Stop function and stop the program on the line where the POS.X#
 variable is 0
   Stop Str$(POS.X#)
   Rem Execute the Stop function and stop the program on the line where the POS.Y#
 variable is 0
   Stop Str$(POS.Y#)

   Print POS.Y# / POS.X#

Return
```

Additional Information

A true/false relational expression (true: -1, false: 0) can also be set as the condition for the Stop function.
Example: Stopping the program when the CORRELATION& variable is less than 60
Set the SetStop function so that the program stops when the relational expression is true (-1).
SetStop Str$(-1)
The Stop function stops the program when the relational expression CORRELATION& < 60 is true.
Stop Str$(CORRELATION& < 60)

6-1-6 Removing the Error

Once you have identified the cause of the error, correct the program to remove the error.
After correcting the program, check operation and verify that the error has been removed.

Example: Use "If" to prevent the denominator from becoming 0 in the *MEASUREPROC subroutine of the Unit Macro.

```
*MEASUERPROC

   Rem Execute the Stop function and stop the program at this line
   Stop

   Rem The actual measurement process is written from here
   POS.X#=(POS0.X@ + POS1.X@) / 2
   POS.Y#=(POS0.Y@ + POS1.Y@) / 2

   Rem Add an "If" statement so that division is only executed when POS.X# is not
0
   If POS.X# <> 0 Then
      Print POS.Y# / POS.X#
   EndIf

Return
```

6-1-7 Exiting Debug

When the error has been removed, exit Debug.
Open the program editing screen and change the parameter of the debug function at the beginning of the *MCRINIT subroutine or the program.

Example: To exit Unit Macro Debug, change the parameter for the debug function in the *MCRINIT subroutine to 1.

```
*MCRINIT

   Rem Specify 1 for the debug function parameter and execute
   Debug 1

Return
```

Additional Information

- In addition to 1 and 18, other values are available for the parameter for the debug function. Use these when appropriate.
  For details, refer to *Macro Functions Used for Debugging* on page 6-3.
- If you delete the debug function from the program, the usage mode will remain the usage mode that was set the last time the debug function was executed. Restart to return the usage mode to its default setting.

7

Troubleshooting

7-1 Troubleshooting 7-2
7-1-1 Troubleshooting for Programming 7-2
7-1-2 Troubleshooting When Checking Operation 7-4
7-1-3 Troubleshooting during debugging 7-9
7-1-4 Troubleshooting during regular operation 7-10

7-1 Troubleshooting

Necessity of referring to this manual	Unit Macro	Communication Command Macro	Scene Control Macro	Unit Calculation Macro
	As needed			

When a unit macro customize function does not operate correctly, refer to the following to correct settings or operation.

7-1-1 Troubleshooting for Programming

Problems that are commonly encountered when creating programs with the macro customize functions are described below, along with the actions to take.

Problem	Cause	Action
No response when "DEL", "BS", or "Enter" is clicked.	The focus may not be on the program window.	Click the place you want to work in and then click the button.
A symbol cannot be entered when using a USB keyboard.	The USB key layout may be different from the keys that appear on the keyboard.	The keyboard of the FH series uses the same layout as an English keyboard. Either read the keys on your keyboard as English layout keys, or click **Keyboard** in the program editing screen and use the screen keyboard that appears.
It takes time for the screen keyboard to appear	The sensor controller and external devices may be connected by Ethernet.	If the sensor controller is connected to external devices by Ethernet, it occasionally takes time for the screen keyboard to appear. In a network environment that does not use a router or DNS server, set the same value for the default gateway and DNS server address as the IP address.
An unintended value is set.	When the same name is applied to multiple variables, unintended values can be set to those variables. For example, when the subroutines of *MEASUREPROC and *MEASUREDISPT have variables of the same name and they are used for different purposes, unintended values can be set to those variables.	Do not assign the same name to multiple variables.
A value is not set in a variable.	The variable in which you are attempting to set a value may have a type mistake.	If the variable name including the identifier is not correct, the variable will be treated as a different variable in the program. Make sure the variable name is correct.

Problem	Cause	Action
A comment or character string output with the Print function becomes corrupted.	Characters other than Japanese or English characters may be used in the Print function or comment.	For Print function, only English is available. Languagecharacters are not available such as "ä" in German. In comments, the following languages can be displayed. • FH series: English and selected language of **Language setting** are displayable.
When attempting to close the program editing screen, an error occurs and the screen does not close.	Full-width space characters or tab characters may be used in the program.	An error will occur if you attempt to close the program screen editing screen when full-width spaces or tabs are included in parts of the program other than character string type constants and variables. Select *Disp unvisible characters* in the Disp option of the program editing screen, and then locate the full-width spaces and tabs in the program and remove.
	An incorrect macro function name may be used in the program.	If an incorrect macro function name is used in the program, an error will occur when the program editing screen is closed. Select *Disp highlight* in the Disp option of the program editing screen, and then locate macro function names that are not color-highlighted and correct the macro function names.
When loading a program created with the text editor on a computer, only part of the program loads.	The program may exceed the maximum number of characters that can be entered.	A program that exceeds the maximum number of characters that can be entered will not load correctly. Check the remaining number of characters that can be entered in the program editing screen, and correct the program so that is does not exceed the character limit.

7-1-2 Troubleshooting When Checking Operation

Problems that are commonly encountered when checking the operation of programs created with the macro customize functions are described below, along with the actions to take.

Troubleshooting when checking the operation of the Unit Calculation Macro

Problem	Cause	Action
An error message appears in the system status console window	An error occurred when the program was executed.	Check the error message and correct the error. For details, refer to *9-1 Error List* on page 9-2. If it is difficult to identify the error, use Debug to determine the cause and correct the error. For details, refer to *6-1 How to Use the Debug Function* on page 6-2.
The calculation result of the Unit Calculation Macro processing unit is unmeasured.	The operator setting may be disabled.	Set the check box of the operator on to enable.
	The calculation judgement conditions may not be correctly set.	Set judgement conditions that are appropriate for the calculation and expected result.
Processing time is taking longer when the system status monitoring console window is minimized.	When the System status monitoring console window is minimized, it takes more time to display text on the monitoring console window because the process of displaying the window on the screen is executed.	Perform measurement with the System status monitoring console window displayed on the screen.

Troubleshooting when checking the operation of the Scene Control Macro

Problem	Cause	Action
An error message appears in the system status console window	An error occurred when the program was executed.	Check the error message and correct the error. For details, refer to *9-1 Error List* on page 9-2. If it is difficult to identify the error, use Debug to determine the cause and correct the error. For details, refer to *6-1 How to Use the Debug Function* on page 6-2.
Data reception during processing sometimes fails	The communication process may not have been stopped with the SetPollingState function.	Before acquiring data with the ReceiveData or other function, use the SetPollingState function to stop the communication process. For details, refer to *SetPollingState* on page 8-283.
Processing unit figure data cannot be set with the SetUnitFigure function, and registered figures are also cleared.	There may not be a measurement image when the figure is set.	When there is no measurement image because the processing unit is in the unmeasured state or otherwise, the figure setting will fail and previous settings will be cleared. Execute measurement before setting a figure, or use the ImageUpdate function to ready an image. For details, refer to *ImageUpdate* on page 8-164.

Problem	Cause	Action
Measurement or remeasurement does not take place in the program	Measure or Remeasure is executed in the program in the measurement prohibited state.	When executing measurement with Measure or Remeasure in the program, use the MeasureStop function and MeasureStart function to appropriately control the measurement trigger input prohibited/allowed state. For details, refer to *Exclusive Control in a Measurement Process* on page 5-20.
Processing time is taking longer when the system status monitoring console window is minimized.	When the System status monitoring console window is minimized, it takes more time to display text on the monitoring console window because the process of displaying the window on the screen is executed.	Perform measurement with the System status monitoring console window displayed on the screen.

Troubleshooting when checking the operation of the communication command macro

Problem	Cause	Action
An error message appears in the system status console window	An error occurred when the program was executed.	Check the error message and correct the error. For details, refer to *9-1 Error List* on page 9-2. If it is difficult to identify the error, use Debug to determine the cause and correct the error. For details, refer to *6-1 How to Use the Debug Function* on page 6-2.
Data reception during processing sometimes fails	The communication process may not have been stopped with the SetPollingState function.	Before acquiring data with the ReceiveData or other function, use the SetPollingState function to stop the communication process. For details, refer to *SetPollingState* on page 8-283.
Processing unit figure data cannot be set with the SetUnitFigure function, and registered figures are also cleared.	There may not be a measurement image when the figure is set.	When there is no measurement image because the processing unit is in the unmeasured state or otherwise, the figure setting will fail and previous settings will be cleared. Execute measurement before setting a figure, or use the ImageUpdate function to ready an image. For details, refer to *ImageUpdate* on page 8-164.

Problem	Cause	Action
The communication command macro does not execute	The command may not be enabled.	In the setting screen of the communication command macro tool, select the checkbox of the command No. that you want to execute.
	You are using simulation software.	The communication function does not operate on a computer. To execute commands created with the communication command macro, execute on the sensor controller.
	Measure or Remeasure is executed in the program when there is a checkmark in BUSY ON.	Before using Measure or Remeasure in command processing to execute measurement, remove the checkmark from BUSY ON of that command in the setting screen of the communication command macro tool. In addition, use the MeasureStop function and MeasureStart function to appropriately control the measurement trigger input prohibited/allowed state. For details, refer to *Exclusive Control in a Measurement Process* on page 5-20.
The normal communication command macro is not executed.	The character string sent as the command may not match the character string set in the command name.	Send the same character string for the command as the command name set in the setting screen of the communication command macro tool.
A communication command macro other than the normal communication command macro is not executed.	The numerical value sent as the command parameter when the command is executed does not match the command number.	As the command, send the same numerical value in hexadecimal (binary in the case of parallel) as the command number shown in the setting screen of the communication command macro tool.
A command parameter cannot be specified in the communication command macro	You may be attempting to specify a command parameter in the parallel communication command macro.	A command parameter cannot be specified in the communication command macro in parallel communication. Consider one of the following methods: • Set the necessary data in the processing unit with the processing unit data setting, and then execute the command. • Use a communication method other than parallel.
BUSY ON does not take place when the communication command macro is executed	The BUSY ON checkbox may not be selected.	Select the BUSY ON checkbox of the command you want to execute in the setting screen of the communication command macro tool.
The communication command macro settings cannot be saved.	You may be attempting to save the settings with **Save to file**.	The communication command macro settings are not saved in the backup data (.bkd). Consider one of the following methods: • Execute **Data save**. • Execute **Export** in the setting screen of the communication command macro tool.
A standard communication command does not execute	The name of a command created with the communication command macro may be the same as the name of the standard communication command.	If the name of a command created with the communication command macro function is the same as the name of a standard communication command, the command created with the communication command macro is given priority. Change the command name to a name that is different from the standard communication command.

Problem	Cause	Action
Processing time is taking longer when the system status monitoring console window is minimized.	When the System status monitoring console window is minimized, it takes more time to display text on the monitoring console window because the process of displaying the window on the screen is executed.	Perform measurement with the System status monitoring console window displayed on the screen.

Troubleshooting when checking the operation of the unit macro

Problem	Cause	Action
An error message appears in the system status console window	An error occurred when the program was executed.	Check the error message and correct the error. For details, refer to *9-1 Error List* on page 9-2. If it is difficult to identify the error, use Debug to determine the cause and correct the error. For details, refer to *6-1 How to Use the Debug Function* on page 6-2.
Data reception during processing sometimes fails	The communication process may not have been stopped with the SetPollingState function.	Before acquiring data with the ReceiveData or other function, use the SetPollingState function to stop the communication process. For details, refer to *SetPollingState* on page 8-283.
Processing unit figure data cannot be set with the SetUnitFigure function, and registered figures are also cleared.	There may not be a measurement image when the figure is set.	When there is no measurement image because the processing unit is in the unmeasured state or otherwise, the figure setting will fail and previous settings will be cleared. Execute measurement before setting a figure, or use the ImageUpdate function to ready an image. For details, refer to *ImageUpdate* on page 8-164.
A figure drawn with a macro function of screen display window control such as DrawLine is drawn at a position different from the actual coordinates	The processing unit that corrects position such as Position Correction may be registered in the measurement flow.	If the processing unit that corrects position is registered in the measurement flow, the coordinate values that can be acquired with the UnitData function are the values before correction. In the parameter of the DrawLine function, specify the processing unit number of the unit macro processing unit for which the program is written.
A measurement result image such as the labeling binary image using the display process subroutine cannot be displayed.	You may be attempting to display the measurement result image of a processing unit that does not have image data.	The main measurement processing items such as labeling do not have image data. The measurement result image of a processing item that does not have image data cannot be acquired in the program. To acquire and display a measurement result image, use a processing item that has image data such as Advanced Filter. For image numbers and image content that can be referenced in each processing item, refer to *9-9 Image Number List* on page 9-78.

Problem	Cause	Action
An error such as "Illegal function call" or "zero divide" occurs when the sensor controller is started	It is possible that a value is used that has not been initialized in the display process subroutine.	If executed with the display process subroutine in the unmeasured state, a data acquire process such as GetGlobalData may fail, or a variable may be used without a set value. Change the program for the display process subroutine so that the drawing process is only executed after measurement.
The processing result of the Unit Macro processing unit text display process and screen display process do not appear	The position list display may be ON.	When the position list display is ON, the text display subroutine and image display subroutine are not executed. Set the position list display to OFF. For details, refer to *Changing the Image Mode and Other Display Contents* in the *Vision System FH/FHV Series User's Manual (Cat. No. Z365)*.
An "illegal function call" error sometimes occurs when search process unit measurement is executed with the MeasureProc function	It is possible that no objects were found in the search process.	An "illegal function call" error occurs if there are 0 detections in the measurement process of the search process unit executed with the MeasureProc function. Use the Try - Catch - End Try function to create a process that handles the occurrence of 0 detections in the program. For details, refer to *Try Catch End Try* on page 8-337.
Processing time is taking longer when the system status monitoring console window is minimized.	When the System status monitoring console window is minimized, it takes more time to display text on the monitoring console window because the process of displaying the window on the screen is executed.	Perform measurement with the System status monitoring console window displayed on the screen.

7-1-3 Troubleshooting during debugging

Problems that are commonly encountered when debugging programs with the macro customize functions are described below, along with the actions to take.

Problem	Cause	Action
An error message appears in the system status console window	An error occurred when the program was executed.	Check the error message and correct the error. For details, refer to *9-1 Error List* on page 9-2. If it is difficult to identify the error, use Debug to determine the cause and correct the error. For details, refer to *6-1 How to Use the Debug Function* on page 6-2.
A comment or character string output with the Print function becomes corrupted.	Characters other than Japanese or English characters may be used in the Print function or comment.	For Print function, only English is available. Languagecharacters are not available such as "ä" in German. In comments, the following languages can be displayed. • FH series: English and selected language of **Language setting** are displayable.
The Stop function does not stop the program	It is possible that release mode is specified in the Debug function.	When release mode is specified in the Debug function, functions such as the Stop function and DebugPrint function are not enabled. Execute the Debug function in the *Mcrinit subroutine to change from release mode to debug mode. For details, refer to *6-1 How to Use the Debug Function* on page 6-2.
A debug character string cannot be output to the system status console window with the DebugPrint function	It is possible that release mode is specified in the Debug function.	When release mode is specified in the Debug function, functions such as the Stop function and DebugPrint function are not enabled. Execute the Debug function in the *Mcrinit subroutine to change from release mode to debug mode. For details, refer to *6-1 How to Use the Debug Function* on page 6-2.
A specific program line does not execute	It is possible that a statement and a comment are written on the same line.	A statement is sometimes not executed correctly when written on the same line as a comment. Write the comment on a separate line.
	It is possible that release mode is specified in the Debug function.	When release mode is specified in the Debug function, functions such as the Stop function and DebugPrint function are not enabled. Execute the Debug function in the *Mcrinit subroutine to change from release mode to debug mode. For details, refer to *6-1 How to Use the Debug Function* on page 6-2.
Processing time is taking longer when the system status monitoring console window is minimized.	When the System status monitoring console window is minimized, it takes more time to display text on the monitoring console window because the process of displaying the window on the screen is executed.	Perform measurement with the System status monitoring console window displayed on the screen.

7-1-4 Troubleshooting during regular operation

Problems that are commonly encountered when using the macro customize functions in regular operation are described below, along with the actions to take.

Problem	Cause	Action
An error message appears in the system status console window	An error occurred when the program was executed.	Check the error message and correct the error. For details, refer to *9-1 Error List* on page 9-2. If it is difficult to identify the error, use Debug to determine the cause and correct the error. For details, refer to *6-1 How to Use the Debug Function* on page 6-2.
A standard communication command does not execute	The name of a command created with the communication command macro may be the same as the name of the standard communication command.	If the name of a command created with the communication command macro function is the same as the name of a standard communication command, the command created with the communication command macro is given priority. Change the command name to a name that is different from the standard communication command.
Communication command macro settings cannot be applied to another sensor controller	It is possible that only backup data (.bkd) is loaded in the other sensor controller.	The communication command macro settings are not saved in the backup data (.bkd). Load the settings file (.mcr) that was saved by executing **Save** in the setting screen of the communication command macro tool into the other sensor controller separately from the backup data (.bkd). To load the settings file (.mcr) in the other sensor controller, execute **Load** in the setting screen of the communication command macro tool of that sensor controller.
The communication command macro settings cannot be saved.	You may be attempting to save the settings with **Save to file**.	The communication command macro settings are not saved in the backup data (.bkd). Consider one of the following methods: • Execute **Data save**. • Execute **Export** in the setting screen of the communication command macro tool.
Measurement flow settings that were saved are cleared after restart	It is possible that the *SaveProc subroutine is written in the scene control macro program.	When the *SaveProc subroutine is written, only the process written in the *SaveProc subroutine is executed when **Data save** is executed or scene data is saved. For this reason, overall scene data changes are not saved in the way they are when regular **Data save** or **Save to file** is executed. Delete the *SaveProc subroutine.
An "illegal function call" error sometimes occurs when search process unit measurement is executed with the MeasureProc function	It is possible that no objects were found in the search process.	An "illegal function call" error occurs if there are 0 detections in the measurement process of the search process unit executed with the MeasureProc function. Use the Try - Catch - End Try function to create a process that handles the occurrence of 0 detections in the program. For details, refer to *Try Catch End Try* on page 8-337.

Problem	Cause	Action
Processing time is taking longer when the system status monitoring console window is minimized.	When the System status monitoring console window is minimized, it takes more time to display text on the monitoring console window because the process of displaying the window on the screen is executed.	Perform measurement with the System status monitoring console window displayed on the screen.

8

Macro Functions

8-1 Macro Function List 8-2
8-1-1 Alphabetical Order 8-2
8-1-2 Function-based Index 8-15
8-2 Macro Command Reference 8-24

8-1 Macro Function List

Macro functions that can be used in macro customize functions are shown below.

8-1-1 Alphabetical Order

Command Name	Function	Classification	Unit Calculation Macro	Scene Control Macro	Communication Command Macro	Unit Macro	References
Abs	Gets the absolute value of the specified expression.	Arithmetic calculation	OK	OK	OK	OK	page 8-24
AbsolutePath $	Gets the absolute path of the folder assigned to the Quick access name.	Others	-	OK	OK	OK	page 8-25
AddGlobalData	Adds the global data.	Control global data	OK	OK	OK	OK	page 8-26
AddSystemData	Adds the system data.	System data	OK	OK	OK	OK	page 8-27
AND	Gets the logical product of twoexpressions.	Arithmetic calculation	OK	OK	OK	OK	page 8-28
ApplicationPath$	Gets the pass name regarding application software.	Others	OK	OK	OK	OK	page 8-29
ApplicationVersion$	Gets the version information of application software.	Others	OK	OK	OK	OK	page 8-30
ApproximationCircle	Gets the approximate circle.	Arithmetic calculation	OK	OK	OK	OK	page 8-31
ArrayDims	Gets the number of dimensions in the array.	Others	-	OK	OK	OK	page 8-32
ArrayLen	Gets the number of elements in the array.	Others	-	OK	OK	OK	page 8-33
Asc	Gets the character code of thespecified character.	String operation	OK	OK	OK	OK	page 8-34
AssignUnit	Registers the processing unit.	Flow control	-	OK	OK	-	page 8-35
Atn	Gets the arctangent of thespecified expression.	Arithmetic calculation	OK	OK	OK	OK	page 8-36
BusyOut	Sets the output state of theprocessing busy signal.	IO module control	-	OK	OK	OK	page 8-37
Call (Structured Subroutine Call)	Executes the specified subroutine with arguments added.	General instruction	OK	OK	OK	OK	page 8-38
Call (User-defined Function Execution)	Executes the registered userdefinedfunction.	Debug command	OK	OK	OK	OK	page 8-40
ChangeScene	Change the scene.	Scene control	-	-	OK	-	page 8-41

Command Name	Function	Classification	Unit Calculation Macro	Scene Control Macro	Communication Command Macro	Unit Macro	References
ChangeSceneGroup	Changes the scene group.	Scene group control	-	-	OK	-	page 8-42
CheckUnit	Checks the registration status of a processing unit.	Flow control	-	OK	OK	-	page 8-43
Chr$	Determining the character of the-specified character code.	String operation	OK	OK	OK	OK	page 8-44
ClearMeasureData	Clears the measurement resultsof the processing unit.	Measurement control	-	OK	OK	OK	page 8-46
ClearScene	Clears the scene data.	Scene control	-	-	OK	-	page 8-47
ClearSceneGroup	Clears scene group data Scene groupcontrol.	Scene group control	-	-	OK	-	page 8-48
Close	Closes up the file.	File control	OK	OK	OK	OK	page 8-49
CloseTextData	Close up a messages file.	Multilingual support message functions	OK	-	-	OK	page 8-50
Cls	Clears the display of the console window.	Others	OK	OK	OK	OK	page 8-52
Cont	Resumes execution of the program after it has beenstopped.	Debug command	OK	OK	OK	OK	page 8-53
ContinuousMeasure	Do the start or stop of thecontinuous measurement.	Measurement control	-	OK	OK	-	page 8-54
CopyMeasureImage	Copies the measurement image as an image of the Unit Macroprocessing unit.	Processing unit control	-	-	-	OK	page 8-55
CopyScene	Copies scene data.	Scene control	-	-	OK	-	page 8-56
CopySceneGroup	Copies scene group data.	Scene group control	-	-	OK	-	page 8-57
CopyUnit	Copies a processing unit.	Flow control	-	OK	OK	-	page 8-58
CopyUnitFigure	Copies figure data to theprocessing unit.	Processing unit control	-	OK	OK	OK	page 8-59
CopyUnitImage	Copies a processing unit image as a unit macro processing unit-image.	Processing unit control	-	-	-	OK	page 8-61
CopyUnitModel	Copies the model data of aprocessing unit.	Processing unit control	-	OK	OK	OK	page 8-62
Cos	Gets the cosine of the specifiedexpression.	Arithmetic calculation	OK	OK	OK	OK	page 8-64
Crspoint	Gets the intersection between 2 straight lines.	Arithmetic calculation	OK	OK	OK	OK	page 8-65
Date$	Reads out the date from the internal clock.	Arithmetic calculation	OK	OK	OK	OK	page 8-67

Command Name	Function	Classification	Unit Calculation Macro	Scene Control Macro	Communication Command Macro	Unit Macro	References
Debug	Set the program execution formand information output method.	Debug command	OK	OK	OK	OK	page 8-68
DebugPrint	Outputs debug information to the system status console window.	Debug command	OK	OK	OK	OK	page 8-69
DeleteProfileData	Deletes the settings file (.ini file) data.	Others	-	OK	OK	OK	page 8-71
DeleteUnit	Deletes a processing unit.	Flow control	-	OK	OK	-	page 8-72
Dim	Defines the array variable.	General instruction	OK	OK	OK	OK	page 8-73
DisableUnit	Switches the measurement ON/OFF of the specified unit.	Flow control	-	OK	OK	-	page 8-74
DisplaySubNo	Get the sub-image number of the displayed sub-image.	Image window control	-	-	-	OK	page 8-75
DisplayUnitNo	Gets the selection state of the processing unit number of theflow window.	Display control	-	OK	OK	-	page 8-76
Do Loop While	Repeatedly executes the statements between Do and Loop while the specifiedcondition meets.	General instruction	OK	OK	OK	OK	page 8-77
Dposline	Gets the shortest distance between the line and point.	Arithmetic calculation	OK	OK	OK	OK	page 8-78
DrawArc	Draw the arc on the image window.	Image window control	-	-	-	OK	page 8-80
DrawArcW	Draw the wide arc on the image window.	Image window control	-	-	-	OK	page 8-82
DrawBox	Draws a rectangle on the image window.	Image window control	-	-	-	OK	page 8-84
DrawCircle	Draw a circle on the image window.	Image window control	-	-	-	OK	page 8-86
DrawCircleW	Draw the wide circle on the image window.	Image window control	-	-	-	OK	page 8-88
DrawCursor	Draw the cross-hair cursor on the image window.	Image window control	-	-	-	OK	page 8-90
DrawEllipse	Draw the ellipse on the image window.	Image window control	-	-	-	OK	page 8-92
DrawFigure	Draw a figure on the image window.	Image window control	-	-	-	OK	page 8-94
DrawFillImage	Draw the fill image on the image window.	Image window control	-	-	-	OK	page 8-96
DrawJudgeText	Draws the judgement result of the character string on the textdisplay screen.	Text window control	-	-	-	OK	page 8-97
DrawLine	Draw a straight line on the image window.	Image window control	-	-	-	OK	page 8-98

Command Name	Function	Classification	Unit Calculation Macro	Scene Control Macro	Communication Command Macro	Unit Macro	References
DrawLineW	Draw the wide straight line on the image window.	Image window control	-	-	-	OK	page 8-100
DrawMeasureImage	Draw the measurement image on the image window.	Image window control	-	-	-	OK	page 8-102
DrawPoint	Draw a point on the image window.	Image window control	-	-	-	OK	page 8-103
DrawPolygon	Draw a polygon on the image window.	Image window control	-	-	-	OK	page 8-105
DrawSearchFigure	Draw the search figure on the image window.	Image window control	-	-	-	OK	page 8-107
DrawText	Draw a character string on the text window.	Text window control	-	-	-	OK	page 8-109
DrawTextG	Draw a character string on the image window.	Image window control	-	-	-	OK	page 8-111
DrawUnitImage	Display the "other unit image" on the image window.	Image window control	-	-	-	OK	page 8-113
Dskf	Gets the free space on disk drives.	File control	OK	OK	OK	OK	page 8-114
ElapsedTime	Gets the elapsed time since starting the measurement.	Others	OK	-	-	OK	page 8-115
Eof	Examines the end of the file.	File control	OK	OK	OK	OK	page 8-116
Erase	Releases array variable.	General instruction	OK	OK	OK	OK	page 8-118
Errcmnd$	Get the function name of the macro where an error occurred.	General instruction	OK	OK	OK	OK	page 8-119
Errno	Gets the error number.	General instruction	OK	OK	OK	OK	page 8-120
ErrorOut	Sets the output state of the Error(ERROR) signal.	IO module control	OK	OK	OK	OK	page 8-121
ExecuteErrorProc	Executes the error processing.	Others	-	OK	OK	-	page 8-123
ExecuteImageLogging	Executes image logging.	Others	-	-	-	OK	page 8-125
ExitFzProcess	Terminate the sensor controller.	Others	-	OK	OK	-	page 8-127
Exp	Gets the value of the exponential function of the basee natural logarithm.	Arithmetic calculation	OK	OK	OK	OK	page 8-128
Fcopy	Copies the file.	File control	OK	OK	OK	OK	page 8-129
FigureType	Gets the figure type that can be set for the processing unit.	Others	OK	-	-	OK	page 8-131
Fix	Gets the integer of a value by rounding off digits to the right ofthe decimal point.	Arithmetic calculation	OK	OK	OK	OK	page 8-130

Command Name	Function	Classification	Unit Calculation Macro	Scene Control Macro	Communication Command Macro	Unit Macro	References
For To Step Next	Repeats the statements between the For and Next statements.	General instruction	OK	OK	OK	OK	page 8-132
GetAll	Gets the input states of all input terminals.	IO module control	-	OK	OK	OK	page 8-134
GetGlobalData	Gets the global data.	Control global data	OK	OK	OK	OK	page 8-135
GetImageSize	Gets the image size of the processing unit image.	Processing unit control	OK	OK	OK	OK	page 8-136
GetImageWindow	Get the state of the image window.	Display control	-	OK	OK	OK	page 8-137
GetMeasureOut	Gets the external output setting for measurement results.	Measurement control	-	OK	OK	-	page 8-139
GetPlcData	Gets data read with the ReadPlcMemory function.	IO module control	-	OK	OK	OK	page 8-140
GetPollingState	Gets the polling state of the communication module.	IO module control	-	OK	OK	OK	page 8-142
GetProfileData	Gets the data of the setting file (.ini file).	Others	-	OK	OK	OK	page 8-145
GetPort	Gets the input state of the specified input terminal.	IO module control	-	OK	OK	OK	page 8-144
GetSceneData	Gets data related to the scene control macro or scene variables.	Scene control	OK	OK	OK	OK	page 8-146
GetSceneGroupData	Gets the scene group data with the specified identification name.	Scene group control	-	OK	OK	-	page 8-147
GetSystemData	Gets the system data or system variable.	System data	OK	OK	OK	OK	page 8-148
GetText$	Get a text data from a messages file.	Multilingual support message functions	OK	-	-	OK	page 8-150
GetTextWindow	Gets the state of the text window.	Display control	-	OK	OK	OK	page 8-152
GetUnitData	Gets the data of a processing unit.	Processing unit control	OK	OK	OK	OK	page 8-153
GetUnitFigure	Gets figure data to the processing unit.	Processing unit control	OK	OK	OK	OK	page 8-154
Gosub	Operate the specified subroutine.	General instruction	OK	OK	OK	OK	page 8-156
Goto	Moves the process to the statement line with a specified label.	General instruction	OK	OK	OK	OK	page 8-157
Hex$	Converts the value in the expression to the hexadecimal value in character string format.	String operation	OK	OK	OK	OK	page 8-158
If Then Else	Controls the process flow according to the specified condition.	General instruction	OK	OK	OK	OK	page 8-159

Command Name	Function	Classification	Unit Calculation Macro	Scene Control Macro	Communication Command Macro	Unit Macro	References
If Then Elseif Else Endif	Controls the process flow according to the specified condition.	General instruction	OK	OK	OK	OK	page 8-160
ImageFormat	Gets the image format of the image in the processing unit.	Processing unit control	OK	OK	OK	OK	page 8-162
ImageUpdate	Updates the image input from the camera.	Measurement control	-	OK	OK	-	page 8-164
Input#	Reads data from the file.	File control	OK	OK	OK	OK	page 8-165
Input$	Reads binary data from the file.	File control	OK	OK	OK	OK	page 8-166
InsertUnit	Inserts a processing unit.	Flow control	-	OK	OK	-	page 8-168
Int	Converts numeric value to integer value.	Arithmetic calculation	OK	OK	OK	OK	page 8-169
Isfile	Checks the attribute and the existence of the file.	File control	OK	OK	OK	OK	page 8-170
ItemCount	Gets the number of usable processing item types.	Control processing item	-	OK	OK	-	page 8-171
ItemIdent$	Gets the identification name of the processing item.	Control processing item	-	OK	OK	-	page 8-172
ItemInfo	Gets the processing item information.	Control processing item	-	OK	OK	-	page 8-173
ItemTitle$	Gets the processing item title.	Control processing item	-	OK	OK	-	page 8-174
JudgeOut	Sets the output state of the overall judgement signal.	IO module control	-	OK	OK	OK	page 8-175
Kill	Deletes a file.	File control	OK	OK	OK	OK	page 8-176
Language-String$	Gets the character string in the language according to the current language setting.	Others	OK	OK	OK	OK	page 8-178
LCase$	Converts an upper case letter to a lower case letter.	String operation	OK	OK	OK	OK	page 8-179
Left$	Extracts the specified length of characters from the left side of character string.	String operation	OK	OK	OK	OK	page 8-180
Len	Gets the length of the specified character string.	String operation	OK	OK	OK	OK	page 8-181
LineInput#	Reads the data of one line from the file.	File control	OK	OK	OK	OK	page 8-182
List	Outputs all or a part of program list in the system status console window.	Debug command	OK	OK	OK	OK	page 8-184
LoadBackup-Data	Loads the system + scene group 0 data.	Data save/load	-	-	OK	-	page 8-185

Command Name	Function	Classification	Unit Calculation Macro	Scene Control Macro	Communication Command Macro	Unit Macro	References
LoadScene	Loads the scene data.	Data save/load	-	-	OK	-	page 8-186
LoadSceneGroup	Loads the scene group data.	Data save/load	-	-	OK	-	page 8-187
LoadSystemData	Loads the system data.	Data save/load	-	OK	OK	-	page 8-188
LoadUnitData	Loads the processing unit data.	Data save/load	-	OK	OK	-	page 8-189
Log	Gets the natural logarithm.	Arithmetic calculation	OK	OK	OK	OK	page 8-191
Lsqumeth	Gets the approximate straight line from the coordinates of multiple points using the least squares method.	Arithmetic calculation	OK	OK	OK	OK	page 8-192
Measure	Executes measurement processing.	Measurement control	-	-	OK	-	page 8-193
MeasureDispG	Executes display of the measurement result of the processing unit.	Processing unit control	-	-	-	OK	page 8-195
MeasureDispI	Displays the image of the Sub-image number of the processing unit.	Processing unit control	-	-	-	OK	page 8-196
MeasureDispT	Displays detailed results from the processing unit.	Processing unit control	-	-	-	OK	page 8-197
MeasureId$	Gets the measurement identification.	Processing unit control	OK	-	-	OK	page 8-198
MeasureProc	Executes measurement processing in a processing unit.	Processing unit control	OK	OK	OK	OK	page 8-199
MeasureStart	Allows input of the measurement trigger.	Measurement control	-	OK	OK	-	page 8-200
MeasureStop	Prohibits input of the measurement trigger.	Measurement control	-	OK	OK	-	page 8-201
MemoryInfo	Gets the current memory usage (in size).	Others	OK	OK	OK	OK	page 8-203
MessageBox	Displays the message box.	Others	OK	OK	OK	OK	page 8-204
Mid$	Extracts a part from the character string.	String operation	OK	OK	OK	OK	page 8-205
Mkdir	Builds a directory.	File control	OK	OK	OK	OK	page 8-207
MOD	Gets the remainder.	Arithmetic calculation	OK	OK	OK	OK	page 8-208
MoveUnit	Moves a processing unit.	Flow control	-	OK	OK	-	page 8-209
NOT	Gets the "not" result (negation) of the expression.	Arithmetic calculation	OK	OK	OK	OK	page 8-210
Open For Append As#	Open the file in append mode.	File control	OK	OK	OK	OK	page 8-211

Command Name	Function	Classification	Unit Calculation Macro	Scene Control Macro	Communication Command Macro	Unit Macro	References
Open For Input As#	Open the file in reading mode.	File control	OK	OK	OK	OK	page 8-212
Open For Output As#	Opens the file in writing mode.	File control	OK	OK	OK	OK	page 8-213
OpenTextData	Opens a messages file.	Multilingual support message functions	OK	-	-	OK	page 8-215
Option Explicit	Finds undefined or duplicate variable that is defined in the Dim variable format.	Others	-	OK	OK	OK	page 8-217
OR	Gets the logical sum of two expressions.	Arithmetic calculation	OK	OK	OK	OK	page 8-218
Piece$	Extract the part of the character string which was separated by delimiter from the string.	String operation	OK	OK	OK	OK	page 8-219
Print	Outputs data in the system status console window.	Debug command	OK	OK	OK	OK	page 8-221
Print#	Outputs data in a file.	File control	OK	OK	OK	OK	page 8-222
PutAll	Sets the output state of all output terminals.	IO module control	-	OK	OK	OK	page 8-223
PutPort	Sets the output state of the specified output terminal.	IO module control	-	OK	OK	OK	page 8-225
RaiseErrorProcEvent	Notifies the error processing on UI screen.	Others	-	OK	OK	-	page 8-226
RaiseOptionEvent	Notifies option events to the UI screen.	Others	-	OK	OK	-	page 8-228
ReadPlcMemory	Reads a value from the PLC memory area.	IO module control	-	OK	OK	OK	page 8-229
ReceiveData	Receives data.	IO module control	-	OK	OK	OK	page 8-231
ReDim	Defines the array.	Others	OK	OK	OK	OK	page 8-233
RefreshImageWindow	Updates the image window.	Display control	-	OK	OK	-	page 8-234
RefreshJudgeWindow	Updates the judgement window.	Display control	-	OK	OK	-	page 8-235
RefreshTextWindow	Updates the text display window.	Display control	-	OK	OK	-	page 8-236
RefreshTimeWindow	Updates the display of the information window.	Display control	-	OK	OK	-	page 8-237
Rem	Puts a comment in the program.	Others	OK	OK	OK	OK	page 8-238
Remeasure	Executes remeasurement.	Measurement control	-	-	OK	-	page 8-239

Command Name	Function	Classification	Unit Calculation Macro	Scene Control Macro	Communication Command Macro	Unit Macro	References
RenumUnitNo	Gets the processing unit number after flow edit.	Others	OK	-	-	OK	page 8-241
RGB	Gets the color value.	Arithmetic calculation	OK	-	-	OK	page 8-242
Right$	Extracts the specified length of characters from the right side of character string.	String operation	OK	OK	OK	OK	page 8-243
Rmdir	Deletes a directory.	File control	OK	OK	OK	OK	page 8-244
RunOut	Sets the output state of the RUN signal.	IO module control	-	OK	OK	OK	page 8-245
SaveBackupData	Saves the system + scene group 0 data.	Data save/load	-	OK	OK	-	page 8-247
SaveData	Saves the data to the controller.	Data save/load	-	OK	OK	-	page 8-248
SaveImage	Saves image data.	Measurement control	-	OK	OK	-	page 8-249
SaveMeasureImage	Saves the measurement image of the processing unit.	Processing unit control	-	-	-	OK	page 8-250
SaveScene	Saves the scene data.	Data save/load	-	OK	OK	-	page 8-252
SaveSceneGroup	Saves the scene group data.	Data save/load	-	OK	OK	-	page 8-253
SaveSystemData	Saves the system data.	Data save/load	-	OK	OK	-	page 8-254
SaveUnitData	Saves a processing unit.	Data save/load	-	OK	OK	-	page 8-255
SceneCount	Gets the number of scenes that can be used.	Scene control	-	OK	OK	-	page 8-257
SceneDescription$	Gets the scene description.	Scene control	-	OK	OK	-	page 8-258
SceneGroupCount	Gets the number of usable scene groups.	Scene group control	-	OK	OK	-	page 8-259
SceneGroupNo	Gets the scene group number of the current scene group.	Scene group control	-	OK	OK	-	page 8-260
SceneGroupTitle$	Gets the title of the scene group.	Scene group control	-	OK	OK	-	page 8-261
SceneMaker$	Gets the scene creator.	Scene control	-	OK	OK	-	page 8-262
SceneNo	Gets the scene number of the current scene.	Scene control	-	-	OK	-	page 8-263
SceneTitle$	Gets the scene title.	Scene control	-	OK	OK	-	page 8-264
ScreenCapture	Saves the capture of the screen.	Others	-	OK	OK	-	page 8-265

Command Name	Function	Classification	Unit Calculation Macro	Scene Control Macro	Communication Command Macro	Unit Macro	References
Select Case Case Else End Select	Controls the process flow according to the specified condition.	General instruction	OK	OK	OK	OK	page 8-266
SendData	Sends data.	IO module control	-	OK	OK	OK	page 8-268
SendString	Sends the character string data.	IO module control	-	OK	OK	OK	page 8-270
SetDisplayUnitNo	Sets the processing unit number in the flow window to the selected state.	Display control	-	OK	OK	-	page 8-271
SetDrawStyle	Set the drawing attributes of the graphic figure.	Image window control	-	-	-	OK	page 8-272
SetForegroundLine	This command specifies the object line for operation in Multiline Random trigger mode or Non-stop Adjustment mode.	Others	-	OK	OK	OK	page 8-274
SetGlobalData	Sets the global data.	Control global data	OK	OK	OK	OK	page 8-275
SetImageWindow	Sets the state of the image window.	Display control	-	OK	OK	OK	page 8-276
SetMeasureImage	Sets the measurement image of the processing unit.	Processing unit control	-	-	-	OK	page 8-279
SetMeasureOut	Sets the external output setting for the measurement result.	Measurement control	-	OK	OK	-	page 8-280
SetPlcData	Creates the data that is written with the WritePlcMemory function.	IO module control	-	OK	OK	OK	page 8-281
SetPollingState	Sets the execution status of the communication module.	IO module control	-	OK	OK	OK	page 8-283
SetProfileData	Sets the data of the setting file (.ini file).	Others	-	OK	OK	OK	page 8-285
SetSceneData	Sets data for the scene control macro or scene variables.	Scene control	OK	OK	OK	OK	page 8-286
SetSceneDescription	Sets the scene description.	Scene control	-	OK	OK	-	page 8-288
SetSceneGroupData	Sets the scene group data with the specified identification name.	Scene group control	-	OK	OK	-	page 8-289
SetSceneGroupTitle	Sets the title of the scene group.	Scene group control	-	OK	OK	-	page 8-290
SetSceneMaker	Sets the creator of the scene.	Scene control	-	OK	OK	-	page 8-291
SetSceneTitle	Sets the title of a scene.	Scene control	-	OK	OK	-	page 8-292
SetStop	Sets the conditions for stopping program execution.	Debug command	OK	OK	OK	OK	page 8-293
SetSystemData	Sets the system data or system variable.	System data	OK	OK	OK	OK	page 8-295

Command Name	Function	Classification	Unit Calculation Macro	Scene Control Macro	Communication Command Macro	Unit Macro	References
SetTextStyle	Set the draw attributes of the character string.	Image window control	-	-	-	OK	page 8-297
SetTextWindow	Set the draw attributes of the character string.	Display control	-	OK	OK	-	page 8-299
SetUnitData	Sets the data of a processing unit.	Processing unit control	OK	OK	OK	OK	page 8-300
SetUnitFigure	Sets the figure data of the processing unit.	Processing unit control	-	OK	OK	OK	page 8-301
SetUnitJudge	Sets the judgement result of a processing unit.	Processing unit control	OK	-	-	OK	page 8-303
SetUnitTitle	Sets the title of a processing unit.	Processing unit control	-	OK	OK	OK	page 8-304
SetUserSubroutine	Register a user-defined function that has been defined in the external DDL file.	Debug command	OK	OK	OK	OK	page 8-305
SetVar	Sets all variables with the specified variable names.	Others	OK	OK	OK	OK	page 8-307
Sin	Gets the sine of the specified expression.	Arithmetic calculation	OK	OK	OK	OK	page 8-309
Sqr	Determines the square root.	Arithmetic calculation	OK	OK	OK	OK	page 8-310
StartTimer	Starts the elapsed time measurement.	Others	OK	OK	OK	OK	page 8-311
Stop	Stops program execution.	Debug command	OK	OK	OK	OK	page 8-312
Str$	Converts a numeric value in the numeric character string.	String operation	OK	OK	OK	OK	page 8-314
Str2$	Converts a value to a numeric character string in the specified formats.	String operation	OK	OK	OK	OK	page 8-315
Sub - End Sub	Defines the subroutine.	General instruction	OK	OK	OK	OK	page 8-318
SubList	Outputs a list of subroutines to the console window.	General instruction	OK	OK	OK	OK	page 8-320
SystemInfo	Gets information on the system.	Others	OK	OK	OK	OK	page 8-321
SystemReset	Reboots the sensor controller.	Others	-	-	OK	-	page 8-322
Tan	Gets the tangent of the specified expression.	Arithmetic calculation	OK	OK	OK	OK	page 8-323
TestMeasure	Executes the test measurement.	Measurement control	-	OK	OK	-	page 8-324
Time$	Reads out the clock time from the internal clock.	Arithmetic calculation	OK	OK	OK	OK	page 8-326
Timer	Gets the elapsed time.	Others	OK	OK	OK	OK	page 8-327

Command Name	Function	Classification	Unit Calculation Macro	Scene Control Macro	Communication Command Macro	Unit Macro	References
TotalJudge	Gets the total judgement result.	Processing unit control	OK	-	-	OK	page 8-328
TransformAngle	Applies the calibration result and position correction amount in the angle value.	Processing unit control	OK	-	-	OK	page 8-330
TransformArea	Applies the calibration result and position correction amount in the area value.	Processing unit control	OK	-	-	OK	page 8-331
TransformDist	Applies a calibration result and position correction amount to a distance value.	Processing unit control	OK	-	-	OK	page 8-332
Transform Line	Applies the calibration result and position correction amount to a line component value.	Processing unit control	OK	-	-	OK	page 8-333
TransformXY	Applies the calibration result and position correction amount to coordinate values.	Processing unit control	OK	-	-	OK	page 8-335
Try Catch End Try	Detects an error occurrence and executes an exception process.	General instruction	OK	OK	OK	OK	page 8-337
UCase$	Converts an lower case letter to a upper case letter.	String operation	OK	OK	OK	OK	page 8-339
UnitCount	Gets the number of registered processing units.	Flow control	-	OK	OK	-	page 8-340
UnitData	Gets the numerical data of a processing unit.	Processing unit control	OK	OK	OK	OK	page 8-341
UnitData$	Gets the character string data of the specified processing unit.	Processing unit control	OK	OK	OK	OK	page 8-342
UnitData2	Gets the drawing coordinate data of a processing unit.	Processing unit control	OK	-	-	OK	page 8-343
UnitInfo	Gets the processing unit information.	Processing unit control	OK	OK	OK	OK	page 8-345
UnitItemIdent$	Gets the processing item identification name of the specified processing unit.	Processing unit control	OK	OK	OK	OK	page 8-347
UnitJudge	Gets the judgement result of a processing unit.	Processing unit control	OK	-	-	OK	page 8-348
UnitNo	Gets the processing unit number.	Processing unit control	OK	-	-	OK	page 8-349
UnitTitle$	Gets the title of a processing unit.	Processing unit control	OK	OK	OK	OK	page 8-350
Ut	Gets a processing unit number based on the specified unit label.	Scene control	OK	OK	OK	OK	page 8-351
Val	Converts a numeric character string to numeric value.	String operation	OK	OK	OK	OK	page 8-352

Command Name	Function	Classification	Unit Calculation Macro	Scene Control Macro	Communication Command Macro	Unit Macro	References
VarList	Outputs a list of the values of the specified variables in the system status console window.	Debug command	OK	OK	OK	OK	page 8-353
Varpop	Restores the value of the variables that are saved temporarily.	Others	OK	OK	OK	OK	page 8-355
Varpush	Saves the value of the variables that are saved temporarily.	Others	OK	OK	OK	OK	page 8-358
VarSave	Saves the values of the variables in the scene data.	Others	-	OK	-	-	page 8-361
Wait	Pauses the program process for the specified amount of time elapses.	Others	-	OK	OK	OK	page 8-362
WritePlcMemory	Writes values in the PLC memory area.	IO module control	-	OK	OK	OK	page 8-363
XOR	Gets the exclusive disjunction (XOR) of two expressions.	Arithmetic calculation	OK	OK	OK	OK	page 8-365

8-1-2 Function-based Index

General Instructions

Command	Function	References
Call (Structured Subroutine Call)	Executes the specified subroutine with arguments added.	page 8-38
Dim	Defines the array variable.	page 8-73
Do Loop While	Repeatedly executes the statements between Do and Loop while the specifiedcondition meets.	page 8-77
Erase	Releases array variable.	page 8-118
Errcmnd$	Get the function name of the macro where an error occurred.	page 8-119
Errno	Gets the error number.	page 8-120
For To Step Next	Repeats the statements between the For and Next statements.	page 8-132
Gosub	Operate the specified subroutine.	page 8-156
Goto	Moves the process to the statement line with a specified label.	page 8-157
If Then Else	Controls the process flow according to the specified condition.	page 8-159
If Then Elseif Else EndIf	Controls the process flow according to the specified condition.	page 8-160
Option Explicit	Finds undefined or duplicate variable that is defined in the Dim variable format.	page 8-217
Redim	Defines the array.	page 8-233
Select Case Case Else End Select	Controls the process flow according to the specified condition.	page 8-266
Sub - End Sub	Defines the subroutine.	page 8-318
SubList	Outputs a list of subroutines to the console window.	page 8-320
Try Catch End Try	Detects an error occurrence and executes an exception process.	page 8-337

Arithmetic Calculations

Command	Function	References
Abs	Gets the absolute value of the specified expression.	page 8-24
AND	Gets the logical product of twoexpressions.	page 8-28
ApproximationCircle	Gets the approximate circle.	page 8-31
Atn	Gets the arctangent of thespecified expression.	page 8-36
Cos	Gets the cosine of the specifiedexpression.	page 8-64
Crspoint	Gets the intersection between 2 straight lines.	page 8-65
Date$	Reads out the date from the internal clock.	page 8-67
Dposline	Gets the shortest distance between the line and point.	page 8-78
Exp	Gets the value of the exponential function of the basee natural logarithm.	page 8-128
Fix	Gets the integer of a value by rounding off digits to the right ofthe decimal point.	page 8-130
Int	Converts numeric value to integer value.	page 8-169
Log	Gets the natural logarithm.	page 8-191

Command	Function	References
Lsqumeth	Gets the approximate straight line from the coordinates of multiple points using the least squares method.	page 8-192
MOD	Gets the remainder.	page 8-208
NOT	Gets the "not" result (negation) of the expression.	page 8-210
OR	Gets the logical sum of two expressions.	page 8-218
RGB	Gets the color value.	page 8-242
Sin	Gets the sine of the specified expression.	page 8-309
Sqr	Determines the square root.	page 8-310
Tan	Gets the tangent of the specified expression.	page 8-323
Time$	Reads out the clock time from the internal clock.	page 8-326
XOR	Gets the exclusive disjunction (XOR) of two expressions.	page 8-365

● String Operations

Command	Function	References
Asc	Gets the character code of thespecified character.	page 8-34
Chr$	Determining the character of thespecified character code.	page 8-44
Hex$	Converts the value in the expression to the hexadecimal value in character string format.	page 8-158
LCase$	Converts an upper case letter to a lower case letter.	page 8-179
Left$	Extracts the specified length of characters from the left side of character string.	page 8-180
Len	Gets the length of the specified character string.	page 8-181
Mid$	Extracts a part from the character string.	page 8-205
Piece$	Extract the part of the character string which was separated by delimiter from the string.	page 8-219
Right$	Extracts the specified length of characters from the right side of character string.	page 8-243
Str$	Converts a numeric value in the numeric character string.	page 8-314
Str2$	Converts a value to a numeric character string in the specified formats.	page 8-315
UCase$	Converts an lower case letter to a upper case letter.	page 8-339
Val	Converts a numeric character string to numeric value.	page 8-352

● Scene Controls

Command	Function	References
ChangeScene	Change the scene.	page 8-41
ClearScene	Clears the scene data.	page 8-47
CopyScene	Copies scene data.	page 8-56
GetSceneData	Gets data related to the scene control macro or scene variables.	page 8-146
SceneCount	Gets the number of scenes that can be used.	page 8-257
SceneDescription$	Gets the scene description.	page 8-258
SceneMaker$	Gets the scene creator.	page 8-262
SceneNo	Gets the scene number of the current scene.	page 8-263
SceneTitle$	Gets the scene title.	page 8-264
SetSceneData	Sets data for the scene control macro or scene variables.	page 8-286

Command	Function	References
SetSceneDescription	Sets the scene description.	page 8-288
SetSceneMaker	Sets the creator of the scene.	page 8-291
SetSceneTitle	Sets the title of a scene.	page 8-292
Ut	Gets a processing unit number based on the specified unit label.	page 8-351

● Scene Group Controls

Command	Function	References
ChangeSceneGroup	Changes the scene group.	page 8-42
ClearSceneGroup	Clears scene group data Scene groupcontrol.	page 8-48
CopySceneGroup	Copies scene group data.	page 8-57
GetSceneGroupData	Gets the scene group data with the specified identification name.	page 8-147
SceneGroupCount	Gets the number of usable scene groups.	page 8-259
SceneGroupNo	Gets the scene group number of the current scene group.	page 8-260
SetSceneGroupData	Sets the scene group data with the specified identification name.	page 8-289
SceneGroupTitle$	Gets the title of the scene group.	page 8-261
SetSceneGroupTitle	Sets the title of the scene group.	page 8-290

● Processing Item Controls

Command	Function	References
ItemCount	Gets the number of usable processing item types.	page 8-171
ItemIdent$	Gets the processing item information.	page 8-172
ItemInfo	Gets the identification name of the processing item.	page 8-173
ItemTitle$	Gets the processing item title.	page 8-174

● Processing Flow Controls

Command	Function	References
AssignUnit	Registers the processing unit.	page 8-35
CheckUnit	Checks the registration status of a processing unit.	page 8-43
CopyUnit	Copies a processing unit.	page 8-58
DeleteUnit	Deletes a processing unit.	page 8-72
DisableUnit	Switches the measurement ON/OFF of the specified unit.	page 8-74
InsertUnit	Inserts a processing unit.	page 8-168
MoveUnit	Moves a processing unit.	page 8-209
UnitCount	Gets the number of registered processing units.	page 8-340

● Processing Unit Controls

Command	Function	References
CopyMeasureImage	Copies the measurement image as an image of the Unit Macroprocessing unit.	page 8-55
CopyUnitFigure	Copies figure data to theprocessing unit.	page 8-59

Command	Function	References
CopyUnitImage	Copies a processing unit image as a unit macro processing unitimage.	page 8-61
CopyUnitModel	Copies the model data of aprocessing unit.	page 8-62
GetImageSize	Gets the image size of the processing unit image.	page 8-136
GetUnitData	Gets the data of a processing unit.	page 8-153
GetUnitFigure	Gets figure data to the processing unit.	page 8-154
ImageFormat	Gets the image format of the image in the processing unit.	page 8-162
MeasureDispG	Executes display of the measurement result of the processing unit.	page 8-195
MeasureDispI	Displays the image of the Sub-image number of the processing unit.	page 8-196
MeasureDispT	Displays detailed results from the processing unit.	page 8-197
MeasureId$	Gets the measurement identification.	page 8-198
MeasureProc	Executes measurement processing in a processing unit.	page 8-199
SaveMeasureImage	Saves the measurement image of the processing unit.	page 8-250
SetMeasureImage	Sets the measurement image of the processing unit.	page 8-279
SetUnitData	Sets the data of a processing unit.	page 8-300
SetUnitFigure	Sets the figure data of the processing unit.	page 8-301
SetUnitJudge	Sets the judgement result of a processing unit.	page 8-303
SetUnitTitle	Sets the title of a processing unit.	page 8-304
TotalJudge	Gets the total judgement result.	page 8-328
TransformAngle	Applies the calibration result and position correction amount in the angle value.	page 8-330
TransformArea	Applies the calibration result and position correction amount in the area value.	page 8-331
TransformDist	Applies a calibration result and position correction amount to a distance value.	page 8-332
TransformLine	Applies the calibration result and position correction amount to a line component value.	page 8-333
TransformXY	Applies the calibration result and position correction amount to coordinate values.	page 8-335
UnitData	Gets the numerical data of a processing unit.	page 8-341
UnitData$	Gets the character string data of the specified processing unit.	page 8-342
UnitData2	Gets the drawing coordinate data of a processing unit.	page 8-343
UnitInfo	Gets the processing unit information.	page 8-345
UnitItemIdent$	Gets the processing item identification name of the specified processing unit.	page 8-347
UnitJudge	Gets the judgement result of a processing unit.	page 8-348
UnitNo	Gets the processing unit number.	page 8-349
UnitTitle$	Gets the title of a processing unit.	page 8-350

● Measurement Controls

Command	Function	References
ClearMeasureData	Clears the measurement resultsof the processing unit.	page 8-46
ContinuousMeasure	Do the start or stop of thecontinuous measurement.	page 8-54
GetMeasureOut	Gets the external output setting for measurement results.	page 8-139
ImageUpdate	Updates the image input from the camera.	page 8-164

Command	Function	References
Measure	Executes measurement processing.	page 8-193
MeasureStart	Allows input of the measurement trigger.	page 8-200
MeasureStop	Prohibits input of the measurement trigger.	page 8-201
Remeasure	Executes remeasurement.	page 8-239
SaveImage	Saves image data.	page 8-249
SetMeasureOut	Sets the external output setting for the measurement result.	page 8-280
TestMeasure	Executes the test measurement.	page 8-324

● IO Module Controls

Command	Function	References
BusyOut	Sets the output state of theprocessing busy signal.	page 8-37
ErrorOut	Sets the output state of the Error(ERROR) signal.	page 8-121
GetAll	Gets the input states of all input terminals.	page 8-134
GetPlcData	Gets data read with the ReadPlcMemory function.	page 8-140
GetPollingState	Gets the polling state of the communication module.	page 8-142
GetPort	Gets the input state of the specified input terminal.	page 8-144
JudgeOut	Sets the output state of the overall judgement signal.	page 8-175
PutAll	Sets the output state of all output terminals.	page 8-223
PutPort	Sets the output state of the specified output terminal.	page 8-225
ReadPlcMemory	Reads a value from the PLC memory area.	page 8-229
ReceiveData	Receives data.	page 8-231
RunOut	Sets the output state of the RUN signal.	page 8-245
SendData	Sends data.	page 8-268
SendString	Sends the character string data.	page 8-270
SetPlcData	Creates the data that is written with the WritePlcMemory function.	page 8-281
SetPollingState	Sets the execution status of the communication module.	page 8-283
WritePlcMemory	Writes values in the PLC memory area.	page 8-363

● Display Controls

Command	Function	References
DisplayUnitNo	Gets the selection state of the processing unit number of theflow window.	page 8-76
GetImageWindow	Get the state of the image window.	page 8-137
GetTextWindow	Gets the state of the text window.	page 8-152
RefreshImageWindow	Updates the image window.	page 8-234
RefreshJudgeWindow	Updates the judgement window.	page 8-235
RefreshTextWindow	Updates the text display window.	page 8-236
RefreshTimeWindow	Updates the display of the information window.	page 8-237
SetDisplayUnitNo	Sets the processing unit number in the flow window to the selected state.	page 8-271
SetImageWindow	Sets the state of the image window.	page 8-276
SetTextWindow	Set the draw attributes of the character string.	page 8-299

Image Window Controls

Command	Function	References
DisplaySubNo	Get the sub-image number of the displayed sub-image.	page 8-75
DrawArc	Draw the arc on the image window.	page 8-80
DrawArcW	Draw the wide arc on the image window.	page 8-82
DrawBox	Draws a rectangle on the image window.	page 8-84
DrawCircle	Draw a circle on the image window.	page 8-86
DrawCircleW	Draw the wide circle on the image window.	page 8-88
DrawCursor	Draw the cross-hair cursor on the image window.	page 8-90
DrawEllipse	Draw the ellipse on the image window.	page 8-92
DrawFigure	Draw a figure on the image window.	page 8-94
DrawFillImage	Draw the fill image on the image window.	page 8-96
DrawLine	Draw a straight line on the image window.	page 8-98
DrawLineW	Draw the wide straight line on the image window.	page 8-100
DrawMeasureImage	Draw the measurement image on the image window.	page 8-102
DrawPoint	Draw a point on the image window.	page 8-103
DrawPolygon	Draw a polygon on the image window.	page 8-105
DrawSearchFigure	Draw the search figure on the image window.	page 8-107
DrawTextG	Draw a character string on the image window.	page 8-111
DrawUnitImage	Display the "other unit image" on the image window.	page 8-113
SetDrawStyle	Set the drawing attributes of the graphic figure.	page 8-272
SetTextStyle	Set the draw attributes of the character string.	page 8-297

Text Window Controls

Command	Function	References
DrawJudgeText	Draws the judgement result of the character string on the textdisplay screen.	page 8-97
DrawText	Draw a character string on the text window.	page 8-109

System Data

Command	Function	References
AddSystemData	Adds the system data.	page 8-27
GetSystemData	Gets the system data or system variable.	page 8-148
SetSystemData	Sets the system data or system variable.	page 8-295

Global Data

Command	Function	References
AddGlobalData	Adds the global data.	page 8-26
GetGlobalData	Gets the global data.	page 8-135
SetGlobalData	Sets the global data.	page 8-275

Data Save/Load

Command	Function	References
LoadBackupData	Loads the system + scene group 0 data.	page 8-185

Command	Function	References
LoadScene	Loads the scene data.	page 8-186
LoadSceneGroup	Loads the scene group data.	page 8-187
LoadSystemData	Loads the system data.	page 8-188
LoadUnitData	Loads the processing unit data.	page 8-189
SaveBackupData	Saves the system + scene group 0 data.	page 8-247
SaveData	Saves the data to the controller.	page 8-248
SaveScene	Saves the scene data.	page 8-252
SaveSceneGroup	Saves the scene group data.	page 8-253
SaveSystemData	Saves the system data.	page 8-254
SaveUnitData	Saves a processing unit.	page 8-255

● File Controls

Command	Function	References
Close	Closes up the file.	page 8-49
Dskf	Gets the free space on disk drives.	page 8-114
Eof	Examines the end of the file.	page 8-116
Fcopy	Copies the file.	page 8-129
Input#	Reads data from the file.	page 8-165
Input$	Reads binary data from the file.	page 8-166
Isfile	Checks the attribute and the existence of the file.	page 8-170
Kill	Deletes a file.	page 8-176
Line Input#	Reads the data of one line from the file.	page 8-182
Mkdir	Builds a directory.	page 8-207
Open For Append As#	Open the file in append mode.	page 8-211
Open For Input As#	Open the file in reading mode.	page 8-212
Open For Output As#	Opens the file in writing mode.	page 8-213
Print#	Outputs data in a file.	page 8-222
Rmdir	Deletes a directory.	page 8-244

● Multilingual Support Message Functions

Command	Function	References
CloseTextData	Close up a messages file.	page 8-50
GetText	Get a text data from a messages file.	page 8-150
OpenTextData	Opens a messages file.	page 8-215

● Debug Commands

Command	Function	References
Call (User-defined Function Execution)	Executes the registered userdefinedfunction.	page 8-40
Cont	Resumes execution of the program after it has beenstopped.	page 8-53
Debug	Set the program execution formand information output method.	page 8-68
DebugPrint	Outputs debug information to the system status console window.	page 8-69

Command	Function	References
List	Outputs all or a part of program list in the system status console window.	page 8-184
Print	Outputs data in the system status console window.	page 8-221
SetStop	Sets the conditions for stopping program execution.	page 8-293
SetUserSubroutine	Register a user-defined function that has been defined in the external DDL file.	page 8-305
Stop	Stops program execution.	page 8-312
VarList	Outputs a list of the values of the specified variables in the system status console window.	page 8-353

● Others

Command	Function	References
AbsolutePath$	Gets the absolute path of the folder assigned to the Quick access name.	page 8-25
ApplicationPath$	Gets the pass name regarding application software.	page 8-29
ApplicationVersion$	Gets the version information of application software.	page 8-30
ArrayDims	Gets the number of dimensions in the array.	page 8-32
ArrayLen	Gets the number of elements in the array.	page 8-33
Cls	Clears the display of the console window.	page 8-52
DeleteProfileData	Deletes the settings file (.ini file) data.	page 8-71
ElapsedTime	Gets the elapsed time since starting the measurement.	page 8-115
ExecuteErrorProc	Executes the error processing.	page 8-123
ExecuteImageLogging	Executes image logging.	page 8-125
ExitFzProcess	Terminate the sensor controller.	page 8-127
FigureType	Gets the figure type that can be set for the processing unit.	page 8-131
GetProfileData	Gets the data of the setting file (.ini file).	page 8-145
LanguageString$	Gets the character string in the language according to the current language setting.	page 8-178
MemoryInfo	Gets the current memory usage (in size).	page 8-203
MessageBox	Displays the message box.	page 8-204
RaiseErrorProcEvent	Notifies the error processing on UI screen.	page 8-226
RaiseOptionEvent	Notifies option events to the UI screen.	page 8-228
Rem	Puts a comment in the program.	page 8-238
RenumUnitNo	Gets the processing unit number after flow edit.	page 8-241
ScreenCapture	Saves the capture of the screen.	page 8-265
SetForegroundLine	This command specifies the object line for operation in Multi-line Random trigger mode or Non-stop Adjustment mode.	page 8-274
SetProfileData	Sets the data of the setting file (.ini file).	page 8-285
SetVar	Sets all variables with the specified variable names.	page 8-307
StartTimer	Starts the elapsed time measurement.	page 8-311
SystemInfo	Gets information on the system.	page 8-321
SystemReset	Reboots the sensor controller.	page 8-322
Timer	Gets the elapsed time.	page 8-327
VarPop	Restores the value of the variables that are saved temporarily.	page 8-355
VarPush	Saves the value of the variables that are saved temporarily.	page 8-358
VarSave	Saves the values of the variables in the scene data.	page 8-361

Command	Function	References
Wait	Pauses the program process for the specified amount of time elapses.	page 8-362

8-2 Macro Command Reference

Abs

Gets the absolute value of the specified expression.

Format:
Abs(<expression>)

Parameter:

Parameter name	Data type	Description
<expression>	Integer Real number	Expression to get the absolute value

Return value:
Returns a double precision real absolute value.

Description:
Gets the absolute value of the expression specified in the <expression> parameter.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Gets the difference between the two points (X1, Y1) and (X2, Y2).

```
X1# = 100
Y1# = 200
X2# = 200
Y2# = 100

DX# = Abs(X1# - X2#)
DY# = Abs(Y1# - Y2#)
```

The result is shown below.

```
DX# = 100
DY# = 100
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

GetUnitData on page 8-153 *UnitData* on page 8-341

AbsolutePath$

Gets the absolute path of the folder assigned to the Quick access name.

Format:
AbsolutePath$(<pathName>)

Parameter:

Parameter name	Data type	Description
<pathName>	Character string	Quick access Ident name For additional information on the Quick access, refer to *Register frequently used folders for easy access [Quick access setting tool]* in *Vision Sensor FH/FHV Series Vision System User's Manual (Z365)*.

Return value:
Returns the absolute path.

Description:
Gets the absolute path assigned to the Quick access name specified in the <pathName> parameter.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Access the folder by obtaining the **Absolute path** of the **Ident name** "Logging" in the folder set with the **Quick access setting tool**.

```
LoggingPATH$ = AbsolutePath$("<Logging>")
testmeasure LoggingPATH$ + "test.ifz"
```

Usable Modules:
Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 6.40 or later

Related Items:
None.

AddGlobalData

Adds the global data.

Format:
AddGlobalData <dataIdent>, <data>

Parameter:

Parameter name	Data type	Description
<dataIdent>	Character string	Identification name of the global data to add
<data>	Integer Real number Character string	Value added in the global data

Return value:
None.

Description:
Adds the data of the identification name specified in the <dataIdent> parameter to the global data, and sets the value specified in the <data> parameter in the added data.
If the global data of the specified identification name already exists, the process ends without taking any action.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a character string longer than 255 characters is specified in the <dataIdent> parameter, a "String too long" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Adds the global data that has the identification name "ABC", and sets 1 as the value.

```
Rem Add global data "ABC", and set 1 as the value
AddGlobalData "ABC", 1

Rem Get the value (integer value) set in the global data "ABC", and store in the v
ariable DATA&.
GetGlobalData "ABC", DATA&
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

GetGlobalData on page 8-135
SetGlobalData on page 8-275

AddSystemData

Adds the system data.

Format:

AddSystemData <dataIdent0>, <dataIdent1>, <data>

Parameter:

Parameter name	Data type	Description
<dataIdent0>	Character string	Data identification name of identification information 0 of system data to be added (specify "PanDA")
<dataIdent1>	Character string	Data identification name of identification information 1 of system data to be added
<data>	Integer Real number Character string	Value of the system data to add

Return value:

None.

Description:

Adds the data of identification information 1 specified in the <dataIdent1> parameter, which belongs to identification information 0 specified in the <dataIdent0> parameter, to the system data, and sets the value specified in the <data> parameter in the added data.
If an identification name other than "PanDA" is specified in the <dataIdent0> parameter, an "Illegal function call" error will occur.
If the system data of identification information 1 that belongs to the specified identification information 0 is already registered, no action is taken and the process ends.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If an identification name that does not exist is specified in the <dataIdent0> parameter, an "Illegal function call" error will occur.
If a character string longer than 255 characters is specified in the <dataIdent1> parameter, a "String too long" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:

None.

Example:

Adds the data of identification information 1, "LoggingCount", to the system data of identification information 0, "PanDA". Sets 20 for the setting data.

```
AddSystemData "PanDA", "LoggingCount", 20
```

Usable Modules:

Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:

Version 3.50 or later

Related Items:

AddGlobalData on page 8-26
GetSystemData on page 8-148
SetSystemData on page 8-295

AND

Gets the logical product of twoexpressions.

Format:
<expression1> AND <expression2>

Parameter:

Parameter name	Data type	Description
<expression1>	Integer	Expression to calculate the logical product
<expression2>	Integer	

Return value:
Returns the logical product as an integer value.

Description:
Gets the logical product by bit of the expression specified in the <expression1> parameter and the expression specified in the <expression2> parameter.
When the values of the <expression1> parameter and <expression2> parameter are double precision real values, the decimal part of the returned logical product is rounded off.
This can also be used as an And condition in an If statement. For details on the logical expression, refer to *4-1-5 Operator* on page 4-12.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a value outside the range -2147483648 to 2147483647 is specified as an integer parameter, an "Overflow" error will occur.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Gets the logical product of variable X and variable Y.

```
X& = 15
Y& = 8

DATA& = X& AND Y&
```

The result is shown below.

```
DATA& = 8
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:

Version 3.50 or later

Related Items:

GetUnitData on page 8-153
OR on page 8-218
XOR on page 8-365
NOT on page 8-210
UnitData on page 8-341

ApplicationPath$

Gets the pass name regarding application software.

Format:
ApplicationPath$(<kind>)

Parameter:

Parameter name	Data type	Description
<kind>	Character string	The following type of pass name related to application software. 0: Pass name installed the application software. 1: Standard pass name for file controlling 2: Default pass name of saved data 3: Pass name of saved system data 10000 + Scene group number: Pass name of saved Scene grope data for corresponding number

Return value:
Returns the information of pass name (character strings) using <kind>.

Description:
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
Type mismatch error is occurred when wrong data is specified as parameter. Illegal function call error is not occurred even if non-exist number, values, combination of data or values.
An error of "String too long" will be occurred when you specify the character strings as character strings type exceeds 255 characters.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Registers the image data as a registered image with non-procedure communication.

```
Rem Acquires the saved pass name of registered image.
USBPATH$ = ApplicationPath$(2)
FILENAME$ = ARGUMENTSTRING$(0)
DATAPATH$ = USBPATH$ + "RegisteredImage\" + FILENAME$ + ".ifz"

Rem Registers the latest input image to registered image.
SaveImage -1, DATAPATH$
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 5.40 or later

Related Items:

SaveImage on page 8-249

ApplicationVersion$

Gets the version information of application software.

Format:
ApplicationVersion$(<kind>)

Parameter:

Parameter name	Data type	Description
<kind>	Integer	The following number is for the version information of application software. 0: Model information 1: Version information 2: Created day

Return value:
Return the version information (character strings) of application software using <kind> parameter.

Description:
Acquires the version information of application software.
Information according to the contents of Version.ini file in the application software will be returned.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
Type mismatch error is occurred when wrong data is specified as parameter. Illegal function call error is not occurred even if non-exist number, values, combination of data or values.
If a value outside the range -2147483648 to 2147483647 is specified as an integer parameter, an "Overflow" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Sends the version information of application software with UDP non-procedure.

```
Rem Acquires the version information of system.
APPVER$ = ApplicationVersion$(1)
SendString "UdpNormal", APPVER$
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 5.40 or later

Related Items:

SendString on page 8-270

ApproximationCircle

Gets the approximate circle.

Format:

ApproximationCircle <count>, <x()>, <y()>, <centerX>, <centerY>, <radius>

Parameter:

Parameter name	Data type	Description
<count>	Integer	Number of the specified coordinate
<x()>	Integer array Real number array	1D array stored each point of the X-coordinate
<y()>	Integer array Real number array	1D array stored each point of the Y-coordinate
<centerX>	Real number	Central X coordinate of the approximate circle
<centerY>	Real number	Central Y coordinate of the approximate circle
<radius>	Real number	Radius of the approximate circle

Return value:

None.

Description:

Gets the approximate circle from the specified number of points, whose coordinates are specified in the <x()> and <y()> parameters, in the <count> parameter. In the <centerX> and <centerY> parameters, respectively specify the variables that will hold the center X coordinate and center Y coordinate of the approximate circle gotten. In the <radius> parameter, specify the variable that will hold the radius of the circle gotten.

In the <count> parameter, specify an integer value of at least 3.

In the <x()> parameter and <y()> parameter, respectively specify a 1D integer array variable or real number array variable, where a number of coordinate values greater than or equal to the number specified in the <count> parameter are stored, without adding element numbers but adding () to the variables.

If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.

If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.

If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:

None.

Example:

Calculates the approximate circle and getting the center coordinates and radius from each of the three points (50, 50), (100, 100), and (150, 50).

```
Dim X&(3), Y&(3)

Rem Assign the three values of the coordinate value to the array.
X&(0) = 50
Y&(0) = 50
X&(1) = 100
Y&(1) = 100
X&(2) = 150
Y&(2) = 50

Rem Calculate the approximate circle and get the center coordinates and the radius
.
ApproximationCircle 3, X&(), Y&(), CENTERX#, CENTERY#, RADIUS#
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

GetUnitData on page 8-153
UnitData on page 8-341
Lsqumeth on page 8-192

ArrayDims

Gets the number of dimensions in the array.

Format:
ArrayDims(<arrayName()>)

Parameter:

Parameter name	Data type	Description
<arrayName()>	-	Variable name for the specified array

Return value:
The number of dimensions in the array.

Description:
Gets the number of dimensions of the array specified by the <arrayName()> parameter.
If an array name that does not exist is specified as the <arrayName()> parameter, an Undefined array error will occur.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
This macro function cannot be used for scene variables and system variables.

Example:
Gets the number of dimensions in the array.

```
Dim AA&(1,1,1)

DIMS& = ArrayDims(AA&())
```

The result is shown below.

```
DIMS& = 3
```

Usable Modules:
Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 6.40 or later

Related Items:

ArrayLen on page 8-33

ArrayLen

Gets the number of elements in the array.

Format:
ArrayLen(<arrayName()>[, <dimension>])

Parameter:

Parameter name	Data type	Description
<arrayName()>	-	Variable name for the specified array
<dimension>	Integer	Dimension number of the array to get the number of elements

Return value:
The number of elements in the array.

Description:
Gets the array element number of the dimension number specified by the <dimension> parameter of the array specified by the <arrayName()> parameter.
If the <dimension> parameter is omitted, the number of elements in all dimensions will be obtained.
If an array name that does not exist is specified as the <arrayName()> parameter, an Undefined array error will occur.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
This macro function cannot be used for scene variables and system variables.

Example:
Gets the number of elements in the array.

```
Dim AA&(1,2)

LEN0& = ArrayLen(AA&(),0)
LEN1& = ArrayLen(AA&(),1)
```

The result is shown below.

```
LEN0& = 2
LEN1& = 3
```

Usable Modules:
Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 6.40 or later

Related Items:

ArrayDims on page 8-32

Asc

Gets the character code of thespecified character.

Format:
Asc <string>

Parameter:

Parameter name	Data type	Description
<string>	Character string	Character string that requests the character code.

Return value:
Returns the integer type character code value in decimal.

Description:
Gets the character code of the first character in a character string specified in the <string> parameter in ASCII code.
Asc is the inverse function of Chr$. Chr$ returns the character corresponds to the specified character code.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
Even if a character string longer than 255 characters is specified for a character string parameter, an error will not occur.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Gets the character code of letter "A".

```
CHARA$ = "A"

CODE& = Asc(CHARA$)
```

The result is shown below.

```
CODE& = 65
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

Chr$ on page 8-44
Hex$ on page 8-158
LCase$ on page 8-179
Left$ on page 8-180
Len on page 8-181
Mid$ on page 8-205
Piece$ on page 8-219
Right$ on page 8-243
Str$ on page 8-314
Str2$ on page 8-315
UCase$ on page 8-339
Val on page 8-352

AssignUnit

Registers the processing unit.

Format:
AssignUnit <unitNo>, <itemIdent>

Parameter:

Parameter name	Data type	Description
<unitNo>	Integer	Processing unit number on the measurement flow to register the processing unit (0 to the enrollment number of the processing units at the current scene)
<itemIdent>	Character string	Identification name of the processing item to register as a processing unit*9-5 List for Processing Item Identifier* on page 9-67

Return value:
None.

Description:
Registers the processing item with the identification name specified in the <itemIdent> parameter as a processing unit in the position in the measurement flow specified in the <unitNo> parameter.
If a processing unit is already registered in the position specified in the <unitNo> parameter, that processing unit is overwritten.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If an identification name that does not exist is specified as the <itemIdent> parameter, an "Illegal function call" error will occur.

If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
Execute this macro function when the BUSY signal or other measurement in progress signal is ON and measurement is prohibited. (For details, refer to *5-1-4 State Transitions and Execution Timing* on page 5-13.)

Example:
Registers the search processing unit at the end of the measurement flow.

```
Rem Get the number of processing units registered in the current measurement flow.
UNUM& = UnitCount

Rem Add the search processing unit.
AssignUnit UNUM&, "Search"
```

Usable Modules:
Scene Control Macro / Communication Command Macro

Supported Versions:
Version 3.50 or later

Related Items:

CheckUnit on page 8-43
CopyUnit on page 8-58
DeleteUnit on page 8-72
InsertUnit on page 8-168
MeasureStart on page 8-200
MeasureStop on page 8-201
MoveUnit on page 8-209
UnitCount on page 8-340

Atn

Gets the arctangent of thespecified expression.

Format:
Atn(<expression>)

Parameter:

Parameter name	Data type	Description
<expression>	Integer Real number	Expression that gets the arc tangent

Return value:
Returns the arctangent as a double precision real value in the range -π/2 to π/2 radians.

Description:
Gets the arctangent of the expression specified in the <expression> parameter.
To convert the gotten value to an angle, multiply by π/180.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Gets the arctangent of the variable X#.

```
X# = 1

XX# = Atn(X#)*180/3.141592
```

The result is shown below.
(The returned value is rounded off to the nearest thousandth.)

```
XX# = 45.000
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

Cos on page 8-64
Sin on page 8-309
UnitData on page 8-341
GetUnitData on page 8-153
Tan on page 8-323

BusyOut

Sets the output state of theprocessing busy signal.

Format:
BusyOut <ioIdent>, <state>

Parameter:

Parameter name	Data type	Description
<ioIdent>	Character string	Identification name of the communication module to be used ("ParallelIo" or "EtherCAT") (Refer to *9-4 List of I/O Modules* on page 9-27.)
<state>	Integer	Output state of terminal 0: Output OFF 1: Output ON

Return value:
None.

Description:
Set the output state specified in the <state> parameter in the processing busy signal, such as the BUSY signal, of the communication module specified in the <ioIdent> parameter.
Normally "ParallelIo" or "EtherCAT" should be specified in the <ioIdent> parameter.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If an identification name that does not exist is specified in the <ioIdent> parameter, an "Illegal function call" error will occur.

Even if an output status parameter value that does not exist (i.e., other than 0 and 1) is specified in the <state> parameter, an error will not occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
In the communication command macro, sets the BUSY signal of parallel I/O to ON.

```
IOMODULE$ = "ParallelIo"

Rem Set the output state.
BusyOut IOMODULE$, 1
```

Usable Modules:
Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

GetAll on page 8-134
GetPort on page 8-144
JudgeOut on page 8-175
PutAll on page 8-223
PutPort on page 8-225
RunOut on page 8-245

Call (Structured Subroutine Call)

Executes the specified subroutine with arguments added.

Format:
Call <label>([, <argument0>...<argumentN>])

Parameter:

Parameter name	Data type	Description
<argument0>	Integer Real number Character string Array Variant type	Data to be passed to the subroutine
:	:	:
<argumentN>	Integer Real number Character string Array Variant type	Data to be passed to the subroutine

Return value:
None.

Description:

Executes the subroutine with arguments whose label name is specified by the <label> parameter using the <argument0> to <argumentN> parameters as the arguments.
The program returns to the calling subroutine when it reaches the End Sub statement at the end of the subroutine and resumes the processing from the next line where this macro function was executed.
If a non-existent label is specified as the parameter, an Undefined label error will occur.
No error will occur even if the number of arguments defined as the subroutine with arguments and the number of arguments specified using the <argument0> to <argumentN> parameters do not match.
If the number of arguments specified in the Call function is greater than the number of arguments defined for the subroutine with arguments, the extra arguments will be ignored. If the number of arguments specified in this function is less than the number of arguments defined in the subroutine with arguments, the initial value for each type of the variable will be passed to the arguments that were not specified.
() cannot be omitted even if the number of arguments is 0.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Executes the *ValueInit subroutine defined separately in the *MeasureProc subroutine of the unit macro.

```
*MCRINIT
  Dim X#(0)
  Dim Y#(0)
  Dim ANG#(0)
  Dim JG&
Return
*MEASUREPROC
  Rem Execute another subroutine.
  Call *ValueInit(X#(), Y#(), ANG#(), JG&)
Return
Rem Initialize various types of measured values.
Sub *ValueInit(%POSX#(), %POSY#(), %ANGLE#(), %JUDGE&)
  ReDim %POSX#(10)
  ReDim %POSY#(10)
  ReDim %ANGLE#(10)
  %JUDGE& = JUDGE_NC
End Sub
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 6.50 or later

Related Items:

Sub - End Sub on page 8-318

Call (User-defined Function Execution)

Executes the registered userdefinedfunction.

Format:
Call <subroutineIdent>[; | , <argument>...]

Parameter:

Parameter name	Data type	Description
<subroutineIdent>	Character string	Identification name of the user-defined function that has been registered
<argument>	Integer Real number Character string	Argument of the user-defined function that has been registered

Return value:
None.

Description:
Calls a user-defined function with the specified identification name by the <subroutineIdent> parameter and executes the function. If the user-defined function has arguments, the specified data by the <argument> parameter is passed to the user-defined function.
In the <argument> parameter, specify an argument according to the user-defined function definition.
Register a user-defined function with the SetUserSubroutine function before executing this function.
How user-defined functions work depends on the defined processes in external DLL files.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a user-defined function that is not registered is specified for a parameter, an "Illegal function call" error will occur.
An error of "String too long" will be occurred when you specify the character strings as character strings type exceeds 255 characters.
In case an error is returned as a result of the user-defined function execution, this Call function will return one of the following errors.
- Syntax error
- Illegal function call
- Out of memory
- Type missmatch

If this macro function calls a user-defined function that has not been programmed with a supported interface, an error occurs during the user-defined function processing.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
- Only the user-defined functions that have been defined in programmed DLL files by the supported interfaces are accepted to this macro function. If other DLL files or user-defined functions than above is used, unexpected operation may occur such as a measurement error or the sensor controller abnormal termination. For user-defined functions creation, refer to the FH-AP1.
- Depending on the called user-defined function processings by this macro function, unexpected operation may occur such as a measurement error or the sensor controller abnormal termination. Be

sure to fully check the operations and debug with external devices disconnected from the sensor controller in advance.
- To operate this macro function in an actual environment with external devices connected to, always apply external fail safe measures to the system.

Example:
With identification name "USR", registers a user-defined function "UserProc0" that has been defined in MacroUserProc.dll. Then, specifies the identification name to call the user-defined function and executes it.

```
Rem Register the user-defined function so that the function can be used in this pr
ogram
SetUserSubroutine "USR", "MacroUserProc", "UserProc0"

Rem Call the registered user-defined function and execute it
Call "USR", 0
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 5.20 or later

Related Items:

SetUserSubroutine on page 8-305

ChangeScene

Change the scene.

Format:
ChangeScene <sceneNo>

Parameter:

Parameter name	Data type	Description
<sceneNo>	Integer	Scene number to change (0 to 127)

Return value:
None.

Description:
Changes the current scene to the scene with the scene number specified in the <sceneNo> parameter.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:

Execute this macro function when the BUSY signal or other measurement in progress signal is ON and measurement is prohibited. (For details, refer to *5-1-4 State Transitions and Execution Timing* on page 5-13.)

Example:
Changes the current scene to scene number 2.

```
ChangeScene 2
```

Usable Modules:
Communication Command Macro

Supported Versions:
Version 3.50 or later

Related Items:

ChangeSceneGroup on page 8-42
MeasureStart on page 8-200
MeasureStop on page 8-201
SceneCount on page 8-257
SceneGroupNo on page 8-260
SceneNo on page 8-263

ChangeSceneGroup

Changes the scene group.

Format:
ChangeSceneGroup <sceneGroupNo>, <sceneNo>

Parameter:

Parameter name	Data type	Description
<sceneGroupNo>	Integer	New scene group (0 to 31)
<sceneNo>	Integer	Scene number to change (0 to 127)

Return value:
None.

Description:
Changes the current scene group to the scene specified in the <sceneNo> parameter, which belongs to the scene group specified in the <sceneGroupNo> parameter.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
Execute this macro function when the BUSY signal or other measurement in progress signal is ON and measurement is prohibited. (For details, refer to *5-1-4 State Transitions and Execution Timing* on page 5-13.)

Example:
Changes the current scene group to scene 2 of scene group 10.

```
ChangeSceneGroup 10, 2
```

Usable Modules:
Communication Command Macro

Supported Versions:
Version 3.50 or later

Related Items:

ChangeScene on page 8-41
MeasureStop on page 8-201
SceneGroupNo on page 8-260
MeasureStart on page 8-200
SceneGroupCount on page 8-259

CheckUnit

Checks the registration status of a processing unit.

Format:
CheckUnit(<unitNo>)

Parameter:

Parameter name	Data type	Description
<unitNo>	Integer	Processing unit number (0 to (number of processing units of current scene minus one)) of processing unit whose registration status is to be checked.

Return value:
Returns the registration status as an integer.
- 0: Processing unit not registered
- 1: Processing unit already registered

Description:
Checks if the processing unit in the position in the measurement flow specified in the <unitNo> parameter has been registered.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
Type mismatch error is occurred when wrong data is specified as parameter. Illegal function call error is not occurred even if non-exist number, values, combination of data or values.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Gets the processing unit title if a processing unit has been registered in processing unit number 3 in the measurement flow.

```
Rem Check the registration status of unit number 3.
If CheckUnit(3) = 1 Then
   Rem Get the title of the processing unit.
   TITLE$ = UnitTitle$(3)
Endif
```

Usable Modules:
Scene Control Macro / Communication Command Macro

Supported Versions:
Version 3.50 or later

Related Items:

AssignUnit on page 8-35
CopyUnit on page 8-58
DeleteUnit on page 8-72
InsertUnit on page 8-168
MeasureStart on page 8-200
MeasureStop on page 8-201
MoveUnit on page 8-209
UnitCount on page 8-340

Chr$

Determining the character of thespecified character code.

Format:
Chr$(<expression>)

Parameter:

Parameter name	Data type	Description
<expression>	Integer	Expression to get the character (0 to 255)

Return value:
Returns the character string type character.

Description:
Gets the character of the ASCII character code specified in the <expression> parameter.
ASCII control codes can also be specified in the <expression> parameter.
Chr$ is the inverse function of Asc. Asc returns the character code in decimal corresponds to the specified character.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If a value outside the range -2147483648 to 2147483647 is specified as an integer parameter, an "Overflow" error will occur.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:

Gets the characters for ASCII codes "48" and "13".

```
CHARA1$ = Chr$(48)
CHARA2$ = Chr$(13)
```

The result is shown below.

```
CHARA1$ = "0"
CHARA2$ = "CR"
```

Usable Modules:

Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:

Version 3.50 or later

Related Items:

Asc on page 8-34
Hex$ on page 8-158
LCase$ on page 8-179
Left$ on page 8-180
Len on page 8-181
Mid$ on page 8-205
Piece$ on page 8-219
Right$ on page 8-243
Str$ on page 8-314
Str2$ on page 8-315
UCase$ on page 8-339
Val on page 8-352

ClearMeasureData

Clears the measurement resultsof the processing unit.

Format:
ClearMeasureData [<unitNo>]

Parameter:

Parameter name	Data type	Description
<unitNo>	Integer	Processing unit number (0 to (the number of registered processing units in the current scene minus one))

Return value:
None.

Description:
Clears the measurement results of the processing unit specified in the <unitNo> parameter.
If the Scene Control Macro or the Communication Command Macro is used and the <unitNo> parameter is omitted or -1 is specified, all the measurement results of the processing units will be cleared.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
- Execute this macro function when the BUSY signal or other measurement in progress signal is ON and measurement is prohibited. (For details, refer to *5-1-4 State Transitions and Execution Timing* on page 5-13.)
- <UnitNo> cannot be omitted in the unit macro.

Example:
In the communication command macro, clears the measurement results of the search processing unit (Processing Unit number 2).

```
Rem Clear the measurement results.
ClearMeasureData 2
```

Usable Modules:
Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

UnitNo on page 8-349

ClearScene

Clears the scene data.

Format:
ClearScene <sceneNo>

Parameter:

Parameter name	Data type	Description
<sceneNo>	Integer	Scene number of scene to be cleared (0 to 127)

Return value:
None.

Description:
Clears the setting information in the scene with the scene number specified in the <sceneNo> parameter, and restores the scene to the factory default state.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
Type mismatch error is occurred when wrong data is specified as parameter. Illegal function call error is not occurred even if non-exist number, values, combination of data or values.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
Execute this macro function when the BUSY signal or other measurement in progress signal is ON and measurement is prohibited. (For details, refer to *5-1-4 State Transitions and Execution Timing* on page 5-13.)

Example:
Clears the scene data of scene number 2.

```
ClearScene 2
```

Usable Modules:
Communication Command Macro

Supported Versions:
Version 3.50 or later

Related Items:

ChangeSceneGroup on page 8-42
MeasureStop on page 8-201
SceneGroupNo on page 8-260
MeasureStart on page 8-200
SceneCount on page 8-257
SceneNo on page 8-263

ClearSceneGroup

Clears scene group data Scene groupcontrol.

Format:
ClearSceneGroup <sceneGroupNo>

Parameter:

Parameter name	Data type	Description
<sceneGroupNo>	Integer	Scene group whose scene group data is to be cleared 0 to 31: scene group number 0 to 31 -1: current scene group number

Return value:
None.

Description:
Clears the scene group settings and saved data of the scene group number specified by <sceneGroupNo> parameter.
If -1 is specified for the <sceneGroupNo> parameter, the settings in the current scene group will be cleared.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
Execute this macro function when the BUSY signal or other measurement in progress signal is ON and measurement is prohibited. (For details, refer to *5-1-4 State Transitions and Execution Timing* on page 5-13.)

Example:
Clears the scene group data of scene group number 1.

```
ClearSceneGroup 1
```

Usable Modules:
Communication Command Macro

Supported Versions:
Version 3.50 or later

Related Items:

ClearScene on page 8-47
MeasureStop on page 8-201
SceneGroupNo on page 8-260
MeasureStart on page 8-200
SceneGroupCount on page 8-259

Close

Closes up the file.

Format:
Close [#<fileNo>[, #<fileNo>]...]

Parameter:

Parameter name	Data type	Description
<fileNo>	Integer	File number (0 to 15) of closed file

Return value:
None.

Description:
Close the file number specified in the <fileNo> parameter.
In the <fileNo> parameter, specify the specified file number in the Open function that has been used to open the file.
If multiple file numbers have been specified in the <fileNo> parameter, the multiple open files are closed.
If the <fileNo> parameter is omitted, all open files are closed.
If a value outside the range of 0 to 15 is specified in the <fileNo> parameter, an "Illegal function call" error will occur.
Be sure to use this macro function to close the opened file with the Open function within the same subroutine as where the Open function is used. File accessing processes such as data writing to a file and data reading from a file may not be completed properly in the following cases.
• This macro function is not executed.
• This macro function is used in a different subroutine from where the Open function is executed.
• This macro function is executed at a different timing from the Open function execution.
To access the files that have been closed by executing this macro function, execute the Open function again to open the closed file.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Opens the file, writes the data in the file, and then closes the file.

```
DATA& = 10

Rem Open the file.
Open "E:\input.dat" For Output As #1

Rem Write the data in the opened file
Print #1 DATA&

Rem Close the opened file
Close #1
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

Open For Append As# on page 8-211
Open For Input As# on page 8-212
Open For Output As# on page 8-213
Print on page 8-221

CloseTextData

Close up a messages file.

Format:
CloseTextData [#<textDataNo>[, #<textDataNo>]...]

Parameter:

Parameter name	Data type	Description
<textDataNo>	Integer	Text data number (0 to 15) of the closed message file

Return value:
None.

Description:
Close the messages file in the text data number specified in the <textDataNo> parameter.
In the <textDataNo> parameter, specify the speficied text data number in the OpenTextData function that has been used to open the message file.
If multiple text data numbers have been specified in the <textDataNo> parameter, the multiple open message files are closed.
Close all the open message file if the <textDataNo> parameter is omitted.
If a value outside the range from 0 to 15 is specified in the <textDataNo> parameter, an "Illegal function call" error will occur.
Be sure to use this macro function to close the opened file with the OpenTextData function within the same subroutine as where the OpenTextData function is used. The message file cannot properly be closed and this macro function may not properly be executed in the subsequent processes in the following cases.
• This macro function is not executed.
• This macro function is used in a different subroutine from where the OpenTextData function is executed.
• This macro function is executed at a different timing from the OpenTextData function execution.
To access the messages file that has been closed with this macro function, execute the OpenTextData function again to open the messages file.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:

Uses the *MEASUREDISPT subroutine of the Unit Macro processing unit to display the measured correlation value by the search processing unit (Processing Unit number 5), along with the gotten text string from the prepared message file for the processing unit, in the text window. The correlation value can be gotten with External Reference Data number 5.

```
*MEASUREDISPT

   Rem Get the measurement result.
   GetUnitData 5, 5, CR#

   Rem Open the messages file
   OpenTextData "Search" As #1

   Rem Get the text
   TEXT$ = GetText$(#1, "Correlation")

   Rem Draw the gotten text string from the messages file without adding any line
break on the text window.
   DrawText TEXT$, UnitJudge(5), 0

   Rem Draw the measurement results on the text window.
   DrawText Str2$(CR#, 4, 4, 0, 0), UnitJudge(5), 1

   Rem Close up the messages file.
   CloseTextData

Return
```

The result is shown below.

```
Correlation value: 90.0000
```

Usable Modules:

Unit Calculation Macro / Unit Macro

Supported Versions:

Version 5.00 or later

Related Items:

DrawText on page 8-109
GetUnitData on page 8-153
UnitJudge on page 8-348
GetText$ on page 8-150
OpenTextData on page 8-215

Cls

Clears the display of the console window.

Format:
Cls

Parameter:
None.

Return value:
None.

Description:
Clears the display of the System status monitoring console window.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Clears the display of the System status monitoring console window.

```
Cls
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 6.50 or later

Related Items:

Cont on page 8-53
Debug on page 8-68
DebugPrint on page 8-69
List on page 8-184
Print on page 8-221
SetStop on page 8-293
SubList on page 8-320
VarList on page 8-353

Cont

Resumes execution of the program after it has beenstopped.

Format:
Cont [<mode>]

Parameter:

Parameter name	Data type	Description
<mode>	Integer	Resuming method of the stopped program This parameter can be omitted. Omitting the parameter resumes the program execution. After the resume, the program runs to the end unless an error occurs. 0: Execute the program by step-in execution. If the current program line calls a subroutine, the subroutine is entered and is executed in steps. Otherwise, the current statement is executed and the program is stopped at the next line. 1: Step-over execution If the current program line calls a subroutine, the entire subroutine is executed and the program stops at the next line after the subroutine call. Otherwise, the current statement is executed and the program is stopped at the next line. 2: Execute the program by step-out execution. If the current program line is a subroutine that was called from a subroutine, the entire subroutine after the current program line is executed, and the program stops at the next line of the subroutine that called the subroutine. Otherwise, the program is executed until it ends or an error occurs.

Return value:
None.

Description:
With a use of the resuming method specified in the <mode> parameter, resumes the program execution from the statement line where the program has been stopped by the Stop function execution. All the statuses before the Stop function execution is handed to the operation after resuming. (For details, refer to *6-1 How to Use the Debug Function* on page 6-2.)
By specifying the <mode> parameter, the program can be executed in steps of one line at a time after the Stop function is executed.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
Type mismatch error is occurred when wrong data is specified as parameter. Illegal function call error is not occurred even if non-exist number, values, combination of data or values.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
After the execution of the Stop function in line 220 of the Unit Macro processing unit (Processing number 1), executes the next single line (line 230).

```
Macro(U1) 220 Stop
Macro(U1) Stop in 220
Macro(U1) 230 POS.X#=(POS0.X@ + POS1.X@) / 2
Macro(U1)>Cont 1
Macro(U1)>
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 5.20 or later

Related Items:

Cls on page 8-52
Debug on page 8-68
DebugPrint on page 8-69
List on page 8-184
Print on page 8-221
SetStop on page 8-293
SetVar on page 8-307
Stop on page 8-312
SubList on page 8-320
VarList on page 8-353

ContinuousMeasure

Do the start or stop of thecontinuous measurement.

Format:
ContinuousMeasure <state>

Parameter:

Parameter name	Data type	Description
<state>	Integer	Continuous measurement is in the running status. False: Stop the Continuous measurement. True: Start the Continuous measurement.

Return value:
None.

Description:
Specifies the Start/Stop of the Continuous measurement using <state> parameter.
Start the Continuous measurement when you set True. Stop the Continuous measurement when you set False.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.

Usage Cautions:
Executes this function in the status when measure signal is OFF.(For details, refer to *5-1-4 State Transitions and Execution Timing* on page 5-13.)

Example:

```
Rem Start or Stop the Continuous measurement.
NO& = int(Argumentvalue#(0))

Rem Start the Continuous measurement if the received value of communication comman
d macro is 1.
If NO& = 1 then
   ContinuousMeasure true
   Rem Stop the Continuous measurement if the received value of communication comm
and macro is except 1.
Else
   ContinuousMeasure false
EndIf
```

Usable Modules:
Scene Control Macro / Communication Command Macro

Supported Versions:
Version 5.40 or later

Related Items:

If Then Else on page 8-159
MeasureStart on page 8-200

CopyMeasureImage

Copies the measurement image as an image of the Unit Macroprocessing unit.

Format:
CopyMeasureImage <measureImageNo>, <myImageNo>

Parameter:

Parameter name	Data type	Description
<measureImageNo>	Integer	Measurement image number (always 0)
<myImageNo>	Integer	Image number of copy destination (0 to 31)

Return value:
None.

Description:
Copies the image with the measurement image number specified in the <measureImageNo> parameter to the image buffer of the image number specified in the <myImageNo> parameter of the Unit Macro processing unit that calls this macro function.
Normally 0 should be specified in the <measureImageNo> parameter.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If a value outside the range -2147483648 to 2147483647 is specified as an integer parameter, an "Overflow" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:

None.

Example:
Stores the measurement image of each measurement in order in the image buffer.

```
Rem Calculate the image buffer number.
MYIMAGENO& = MYIMAGENO& + 1

Rem Overwrite the 32nd and following images in order from the oldest image.
If MYIMAGENO& > 31 Then
   MYIMAGENO& = 0
Endif

Rem Store the measurement image in the image buffer of the Unit Macro processing u
nit.
CopyMeasureImage 0, MYIMAGENO&
```

Usable Modules:
Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

CopyUnitImage on page 8-61
SetMeasureImage on page 8-279

CopyScene

Copies scene data.

Format:
CopyScene <srcSceneNo>, <destSceneNo>

Parameter:

Parameter name	Data type	Description
<srcSceneNo>	Integer	Scene number (0 to 127) of the scene to be copied.
<destSceneNo>	Integer	Scene number (0 to 127) of destination scene.

Return value:
None.

Description:
Copies the scene data of the scene number specified in the <srcSceneNo> parameter to the scene data of the scene number specified in the <destSceneNo> parameter.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If the same scene number is specified in the <destSceneNo> parameter as the <srcSceneNo> parameter, an "Illegal function call" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If there is insufficient free working memory to copy the data, an "Illegal function call" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
Execute this macro function when the BUSY signal or other measurement in progress signal is ON and measurement is prohibited. (For details, refer to *5-1-4 State Transitions and Execution Timing* on page 5-13.)

Example:
Copies the data of scene 2 to scene 3.

```
CopyScene 2, 3
```

Usable Modules:
Communication Command Macro

Supported Versions:
Version 3.50 or later

Related Items:

CopySceneGroup on page 8-57
MeasureStop on page 8-201
SceneGroupNo on page 8-260
MeasureStart on page 8-200
SceneCount on page 8-257
SceneNo on page 8-263

CopySceneGroup

Copies scene group data.

Format:
CopySceneGroup <srcSceneGroupNo>, <destSceneGroupNo>

Parameter:

Parameter name	Data type	Description
<srcSceneGroup-No>	Integer	Scene group number (0 to 31) of the scene group to be copied.
<destSceneGroup-No>	Integer	Scene group number (0 to 31) of the destination scene group.

Return value:
None.

Description:
Copies the scene group data of the scene group number specified in the <srcSceneGroupNo> parameter to the scene group data of the scene group specified in the <destSceneGroupNo> parameter.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If the same scene number is specified in the <destSceneNo> parameter as the <srcSceneNo> parameter, an "Illegal function call" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If there is insufficient free working memory to copy the data, an "Illegal function call" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:

Execute this macro function when the BUSY signal or other measurement in progress signal is ON and measurement is prohibited. (For details, refer to *5-1-4 State Transitions and Execution Timing* on page 5-13.)

Example:
Copies the data of scene group 0 to scene group 1.

```
CopySceneGroup 0, 1
```

Usable Modules:
Communication Command Macro

Supported Versions:
Version 3.50 or later

Related Items:

CopyScene on page 8-56
MeasureStart on page 8-200
MeasureStop on page 8-201
SceneCount on page 8-257
SceneGroupNo on page 8-260
SceneNo on page 8-263

CopyUnit

Copies a processing unit.

Format:
CopyUnit [<srcSceneNo>,] <srcUnitNo>, <destUnitNo>, <mode>

Parameter:

Parameter name	Data type	Description
<srcSceneNo>	Integer	Scene number of the origin of copy (0 to 127)
<srcUnitNo>	Integer	Processing unit number of copy source (0 to (number of scene processing units of copy source minus one))
<destUnitNo>	Integer	Processing unit number that is to copy to (0 to the enrollment number of the processing units at the current scene -1)
<mode>	Integer	Processing mode 0: Overwrites the processing unit of the processing unit number of the copy destination 1: Insert the loaded processing unit in front of the processing unit number.

Return value:
None.

Description:
Copies the processing unit specified in the <srcUnitNo> parameter of the scene number specified in the <srcSceneNo> parameter to the measurement flow position specified in the <destUnitNo> parameter of the current scene, using the mode specified in the <mode> parameter.
If the <srcSceneNo> parameter is omitted, the copy source scene is the current scene.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.

An "Illegal function call" error will also occur if you specify Camera Image Input (processing unit 0) of an unselected scene in the Copy source processing unit.
Unprocessed scenes do not contain processing units, and Camera Image Input is input as processing unit 0 after scene switching. Please switch scenes before executing the CopyUnit command.
When 0 is specified in the <mode> parameter, the processing unit that has the processing unit number of the copy destination is overwritten by the copy source processing unit. When 1 is specified in the <mode> parameter, the processing unit of the copy source is inserted in the processing unit number of the copy destination, and the processing unit number of the processing unit of the copy destination is moved down by 1.
When the <srcSceneNo> parameter is omitted and 0 is specified in the <mode> parameter, specify different values in the <srcUnitNo> parameter and <destUnitNo> parameter. If the same value is specified, an "Illegal function call" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
Execute this macro function when the BUSY signal or other measurement in progress signal is ON and measurement is prohibited. (For details, refer to *5-1-4 State Transitions and Execution Timing* on page 5-13.)

Example:
Copies processing unit 3 of scene 2 and inserts in front of processing unit 4 of the current scene.

```
CopyUnit 2, 3, 4, 1
```

Usable Modules:
Scene Control Macro / Communication Command Macro

Supported Versions:
Version 3.50 or later

Related Items:

AssignUnit on page 8-35
DeleteUnit on page 8-72
MeasureStart on page 8-200
MoveUnit on page 8-209
CheckUnit on page 8-43
InsertUnit on page 8-168
MeasureStop on page 8-201
UnitCount on page 8-340

CopyUnitFigure

Copies figure data to theprocessing unit.

Format:
CopyUnitFigure <srcSceneNo>, <srcUnitNo>, <srcFigureNo>, <destUnitNo>, <destFigureNo>

Parameter:

Parameter name	Data type	Description
<srcSceneNo>	Integer	Scene number of the origin of copy (0 to 127)
<srcUnitNo>	Integer	Unit number of copy source (0 to (number of scene processing units of copy source minus one))

Parameter name	Data type	Description
<srcFigureNo>	Integer	Figure number of copy source (Refer to *9-7 List of Figure Numbers* on page 9-73.)
<destUnitNo>	Integer	Unit number of copy destination (0 to (number of scene processing units of copy destination minus one))
<destFigureNo>	Integer	Figure number of copy destination (Refer to *9-7 List of Figure Numbers* on page 9-73)

Return value:
None.

Description:
For the scene specified in the <srcSceneNo> parameter, copies the figure data of the figure number specified in the <srcFigureNo> parameter, of the processing unit specified in the <srcUnitNo> parameter, to the figure data of the figure number specified in the <destFigureNo> parameter, of the processing unit specified in the <destUnitNo> parameter.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If a value outside the range -2147483648 to 2147483647 is specified as an integer parameter, an "Overflow" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
- Execute this macro function when the BUSY signal or other measurement in progress signal is ON and measurement is prohibited. (For details, refer to *5-1-4 State Transitions and Execution Timing* on page 5-13.)
- Use this macro function with the measurement image displayed after one or more measurements, or after the image file is specified and re-measured.
- Set the figure data so that pixels from outside the image are not included in the figure.

Example:
Copies the measurement region figure of the Shape Search III processing unit of Processing Unit number 2 in the measurement flow of scene 0 to the measurement region of the Shape Search III processing unit of Processing Unit number 5. The measurement region figure of the Shape Search III processing item is figure 1.

```
Rem Copy the figure of the measurement region
CopyUnitFigure 0, 2, 1, 5, 1
```

Usable Modules:
Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

GetUnitFigure on page 8-154
MeasureStop on page 8-201
SetUnitFigure on page 8-301
Ut on page 8-351
MeasureStart on page 8-200
SceneNo on page 8-263
UnitNo on page 8-349

CopyUnitImage

Copies a processing unit image as a unit macro processing unitimage.

Format:
CopyUnitImage <unitNo>, <imageNo>, <myImageNo>

Parameter:

Parameter name	Data type	Description
<UnitNo>	Integer	Processing unit number that is to be copied (0 to (the number of registered processing units in the current scene -1))
<imageNo>	Integer	Image number of copy source (Refer to *9-9 Image Number List* on page 9-78.)
<myImageNo>	Integer	Image number of copy destination (0 to 31)

Return value:
None.

Description:
Copies the image with the image number specified in the <ImageNo> parameter, of the processing unit specified in the <unitNo> parameter, to the image buffer of the unit macro processing unit that calls this macro function with the image number specified in the <myImageNo> parameter.
Normally 0 should be specified in the <imageNo> parameter.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If a value outside the range -2147483648 to 2147483647 is specified as an integer parameter, an "Overflow" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
- Execute this macro function when the BUSY signal or other measurement in progress signal is ON and measurement is prohibited. (For details, refer to *5-1-4 State Transitions and Execution Timing* on page 5-13.)

Example:
Stores the image of each measurement in order in the image buffer after filtering by the color gray filter processing unit of processing unit number 1.

```
Rem Calculate the image buffer number.
MYIMAGENO& = MYIMAGENO& + 1

Rem Overwrite the 32nd and following images in order from the oldest image.
If MYIMAGENO& > 31 Then
   MYIMAGENO& = 0
Endif

Rem Store the filtered image in the image buffer of the "Unit Macro" processing un
it.
CopyUnitImage 1, 0, MYIMAGENO&
```

Usable Modules:
Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

CopyMeasureImage on page 8-55
SetMeasureImage on page 8-279
UnitNo on page 8-349
Ut on page 8-351

CopyUnitModel

Copies the model data of aprocessing unit.

Format:
CopyUnitModel <srcSceneNo>, <srcUnitNo>, <srcModelNo>, <destUnitNo>, <destModelNo>

Parameter:

Parameter name	Data type	Description
<srcSceneNo>	Integer	Scene number of the origin of copy (0 to 127)
<srcUnitNo>	Integer	Unit number of copy source (0 to (number of scene processing units of copy source minus one))
<srcModelNo>	Integer	Model number of copy source (Refer to *9-8 Model Number List* on page 9-76.)
<destUnitNo>	Integer	Unit number of copy destination (0 to (number of scene processing units of copy destination minus one))
<destModelNo>	Integer	Model number of copy destination (Refer to *9-8 Model Number List* on page 9-76.)

Return value:
None.

Description:
Copies the model data of the model number specified in the <srcModelNo> parameter, of the processing unit specified in the <srcUnitNo> parameter, of the scene specified in the <srcSceneNo> parameter, to the model data of the model number specified in the <destModelNo> parameter, of the processing unit specified in the <destUnitNo> parameter.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If a value outside the range -2147483648 to 2147483647 is specified as an integer parameter, an "Overflow" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.
On models that use a processing item for measurement, there is also the model data, model figure, model parameter, and other information. The model information varies by processing item, and correct operation does not always result from simply copying the model data, but it is also possible to use a variant-type variable in the scene control macro to copy model data. For details, refer to Application Producer (FH-AP1, sold separately).

Usage Cautions:

- Execute this macro function when the BUSY signal or other measurement in progress signal is ON and measurement is prohibited. (For details, refer to *5-1-4 State Transitions and Execution Timing* on page 5-13.)
- Use this macro function with the measurement image displayed after one or more measurements, or after the image file is specified and re-measured.
- Set the figure data so that pixels from outside the image are not included in the figure.

Example:

Copies the Shape Search III processing unit model of Processing Unit number 2 to the Shape Search III processing unit of Processing Unit number 3.

```
Rem Copy the model figure of Shape Search III.
CopyUnitFigure 0, 2, 0, 3, 0

Rem Copy the detection point setting of Shape Search III
GetUnitData 2, "detectionPosX", PosX#
GetUnitData 2, "detectionPosY", PosY#
SetUnitData 3, "detectionPosX", PosX#
SetUnitData 3, "detectionPosY", PosY#

Rem Copy the model data of Shape Search III.
CopyUnitModel 0, 2, 0, 3, 0
```

Usable Modules:

Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:

Version 3.50 or later

Related Items:

CopyUnitFigure on page 8-59
MeasureStart on page 8-200
SceneNo on page 8-263
UnitNo on page 8-349
GetUnitData on page 8-153
MeasureStop on page 8-201
SetUnitData on page 8-300
Ut on page 8-351

Cos

Gets the cosine of the specifiedexpression.

Format:
Cos(<expression>)

Parameter:

Parameter name	Data type	Description
<expression>	Integer Real number	Expression to calculate the cosine

Return value:
Returns the cosine as a double precision real value in the range -1 to 1.

Description:
Gets the cosine of the expression specified in the <expression> parameter.
To convert the gotten value to an angle, multiply by π/180.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Gets the cosine of 60°.

```
DATA# = Cos(60/180*3.141592)
```

The result is shown below.

```
DATA# = 0.5
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

Atn on page 8-36
Sin on page 8-309
UnitData on page 8-341
GetUnitData on page 8-153
Tan on page 8-323

Crspoint

Gets the intersection between 2 straight lines.

Format:

Crspoint <line1()>, <line2()>, <x>, <y>

Parameter:

Parameter name	Data type	Description
<line1()>	Real number array	Straight line component of the straight line 1 to get the intersection
<line2()>	Real number array	Straight line component of the straight line 2 to get the intersection
<x>	Real number	Variable that holds the X coordinate of the intersection gotten
<y>	Real number	Variable that holds the Y coordinate of the intersection gotten

Return value:

None.

Description:

Gets the intersection point of the line component specified in the <line1()> parameter and the line component specified in the <line2()> parameter. In the <x> parameter and <y> parameter, specify the respective variables that will hold the X coordinate and Y coordinate of the intersection point.
In the <line1()> parameter and in the <line2()> parameter, specify a 1D double precision real number array with "a" in element 0, "b" in element 1, and "c" in element 2, where a, b, and c satisfy the linear equation ax + by + c = 0 , without adding element numbers but adding () to the variables.
This macro function is mainly used to get the intersection point of lines gotten with the Lsqumeth function.
If an undefined array is specified a parameter, an "Undefined array" error will occur.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:

None.

Example:

Gets the intersection point of two gotten lines. The two lines are respectively gotten using Processing Units 1 to 4 and Processing Units 5 to 8.

```
Dim POS1X#(3), POS1Y#(3), POS2X#(3), POS2Y#(3), PARAM1#(2), PARAM2#(2)

Rem Initialize variables for straight line 1
For I&=0 To 3
   GetUnitData I&+1, "X", POS1X#(I&)
   GetUnitData I&+1, "Y", POS1Y#(I&)
Next

Rem Get the straight line 1st component.
Lsqumeth 4, POS1X#(), POS1Y#(), PARAM1#()

Rem Initialize variables for straight line 2
For I&=0 To 3
   GetUnitData I&+5, "X", POS2X#(I&)
   GetUnitData I&+5, "Y", POS2Y#(I&)
Next

Rem Get the straight line 2nd component.
Lsqumeth 4, POS2X#(), POS2Y#(), PARAM2#()

Rem 2 Get the intersection between 2 straight lines.
Crspoint PARAM1#(), PARAM2#(), CRSX#, CRSY#
Erase POS1X#(), POS1Y#(), POS2X#(), POS2Y#(), PARAM1#(), PARAM2#()
```

Usable Modules:

Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:

Version 3.50 or later

Related Items:

Erase on page 8-118
Lsqumeth on page 8-192
GetUnitData on page 8-153
UnitData on page 8-341

Date$

Reads out the date from the internal clock.

Format:
Date$

Parameter:
None.

Return value:
Returns the date as a character string value.
The date value is a character string of the internal clock date whose year (YY), month (MM), and day (DD) separated by a slash (/). The range of each is indicated below.

- Year (YY): 00 to 80
- Month (MM): 01 to 12
- Day (DD): 01 to 31

Description:
Reads the date from the internal clock and returns the date value (YY/MM/DD) in character string format.
The year is expressed as a value from 00 to 80, representing 2000 to 2080.
The internal clock can be adjusted in Date-time Settings under System settings. (For details, refer to *Date-time setting [Other]* in the *Vision System FH/FHV Series User's Manual (Cat. No. Z365)).*
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Reads the date in the internal clock and outputs the date to the system status console window.

```
Rem Read out the date from the internal clock.
TODAY$ = Date$

Rem Output the read date to the system status console window.
Print "20";TODAY$
```

The result is shown below.

```
2011/03/10
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

GetSystemData on page 8-148
Piece$ on page 8-219
SetSystemData on page 8-295
Mid$ on page 8-205
Print on page 8-221
Time$ on page 8-326

Debug

Set the program execution formand information output method.

Format:
Debug <mode>

Parameter:

Parameter name	Data type	Description
<mode>	Integer	Execution form and information output method 0: Release mode, no error description is output when an error occurs 1: Release mode, an error description is output to the system status console window when an error occurs. 2: Release mode, the contents of each line are output to the system status console window when the program is executed. 3: Release mode, an error description is output to the message box when an error occurs. 16: Debug mode, no error description is output when an error occurs 17: Debug mode, an error description is output to the system status console window when an error occurs. 18: Debug mode, the contents of each line are output to the system status console window when the program is executed. 19: Debug mode, an error description is output to the error box when an error occurs.

Return value:
None.

Description:
Sets the program execution form and information output method specified in the <mode> parameter. Debug mode can be set to debug the program using macro functions that are only executed in debug mode. Set the mode to release mode after the debug so that there will be no need of removing Debug-Print functions and other macro functions that are only used in debug mode from the program. (For details, refer to *6-1 How to Use the Debug Function* on page 6-2.)
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Uses the MCRINIT subroutine in the unit macro processing unit to set the program execution form to "debug mode" and information output method so as to output error descriptions to the system status console window at an error occurrence.

```
*MCRINIT

   Rem Output an error description to the system status console window when an err
or occurs in debug mode.
   Debug 17

Return
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 5.20 or later

Related Items:

Cls on page 8-52
Cont on page 8-53
DebugPrint on page 8-69
List on page 8-184
Print on page 8-221
SetStop on page 8-293
SetVar on page 8-307
Stop on page 8-312
SubList on page 8-320
VarList on page 8-353

DebugPrint

Outputs debug information to the system status console window.

Format:
DebugPrint <expression>[;|, <expression>...]

Parameter:

Parameter name	Data type	Description
<expression>	Integer Real number Character string Array	Numerical expression or character string to be output

Return value:
None.

Description:
Outputs the numerical expression or character string specified in the <expression> parameter to the system status console window. (For details, refer to *3-1-2 Description of the System Status Console Window* on page 3-7.)
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
- This macro function is only enabled when specified in debug mode with the Debug function. Specifying other values than the range above will treat the statement with this function in the same manner

with the Rem function (i.e., ignores the statement). (For details, refer to *6-1 How to Use the Debug Function* on page 6-2.)

- After the data output to the system status console window, the window is displayed on top of the sensor controller main screen. To display the system status console window on top of the main screen, click [_] on the upper-right of the system status console window or press [Alt] + [Tab] on the connected USB keyboard to the sensor controller.

Example:

Outputs a debug information (character string) to the system status console window in debug mode.

```
Rem Set the execution form to debug mode.
Debug 18

Rem Output character string "Result = OK" as the debug information.
DebugPrint "Result = " + "OK"

Rem Set the execution form to release mode.
Debug 1
```

Usable Modules:

Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:

Version 5.20 or later

Related Items:

Cls on page 8-52
Cont on page 8-53
Debug on page 8-68
List on page 8-184
Print on page 8-221
SetStop on page 8-293
SetVar on page 8-307
Stop on page 8-312
SubList on page 8-320
VarList on page 8-353

DeleteProfileData

Deletes the settings file (.ini file) data.

Format:

DeleteProfileData <fileName>, <section>[, <deleteSectionHeader> | <key>]

Parameter:

Parameter name	Data type	Description
<fileName>	Character string	Absolute path to the file
<section>	Character string	The section name
<deleteSection-Header>	Integer	Delete section • 0: Don't delete section • 1: Delete section
<key>	Character string	The Key name

Return value:

None.

Description:

Deletes the data in the section specified by the <section> parameter of the file specified by the <fileName> parameter.
If you enter an integer of 1 or more in the <deleteSectionHeader> parameter, the entire section specified by the <section> parameter will be deleted.
If the <key> parameter is specified, only the data of the corresponding Key name included in the section specified by the <section> parameter will be deleted.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:

None.

Example:

In the "Setting" section of the .ini file, delete the data with the Key name "Logging".

```
DeleteProfileData "E:\Setting.ini", "Setting", "Logging"
```

Usable Modules:

Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:

Version 6.40 or later

Related Items:

GetProfileData on page 8-145

SetProfileData on page 8-285

DeleteUnit

Deletes a processing unit.

Format:
DeleteUnit <unitNo>

Parameter:

Parameter name	Data type	Description
<unitNo>	Integer	Processing unit number (0 to (number of processing units of current scene minus one)) of processing unit to be deleted.

Return value:
None.

Description:
Deletes the processing unit specified in the <unitNo> parameter of the current scene from the measurement flow.
The processing unit numbers of processing units after the processing unit number specified in the <unitNo> parameter are moved up by 1.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
Execute this macro function when the BUSY signal or other measurement in progress signal is ON and measurement is prohibited. (For details, refer to *5-1-4 State Transitions and Execution Timing* on page 5-13.)

Example:
Deletes the processing unit of unit number 2.

```
DeleteUnit 2
```

Usable Modules:
Scene Control Macro / Communication Command Macro

Supported Versions:
Version 3.50 or later

Related Items:

AssignUnit on page 8-35
CopyUnit on page 8-58
MeasureStart on page 8-200
MoveUnit on page 8-209
CheckUnit on page 8-43
InsertUnit on page 8-168
MeasureStop on page 8-201
UnitCount on page 8-340

Dim

Defines the array variable.

Format:

Dim <arrayName>(<maxCount>[, <maxCount>[, <maxCount>[, <maxCount>]]])

Parameter:

Parameter name	Data type	Description
<arrayName>	-	Used array variable name
<maxCount>	Integer	Maximum value of the subscript

Return value:

None.

Description:

Defines a 1D to 4D array with maximum dimensional length specified in the <maxCount> parameter for each dimension.
Add one of type identifiers to the end of the parameter specified in the <arrayName>. (Refer to *4-1-3 Variable* on page 4-6.)
Release the array variables defined with this macro function by executing the Erase function.
If the number of array dimension is different, two arrays with the same variable name are treated as the same variable.
An array variable and a variable with the same name are treated as different variables.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.
Dim is used when defining temporary variables, array variables, and reference variables by the Option Explicit command. (For details, refer to *Macro Variable Check Function* on page 4-10.)

Usage Cautions:

- The behavior changes depending on the use of the Option Explicit command.
  When the Option Explicit command is used, and if array variables are defined twice due to check of undefined or duplicate variables by the command, an error occurs for multiple definitions.
  If the array variables are re-defined while the Option Explicit command is not used, the previously defined array variables are released first, and then they will be redefined.

Example:

Defines the array.

```
Dim XY&(3)
Dim XY#(7, 15)
Dim CHARA$(31, 63, 127, 255)
```

Usable Modules:

Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:

Version 5.40 or later

Related Items:

Erase on page 8-118
ReDim on page 8-233
Option Explicit on page 8-217

DisableUnit

Switches the measurement ON/OFF of the specified unit.

Format:
DisableUnit <unitNo>, <disabled>

Parameter:

Parameter name	Data type	Description
<unitNo>	Integer	Processing unit number
<disabled>	Integer	0: Measurement ON 1: Measurement OFF

Return value:
None.

Description:
Changes the measurement ON/OFF status of the processing unit specified by the <unitNo> parameter to the status specified by the <disabled> property.
If a non-existent unit number is specified as the <unitNo> parameter, an Illegal function call error will occur.
No error will occur even if an undefined value is specified for the <disabled> parameter.
The measurement OFF status cannot be set for the unit placed at the top of the flow. An Illegal function call error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
Execute this macro function when the BUSY signal or other measurement in progress signal is ON and measurement is prohibited. (For details, refer to *5-1-4 State Transitions and Execution Timing* on page 5-13.)

Example:
Sets the measurement OFF status for the unit of unit number 3.

```
MeasureStop
DisableUnit 3, 1
MeasureStart
```

Usable Modules:
Scene Control Macro/Communication Command Macro

Supported Versions:
Version 6.50 or later

Related Items:

MeasureStart on page 8-200
MeasureStop on page 8-201

DisplaySubNo

Get the sub-image number of the displayed sub-image.

Format:
DisplaySubNo

Parameter:
None.

Return value:
Returns the sub-image number as an integer value.

Description:
Gets the sub-image number of the displayed sub-image set in the image window on the main screen. If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Uses the *MEASUREDISPG subroutine in the unit macro processing unit to change the display in the image window according to the set image display sub-number in the image window of the main screen.

```
*MEASUREDISPG

   Rem Get the displayed sub-image number.
   SUBNO& = DisplaySubNo

   Rem Change the display on the image window according to the sub-image number of
 the sub-image to be displayed.
   Select SUBNO&
   Case 1
      Rem If the gotten sub-image number is 1, the title of processing unit 1 is d
isplayed with the color in accordance with the judgment result.
      SetTextStyle 24, TA_LEFT, UnitJudge(1), 0, FONTSTYLE_NORMAL
      TEXT$ = UnitTitle$(1)
   Case 2
      Rem If the gotten sub-image number is 2, the title of processing unit 2 is d
isplayed with the color in accordance with the judgment result.
      SetTextStyle 24, TA_LEFT, UnitJudge(2), 0, FONTSTYLE_NORMAL
      TEXT$ = UnitTitle$(2)
   Case Else
      Rem If the gotten sub-image number is other than 1 and 2, "Error" is display
ed in the "unmeasured" color.
      SetTextStyle 24, TA_LEFT, JUDGE_NC, 0, FONTSTYLE_NORMAL
      TEXT$ = "Error"
   End Select

   Rem Displays text on the image window.
   DrawTextG TEXT$, 50, 0, 0, UnitNo

Return
```

Usable Modules:
Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

DrawTextG on page 8-111
UnitJudge on page 8-348
UnitTitle$ on page 8-350
SetTextStyle on page 8-297
UnitNo on page 8-349

DisplayUnitNo

Gets the selection state of the processing unit number of theflow window.

Format:
DisplayUnitNo

Parameter:
None.

Return value:
Returns the processing unit number as an integer value.

Description:
Gets the processing unit number of the unit selected in the flow window.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
In the communication command macro, gets the processing unit number selected in the flow window.

```
Rem Get the processing unit number selected in the flow window.
NO& = DisplayUnitNo

Rem Set the processing unit number in the response data of the communication comma
nd.
ResponseString$ = Str$(NO&)
CommandResponse& = 0
```

Usable Modules:
Scene Control Macro / Communication Command Macro

Supported Versions:
Version 3.50 or later

Related Items:

GetImageWindow on page 8-137
UnitNo on page 8-349
SetDisplayUnitNo on page 8-271
Ut on page 8-351

Do Loop While

Repeatedly executes the statements between Do and Loop while the specifiedcondition meets.

Format:
Do
<statement>
Loop While <expression>

Parameter:

Parameter name	Data type	Description
<statement>	-	Statement to be executed repeatedly
<expression>	-	Conditional logical expression for which gets a repetition of operation (Refer to *4-1-5 Operator* on page 4-12)

Return value:
None.

Description:
The statement is repeatedly executed if the specified logical expression by the <expression> parameter is true as a result of the specified Do block execution by the <statement> parameter.
If the Exit Do statement is used in the Do block statement, the statement force stops the repeating execution of the program immediately.
If the program process is jumped into or out of the Do block statement using the Goto or Gosub function, unexpected operation may occur.
If neither the Do statement nor the Loop While statement is used, either the "DO without LOOP", "LOOP without DO", or "EXIT without DO" error will occur depending on the statement that is used.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Repeats the process until the loop counter reaches a constant value.

```
NUM& = 0

Rem Repeat the process.
Do
   NUM& = NUM& + 1
Loop While NUM& < 100

Print NUM&
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

For To Step Next on page 8-132 *Print* on page 8-221

Dposline

Gets the shortest distance between the line and point.

Format:
Dposline (<x>, <y>, <line()>)

Parameter:

Parameter name	Data type	Description
<x>	Real number	X coordinate of the points to get the distance
<y>	Real number	Y coordinate of the points to get the distance
<line()>	Real number array	Straight line component of the straight line to get the distance

Return value:
Returns the shortest distance as a real number value.

Description:
Gets the shortest distance between the point that has the X coordinate specified in the <x> parameter and the Y coordinate specified in the <y> parameter, and the line component specified in the <line()> parameter.
In the <line()> parameter, specify a 1D real number array with "a" in element 0, "b" in element 1, and "c" in element 2, where a, b, and c satisfy the linear equation ax + by + c = 0, without adding element numbers but adding () to the variables.
This macro function is mainly used to get the deviation and distribution from an origin point for a line gotten with the Lsqumeth function.
If an undefined array is specified a parameter, an "Undefined array" error will occur.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Gets the distribution and deviation for a line gotten from four points.

```
Dim POSX#(3), POSY#(3), PARAM#(2), DIST#(3)

Rem Initialize variables for straight line.
For I&=0 To 3
   GetUnitData I&+1, "X", POSX#(I&)
   GetUnitData I&+1, "Y", POSY#(I&)
Next

Rem Get the straight line component.
Lsqumeth 4, POSX#(), POSY#(), PARAM#()
SUMDIST# = 0
For I&=0 To 3
   Rem Calculate the shortest distance between the straight line and point.
   DIST#(I&) = Dposline(POSX#(I&), POSY#(I&), PARAM#())
   SUMDIST# = SUMDIST# + DIST#(I&)
Next

Erase POSX#(), POSY#(), PARAM#(), DIST#()
```

Usable Modules:

Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:

Version 3.50 or later

Related Items:

Erase on page 8-118
Lsqumeth on page 8-192
GetUnitData on page 8-153
UnitData on page 8-341

DrawArc

Draw the arc on the image window.

Format:
DrawArc <x>, <y>, <radius>, <start>, <end>, <imageNo>[, <unitNo>]

Parameter:

Parameter name	Data type	Description
<x>	Integer	The center X coordinate of the drawn arc
<y>	Integer	The center Y coordinate of the drawn arc
<radius>	Integer	Radius of the drawn arc
<start>	Integer	Starting angle of the drawn arc
<end>	Integer	Ending angle of the drawn arc
<imageNo>	Integer	Measurement image number to draw on (always 0)
<unitNo>	Integer	Processing unit number to display the processing unit (0 to (the number of registered processing units in the current scene minus one))

Return value:
None.

Description:
On the measurement image whose specified image number is in the <imageNo> parameter, draw an arc of the specified radius by the <radius> parameter and whose starting angle and ending angle are specified in the <start> and <end> parameters respectively at the center coordinates specified in the <x> and <y> parameters.
Specify the corresponding Unit Macro processing unit number in the <unitNo> parameter to draw the image at the position coordinates before applying position compensation. If the <unitNo> parameter is omitted, the image is drawn at the position coordinates after applying position compensation.
In the <X> and <Y> parameters, specify the camera coordinates whose origin is at the upper-left corner of the image.
In the <start> and <end> parameters, specify the angle so that the angle increases in a clockwise direction respect to the positive X-axis of the camera coordinates.
Normally 0 should be specified in the <imageNo> parameter.
In the <unitNo> parameter, normally specify the processing unit number of the Unit Macro processing unit that executes this process.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If a value outside the range -2147483648 to 2147483647 is specified as an integer parameter, an "Overflow" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
- This macro function can only be used in the *MEASUREDISPI or *MEASUREDISPG subroutine. If used in another subroutine, an "Illegal function call" error will occur.

- Set the line type, color, or width in SetDrawStyle before drawing. If you don't specify these settings, the image is drawn with the previous settings.

Example:

Uses the *MEASUREDISPG subroutine of the Unit Macro processing unit to draw an arc of the measured radius whose starting angle is -90° and ending angle is 180° at the measured center coordinates by the circular scan edge position processing unit (Processing Unit number 5). The measured X and Y coordinates and radius can be gotten with External Reference Data numbers 5 to 7 respectively.
To display the string at the fixed position regardless of the position compensation result, specify the assigned processing unit number to this Unit Macro processing unit (where the *MEASUREDISPG subroutine is used) for the <unitNo> parameter.

```
*MEASUREDISPG

   Rem Get the measurement result.
   GetUnitData 5, 5, X#
   GetUnitData 5, 6, Y#
   GetUnitData 5, 7, R#

   Rem Set the drawing attribute.
   SetDrawStyle PS_SOLID,1,JUDGE_OK

   Rem Draw the image.
   DrawArc Int(X#), Int(Y#), Int(R#), -90, 180, 0, UnitNo

Return
```

Usable Modules:

Unit Macro

Supported Versions:

Version 3.50 or later

Related Items:

GetUnitData on page 8-153
SetDrawStyle on page 8-272
UnitNo on page 8-349
Int on page 8-169
UnitData on page 8-341
Ut on page 8-351

DrawArcW

Draw the wide arc on the image window.

Format:
DrawArcW <x>, <y>, <radius>, <start>, <end>, <width>, <imageNo>[, <unitNo>]

Parameter:

Parameter name	Data type	Description
<x>	Integer	The center X coordinate of the drawn wide arc
<y>	Integer	The center Y coordinate of the drawn wide arc
<radius>	Integer	Radius of the drawn wide arc
<start>	Integer	Starting angle of the drawn wide arc
<end>	Integer	Ending angle of the drawn wide arc
<width>	Integer	Width of the drawn wide arc
<imageNo>	Integer	Measurement image number to draw on (always 0)
<unitNo>	Integer	Processing unit number to display the processing unit (0 to (the number of registered processing units in the current scene minus one))

Return value:
None.

Description:
On the measurement image whose specified image number is in the <imageNo> parameter, draw a wide arc of whose radius, width, starting angle, and ending angle are specified in the <radius>, <width>, <start>, and <end> parameters respectively at the center coordinates specified in the <x> and <y> parameters.
Specify the corresponding Unit Macro processing unit number in the <unitNo> parameter to draw the image at the position coordinates before applying position compensation. If the <unitNo> parameter is omitted, the image is drawn at the position coordinates after applying position compensation.
In the <X> and <Y> parameters, specify the camera coordinates whose origin is at the upper-left corner of the image.
In the <start> and <end> parameters, specify the angle so that the angle increases in a clockwise direction respect to the positive X-axis of the camera coordinates.
Normally 0 should be specified in the <imageNo> parameter.
In the <unitNo> parameter, normally specify the processing unit number of the Unit Macro processing unit that executes this process.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If a value outside the range -2147483648 to 2147483647 is specified as an integer parameter, an "Overflow" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
- This macro function can only be used in the *MEASUREDISPI or *MEASUREDISPG subroutine. If used in another subroutine, an "Illegal function call" error will occur.

- Set the line type, color, or width in SetDrawStyle before drawing. If you don't specify these settings, the image is drawn with the previous settings.

Example:

Uses the *MEASUREDISPG subroutine of the Unit Macro processing unit to draw an arc of the measured radius whose starting angle is -90° and ending angle is 180° at the measured center coordinates by the circular scan edge position processing unit (Processing Unit number 5). The displayed arc is a wide arc whose outer radius is the measured maximum radius and whose inner radius is the measured minimum radius. The measured X/Y coordinates and radius, and maximum/minimum radii can be gotten with External Reference Data numbers 5, 6, 7, 8, and 9 respectively.
To display the string at the fixed position regardless of the position compensation result, specify the assigned processing unit number to this Unit Macro processing unit (where the *MEASUREDISPG subroutine is used) for the <unitNo> parameter.

```
*MEASUREDISPG

   Rem Get the measurement result.
   GetUnitData 5, 5, X#
   GetUnitData 5, 6, Y#
   GetUnitData 5, 7, R#
   GetUnitData 5, 8, R_MAX#
   GetUnitData 5, 9, R_MIN#

   Rem Determine the width.
   W# = R_MAX# - R_MIN#

   Rem Set the drawing attribute.
   SetDrawStyle PS_SOLID,1,JUDGE_OK

   Rem Draw the image.
   DrawArcW Int(X#), Int(Y#), Int(R#), -90, 180, Int(W#), 0, UnitNo

Return
```

Usable Modules:

Unit Macro

Supported Versions:

Version 3.50 or later

Related Items:

GetUnitData on page 8-153
SetDrawStyle on page 8-272
UnitNo on page 8-349
Int on page 8-169
UnitData on page 8-341
Ut on page 8-351

DrawBox

Draws a rectangle on the image window.

Format:
DrawBox <x0>, <y0>, <x1>, <y1>, <imageNo>[, <unitNo>]

Parameter:

Parameter name	Data type	Description
<x0>	Integer	The upper-left corner X coordinate of the drawn rectangle
<y0>	Integer	The upper-left corner Y coordinate of the drawn rectangle
<x1>	Integer	The lower-right corner X coordinate of the drawn rectangle
<y1>	Integer	The lower-right corner Y coordinate of the drawn rectangle
<imageNo>	Integer	Measurement image number to draw on (always 0)
<unitNo>	Integer	Processing unit number to display the processing unit (0 to (the number of registered processing units in the current scene minus one))

Return value:
None.

Description:
On the measurement image whose specified image number is in the <imageNo> parameter, draws a rectangle that has the specified upper left vertex coordinates by the <x0> and <y0> parameters and the specified lower right vertex coordinates by the <x1> and <y1> parameters.
Specify the corresponding Unit Macro processing unit number in the <unitNo> parameter to draw the image at the position coordinates before applying position compensation. If the <unitNo> parameter is omitted, the image is drawn at the position coordinates after applying position compensation.
In the <x0>, <y0>, <x1>, and <y1> parameters, specify the camera coordinates whose origin is at the upperleft corner of the image.
Normally 0 should be specified in the <imageNo> parameter.
In the <unitNo> parameter, normally specify the processing unit number of the Unit Macro processing unit that executes this process.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If a value outside the range -2147483648 to 2147483647 is specified as an integer parameter, an "Overflow" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
- This macro function can only be used in the *MEASUREDISPI or *MEASUREDISPG subroutine. If used in another subroutine, an "Illegal function call" error will occur.
- Set the line type, color, or width in SetDrawStyle before drawing. If you don't specify these settings, the image is drawn with the previous settings.

Example:
Uses the *MEASUREDISPG subroutine of the Unit Macro processing unit to display the rectangle whose upper-left and lower-right corner coordinates are the coordinates measured by the Processing

Unit numbers 5 and 6 search processing units respectively. The measured X and Y coordinates can be gotten with External Reference Data numbers 6 and 7 respectively.
To display the string at the fixed position regardless of the position compensation result, specify the assigned processing unit number to this Unit Macro processing unit (where the *MEASUREDISPG subroutine is used) for the <unitNo> parameter.

```
*MEASUREDISPG

   Rem Get the measurement result.
   GetUnitData 5, 6, X_LEFTTOP#
   GetUnitData 5, 7, Y_LEFTTOP#
   GetUnitData 6, 6, X_RIGHTBOTTOM#
   GetUnitData 6, 7, Y_RIGHTBOTTOM#

   Rem Set the drawing attribute.
   SetDrawStyle PS_SOLID,1,JUDGE_OK

   Rem Draw the image.
   DrawBox Int(X_LEFTTOP#), Int(Y_LEFTTOP#), Int(X_RIGHTBOTTOM#), Int(Y_RIGHTBOTTO
M#), 0,
   UnitNo

Return
```

Usable Modules:
Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

GetUnitData on page 8-153
SetDrawStyle on page 8-272
UnitNo on page 8-349
Int on page 8-169
UnitData on page 8-341
Ut on page 8-351

DrawCircle

Draw a circle on the image window.

Format:
DrawCircle <x>, <y>, <radius>, <imageNo>[, <unitNo>]

Parameter:

Parameter name	Data type	Description
<x>	Integer	The center X coordinate of the drawn circle
<y>	Integer	The center Y coordinate of the drawn circle
<radius>	Integer	Radius of the drawn circle
<imageNo>	Integer	Measurement image number to draw on (always 0)
<unitNo>	Integer	Processing unit number to display the processing unit (0 to (the number of registered processing units in the current scene minus one))

Return value:
None.

Description:
On the measurement image whose specified image number is in the <imageNo> parameter, draw a circle of the specified radius by the <radius> parameter at the center coordinates specified in the <x> and <y> parameters.
Specify the corresponding Unit Macro processing unit number in the <unitNo> parameter to draw the image at the position coordinates before applying position compensation. If the <unitNo> parameter is omitted, the image is drawn at the position coordinates after applying position compensation.
In the <X> and <Y> parameters, specify the camera coordinates whose origin is at the upper-left corner of the image.
Normally 0 should be specified in the <imageNo> parameter.
In the <unitNo> parameter, normally specify the processing unit number of the Unit Macro processing unit that executes this process.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If a value outside the range -2147483648 to 2147483647 is specified as an integer parameter, an "Overflow" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
- This macro function can only be used in the *MEASUREDISPI or *MEASUREDISPG subroutine. If used in another subroutine, an "Illegal function call" error will occur.
- Set the line type, color, or width in SetDrawStyle before drawing. If you don't specify these settings, the image is drawn with the previous settings.

Example:
Uses the *MEASUREDISPG subroutine of the Unit Macro processing unit to draw a circle of the measured radius by the circular scan edge position processing unit (Processing Unit number 5) at the measured center coordinates by the same circular scan edge position processing unit used for

measuring the radius. The measured X and Y coordinates and radius can be gotten with External Reference Data numbers 5 to 7 respectively.
To display the string at the fixed position regardless of the position compensation result, specify the assigned processing unit number to this Unit Macro processing unit (where the *MEASUREDISPG subroutine is used) for the <unitNo> parameter.

```
*MEASUREDISPG

   Rem Get the measurement result.
   GetUnitData 5, 5, X#
   GetUnitData 5, 6, Y#
   GetUnitData 5, 7, R#

   Rem Set the drawing attribute.
   SetDrawStyle PS_SOLID,1,JUDGE_OK

   Rem Draw the image.
   DrawCircle Int(X#), Int(Y#), Int(R#), 0, UnitNo

Return
```

Usable Modules:
Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

GetUnitData on page 8-153
SetDrawStyle on page 8-272
UnitNo on page 8-349
Int on page 8-169
UnitData on page 8-341
Ut on page 8-351

DrawCircleW

Draw the wide circle on the image window.

Format:
DrawCircleW <x>, <y>, <width>, <radius>, <imageNo>[, <unitNo>]

Parameter:

Parameter name	Data type	Description
<x>	Integer	The center X coordinate of the drawn circle
<y>	Integer	The center Y coordinate of the drawn circle
<width>	Integer	The width of the drawn wide circle
<radius>	Integer	Radius of the drawn circle
<imageNo>	Integer	Measurement image number to draw on (always 0)
<unitNo>	Integer	Processing unit number to display the processing unit (0 to (the number of registered processing units in the current scene minus one))

Return value:
None.

Description:
On the measurement image whose specified image number is in the <imageNo> parameter, draw a wide circle of the specified radius by the <radius> parameter and the specified width by the <width> parameter at the center coordinates specified in the <x> and <y> parameters.
Specify the corresponding Unit Macro processing unit number in the <unitNo> parameter to draw the image at the position coordinates before applying position compensation. If the <unitNo> parameter is omitted, the image is drawn at the position coordinates after applying position compensation.
In the <X> and <Y> parameters, specify the camera coordinates whose origin is at the upper-left corner of the image.
Normally 0 should be specified in the <imageNo> parameter.
In the <unitNo> parameter, normally specify the processing unit number of the Unit Macro processing unit that executes this process.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If a value outside the range -2147483648 to 2147483647 is specified as an integer parameter, an "Overflow" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
- This macro function can only be used in the *MEASUREDISPI or *MEASUREDISPG subroutine. If used in another subroutine, an "Illegal function call" error will occur.
- Set the line type, color, or width in SetDrawStyle before drawing. If you don't specify these settings, the image is drawn with the previous settings.

Example:
Uses the *MEASUREDISPG subroutine of the Unit Macro processing unit to draw a circle of the measured radius by the circular scan edge position processing unit (Processing Unit number 5) at the

measured center coordinates by the same circular scan edge position processing unit used for measuring the radius. This wide circle has an outer radius of the measured maximum radius and an inner radius of the measured minimum radius. The measured X/Y coordinates and radius, and maximum/minimum radii can be gotten with External Reference Data numbers 5, 6, 7, 8, and 9 respectively.
To display the string at the fixed position regardless of the position compensation result, specify the assigned processing unit number to this Unit Macro processing unit (where the *MEASUREDISPG subroutine is used) for the <unitNo> parameter.

```
*MEASUREDISPG

   Rem Get the measurement result.
   GetUnitData 5, 5, X#
   GetUnitData 5, 6, Y#
   GetUnitData 5, 7, R#
   GetUnitData 5, 8, R_MAX#
   GetUnitData 5, 9, R_MIN#

   Rem Determine the width.
   W# = R_MAX# - R_MIN#

   Rem Set the drawing attribute.
   SetDrawStyle PS_SOLID,1,JUDGE_OK

   Rem Draw the image.
   DrawCircleW Int(X#), Int(Y#), Int(W#), Int(R#), 0, UnitNo

Return
```

Usable Modules:
Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

GetUnitData on page 8-153
SetDrawStyle on page 8-272
UnitNo on page 8-349
Int on page 8-169
UnitData on page 8-341
Ut on page 8-351

DrawCursor

Draw the cross-hair cursor on the image window.

Format:
DrawCursor <x>, <y>, <imageNo>[, <unitNo>]

Parameter:

Parameter name	Data type	Description
<x>	Integer	The center X coordinate of the drawn cross-hair cursor
<y>	Integer	The center Y coordinate of the drawn cross-hair cursor
<imageNo>	Integer	Measurement image number to draw on (always 0)
<unitNo>	Integer	Processing unit number to display the processing unit (0 to (the number of registered processing units in the current scene minus one))

Return value:
None.

Description:
On the measurement image whose specified image number is in the <imageNo> parameter, draw a crosshair cursor at the center coordinates specified in the <x> and <y> parameters.
Specify the corresponding Unit Macro processing unit number in the <unitNo> parameter to draw the image at the position coordinates before applying position compensation. If the <unitNo> parameter is omitted, the image is drawn at the position coordinates after applying position compensation.
In the <X> and <Y> parameters, specify the camera coordinates whose origin is at the upper-left corner of the image.
Normally 0 should be specified in the <imageNo> parameter.
In the <unitNo> parameter, normally specify the processing unit number of the Unit Macro processing unit that executes this process.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If a value outside the range -2147483648 to 2147483647 is specified as an integer parameter, an "Overflow" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
- This macro function can only be used in the *MEASUREDISPI or *MEASUREDISPG subroutine. If used in another subroutine, an "Illegal function call" error will occur.
- Set the line type, color, or width in SetDrawStyle before drawing. If you don't specify these settings, the image is drawn with the previous settings.

Example:
Uses the *MEASUREDISPG subroutine of the Unit Macro processing unit to draw a cross-hair cursor at the measured coordinates by the search processing unit (Processing Unit number 5). The measured X and Y coordinates can be gotten with External Reference Data numbers 6 and 7 respectively.

To display the string at the fixed position regardless of the position compensation result, specify the assigned processing unit number to this Unit Macro processing unit (where the *MEASUREDISPG subroutine is used) for the <unitNo> parameter.

```
*MEASUREDISPG

   Rem Get the measurement result.
   GetUnitData 5, 6, X#
   GetUnitData 5, 7, Y#

   Rem Set the drawing attribute.
   SetDrawStyle PS_SOLID,1,JUDGE_OK

   Rem Draw the image.
   DrawCursor Int(X#), Int(Y#), 0, UnitNo

Return
```

Usable Modules:

Unit Macro

Supported Versions:

Version 3.50 or later

Related Items:

GetUnitData on page 8-153
SetDrawStyle on page 8-272
UnitNo on page 8-349
Int on page 8-169
UnitData on page 8-341
Ut on page 8-351

DrawEllipse

Draw the ellipse on the image window.

Format:
DrawEllipse <x>, <y>, <radiusX>, <radiusY>, <imageNo>[, <unitNo>]

Parameter:

Parameter name	Data type	Description
<x>	Integer	The center X coordinate of the drawn ellipse
<y>	Integer	The center Y coordinate of the drawn ellipse
<radiusX>	Integer	Radius in the X direction of the drawn ellipse
<radiusY>	Integer	Radius in the Y direction of the drawn ellipse
<imageNo>	Integer	Measurement image number to draw on (always 0)
<unitNo>	Integer	Processing unit number to display the processing unit (0 to (the number of registered processing units in the current scene minus one))

Return value:
None.

Description:
On the measurement image whose specified image number is in the <imageNo> parameter, draw an ellipse of the specified radius in the X direction by the <radiusX> parameter and the specified radius in the Y direction by the <radius Y> parameter at the center coordinates specified in the <x> and <y> parameters.
Specify the corresponding Unit Macro processing unit number in the <unitNo> parameter to draw the image at the position coordinates before applying position compensation. If the <unitNo> parameter is omitted, the image is drawn at the position coordinates after applying position compensation.
In the <X> and <Y> parameters, specify the camera coordinates whose origin is at the upper-left corner of the image.
Normally 0 should be specified in the <imageNo> parameter.
In the <unitNo> parameter, normally specify the processing unit number of the Unit Macro processing unit that executes this process.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If a value outside the range -2147483648 to 2147483647 is specified as an integer parameter, an "Overflow" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
- This macro function can only be used in the *MEASUREDISPI or *MEASUREDISPG subroutine. If used in another subroutine, an "Illegal function call" error will occur.
- Set the line type, color, or width in SetDrawStyle before drawing. If you don't specify these settings, the image is drawn with the previous settings.

Example:

Uses the *MEASUREDISPG subroutine of the Unit Macro processing unit to display the ellipse whose radii in the X and Y directions are the measured coordinates of the center of gravity by the labeling processing unit (Processing Unit number 5). In this example, set the judgement conditions for the labeling processing unit to "Gravity X", "Gravity Y", "Elliptic major axis", and "Elliptic minor axis" from label number 0. Therefore, the assigned external reference data numbers to the gravity X, gravity Y, elliptic major axis, and elliptic minor axis parameters are 1000, 1100, 1200, and 1300 respectively.
To display the string at the fixed position regardless of the position compensation result, specify the assigned processing unit number to this Unit Macro processing unit (where the *MEASUREDISPG subroutine is used) for the <unitNo> parameter.

```
*MEASUREDISPG

   Rem Get the measurement result.
   GetUnitData 5, 1000, X#
   GetUnitData 5, 1100, Y#
   GetUnitData 5, 1200, XDIAMETER#
   GetUnitData 5, 1300, YDIAMETER#

   Rem Set the drawing attribute.
   SetDrawStyle PS_SOLID,1,JUDGE_OK

   Rem Draw the image.
   DrawEllipse Int(X#), Int(Y#), Int(XDIAMETER# / 2), Int(YDIAMETER# / 2), 0, Unit
No

Return
```

Usable Modules:
Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

GetUnitData on page 8-153
SetDrawStyle on page 8-272
UnitNo on page 8-349
Int on page 8-169
UnitData on page 8-341
Ut on page 8-351

DrawFigure

Draw a figure on the image window.

Format:
DrawFigure <figure()>, <imageNo>[, <unitNo>]

Parameter:

Parameter name	Data type	Description
<figure()>	Integer array	1D array that stores the drawn figure data (Refer to *9-6 Figure Data List* on page 9-70.)
<imageNo>	Integer	Measurement image number to draw on (always 0)
<unitNo>	Integer	Processing unit number to display the processing unit (0 to (the number of registered processing units in the current scene minus one))

Return value:
None.

Description:
On the measurement image whose specified image number is in the <imageNo> parameter, draw a figure specified in the <figure()> parameter.
Specify the corresponding Unit Macro processing unit number in the <unitNo> parameter to draw the image at the position coordinates before applying position compensation. If the <unitNo> parameter is omitted, the image is drawn at the position coordinates after applying position compensation.
In the <figure()> parameter, specify the 1D integer array variable that will hold the figure data by adding only () without specifying an element number.
Normally 0 should be specified in the <imageNo> parameter.
In the <unitNo> parameter, normally specify the processing unit number of the Unit Macro processing unit that executes this process.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If a value outside the range -2147483648 to 2147483647 is specified as an integer parameter, an "Overflow" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
- This macro function can only be used in the *MEASUREDISPI or *MEASUREDISPG subroutine. If used in another subroutine, an "Illegal function call" error will occur.
- Set the line type, color, or width in SetDrawStyle before drawing. If you don't specify these settings, the image is drawn with the previous settings.

Example:
Uses the *MEASUREDISPG subroutine of the Unit Macro processing unit to simultaneously display registered model figures for the search processing units (Processing Unit numbers 5, 6, 7, and 8). The model registration figure of the Search processing item is figure 0. (Refer to *9-7 List of Figure Numbers* on page 9-73.)

Use the Dim function to define an array with element number that is larger than the element number used for the figure data.

```
*MEASUREDISPG

   Dim FIGURE1&(255), FIGURE2&(255), FIGURE3&(255), FIGURE4&(255)

   Rem Get the figure data of the model figure.
   GetUnitFigure 5, 0, FIGURE1&()
   GetUnitFigure 6, 0, FIGURE2&()
   GetUnitFigure 7, 0, FIGURE3&()
   GetUnitFigure 8, 0, FIGURE4&()

   Rem Set the drawing attribute.
   SetDrawStyle PS_SOLID,1,JUDGE_OK

   Rem Draw the image.
   DrawFigure FIGURE1&(), 0, UnitNo
   DrawFigure FIGURE2&(), 0, UnitNo
   DrawFigure FIGURE3&(), 0, UnitNo
   DrawFigure FIGURE4&(), 0, UnitNo

Return
```

Usable Modules:

Unit Macro

Supported Versions:

Version 3.50 or later

Related Items:

GetUnitData on page 8-153
SetDrawStyle on page 8-272
UnitNo on page 8-349
Int on page 8-169
UnitData on page 8-341
Ut on page 8-351

DrawFillImage

Draw the fill image on the image window.

Format:
DrawFillImage <color>

Parameter:

Parameter name	Data type	Description
<color>	Integer	Color value of the color to fill with (Refer to *RGB* on page 8-242.)

Return value:
None.

Description:
Draw the color filled image specified in the <color> parameter.
In <color> parameter, specify the color value gotten with the RGB function.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a value outside the range -2147483648 to 2147483647 is specified as an integer parameter, an "Overflow" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
- This macro function can only be used in the *MEASUREDISPI or *MEASUREDISPG subroutine. If used in another subroutine, an "Illegal function call" error will occur.

Example:
Uses the *MEASUREDISPI subroutine in the Unit Macro processing unit to display a white-filled figure.

```
*MEASUREDISPI

   DrawFillImage RGB(255, 255, 255)

Return
```

Usable Modules:
Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

DrawMeasureImage on page 8-102
DrawUnitImage on page 8-113
RGB on page 8-242

DrawJudgeText

Draws the judgement result of the character string on the textdisplay screen.

Format:
DrawJudgeText <judge>

Parameter:

Parameter name	Data type	Description
<judge>	Integer	Judgement results to be drawn JUDGE_NC: "No judgement (unmeasured)" JUDGE_OK: Judgement: OK JUDGE_NG: Judgement: NG JUDGE_IMAGEERROR: Judgement: NG (Image mismatch) JUDGE_MODELERROR: Judgement: NG (model not register) JUDGE_MEMORYERROR: Judgement: NG (Out of memory) JUDGE_ERRORJudgement: NG (immeasurable)

Return value:
None.

Description:
Draw the specified judgement result string by the <judge> parameter in the text window.
Gotten value with the UnitJudge function can be specified in the <judge> parameter. (Refer to *UnitJudge* on page 8-348.)
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
Type mismatch error is occurred when wrong data is specified as parameter. Illegal function call error is not occurred even if non-exist number, values, combination of data or values.
If a value outside the range -2147483648 to 2147483647 is specified as an integer parameter, an "Overflow" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
- This macro function can only be used in the *MEASUREDISPT subroutine. If used in another subroutine, an "Illegal function call" error will occur.

Example:
Uses the *MEASUREDISPT subroutine in the Unit Macro processing unit to draw the judgement result text string for the processing unit in the text window.

```
*MEASUREDISPT

   DrawJudgeText UnitJudge(UnitNo)

Return
```

Usable Modules:
Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

DrawText on page 8-109
GetUnitData on page 8-153
UnitData on page 8-341
UnitJudge on page 8-348
UnitNo on page 8-349

DrawLine

Draw a straight line on the image window.

Format:
DrawLine <x0>, <y0>, <x1>, <y1>, <imageNo>[, <unitNo>]

Parameter:

Parameter name	Data type	Description
<x0>	Integer	The starting point X coordinate of the drawn straight line
<y0>	Integer	The starting point Y coordinate of the drawn straight line
<x1>	Integer	The ending point X coordinate of the drawn straight line
<y1>	Integer	The ending point Y coordinate of the drawn straight line
<imageNo>	Integer	Measurement image number to draw on (always 0)
<unitNo>	Integer	Processing unit number to display the processing unit (0 to (the number of registered processing units in the current scene minus one))

Return value:
None.

Description:
On the measurement image whose specified image number is in the <imageNo> parameter, draws a line that starts at the specified X- and Y-coordinates by the <x0> and <y0> parameters and ends at the specified Xand Y-coordinates by the <x1> and <y1> parameters.
Specify the corresponding Unit Macro processing unit number in the <unitNo> parameter to draw the image at the position coordinates before applying position compensation. If the <unitNo> parameter is omitted, the image is drawn at the position coordinates after applying position compensation.
In the <x0>, <y0>, <x1>, and <y1> parameters, specify the camera coordinates whose origin is at the upperleft corner of the image.
Normally 0 should be specified in the <imageNo> parameter.
In the <unitNo> parameter, normally specify the processing unit number of the Unit Macro processing unit that executes this process.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If a value outside the range -2147483648 to 2147483647 is specified as an integer parameter, an "Overflow" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
- This macro function can only be used in the *MEASUREDISPI or *MEASUREDISPG subroutine. If used in another subroutine, an "Illegal function call" error will occur.

- Set the line type, color, or width in SetDrawStyle before drawing. If you don't specify these settings, the image is drawn with the previous settings.

Example:
Uses the *MEASUREDISPG subroutine of the Unit Macro processing unit to display the line whose starting and ending point coordinates are the coordinates measured by the Processing Unit numbers 5 and 6 search processing units respectively. The measured X and Y coordinates can be gotten with External Reference Data numbers 6 and 7 respectively.
To display the string at the fixed position regardless of the position compensation result, specify the assigned processing unit number to this Unit Macro processing unit (where the *MEASUREDISPG subroutine is used) for the <unitNo> parameter.

```
*MEASUREDISPG

   Rem Get the measurement result.
   GetUnitData 5, 6, X_START#
   GetUnitData 5, 7, Y_START#
   GetUnitData 6, 6, X_END#
   GetUnitData 6, 7, Y_END#

   Rem Set the drawing attribute.
   SetDrawStyle PS_SOLID,1,JUDGE_OK

   Rem Draw the image.
   DrawLine Int(X_START#), Int(Y_START#), Int(X_END#), Int(Y_END#), 0, UnitNo

Return
```

Usable Modules:
Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

GetUnitData on page 8-153
SetDrawStyle on page 8-272
UnitNo on page 8-349
Int on page 8-169
UnitData on page 8-341
Ut on page 8-351

DrawLineW

Draw the wide straight line on the image window.

Format:
DrawLineW <x0>, <y0>, <x1>, <y1>, <width>, <imageNo>[, <unitNo>]

Parameter:

Parameter name	Data type	Description
<x0>	Integer	The starting point X coordinate of the drawn straight line
<y0>	Integer	The starting point Y coordinate of the drawn straight line
<x1>	Integer	The ending point X coordinate of the drawn straight line
<y1>	Integer	The ending point Y coordinate of the drawn straight line
<width>	Integer	Width of the drawn straight line
<imageNo>	Integer	Measurement image number to draw on (always 0)
<unitNo>	Integer	Processing unit number to display the processing unit (0 to (the number of registered processing units in the current scene minus one))

Return value:
None.

Description:
On the measurement image whose specified image number is in the <imageNo> parameter, draws a wide line that starts at the specified X and Y coordinates by the <x0> and <y0> parameters and ends at the specified X and Y coordinates by the <x1> and <y1> parameters with the specified width by the <width> parameter.
Specify the corresponding Unit Macro processing unit number in the <unitNo> parameter to draw the image at the position coordinates before applying position compensation. If the <unitNo> parameter is omitted, the image is drawn at the position coordinates after applying position compensation.
In the <x0>, <y0>, <x1>, and <y1> parameters, specify the camera coordinates whose origin is at the upperleft corner of the image.
Normally 0 should be specified in the <imageNo> parameter.
In the <unitNo> parameter, normally specify the processing unit number of the Unit Macro processing unit that executes this process.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If a value outside the range -2147483648 to 2147483647 is specified as an integer parameter, an "Overflow" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
- This macro function can only be used in the *MEASUREDISPI or *MEASUREDISPG subroutine. If used in another subroutine, an "Illegal function call" error will occur.
- Set the line type, color, or width in SetDrawStyle before drawing. If you don't specify these settings, the image is drawn with the previous settings.

Example:

Uses the *MEASUREDISPG subroutine of the Unit Macro processing unit to display the wide line with width of 10 whose starting and ending point coordinates are the coordinates measured by the Processing Unit numbers 5 and 6 search processing units respectively. The measured X and Y coordinates can be gotten with External Reference Data numbers 6 and 7 respectively.
To display the string at the fixed position regardless of the position compensation result, specify the assigned processing unit number to this Unit Macro processing unit (where the *MEASUREDISPG subroutine is used) for the <unitNo> parameter.

```
*MEASUREDISPG

   Rem Get the measurement result.
   GetUnitData 5, 6, X_START#
   GetUnitData 5, 7, Y_START#
   GetUnitData 6, 6, X_END#
   GetUnitData 6, 7, Y_END#

   Rem Set the drawing attribute.
   SetDrawStyle PS_SOLID,1,JUDGE_OK

   Rem Rem Draw the image.
   DrawLineW Int(X_START#), Int(Y_START#), Int(X_END#), Int(Y_END#), 10, 0, UnitNo

Return
```

Usable Modules:
Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

GetUnitData on page 8-153
SetDrawStyle on page 8-272
UnitNo on page 8-349
Int on page 8-169
UnitData on page 8-341
Ut on page 8-351

DrawMeasureImage

Draw the measurement image on the image window.

Format:
DrawMeasureImage <imageNo>

Parameter:

Parameter name	Data type	Description
<imageNo>	Integer	Measurement image number to draw on (always 0)

Return value:
None.

Description:
Display an image that has been registered to the Unit Macro processing unit where this macro function is executed and whose image number is specified in the <imageNo> parameter.
Even if a value that does not exist is specified for the <imageNo> parameter, an error will not occur.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If a value outside the range -2147483648 to 2147483647 is specified as an integer parameter, an "Overflow" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
- This macro function can only be used in the *MEASUREDISPI or *MEASUREDISPG subroutine. If used in another subroutine, an "Illegal function call" error will occur.

Example:
Uses the *MEASUREDISPI subroutine in the Unit Macro processing unit to display a measurement image.

```
*MEASUREDISPI

   DrawMeasureImage 0

Return
```

Usable Modules:
Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

DrawFillImage on page 8-96
DrawUnitImage on page 8-113

DrawPoint

Draw a point on the image window.

Format:
DrawPoint <x>, <y>, <imageNo>[, <unitNo>]

Parameter:

Parameter name	Data type	Description
<x>	Integer	The X coordinate of the drawn point
<y>	Integer	The Y coordinate of the drawn point
<imageNo>	Integer	Measurement image number to draw on (always 0)
<unitNo>	Integer	Processing unit number to display the processing unit (0 to (the number of registered processing units in the current scene minus one))

Return value:
None.

Description:
On the measurement image whose specified image number is in the <imageNo> parameter, draw a point at the coordinates specified in the <x> and <y> parameters.
Specify the corresponding Unit Macro processing unit number in the <unitNo> parameter to draw the image at the position coordinates before applying position compensation. If the <unitNo> parameter is omitted, the image is drawn at the position coordinates after applying position compensation.
In the <X> and <Y> parameters, specify the camera coordinates whose origin is at the upper-left corner of the image.
Normally 0 should be specified in the <imageNo> parameter.
In the <unitNo> parameter, normally specify the processing unit number of the Unit Macro processing unit that executes this process.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If a value outside the range -2147483648 to 2147483647 is specified as an integer parameter, an "Overflow" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
- This macro function can only be used in the *MEASUREDISPI or *MEASUREDISPG subroutine. If used in another subroutine, an "Illegal function call" error will occur.
- Set the line type, color, or width in SetDrawStyle before drawing. If you don't specify these settings, the image is drawn with the previous settings.

Example:
Uses the *MEASUREDISPG subroutine of the Unit Macro processing unit to draw a point at the measured coordinates by the search processing unit (Processing Unit number 5). The measured X and Y coordinates can be gotten with External Reference Data numbers 6 and 7 respectively.

To display the string at the fixed position regardless of the position compensation result, specify the assigned processing unit number to this Unit Macro processing unit (where the *MEASUREDISPG subroutine is used) for the <unitNo> parameter.

```
*MEASUREDISPG

   Rem Get the measurement result.
   GetUnitData 5, 6, X#
   GetUnitData 5, 7, Y#

   Rem Set the drawing attribute.
   SetDrawStyle PS_SOLID,1,JUDGE_OK

   Rem Draw the image.
   DrawPoint Int(X#), Int(Y#), 0, UnitNo

Return
```

Usable Modules:

Unit Macro

Supported Versions:

Version 3.50 or later

Related Items:

GetUnitData on page 8-153
SetDrawStyle on page 8-272
UnitNo on page 8-349
Int on page 8-169
UnitData on page 8-341
Ut on page 8-351

DrawPolygon

Draw a polygon on the image window.

Format:

DrawPolygon <count>, <x()>, <y()>, <imageNo>[, <unitNo>]

Parameter:

Parameter name	Data type	Description
<count>	Integer	Number of polygon vertices (0 or larger) to be drawn
<x()>	Integer array	1D array that stores the X coordinate of the drawn polygon
<y()>	Integer array	1D array that stores the Y coordinate of the drawn polygon
<imageNo>	Integer	Measurement image number to draw on (always 0)
<unitNo>	Integer	Processing unit number to display the processing unit (0 to (the number of registered processing units in the current scene minus one))

Return value:

None.

Description:

On the measurement image whose specified image number is in the <imageNo> parameter, draw a polygon having the specified number of vertices by the <count> parameter and whose vertices are at the specified coordinates by the <x()> and <y()> parameters.
Specify the corresponding Unit Macro processing unit number in the <unitNo> parameter to draw the image at the position coordinates before applying position compensation. If the <unitNo> parameter is omitted, the image is drawn at the position coordinates after applying position compensation.
In the <x()> parameter and in the <y()> parameter, specify a 1D integer number array variable that stores a number of coordinate values greater than or equal to the number specified in the <count> parameter, without adding element numbers but adding () to the variables.
In the <X> and <Y> parameters, specify the camera coordinates whose origin is at the upper-left corner of the image.
Normally 0 should be specified in the <imageNo> parameter.
In the <unitNo> parameter, normally specify the processing unit number of the Unit Macro processing unit that executes this process.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If a value outside the range -2147483648 to 2147483647 is specified as an integer parameter, an "Overflow" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:

- This macro function can only be used in the *MEASUREDISPI or *MEASUREDISPG subroutine. If used in another subroutine, an "Illegal function call" error will occur.
- Set the line type, color, or width in SetDrawStyle before drawing. If you don't specify these settings, the image is drawn with the previous settings.

Example:

Uses the *MEASUREDISPG subroutine of the Unit Macro processing unit to draw a polygon whose vertices are at the measured positions by the search processing units (Processing unit numbers 5 to 7). The measured X and Y coordinates can be gotten with External Reference Data numbers 6 and 7 respectively.
To display the string at the fixed position regardless of the position compensation result, specify the assigned processing unit number to this Unit Macro processing unit (where the *MEASUREDISPG subroutine is used) for the <unitNo> parameter.

```
*MEASUREDISPG

    Dim X&(2), Y&(2)

    Rem Get the measurement result.
    GetUnitData 5, 6, X&(0)
    GetUnitData 5, 7, Y&(0)
    GetUnitData 6, 6, X&(1)
    GetUnitData 6, 7, Y&(1)
    GetUnitData 7, 6, X&(2)
    GetUnitData 7, 7, Y&(2)

    Rem Set the drawing attribute.
    SetDrawStyle PS_SOLID,1,JUDGE_OK

    Rem Draw the image.
    DrawPolygon 3, X&(), Y&(), 0, UnitNo

Return
```

Usable Modules:
Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

GetUnitData on page 8-153
SetDrawStyle on page 8-272
UnitNo on page 8-349
Int on page 8-169
UnitData on page 8-341
Ut on page 8-351

DrawSearchFigure

Draw the search figure on the image window.

Format:

DrawSearchFigure <figure()>, <referenceX>, <referenceY>, <measureX>, <measureY>, <measureAngle>, <imageNo>, <unitNo>

Parameter:

Parameter name	Data type	Description
<figure()>	Integer array	1D array that stores the drawn figure data (Refer to *9-6 Figure Data List* on page 9-70.)
<referenceX>	Real number	Detection point X coordinate used for the drawn figure position specification
<referenceY>	Real number	Detection point Y coordinate used for the drawn figure position specification
<measureX>	Real number	Measurement X coordinate used for the drawn figure position specification
<measureY>	Real number	Measurement Y coordinate used for the drawn figure position specification
<measureAngle>	Real number	Measured angle used for the drawn figure angle specification
<imageNo>	Integer	Measurement image number to draw on (always 0)
<unitNo>	Integer	Processing unit number to display the processing unit (0 to (the number of registered processing units in the current scene minus one))

Return value:

None.

Description:

On the measurement image whose specified image number is in the <imageNo> parameter, draw a figure specified in the <figure()> parameter at the specified position with the <referenceX>, <referenceY>, <measureX>, and <measureY> parameter at an angle specified in the <measureAngle>.
With the use of this macro function, drawing of a figure composed of registered model figures and detected points gotten mainly from search processing units and shape search III processing units. This macro function cannot draw images properly if the referenced measurement results from processing units where the model registration or detection point specification is not implemented are specified in the function arguments.
In the <referenceX> parameter and the <referenceY> parameter, normally specify detection points X and Y of the referenced search processing unit.
In the <measureX> parameter, the <measureY> parameter, and the <measureAngle> parameter, normally specify the measurement coordinates X and Y and the measurement angle of a referenced search processing unit.
Specify the corresponding Unit Macro processing unit number in the <unitNo> parameter to draw the image at the position coordinates before applying position compensation. If the <unitNo> parameter is omitted, the image is drawn at the position coordinates after applying position compensation.
In the <figure()> parameter, specify the 1D integer array variable that will hold the figure data by adding only () without specifying an element number.
Normally 0 should be specified in the <imageNo> parameter.

In the <unitNo> parameter, normally specify the processing unit number of the Unit Macro processing unit that executes this process.
This macro function is mainly used for drawing the model figure detected by the search processing unit.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If a value outside the range -2147483648 to 2147483647 is specified as an integer parameter, an "Overflow" error will occur.
If a value outside the range -1.0E30 to 1.0E30 is specified for a real number parameter, an "Overflow" error might occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:

- This macro function can only be used in the *MEASUREDISPI or *MEASUREDISPG subroutine. If used in another subroutine, an "Illegal function call" error will occur.
- Set the line type, color, or width in SetDrawStyle before drawing. If you don't specify these settings, the image is drawn with the previous settings.

Example:

Uses the *MEASUREDISPG subroutine of the Unit Macro processing unit to display a registered model figure for the search processing unit (Processing Unit number 5). The measured X/Y coordinates and radius, and the detection point X/Y coordinates can be gotten with External Reference Data numbers 6, 7, 8, 132, and 133 respectively.
To display the string at the fixed position regardless of the position compensation result, specify the assigned processing unit number to this Unit Macro processing unit (where the *MEASUREDISPG subroutine is used) for the <unitNo> parameter.

```
*MEASUREDISPG

    Dim FIGURE&(255)

    Rem Get the model figure.
    GetUnitFigure 5, 0, FIGURE&()

    Rem Get the measurement result.
    GetUnitData 5, 6, X#
    GetUnitData 5, 7, Y#
    GetUnitData 5, 8, TH#
    GetUnitData 5, 132, RX#
    GetUnitData 5, 133, RY#

    Rem Set the drawing attribute.
    SetDrawStyle PS_SOLID,1,JUDGE_OK

    Rem Draw the image.
    DrawSearchFigure FIGURE&(), RX#, RY#, X#, Y#, TH#, 0, UnitNo

Return
```

Usable Modules:

Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

GetUnitData on page 8-153
SetDrawStyle on page 8-272
UnitNo on page 8-349
Int on page 8-169
UnitData on page 8-341
Ut on page 8-351

DrawText

Draw a character string on the text window.

Format:
DrawText <string>, <color>, <newLine>

Parameter:

Parameter name	Data type	Description
<string>	Character string	Character string to display
<color>	Integer	Color value of character string color to be drawn JUDGE_NC: Unmeasured color (Grey) JUDGE_OK: OK judgement color (Green) JUDGE_NG: NG judgement color (Red) RGB Function: Any color
<newLine>	Integer	Line break after display 0: Do not break a line 1: Break a line

Return value:
None.

Description:
Use the color specified in the <color> parameter and with the line break method specified in the <newLine> parameter to draw the character string specified in the <string> parameter in the text window.
The gotten color value by the RGB function can be set for the <color> parameter. (Refer to *RGB* on page 8-242.)
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
Even if a value that does not exist is specified for a parameter, an error will not occur.
If a value outside the range -2147483648 to 2147483647 is specified as an integer parameter, an "Overflow" error will occur.
An error of "String too long" will be occurred when you specify the character strings as character strings type exceeds 255 characters.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
- This macro function can only be used in the *MEASUREDISPT subroutine. If used in another subroutine, an "Illegal function call" error will occur.
- You can add comments in the program area.

Note that allowable character strings differ depending on the type of the sensor controller as shown below.
English characters and characters for the language selected in *Language setting* are allowed.
Refer to *Comment* on page 4-4.

Example:
Uses the *MEASUREDISPT subroutine of the Unit Macro processing unit to display the provided measurement results by the search processing unit (Processing Unit number 5) in the text window.
The measured X and Y coordinates and angle can be gotten with External Reference Data numbers 6, 7, and 8 respectively.

```
*MEASUREDISPT

   Rem Get the measurement result.
   GetUnitData 5, 6, X#
   GetUnitData 5, 7, Y#
   GetUnitData 5, 8, TH#

   Rem Draw the delimiter after drawing the character string with the "OK" judgeme
nt color without adding any line break.
   DrawText Str2$(X#, 4, 4, 0, 0), JUDGE_OK, 0
   DrawText ", ", JUDGE_OK, 0

   Rem Draw the character string with the "OK" judgement color and add a line brea
k.
   DrawText Str2$(Y#, 4, 4, 0, 0), JUDGE_OK, 1

   Rem Draw the character string with the "OK" judgement color without adding any
line break.
   DrawText Str2$(TH#, 4, 4, 0, 0), JUDGE_OK, 0

Return
```

The result is shown below.

```
123.4567, 10.5000
90.0000
```

Usable Modules:
Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

DrawJudgeText on page 8-97
GetUnitData on page 8-153
RGB on page 8-242
Str$ on page 8-314
Str2$ on page 8-315
UnitData on page 8-341

DrawTextG

Draw a character string on the image window.

Format:
DrawTextG <string>, <x>, <y>, <imageNo>[, <unitNo>]

Parameter:

Parameter name	Data type	Description
<string>	Character string	Character string to display
<x>	Integer	Upper left X coordinate value of the drawn area
<y>	Integer	Upper left Y coordinate value of the drawn area
<imageNo>	Integer	Measurement image number to draw on (always 0)
<unitNo>	Integer	Processing unit number to display the processing unit (0 to (the number of registered processing units in the current scene minus one))

Return value:
None.

Description:
On the measurement image whose specified image number is in the <imageNo> parameter, draw a character string specified in the <string> parameter at the position coordinates specified in the <x> and <y> parameters.
To display a "%" character on the Image window, "%%" should be specified in the <string> parameter.
Specify the corresponding Unit Macro processing unit number in the <unitNo> parameter to draw the image at the position coordinates before applying position compensation. If the <unitNo> parameter is omitted, the image is drawn at the position coordinates after applying position compensation.
In the <X> and <Y> parameters, specify the camera coordinates whose origin is at the upper-left corner of the image.
Normally 0 should be specified in the <imageNo> parameter.
In the <unitNo> parameter, normally specify the processing unit number of the Unit Macro processing unit that executes this process.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If a value outside the range -2147483648 to 2147483647 is specified as an integer parameter, an "Overflow" error will occur.
An error of "String too long" will be occurred when you specify the character strings as character strings type exceeds 255 characters.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
- This macro function can only be used in the *MEASUREDISPI or *MEASUREDISPG subroutine. If used in another subroutine, an "Illegal function call" error will occur.
- You can add comments in the program area.
  Note that allowable character strings differ depending on the type of the sensor controller as shown below.

English characters and characters for the language selected in *Language setting* are allowed. Refer to *Comment* on page 4-4.

Example:

Uses the *MEASUREDISPG subroutine of the Unit Macro processing unit to display the string "OK" at fixed position coordinates (100, 100). To display the string at the fixed position regardless of the position compensation result, specify the assigned processing unit number to this Unit Macro processing unit (where the *MEASUREDISPG subroutine is used) for the <unitNo> parameter.

```
*MEASUREDISPG

   DrawTextG "OK", 100, 100, 0, UnitNo

Return
```

Usable Modules:

Unit Macro

Supported Versions:

Version 3.50 or later

Related Items:

GetUnitData on page 8-153
Str$ on page 8-314
UnitData on page 8-341
Ut on page 8-351
SetTextStyle on page 8-297
Str2$ on page 8-315
UnitNo on page 8-349

DrawUnitImage

Display the "other unit image" on the image window.

Format:
DrawUnitImage <unitNo>, <imageNo>

Parameter:

Parameter name	Data type	Description
<unitNo>	Integer	Processing unit number to display the processing unit (0 to (the number of registered processing units in the current scene minus one))
<imageNo>	Integer	Measurement image number to draw on (Refer to *9-9 Image Number List* on page 9-78.)

Return value:
None.

Description:
Draw the image of the specified image number by the <imageNo> parameter that is held by the specified processing unit number by the <unitNo> parameter.
If a non-existent number, numerical value, or combination of data types or values is specified for the <unitNo> parameter, an "Illegal function call" error will occur.
Even if a value that does not exist is specified for the <imageNo> parameter, an error will not occur.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a value outside the range -2147483648 to 2147483647 is specified as an integer parameter, an "Overflow" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
- This macro function can only be used in the *MEASUREDISPI or *MEASUREDISPG subroutine. If used in another subroutine, an "Illegal function call" error will occur.

Example:
Uses the *MEASUREDISPG subroutine of the Unit Macro processing unit to display image number 1 that is held in advance by the advanced filter processing unit (Processing Unit number 5) and is allocated to output image 1.

```
*MEASUREDISPG

   Rem Display the "other unit image"
   DrawUnitImage 5, 1

Return
```

Usable Modules:
Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

DrawFillImage on page 8-96
DrawMeasureImage on page 8-102
UnitNo on page 8-349
Ut on page 8-351

Dskf

Gets the free space on disk drives.

Format:
Dskf(<driveName>)

Parameter:

Parameter name	Data type	Description
<driveName>	Character string	Drive name whose free space is to be gotten

Return value:
Returns free space (in bytes) on the disk drive as a double-precision real value.
Returns -1 when the specified disk drive does not exist.

Description:
Determinate the free space (in bytes) on the disk drive specified in the <driveName> parameter.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
An error of "String too long" will be occurred when you specify the character strings as character strings type exceeds 255 characters.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Turns ON the ERROR signal when the free space on the disk drive reaches less than KB (1,024 bytes).

```
Rem Check the free space on the disk drive.
DiskSize# = Dskf("E:\")
If DiskSize# < 1024 Then
   Rem ERROR Turn ON the ERROR Signal.
   PutPort "ParallelIo", 103, 1
Endif
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

Fcopy on page 8-129
Isfile on page 8-170
Kill on page 8-176
Mkdir on page 8-207
PutPort on page 8-225
Rmdir on page 8-244

ElapsedTime

Gets the elapsed time since starting the measurement.

Format:
ElapsedTime(<mode>)

Parameter:

Parameter name	Data type	Description
<mode>	Integer	Unit of the elapse time to get 0: ms unit 1: μs unit

Return value:
Returns the elapsed time as an integer value.

Description:
Gets the elapsed time since starting the measurement with the unit specified in the <mode> parameter.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Uses the MEASUREPROC subroutine of the Unit Macro processing unit to get the elapsed time until this macro function is executed. If the elapsed time is 1,000 ms or longer, the error character string is displayed in the text window.

```
*MEASUREPROC

   Rem Get the elapsed time.
   TIME& = ElapsedTime(0)

   Return


*MEASUREDISPT

   Rem If the elapsed time is 1,000 ms or longer, the error character string is di
splayed in the NG color.
   If TIME& > 999 Then
       DrawText "Error", JUDGE_NG, 1
   Endif

Return
```

Usable Modules:
Unit Calculation Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

DrawText on page 8-109
Timer on page 8-327
StartTimer on page 8-311
Wait on page 8-362

Eof

Examines the end of the file.

Format:
Eof(<fileNo>)

Parameter:

Parameter name	Data type	Description
<fileNo>	Integer	File number (0 to 15) of the examined file end.

Return value:
Returns an integer value that notifies if the end of the file is reached.
- 0: The end of the file is reached.
- -1: The end of file is not reached.

Description:
Check if the end of the file of the file number specified in the <fileNo> is reached.
If an unopened file number is specified in the <fileNo> parameter, an "Illegal function call" error will occur.
If a value outside the range of 0 to 15 is specified in the <fileNo> parameter, an "Illegal function call" error will occur.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a value outside the range -2147483648 to 2147483647 is specified as an integer parameter, an "Overflow" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Reads the data until the end of the file.

```
Dim ALLDATA$(255)

Rem Open the file.
Open "E:\input.dat" For Input As #1

For I&=0 to 255
   Rem Read line by line from the top of the file.
   Input #1, DATA$
   ALLDATA$(I&) = DATA$

   Rem Check if the end of the file is reached.
   If Eof(1) <> 0 Then
      Exit For
   Endif
Next

Rem Close up the file.
Close #1
```

Usable Modules:

Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:

Version 3.50 or later

Related Items:

Close on page 8-49
Open For Append As# on page 8-211
Open For Output As# on page 8-213
Input# on page 8-165
Open For Input As# on page 8-212

Erase

Releases array variable.

Format:
Erase <array>[, <array>...]

Parameter:

Parameter name	Data type	Description
<array>	-	Released array variable

Return value:
None.

Description:
Releases the allocated memory area of the predefined array variable with the Dim function that is specified in the <array> parameter. By releasing the temporarily used array variables with this function, the allocated memory areas of the variables can be released so that the released memory areas can be efficiently utilized.
If an array variable is redefined without being released, its allocated memory area is released before the redefinition.
If variables other than array variables are specified in the <array> parameter, "Syntax error" will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Releases defined array.

```
Dim XY&(3)
Dim XY#(7, 15)
Dim CHARA$(31, 63, 127, 255)

Rem Releases array variable.
Erase XY&(), XY#(), CHARA$()
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

Dim on page 8-73

Errcmnd$

Get the function name of the macro where an error occurred.

Format:
Errcmnd$

Parameter:
None.

Return value:
Returns the character string type value of the upper case letters that represents the macro function name where the error occurred.

Description:
Gets the character string of the macro function name where the error occurred at error occurrence in the program.
If there is no error in the program or an error occurred as a result of the unrelated operations to the macro functions such as a division by zero operation, a null string is returned.
Use this macro function in the Try-Catch-End Try statement.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Outputs the error information to the system status console window only if the error is occurred in the GetUnitData statement process in the *MEASUREPROC subroutine within the Unit Macro processing unit. If an error is occurred in other macro function statement in the subroutine, the measurement processing ends without taking any action.

```
*MEASUREPROC

   Try
      GetUnitData 5, 5, CR#

      SetUnitData 6, 143, CR#
   Catch
      If Errcmnd$ = "GetUnitData" Then
         Print Errcmnd$
      Endif
   End Try

Return
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

Errno on page 8-120
Print on page 8-221
Try Catch End Try on page 8-337
GetUnitData on page 8-153
SetUnitData on page 8-300

Errno

Gets the error number.

Format:
Errno

Parameter:
None.

Return value:
Returns the error number as an integer value. (Refer to *9-1 Error List* on page 9-2.)

Description:
Gets the error number of the error which occurred in the program.
Use this macro function in the Try-Catch-End Try statement.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Uses the Try Catch-End Try statement in the *MEASUREPROC subroutine of the Unit Macro processing unit to detect the error occurrence and get the detected error number.

```
*MEASUREPROC

   Try
      WORK& = 0
      SUMM& = 100 + 200 + 300
      ANS& = SUMM& / WORK&
   Catch
      If Errno = 11 Then
         Rem Output the error number and the error content on the system status co
nsole window.
         Print "Error Number = " + Str$(Errno) + ", Division by Zero"
      Endif
   End Try

Return
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

Errcmnd$ on page 8-119
Print on page 8-221

Try Catch End Try on page 8-337

ErrorOut

Sets the output state of the Error(ERROR) signal.

Format:
ErrorOut [<ioIdent>,] <errorKind>

Parameter:

Parameter name	Data type	Description
<ioIdent>	Character string	Identification name of the communication module to be used (Refer to *9-4 List of I/O Modules* on page 9-27.)
<errorKind>	Integer	Select the number of error type to notify. If 0 or a larger number is selected, ERROR output turns ON. -1: ERROR output OFF 0: System error 1: System error (Fan or voltage error) 3: Battery error 4: UWF overlay depletion 10: Camera connection error 11: Connected camera has been changed 12: Detection of camera over current 13: Configuration error of light device connection 20: Write error of logging disk 30: Time out of parallel output 31: PLC link communication error 32: Detection of parallel I/O camera over current 40: Data load error 41: Data transfer error 42: Incorrect number of start-up Scene group 43: Incorrect number of start-up Scene

Return value:
None.

Description:
Set the ERROR signal status of a communication module selected in <ioIdent> parameter. When the EtherCAT communication module is used, set the ERROR signal status and error type number.
All of the communication module will be as an object if you set " "(specify an empty character strings) or omit <ioIdent> parameter.
If 0 or a larger number is set to <errorKind> parameter, ERROR signal turns ON and if -1 is set, ERROR signal turns OFF.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
Type mismatch error is occurred when wrong data is specified as parameter. Illegal function call error is not occurred even if non-exist number, values, combination of data or values.
If a value outside the range -2147483648 to 2147483647 is specified as an integer parameter, an "Overflow" error will occur.
An error of "String too long" will be occurred when you specify the character strings as character strings type exceeds 255 characters.

If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:

- Only action of notified in error message when you write *ErrorProc.
  Normally error processing (Refer to the following example)
  RaiseErrorProcEvent ERRORKIND&, 0, True
  ErrorOut "", ERRORKIND&
  If you write the above parameters, normally error processing can be executed when except specified error is occurred.

Example:

```
Rem Displays a message box when logging error is occurred.
*ERRORPROC

If ERRORKIND& = 20 Then
   MessageBox "error"

Rem Executes normally error processing when an error except logging error is occur
red.
Else
   RaiseErrorProcEvent ERRORKIND&, 0, True
   ErrorOut "", ERRORKIND&
EndIf

Return
```

Usable Modules:

Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:

Version 5.40 or later

Related Items:

MessageBox on page 8-204

RaiseErrorProcEvent on page 8-226

ExecuteErrorProc

Executes the error processing.

Format:

ExecuteErrorProc <errorKind>, <parameter>

Parameter:

Parameter name	Data type	Description
<errorKind>	Integer	Error number to notify the error processing. 0: System error 1: System error (Fan or voltage error) 3: Battery error 4: UWF overlay depletion 10: Camera connection error 11: Connected camera has been changed 12: Detection of camera over current 13: Configuration error of light device connection 20: Write error of logging disk 30: Time out of parallel output 31: PLC link communication error 32: Detection of parallel I/O camera over current 40: Data load error 41: Data transfer error 42: Incorrect number of start-up Scene group 43: Incorrect number of start-up Scene
<parameter>	Integer	Error processing parameter to notify This parameter is not used in this version. Therefore, specify to 0.

Return value:

None.

Description:

Notify a specified error is occurred to system.
Defining your original error number and its error processing to *ErrorProc subroutine of unit macro, you can create your original error processing.
When you use original error number, specify after number 10000.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
Type mismatch error is occurred when wrong data is specified as parameter. Illegal function call error is not occurred even if non-exist number, values, combination of data or values.
If a value outside the range -2147483648 to 2147483647 is specified as an integer parameter, an "Overflow" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:

- Do not write to *MCRINIT.

Example:

Notifies an error to system when acquired character of OCR is 0, and defines the number of original error. Creates an error processing (error message) according to the number of its error.

```
Rem Force an error to be occurred when acquired character of OCR is 0 in Unit1.
●Unit macro
GetUnitData 1, "characterNum0", NUM0&
SetGlobalData "NUM", NUM0&

●Communication command macro1 (CMD0000)
Measure 1

GetGlobalData "NUM", NUM0&

If NUM0& = 0 Then
   ExecuteErrorProc 10000, 0
EndIf

●Communication command macro 2(CMD0001)
*ERRORPROC

Rem Defines an error automatically (10000).
If ERRORKIND& = 10000 Then
   MessageBox "ERROR"
   ErrorOut "", 10000
Else
   Rem Normal error processing is executed when an error is occurred except number
 of error is 10000.
   RaiseErrorProcEvent ERRORKIND&, 0, True
   ErrorOut "", ERRORKIND&
EndIf

Return
```

Usable Modules:

Scene Control Macro / Communication Command Macro

Supported Versions:

Version 5.40 or later

Related Items:

ErrorOut on page 8-121
GetUnitData on page 8-153
RaiseErrorProcEvent on page 8-226
SetUnitData on page 8-300
GetGlobalData on page 8-135
Measure on page 8-193
SetGlobalData on page 8-275

ExecuteImageLogging

Executes image logging.

Format:
ExecuteImageLogging <directory>, <header>

Parameter:

Parameter name	Data type	Description
<directory>	Character string	Subdirectory name that saves the logging images
<header>	Character string	Header string added to the file name of the logged image

Return value:
None.

Description:
Executes image logging and saves the logged images using the file name (composed from the header character string specified in the <header> parameter and a measurement identification) in the subdirectory specified in the <directory> parameter.
Logged images can be saved in the file format (IFZ, BMP(BFZ), JPG(JFZ), PNG(PFZ)) specified in the system settings.
Logged images are saved in the destination folder specified in System setting - Logging setting. (For details, refer to *Setting Logging Conditions [Logging Setting]* in the *Vision System FH/FHV Series User's Manual (Cat. No. Z365).)*
If a subdirectory name is specified in the <directory> parameter, the logged file is saved in a subdirectory created under the logged image files destination folder.
If any empty string "" is specified in the <directory> parameter, the subdirectory is not created.
If a header character string is specified in the <header> parameter, the specified character string is added to the saved file name. If an empty string "" is specified in the <header> parameter, no header character string is added to the file name.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
An error of "String too long" will be occurred when you specify the character strings as character strings type exceeds 255 characters.
If a character string longer than 63 characters is specified in the <directory> parameter, the 63-character string before the 64th character is used for the macro function processing. Characters after the 64th character will be discarded.
If a character string longer than 31 characters is specified in the <header> parameter, the 31-character string before the 32nd character is used for the macro function processing. Characters after the 32nd character will be discarded.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
- This macro function can only be used in the *MEASUREPROC subroutine. If used in any other subroutines, an error will occur and the function will not be executed.

Example:

Uses the MEASUREPROC subroutine in the Unit Macro processing unit to save the logged images using the file names with a header "new_" in the subdirectory named "Image" under the "E:\" directory set as the logged files destination in the system settings.

```
*MEASUREPROC

   Rem Execute image logging.
   ExecuteImageLogging "Image", "new_"

Return
```

After the measurement, a file with file name "new_2012-11-01_13-11-25-0025.ifz" is saved under "E:\Image".

Usable Modules:
Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

GetSystemData on page 8-148
GetUnitData on page 8-153
Str$ on page 8-314
Str2$ on page 8-315

ExitFzProcess

Terminate the sensor controller.

Format:
ExitFzProcess

Parameter:
None.

Return value:
None.

Description:
Terminate the FH process execution.
If this macro function macro is executed on the sensor controller, turn OFF the power to the sensor controller after the execution.

Usage Cautions:
None.

Example:
After saving the data to the controller, terminates the controller.

```
Rem Carry out the 'Data save'.
SaveData

Rem Terminate the sensor controller.
ExitFzProcess
```

Usable Modules:
Scene Control Macro / Communication Command Macro

Supported Versions:
Version 3.50 or later

Related Items:

SaveData on page 8-248
SystemReset on page 8-322

Exp

Gets the value of the exponential function of the basee natural logarithm.

Format:
Exp(<expression>)

Parameter:

Parameter name	Data type	Description
<expression>	Integer Real number	Expression to get the exponential value

Return value:
Returns the calculated exponent (power) as a real number value.

Description:
Calculates the exponent (power) of the base e natural logarithm of the expression specified in the <expression> parameter.
The Exp function is the inverse function of the Log function. The Exp function can be used to derive other mathematical functions, such as the hyperbolic sine function.
In the <expression> parameter, specify a value no greater than 21.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Gets the values of the hyperbolic sine function and hyperbolic cosine function of the value TH&.

```
SINH& = (Exp(TH&)-Exp(-TH&)) / 2
COSH& = (Exp(TH&) + Exp(-TH&)) / 2
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

Cos on page 8-64
Log on page 8-191
UnitData on page 8-341
GetUnitData on page 8-153
Sin on page 8-309

Fcopy

Copies the file.

Format:
Fcopy <srcPath>, <dstPath>

Parameter:

Parameter name	Data type	Description
<srcPath>	Character string	Absolute path of the original file to be copied
<dstPath>	Character string	Absolute path for the copy destination file

Return value:
None.

Description:
Copies the file of the file name specified in the <srcPath> parameter as file of the file name specified in the <dstPath> parameter.
In the <srcPath> parameter and the <dstPath> parameter, specify with the absolute path the copy source file and the file name of the copy destination file.
Overwrite if the copy destination file already exists. If it does not exist, create a new one.
In the following cases, the file cannot be copied.
- The original file to copy from does not exist.
- The destination directory does not exist.
- The external storage has not been inserted.
- There is insufficient free space on the external storage.

If only the directory name is specified in the <srcPath> parameter or the <dstPath> parameter, an "Illegal function call" error will occur.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
An error of "String too long" will be occurred when you specify the character strings as character strings type exceeds 255 characters.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Copies a file named "1280-720.bmp" under the directory "E:\" to the directory "F:\"

```
Fcopy "E:\1280-720.bmp", "F:\1280-720.bmp"
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

Dskf on page 8-114
Kill on page 8-176
Isfile on page 8-170
Mkdir on page 8-207

Rmdir on page 8-244

Fix

Gets the integer of a value by rounding off digits to the right ofthe decimal point.

Format:
Fix(<expression>)

Parameter:

Parameter name	Data type	Description
<expression>	Real number	Expression to truncate after the decimal point

Return value:
Returns an integer value gotten by rounding off digits to the right of the decimal point.

Description:
Gets the integer value of the expression specified in the <expression> parameter by rounding off digits to the right of the decimal point.
If a negative value is specified in the <expression> parameter, the Fix function will return the least negative integer value greater than the specified negative value. This contrasts with the Int function that returns the greatest negative value that does not exceed the specified negative value. For example, Int(-7.2) returns -8 and Fix(-7.2) returns -7.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Changes the real number value of a measurement result to an integer by rounding off digits to the right of the decimal point.

```
NUMBER1& = Fix(9.7)
NUMBER2& = Fix(-9.7)
NUMBER3& = Fix(-9.2)
```

The result is shown below.

```
NUMBER1& = 9
NUMBER2& = -9
NUMBER3& = -9
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

Int on page 8-169
GetUnitData on page 8-153
UnitData on page 8-341

FigureType

Gets the figure type that can be set for the processing unit.

Format:
FigureType(<unitNo>, <figureNo>)

Parameter:

Parameter name	Data type	Description
<unitNo>	Integer	Processing unit number (0 to (the number of registered processing units in the current scene minus one))
<figureNo>	Integer	Figure number to be acquired (Refer to *9-7 List of Figure Numbers* on page 9-73.)

Return value:
Returns figure type information.

Description:
Gets figure type information on the figure specified by the <figureNo> parameter for the processing unit specified by the <unitNo> parameter.
The figure type information to be acquired is the OR of the figure types. For example, if the figure supports rectangle (figure type 8) and circle (figure type 32), 40 will be returned.
Refer to *9-6 Figure Data List* on page 9-70 for figure type information.
If a non-existent unit number is specified, an "Illegal function call" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Confirm that the figure of figure number 1 for processing unit number 2 supports rectangle, and set the region.

```
Rem Gets the figure types that can be set for the target figure.
TYPE& = FigureType(2,1)

Rem If rectangle can be set for the target figure, set the figure.
If (TYPE& AND 8) <> 0 Then
  Dim DATA&(5)
  DATA&(0) = 65560
  DATA&(1) = 8
  DATA&(2) = 0
  DATA&(3) = 0
  DATA&(4) = 639
  DATA&(5) = 479
  SetUnitFigure 2, 1, DATA&()
EndIf
```

Usable Modules:
Unit Calculation Macro/Unit Macro

Supported Versions:
Version 6.40 or later

Related Items:

GetUnitFigure on page 8-154
SetUnitFigure on page 8-301

For To Step Next

Repeats the statements between the For and Next statements.

Format:
For <variable> = <startValue> To <endValue> [Step <increment>]
<statement>
Next [<variable>]

Parameter:

Parameter name	Data type	Description
<variable>	Integer	Loop counter variable of repetition process
<startValue>	Integer	Initial value of loop counter variable
<endValue>	Integer	Loop counter variable value that end up repetition process
<increment>	Integer	Increment of the loop counter variable
<statement>	-	Statement to be executed repeatedly

Return value:
None.

Description:
Repeats the specified For block statement in the <statement> parameter until the loop counter variable specified in the <variable> parameter reaches the <endValue> parameter value. The loop counter value starts from the <startValue> parameter value. Every repeating process increments the loop counter value by the <increment> parameter value.
If the <increment> parameter is omitted, the every repeating process increments the loop counter variable value by one.
If the Exit For statement is used in the For block statement, the statement force stops the repeating operation immediately.
If the program process is jumped into or out of the For block statement using the Goto or Gosub function, the resulting operation may be unpredictable.
If neither the For statement nor the Next statement is used, either the "NEXT without FOR", "FOR without NEXT", or "EXIT without FOR" error will occur depending on the statement that is used.
If the Next statement is not followed by the For statement, the "NEXT without FOR" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:

Uses the *MEASUREPROC subroutine in the Unit Macro processing unit to set the gotten the edge position X values with the edge position processing units (Processing Unit numbers 1 to 4) for the upper limits of measure X of the search processing units (Processing Unit numbers 6 to 9) respectively.

```
*MEASUREPROC

   Dim POS#(3)

   Rem Get the measurement result.
   GetUnitData 1, 5, POS#(0)
   GetUnitData 2, 5, POS#(1)
   GetUnitData 3, 5, POS#(2)
   GetUnitData 4, 5, POS#(3)

   For NUM& = 0 To 3
      Rem Set the setting data.
      SetUnitData NUM&+6, 136, POS#(NUM&)
   Next

Return
```

Usable Modules:

Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:

Version 3.50 or later

Related Items:

Do Loop While on page 8-77
GetUnitData on page 8-153
SetUnitData on page 8-300

GetAll

Gets the input states of all input terminals.

Format:
GetAll(<ioIdent>)

Parameter:

Parameter name	Data type	Description
<ioIdent>	Character string	Identification name of the communication module to be used (always "ParallelIo") (Refer to *9-4 List of I/O Modules* on page 9-27.)

Return value:
Returns the input states of all input terminals as integer values.
The input state of each input terminal is expressed as an integer value (OFF (0) or ON (1)) in each digit of a character string in binary notation.
In parallel I/O, integer values are returned expressing DI0 to DI7 in the 1st digit to the 8th digit.

- Example: When DI0 to DI5 are ON and DI6 to DI7 are OFF
  Binary notation: 0011 1111
  Value of input states that can be gotten: 63

Description:
Gets the input states of all input terminals of the communication module specified in the <ioIdent> parameter.
Normally "ParallelIo" should be specified in the <ioIdent> parameter.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
In the communication command macro, gets the input state of DI in parallel I/O.

```
IOMODULE$ = "ParallelIo"

Rem Get the input state.
STATE& = GetAll(IOMODULE$)
```

Usable Modules:
Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

BusyOut on page 8-37
JudgeOut on page 8-175
GetPort on page 8-144
PutAll on page 8-223

PutPort on page 8-225 *RunOut* on page 8-245

GetGlobalData

Gets the global data.

Format:
GetGlobalData <dataIdent>, <data>

Parameter:

Parameter name	Data type	Description
<dataIdent>	Character string	Identification name of the global data to get
<data>	Integer Real number Character string	Gotten data

Return value:
None.

Description:
Gets the value of the global data that has the identification name specified in the <dataIdent> parameter. The value of the gotten global data is converted to the specified variable type and stored in the <data> parameter. If a character string that cannot be converted to a numerical value is gotten and an integer or real type variable is specified for the <data> parameter, 0 is stored in the gotten data.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If a character string longer than 255 characters is specified in the <dataIdent> parameter, a "String too long" error will occur.
If a character string longer than 255 characters is specified in the <dataIdent> parameter, a "String too long" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Gets the integer value set in the global data that has the identification name "ABC".

```
Rem Set the integer value 1 in the "ABC" global data value.
SetGlobalData "ABC", 1

Rem Get the value of the global data "ABC" and store in the integer variable DATA&
.
GetGlobalData "ABC", DATA&
```

Gets the value of the current layout number store in "LayoutNoLocal_IoCommand" of the reserved global data words.

```
Rem Get the value of the global data "LayoutNoLocal_IoCommand".
GetGlobalData "LayoutNoLocal_IoCommand", DATA&
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

AddGlobalData on page 8-26
SetGlobalData on page 8-275

GetImageSize

Gets the image size of the processing unit image.

Format:
GetImageSize <unitNo>, <measureImageNo>, <sizeX>, <sizeY>

Parameter:

Parameter name	Data type	Description
<unitNo>	Integer	Processing unit number (0 to (the number of registered processing units in the current scene minus one))
<measureImageNo>	Integer	Image number of the current unit to be acquired (Refer to *9-9 Image Number List* on page 9-78, *9-10 List of Sub-Image Numbers* on page 9-82)
<sizeX>	Integer	X size of the gotten image
<sizeY>	Integer	Y size of the gotten image

Return value:
None.

Description:
Gets the size of the image data of the image number specified in the <measureImageNo> parameter, held by the processing unit specified in the <unitNo> parameter.
If a camera image input (Camera Image Input FH, Camera Image Input HDR, Camera Image Input HDR Lite, Photometric Stereo Image Input, etc.) processing item in the <unitNo> parameter, an "Illegal function call" error will occur.
In <sizeX> parameter and <sizeY> parameter, specify variables to store the gotten image size.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
- Use this macro function after executing the following procedures while the measurement image is displayed.
  - Measure the image once, or more.
  - Specify the image file and re-measure it.

Example:

Gets the size of the image of Image number 0 in Processing Unit number 2.

```
GetImageSize 2, 0, SIZEX&, SIZEY&
```

Usable Modules:

Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:

Version 3.50 or later

Related Items:

SaveMeasureImage on page 8-250
UnitNo on page 8-349
Ut on page 8-351

GetImageWindow

Get the state of the image window.

Format:

[Scene Control Macro / Communication Command Macro]

GetImageWindow <windowNo>, <locationX>, <locationY>, <width>, <height>, <unitNo>, <subNo>, <magnification>, <originX>, <originY>, <update>, <visible>

[Unit Macro]

GetImageWindow <locationX>, <locationY>, <width>, <height>, <unitNo>, <subNo>, <magnification>, <originX>, <originY>, <update>, <visible>

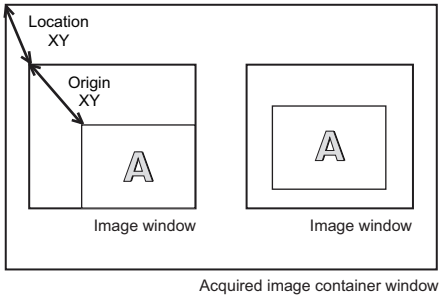
Parameter:

Parameter name	Data type	Description
<windowNo>	Integer	Image window number to get the status
<locationX>	Integer	Upper left X coordinate value of the image window
<locationY>	Integer	Upper left Y coordinate value of the image window
<width>	Integer	Width of the image window
<height>	Integer	Height of the image window
<unitNo>	Integer	Processing unit number of the target processing unit to display
<subNo>	Integer	Sub-image number of the target image to display
<magnification>	Real number	Display magnification
<originX>	Integer	Upper left X coordinate of the image display relative to the upper left coordinate of the image window.
<originY>	Integer	Upper left Y coordinate of the image display relative to the upper left coordinate of the image window
<update>	Integer	Update timing of image window 0: Every measurement (Image mode Freeze) 1: Only when an overall judgement result is NG at the time of measurement (Last NG image). 2: Only when a target processing unit is NG at the time of measurement. 3: Always updated (through display)

Parameter name	Data type	Description
<visible>	Integer	Setting of whether to display 0: Window invisible 1: Window visible

Return value:
None.

Description:
Gets the state of the image window specified in the <windowNo> parameter. When this macro function is used with the unit macro, the state of the image window displayed using the MEASUREDISPI subroutine is set.
In the <locationX> parameter and <locationY> parameter, specify the variables that store the relative coordinate values from the upper left coordinates of the gotten image container window to the upper left coordinates of the image window.
In the <width> parameter and <height> parameter, specify the variables that store the values of the gotten image window width and height.
In the <unitNo> parameter, specify the variable that stores the value of the gotten processing unit number that is displayed. When the processing unit displayed in the image window is linked to flow display, -1 is stored.
In the <subNo> parameter, specify the variable that stores the value of the gotten sub image number that is displayed. When the content displayed in the image window is the position list, -1 is stored.
In the <magnification> parameter, specify the variable that stores the value of the display zoom of the gotten image window. When the display zoom is auto, -1 is stored.
In the <originX> parameter and the <originY> parameter, specify the variables that store the values of the relative coordinates from the upper left coordinates of the gotten image window to the upper left coordinates of the displayed image.



In the <update> parameter, specify the variable that stores the value of the image mode of the gotten image window.
In the <visible> parameter, specify the variable that stores the value of the display state of the gotten image window.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
[Unit Macro]
- This macro function can only be used in the *MEASUREDISPI or *MEASUREDISPG subroutine. If used in another subroutine, an "Illegal function call" error will occur.

Example:

In the communication command macro, changes the image mode of image windows 0 to 3 to Through. Setting the BusyOn flag to ON in advance in the communication command macro.

```
For I& = 0 To 3
   Rem Get the state of the image window.
   GetImageWindow I&, LOCATIONX&, LOCATIONY&, WIDTH&, HEIGHT&, UNITNO&, SUBNO&, MA
G#, ORIGINX&, ORIGINY&, UPDATE&, VISIBLE&

   Rem Change the update timing to Through.
   UPDATE& = 3

   Rem Set the state of the image window.
   SetImageWindow I&, LOCATIONX&, LOCATIONY&, WIDTH&, HEIGHT&, UNITNO&, SUBNO&, MA
G#, ORIGINX&, ORIGINY&, UPDATE&, VISIBLE&

Next
```

Usable Modules:
Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

DisplayUnitNo on page 8-76
SetDisplayUnitNo on page 8-271
UnitNo on page 8-349
GetTextWindow on page 8-152
SetImageWindow on page 8-276
Ut on page 8-351

GetMeasureOut

Gets the external output setting for measurement results.

Format:
GetMeasureOut

Parameter:
None.

Return value:
Returns the external output setting as an integer value.
- 0: Not output externally
- 1: Output externally

Description:
Gets the *External output* setting in the layout settings as the external output setting for measurement results. (For details, refer to *Setting the Behavior of Output Signals for Each Layout (Layout Settings)* in the *Vision System FH/FHV Series User's Manual (Cat. No. Z365).)*
Even when the measurement result external output setting is 0, data can be output using the SendData function or SendString function in the macro customize function.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.

Usage Cautions:

- Execute this macro function when the BUSY signal or other measurement in progress signal is ON and measurement is prohibited. (For details, refer to *5-1-4 State Transitions and Execution Timing* on page 5-13.)

Example:
In the scene control macro, outputs the measurement results to an external device when *External output* is ON, and outputs the measurement results to the system status console window when OFF.

```
Rem Get the read character string of the 2D code processing unit of Processing Uni
t number 3.
GetUnitData 3, "decodeCharStr", RESULT$

Rem Branch the processing based on the external output setting.
If GetMeasureOut = 1 Then
   SendString "TcpNormal", RESULT$
Else
   Print RESULT$
EndIf
```

Usable Modules:
Scene Control Macro / Communication Command Macro

Supported Versions:
Version 3.50 or later

Related Items:

SendData on page 8-268
SendString on page 8-270
SetMeasureOut on page 8-280

GetPlcData

Gets data read with the ReadPlcMemory function.

Format:
GetPlcData <ioIdent>, <readData()>, <offset>, <size>, <data>

Parameter:

Parameter name	Data type	Description
<ioIdent>	Character string	Identification name of the communication module to be used (Refer to *9-4 List of I/O Modules* on page 9-27.)
<readData()>	Integer array	Loaded data
<offset>	Integer	Offset from the top of the loaded data to the head of the loaded data to read in (byte unit)
<size>	Integer	Size of the data to get (byte unit)
<data>	Integer Real number Character string	Gotten data

Return value:
None.

Description:

Using the communication module specified in the <ioIdent> parameter, the data size specified in the <size> parameter is gotten from the position that is offset by the value specified in the <offset> parameter from the start of the data array specified in the <readData> parameter.
Use this macro function to get the data after executing the ReadPlcMemory function to read the value in the PLC memory area.
In the <readData()> parameter, specify the 1D integer array variable that stores the data read with the ReadPlcMemory function. Add () without specifying element numbers.
In the <offset> parameter and <size> parameter, specify the offset and size in units of bytes. These units are different from the units specified in the ReadPlcMemory function (channel units).
Specify 2, 4, or 8 in the <size> parameter. These will respectively get a 2 byte integer, 4 byte integer, or 8 byte real number.
In the <data> parameter, specify the variable that will store the gotten data.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:

- If the value in the PLC memory area is read using the ReadPlcMemory function in PLC link communication, always use this macro function to get the value from the data that is read. If the value is directly gotten from the ReadPlcMemory function parameter without using this macro function, the correct value may not be gotten.

Example:

In the communication command macro, reads multiple data from the PLC connected by PLC link.

```
IOMODULE$ = "UdpPlcLink"

Rem Get the settings of the output data area.
GetSystemData IOMODULE$, "outputArea", AREA&
GetSystemData IOMODULE$, "outputMemoryAddress", ADDRESS&

Rem Create the integer array variable to store the read data.
Dim DATA&(1)

Rem Load the data (4ch) from data output area.
ReadPlcMemory IOMODULE$, AREA&, ADDRESS&, 4, DATA&()

Rem Get the values from the read data.
GetPlcData IOMODULE$, DATA&(), 0, 4, VALUE0&
GetPlcData IOMODULE$, DATA&(), 4, 4, VALUE1&
```

Usable Modules:

Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:

Version 4.20 or later

Related Items:

ReadPlcMemory on page 8-229
SetPlcData on page 8-281
WritePlcMemory on page 8-363

GetPollingState

Gets the polling state of the communication module.

Format:
GetPollingState(<ioIdent>)

Parameter:

Parameter name	Data type	Description
<ioIdent>	Character string	Identification name of communication module whose polling state is to be gotten (Refer to *9-4 List of I/O Modules* on page 9-27.)

Return value:
Returns the polling state as an integer value.
- False: Stopped
- True: Operating

Description:
Gets the polling state of the communication module specified in the <ioIdent> parameter.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Receives normal TCP communication data in the *MEASUREPROC subroutine of the Unit Macro processing unit.

```
Rem Prepare a buffer that can receive 12 bytes of data.
Dim BUFFER&(11)
IOMODULE$ = "TcpNormal"

Rem Set the polling state of the communication module to stopped in order to recei
ve the data.
SetPollingState IOMODULE$, False

Rem Executing the initialization of the reception data size.
SIZE& = 0

Rem Repeat the reception process until the data has been received.
Try
   Do
      Rem Attempting the data reception.
      ReceiveData IOMODULE$, BUFFER&(), 12, SIZE&

      Rem Once the data has been received, display the data size in the system sta
tus console window.
      If(SIZE& > 0) Then
         Print "Received data size = " + Str$(SIZE&)
      Endif
   Loop While SIZE& = 0

   Rem Data has been received, so set the polling state of the communication modul
e to running.
   SetPollingState IOMODULE$, True

Catch
   Rem Return the polling state of the stopped communication module to running.
   If GetPollingState(IOMODULE$) = False Then
      SetPollingState IOMODULE$, True
   Endif

End Try
```

Usable Modules:

Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:

Version 4.20 or later

Related Items:

SendData on page 8-268
SendString on page 8-270
SetPollingState on page 8-283
ReceiveData on page 8-231

GetPort

Gets the input state of the specified input terminal.

Format:
GetPort(<ioIdent>, <portNo>)

Parameter:

Parameter name	Data type	Description
<ioIdent>	Character string	Identification name of the communication module to be used (always "ParallelIo") (Refer to *9-4 List of I/O Modules* on page 9-27.)
<portNo>	Integer	Terminal number of input terminal whose input state is to be gotten. Parallel I/O: • FH DI0 to DI7: 0 to 7 DSA N: 100 + N x 8 (N: Line number (0 to 7)) STEP N: 101 + N x 8 DILINE0 to DILINE2: 200 to 202

Return value:
Returns the input state of the input terminal as an integer value.
- 0: Input OFF state
- 1: Input ON state

Description:
Gets the state of the input terminal of the terminal number specified in the <portNo> parameter of the communication module specified in the <ioIdent> parameter.
Normally "ParallelIo" should be specified in the <ioIdent> parameter.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
In the communication command macro, gets the input state of DI7 of parallel I/O.

```
IOMODULE$ = "ParallelIo"

Rem Get the input state.
STATE& = GetPort(IOMODULE$, 7)
```

Usable Modules:
Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

BusyOut on page 8-37
JudgeOut on page 8-175
PutPort on page 8-225
GetPort on page 8-144
PutAll on page 8-223
RunOut on page 8-245

GetProfileData

Gets the data of the setting file (.ini file).

Format:
GetProfileData <fileName>, <section>, <key>, <defaultData>, <data>
GetProfileData <fileName>, <section>, <key()>, <data()>

Parameter:

Parameter name	Data type	Description
<fileName>	Character string	Absolute path to the file
<section>	Character string	The section name
<key>, <key()>	Character string	The Key name
<defaultData>	Character string	Data that cannot be acquired
<data>, <data0>	Character string	Acquired data

Return value:
None.

Description:
The data corresponding to the section name specified by the <section> parameter and the key name specified by the <key> parameter of the file specified by the <fileName> parameter is stored in the <data> parameter.
If a nonexistent file name is specified as the <fileName> parameter, an Illegal function call error will occur.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Gets the data with the Key name "Logging" in the "Setting" section of the .ini file and set it as system data.

```
GetProfileData "E:\Setting.ini", "Setting", "Logging", "E:\tmp", Logging$
SetSystemData "Logging","imageLoggingDirectory", Logging$
```

Usable Modules:
Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 6.40 or later

Related Items:

DeleteProfileData on page 8-71 *SetProfileData* on page 8-285

GetSceneData

Gets data related to the scene control macro or scene variables.

Format:
GetSceneData<dataIdent>,<data>

Parameter:

Parameter name	Data type	Description
<dataIdent>	Character string	Identification name of data to be gotten
<data>	Integer Real number Character string	Gotten data

Return value:
None.

Description:
In the <dataIdent> parameter, specify the variable name to be used in the scene control macro program or the scene variable name. The scene variable name is a character string following "SC." displayed on the scene variable registration screen of TDM editor.
Gets the data identified by the identification name specified in the <dataIdent> parameter.
In the <data> parameter, specify the variable that will store the gotten data.
If the data type of the data to be gotten is different from the data type of the variable specified in the <data> parameter, the gotten data will be converted to the data type of the <data> parameter.
If an identification name that does not exist is specified as the parameter, an "Illegal function call" error will occur.
Even if a combination of different data types is specified as parameters, an error will not occur.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
- Execute this macro function when the BUSY signal or other measurement in progress signal is ON and measurement is prohibited. (For details, refer to *5-1-4 State Transitions and Execution Timing* on page 5-13.)

Example:
Substitute the value for a scene variable to an arbitrary variable.

```
Rem When substituting the value stored in the SC.ABC& to TMP& with integer type.
GetSceneData "ABC&", TMP&

Rem When substituting the value in the first array element in the SC_ASDF&() array
 variable of the scene variable
Rem to TMP& with integer type.
GetSceneData "ASDF&(0)", TMP&
```

Uses the unit macro to get the integer variables "SearchResult0&" and "SEARCHRESULT1&" defined in the communication command macro.

```
Rem Get the value of the variable that has been defined in the scene control macro
.
GetSceneData "SEARCHRESULT0&", RET0&
GetSceneData "SEARCHRESULT1&", RET1&

Rem Calculate the result based on the gotten variable value.
RESULTDATA& = (RET0& + RET1&) / 2
```

Usable Modules:

- When applying to scene variables:
  Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro
- When applying to other than scene variables:
  Communication Command Macro

Supported Versions:

Version 5.20 or later
(For scene variables, Version 6.30 or later)

Related Items:

SetSceneData on page 8-286

GetSceneGroupData

Gets the scene group data with the specified identification name.

Format:

GetSceneGroupData <sceneGroupNo>, <dataIdent>, <data>

Parameter:

Parameter name	Data type	Description
<sceneGroupNo>	Integer	Scene group number
<dataIdent>	Character string	Identification name of scene group data "sceneGroupTitle": Gets the scene group title in the character string variable specified in <data>. The scene group titles can be specified in multilingual format using language identifier. For the description method using language identifier, refer to the *Inputting text* in the *Vision System FH/FHV Series User's Manual (Cat. No. Z365)*.
<data>	Character string	Gotten data

Return value:

None.

Description:

Gets the data corresponding to the identification name specified by the <dataIdent> parameter for the scene group specified by the <sceneGroupNo> parameter. For the <data> parameter, specify a variable that stores the acquired data.

If an identification name that does not exist is specified as the parameter, an "Illegal function call" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:

- Execute this macro function when the BUSY signal or other measurement in progress signal is ON and measurement is prohibited. (For details, refer to *5-1-4 State Transitions and Execution Timing* on page 5-13.)

Example:

Displays the title of the current scene group in the System status monitoring console window.

```
MeasureStop
  GetSceneGroupData -1, "sceneGroupTitle", TITLE$
  Print LanguageString$(TITLE$)
MeasureStart
```

Usable Modules:

Scene Control Macro/Communication Command Macro

Supported Versions:

Version 6.50 or later

Related Items:

MeasureStart on page 8-200
MeasureStop on page 8-201
SetSceneGroupData on page 8-289

GetSystemData

Gets the system data or system variable.

Format:

GetSystemData <dataIdent0>, <dataIdent1>, <data>

Parameter:

Parameter name	Data type	Description
<dataIdent0>	Character string	Data identification name of identification information 0 of system data to be gotten
<dataIdent1>	Character string	Data identification name of identification information 1 of system data to be gotten
<data>	Integer Real number Character string	Value of the gotten system data

Return value:

None.

Description:

- When getting values of the system data:
  Gets the system data of identification information 1 specified in the <dataIdent1> parameter, which belongs to identification information 0 specified in the <dataIdent0> parameter. In the <data>

parameter, specify the variable that will hold the gotten system data. For the identification information list, refer to *9-3 System Data List* on page 9-10.

- When getting values of system variables:
  Set a character string for "macro_sy" to <dataident0>, and the variable name following "SY." displayed on the system variable registration screen of TDM editor to <dataident1>. In the <data> parameter, specify the variable that will hold the gotten system data.

If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If an identification name that does not exist is specified as the parameter, an "Illegal function call" error will occur.
If a character string longer than 255 characters is specified in the <dataIdent1> parameter, a "String too long" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:

- In version 6.40 or later, the absolute path of RAMDisk and USBDisk cannot be get because it is registered with the **Quick access setting tool**. Use the AbsolutePath$ function to get their absolute paths.
- If "MultiLineRandom" is specified for the <dataIdent0> parameter, the system data value cannot be get correctly from line 0.

Example:

Substitute the value for a system variable in an arbitrary variable.

```
Rem When substituting the value stored in the SY.ABC& of the system variable to TM
P& with integer type.
GetSystemData "macro_sy", "ABC&", TMP&

Rem When substituting the value of the first array element in the SY.ASDF&() of th
e system variable
Rem to TMP& with integer type.
GetSystemData "macro_sy", "ASDF&(0)", TMP&
```

Gets the value set for the screen capture destination folder of identification information 1, "captureDirectory", which belongs to the measurement control settings of identification information 0, "Measure", and copies the Sample.bmp file to "E:\temp\bmp".

```
Rem Get the screen capture destination folder that belongs to the measurement cont
rol settings.
GetSystemData "Measure", "captureDirectory", DIRNAME$

Rem Get the file name, including the copy destination path, of the file to be copi
ed.
FILE$ = DIRNAME$ + "\Sample.bmp"

Rem Copy the file.
Fcopy FILE$, "E:\temp\bmp\Sample.bmp"
```

Usable Modules:

Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:

Version 3.50 or later

(For scene variables, Version 6.30 or later)

Related Items:

AbsolutePath$ on page 8-25
Fcopy on page 8-129
GetUnitData on page 8-153
AddSystemData on page 8-27
GetGlobalData on page 8-135
SetSystemData on page 8-295

GetText$

Get a text data from a messages file.

Format:
GetText$(#<textDataNo>, <textIdent>)

Parameter:

Parameter name	Data type	Description
<textDataNo>	Integer	Text data number of the messages file (0 to 15) that contains the gotten text
<textIdent>	Character string	Identification name of the text data to be gotten

Return value:
Returns an gotten text data value in character string type.

Description:
- Message file defines the displayed messages. The message file is configured one file for each languages.
- The message file name is configured as the below.
  <Message file Data ident>_<Language Data ident>.msg
  <Message file Data ident> indicates an ident specified <ident> in OpenTextData function.
  <Language Data ident> is a character string.
  Each languages and its Data ident are as the following:
  Simplified Chinese: chs
  Traditional Chinese: cht
  German: deu
  English: eng
  Spanish: esp
  French: fra
  Italian: ita
  Japanese: jpn
  Korean: kor
- A Data ident specified <textIdent> for each lines and a text data which is corresponded to <textIdent>.
  <Data ident 1> = <Text data1>
  <Data ident 2> = <Text data 2>
  <Data ident 3> = <Text data 3>
  :
  <Data ident n> = <Text data n>

Get the text data of the identification name specified in the <textIdent> parameter, from the message file of the text data number specified in the <textDataNo> parameter.

In the <textDataNo> parameter, specify the speficied text data number in the OpenTextData function that has been used to open the message file.
If a value outside the range from 0 to 15 is specified in the <textDataNo> parameter, an "Illegal function call" error will occur
If the text data number that has not been opened is specified in the <textDataNo> parameter, an "Illegal function call" error will occur.
Even if an identification name that does not exist is specified in the <textIdent> parameter, an error will not occur. If an identification name that does not exist is specified for a parameter, a text string of "#ERROR" will be returned.
An error of "String too long" will be occurred when you specify the character strings as character strings type exceeds 255 characters.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Uses the *MEASUREDISPT subroutine of the Unit Macro processing unit to display the measured correlation value by the search processing unit (Processing Unit number 5), along with the gotten text string from the prepared message file for the processing unit, in the text window. The correlation value can be gotten with External Reference Data number 5.

```
*MEASUREDISPT

   Rem Get the measurement result.
   GetUnitData 5, 5, CR#

   Rem Open the messages file
   OpenTextData "Search" As #1

   Rem Get the text
   TEXT$ = GetText$(#1, "Correlation")

   Rem Draw the gotten text string from the messages file without adding any line
break on the text window.
   DrawText TEXT$, UnitJudge(5), 0

   Rem Draw the measurement results on the text window.
   DrawText Str2$(CR#, 4, 4, 0, 0), UnitJudge(5), 1

   Rem Close up the messages file.
   CloseTextData

Return
```

The result is shown below.

```
Correlation value: 90.0000
```

Usable Modules:
Unit Calculation Macro / Unit Macro

Supported Versions:

Version 5.00 or later

Related Items:

CloseTextData on page 8-50
GetUnitData on page 8-153
UnitJudge on page 8-348
DrawText on page 8-109
OpenTextData on page 8-215

GetTextWindow

Gets the state of the text window.

Format:
GetTextWindow <unitNo>, <subNo>, <update>, <visible>

Parameter:

Parameter name	Data type	Description
<unitNo>	Integer	Processing unit number of the target processing unit to display
<subNo>	Integer	Sub number of the target image to display
<update>	Integer	Updated timing (always 0)
<visible>	Integer	Setting of whether to display 0: Window invisible 1: Window visible

Return value:
None.

Description:
Gets the state of the text window.
In the <unitNo> parameter, specify the variable that stores the value of the gotten processing unit number that is displayed. When the processing unit displayed in the text window is linked to flow display, -1 is stored.
In the <subNo> parameter, specify the variable that stores the value of the gotten sub image number that is displayed.
In the <update> parameter, specify the variable that stores the value of the gotten update timing.
In the <visible> parameter, specify the variable that stores the display state of the gotten text window.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
In the communication command macro, changes the processing unit number of the processing unit displayed in the text display window to the number specified in the communication command argument.

```
Rem Get the state of the text window.
GetTextWindow UNITNO&, SUBNO&, UPDATE&, VISIBLE&

Rem Set the number specified in the command argument in the processing unit number
 that is displayed.
SetTextWindow argumentValue#(0), SUBNO&, UPDATE&, VISIBLE&
```

Usable Modules:
Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

DisplayUnitNo on page 8-76
SetDisplayUnitNo on page 8-271
UnitNo on page 8-349
GetImageWindow on page 8-137
SetTextWindow on page 8-299
Ut on page 8-351

GetUnitData

Gets the data of a processing unit.

Format:
GetUnitData <unitNo>, <dataNo>, <data>
GetUnitData <unitNo>, <dataIdent>, <data>

Parameter:

Parameter name	Data type	Description
<unitNo>	Integer	Processing unit number (0 to (the number of registered processing units in the current scene minus one))
<dataNo>	Integer	External reference data of the processing unit data to get (For details, refer to *Vision System FH/FHV Series Processing Item Function Reference Manual (Cat. No. Z341)*.)
<dataIdent>	Character string	Data identification name of processing unit data to be gotten.
<data>	Integer Real number Character string	Gotten processing unit data

Return value:
None.

Description:
Gets the data of the external reference data number specified in the <dataNo> parameter, held by the processing unit specified in the <unitNo> parameter. In the <data> parameter, specify the variable that will store the gotten data. The data can also be gotten by specifying the <dataIdent> parameter instead of the <dataNo> parameter.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Gets the judgement result of the processing unit of Processing Unit number 2. The judgement result is external reference data number 0 and external reference data identification name "JG".

```
GetUnitData 2, 0, JUDGE&

Rem The same result can be gotten by specifying "JG" instead of 0.
GetUnitData 2, "JG", JUDGE&
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

GetUnitFigure on page 8-154
SetUnitFigure on page 8-301
UnitData$ on page 8-342
UnitNo on page 8-349
SetUnitData on page 8-300
UnitData on page 8-341
UnitData2 on page 8-343
Ut on page 8-351

GetUnitFigure

Gets figure data to the processing unit.

Format:
GetUnitFigure <unitNo>, <figureNo>, <figure()>

Parameter:

Parameter name	Data type	Description
<unitNo>	Integer	Processing unit number (0 to (the number of registered processing units in the current scene minus one))
<figureNo>	Integer	Figure data to get (Refer to *9-7 List of Figure Numbers* on page 9-73.)
<figure()>	Integer array	Gotten figure data (Refer to *9-7 List of Figure Numbers* on page 9-73.)

Return value:
None.

Description:
Gets the figure data of the figure specified in the <figureNo> parameter, of the processing unit specified in the <unitNo> parameter.
In the <figure()> parameter, specify the 1D integer array variable that will hold the figure data by adding only () without specifying an element number.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.

If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Changes the value of wide arc line thickness if the figure of the edge processing unit of Processing Unit number 2 is a wide arc

```
Dim FIGURE&(10)

Rem Get the figure data of the processing unit.
GetUnitFigure 2, 0, FIGURE&()

Rem Case of width of the wide arc.
If FIGURE&(1) = 256 Then
   Rem Set the thickness value.
   FIGURE&(7) = 64
   SetUnitFigure 2, 0, FIGURE&()
Endif
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

FigureType on page 8-131
GetUnitData on page 8-153
SetUnitData on page 8-300
SetUnitFigure on page 8-301
UnitData on page 8-341
UnitNo on page 8-349
Ut on page 8-351

Gosub

Operate the specified subroutine.

Format:
Gosub <label>

Parameter:

Parameter name	Data type	Description
<label>	Character string	Label name for the executing subroutine

Return value:
None.

Description:
Execute the subroutine whose label name is specified in the <label> parameter.
At the Return statement in the end of a subroutine, the program resumes the operation from the next statement of the Gosub statement in the calling subroutine.
If the Return statement will not resume the operation in the original subroutine, use the Goto function.
If the specified label does not exist for a parameter, an "Undefined label" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Uses the *MCRINIT subroutine in the Unit Macro processing unit to execute the process in the *INITPROC subroutine (defined separately from *MCRINIT).

```
*MCRINIT

   Rem Execute another subroutine.
   Gosub *INITPROC

Return

*INITPROC

   Dim DATA$(255)

Return
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

Goto on page 8-157

Goto

Moves the process to the statement line with a specified label.

Format:
Goto <label>

Parameter:

Parameter name	Data type	Description
<label>	Integer Character string	Move destination line number or move destination label name

Return value:
None.

Description:
Move the process to the specified line number or label name in the <label> parameter.
To return to the calling subroutine where the Return statement is used, use the Gosub function.
If the specified label does not exist for a parameter, an "Undefined label" error will occur.
If the specified line number does not exist for a parameter, an "Undefined line number" error will occur.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Uses the *MEASUREPROC subroutine in the Unit Macro processing unit to move the programming process to the *PROC2 labeled line and output the text string to the system status console window. If the first Goto statement is rewritten to jump the process to the *PROC1 labeled line or the *PROC3 labeled line, the process ends without taking any action. Similarly, if the first Goto statement is skipped, the process moves to the *PROC3 labeled line and the process ends without taking any action.

```
*MEASUREPROC

    Rem Move the process to another line
    Goto *PROC2

    *PROC1
    Goto *PROC3

    *PROC2
    Print "PROC1"

    *PROC3

Return
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:

Version 3.50 or later

Related Items:

Gosub on page 8-156

Hex$

Converts the value in the expression to the hexadecimal value in character string format.

Format:
Hex$(<expression>)

Parameter:

Parameter name	Data type	Description
<expression>	Integer	Expression to be converted to a character string

Return value:
Returns the character string type hexadecimal value.
If the value in the expression is negative, the two's complement hexadecimal value is returned.
(&H) to express a hexadecimal number is not added to the return value.

Description:
Converts the value specified in the <expression> parameter to the hexadecimal value in character string format.
To specify a real number type expression in the <expression> parameter, convert the expression to the integer type with Fix or Int in advance.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a value outside the range -2147483648 to 2147483647 is specified as an integer parameter, an "Overflow" error will occur.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Gets the read letter "A" with the OCR processing unit (Processing Unit number 5) using the unit calculation macro processing unit and converts the letter into a hexadecimal value. The read character string can be gotten with External Reference Data number 20.

```
Rem Get the measurement result of the processing unit.
GetUnitData 5, 20, CHARA1$

Rem Convert the character to the character code.
CODE& = Asc(CHARA1$)

Rem Convert the character code to the hexadecimal value in character string format
.
CHARA2$ = Hex$(CODE&)
```

The result is shown below.

```
CHARA1$ = "A"
CODE& = 65
CHARA2$ = 41
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

Asc on page 8-34
GetUnitData on page 8-153
Left$ on page 8-180
Mid$ on page 8-205
Right$ on page 8-243
Str2$ on page 8-315
Val on page 8-352
Chr$ on page 8-44
LCase$ on page 8-179
Len on page 8-181
Piece$ on page 8-219
Str$ on page 8-314
UCase$ on page 8-339

If Then Else

Controls the process flow according to the specified condition.

Format:
If <expression> Then <statement>|<label>[Else <statement>|<label>]

Parameter:

Parameter name	Data type	Description
<expression>	-	Logical expression that controls the process flow. (Refer to *4-1-5 Operator* on page 4-12.)
<statement>	-	Processed statement
<label>	Character string	Label name for the jump destination

Return value:
None.

Description:
If the specified condition by the <expression> parameter is true, the specified If block statement by the <statement> parameter is executed. If a label name is specified in the "Then" part statement, the process is jumped to the line with a label name specified in the <label> parameter. If the condition specified in the <expression> parameter is false, the specified Else block with the <statement> parameter in the "Else" part statement is executed. In the similar way as the "Then" part, if a label name is specified in the "Else" part statement, the process is jumped to the line with a label name specified in the <label> parameter.
If the "Else" part statement is omitted, the process flow is controlled only when the specified condition by the <expression> parameter is true.
The Else statement line cannot be broken into multiple parts. Write the If-Else statement in a single line.

If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Uses the *MEASUREPROC subroutine in the Unit Macro processing unit to branch the processing depending on the gotten judgement result by the Processing Unit number 0.

```
*MEASUREPROC

   If UnitJudge(0)=JUDGE_OK Then Gosub *OKOUT Else Gosub *NGOUT

Return

*OKOUT
   Print "OK"
Return

*NGOUT
   Print "NG"
Return
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

Gosub on page 8-156
Print on page 8-221
Select Case Case Else End Select on page 8-266
If Then Elseif Else EndIf on page 8-160
UnitJudge on page 8-348

If Then Elseif Else EndIf

Controls the process flow according to the specified condition.

Format:
If <expression> Then
<ifStatement>
[Elseif <expression> Then
<elseifStatement>]
:
:
[Else
<elseStatement>]
EndIf

Parameter:

Parameter name	Data type	Description
<expression>	-	Logical expression that controls the process flow. (Refer to *4-1-5 Operator* on page 4-12.)
<ifStatement>	-	Executed statement if the following logical expression after the If statement is true
<elseifStatement>	-	Executed statement if the following logical expression after the Elseif statement is true
<elseStatement>	-	Executed statement if all logical expressions in the statement are false

Return value:
None.

Description:
If the logical expression specified in the <expression> parameter is true, the specified statement by the <statement> parameter in the If block or multiple of Elseif blocks is executed.
If the logical expression specified in the <expression> parameter in the "If" part statement is true, the If block statement specified in the <ifStatement> parameter is executed.
If the logical expression specified in the <expression> parameter in the "If" part statement is false, and the logical expression specified in the <expression> parameter in the "Elseif" part statement is true, the Elseif block statement specified in the <elseifStatement> parameter is executed.
If all logical expressions specified in the <expression> parameter are false, the specified Else block statement by the <elseStatement> parameter is executed.
If there are multiple of If, Elseif, and Else statements having a true logical expression within the statement, only the first statement from the beginning of the block statement having a true logical expression is executed.
ElseIf block statements and Else block statement are optional.
If the program process is jumped into or out of the Do block statement using the Goto function in a statement, unexpected operation may occur.
If neither the If statement nor the EndIf statement is used, either the "ELSEIF without IF", "ELSE without IF", "ENDIF without IF", "IF without ENDIF", "ELSEIF without ENDIF", or "ELSE without ENDIF" error will occur depending on the statement that is used.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Uses the *MEASUREDISPG subroutine in the Unit Macro processing unit to output the measured correlation value with the search processing unit (Processing Unit number 1) to the system status console window.

```
*MEASUREDISPG

   Rem Get the measurement result.
   GetUnitData 1, 5, RESULT&

   Rem Branch the process according the measured value.
   If RESULT&>=80 Then
      DrawTextG "Excellent", 100, 100, 0
   Elseif RESULT&>=60 Then
      DrawTextG "Good", 100, 100, 0
   Else
      DrawTextG "Bad", 100, 100, 0
   EndIf

Return
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

DrawTextG on page 8-111
GetUnitData on page 8-153
If Then Else on page 8-159
Select Case Case Else End Select on page 8-266

ImageFormat

Gets the image format of the image in the processing unit.

Format:
ImageFormat(<unitNo>, <measureImageNo>)

Parameter:

Parameter name	Data type	Description
<unitNo>	Integer	Processing unit number (0 to (the number of registered processing units in the current scene minus one))
<measureImageNo>	Integer	Measurement image number of image format to be gotten (always 0)

Return value:
Returns the image format as an integer value.
- 0: Binary image
- 1: Monochrome image
- 2: Color image
- -1: Invalid image

Description:
Gets the format of the image of the image number specified in the <measureImageNo> parameter, of the processing unit specified in the <unitNo> parameter.

If a camera image input (Camera Image Input FH, Camera Image Input HDR, Camera Image Input HDR Lite, Photometric Stereo Image Input, etc.) processing item in the <unitNo> parameter, an "Illegal function call" error will occur.
Normally 0 should be specified in the <measureImageNo> parameter.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If a value outside the range -2147483648 to 2147483647 is specified as an integer parameter, an "Overflow" error will occur.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
In the *MEASUREDISPT subroutine of the unit macro, displays the character string corresponding to the image format in the text window.

```
*MEASUREDISPT

   Rem Get the judgement result of the processing unit.
   JUDGE& = UnitJudge(UnitNo)

   Rem Get the image format.
   FORMAT& = ImageFormat(UnitNo, 0)

   If FORMAT& = 2 Then
      Rem If the format is a color image, display "Color".
      DrawText "Color", JUDGE&, 1
   Elseif FORMAT& = 1 Then
      Rem If the format is a monochrome image, display "Monochrome".
      DrawText "Monochrome", JUDGE&, 1
   EndIf
Return
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

DrawText on page 8-109
UnitJudge on page 8-348
UnitTitle$ on page 8-350
Ut on page 8-351

ImageUpdate

Updates the image input from the camera.

Format:
ImageUpdate

Parameter:
None.

Return value:
None.

Description:
Updates the images of processing units related to image input in the measurement flow and processing units related to image conversion to the image that uses the most recent image from the camera.

Usage Cautions:
None.

Example:
In the communication command macro, updates an image displayed in *Camera Image Freeze* image mode to the most recent image.

```
Rem Update the freeze image to the most recent image.
ImageUpdate

Rem Apply the updated image to the display in the image window.
RefreshImageWindow
```

Usable Modules:
Scene Control Macro / Communication Command Macro

Supported Versions:
Version 3.50 or later

Related Items:

GetImageWindow on page 8-137
SetImageWindow on page 8-276
RefreshImageWindow on page 8-234

Input#

Reads data from the file.

Format:
Input# <fileNo>, <data>[, <data>...]

Parameter:

Parameter name	Data type	Description
<fileNo>	Integer	Read file number (0 to 15)
<data>	Integer Real number Character string Array	Loaded data

Return value:
None.

Description:
Reads the comma separated data in a separated line by a line break code within the specified file number by the <fileNo> parameter.
Specify the variable to store the read data in the <data> parameter.
The value of the read global data is converted to the specified variable type and stored in the <data> parameter. If a character string that cannot be converted to a numerical value is read and an integer or real type variable is specified for the <data> parameter, 0 is stored in the read data.
If the read data contains a double quoted character string, the double quotation marks (") are also read as characters.
If a line break code does not exist at the end of the line of data to be read, an Illegal function call error will occur.
If an unopened file number is specified in the <fileNo> parameter, an "Illegal function call" error will occur.
If a file number of the opened file by a macro function other than the Open For Input As# function is specified in the <fileNo> parameter, an "Illegal function call" error will occur.
If a value outside the range of 0 to 15 is specified in the <fileNo> parameter, an "Illegal function call" error will occur.
If the number of variables specified in the <data> parameter and the number of data to be read do not match, an "Illegal function call" error will occur.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Reads the data until the end of the file.

```
Dim ALLDATA$(255)

Rem Open the file.
Open "E:\input.dat" For Input As #1

For I&=0 to 255
   Rem Read line by line from the top of the file.
   Input #1, DATA$
   ALLDATA$(I&) = DATA$

   Rem Check if the end of the file is reached.
   If Eof(1) <> 0 Then
      Exit For
   Endif

Next

Rem Close up the file.
Close #1
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

Close on page 8-49
Input$ on page 8-166
Open For Input As# on page 8-212
Eof on page 8-116
Line Input# on page 8-182

Input$

Reads binary data from the file.

Format:
Input$(<length>[, #<fileNo>])

Parameter:

Parameter name	Data type	Description
<length>	Integer	Bytes number (0 to 255) of read data
<fileNo>	Integer	Read file number (0 to 15)

Return value:
Returns the read binary data value in the character string format.

Description:
Reads specified size of binary data whose read size is specified in the <length> parameter from the file with the specified file number in the <fileNo> parameter.
If a larger byte number than 255 is specified in the <length> parameter, the 255 byte data from the beginning is read.

If a less byte number is specified in the <length> parameter than the byte number of data in the file to be read, only the specified inte number of data is read. Unread data after the specified read range is read next time the Input$ function is executed.
If the read data contains a double quoted character string, the double quotation marks (") are also read as characters.
If a larger value than the byte number of all read data is specified in the <length> parameter, an "Illegal function call" error will occur.
If an unopened file number is specified in the <fileNo> parameter, an "Illegal function call" error will occur.
If a file number of the opened file by a macro function other than the Open For Input As# function is specified in the <fileNo> parameter, an "Illegal function call" error will occur.
If a value outside the range of 0 to 15 is specified in the <fileNo> parameter, an "Illegal function call" error will occur.
If a value outside the range -2147483648 to 2147483647 is specified as an integer parameter, an "Overflow" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Reads 6 bytes of binary data from the file.

```
Rem Open the file.
Open "E:\input.dat" For Input As #1

Rem Read 6 bytes of data from the file.
DATA$ = Input$(6, #1)

Rem Close up the file.
Close #1
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

Close on page 8-49
Line Input# on page 8-182
Input$ on page 8-166
Open For Input As# on page 8-212

InsertUnit

Inserts a processing unit.

Format:
InsertUnit <unitNo>, <itemIdent>

Parameter:

Parameter name	Data type	Description
<unitNo>	Integer	Processing unit number on the measurement flow to insert the processing unit (0 to the number of registered processing units in the current scene)
<itemIdent>	Character string	Identification name of the processing item to insert the processing unit*9-5 List for Processing Item Identifier* on page 9-67

Return value:
None.

Description:
Registers the processing item with the identification name specified in the <itemIdent> parameter as a processing unit in the position in the measurement flow specified in the <unitNo> parameter.
The processing unit numbers of processing units after the processing unit number specified in the <unitNo> parameter are moved down by 1.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If an identification name that does not exist is specified as the <itemIdent> parameter, an "Illegal function call" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
- Execute this macro function when the BUSY signal or other measurement in progress signal is ON and measurement is prohibited. (For details, refer to *5-1-4 State Transitions and Execution Timing* on page 5-13.)

Example:
Inserts the search processing unit between Processing Unit number 2 and Processing Unit number 3.

```
InsertUnit 3, "Search"
```

Usable Modules:
Scene Control Macro / Communication Command Macro

Supported Versions:
Version 3.50 or later

Related Items:

AssignUnit on page 8-35
CopyUnit on page 8-58
MeasureStart on page 8-200
CheckUnit on page 8-43
DeleteUnit on page 8-72
MeasureStop on page 8-201

MoveUnit on page 8-209 *UnitCount* on page 8-340

Int

Converts numeric value to integer value.

Format:
Int(<expression>)

Parameter:

Parameter name	Data type	Description
<expression>	Real number	Expression to get the integer value

Return value:
Returns an integer value.

Description:
Rounds off digits to the right of the decimal point in the expression specified in the <expression> parameter, and converts the value to the maximum integer value that does not exceed the value of the specified expression.
If a negative value is specified in the <expression> parameter, the Int function will return the greatest negative value that does not exceed the specified negative value. This contrasts with the Fix function that returns the least negative integer value greater than the specified negative value. For example, Fix(-7.2) returns -7 and Int(-7.2) returns -8.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Changes the real number value of a measurement result to an integer by rounding off digits to the right of the decimal point.

```
NUMBER1& = Int(9.7)
NUMBER2& = Int(-9.7)
NUMBER3& = Int(-9.2)
```

The result is shown below.

```
NUMBER1& = 9
NUMBER2& = -10
NUMBER3& = -10
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

Fix on page 8-130
GetUnitData on page 8-153
UnitData on page 8-341

Isfile

Checks the attribute and the existence of the file.

Format:
Isfile(<fileName>)

Parameter:

Parameter name	Data type	Description
<fileName>	Character string	Absolute path of the file to be checked

Return value:
Returns integer values representing the attribute and the existence of the file.
- 0: There is no file
- 1: File attribute is "file"
- 2: File attribute is "directory"

Description:
Checks the attribute and the existence of the file specified in the <fileName> parameter.
In the <fileName> parameter, use an absolute path to specify the file name of the file to be checked.
If the file in the external storage is specified in the <fileName> parameter without the external storage being inserted to the sensor controller, "0" (There is no file) is returned.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
Even if a character string longer than 255 characters is specified for a character string parameter, an error will not occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Checks if the file exists before opening it.

```
Rem Check the file existence.
If Isfile("E:\input.dat") <> 1 Then
   Print "There is no file"
Endif
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

Dskf on page 8-114
Fcopy on page 8-129

GetSystemData on page 8-148
Kill on page 8-176
Open For Append As# on page 8-211
Open For Input As# on page 8-212
Open For Output As# on page 8-213
Print on page 8-221

ItemCount

Gets the number of usable processing item types.

Format:
ItemCount

Parameter:
None.

Return value:
Returns the number of usable processing item types as an integer value.

Description:
Gets the number of processing item types that can be used on the sensor controller.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Gets the number of processing item types, and by repeated processing, searches for search processing items and gets the processing item type of each search processing item.

```
Rem Get the number of the available processing items.
INUM& = ItemCount

Rem Search for search processing items a number of times equal to the number of pr
ocessing items.
For I&=0 To INUM&-1
   If ItemIdent$(I&) = "Search" Then
      Rem Get the processing item type of the search processing item.
      FIGNAME& = ItemInfo(I&, 0)
   Endif
Next
```

Usable Modules:
Scene Control Macro / Communication Command Macro

Supported Versions:
Version 3.50 or later

Related Items:

ItemIdent$ on page 8-172
ItemInfo on page 8-173
ItemTitle$ on page 8-174

ItemIdent$

Gets the identification name of the processing item.

Format:
ItemIdent$(<itemNo>)

Parameter:

Parameter name	Data type	Description
<itemNo>	Integer	Processing item number to get the identification name of the processing item

Return value:
Returns the value of the processing item identification name as a character string.

Description:
Gets the identification name of the processing item with the processing item number specified in the <itemNo> parameter.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
Type mismatch error is occurred when wrong data is specified as parameter. Illegal function call error is not occurred even if non-exist number, values, combination of data or values.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Gets the number of processing item types, and by repeated processing, searches for search processing items and gets the processing item type of each search processing item.

```
Rem Get the number of the available processing items.
INUM& = ItemCount

Rem Search for search processing items a number of times equal to the number of pr
ocessing items.
For I&=0 To INUM&-1
   If ItemIdent$(I&) = "Search" Then
      Rem Get the processing item type of the search processing item.
      FIGNAME& = ItemInfo(I&, 0)
   Endif
Next
```

Usable Modules:
Scene Control Macro / Communication Command Macro

Supported Versions:
Version 3.50 or later

Related Items:

ItemCount on page 8-171
ItemInfo on page 8-173
ItemTitle$ on page 8-174

ItemInfo

Gets the processing item information.

Format:
ItemInfo(<itemNo>, <kind>)

Parameter:

Parameter name	Data type	Description
<itemNo>	Integer	Processing item number to get the information
<kind>	Integer	Type of information 0: Processing item type Number that indicates the processing item type. The values below can be gotten. • 0: Inspect and Measure (measurement) • 1: Input image (Image input) • 2: Compensate image (image correction) • 3: Support Inspection and Measurement (supplementary measurement) • 4: Branch (branch control) • 5: Output result (result output) • 6: Display result (result display)

Return value:
Returns processing item information as an integer. Returns -1 if information does not exist.

Description:
Gets the information specified in the <kind> parameter of the processing item specified in the <item-No> parameter.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
Type mismatch error is occurred when wrong data is specified as parameter. Illegal function call error is not occurred even if non-exist number, values, combination of data or values.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Gets the number of processing item types, and by repeated processing, searches for search processing items and gets the processing item type of each search processing item.

```
Rem Get the number of the available processing items.
INUM& = ItemCount

Rem Search for search processing items a number of times equal to the number of pr
ocessing items.
For I&=0 To INUM&-1
   If ItemIdent$(I&) = "Search" Then
      Rem Get the processing item type of the search processing item.
      FIGNAME& = ItemInfo(I&, 0)
   Endif
Next
```

Usable Modules:
Scene Control Macro / Communication Command Macro

Supported Versions:
Version 3.50 or later

Related Items:

ItemCount on page 8-171
ItemIdent$ on page 8-172
ItemTitle$ on page 8-174
UnitInfo on page 8-345

ItemTitle$

Gets the processing item title.

Format:
ItemTitle$(<itemNo>)

Parameter:

Parameter name	Data type	Description
<itemNo>	Integer	Processing item number to get the title name

Return value:
Returns the title as a character string.

Description:
Gets the title of the processing item specified in the <itemNo> parameter.
The title can be gotten in a language based on the language setting.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
Type mismatch error is occurred when wrong data is specified as parameter. Illegal function call error is not occurred even if non-exist number, values, combination of data or values.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:

Gets the number of processing item types, and gets the title of the processing item of each processing item number by repeated processing.

```
Rem Get the number of the available processing items.
INUM& = ItemCount

Rem Repeat the process up to the number of the processing items.
For I&=0 To INUM&-1
   Rem Get the title of the processing item.
   TITLE$ = ItemTitle$(I&)
Next
```

Usable Modules:
Scene Control Macro / Communication Command Macro

Supported Versions:
Version 3.50 or later

Related Items:

ItemCount on page 8-171
ItemIdent$ on page 8-172
ItemInfo on page 8-173

JudgeOut

Sets the output state of the overall judgement signal.

Format:
JudgeOut <ioIdent>, <state>

Parameter:

Parameter name	Data type	Description
<ioIdent>	Character string	Identification name of the communication module to be used ("ParallelIo" or "EtherCAT") (Refer to *9-4 List of I/O Modules* on page 9-27.)
<state>	Integer	Total judgement result to be output JUDGE_OK: Total judgement result OK JUDGE_NG: Total judgement result NG The actual signal output depends on this parameter setting and the total judgement (Total Judgement or OR) signal output polarity setting. Specify ON at NG for the total judgement signal polarity when using the communication modules other than ParallelIO.

Return value:
None.

Description:
Sets the output state specified in the <state> parameter of the OR signal or other overall judgement signal of the communication module specified in the <ioIdent> parameter.
Normally "ParallelIo" or "EtherCAT" should be specified in the <ioIdent> parameter.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.

If an identification name that does not exist is specified in the <ioIdent> parameter, an "Illegal function call" error will occur.
Even if an output status parameter value that does not exist (i.e., other than 0 and 1) is specified in the <state> parameter, an error will not occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
In the communication command macro, sets the OR signal of parallel I/O to ON.

```
IOMODULE$ = "ParallelIo"

Rem Set the output state.
JudgeOut IOMODULE$, 1
```

Usable Modules:
Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

BusyOut on page 8-37
GetPort on page 8-144
PutPort on page 8-225
TotalJudge on page 8-328
GetAll on page 8-134
PutAll on page 8-223
RunOut on page 8-245

Kill

Deletes a file.

Format:
Kill <fileName>

Parameter:

Parameter name	Data type	Description
<fileName>	Character string	Absolute path of the file to delete

Return value:
None.

Description:
Deletes the file of the file name specified in the <fileName> parameter.
In the <fileName> parameter, use an absolute path to specify the file name including the drive name.
Characters * (character string wildcard operator) and ? (single character wildcard operator) can be used as wildcards for the <fileName> specification. These wildcards can be used to specify file names

and cannot be used for the directory name specification. Directory names that match the wildcard pattern will not be deleted. For example, the "Kill "E:\img*" statement will not delete the "M:\img\caputure \" directory.
Wildcards can be used to specify file names in the following manner.

.	Specify all file with the extension.
*	Specify all files.
???.*	Specify the files with file name having an extension with a three-character file root name.
???????	Specify the files with file name having a seven-character file name including the file name extension separator (.).
A*.*	Specify the files with file name having an extension with file name starting with "A".
A*A*A	Specify the files with file name (including file extension) having three and more "A"s.
????*	Specify the files with file name (including file extension) having four and more characters.
????*.*	Specify the files with a file name having a file root name with four or more characters andan extension.
*.??	Specify the file with 2 characters extension name.

In the following cases, the file cannot be deleted.
- The specified file does not exist.
- The external storage has not been inserted.

If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
An error of "String too long" will be occurred when you specify the character strings as character strings type exceeds 255 characters.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Deletes a file named "1280-720.bmp" under the drive named "M:\"

```
Kill "M:\1280-720.bmp"
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

Dskf on page 8-114
Fcopy on page 8-129
Isfile on page 8-170

LanguageString$

Gets the character string in the language according to the current language setting.

Format:
LanguageString$(<srcString>)

Parameter:

Parameter name	Data type	Description
<srclabel>	Character string	Character string format (Description using language identifier) For the description method using language identifier, refer to the *Inputting text* in the *Vision System FH/FHV Series User's Manual (Cat. No. Z365)*.

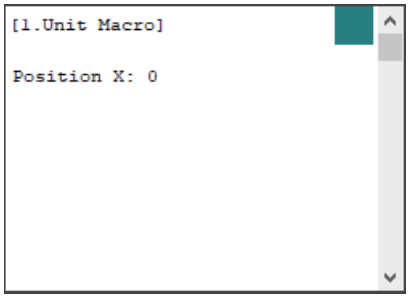
Return value:
Returns a character string.

Description:
Gets a character string according to the current language setting from the character string format (description using language identifier) specified by the <srclabel> parameter.
If a character string that is not described using language identifier or an empty character string is specified, the specified string will be returned as it is.
If a character string format (description using language identifier) that is not supported in the current language is specified, the character string supported in English will be returned.
If the specified string contains neither a string supported in the current language nor a string supported in English, an empty character string will be returned.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Displays a character string according to the current language setting in the detailed result display.

```
*MEASUREDISPT
  TEXT$ = LanguageString$("\jpn\X zahyou: \eng\Position X: ")
  TEXT$ = TEXT$ + Str$(POSX#)
  DrawText TEXT$, UnitJudge(UnitNo), 0
Return
```



Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 5.75 or later

Related Items:
None.

LCase$

Converts an upper case letter to a lower case letter.

Format:
LCase$(<string>)

Parameter:

Parameter name	Data type	Description
<string>	Character string	Character string contains an alphabet to be converted to lower case.

Return value:
Returns the case converted character string type value.

Description:
Converts the upper case letters in the character strings specified in the <string> parameter to lower case.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a character string longer than 255 characters is specified for a character string parameter, the 255-character string before the 256th character is used for the macro function processing. Characters after the 256th character will be discarded.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Converts an upper case letter to a lower case letter.

```
CHARA1$ = "Measurement Result = 100.0(OK)"

Rem Convert the upper case letters in the character strings to lower case.
CHARA2$ = LCase$(CHARA1$)
```

The result is shown below.

```
CHARA2$ = "measurement result = 100.0(ok)"
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 4.20 or later

Related Items:

Asc on page 8-34
Chr$ on page 8-44
Hex$ on page 8-158
Left$ on page 8-180
Len on page 8-181
Mid$ on page 8-205
Piece$ on page 8-219
Right$ on page 8-243
Str$ on page 8-314
Str2$ on page 8-315
UCase$ on page 8-339
Val on page 8-352

Left$

Extracts the specified length of characters from the left side of character string.

Format:
Left$(<string>, <length>)

Parameter:

Parameter name	Data type	Description
<string>	Character string	Extraction target character string
<length>	Integer	Length of characters to be extracted (1 to the length of the target character string)

Return value:
Returns the character string type value of the extracted character string.

Description:
If 0 or less number is specified in the <length> parameter, an "Illegal function call" error will occur.
Specify the length of characters to be extracted in bytes for the <length> parameter. Each single-byte character (i.e., half-width alphanumeric character and symbol) consumes one byte, whereas each doublebyte character consumes two bytes.
Extracts the specified length in the <length> parameter from the left side of specified character string in the <string> parameter.
If the length specified in the <length> parameter is longer than the length of the character string specified in the <string> parameter, the whole character string in the parameter is extracted.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a value outside the range -2147483648 to 2147483647 is specified as an integer parameter, an "Overflow" error will occur.
If a character string longer than 255 characters is specified for a character string parameter, the 255-character string before the 256th character is used for the macro function processing. Characters after the 256th character will be discarded.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:

Extracts 11-byte length of characters from the left side of the character string. Because one half-width alphabet consumes single byte, this example extracts 11 characters from the character string.

```
CHARA$ = "Measurement Result"

Rem Extract 11-byte length of characters from the left side of the character strin
g.
TITLE$ = Left$(CHARA$, 11)
```

The result is shown below.

```
TITLE$ = "Measurement"
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

Asc on page 8-34
Hex$ on page 8-158
Len on page 8-181
Piece$ on page 8-219
Str$ on page 8-314
UCase$ on page 8-339
Chr$ on page 8-44
LCase$ on page 8-179
Mid$ on page 8-205
Right$ on page 8-243
Str2$ on page 8-315
Val on page 8-352

Len

Gets the length of the specified character string.

Format:
Len(<string>)

Parameter:

Parameter name	Data type	Description
<string>	Character string	Character string to be gotten the length of

Return value:
Returns the character string length in bytes as an integer value.

Description:
Gets the length of the specified character string in the <string> parameter in bytes.
Each single-byte character (i.e., half-width alphanumeric character and symbol) consumes one byte, whereas each double-byte character consumes two bytes.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
Even if a character string longer than 255 characters is specified for a character string parameter, an error will not occur.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.

If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Gets the length of the character string "OMRON".

```
CHRLEN& = Len("OMRON")
```

The result is shown below.

```
CHRLEN& = 5
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

Asc on page 8-34
Chr$ on page 8-44
Hex$ on page 8-158
LCase$ on page 8-179
Left$ on page 8-180
Mid$ on page 8-205
Piece$ on page 8-219
Right$ on page 8-243
Str$ on page 8-314
Str2$ on page 8-315
UCase$ on page 8-339
Val on page 8-352

Line Input#

Reads the data of one line from the file.

Format:
Line Input #<fileNo>, <data>

Parameter:

Parameter name	Data type	Description
<fileNo>	Integer	Read file number (0 to 15)
<data>	Character string	Loaded data

Return value:
None.

Description:
Read the data of one line separated by the line break code from the file of the file number specified in the <fileNo> parameter. The Input# function operates in similar way. This Line Input# function reads all data within a single line as a character string type data, whereas the Input# function reads multiple data within a single line and data type specifications for each piece of read data are required in advance.
Specify the variable to store the read data in the <data> parameter.
When a read line only has a line break code, a null character string is stored in the read data.

If the read data contains a double quoted character string, the double quotation marks (") are also read as characters.
If an unopened file number is specified in the <fileNo> parameter, an "Illegal function call" error will occur.
If a file number of the opened file by a macro function other than the Open For Input As# function is specified in the <fileNo> parameter, an "Illegal function call" error will occur.
If a value outside the range of 0 to 15 is specified in the <fileNo> parameter, an "Illegal function call" error will occur.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Reads the data until the end of the file.

```
Dim ALLDATA$(255)

Rem Open the file.
Open "E:\input.dat" For Input As #1

For I&=0 to 255
   Rem Read line by line from the top of the file.
   Line Input #1, DATA$
   ALLDATA$(I&) = DATA$

   Rem Check if the end of the file is reached.
   If Eof(1) <> 0 Then
      Exit For
   Endif
Next

Rem Close up the file.
Close #1
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

Close on page 8-49
Input# on page 8-165
Open For Input As# on page 8-212
Eof on page 8-116
Input$ on page 8-166

List

Outputs all or a part of program list in the system status console window.

Format:
List [<lineNo1>][-<lineNo2>]

Parameter:

Parameter name	Data type	Description
<lineNo1>	Integer	Line number where the output starts
<lineNo2>	Integer	Line number where the output ends

Return value:
None.

Description:
Program list from the specified line number in the <lineNo1> parameter to the specified line number in the <lineNo2> parameter is output to the system status console window.
If the <lineNo1> and <lineNo2> parameters are omitted, all lines in the program are output. If only the <lineNo1> parameter is omitted, program lines from the beginning to the specified line number by the <lineNo2> parameter are output.
If only the <lineNo2> parameter is omitted, only the program line specified in the <lineNo1> parameter is output.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
- This macro function is only enabled when specified in debug mode with the Debug function. Specifying other values than the range above will treat the statement with this function in the same manner with the Rem function (i.e., ignores the statement). (For details, refer to *6-1 How to Use the Debug Function* on page 6-2.)

Example:
After stopping the program execution with the Stop function used in the Unit Macro processing unit, executes the List function in the system status console window to output the program line numbers from 230 to 240 to the window.

```
Macro(U1) Stop in 220
Macro(U1) List 230-240
230 POS.X#=(POS0.X@ + POS1.X@) / 2
240 POS.Y#=(POS0.Y@ + POS1.Y@) / 2
Macro(U1)>
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

Cls on page 8-52
Cont on page 8-53
Debug on page 8-68
DebugPrint on page 8-69
Print on page 8-221
SetStop on page 8-293
SetVar on page 8-307
Stop on page 8-312
SubList on page 8-320
VarList on page 8-353

LoadBackupData

Loads the system + scene group 0 data.

Format:
LoadBackupData(<fileName>)

Parameter:

Parameter name	Data type	Description
<fileName>	Character string	File name of bkd file to read in (System data + scene group 0 data (*.bkd))

Return value:
None.

Description:
Loads the system + scene group 0 data file specified in the <fileName> parameter.
In the <fileName> parameter, use an absolute path to specify the file name of the file to be loaded.
In the <fileName> parameter, specify a system + scene group 0 data file name. If a file other than a system + scene group 0 data file is specified, an "Illegal function call" error will occur.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
An error of "String too long" will be occurred when you specify the character strings as character strings type exceeds 255 characters.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
- Execute this macro function when the BUSY signal or other measurement in progress signal is ON and measurement is prohibited. (For details, refer to *5-1-4 State Transitions and Execution Timing* on page 5-13.)
- After loading a file, execute **Save data**. To apply the loaded settings in the sensor controller, restart the sensor controller.

Example:
Loads a system + scene group 0 data file, and execute **Save data**. To apply the loaded settings in the sensor controller, restart the sensor controller.

```
Rem Load the system + scene group 0 data to a file.
LoadBackupData "E:\BACKDIR¥BackupData.bkd"

Rem Save to the controller.
SaveData

Rem Reboot the sensor controller.
SystemReset
```

Usable Modules:
Communication Command Macro

Supported Versions:
Version 3.50 or later

Related Items:

GetSystemData on page 8-148
LoadSceneGroup on page 8-187
LoadUnitData on page 8-189
SaveData on page 8-248
LoadScene on page 8-186
LoadSystemData on page 8-188
SaveBackupData on page 8-247
SystemReset on page 8-322

LoadScene

Loads the scene data.

Format:
LoadScene <sceneNo>, <fileName>

Parameter:

Parameter name	Data type	Description
<sceneNo>	Integer	Scene number of the destination to read in (0 to 127)
<fileName>	Character string	File name of the scene data to read in (*.scn)

Return value:
None.

Description:
The scene data file specified in the <fileName> parameter is loaded into the scene number specified in the <sceneNo> parameter.
In the <fileName> parameter, use an absolute path to specify the file name of the file to be loaded.
In the <fileName> parameter, specify a scene data file name. If a file other than a scene data file is specified, an "Illegal function call" error will occur
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
An error of "String too long" will be occurred when you specify the character strings as character strings type exceeds 255 characters.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:

- Execute this macro function when the BUSY signal or other measurement in progress signal is ON and measurement is prohibited. (For details, refer to *5-1-4 State Transitions and Execution Timing* on page 5-13.)

Example:
Loads scene data into scene 2 and then changes the scene used to scene 2.

```
Rem Load the file of the scene data.
LoadScene 2, "E:\BACKDIR\scene02.scn"

Rem Switch the scene.
ChangeScene 2
```

Usable Modules:
Communication Command Macro

Supported Versions:
Version 3.50 or later

Related Items:

ChangeScene on page 8-41
LoadBackupData on page 8-185
LoadSystemData on page 8-188
SaveScene on page 8-252
GetSystemData on page 8-148
LoadSceneGroup on page 8-187
LoadUnitData on page 8-189
SceneNo on page 8-263

LoadSceneGroup

Loads the scene group data.

Format:
LoadSceneGroup <sceneGroupNo>, <fileName>

Parameter:

Parameter name	Data type	Description
<sceneGroupNo>	Integer	Scene group number of the destination to read in (0 to 31)
<fileName>	Character string	File name of the scene group data to read in (*.sgp)

Return value:
None.

Description:
Loads the scene group data file specified in the <fileName> parameter into the scene group number specified in the <sceneGroupNo> parameter.
In the <fileName> parameter, use an absolute path to specify the file name of the file to be loaded.
In the <fileName> parameter, specify the file name of the scene group data. If a file other than a scene group data file is specified, an "Illegal function call" error will occur.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
An error of "String too long" will be occurred when you specify the character strings as character strings type exceeds 255 characters.

If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:

- Execute this macro function when the BUSY signal or other measurement in progress signal is ON and measurement is prohibited. (For details, refer to *5-1-4 State Transitions and Execution Timing* on page 5-13.)

Example:

Loads scene data into scene group 2 and then chenges the scene used to scene 0.

```
Rem Load the file of the scene group data.
LoadSceneGroup 2, "E:\BACKDIR\scenegroup02.sgp"

Rem Switch the scene.
ChangeScene 0
```

Usable Modules:

Communication Command Macro

Supported Versions:

Version 3.50 or later

Related Items:

ChangeScene on page 8-41
GetSystemData on page 8-148
LoadBackupData on page 8-185
LoadScene on page 8-186
LoadSystemData on page 8-188
LoadUnitData on page 8-189
SaveSceneGroup on page 8-253
SceneGroupNo on page 8-260

LoadSystemData

Loads the system data.

Format:

LoadSystemData <fileName>

Parameter:

Parameter name	Data type	Description
<fileName>	Character string	File name of the system data to read in (*.ini)

Return value:

None.

Description:

Loads the system data file specified in the <fileName> parameter.
In the <fileName> parameter, use an absolute path to specify the file name of the file to be loaded.
In the <fileName> parameter, specify a system data file name. If a file other than a system data file is specified, and "Illegal function call" error will occur.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.

An error of "String too long" will be occurred when you specify the character strings as character strings type exceeds 255 characters.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:

- Execute this macro function when the BUSY signal or other measurement in progress signal is ON and measurement is prohibited. (For details, refer to *5-1-4 State Transitions and Execution Timing* on page 5-13.)
- After loading a file, execute **Save data**. To apply the loaded settings in the sensor controller, restart the sensor controller.

Example:

After loading the system data, executes **Save data**. To apply the loaded settings in the sensor controller, restart the sensor controller.

```
Rem Load the file of the system data
LoadSystemData "E:\BACKDIR\backupsysset.ini"

Rem Save to the controller.
SaveData

Rem Reboot the sensor controller.
SystemReset
```

Usable Modules:

Communication Command Macro

Supported Versions:

Version 3.50 or later

Related Items:

GetSystemData on page 8-148
LoadBackupData on page 8-185
LoadScene on page 8-186
LoadSceneGroup on page 8-187
LoadUnitData on page 8-189
SaveData on page 8-248
SaveSystemData on page 8-254
SystemReset on page 8-322

LoadUnitData

Loads the processing unit data.

Format:

LoadUnitData <sceneNo>, <unitNo>, <unitCount>, <mode>, <fileName>

Parameter:

Parameter name	Data type	Description
<sceneNo>	Integer	Scene number of the destination to read in (-1 to 127)
<unitNo>	Integer	Processing unit number to begin to read in (0 to (the number of registered processing units in the current scene minus one))

Parameter name	Data type	Description
<unitCount>	Integer	Number of pieces of the processing unit to read in (-1, 1 to (the number of registered processing units in the current scene) - (the processing unit number to begin to read in))
<mode>	Integer	Read mode 0: Overwrite the processing unit by the loaded processing unit. 1: Insert the loaded processing unit in front of the processing unit.
<fileName>	Character string	File name of the processing unit to read in (*unt)

Return value:

None.

Description:

Loads the processing unit data file specified in the <fileName> parameter in the mode specified in the <mode> parameter from processing units whose processing unit numbers are specified in the <unitNo> parameter, with the number of processing units specified in the <unitCount> parameter, of the scene number specified in the <sceneNo> parameter.
If -1 is specified in the <sceneNo> parameter, the scene number of the current scene is specified in the scene number of the destination scene.
If -1 is specified in the <unitCount> parameter, all processing unit data included in the processing unit data file is loaded.
Designate a read mode with the <mode> parameter. For processing units with multiple processing items combined, such as folder, parallelization, or conditional execution (if), do not load a unit using the overwrite mode. To swap with a unit saved in the file, delete a folder using the DeleteUnit function, and then load a unit saved in the file using the insert mode.
In the <fileName> parameter, use an absolute path to specify the file name of the file to be loaded.
In the <fileName> parameter, specify the file name of a processing unit data file. If a file other than a processing unit data file is specified, an "Illegal function call" error will occur, and the scene data in the load destination will be cleared.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
An error of "String too long" will be occurred when you specify the character strings as character strings type exceeds 255 characters.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:

- Execute this macro function when the BUSY signal or other measurement in progress signal is ON and measurement is prohibited. (For details, refer to *5-1-4 State Transitions and Execution Timing* on page 5-13.)

Example:

Loads a processing unit data file and inserting five of the same processing units between Processing Unit number 3 and Processing Unit number 4 of scene 2.

```
Rem Load the file of the scene data.
LoadUnitData 2, 4, 5, 1, "E:\BACKDIR\unitsave.unt"
```

Usable Modules:

Communication Command Macro

Supported Versions:
Version 3.50 or later

Related Items:

GetSystemData on page 8-148
LoadScene on page 8-186
LoadSystemData on page 8-188
SceneNo on page 8-263
Ut on page 8-351
LoadBackupData on page 8-185
LoadSceneGroup on page 8-187
SaveUnitData on page 8-255
UnitNo on page 8-349

Log

Gets the natural logarithm.

Format:
Log(<expression>)

Parameter:

Parameter name	Data type	Description
<expression>	Integer Real number	Expression to get the natural logarithm

Return value:
Returns the natural logarithm value as a real number.

Description:
Gets the value of the natural logarithm of the expression specified in the <expression> parameter.
In the <expression> parameter, specify a positive number.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Gets the natural logarithm of the integer 25 assigned to the variable X&.

```
X& = 25
XLOG# = Log(X&)
```

The result is shown below.

```
XLOG# = 3.21887582487
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

Exp on page 8-128
GetUnitData on page 8-153
UnitData on page 8-341

Lsqumeth

Gets the approximate straight line from the coordinates of multiple points using the least squares method.

Format:
Lsqumeth <count>, <x()>, <y()>, <line()>

Parameter:

Parameter name	Data type	Description
<count>	Integer	Number of coordinates calculated for the approximate line
<x>	Real number array	1D array that stores the X coordinates of the points from which the approximate straight line is to be gotten
<y>	Real number array	1D array that stores the Y coordinates of the points from which the approximate straight line is to be gotten
<line()>	Real number array	Straight line component gotten by the approximate line

Return value:
None.

Description:
Gets the approximate straight line from the coordinates specified in the <x()> parameter and <y()> parameter. The number of coordinates is specified in the <count> parameter.
In the <line()> parameter, specify the variable that will hold the line components of the gotten approximate straight line, without adding element numbers but adding () to the variables. In the <line()> parameter, the parameters a, b, and c that satisfy the linear equation ax + by + c = 0 are stored in the array elements of the double precision real number array variable, with "a" in element 0, "b" in element 1, and "c" in element 2.
0, "b" in element 1, and "c" in element 2.
In the <x()> parameter and in the <y()> parameter, specify a 1D double precision real number array variable that stores a number of coordinate values greater than or equal to the number specified in the <count> parameter, without adding element numbers but adding () to the variables.
This macro function is mainly used to get the straight line of an edge from the measurement points gotten in multiple edge measurements.
If an undefined array is specified a parameter, an "Undefined array" error will occur.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:

None.

Example:
Gets the straight line of the edge of a workpiece from four edge points.

```
Dim POSX#(3), POSY#(3), PARAM#(2)

Rem Calculate the edge points from the measurement result.
For I&=0 To 3
   GetUnitData I&+1, "X", POSX#(I&)
   GetUnitData I&+1, "Y", POSY#(I&)
Next

Rem Get the straight line component.
Lsqumeth 4, POSX#(), POSY#(), PARAM#()

Erase POSX#(), POSY#(), PARAM#(), DIST#()
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

ApproximationCircle on page 8-31
Dposline on page 8-78
GetUnitData on page 8-153
Crspoint on page 8-65
Erase on page 8-118

Measure

Executes measurement processing.

Format:
Measure[<wait>]

Parameter:

Parameter name	Data type	Description
<wait>	Integer	Recovery timing of macro function 0: Executes subsequent program lines without waiting for measurement to end. 1: Waits for measurement to end and then executes subsequent program lines. 2: Waits for measurement to end and measurement result display to end and then executes subsequent program lines. 3: Waits for measurement to end, measurement result display to end, and UI display update to end then executes subsequent program lines.

Return value:
None.

Description:

Executes measurement processing at the recovery timing specified in the <wait> parameter.
If the <wait> parameter is omitted, operation is the same as when the <wait> parameter is set to 0.
When 0 is specified for the <wait> parameter, there is a possibility that the measurement processing executed immediately after execution of this macro function will not have ended and the measurement result cannot be properly gotten. If you want to get the measurement result, specify 1 or 2 or 3 for the <wait> parameter.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
Type mismatch error is occurred when wrong data is specified as parameter. Illegal function call error is not occurred even if non-exist number, values, combination of data or values.
If a value outside the range -2147483648 to 2147483647 is specified as an integer parameter, an "Overflow" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:

- Executes this function in the status when measure signal is OFF.(For details, refer to *5-1-4 State Transitions and Execution Timing* on page 5-13.)

Example:

In the communication command macro, gets the measurement X coordinate and measurement Y coordinate of the search processing unit of Processing Unit number 2 after measurement is executed.
The measured X and Y coordinates can be gotten with External Reference Data numbers 6 and 7 respectively.

```
Rem Execute measurement and wait until measurement ends.
Measure 1

Rem Get the measurement result.
GetUnitData 2, 6, POSX#
GetUnitData 2, 7, POSY#
```

Usable Modules:

Communication Command Macro

Supported Versions:

Version 3.50 or later

Related Items:

MeasureStart on page 8-200
MeasureStop on page 8-201
Remeasure on page 8-239

MeasureDispG

Executes display of the measurement result of the processing unit.

Format:
MeasureDispG <unitNo>, <subNo>

Parameter:

Parameter name	Data type	Description
<unitNo>	Integer	Processing unit number (0 to (the number of registered processing units in the current scene minus one))
<subNo>	Integer	Image display sub-number of the image window to be used for display (Refer to *9-10 List of Sub-Image Numbers* on page 9-82.)

Return value:
None.

Description:
Graphically displays the judgement result in the image of the image display sub-number specified in the <subNo> parameter, of the processing unit specified in the <unitNo> parameter.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for the <unitNo> parameter, an "Illegal function call" error will occur.
Even if a non-existent number, numerical value, or combination of data types or values is specified for the <subNo> parameter, an error will not occur.
If a value outside the range -2147483648 to 2147483647 is specified as an integer parameter, an "Overflow" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
- This macro function can only be used in the *MEASUREDISPI or *MEASUREDISPG subroutine. If used in another subroutine, an "Illegal function call" error will occur.

Example:
In the*MEASUREDISPG subroutine of the Unit Macro processing unit, executes the analysis result display process (graphic display) of sub number 0 of Processing Unit number 2.

```
*MEASUREDISPG
   MeasureDispG 2, 0
Return
```

Usable Modules:
Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

UnitNo on page 8-349
Ut on page 8-351

MeasureDispI

Displays the image of the Sub-image number of the processing unit.

Format:
MeasureDispI <unitNo>, <subNo>

Parameter:

Parameter name	Data type	Description
<unitNo>	Integer	Processing unit number (0 to (the number of registered processing units in the current scene minus one))
<subNo>	Integer	Image display Sub-number of processing item (Refer to *9-10 List of Sub-Image Numbers* on page 9-82.)

Return value:
None.

Description:
Graphically displays the judgement result in the image of the image display sub-number specified in the <subNo> parameter, of the processing unit specified in the <unitNo> parameter.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for the <unitNo> parameter, an "Illegal function call" error will occur.
Even if a non-existent number, numerical value, or combination of data types or values is specified for the <subNo> parameter, an error will not occur.
If a value outside the range -2147483648 to 2147483647 is specified as an integer parameter, an "Overflow" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
- This macro function can only be used in the *MEASUREDISPI or *MEASUREDISPG subroutine. If used in another subroutine, an "Illegal function call" error will occur.

Example:
In the * MEASUREDISPI subroutine of the Unit Macro processing unit, the shape 2 filter image of the Photometric Stereo Image Input processing item with the processing unit number 0 is displayed as the output image of Unit Macro.

```
*MEASUREDISPI
   MeasureDispI 0, 2
Return
```

Usable Modules:
Unit Macro

Supported Versions:
Version 6.40 or later

Related Items:

UnitNo on page 8-349
Ut on page 8-351

MeasureDispT

Displays detailed results from the processing unit.

Format:
MeasureDispT <unitNo>, <subNo>

Parameter:

Parameter name	Data type	Description
<unitNo>	Integer	Processing unit number (0 to (the number of registered processing units in the current scene minus one))
<subNo>	Integer	IImage display Sub-number of processing item (Refer to *9-10 List of Sub-Image Numbers* on page 9-82.)

Return value:
None.

Description:
Graphically displays the judgement result in the image of the image display sub-number specified in the <subNo> parameter, of the processing unit specified in the <unitNo> parameter.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for the <unitNo> parameter, an "Illegal function call" error will occur.
Even if a non-existent number, numerical value, or combination of data types or values is specified for the <subNo> parameter, an error will not occur.
If a value outside the range -2147483648 to 2147483647 is specified as an integer parameter, an "Overflow" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
- This macro function can only be used in the *MEASUREDISPT subroutine. If used in another subroutine, an "Illegal function call" error will occur.

Example:
In the * MEASUREDISPT Subroutine of the Unit Macro processing unit, the detailed result of the processing item of processing unit number 2 is displayed in the detailed result of Unit Macro.

```
*MEASUREDISPT
   MeasureDispT 2, 0
Return
```

Usable Modules:
Unit Macro

Supported Versions:
Version 6.40 or later

Related Items:

UnitNo on page 8-349
Ut on page 8-351

MeasureId$

Gets the measurement identification.

Format:
MeasureId$

Parameter:
None.

Return value:
Returns the measurement identification as a character string.
Measurement ID: Information of time when the sensor controller receives the measurement trigger and the line no.
Format of measurement ID: YYYY-MM-DD_HH-MM-SS-XXXN
(YYYY: Year, MM: Month, DD: Date, HH: Hour, MM: Minute, SS: Second, XXX: Millisecond, N: Line number)
Example
Measurement time: 11:10:25.500 AM, December 24, 2007 and Line 0, the measurement ID is "2007-12-24_11-10-25-5000".

Description:
Gets the measurement identification.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.

Usage Cautions:
- This macro function can only be used in the *MEASUREPROC subroutine. If used in any other subroutines, an error will occur and the function will not be executed.

Example:
In the *MEASUREPROC subroutine of the Unit Macro processing unit, gets the measurement identification and saves the processing unit measurement image with the measurement identification as the file name.

```
*MEASUREPROC

  Rem Get the measurement identification to set the character string for the file
name.
  FILENAME$ = MeasureId$ + ".bmp"

  Rem Output measurement image 0 in bitmap format.
  SaveMeasureImage 0, FILENAME$, 0

Return
```

Usable Modules:
Unit Calculation Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

SaveMeasureImage on page 8-250

MeasureProc

Executes measurement processing in a processing unit.

Format:
MeasureProc <unitNo>

Parameter:

Parameter name	Data type	Description
<unitNo>	Integer	Processing unit number (0 to (the number of registered processing units in the current scene minus one))

Return value:
None.

Description:
Executes measurement processing in the processing unit specified in the <unitNo> parameter.
If a processing unit related to image input is specified in the <unitNo> parameter, processing may not take place correctly.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If a value outside the range -2147483648 to 2147483647 is specified as an integer parameter, an "Overflow" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Executes measurement processing in the search processing unit of Processing Unit number 2.

```
MeasureProc 2
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 5.40 or later

Related Items:

UnitNo on page 8-349
Ut on page 8-351

MeasureStart

Allows input of the measurement trigger.

Format:
MeasureStart

Parameter:
None.

Return value:
None.

Description:
Allows input of the measurement trigger and input of communication commands, and changes to the measurement allowed state.
After setting the measurement prohibited state with the MeasureStop function, execute this macro function to return to the measurement allowed state.
If input of the measurement trigger is allowed, measurement-in-progress signals such as the BUSY signal turn OFF.

Usage Cautions:
- Execute this macro function when the BUSY signal or other measurement in progress signal is ON and measurement is prohibited. (For details, refer to *5-1-4 State Transitions and Execution Timing* on page 5-13.)
- When executing a macro function that must be executed in the measurement prohibited state, use the MeasureStop function to change the state to the measurement allowed state before executing the function. Otherwise, unexpected behavior may occur.
  There are macro functions that can and cannot be executed in the measurement allowed state and in the measurement prohibited state. For details, refer to the explanations of the macro functions and usage cautions.

Example:
In the communication command macro, executes measurement on the Shape Search III processing unit of Processing Unit number 1 after input of the measurement trigger is allowed.
After the measurement, the measurement trigger is prohibited and then gets the measurement result.

```
Rem Allow input of the measurement trigger.
MeasureStart

Rem Execute measurement.
Measure 1

Rem Prohibit input of the measurement trigger.
MeasureStop

Rem Get the measurement result of the Shape Search III processing unit.
GetUnitData 1, 6, POSX#
GetUnitData 1, 7, POSY#
```

Usable Modules:
Scene Control Macro / Communication Command Macro

Supported Versions:
Version 3.50 or later

Related Items:

DisableUnit on page 8-74
GetUnitData on page 8-153
MeasureStop on page 8-201
SetUnitData on page 8-300
GetSceneGroupData on page 8-147
Measure on page 8-193
SetSceneGroupData on page 8-289

MeasureStop

Prohibits input of the measurement trigger.

Format:
MeasureStop [<mode>]

Parameter:

Parameter name	Data type	Description
<mode>	Integer	Measurement trigger input prohibited mode (always 0)

Return value:
None.

Description:
Prohibits measurement trigger input and communication command input, and sets the measurement prohibited state. After waiting for the measurement to finish and the measurement result display to finish, sets the state.
After executing this macro function to prohibit measurement trigger input, execute the MeasureStart function to return to the measurement allowed state.
If the <mode> parameter is omitted, operation is the same as when 0 is specified for the <mode> parameter.
There are macro functions that can and cannot be executed in the measurement allowed state and in the measurement prohibited state. For details, refer to the explanations of the macro functions and usage cautions.
When measurement trigger input is prohibited, measurement in progress signals such as the BUSY signal turn ON.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
Type mismatch error is occurred when wrong data is specified as parameter. Illegal function call error is not occurred even if non-exist number, values, combination of data or values.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
- Executes this function in the status when measure signal is OFF.(For details, refer to *5-1-4 State Transitions and Execution Timing* on page 5-13.)
- When executing a macro function that must be executed in the measurement prohibited state, use the MeasureStop function to change the state to the measurement allowed state before executing the function. Otherwise, unexpected behavior may occur.

There are macro functions that can and cannot be executed in the measurement allowed state and in the measurement prohibited state. For details, refer to the explanations of the macro functions and usage cautions.

Example:

In the communication command macro, executes measurement on the Shape Search III processing unit of Processing Unit number 1 after input of the measurement trigger is allowed.
After the measurement, the measurement trigger is prohibited and then gets the measurement result.

```
Rem Allow input of the measurement trigger.
MeasureStart

Rem Execute measurement.
Measure 1

Rem Prohibit input of the measurement trigger.
MeasureStop

Rem Get the measurement result of the Shape Search III processing unit.
GetUnitData 1, 6, POSX#
GetUnitData 1, 7, POSY#
```

Usable Modules:

Scene Control Macro / Communication Command Macro

Supported Versions:

Version 3.50 or later

Related Items:

DisableUnit on page 8-74
GetUnitData on page 8-153
MeasureStart on page 8-200
SetUnitData on page 8-300
GetSceneGroupData on page 8-147
Measure on page 8-193
SetSceneGroupData on page 8-289

MemoryInfo

Gets the current memory usage (in size).

Format:
MemoryInfo (<kind>)

Parameter:

Parameter name	Data type	Description
<kind>	Integer	Type of acquired information 0: Overall memory size recognized by the OS 1: Usable memory size recognized by the OS 2: Overall size of extended memory 3: Available size of extended memory 4: Overall size of CoreRA process memory 5: Available size of CoreRA process memory 6: Overall size of processing unit data 7: Usable size of processing unit data 10000: Overall size of physical memory (MB) 10001: Available physical memory size (MB) 10002: Overall size of page file (MB) 10003: Available size of page file (MB) 10004: Overall size of virtual memory (MB) 10005: Available size of virtual memory (MB) 10010: Overall processing unit data index (number of memory blocks) 10011: Used processing unit data index (number of memory blocks) 10012: Processing unit data index available (number of memory blocks) 10014: Used size of processing unit data (MB) 10020: Overall image data index (number of memory blocks) 10021: Used image data index (number of memory blocks) 10022: Image data index available (number of memory blocks) 10024: Used size of image data (MB) 10100: Overall size of UWF overlay (MB) 10101: Available sizes of UWF overlays (MB)

Return value:
Returns the acquired memory size (integer value).

Description:
Gets the information of the type specified by the <kind> parameter. No error will occur even if an undefined value is specified for the <kind> parameter.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:

Checks if the physical memory has a certain amount of available space.

```
If MemoryInfo(5) > 100000000 Then
  Rem Describe the processing to be done if the physical memory has a certain amou
nt of available space.
EndIf
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 6.41 or later

Related Items:
None.

MessageBox

Displays the message box.

Format:
MessageBox <message>

Parameter:

Parameter name	Data type	Description
<message>	Character string	This character strings is displayed on the message box.

Return value:
None.

Description:
The following message box will be displayed.

If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
Type mismatch error is occurred when wrong data is specified as parameter. Illegal function call error is not occurred even if non-exist number, values, combination of data or values.
An error of "String too long" will be occurred when you specify the character strings as character strings type exceeds 255 characters.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
If you run a function in a Unit Macro, the message box may not be displayed on the front. If it is not displayed on the front, connect the USB keyboard to the sensor controller and switch the displayed screen with **Alt** + **Tab**.

Example:
Displays the NG message and forces the measurement to stop when the measurement is NG (failed).

```
Rem Displays the MessageBox message and forces the measurement to stop when the me
asurement is NG (failed).

If totaljudge = -1 then
   MessageBox "NG"
Endif
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 5.40 or later

Related Items:

If Then Elseif Else EndIf on page 8-160

Mid$

Extracts a part from the character string.

Format:
Mid$(<string>, <start>, <length>)

Parameter:

Parameter name	Data type	Description
<string>	Character string	Extraction target character string
<start>	Integer	Starting position of extraction (1 to the length of the target character string)
<length>	Integer	Length of characters to be extracted (1 to the remaining length from specified starting position)

Return value:
Returns the character string type value of the extracted character string.

Description:
Extracts the specified length of the character string in the <length> parameter after the positon specified in the <start> parameter from the <string> parameter.
Specify the length of characters to be extracted in bytes for the <length> parameter. Each single-byte character (i.e., half-width alphanumeric character and symbol) consumes one byte, whereas each doublebyte character consumes two bytes.
If the length specified in the <length> parameter is longer than the length of the character string between the <start> parameter position and the end of the character string, the all characters from the <start> parameter position to the end of the character string are extracted.
If a larger value than the length of the target character string is specified in the <start> parameter, an "Illegal function call" error will occur.
If 0 or smaller value is specified in the <start> parameter or <length> parameter, an "Illegal function call" error will occur.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.

If a value outside the range -2147483648 to 2147483647 is specified as an integer parameter, an "Overflow" error will occur.
If a character string longer than 255 characters is specified for a character string parameter, the 255-character string before the 256th character is used for the macro function processing. Characters after the 256th character will be discarded.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Extracts four characters from the second character and eight characters from the third character in the halfwidth alphanumeric character string. For the latter operation, because the specified length exceeds the target character string length of eight in this example, only five characters from the third to the end of the character string.

```
INPUTSTR$ = "ABCDEFG"

Rem Extract four characters from the second character of the character string.
OUTPUTSTR1$ = Mid$(INPUTSTR$, 2, 4)
Rem Extract eight characters from the third character of the character string.
OUTPUTSTR2$ = Mid$(INPUTSTR$, 3, 8)
```

The result is shown below.

```
OUTPUTSTR1$ = "BCDE"
OUTPUTSTR2$ = "CDEFG"
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

Asc on page 8-34
Chr$ on page 8-44
Hex$ on page 8-158
LCase$ on page 8-179
Left$ on page 8-180
Len on page 8-181
Piece$ on page 8-219
Right$ on page 8-243
Str$ on page 8-314
Str2$ on page 8-315
UCase$ on page 8-339
Val on page 8-352

Mkdir

Builds a directory.

Format:
Mkdir <directoryName>

Parameter:

Parameter name	Data type	Description
<directoryName>	Character string	Directory name of built directory

Return value:
None.

Description:
Build the directory specified in the <directoryName> parameter.
In the <directoryName> parameter, use an absolute path to specify the directory name of the directory to be built.
In the following case, a "Illegal function call" error occurs without building a directory.

- If the specified directory already exists
- If the external storage is specified for where to build a directory in with no external storage inserted
- Free space in the disk drive specified for the directory building is insufficient for building a new directory

If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
An error of "String too long" will be occurred when you specify the character strings as character strings type exceeds 255 characters.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Builds a directory named "IMAGE2" under the root folder of the E drive.

```
Mkdir "E:\IMAGE2"
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

Dskf on page 8-114
GetSystemData on page 8-148
Kill on page 8-176
Fcopy on page 8-129
Isfile on page 8-170
Rmdir on page 8-244

MOD

Gets the remainder.

Format:
<expression1> MOD <expression2>

Parameter:

Parameter name	Data type	Description
<expression1>	Integer	Expression of the dividend to calculate the remainder
<expression2>	Integer	Expression of the divisor to calculate the remainder

Return value:
Returns the remainder as an integer value.

Description:
Gets the remainder from division of the expression specified in the <expression1> parameter by the expression specified in the <expression2> parameter.
If real number expressions are specified, the values are treated as values with digits to the right of the decimal point rounded off.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a value outside the range -2147483648 to 2147483647 is specified as an integer parameter, an "Overflow" error will occur.
If 0 is specified in the <expression2> parameter, a "Division by zero" error will occur.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Increments a counter from 0 to 100.

```
I& = (I&+1) MOD 100
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

Fix on page 8-130
GetUnitData on page 8-153
UnitData on page 8-341

MoveUnit

Moves a processing unit.

Format:
MoveUnit <srcUnitNo>, <destUnitNo>

Parameter:

Parameter name	Data type	Description
<srcUnitNo>	Integer	Processing unit number (0 to (number of processing units of current scene minus one)) of source
<destUnitNo>	Integer	Processing unit number (0 to (number of processing units of current scene minus one)) indicating the position of the destination in the measurement flow

Return value:
None.

Description:
Moves the processing unit specified in the <srcUnitNo> parameter to the position in the measurement flow specified in the <destUnitNo> parameter.
The processing unit numbers of processing units after the processing unit number specified in the <srcUnitNo> parameter are moved up by 1. The processing unit numbers of processing units after the processing unit number specified in the <destUnitNo> parameter are moved down by 1.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
- Execute this macro function when the BUSY signal or other measurement in progress signal is ON and measurement is prohibited. (For details, refer to *5-1-4 State Transitions and Execution Timing* on page 5-13.)

Example:
Moves the processing unit of Processing Unit number 2 between Processing Unit number 5 and Processing Unit number 6.

```
MoveUnit 2, 5
```

Usable Modules:
Scene Control Macro / Communication Command Macro

Supported Versions:
Version 3.50 or later

Related Items:

AssignUnit on page 8-35
CopyUnit on page 8-58
InsertUnit on page 8-168
CheckUnit on page 8-43
DeleteUnit on page 8-72
MeasureStart on page 8-200

MeasureStop on page 8-201
UnitCount on page 8-340

NOT

Gets the "not" result (negation) of the expression.

Format:
NOT(<expression>)

Parameter:

Parameter name	Data type	Description
<expression>	Integer	Expression to calculate the negation

Return value:
Returns an integer "not" value.

Description:
Gets the "not" result of the 32-digit binary value specified in the <expression> parameter by inverting each bit.
If a real number expression is specified in the <expression> parameter, the value is treated as a value with digits to the right of the decimal point rounded off.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a value outside the range -2147483648 to 2147483647 is specified as an integer parameter, an "Overflow" error will occur.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Gets the "not" value of the integer 0 assigned to the variable X&.

```
X& = 0
XX& = NOT(X&)
```

The result is shown below.

```
XX& = -1
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

AND on page 8-28
GetUnitData on page 8-153
UnitData on page 8-341
OR on page 8-218
XOR on page 8-365

Open For Append As#

Open the file in append mode.

Format:
Open <fileName> For Append As #<fileNo>

Parameter:

Parameter name	Data type	Description
<fileName>	Character string	Absolute path of the file to be opened
<fileNo>	Integer	Assigned file number (0 to 15) to the opened file

Return value:
None.

Description:
Assigns the specified file number in the <fileNo> parameter to the specified file by the <fileName> parameter and open the file in append mode.
Be sure to execute the Close function to close the file opended with this macro function within the same subroutine. File accessing processes such as data writing to a file and data reading from a file may not be completed properly in the following cases.
The Close function is not executed. The Close function is used in a different subroutine from where this macro function is executed.
This macro function is executed at a different timing from the Open function execution.
In the <fileName> parameter, use an absolute path to specify the file name of the file to be opened.
If the file of the file name specified in the <fileName> parameter does not exist, the file is newly created.
If the file of the file name specified in the <fileName> parameter already exists, open the existing file so that the additional writing in the file is possible.
If a value outside the range of 0 to 15 is specified in the <fileNo> parameter, an "Illegal function call" error will occur.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
An error of "String too long" will be occurred when you specify the character strings as character strings type exceeds 255 characters.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Opens the file named "input.txt" under the E drive and writes the data to the file.

```
STRING$ = "Sample"

Rem Open the file.
Open "E:\input.txt" For Append As #1

Print #1, STRING$

Rem Close up the file.
Close #1
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

Close on page 8-49
GetSystemData on page 8-148
Input$ on page 8-166
Line Input# on page 8-182
Open For Output As# on page 8-213
Eof on page 8-116
Input# on page 8-165
Isfile on page 8-170
Open For Input As# on page 8-212
Print# on page 8-222

Open For Input As#

Open the file in reading mode.

Format:
Open <fileName> For Input As #<fileNo>

Parameter:

Parameter name	Data type	Description
<fileName>	Character string	Absolute path of the file to be opened
<fileNo>	Integer	Assigned file number (0 to 15) to the opened file

Return value:
None.

Description:
Assigns the specified file number in the <fileNo> parameter to the specified file by the <fileName> parameter and open the file in reading mode.
Be sure to execute the Close function to close the file opended with this macro function within the same subroutine. File accessing processes such as data writing to a file and data reading from a file may not be completed properly in the following cases.
The Close function is not executed. The Close function is used in a different subroutine from where this macro function is executed.
This macro function is executed at a different timing from the Open function execution.
In the <fileName> parameter, use an absolute path to specify the file name of the file to be opened.
If the file of the file name specified in the <fileName> parameter does not exist, an "Illegal function call" error will occur.

If a value outside the range of 0 to 15 is specified in the <fileNo> parameter, an "Illegal function call" error will occur.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
An error of "String too long" will be occurred when you specify the character strings as character strings type exceeds 255 characters.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Reads the data until the end of the file.

```
Dim ALLDATA$(255)

Rem Open the file.
Open "E:\input.dat" For Input As #1

For I&=0 to 255
   Rem Read line by line from the top of the file.
   Input #1, DATA$
   ALLDATA$(I&) = DATA$

   Rem Check if the end of the file is reached.
   If Eof(1) <> 0 Then
      Exit For
   Endif
Next

Rem Close up the file.
Close #1
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

Close on page 8-49
GetSystemData on page 8-148
Input$ on page 8-166
Line Input# on page 8-182
Open For Output As# on page 8-213
Eof on page 8-116
Input# on page 8-165
Isfile on page 8-170
Open For Append As# on page 8-211
Print# on page 8-222

Open For Output As#

Opens the file in writing mode.

Format:
Open <fileName> For Output As #<fileNo>

Parameter:

Parameter name	Data type	Description
<fileName>	Character string	Absolute path of the file to be opened
<fileNo>	Integer	Assigned file number (0 to 15) to the opened file

Return value:
None.

Description:
Assigns the specified file number in the <fileNo> parameter to the specified file by the <fileName> parameter and open the file in writing mode.
Be sure to execute the Close function to close the file opended with this macro function within the same subroutine. File accessing processes such as data writing to a file and data reading from a file may not be completed properly in the following cases.
The Close function is not executed. The Close function is used in a different subroutine from where this macro function is executed.
This macro function is executed at a different timing from the Open function execution.
In the <fileName> parameter, use an absolute path to specify the file name of the file to be opened.
If the file of the file name specified in the <fileName> parameter does not exist, an "Illegal function call" error will occur.
If the file of the file name specified in the <fileName> parameter already exists, the existing file content is once deleted, then opens the existing file so that the writing in the file is possible.
If a value outside the range of 0 to 15 is specified in the <fileNo> parameter, an "Illegal function call" error will occur.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
An error of "String too long" will be occurred when you specify the character strings as character strings type exceeds 255 characters.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Opens the file, writes the data in the file, and then closes the file.

```
DATA& = 10

Rem Open the file.
Open "E:\input.dat" For Output As #1

Rem Write the data in the opened file
Print #1 DATA&

Rem Close the opened file
Close #1
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:

Version 3.50 or later

Related Items:

Close on page 8-49
GetSystemData on page 8-148
Input$ on page 8-166
Line Input# on page 8-182
Open For Input As# on page 8-212
Eof on page 8-116
Input# on page 8-165
Isfile on page 8-170
Open For Append As# on page 8-211
Print# on page 8-222

OpenTextData

Opens a messages file.

Format:
OpenTextData <ident> As #<textDataNo>

Parameter:

Parameter name	Data type	Description
<Ident>	Character string	Identification name for the message file that is opened as text data
<textDataNo>	Integer	Text data number (0 to 15) that is assigned to the message file opened as text data

Return value:
None.

Description:
- Message file defines the displayed messages. The message file is configured one file for each languages.
- The message file name is configured as the below.
  <Message file Data ident>_<Language Data ident>.msg
  <Message file Data ident> indicates an ident specified <ident> in OpenTextData function.
  <Language Data ident> is a character string.
  Each languages and its Data ident are as the following:
  Simplified Chinese: chs
  Traditional Chinese: cht
  German: deu
  English: eng
  Spanish: esp
  French: fra
  Italian: ita
  Japanese: jpn
  Korean: kor
- A Data ident specified <textIdent> for each lines and a text data which is corresponded to <textIdent>.
  <Data ident 1> = <Text data1>
  <Data ident 2> = <Text data 2>
  <Data ident 3> = <Text data 3>
  :

<Data ident n> = <Text data n>

Acquires the specified text data in <textIdent> parameter from the message file of the text data No. which is specified in <textDataNo> parameter. Assigns the specified text data number in the <textDataNo> parameter to the specified file by the <ident> parameter and opens the file.
Be sure to execute the CloseTextData function to close the file opended with this macro function within the same subroutine. The message file cannot properly be closed and this macro function may not properly be executed in the subsequent processes in the following cases.
The CloseTextData function is not executed.
This CloseTextData function is used in a different subroutine from where this function is executed.
This CloseTextData function is executed at a different timing from this function execution.
If a value outside the range from 0 to 15 is specified in the <textDataNo> parameter, an "Illegal function call" error will occur
If the text data number that is already opened is specified in the <textDataNo> parameter, an "Illegal function call" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
An error of "String too long" will be occurred when you specify the character strings as character strings type exceeds 255 characters.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:

None.

Example:

Uses the *MEASUREDISPT subroutine of the Unit Macro processing unit to display the measured correlation value by the search processing unit (Processing Unit number 5), along with the gotten text string from the prepared message file for the processing unit, in the text window. The correlation value can be gotten with External Reference Data number 5.

```
*MEASUREDISPT

   Rem Get the measurement result.
   GetUnitData 5, 5, CR#

   Rem Open the messages file
   OpenTextData "Search" As #1

   Rem Get the text
   TEXT$ = GetText$(#1, "Correlation")

   Rem Draw the gotten text string from the messages file without adding any line
break on the text window.
   DrawText TEXT$, UnitJudge(5), 0

   Rem Draw the measurement results on the text window.
   DrawText Str2$(CR#, 4, 4, 0, 0), UnitJudge(5), 1

   Rem Close up the messages file.
   CloseTextData

Return
```

The result is shown below.

```
Correlation value: 90.0000
```

Usable Modules:
Unit Calculation Macro / Unit Macro

Supported Versions:
Version 5.00 or later

Related Items:

CloseTextData on page 8-50
DrawText on page 8-109
GetText$ on page 8-150
GetUnitData on page 8-153
UnitJudge on page 8-348

Option Explicit

Finds undefined or duplicate variable that is defined in the Dim variable format.

Format:
Option Explicit

Parameter:
None.

Return value:
None.

Description:
When you execute the Option Explicit command, only defined variables will be available to use, and duplicate of the defined variables will be checked.

Usage Cautions:
- When you execute the Option Explicit command, the Macro Variable Check function will be effective for lines after the line where the Option Explicit command is executed.
- Undefinition and duplicates of variables in preceding lines will not be checked. In other word, they are not the subject to the "Un-defined variable2 error or "Duplicated variable" error.
- The Option Explicit command is unavailable when the power is on or when macros are loaed. It is available from the time a macro and the Option Explicit command are both executed until the next macro is loaded or the power is turned off.

Example:
Option Explicit can be used to check macro variables.

```
Option Explicit
Rem Definition of variable by the Dim variable
Dim A&
Dim B&
Dim C&
A& = 0
B& = 1
C& = A& + B&
Print Str$(C&)
```

Usable Modules:
Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 5.40 or later

Related Items:

Dim on page 8-73
ReDim on page 8-233
VarList on page 8-353

OR

Gets the logical sum of two expressions.

Format:
<expression1> OR <expression2>

Parameter:

Parameter name	Data type	Description
<expression1>	Integer	Expression to calculate the logical sum
<expression2>	Integer	

Return value:
Returns the logical sum as an integer value.

Description:
Gets the logical sum by bit of the expression specified in the <expression1> parameter and the expression specified in the <expression2> parameter.
When the values of the <expression1> parameter and <expression2> parameter are double precision real values, the decimal part is rounded off.
This can also be used as an Or condition in an If statement. For details on the logical expression, refer to *4-1-5 Operator* on page 4-12.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a value outside the range -2147483648 to 2147483647 is specified as an integer parameter, an "Overflow" error will occur.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Gets the logical sum of 1 assigned to the variable EXP1& and 4 assigned to the variable EXP2&.

```
EXP1& = 1
EXP2& = 4

EXPALL& = EXP1& OR EXP2&
```

The result is shown below.

```
EXPALL& = 5
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

AND on page 8-28
GetUnitData on page 8-153
UnitData on page 8-341
NOT on page 8-210
XOR on page 8-365

Piece$

Extract the part of the character string which was separated by delimiter from the string.

Format:
Piece$(<string>, <delimiter>, <start>, <end>)

Parameter:

Parameter name	Data type	Description
<string>	Character string	Extraction target character string
<delimiter>	Character string	Delimiter
<start>	Integer	Index number of the character string that the extraction is started (Number 1 to number of substrings)
<end>	Integer	Index number of the character string that the extraction is finished (Number 1 to number of substrings)

Return value:
Returns the character string type value of the extracted character string.

Description:
Extracts the character string portions from the starting index number specified in the <start> parameter to the ending index number specified in the <end> parameter after assigning index numbers to the portions separated with the separator character specified in the <delimiter> parameter.
If the character string in the <string> parameter cannot be separated with the character character in the <delimiter> parameter, all characters in the character string is extracted as a portion.
If a character string longer than 255 characters is specified in the <string> parameter, the 255-character string before the 256th character is used for the macro function processing. Characters after the 256th character will be discarded.
If a character string exceeding one character is specified in the <delimiter> parameter, the substring cannot be extracted correctly.
Specify the starting index number to be extracted in the <start> parameter. The index numbers are assigned to the portions in ascending order starting with 1 to the first portion.
If a larger value than the number of separated portions is specified in the <start> parameter, an "Illegal function call" error will occur.

If the index number specified in the <end> parameter is larger than the index number specified in the <start> parameter, an "Illegal function call" error will occur.
If 0 or smaller value is specified in the <start> parameter or <end> parameter, an "Illegal function call" error will occur.
If a larger value than the number of separated portions is specified in the <end> parameter, all portions from the starting index number in the <start> parameter to the end of the character string are extracted.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a value outside the range -2147483648 to 2147483647 is specified as an integer parameter, an "Overflow" error will occur.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Gets the part of the character string which was separated by a semicolon (;) delimiter from the string.

```
INPUTSTR$ = "PIECE1;PIECE2;PIECE3;PIECE4"
DELIMITER$ = ";"

Rem Extract the first substring of the character string.
OUTPUTSTR1$ = Piece$(INPUTSTR$, DELIMITER$, 1, 1)
Rem Extract the third and forth substrings from the character string.
OUTPUTSTR2$ = Piece$(INPUTSTR$, DELIMITER$, 3, 4)
```

The result is shown below.

```
OUTPUTSTR1$ = "PIECE1"
OUTPUTSTR2$ = "PIECE3;PIECE4"
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

Asc on page 8-34
Chr$ on page 8-44
Hex$ on page 8-158
LCase$ on page 8-179
Left$ on page 8-180
Len on page 8-181
Mid$ on page 8-205
Right$ on page 8-243
Str$ on page 8-314
Str2$ on page 8-315
UCase$ on page 8-339
Val on page 8-352

Print

Outputs data in the system status console window.

Format:
Print <expression>[;|, <expression>...][;|,]

Parameter:

Parameter name	Data type	Description
<expression>	Integer Real number Character string Array	Numerical expression or character string to be output

Return value:
None.

Description:
Outputs the numerical expression or character string specified in the <expression> parameter to the system status console window. (For details, refer to *3-1-2 Description of the System Status Console Window* on page 3-7.)
If the parameters are separated with commas (,), the specified expressions and character strings in the <expression> parameters are separated by tab delimiters and output.
If the parameters are separated with semicolons (;), the specified expressions and character strings in the <expression> parameters are output subsequently to the output expression or text string immediately before.
If any of semicolon (;) and comma (,) are specified at the end of parameters, a line break is added after all the parameters are output to the system status console window.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
- After the data output to the system status console window, the window is displayed on top of the sensor controller main screen. To display the system status console window on top of the main screen, click [_] on the upper-right of the system status console window or press [Alt] + [Tab] on the connected USB keyboard to the sensor controller.

Example:
Gets the gotten values of "correlation value", "measurement coordinate X", and "measurement coordinate Y" by the search processing unit (Processing Unit number 2) and outputs to the system status console window. The correlation value, measured position coordinates X and Y can be gotten with External Reference Data numbers 5, 6, and 7 respectively.

```
Rem Get the measurement result.
GetUnitData 2, 5, CR#
GetUnitData 2, 6, X#
GetUnitData 2, 7, Y#

Rem Output the gotten measurement results to the system status console window.
Print CR#; ","; X#; ","; Y#
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

Cls on page 8-52
Cont on page 8-53
Debug on page 8-68
DebugPrint on page 8-69
GetUnitData on page 8-153
List on page 8-184
SetStop on page 8-293
SetVar on page 8-307
Stop on page 8-312
SubList on page 8-320
VarList on page 8-353

Print#

Outputs data in a file.

Format:
Print #<fileNo>[, <expression>[;|, <expression>...][;|,]

Parameter:

Parameter name	Data type	Description
<fileNo>	Integer	File number (0 to 15) of the output destination file
<expression>	Integer Real number Character string Array	Numerical expression or character string to be output

Return value:
None.

Description:
Outputs an expression or a character string specified in the <expression> parameter in the file with the file number specified in the <fileNo> parameter.
If the parameters are separated with commas (,), the specified expressions and character strings in the <expression> parameters are separated by tab delimiters and output.
If the parameters are separated with semicolons (;), the specified expressions and character strings in the <expression> parameters are output subsequently to the output expression or text string immediately before.
If any of semicolon (;) and comma (,) are specified at the end of parameters, a line break is added after all the parameters are output.
If an unopened file number is specified in the <fileNo> parameter, an "Illegal function call" error will occur.
If a file number of the opened file by a macro function other than the Open For Append As# function and the Open For Output As# function is specified in the <fileNo> parameter, an "Illegal function call" error will occur.
If a value outside the range of 0 to 15 is specified in the <fileNo> parameter, an "Illegal function call" error will occur.

If a character string longer than 127 characters is specified in the <expression> parameter, an "Illegal function call" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Outputs a character string to the file "E:\input.txt".

```
STRING$ = "Sample"

Rem Open the file.
Open "E:\input.txt" For Append As #1

Print #1, STRING$

Rem Close up the file.
Close #1
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

Close on page 8-49
Open For Output As# on page 8-213
Open For Append As# on page 8-211

PutAll

Sets the output state of all output terminals.

Format:
PutAll <ioIdent>, <state>

Parameter:

Parameter name	Data type	Description
<ioIdent>	Character string	Identification name of the communication module to be used (always "ParallelIo") (Refer to *9-4 List of I/O Modules* on page 9-27.)
<state>	Integer	Output state of terminal

Return value:
None.

Description:
Sets the output state of all output terminals of the communication module specified in the <ioIdent> parameter to the state specified in the <state> parameter.
Normally "ParallelIo" should be specified in the <ioIdent> parameter.

The output state of each output terminal is expressed as an integer value (OFF (0) or ON (1)) in each digit of a character string in binary notation.
In parallel I/O, integer values are returned expressing DO0 to DO15 in the 1st digit to the 16th digit.
Example: When DO0 to DO5 are ON and DO6 to DO15 are OFF
• Binary notation: 0000 0000 0011 1111
• Value of output states that are set: 63
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If a value outside the range -2147483648 to 2147483647 is specified as an integer parameter, an "Overflow" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:

- When the operation mode is multi-line random trigger and there are three or more lines, the output states of the DO signal cannot be set. Even if this macro function is used when there are three or more lines, an "Illegal function call" error will not occur.

Example:

In the communication command macro, sets all DO output states of parallel I/O to ON.

```
IOMODULE$ = "ParallelIo"

Rem Set the output state.
PutAll IOMODULE$, 65535
```

Usable Modules:

Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:

Version 3.50 or later

Related Items:

BusyOut on page 8-37
GetPort on page 8-144
PutPort on page 8-225
GetAll on page 8-134
JudgeOut on page 8-175
RunOut on page 8-245

PutPort

Sets the output state of the specified output terminal.

Format:
PutPort <ioIdent>, <portNo>, <state>

Parameter:

Parameter name	Data type	Description
<ioIdent>	Character string	Identification name of the communication module to be used (always "ParallelIo") (Refer to *9-4 List of I/O Modules* on page 9-27.)
<portNo>	Integer	Terminal number of output terminal whose output state is to be set. Parallel I/O: • FH DO0 to DO15: 0 to 15 GATE N: 100 + N x 8 (N: Line number (0 to 7)) BUSY N: 101 + N x 8 OR N: 102 + N x 8 ERROR N: 103 + N x 8 RUN N: 104 + N x 8 READY N: 105 + N x 8 ACK: 200
<state>	Integer	Output state of output terminal 0: Output OFF 1: Output ON

Return value:
None.

Description:
Sets the state of the output terminal of the terminal number specified in the <portNo> parameter of the communication module specified in the <ioIdent> parameter to the output state specified in the <state> parameter.
Normally "ParallelIo" should be specified in the <ioIdent> parameter.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified in any of the <ioIdent> and <portNo> parameters, an "Illegal function call" error will occur.
Even if an output status parameter value that does not exist (i.e., other than 0 and 1) is specified in the <state> parameter, an error will not occur.
If a value outside the range -2147483648 to 2147483647 is specified as an integer parameter, an "Overflow" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
- Do not execute at the same time as other parallel output control.

Example:
In the communication command macro, sets the output state of DO3 of parallel I/O to ON.

```
IOMODULE$ = "ParallelIo"

Rem Set the output state.
PutPort IOMODULE$, 3, 1
```

Usable Modules:

Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:

Version 3.50 or later

Related Items:

BusyOut on page 8-37
GetPort on page 8-144
PutAll on page 8-223
GetAll on page 8-134
JudgeOut on page 8-175
RunOut on page 8-245

RaiseErrorProcEvent

Notifies the error processing on UI screen.

Format:

RaiseErrorProcEvent <errorKind>, <parameter>[, <wait>]

Parameter:

Parameter name	Data type	Description
<errorKind>	Integer	Error number to notify the error processing. 0: System error 1: System error (Fan or voltage error) 3: Battery error 4: UWF overlay depletion 10: Camera connection error 11: Connected camera has been changed 12: Detection of camera over current 13: Configuration error of light device connection 20: Write error of logging disk 30: Time out of parallel output 31: PLC link communication error 32: Detection of parallel I/O camera over current 40: Data load error 41: Data transfer error 42: Incorrect number of start-up Scene group 43: Incorrect number of start-up Scene
<parameter>	Integer	Error processing parameter to notify This parameter is not used in this version. Therefore, specify to 0.

Parameter name	Data type	Description
<wait>	Integer	Recovery timing of macro function 0: Executes subsequent program lines without waiting for measurement to end. 1: Waits for measurement to end and then executes subsequent program lines. 2: Waits for measurement to end and measurement result display to end and then executes subsequent program lines. 3: Waits for measurement to end, measurement result display to end, and UI display update to end then executes subsequent program lines.

Return value:

None.

Description:

Notify the error processing of error number specified <errorKind> and a parameter specified <parameter> on the PanDA window. What process is executed on the UI screen that receives the error processing event depends on the process specification of the UI screen.
<parameter> is not used in this version. Therefore, specify to 0.
<wait> is optional.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
Type mismatch error is occurred when wrong data is specified as parameter. Illegal function call error is not occurred even if non-exist number, values, combination of data or values.
If a value outside the range -2147483648 to 2147483647 is specified as an integer parameter, an "Overflow" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:

- Do not write to *MCRINIT.

Example:

Communication Command Macro example.
When the logging error is occurred, ERROR signal is not turned to ON, and notices the error processing of event. When other error is occurred, normal error processing is executed.

```
*ERRORPROC

   If ERRORKIND& <>20 Then
      Rem When the logging error is occurred, ERROR signal is not turned to ON, an
d notices the error processing of event.
      ERROROUT "",ERRORKIND&
   Endif

   Rem Notices the error processing event on the window.
   RaiseErrorProcEvent ERRORKIND&, 0, 0

Return
```

Usable Modules:

Scene Control Macro / Communication Command Macro

Supported Versions:

Version 5.40 or later

Related Items:

ErrorOut on page 8-121
MessageBox on page 8-204
If Then Else on page 8-159

RaiseOptionEvent

Notifies option events to the UI screen.

Format:
RaiseOptionEvent <eventNo>, <parameter>

Parameter:

Parameter name	Data type	Description
<eventNo>	Integer	Event number to be notified 255: Layout switch in the main screen of both sensor controller and remote operation tool 511: Layout switch in the main screen of the sensor controller 767: Layout switch in the main screen of the remote operation tool
<parameter>	Integer	Event parameter to be notified If the event is layout switch: 0 to 127: New layout number

Return value:
None.

Description:
Notifies the UI screen of the event specified in the <eventNo> parameter and the parameter specified in the <parameter> parameter.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
Type mismatch error is occurred when wrong data is specified as parameter. Illegal function call error is not occurred even if non-exist number, values, combination of data or values.
If a value outside the range -2147483648 to 2147483647 is specified as an integer parameter, an "Overflow" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.
Through the use of the FH-AP1 Application Producer (sold separately), it is possible to create a new event in the UI screen.

Usage Cautions:
Execute this macro function when the BUSY signal or other measurement in progress signal is ON and measurement is prohibited. (For details, refer to *5-1-4 State Transitions and Execution Timing* on page 5-13.)

Example:
Switches the main screen layout of the sensor controller when the BUSY flag set to ON in the communications command macro.

```
Rem Switch to layout 1.
RaiseOptionEvent 511, 1
```

Usable Modules:
Scene Control Macro / Communication Command Macro

Supported Versions:
Version 3.50 or later

Related Items:
None.

ReadPlcMemory

Reads a value from the PLC memory area.

Format:
ReadPlcMemory <ioIdent>, <area>, <channelOffset>, <channelCount>, <readData()>

Parameter:

Parameter name	Data type	Description
<ioIdent>	Character string	Identification name of the communication module to be used (Refer to *9-4 List of I/O Modules* on page 9-27.)
<area>	Integer	Area classification number of data output area to read in the target
<channelOffset>	Integer	Offset from beginning of data output area to address where reading is to start
<channelCount>	Integer	Data size to read in (channel unit)
<readData()>	Integer array	Loaded data

Return value:
None.

Description:
Using the communication module specified in the <ioIdent> parameter, the data size specified in the <channelCount> parameter is read from the address that is offset by the value specified in the <channelOffset> parameter, of the PLC area type specified in the <area> parameter.
After reading the value from the PLC memory area using this macro function, execute the GetPlcData function to get the read value.
In the <readData()> parameter, specify the 1D integer array variable that stores the data that was read. Add () without specifying element numbers.
In the <area> parameter, specify the identification of the register that is set with the PLC link setting in the system settings. For Fieldbus, specify 0 because the identification cannot be specified.
In the <channelCount> parameter, specify the size in channel units. The size of one integer type data item is two channels (four bytes), and thus to read one integer value, a one-element array should be prepared with the <readData()> parameter, and 2 should be specified in the <channelCount> parameter.
If a size larger than the array size specified in the <readData()> parameter is specified in the <channelCount> parameter, a "Subscript out of range" error will occur.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.

If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:

- After using this macro function to read data, always use the GetPlcData function to get the value from the data that was read. If the value is gotten directly from the <readData()> parameter without using the GetPlcData function, the correct value may not be gotten.

Example:

In the communication command macro, reads multiple data from the PLC connected by PLC link.

```
IOMODULE$ = "UdpPlcLink"

Rem Get the settings of the output data area.
GetSystemData IOMODULE$, "outputArea", AREA&
GetSystemData IOMODULE$, "outputMemoryAddress", ADDRESS&

Rem Create the integer array variable to store the read data.
Dim DATA&(1)

Rem Load the data (4ch) from data output area.
ReadPlcMemory IOMODULE$, AREA&, ADDRESS&, 4, DATA&()

Rem Get the values from the read data.
GetPlcData IOMODULE$, DATA&(), 0, 4, VALUE0&
GetPlcData IOMODULE$, DATA&(), 4, 4, VALUE1&
```

Usable Modules:

Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:

Version 4.20 or later

Related Items:

GetPlcData on page 8-140
SetPlcData on page 8-281
WritePlcMemory on page 8-363

ReceiveData

Receives data.

Format:

ReceiveData <ioIdent>, <inputData()>, <inputMaxSize>, <inputSize>[, <parameter()>, <parameterSize>]

Parameter:

Parameter name	Data type	Description
<ioIdent>	Character string	Identification name of the communication module to be used (Refer to *9-4 List of I/O Modules* on page 9-27.)
<inutData()>	Integer array	Received data
<inputMaxSize>	Integer	Maximum size of data to be received
<inputSize>	Integer	Received data size
<parameter()>	Integer array	The parameter data specified in the option
<parameterSize>	Integer	Size of parameter data specified optionally

Return value:

None.

Description:

Receive data up to the size specified in the <inputMaxSize> parameter by using the communication module specified in the <ioIdent> parameter.
In the <inputData()> parameter, specify the 1D integer array variable that will hold the data to be received, without adding element numbers but adding () to the variables. In the <inputSize> parameter, specify the integer variable that will hold the size of the received data.
Values that can be set in the <parameter()> parameter and <parameterSize> parameter depend on the communication module specified in the <ioIdent> parameter. The <parameter()> parameter and <parameterSize> parameter can be omitted.
With this macro function, data that has arrived at the sensor controller from an external device is received when the macro function is executed. If no data has arrived, 0 is stored in the <inputSize> parameter. If the desired data is not received the first time the macro function is executed, execute repeatedly until all data has been received.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If Handshake is on and data reception fails due to a communication timeout or other reason, an "Illegal function call" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:

- Execute this macro function when the target communication module is stopped. (Refer to *5-1-5 Exclusive Control in a Process* on page 5-20.)

Example:

Receives data from an external device in the measurement process of the unit macro.

```
Rem Prepare a buffer that can receive 12 bytes of data.
Dim BUFFER&(11)
IOMODULE$ = "TcpNormal"

Rem Set the polling state of the communication module to stopped in order to recei
ve the data.
SetPollingState IOMODULE$, False

Rem Executing the initialization of the reception data size.
SIZE& = 0

Rem Repeat the reception process until the data has been received.
Try
   Do
      Rem Attempting the data reception.
      ReceiveData IOMODULE$, BUFFER&(), 12, SIZE&

      Rem Once the data has been received, display the data size in the system sta
tus console window.
      If(SIZE& > 0) Then
         Print "Received data size = " + Str$(SIZE&)
      Endif
   Loop While SIZE& = 0

   Rem Data has been received, so set the polling state of the communication modul
e to running.
   SetPollingState IOMODULE$, True

Catch
   Rem Return the polling state of the stopped communication module to running.
   If GetPollingState(IOMODULE$) = False Then
      SetPollingState IOMODULE$, True
   Endif

End Try
```

Usable Modules:

Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:

Version 3.50 or later

Related Items:

GetPollingState on page 8-142
SendData on page 8-268
SendString on page 8-270
SetPollingState on page 8-283

ReDim

Defines the array.

Format:

ReDim <arrayName>(<maxCount>[, <maxCount>[, <maxCount>[, <maxCount>]]])

Parameter:

Parameter name	Data type	Description
<arrayName>	-	Used array variable name
<maxCount>	Integer	Maximum value of the subscript

Return value:

None.

Description:

Defines a 1D to 4D array with maximum dimensional length specified in the <maxCount> parameter for each dimension.
Add one of type identifiers to the end of the parameter specified in the <arrayName>. (Refer to *4-1-3 Variable* on page 4-6.)
Release the array variables defined with this macro function by executing the Erase function.
If the number of array dimension is different, two arrays with the same variable name are treated as the same variable.
An array variable and a variable with the same name are treated as different variables.
If the number of elements of an array variable is undefined, a "Syntax error" will occur.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:

- The behavior changes depending on the use of the Option Explicit command.
  When the Option Explicit command is used, and if variables that are not predefined by the Dim function are redefined, an error will occur for the undefined variables when loading macros.
  When the Option Explicit command is not used, predefinition by the Dim function is not required. Array variables are defined the same as when using the Dim function.

Example:

Defines the array.

```
Dim AA&()
ReDim AA&(10)
Dim BB&(3)
Redim BB&(10)
```

Usable Modules:

Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:

Version 5.40 or later

Related Items:

Dim on page 8-73
Option Explicit on page 8-217
Erase on page 8-118

RefreshImageWindow

Updates the image window.

Format:
RefreshImageWindow

Parameter:
None.

Return value:
None.

Description:
Updates the image window.
Execute this macro function to update the display after executing the ImageUpdate function when a graphic is redrawn in the image window or the image is redrawn.

Usage Cautions:
None.

Example:
In the communication command macro, updates an image displayed in *Camera Image Freeze* image mode to the most recent image.

```
Rem Update the freeze image to the most recent image.
ImageUpdate

Rem Apply the updated image to the display in the image window.
RefreshImageWindow
```

Usable Modules:
Scene Control Macro / Communication Command Macro

Supported Versions:
Version 3.50 or later

Related Items:

ImageUpdate on page 8-164
RefreshJudgeWindow on page 8-235
RefreshTextWindow on page 8-236
RefreshTimeWindow on page 8-237
SetImageWindow on page 8-276

RefreshJudgeWindow

Updates the judgement window.

Format:
RefreshJudgeWindow

Parameter:
None.

Return value:
None.

Description:
Updates the judgement window.

Usage Cautions:
None.

Example:
Updates the judgement window in the communication command macro.

```
RefreshJudgeWindow
```

Usable Modules:
Scene Control Macro / Communication Command Macro

Supported Versions:
Version 3.50 or later

Related Items:

RefreshImageWindow on page 8-234
RefreshTimeWindow on page 8-237
RefreshTextWindow on page 8-236

RefreshTextWindow

Updates the text display window.

Format:
RefreshTextWindow

Parameter:
None.

Return value:
None.

Description:
Updating the text display window.
Execute this macro function to update the display after redrawing detailed results in the text window.

Usage Cautions:
None.

Example:
In the communication macro, sets the value of the measurement result of calculation expression 0 of the calculation processing unit of Processing Unit number 5 of the current scene, and then updating the display of the text window."Calculation result of calculation expression 0" is external reference data number 5.

```
Rem Set the value as the measurement result.
SetUnitData 5, 5, 100

Rem Update the text window display.
RefreshTextWindow
```

Usable Modules:
Scene Control Macro / Communication Command Macro

Supported Versions:
Version 3.50 or later

Related Items:

RefreshImageWindow on page 8-234
RefreshJudgeWindow on page 8-235
RefreshTimeWindow on page 8-237
SetTextWindow on page 8-299

RefreshTimeWindow

Updates the display of the information window.

Format:
RefreshTimeWindow

Parameter:
None.

Return value:
None.

Description:
Updates the display of the information window.

Usage Cautions:
None.

Example:
Updates the display of the information window in the communication macro.

```
RefreshTimeWindow
```

Usable Modules:
Scene Control Macro / Communication Command Macro

Supported Versions:
Version 3.50 or later

Related Items:

RefreshImageWindow on page 8-234
RefreshJudgeWindow on page 8-235
RefreshTextWindow on page 8-236
SetTextWindow on page 8-299

Rem

Puts a comment in the program.

Format:
Rem

Parameter:
None.

Return value:
None.

Description:
Add a comment or a description in the program. The readability of the program is improved by adding comments.
For details on comment, refer to the *Comment* on page 4-4.

Usage Cautions:
- Do not mix a non-comment statement together with a comment on the same line. If a comment and another type of statement are written in one same line, the comment may not be correctly recognized and the program may not operate properly.

Example:
Inserts a comment statement in a program to describe the program process.

```
Rem Display the judgement result provided by the latest executed processing unit o
n the system status console window.
Print UnitJudge(UnitNo - 1)
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

Print on page 8-221
UnitJudge on page 8-348
UnitNo on page 8-349

Remeasure

Executes remeasurement.

Format:
Remeasure <preImageNo>[, <wait>]
Remeasure <fileName>[, <wait>]

Parameter:

Parameter name	Data type	Description
<preImageNo>	Integer	Image Logging No. which is re-measured. (0 to the number of Image logging which has been logginged.)
<fileName>	Character string	Image file name of image to be remeasured.
<wait>	Integer	Recovery timing of macro function 0: Executes subsequent program lines without waiting for measurement to end. 1: Waits for measurement to end and then executes subsequent program lines. 2: Waits for measurement to end and measurement result display to end and then executes subsequent program lines. 3: Waits for measurement to end, measurement result display to end, and UI display update to end then executes subsequent program lines.

Return value:
None.

Description:
Executes remeasurement at the recovery timing specified in the <wait> parameter for the image of the image logging number specified in the <preImageNo> parameter or the image of the image file name specified in the <fileName> parameter.
In the <preImageNo> parameter, specify the number of the logging image already logged as a logging image in the main unit. (For details, refer to *Setting Logging Conditions [Logging Setting]* in the *Vision System FH/FHV Series User's Manual (Cat. No. Z365)*.)
If 0 is specified for the <preImageNo> parameter, remeasurement is executed using the most recent logged image of the main unit.
In the <fileName> parameter, use an absolute path to specify the image file name of image to be remeasured.
If the <wait> parameter is omitted, operation is the same as when the <wait> parameter is set to 0.
When 0 is specified for the <wait> parameter, there is a possibility that the measurement processing executed immediately after execution of this macro function will not have ended and the measurement result cannot be properly gotten. If you want to get the measurement result, specify 1 or 2 or 3 for the <wait> parameter.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
Type mismatch error is occurred when wrong data is specified as parameter. Illegal function call error is not occurred even if non-exist number, values, combination of data or values.
If a value outside the range -2147483648 to 2147483647 is specified as an integer parameter, an "Overflow" error will occur.
An error of "String too long" will be occurred when you specify the character strings as character strings type exceeds 255 characters.

If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:

- Executes this function in the status when measure signal is OFF.(For details, refer to *5-1-4 State Transitions and Execution Timing* on page 5-13.)

Example:

In the communication command macro, gets the measurement X coordinate and measurement Y coordinate of the search processing unit of Processing Unit number 2 after remeasurement is executed using the most recent logging image. The measured X and Y coordinates can be gotten with External Reference Data numbers 6 and 7 respectively

```
Rem Execute remeasurement and wait until measurement ends.
Remeasure 0, 1

Rem Get the measurement result.
GetUnitData 2, 6, POSX#
GetUnitData 2, 7, POSY#
```

Usable Modules:

Communication Command Macro

Supported Versions:

Version 3.50 or later

Related Items:

Measure on page 8-193
MeasureStart on page 8-200
MeasureStop on page 8-201

RenumUnitNo

Gets the processing unit number after flow edit.

Format:
RenumUnitNo(<oldUnitNo>)

Parameter:

Parameter name	Data type	Description
<oldUnitNo>	Integer	Processing unit number before edit (0 to (Processing unit number of current scene minus one))

Return value:
Returns the processing unit number after update of measurement flow as an integer.

Description:
Gets the processing unit number specified in the <oldUnitNo> parameter after editing the measurement flow.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
Type mismatch error is occurred when wrong data is specified as parameter. Illegal function call error is not occurred even if non-exist number, values, combination of data or values.
If a value outside the range -2147483648 to 2147483647 is specified as an integer parameter, an "Overflow" error will occur.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.
Besides using this macro function, using reference variables follows the changes of the processing unit numbers resulting from editing the measurement flow. (For details, refer to *4-1-3 Variable* on page 4-6.)

Usage Cautions:
- This macro function can only be used in the *RENUMPROC subroutine. If used in another subroutine, an "Illegal function call" will occur.

Example:
Uses the *RENUMPROC subroutine in the Unit Macro processing unit to get the processing unit number (whose original processing unit number was 5) after editing the measurement flow.

```
*RENUMPROC

    LATESTUNITNO& = 5

    Rem Get the processing unit number after flow edit.
    LATESTUNITNO& = RenumUnitNo(LATESTUNITNO&)

Return
```

Usable Modules:
Unit Calculation Macro / Unit Macro

Supported Versions:

Version 3.50 or later

Related Items:

DeleteUnit on page 8-72
InsertUnit on page 8-168
UnitNo on page 8-349
Ut on page 8-351

RGB

Gets the color value.

Format:
RGB(<red>, <green>, <blue>)

Parameter:

Parameter name	Data type	Description
<red>	Integer	Red component of the color value being gotten (0 to 255)
<green>	Integer	Green component of the color value being gotten (0 to 255)
<blue>	Integer	Blue component of the color value being gotten (0 to 255)

Return value:
Returns the color value as an integer value.
The red component is stored in the lower byte of the color value, green component is stored is stored in the middle byte of the color value, and blue component is stored in the upper byte of the color value.

Description:
Gets the color value of the color that has the red component specified in the <red> parameter, the green component specified in the <green> parameter, and the blue component specified in the <blue> parameter. A drawing color can be specified in specifying a color value in a macro function that sets a drawing style and in some drawing macro functions.
If a value that exceeds the allowed setting range is specified in the <red> parameter, <green> parameter, or <blue> parameter, the value is handled as being 255.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
Type mismatch error is occurred when wrong data is specified as parameter. Illegal function call error is not occurred even if non-exist number, values, combination of data or values.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Displays a green character string in the text window.

```
Rem Get the color value of green.
COLOR& = RGB(0, 255, 0)

Rem Draw the character string in the text window.
DrawText "Processing OK", COLOR&, 1
```

Usable Modules:
Unit Calculation Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

DrawFillImage on page 8-96
DrawText on page 8-109
SetDrawStyle on page 8-272
SetTextStyle on page 8-297

Right$

Extracts the specified length of characters from the right side of character string.

Format:
Right$(<string>, <length>)

Parameter:

Parameter name	Data type	Description
<string>	Character string	Extraction target character string
<length>	Integer	Length of characters to be extracted (1 to the length of the target character string)

Return value:
Returns the character string type value of the extracted character string.

Description:
Extracts the specified length in the <length> parameter from the right side of specified character string in the <string> parameter.
Specify the length of characters to be extracted in bytes for the <length> parameter. Each single-byte character (i.e., half-width alphanumeric character and symbol) consumes one byte, whereas each doublebyte character consumes two bytes.
Extracts the specified length in the <length> parameter from the left side of specified character string in the <string> parameter.
If the length specified in the <length> parameter is longer than the length of the character string specified in the <string> parameter, the whole character string in the parameter is extracted.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a value outside the range -2147483648 to 2147483647 is specified as an integer parameter, an "Overflow" error will occur.
Even if a character string longer than 255 characters is specified for a character string parameter, an error will not occur.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:

Extracts 6-byte length of characters from the right side of the character string. Because one half-width alphabet consumes single byte, this example extracts 6 characters from the character string.

```
CHARA$ = "Measurement Result"

Rem Extract 6-byte length of characters from the right side of the character strin
g.
TITLE$ = Right$(CHARA$, 6)
```

The result is shown below.

```
TITLE$ = "Result"
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

Asc on page 8-34
Chr$ on page 8-44
Hex$ on page 8-158
LCase$ on page 8-179
Left$ on page 8-180
Len on page 8-181
Mid$ on page 8-205
Piece$ on page 8-219
Str$ on page 8-314
Str2$ on page 8-315
UCase$ on page 8-339
Val on page 8-352

Rmdir

Deletes a directory.

Format:
Rmdir <directoryName>

Parameter:

Parameter name	Data type	Description
<directoryName>	Character string	Name of a directory to be deleted

Return value:
None.

Description:
Deletes the directory specified in the <directoryName> parameter.
In the <directoryName> parameter, use an absolute path to specify the directory name of the directory to be deleted.
In the following case, an "Illegal function call" error occurs without deleting a directory.

- The specified directory does not exist
- If the external storage is specified for where to delete a directory from with no external storage inserted
- If more than one file is in the specified directory

If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.

An error of "String too long" will be occurred when you specify the character strings as character strings type exceeds 255 characters.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Deletes a directory named "IMAGE2" under the root folder of the E drive.

```
Rmdir "E:\IMAGE2"
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

Dskf on page 8-114
GetSystemData on page 8-148
Kill on page 8-176
Fcopy on page 8-129
Isfile on page 8-170
Mkdir on page 8-207

RunOut

Sets the output state of the RUN signal.

Format:
RunOut <ioIdent>, <state>

Parameter:

Parameter name	Data type	Description
<ioIdent>	Character string	Identification name of the communication module to be used ("ParallelIo" or "EtherCAT") (Refer to *9-4 List of I/O Modules* on page 9-27.)
<state>	Integer	Output state of terminal 0: Output OFF 1: Output ON

Return value:
None.

Description:
Sets the RUN signal of the communication module specified in the <ioIdent> parameter to the output state specified in the <state> parameter
Normally "ParallelIo" or "EtherCAT" should be specified in the <ioIdent> parameter.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If an identification name that does not exist is specified in the <ioIdent> parameter, an "Illegal function call" error will occur.

If specifying a blank character string ("") as <ioIdent> parameter or omitting <ioIdent> parameter, the setting is applied to all communication modules.
Even if an output status parameter value that does not exist (i.e., other than 0 and 1) is specified in the <state> parameter, an error will not occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
In the communication command macro, sets the BUSY signal of parallel I/O to ON.

```
IOMODULE$ = "ParallelIo"

Rem Set the output state.
RunOut IOMODULE$, 1
```

Usable Modules:
Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 5.40 or later

Related Items:

BusyOut on page 8-37
GetAll on page 8-134
GetPort on page 8-144
JudgeOut on page 8-175
PutAll on page 8-223
PutPort on page 8-225

SaveBackupData

Saves the system + scene group 0 data.

Format:
SaveBackupData(<fileName>)

Parameter:

Parameter name	Data type	Description
<fileName>	Character string	File name of bkd file to save (system + scene group 0 data (*.bkd))

Return value:
None.

Description:
Saves the system + scene group 0 in the file with the file name specified in the <fileName> parameter.
In the <fileName> parameter, use an absolute path to specify the file name of the file to be saved.
Specify the file extension ".bkd" in the file name specified in the <fileName> parameter.
If the file specified in the <fileName> parameter already exists, it is overwritten.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
An error of "String too long" will be occurred when you specify the character strings as character strings type exceeds 255 characters.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
- Execute this macro function when the BUSY signal or other measurement in progress signal is ON and measurement is prohibited. (For details, refer to *5-1-4 State Transitions and Execution Timing* on page 5-13.)
- When you use FH series, do not save to the any folder except RAMDisk and external storage (such as C:\ProgramFiles\FZ). It is possible not to perform correctly due to the decrease of Scene data storage region.

Save Destination	FH series
RAMDisk	C:\Data\RAMDisk
External Storage	E:\, F:\, G:\, H:\, M:\, S:\, T:\, U:\, V:\, W:\, X:\, Y:\, Z:\

Example:
Saves the system + scene group 0 data to a file.

```
Rem Save the system + scene group 0 data to a file.
SaveBackupData "E:\BACKDIR\BackupData.bkd"
```

Usable Modules:
Communication Command Macro

Supported Versions:
Version 3.50 or later

Related Items:

GetSystemData on page 8-148
SaveData on page 8-248
SaveSceneGroup on page 8-253
SaveUnitData on page 8-255
LoadBackupData on page 8-185
SaveScene on page 8-252
SaveSystemData on page 8-254

SaveData

Saves the data to the controller.

Format:
SaveData

Parameter:
None.

Return value:
None.

Description:
Saves the current system group data and the system data to the sensor controller.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
- Execute this macro function when the BUSY signal or other measurement in progress signal is ON and measurement is prohibited. (For details, refer to *5-1-4 State Transitions and Execution Timing* on page 5-13.)

Example:
Saves the current system group data and the system data to the sensor controller.

```
Rem Save the data to the controller.
SaveData
```

Usable Modules:
Communication Command Macro

Supported Versions:
Version 3.50 or later

Related Items:

SaveBackupData on page 8-247
SaveSceneGroup on page 8-253
SaveUnitData on page 8-255
SaveScene on page 8-252
SaveSystemData on page 8-254

SaveImage

Saves image data.

Format:
SaveImage <preImageNo>, <fileName>

Parameter:

Parameter name	Data type	Description
<preImageNo>	Integer	Number (-1 to (number of images already logged in main unit minus one)) of main unit logging image to be saved
<fileName>	Character string	File name of file to be saved

Return value:
None.

Description:
Saves the image that has the image logging number specified in the <preImageNo> parameter, using the file name specified in the <fileName> parameter, in ifz format.
In the <preImageNo> parameter, specify the number of the logging image already logged as a logging image in the main unit. (For details, refer to *Setting Logging Conditions [Logging Setting]* in the *Vision System FH/FHV Series User's Manual (Cat. No. Z365))*.)
If -1 is specified for the <preImageNo> parameter, the most recent input image is saved.
In the <fileName> parameter, use an absolute path to specify the file name of the file to be saved.
Specify the file extension ".ifz" in the file name specified in the <fileName> parameter.
If the file specified in the <fileName> parameter already exists, it is overwritten.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If a value outside the range -2147483648 to 2147483647 is specified as an integer parameter, an "Overflow" error will occur.
An error of "String too long" will be occurred when you specify the character strings as character strings type exceeds 255 characters.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
- Regardless of logging settings, this command saves the image file in the IFZ format.
- When you use FH series, do not save to the any folder except RAMDisk and external storage (such as C:\ProgramFiles\FZ). It is possible not to perform correctly due to the decrease of Scene data storage region.

Save Destination	FH series
RAMDisk	C:\Data\RAMDisk
External Storage	E:\, F:\, G:\, H:\, M:\, S:\, T:\, U:\, V:\, W:\, X:\, Y:\, Z:\

Example:
Saves the most recent measurement image.

```
Rem Save the most recent input image as a file.
SaveImage -1, "E:\IMAGE\sample.ifz"
```

Usable Modules:
Scene Control Macro / Communication Command Macro

Supported Versions:
Version 3.50 or later

Related Items:

GetSystemData on page 8-148
Remeasure on page 8-239

SaveMeasureImage

Saves the measurement image of the processing unit.

Format:
SaveMeasureImage <measureImageNo>, <fileName>, <imageFormat>[, <startX>, <startY>, <sizeX>, <sizeY>]

Parameter:

Parameter name	Data type	Description
<measureImageNo>	Integer	Measurement image number of the measurement image to be saved (always 0)
<fileName>	Character string	File name of file to be saved
<imageFormat>	Integer	Image format of image to be saved 0: BMP format 10000 to 10100: JPEG format (10000 + JPEG image quality (0 to 100)) 20000 to 20009: PNG format (20000 + PNG image compression level (0 to 9))
<startX>	Integer	Start point X of image region to be saved
<startY>	Integer	Start point Y of image region to be saved
<sizeX>	Integer	X dimension of image to be saved (at least 1)
<sizeY>	Integer	Y dimension of image to be saved (at least 1)

Return value:
None.

Description:
Saves the measurement image specified in the <measureImageNo> parameter in a file with the file name specified in the <fileName> parameter, the image format specified in the <imageFormat> parameter, and the pixel size specified in the <sizeX> and <sizeY> parameters cut off from the position in camera coordinates that starts from the upper left point specified in the <startX> parameter and <startY> parameter.
If the <start X>, <start Y>, <size X>, and <size Y> parameters are omitted, the entire image is saved.
Always specify 0 in the <measureImageNo> parameter.
In the <fileName> parameter, use an absolute path to specify the file name of the file to be saved.
In the file name specified in the <fileName> parameter, specify the file extension (such as ".bmp").
If the file specified in the <fileName> parameter already exists, it is overwritten.

Specify a value of at least 1 in the <sizeX> and <sizeY> parameters.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If a value outside the range -2147483648 to 2147483647 is specified as an integer parameter, an "Overflow" error will occur.
An error of "String too long" will be occurred when you specify the character strings as character strings type exceeds 255 characters.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:

- This macro function can only be used in the *MEASUREPROC subroutine. If used in another subroutine, an "Illegal function call" error will occur.
- When you use FH series, do not save to the any folder except RAMDisk and external storage (such as C:\ProgramFiles\FZ). It is possible not to perform correctly due to the decrease of Scene data storage region.

Save Destination	FH series
RAMDisk	C:\Data\RAMDisk
External Storage	E:\, F:\, G:\, H:\, M:\, S:\, T:\, U:\, V:\, W:\, X:\, Y:\, Z:\

Example:

Saves the entire image of measurement image 0 in a file in BMP format.

```
SaveMeasureImage 0, "E:\IMAGE¥sample.bmp", 0
```

Usable Modules:

Unit Macro

Supported Versions:

Version 4.00 or later

Related Items:

GetImageSize on page 8-136
GetSystemData on page 8-148
GetUnitData on page 8-153

SaveScene

Saves the scene data.

Format:
SaveScene <sceneNo>, <fileName>

Parameter:

Parameter name	Data type	Description
<sceneNo>	Integer	Scene number to save the scene (0 to 127)
<fileName>	Character string	File name of the scene data to save (*.scn)

Return value:
None.

Description:
Saves the scene data of the scene number specified in the <sceneNo> parameter in the file with the file name specified in the <fileName> parameter.
In the <fileName> parameter, use an absolute path to specify the file name of the file to be saved.
Specify the file extension ".scn" in the file name specified in the <fileName> parameter.
If the file specified in the <fileName> parameter already exists, it is overwritten.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
An error of "String too long" will be occurred when you specify the character strings as character strings type exceeds 255 characters.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
- Execute this macro function when the BUSY signal or other measurement in progress signal is ON and measurement is prohibited. (For details, refer to *5-1-4 State Transitions and Execution Timing* on page 5-13.)
- When you use FH series, do not save to the any folder except RAMDisk and external storage (such as C:\ProgramFiles\FZ). It is possible not to perform correctly due to the decrease of Scene data storage region.

Save Destination	FH series
RAMDisk	C:\Data\RAMDisk
External Storage	E:\, F:\, G:\, H:\, M:\, S:\, T:\, U:\, V:\, W:\, X:\, Y:\, Z:\

Example:
Saves the scene data of scene 2 in a file.

```
Rem Save the scene data of scene 2 in a file.
SaveScene 2, "E:\BACKDIR\scene02.scn"
```

Usable Modules:
Communication Command Macro

Supported Versions:

Version 3.50 or later

Related Items:

GetSystemData on page 8-148
LoadBackupData on page 8-185
SaveSceneGroup on page 8-253
SaveUnitData on page 8-255
LoadScene on page 8-186
SaveData on page 8-248
SaveSystemData on page 8-254
SceneNo on page 8-263

SaveSceneGroup

Saves the scene group data.

Format:
SaveSceneGroup <sceneGroupNo>, <fileName>

Parameter:

Parameter name	Data type	Description
<sceneGroupNo>	Integer	Scene group number of the scene group to save (0 to 31)
<fileName>	Character string	File name of the scene group data to save (*.sgp)

Return value:
None.

Description:
Saves the scene group data of the scene group number specified in the <sceneNo> parameter in the file with the file name specified in the <fileName> parameter.
In the <fileName> parameter, use an absolute path to specify the file name of the file to be saved.
Specify the file extension ".sgp" in the file name specified in the <fileName> parameter.
If the file specified in the <fileName> parameter already exists, it is overwritten.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
An error of "String too long" will be occurred when you specify the character strings as character strings type exceeds 255 characters.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
- Execute this macro function when the BUSY signal or other measurement in progress signal is ON and measurement is prohibited. (For details, refer to *5-1-4 State Transitions and Execution Timing* on page 5-13.)
- When you use FH series, do not save to the any folder except RAMDisk and external storage (such as C:\ProgramFiles\FZ). It is possible not to perform correctly due to the decrease of Scene data storage region.

Save Destination	FH series
RAMDisk	C:\Data\RAMDisk
External Storage	E:\, F:\, G:\, H:\, M:\, S:\, T:\, U:\, V:\, W:\, X:\, Y:\, Z:\

Example:

Saves the scene group data of scene group 2 in a file.

```
Rem Save the scene group data of scene group 2 in a file.
SaveSceneGroup 2, "E:\BACKDIR\scenegroup02.sgp"
```

Usable Modules:
Communication Command Macro

Supported Versions:
Version 3.50 or later

Related Items:

GetSystemData on page 8-148
LoadBackupData on page 8-185
SaveScene on page 8-252
SaveUnitData on page 8-255
LoadSceneGroup on page 8-187
SaveData on page 8-248
SaveSystemData on page 8-254
SceneGroupNo on page 8-260

SaveSystemData

Saves the system data.

Format:
SaveSystemData <fileName>

Parameter:

Parameter name	Data type	Description
<fileName>	Character string	File name of the system data to save (*.ini)

Return value:
None.

Description:
Saves the system data in the file with the file name specified in the <fileName> parameter.
In the <fileName> parameter, use an absolute path to specify the file name of the file to be saved.
Specify the file extension ".ini" in the file name specified in the <fileName> parameter.
If the file specified in the <fileName> parameter already exists, it is overwritten.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
An error of "String too long" will be occurred when you specify the character strings as character strings type exceeds 255 characters.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
- Execute this macro function when the BUSY signal or other measurement in progress signal is ON and measurement is prohibited. (For details, refer to *5-1-4 State Transitions and Execution Timing* on page 5-13.)
- When you use FH series, do not save to the any folder except RAMDisk and external storage (such as C:\ProgramFiles\FZ). It is possible not to perform correctly due to the decrease of Scene data storage region.

Save Destination	FH series
RAMDisk	C:\Data\RAMDisk
External Storage	E:\, F:\, G:\, H:\, M:\, S:\, T:\, U:\, V:\, W:\, X:\, Y:\, Z:\

Example:

Saves the system data in a file.

```
Rem Save the system data in a file.
SaveSystemData "E:\BACKDIR\backupsysset.ini"
```

Usable Modules:

Communication Command Macro

Supported Versions:

Version 3.50 or later

Related Items:

GetSystemData on page 8-148
LoadBackupData on page 8-185
SaveScene on page 8-252
SaveUnitData on page 8-255
LoadSystemData on page 8-188
SaveData on page 8-248
SaveSceneGroup on page 8-253

SaveUnitData

Saves a processing unit.

Format:

SaveUnitData <sceneNo>, <unitNo>, <unitCount>, <fileName>

Parameter:

Parameter name	Data type	Description
<sceneNo>	Integer	Scene number to save the scene (-1 to 127)
<unitNo>	Integer	Processing unit number to begin to save (0 to (the number of registered processing units in the current scene minus one))
<unitCount>	Integer	Number of pieces of the processing unit to save (-1, 1 to (the number of registered processing units in the current scene) - (the processing unit number to begin to save))
<fileName>	Character string	File name of the processing unit to save (*unt)

Return value:

None.

Description:

Saves processing unit data in the file with the file name specified in the <fileName> parameter from processing units whose processing unit numbers are specified in the <unitNo> parameter, with the number of processing units specified in the <unitCount> parameter, of the scene number specified in the <sceneNo> parameter.

When -1 is specified in the <sceneNo> parameter, the scene number of the current scene is specified in the scene number of the scene to be saved.

If -1 is specified in the <unitCount> parameter, all processing unit data included in the processing unit data file is saved.

In the <fileName> parameter, use an absolute path to specify the file name of the file to be saved.
Specify the file extension ".unt" in the file name specified in the <fileName> parameter.
If the file specified in the <fileName> parameter already exists, it is overwritten.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
An error of "String too long" will be occurred when you specify the character strings as character strings type exceeds 255 characters.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:

- Execute this macro function when the BUSY signal or other measurement in progress signal is ON and measurement is prohibited. (For details, refer to *5-1-4 State Transitions and Execution Timing* on page 5-13.)
- When you use FH series, do not save to the any folder except RAMDisk and external storage (such as C:\ProgramFiles\FZ). It is possible not to perform correctly due to the decrease of Scene data storage region.

Save Destination	FH series
RAMDisk	C:\Data\RAMDisk
External Storage	E:\, F:\, G:\, H:\, M:\, S:\, T:\, U:\, V:\, W:\, X:\, Y:\, Z:\

Example:

Saves the processing units of Processing Unit number 2 to Processing Unit number 4 of the current scene in a processing unit data file.

```
Rem Save Processing Unit number 2 to Processing Unit number 4 of the current scene
 in a processing unit data file.
SaveUnitData -1, 2, 3, "E:\BACKDIR\unitsave.unt"
```

Usable Modules:

Communication Command Macro

Supported Versions:

Version 3.50 or later

Related Items:

GetSystemData on page 8-148
SaveBackupData on page 8-247
SaveScene on page 8-252
SaveSystemData on page 8-254
UnitNo on page 8-349
LoadUnitData on page 8-189
SaveData on page 8-248
SaveSceneGroup on page 8-253
SceneNo on page 8-263
Ut on page 8-351

SceneCount

Gets the number of scenes that can be used.

Format:
SceneCount

Parameter:
None.

Return value:
Returns the number of scenes that can be used as an integer.

Description:
Gets the number of scenes that can be used.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Gets the number of scenes that can be used.

```
NUM& = SceneCount
```

Usable Modules:
Scene Control Macro / Communication Command Macro

Supported Versions:
Version 3.50 or later

Related Items:

ChangeScene on page 8-41
CopyScene on page 8-56
ClearScene on page 8-47
SceneNo on page 8-263

SceneDescription$

Gets the scene description.

Format:
SceneDescription$(<sceneNo>)

Parameter:

Parameter name	Data type	Description
<sceneNo>	Integer	Scene number (0 to 127) of scene whose description is to be gotten.

Return value:
Returns the scene description as a character string.

Description:
Gets the description set in the scene of the scene number specified in the <sceneNo> parameter.
If a description is not set, the null character string ("") is returned.
The scene description can be set in the maintenance screen or by executing the SetSceneDescription function. (Refer to *SetSceneDescription* on page 8-288.), (For details, refer to *Editing Scenes* in the *Vision System FH/ FHV Series User's Manual (Cat. No. Z365)*.)
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Gets the description of scene 1, and if a description is not set, setting the description.

```
Rem Get the scene description.
DESCRIPTION$ = SceneDescription$(1)

If DESCRIPTION$ = "" Then
   Rem Set the scene description.
   SetSceneDescription 1, "Description 1"
Endif
```

Usable Modules:
Scene Control Macro / Communication Command Macro

Supported Versions:
Version 3.50 or later

Related Items:

SceneMaker$ on page 8-262
SceneTitle$ on page 8-264
SetSceneDescription on page 8-288
SetSceneMaker on page 8-291

SetSceneTitle on page 8-292

SceneGroupCount

Gets the number of usable scene groups.

Format:
SceneGroupCount

Parameter:
None.

Return value:
The number of usable scene groups is returned as an integer value.

Description:
Gets the number of usable scene groups.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Gets the number of usable scene groups.

```
NUM& = SceneGroupCount
```

Usable Modules:
Scene Control Macro / Communication Command Macro

Supported Versions:
Version 3.50 or later

Related Items:

ChangeSceneGroup on page 8-42
CopySceneGroup on page 8-57
ClearSceneGroup on page 8-48
SceneGroupNo on page 8-260

SceneGroupNo

Gets the scene group number of the current scene group.

Format:
SceneGroupNo

Parameter:
None.

Return value:
The scene group number of the current scene group is returned as an integer value.

Description:
Gets the scene group number of the current scene group.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Gets the scene group number of the current scene group, and if the scene group number is 2, change to scene 3.

```
Rem Get the scene group number of the current scene group.
NO& = SceneGroupNo

Rem Get the scene group number, and if 2, change to scene 3.
If NO& = 2 Then
   ChangeScene 3
Endif
```

Usable Modules:
Scene Control Macro / Communication Command Macro

Supported Versions:
Version 3.50 or later

Related Items:

ChangeScene on page 8-41
ChangeSceneGroup on page 8-42
ClearSceneGroup on page 8-48
CopySceneGroup on page 8-57
SceneGroupCount on page 8-259
SceneNo on page 8-263

SceneGroupTitle$

Gets the title of the scene group.

Format:
SceneGroupTitle$(<sceneGroupNo>)

Parameter:

Parameter name	Data type	Description
<sceneGroupNo>	Integer	Scene group number (0 to 31) of the scene group whose scene group title is gotten.

Return value:
Returns the title value of the character string scene group.

Description:
Gets the title set in the scene group that has the scene group number specified in the <sceneGroupNo> parameter.
If the title is not set, returns the default character string such as "scene group 0".
The scene group title can be set by executing the SetSceneGroupTitle function, or in the scene maintenance screen. (Refer to *SetSceneGroupTitle* on page 8-290.), (For details, refer to *Editing Scenes* in the *Vision System FH/ FHV Series User's Manual (Cat. No. Z365)*.)
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Gets the title of scene group 2.

```
TITLE$ = SceneGroupTitle$(2)
```

Usable Modules:
Scene Control Macro / Communication Command Macro

Supported Versions:
Version 3.50 or later

Related Items:

SceneTitle$ on page 8-264
SetSceneTitle on page 8-292

SceneMaker$

Gets the scene creator.

Format:
SceneMaker$(<sceneNo>)

Parameter:

Parameter name	Data type	Description
<sceneNo>	Integer	Scene number (0 to 127) of scene whose creator is to be gotten.

Return value:
Returns the value of the scene creator as a character string.

Description:
Gets the name of the creator set in the scene that has the screen number specified in the <sceneNo> parameter.
If a creator name is not set, returns the null character string ("").
The scene creator can be set in the scene maintenance screen, or by executing the SetSceneMaker function. (Refer to *SetSceneMaker* on page 8-291.), (For details, refer to *Editing Scenes* in the *Vision System FH/ FHV Series User's Manual (Cat. No. Z365)*.)
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Gets the creator of scene 3, and if not set, setting the creator.

```
Rem Get the creator of the scene.
NAME$ = SceneMaker$(3)

If NAME$ = "" Then
   Rem Set the creator of the scene.
   SetSceneMaker 3, "Maker"
Endif
```

Usable Modules:
Scene Control Macro / Communication Command Macro

Supported Versions:
Version 3.50 or later

Related Items:

SceneDescription$ on page 8-258
SceneTitle$ on page 8-264
SetSceneDescription on page 8-288
SetSceneMaker on page 8-291

SetSceneTitle on page 8-292

SceneNo

Gets the scene number of the current scene.

Format:
SceneNo

Parameter:
None.

Return value:
Returns the scene number of the current scene as an integer value.

Description:
Gets the scene number of the current scene.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Gets the scene number of the current scene, and if not 2, changing to scene 2.

```
Rem Gets the scene number of the current scene.
NO& = SceneNo

Rem If the scene number is not 2, change to scene 2.
If NO& <> 2 Then
   ChangeScene 2
Endif
```

Usable Modules:
Communication Command Macro

Supported Versions:
Version 3.50 or later

Related Items:

ChangeScene on page 8-41
CopyScene on page 8-56
ClearScene on page 8-47
SceneCount on page 8-257

SceneTitle$

Gets the scene title.

Format:
SceneTitle$(<sceneNo>)

Parameter:

Parameter name	Data type	Description
<sceneNo>	Integer	Scene number (0 to 127) of scene whose title is to be gotten

Return value:
Returns the scene title as a character string.

Description:
Gets the title set in the scene that has the scene number specified in the <sceneNo> parameter.
If a title is not set, returns the default character string, such as "Scene0".
The scene title can be set in the scene maintenance screen or flow edit screen, or by executing the SetSceneTitle function. (Refer to *SetSceneTitle* on page 8-292.), (For details, refer to *Editing Scenes* and *Editing Processing Units in Scenes* in the *Vision System FH/FHV Series User's Manual (Cat. No. Z365)*.)
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Gets the title of scene 2.

```
TITLE$ = SceneTitle$(2)
```

Usable Modules:
Scene Control Macro / Communication Command Macro

Supported Versions:
Version 3.50 or later

Related Items:

SceneDescription$ on page 8-258
SetSceneDescription on page 8-288
SetSceneTitle on page 8-292
SceneMaker$ on page 8-262
SetSceneMaker on page 8-291

ScreenCapture

Saves the capture of the screen.

Format:
ScreenCapture <fileName>

Parameter:

Parameter name	Data type	Description
<fileName>	Character string	File name that saves the capture of the screen.

Return value:
None.

Description:
Takes a screen capture of the sensor controller screen and saves it in BMP format with the file name specified in the <fileName> parameter.
In the <fileName> parameter, use an absolute path to specify the file name of the file to be saved.
In the <fileName> parameter, specify the file name with the file extension ".bmp" to save as BMP. It is not possible to capture the screen correctly if an extension other than ".bmp" is specified.
If the file specified in the <fileName> parameter already exists, it is overwritten.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
An error of "String too long" will be occurred when you specify the character strings as character strings type exceeds 255 characters.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
- When you use FH series, do not save to the any folder except RAMDisk and external storage (such as C:\ProgramFiles\FZ). It is possible not to perform correctly due to the decrease of Scene data storage region.

Save Destination	FH series
RAMDisk	C:\Data\RAMDisk
External Storage	E:\, F:\, G:\, H:\, M:\, S:\, T:\, U:\, V:\, W:\, X:\, Y:\, Z:\

Example:
Captures the screen and save the screen capture to a file with a file name "E:\IMAGE\samplecapture.bmp".

```
ScreenCapture "E:\IMAGE\samplecapture.bmp"
```

Usable Modules:
Scene Control Macro / Communication Command Macro

Supported Versions:
Version 3.50 or later

Related Items:

GetSystemData on page 8-148
Str$ on page 8-314
Str2$ on page 8-315

Select Case Case Else End Select

Controls the process flow according to the specified condition.

Format:
Select <expression>
[Case <value>
<caseStatement>]
:
:
[Case Else
<elseStatement>]
End Select

Parameter:

Parameter name	Data type	Description
<expression>	Integer	Expression that controls the process flow.
<value>	Integer	Numeric value that is compared with the expression value
<caseStatement>	-	Statement that is executed when a result value of the expression and the numeric value match
<elseStatement>	-	Statement that is executed when any of numeric values did not match a result value of the expression.

Return value:
None.

Description:
Among the multiple Case block statement in the statement, executes the statements whose specified value in the <value> parameter match the value of the specified expression in the <expression> parameter.
If any of values specified in the <value> parameters did not match the result value of the specified expression in the <expression> parameter, the Case Else block statement specified in the <elseStatement> parameter is executed.
If there are multiple of <value> parameters having a value that matches a result value of the expression, only the first statement from the beginning of the Case block statement having a value that matches a result value of the expression is executed.
Case block statements and Case Else block statement are optional.
If the program process is jumped into or out of the Case and Case Else block statements using the Goto function in a statement, unexpected operation may occur.
If neither the Select statement nor the End Select statement is used, either the "CASE without SELECT", "END SELECT without SELECT", "SELECT without END SELECT", or "CASE without END SELECT" error will occur depending on the statement that is used.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:

Uses the *MEASUREDISPG subroutine in the unit macro processing unit to change the display in the image window according to the set image display sub-number in the image window of the main screen.

```
*MEASUREDISPG

   Rem Get the displayed sub-image number.
   SUBNO& = DisplaySubNo

   Rem Change the display on the image window according to the sub-image number of
 the sub-image to be displayed.
   Select SUBNO&
   Case 1
      Rem If the gotten sub-image number is 1, the title of processing unit 1 is d
isplayed with the color in accordance with the judgment result.
      SetTextStyle 24, TA_LEFT, UnitJudge(1), 0, FONTSTYLE_NORMAL
      TEXT$ = UnitTitle$(1)
   Case 2
      Rem If the gotten sub-image number is 2, the title of processing unit 2 is d
isplayed with the color in accordance with the judgment result.
      SetTextStyle 24, TA_LEFT, UnitJudge(2), 0, FONTSTYLE_NORMAL
      TEXT$ = UnitTitle$(2)
   Case Else
      Rem If the gotten sub-image number is other than 1 and 2, "Error" is display
ed in the "unmeasured" color.
      SetTextStyle 24, TA_LEFT, JUDGE_NC, 0, FONTSTYLE_NORMAL
      TEXT$ = "Error"
   End Select

   Rem Displays text on the image window.
   DrawTextG TEXT$, 50, 0, 0, UnitNo

Return
```

Usable Modules:

Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:

Version 3.50 or later

Related Items:

DisplaySubNo on page 8-75
Gosub on page 8-156
If Then Elseif Else EndIf on page 8-160
UnitJudge on page 8-348
DrawTextG on page 8-111
If Then Else on page 8-159
SetTextStyle on page 8-297
UnitNo on page 8-349

SendData

Sends data.

Format:

SendData <ioIdent>, <outputData()>, <outputSize>[, <parameter()>, <parameterSize>]

Parameter:

Parameter name	Data type	Description
<ioIdent>	Character string	Identification name of the communication module to be used (Refer to *9-4 List of I/O Modules* on page 9-27.)
<outputData()>	Integer array	Data to send
<outputSize>	Integer	Data size to send
<parameter()>	Integer array	The parameter data specified in the option
<parameterSize>	Integer	Size of parameter data specified optionally

Return value:

None.

Description:

Sends the amount, specified in the <outputSize> parameter, of the data specified in the <outputData()> parameter by using the communication module specified in the <ioIdent> parameter.
In the <outputData()> parameter, specify the 1D integer array variable that stores the data to be sent, without adding element numbers but adding () to the variables. No delimiter is required in Non-procedure UDP communications and Non-procedure TCP communications .
Values that can be set in the <parameter()> parameter and <parameterSize> parameter depend on the communication module specified in the <ioIdent> parameter. For details, refer to *9-4 List of I/O Modules* on page 9-27. The <parameter()> parameter and <parameterSize> parameter can be omitted.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If Handshake is on and data sending fails due to a communication timeout or other reason, an "Illegal function call" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:

None.

Example:

In normal UDP communication, specifies the destination and sends data.

```
Rem Create the destination address information (10.1.1.101)
Dim IPADDR&(4)
IPADDR&(0) = 10
IPADDR&(1) = 1
IPADDR&(2) = 1
IPADDR&(3) = 101

Rem Make the transmit data.
Dim BUFFER&(4)
BUFFER&(0) = 1
BUFFER&(1) = 2
BUFFER&(2) = 3
BUFFER&(3) = 4
BUFFER&(4) = 5

Rem Transmit the data selected address.
SendData "UdpNormal", BUFFER&(), 4 * 5, IPADDR&(), 4 * 4
```

In PLC link, specifying the offset value and writing data to the data output area.

```
Rem Create the offset data.
Dim OFFSET&(0)
OFFSET&(0) = 2

Rem Make the transmit data.
Dim BUFFER&(4)
BUFFER&(0) = 1
BUFFER&(1) = 2
BUFFER&(2) = 3
BUFFER&(3) = 4
BUFFER&(4) = 5

Rem Use the offset value to send the data.
SendData "SerialPlcLink", BUFFER&(), 4 * 5, OFFSET&(), 4 * 1
```

Usable Modules:

Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:

Version 3.50 or later

Related Items:

ReceiveData on page 8-231

SendString on page 8-270

SendString

Sends the character string data.

Format:
SendString <ioIdent>, <outputString>

Parameter:

Parameter name	Data type	Description
<ioIdent>	Character string	Identification name of the communication module to be used (Refer to *9-4 List of I/O Modules* on page 9-27.)
<outputString>	Character string	Character string to send

Return value:
None.

Description:
Sends the character string specified in the <outputString()> parameter by using the communication module specified in the <ioIdent> parameter. No delimiter is required in Non-procedure UDP communications. A CR delimiter is required in Non-procedure TCP communications. The delimiter cannot be changed.
Some communication modules do not support this macro function. (Refer to *9-4 List of I/O Modules* on page 9-27.)
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
Type mismatch error is occurred when wrong data is specified as parameter. Illegal function call error is not occurred even if non-exist number, values, combination of data or values.
An error of "String too long" will be occurred when you specify the character strings as character strings type exceeds 255 characters.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
- If the Scene Control Macro or the Unit Macro using UDP non-procedure communication, is used and never received any command from external devices such as a PLC, an "Illegal function call" error will occur after SendString is executed.
- On the FH Series, English characters and characters for the language selected in *Language setting* are allowed.

Example:
Sends the character string in TCP normal communication.

```
Rem Make the transmit characters.
DATA$ = "Test string"

Rem Send a character string.
SendString "TcpNormal", DATA$
```

Usable Modules:
Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:

Version 3.50 or later

Related Items:

ReceiveData on page 8-231
SendData on page 8-268

SetDisplayUnitNo

Sets the processing unit number in the flow window to the selected state.

Format:
SetDisplayUnitNo <unitNo>

Parameter:

Parameter name	Data type	Description
<unitNo>	Integer	Processing unit number to be selected

Return value:
None.

Description:
Sets the processing unit of the processing unit number specified in the <unitNo> parameter to the selected state in the flow window.
When the processing unit displayed in the image window and the text window is set to *Link to Flow Display*, the information of the processing unit selected in the flow window is displayed.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
Type mismatch error is occurred when wrong data is specified as parameter. Illegal function call error is not occurred even if non-exist number, values, combination of data or values.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Sets the unit specified in the command argument of the communication command macro to the selected state in the flow window.

```
Rem Select the processing unit that has the number specified in the argument of th
e communication command.
SetDisplayUnitNo argumentValue#(0)
```

Usable Modules:
Scene Control Macro / Communication Command Macro

Supported Versions:
Version 3.50 or later

Related Items:

DisplayUnitNo on page 8-76
SetImageWindow on page 8-276
UnitNo on page 8-349
Ut on page 8-351

SetDrawStyle

Set the drawing attributes of the graphic figure.

Format:

SetDrawStyle <style>, <width>, <color>

Parameter:

Parameter name	Data type	Description
<style>	Integer	Type of the drawn line PS_SOLID: Solid line PS_DASH: Dashed line*1 PS_DOT: Dotted line*1 PS_DASHDOT: Long dashed short dashed line*1 PS_DASHDOTDOT: Long dashed double-short dashed*1 PS_NULL: No line PS_INSIDEFRAME: Solid line*2
<width>	Integer	Line width of the drawn graphic line
<color>	Integer	Line color value of the drawn graphic line JUDGE_NC: Unmeasured color (Grey) JUDGE_OK: OK judgement color (Green) JUDGE_NG: NG judgement color (Red) RGB Function: Any color

*1. This selection is valid only when the specified line width is 1.

*2. Enabled for the following figures:
- Circle-with-width
- Ellipse
- Arc-with-width

Return value:

None.

Description:

Sets the specified line type by the <style> parameter, the specified line width by the <width> parameter, and specified line color by the <color> parameter as the drawing attributes. Before executing the image screen window control macro function that draws graphic figure, execute this macro function to draw the graphic figure using the set drawing attribute. Use the SetTextStyle function to set the drawing attribute used for the DrawTextG function. (Refer to *SetTextStyle* on page 8-297.)

If any of "PS_DASH", "PS_DASHDOT", and "PS_DASHDOTDOT" is specified in the <style> parameter, specify 1 in the <width> parameter. If other than 1 is specified, a solid line will be drawn.

If circle, wide circle, ellipse, arc, wide arc is drawn with specification of "PS_INSIDEFRAME" for the <style> parameter, the figure with specified line width by the <width> parameter is drawn and diminished so that the drawn figure is within the figure. Other figure types than ones mentioned above are drawn with a solid line (i.e., the same line type as when "PS_SOLID" is specified for the <style> parameter).

The gotten color value by the RGB function can be set for the <color> parameter. (Refer to *RGB* on page 8-242.)

If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.

If a value outside the range -2147483648 to 2147483647 is specified as an integer parameter, an "Overflow" error will occur.

If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:

- This macro function can only be used in the *MEASUREDISPI or *MEASUREDISPG subroutine. If used in another subroutine, an "Illegal function call" error will occur.

Example:

Uses the *MEASUREDISPG subroutine of the Unit Macro processing unit to draw a straight line whose line type is "dashed line" and whose color is "OK Color".

```
*MEASUREDISPG

   Rem Set the drawing attribute.
   SetDrawStyle PS_DASH, 1, JUDGE_OK

   Rem Draw the image.
   DrawLine 100, 100, 500, 400, 0, UnitNo

Return
```

Usable Modules:

Unit Macro

Supported Versions:

Version 3.50 or later

Related Items:

DrawArc on page 8-80
DrawArcW on page 8-82
DrawBox on page 8-84
DrawCircle on page 8-86
DrawCircleW on page 8-88
DrawCursor on page 8-90
DrawEllipse on page 8-92
DrawFigure on page 8-94
DrawLine on page 8-98
DrawLineW on page 8-100
DrawPoint on page 8-103
DrawPolygon on page 8-105
DrawSearchFigure on page 8-107
RGB on page 8-242
SetTextStyle on page 8-297

SetForegroundLine

This command specifies the object line for operation in Multiline Random trigger mode or Non-stop Adjustment mode.

Format:
SetForegroundLine <lineNo>

Parameter:

Parameter name	Data type	Description
<lineNo>	Integer	Operation object of the line number 0 to 7. Line numbers of the Non-stop Adjustment mode are the below. • 0: Measurement operation side • 1: Non-stop Adjustment side

Return value:
None.

Description:
Specify the operation object of the line number using this parameter. The screen of the specified line number is displayed on the top of other screens.
Execute this command while the operation object of the line is running. If it is not running, an "Illegal function call" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.

Usage Cautions:
None.

Example:

```
Rem Switch the line displayed with non-procedure communication.
NO& = int(Argumentvalue#(0))
SetForegroundLine NO&
```

Usable Modules:
Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 5.40 or later

Related Items:
None.

SetGlobalData

Sets the global data.

Format:
SetGlobalData <dataIdent>, <data>

Parameter:

Parameter name	Data type	Description
<dataIdent>	Character string	Identification name of the global data to set the value
<data>	Integer Real number Character string	Value set in the global data

Return value:
None.

Description:
Sets the value specified in the <data> parameter in the global data with the identification name specified in the <dataIdent> parameter.
If global data with the specified identification name does not exist, global data with the identification name specified in the <dataIdent> parameter is added, and the value specified in the <data> parameter is set in the added data.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a character string longer than 255 characters is specified in the <dataIdent> parameter, a "String too long" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
- Execute this macro function when the BUSY signal or other measurement in progress signal is ON and measurement is prohibited. (For details, refer to *5-1-4 State Transitions and Execution Timing* on page 5-13.)

Example:
Sets 1 as the value in the global data with the identification name "ABC".

```
Rem Set 1 in the value of the global data "ABC".
SetGlobalData "ABC", 1

Rem Get the value (integer value) set in the global data "ABC", and store in the v
ariable DATA&.
GetGlobalData "ABC", DATA&
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

AddGlobalData on page 8-26
GetGlobalData on page 8-135

SetImageWindow

Sets the state of the image window.

Format:

[Scene Control Macro / Communication Command Macro]

SetImageWindow <windowNo>, <locationX>, <locationY>, <width>, <height>, <unitNo>, <subNo>, <magnification>, <originX>, <originY>, <update>, <visible>

[Unit Macro]

SetImageWindow <magnification>, <originX>, <originY>

Parameter:

Parameter name	Data type	Description
<windowNo>	Integer	Number of the image window whose state is to be set (0 to 23)
<locationX>	Integer	Upper left X coordinate value of the image window
<locationY>	Integer	Upper left Y coordinate value of the image window
<width>	Integer	Width of the image window
<height>	Integer	Height of the image window
<unitNo>	Integer	Processing unit number of the target processing unit to display (-1 to (the number of registered processing units in the current scene minus one))
<subNo>	Integer	Sub-image number of the target image to display (-1 to 100)
<magnification>	Real number	Display magnification (-1, 1 to 16)
<originX>	Integer	Upper left X coordinate of the image display relative to the upper left coordinate of the image window.
<originY>	Integer	Upper left Y coordinate of the image display relative to the upper left coordinate of the image window
<update>	Integer	Update timing of image window 0: Every measurement (Image mode Freeze) 1: Only when an overall judgement result is NG at the time of measurement (Last NG image). 2: Only when a target processing unit is NG at the time of measurement. 3: Always updated (through display)
<visible>	Integer	Setting of whether to display 0: Window invisible 1: Window visible

Return value:

None.

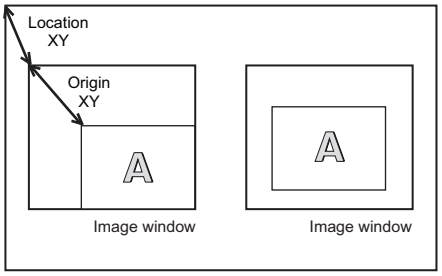
Description:

Sets the state of the image window specified in the <windowNo> parameter. When this macro function is used with the unit macro, the state of the image window displayed using the MEASUREDISPI subroutine is set.

In the <locationX> parameter and <locationY> parameter, specify the relative coordinate values from the upper left coordinates of the gotten image container window to the upper left coordinates of the image window.

In the <width> parameter and <height> parameter, set the values of the image window width and height.

Specify the value of the displayed processing unit number in the <unitNo> parameter. To link the processing unit displayed in the image window to the flow display, specify -1.
Specify the value of the displayed sub image number in the <subNo> parameter. To display the contents of the image window as a position list, specify -1.
Specify the zoom of the image window in the <magnification> parameter. Do not specify 0. To set the zoom to auto, specify -1.
In the <originX> parameter and the <originY> parameter, specify the values of the relative coordinates from the upper left coordinates of the image window to the upper left coordinates of the displayed image.



In the <update> parameter, specify the value of the image mode of the image window.
In the <visible> parameter, specify the value of the display state of the image window.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
Type mismatch error is occurred when wrong data is specified as parameter. Illegal function call error is not occurred even if non-exist number, values, combination of data or values.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:

[Scene Control Macro / Communication Command Macro]

- Execute this macro function when the BUSY signal or other measurement in progress signal is ON and measurement is prohibited. (For details, refer to *5-1-4 State Transitions and Execution Timing* on page 5-13.)

[Unit Macro]

- This macro function can only be used in the *MEASUREDISPI or *MEASUREDISPG subroutine. If used in another subroutine, an "Illegal function call" error will occur.

Example:

In the communication command macro, changes the image mode of image windows 0 to 3 to Through. Setting the BusyOn flag to ON in advance in the communication command macro.

```
For I& = 0 To 3
   Rem Get the state of the image window.
   GetImageWindow I&, LOCATIONX&, LOCATIONY&, WIDTH&, HEIGHT&, UNITNO&, SUBNO&, MA
G#, ORIGINX&, ORIGINY&, UPDATE&, VISIBLE&

   Rem Change the update timing to Through.
   UPDATE& = 3

   Rem Set the state of the image window.
   SetImageWindow I&, LOCATIONX&, LOCATIONY&, WIDTH&, HEIGHT&, UNITNO&, SUBNO&, MA
G#, ORIGINX&, ORIGINY&, UPDATE&, VISIBLE&

Next
```

Usable Modules:

Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:

Version 3.50 or later

Related Items:

DisplayUnitNo on page 8-76
GetTextWindow on page 8-152
UnitNo on page 8-349
GetImageWindow on page 8-137
SetDisplayUnitNo on page 8-271

SetMeasureImage

Sets the measurement image of the processing unit.

Format:

SetMeasureImage <measureImageNo>, <unitNo>, <imageNo>

Parameter:

Parameter name	Data type	Description
<measureImageNo>	Integer	Measurement image number to set to the target image of the measurement (always 0)
<unitNo>	Integer	Processing unit number of the processing unit that holds the image to be set as the measurement image.
<ImageNo>	Integer	Image number of the image that is set to the measurement image

Return value:

None.

Description:

Sets the image of the image number specified in the <imageNo> parameter, which is held by the processing unit specified in the <unitNo> parameter, as the measurement image specified in the <measureImageNo> parameter. The measurement image is an image that can be used for measurement or filtering by a succeeding processing unit in the measurement flow.
Normally 0 should be specified in the <measureImageNo> parameter.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:

- This macro function can only be used in the *MEASUREPROC subroutine. If used in any other subroutines, an error will occur and the function will not be executed.

Example:

Changes the measurement image used by a succeeding processing unit to the camera change image of processing unit 4.

```
Rem Set camera change measurement image 0 of processing unit 4 as the measurement
image.
SetMeasureImage 0, 4, 0
```

Usable Modules:

Unit Macro

Supported Versions:

Version 3.50 or later

Related Items:

CopyMeasureImage on page 8-55
CopyUnitImage on page 8-61
UnitNo on page 8-349
Ut on page 8-351

SetMeasureOut

Sets the external output setting for the measurement result.

Format:
SetMeasureOut <mode>

Parameter:

Parameter name	Data type	Description
<mode>	Integer	External output setting 0: Not output externally 1: Output externally

Return value:
None.

Description:
Sets the *External Output* value in the layout settings to the external output setting value specified in the <mode> parameter. (For details, refer to *Setting the Behavior of Output Signals for Each Layout (Layout Settings)* in the *Vision System FH/FHV Series User's Manual (Cat. No. Z365).)*
Even if 0 is set for the measurement result external output setting value, data is output if the SendData function or SendString function is used in the macro customize functions.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
Type mismatch error is occurred when wrong data is specified as parameter. Illegal function call error is not occurred even if non-exist number, values, combination of data or values.
If a value outside the range -2147483648 to 2147483647 is specified as an integer parameter, an "Overflow" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
- Execute this macro function when the BUSY signal or other measurement in progress signal is ON and measurement is prohibited. (For details, refer to *5-1-4 State Transitions and Execution Timing* on page 5-13.)
- Even when *External Output* is OFF, data output that uses a macro function is executed.

Example:
Sets *External Output* to ON in the communication command macro.

```
SetMeasureOut 1
```

Usable Modules:
Scene Control Macro / Communication Command Macro

Supported Versions:
Version 3.50 or later

Related Items:

GetMeasureOut on page 8-139
SendData on page 8-268
SendString on page 8-270

SetPlcData

Creates the data that is written with the WritePlcMemory function.

Format:

SetPlcData <ioIdent>, <writeData()>, <offset>, <size>, <data>

Parameter:

Parameter name	Data type	Description
<ioIdent>	Character string	Identification name of the communication module to be used (Refer to *9-4 List of I/O Modules* on page 9-27.)
<writeData()>	Integer array	Data to write
<offset>	Integer	Offset to address from which the beginning of the data is to be written (byte units).
<size>	Integer	Data size to set (byte unit)
<data>	Integer Real number Character string	Data to set

Return value:

None.

Description:

Sets the amount, specified in the <size> parameter, of the data specified in the <data> parameter by using the communication module specified in the <ioIdent> parameter. The data is set from the beginning of the data array specified in the <writeData()> parameter, in the position offset by the amount of the value specified in the <offset> parameter.

After creating data with this macro function, execute the WritePlcMemory function to write the data to the PLC memory area.

In the <writeData()> parameter, specify the 1D integer array variable that stores the data to be written, without adding element numbers but adding () to the variables.

In the <offset> parameter and <size> parameter, specify the offset and size in units of bytes. These units are different from the units used in the WritePlcMemory function (units of channels).

Specify 2, 4, or 8 in the <size> parameter. These respectively set a 2-byte integer, 4-byte integer, or 8-byte real number.

If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.

If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.

If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:

- Before using the WritePlcMemory function in PLC link communication to write data to the PLC memory area, always use this macro function to create the data to be written. If the data is directly set in the WritePlcMemory parameter without using this macro function, the correct data may not be set.

Example:

In the communication macro, writes measurement coordinate X and measurement coordinate Y of the search processing unit of Processing Unit number 2 to the PLC connected by PLC link. Measurement coordinate X is external data number 6, and measurement coordinate Y is external data number 7.

```
IOMODULE$ = "UdpPlcLink"

Rem Get the measurement result.
GetUnitData 2, 6, X#
GetUnitData 2, 7, Y#

Rem Convert the real number value multiplied by 1,000 to the integer value.
VALUE0& = Int(X# * 1000)
VALUE1& = Int(Y# * 1000)

Rem Get the settings of the output data area.
GetSystemData IOMODULE$, "outputArea", AREA&
GetSystemData IOMODULE$, "outputMemoryAddress", ADDRESS&

Rem Store the data to be written in an integer array variable.
Dim DATA&(1)
SetPlcData IOMODULE$, DATA&(), 0, 4, VALUE0&
SetPlcData IOMODULE$, DATA&(), 4, 4, VALUE1&

Rem Write the data (4ch) in data output area.
WritePlcMemory IOMODULE$, AREA&, ADDRESS&, 4, DATA&()
```

Usable Modules:

Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:

Version 4.20 or later

Related Items:

GetPlcData on page 8-140
ReadPlcMemory on page 8-229
WritePlcMemory on page 8-363

SetPollingState

Sets the execution status of the communication module.

Format:
SetPollingState <ioIdent>, <state>

Parameter:

Parameter name	Data type	Description
<ioIdent>	Character string	Identification name of communication module whose polling state is to be set (Refer to *9-4 List of I/O Modules* on page 9-27.)
<state>	Integer	Execution status of the communication module to set False: Stopped True: Operating

Return value:
None.

Description:
Sets the polling state specified in the <state> parameter in the communication module specified in the <ioIdent> parameter.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
- After using this macro function to set the polling state of the communication module to the stopped state, always return the polling state to the running state. If the polling state of the communication module is left in the stopped state, the communication module will not be able to receive communication commands.

Example:
Receives normal TCP communication data in the *MEASUREPROC subroutine of the Unit Macro processing unit.

```
Rem Prepare a buffer that can receive 12 bytes of data.
Dim BUFFER&(11)
IOMODULE$ = "TcpNormal"

Rem Set the polling state of the communication module to stopped in order to recei
ve the data.
SetPollingState IOMODULE$, False

Rem Executing the initialization of the reception data size.
SIZE& = 0

Rem Repeat the reception process until the data has been received.
Try
   Do
      Rem Attempting the data reception.
      ReceiveData IOMODULE$, BUFFER&(), 12, SIZE&

      Rem Once the data has been received, display the data size in the system sta
tus console window.
      If(SIZE& > 0) Then
         Print "Received data size = " + Str$(SIZE&)
      Endif
   Loop While SIZE& = 0

   Rem Data has been received, so set the polling state of the communication modul
e to running.
   SetPollingState IOMODULE$, True

Catch
   Rem Return the polling state of the stopped communication module to running.
   If GetPollingState(IOMODULE$) = False Then
      SetPollingState IOMODULE$, True
   Endif

End Try
```

Usable Modules:

Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:

Version 4.20 or later

Related Items:

GetPollingState on page 8-142
ReceiveData on page 8-231
SendData on page 8-268
SendString on page 8-270

SetProfileData

Sets the data of the setting file (.ini file).

Format:
SetProfileData <fileName>, <section>, <key>, <data>

Parameter:

Parameter name	Data type	Description
<fileName>	Character string	Absolute path to the file
<section>	Character string	The section name
<key>	Character string	The Key name
<data>	Character string	Data to be set

Return value:
None.

Description:
Writes the value specified in the <data> parameter in the format of the .ini file for the section name specified in the <section> parameter and the key name specified in the <key> parameter in the file specified in the <fileName> parameter.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Set the folder path data to the Key name "Logging" in the "Setting" section of the .ini file.

```
SetProfileData "E:\Setting.ini", "Setting", "Logging", "E:\Logging_Folder"
```

Usable Modules:
Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 6.40 or later

Related Items:

DeleteProfileData on page 8-71
GetProfileData on page 8-145

SetSceneData

Sets data for the scene control macro or scene variables.

Format:
SetSceneData <dataIdent>, <data>

Parameter:

Parameter name	Data type	Description
<dataIdent>	Character string	Identification name of data to be set
<data>	Integer Real number Character string	Data to set

Return value:
None.

Description:
Executes the process associated with the identification name specified in the <dataIdent> parameter. In addition to the variable name used in the scene control macro program, the following identification names can be specified in the <dataIdent> parameter.

- Scene variable name: Set the value specified with <data> to the variable of "Scene variable name". The Scene variable name is a character string following "SC." displayed on the scene variable registration screen of TDM editor.
- "direct": Executes the scene control macro specified in the <data> parameter.
- "gosub": Executes the subroutine of the scene control macro specified in the <data>.

If an error occurs during execution of the process when "direct" or "gosub" is specified in the <dataIdent> parameter, the error will occur in the corresponding location as a scene control macro error, and then an "Illegal function call" error will occur in this macro function as an error of the module that called the macro function.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
Type mismatch error is occurred when wrong data is specified as parameter. Illegal function call error is not occurred even if non-exist number, values, combination of data or values.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:

- Execute this macro function when the BUSY signal or other measurement in progress signal is ON and measurement is prohibited. (For details, refer to *5-1-4 State Transitions and Execution Timing* on page 5-13.)

Example:
Set an arbitrary value to a system variable.

```
Rem Set an integer value of 123 when a variable of SC.ABC& is registered in the sc
ene variable.
SetSceneData "ABC&", 123

Rem When an array variable of SC.ASDF&() is registered in the scene variable,
Rem set an integer value of 123 to the first array element.
SetSceneData "ASDF&(0)", 123
```

Sets the communication command parameter received by the communication command macro in the variable of the scene control macro, and executes the subroutine that applies this value in the appropriate unit.

```
Rem Set the parameter received by the communication command in the variable of the
 scene control macro.
SetSceneData "PARAM0&", ARGUMENTVALUE#(0)
SetSceneData "PARAM1&", ARGUMENTVALUE#(1)

Rem Execute the UPDATE_PARAM subroutine that has been defined in the scene control
 macro.
SetSceneData "gosub", "*UPDATE_PARAM"

Rem Execute the Print command that has been defined in the scene control macro.
SetSceneData "direct", "print ValueN&"
```

Usable Modules:

- When applying to scene variables:
  Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro
- When applying to other than scene variables:
  Communication Command Macro

Supported Versions:

Version 5.20 or later
(For scene variables, Version 6.30 or later)

Related Items:

GetSceneData on page 8-146

SetSceneDescription

Sets the scene description.

Format:
SetSceneDescription <sceneNo>, <sceneDescription>

Parameter:

Parameter name	Data type	Description
<sceneNo>	Integer	Scene number (0 to 127) of scene whose description is to be gotten.
<sceneDescription>	Character string	Scene description

Return value:
None.

Description:
Sets the description specified in the <sceneDescription> parameter in the description of the scene of the scene number specified in the <sceneNo> parameter.
The scene description can be set in the maintenance screen or by executing the SetSceneDescription function. (For details, refer to *Editing Scenes* in the *Vision System FH/ FHV Series User's Manual (Cat. No. Z365)*.)
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
An error of "String too long" will be occurred when you specify the character strings as character strings type exceeds 255 characters.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
- Execute this macro function when the BUSY signal or other measurement in progress signal is ON and measurement is prohibited. (For details, refer to *5-1-4 State Transitions and Execution Timing* on page 5-13.)

Example:
Gets the description of scene 1, and if a description is not set, setting the description.

```
Rem Get the scene description.
DESCRIPTION$ = SceneDescription$(1)

If DESCRIPTION$ = "" Then
   Rem Set the scene description.
   SetSceneDescription 1, "Description 1"
Endif
```

Usable Modules:
Scene Control Macro / Communication Command Macro

Supported Versions:
Version 3.50 or later

Related Items:

SceneMaker$ on page 8-262
SceneDescription$ on page 8-258
SetSceneTitle on page 8-292
SceneTitle$ on page 8-264
SetSceneMaker on page 8-291

SetSceneGroupData

Sets the scene group data with the specified identification name.

Format:
SetSceneGroupData <sceneGroupNo>, <dataIdent>, <data>

Parameter:

Parameter name	Data type	Description
<sceneGroupNo>	Integer	Scene group number
<dataIdent>	Character string	Identification name of scene group data For the <dataIdent> parameter, one of the following identification names can be specified. "saveReload": Specify -1 for <sceneGroupNo> and 1 for <data>. The current scene group will be reloaded. The current scene group will be saved when the scene group is reloaded. "sceneGroupTitle": The character string specified in <data> will be set to the scene group title. The scene group titles can be specified in multilingual format using language identifier. For the description method using language identifier, refer to the *Inputting text* in the *Vision System FH/FHV Series User's Manual (Cat. No. Z365)*.
<data>	Character string	Data to be set

Return value:
None.

Description:
Executes the processing corresponding to the identification name specified by the <dataIdent> parameter for the scene group specified by the <sceneGroupNo> parameter.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, no error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
- Execute this macro function when the BUSY signal or other measurement in progress signal is ON and measurement is prohibited. (For details, refer to *5-1-4 State Transitions and Execution Timing* on page 5-13.)

Example:
Changes the title of the current scene group.

```
MeasureStop
  SetSceneGroupData -1, "sceneGroupTitle", "\eng\Sample\esp\Muestra"
MeasureStart
```

Usable Modules:
Scene Control Macro / Communication Command Macro

Supported Versions:
Version 6.50 or later

Related Items:

GetSceneGroupData on page 8-147
MeasureStart on page 8-200
MeasureStop on page 8-201

SetSceneGroupTitle

Sets the title of the scene group.

Format:
SetSceneGroupTitle <sceneGroupNo>, <title>

Parameter:

Parameter name	Data type	Description
<sceneGroupNo>	Integer	Scene group number of the scene group whose title is to be set (-1 only)
<title>	Character string	Title of scene group

Return value:
None.

Description:
Sets the title specified in the <title> parameter in the title of the scene group with the scene group number specified in the <sceneGroupNo> parameter.
If a title with 32 or more characters is specified in the <title> parameter, the first 31 characters are set in the title.
The scene group title can be set by executing the SetSceneGroupTitle function, or in the scene maintenance screen. (For details, refer to *Editing Scenes* in the *Vision System FH/ FHV Series User's Manual (Cat. No. Z365)*.)
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
An error of "String too long" will be occurred when you specify the character strings as character strings type exceeds 255 characters.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
- Execute this macro function when the BUSY signal or other measurement in progress signal is ON and measurement is prohibited. (For details, refer to *5-1-4 State Transitions and Execution Timing* on page 5-13.)

Example:
Sets the title of the current scene group.

```
SetSceneGroupTitle -1, "Title"
```

Usable Modules:
Scene Control Macro / Communication Command Macro

Supported Versions:
Version 3.50 or later

Related Items:

SceneGroupTitle$ on page 8-261
SceneTitle$ on page 8-264
SetSceneTitle on page 8-292

SetSceneMaker

Sets the creator of the scene.

Format:
SetSceneMaker <sceneNo>, <sceneMaker>

Parameter:

Parameter name	Data type	Description
<sceneNo>	Integer	Scene number (0 to 127) of the scene whose creator is to be set.
<sceneMaker>	Character string	Creator of the scene

Return value:
None.

Description:
Sets the creator specified in the <sceneMaker> parameter in the creator of the scene that has the scene number specified in the <sceneNo> parameter.
When a creator name with 32 or more characters is specified in the <sceneMaker> parameter, the first 31 characters are set in the creator.
The scene creator can be set in the scene maintenance screen, or by executing the SetSceneMaker function. (For details, refer to *Editing Scenes* in the *Vision System FH/ FHV Series User's Manual (Cat. No. Z365)*.)
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
An error of "String too long" will be occurred when you specify the character strings as character strings type exceeds 255 characters.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
- Execute this macro function when the BUSY signal or other measurement in progress signal is ON and measurement is prohibited. (For details, refer to *5-1-4 State Transitions and Execution Timing* on page 5-13.)

Example:
Gets the creator of scene 3, and if not set, setting the creator.

```
Rem Get the creator of the scene.
NAME$ = SceneMaker$(3)

If NAME$ = "" Then
   Rem Set the creator of the scene.
   SetSceneMaker 3, "Maker"
Endif
```

Usable Modules:
Scene Control Macro / Communication Command Macro

Supported Versions:
Version 3.50 or later

Related Items:

SceneDescription$ on page 8-258
SceneTitle$ on page 8-264
SetSceneTitle on page 8-292
SceneMaker$ on page 8-262
SetSceneDescription on page 8-288

SetSceneTitle

Sets the title of a scene.

Format:
SetSceneTitle <sceneNo>, <title>

Parameter:

Parameter name	Data type	Description
<sceneNo>	Integer	Scene number (0 to 127) of the scene for which a title is to be set.
<title>	Character string	Scene title

Return value:
None.

Description:
Sets the title specified in the <title> parameter in the title of the scene with the scene number specified in the <sceneNo> parameter.
If a title with 32 or more characters is specified in the <title> parameter, the first 31 characters are set in the title.
The scene title can be set in the scene maintenance screen or flow edit screen, or by executing the SetSceneTitle function. (For details, refer to *Editing Scenes* and *Editing Processing Units in Scenes* in the *Vision System FH/FHV Series User's Manual (Cat. No. Z365)*.)
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
An error of "String too long" will be occurred when you specify the character strings as character strings type exceeds 255 characters.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:

- Execute this macro function when the BUSY signal or other measurement in progress signal is ON and measurement is prohibited. (For details, refer to *5-1-4 State Transitions and Execution Timing* on page 5-13.)

Example:
Sets the title of scene 2.

```
SetSceneTitle 2, "Title"
```

Usable Modules:
Scene Control Macro / Communication Command Macro

Supported Versions:
Version 3.50 or later

Related Items:

SceneDescription$ on page 8-258
SceneMaker$ on page 8-262
SceneTitle$ on page 8-264
SetSceneDescription on page 8-288
SetSceneMaker on page 8-291

SetStop

Sets the conditions for stopping program execution.

Format:
SetStop <string>

Parameter:

Parameter name	Data type	Description
<string>	Character string	Execution stop condition label

Return value:
None.

Description:
Set the character string specified in the <string> parameter as the stop condition for stopping program execution with the Stop function.
By specifying the execution stop condition character string set with the SetStop function as the parameter for the Stop function, you can stop program execution when the execution form is debug mode.
Characters * (character string wildcard operator) and ? (single character wildcard operator) can be used as wildcards for the <string> specification.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
An error of "String too long" will be occurred when you specify the character strings as character strings type exceeds 255 characters.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:

Stops the program execution using the specified condition using the SetStop function in debug mode.

```
Rem Set the execution form to debug mode.
Debug 18
SetStop "AA?"
SetStop "B*"

Rem If character string "AAB" specified in the Stop function argument matches patt
ern "AA?", the program stops.
Stop "AAB"

Rem If character string "AABB" specified in the Stop function argument does not ma
tch pattern "AA?", the program does not stop.
Stop "AABB"

Rem If character string "BCDEF" specified in the Stop function argument matches pa
ttern "B*", the program stops.
Stop "BCDEF"

Rem If character string "CDEF" specified in the Stop function argument does not ma
tch pattern "B*", the program does not stop.
Stop "CDEF"

Rem Set the execution form to release mode.
Debug 1
```

Usable Modules:

Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:

Version 5.20 or later

Related Items:

Cls on page 8-52
Cont on page 8-53
Debug on page 8-68
DebugPrint on page 8-69
List on page 8-184
Print on page 8-221
SetVar on page 8-307
Stop on page 8-312
SubList on page 8-320
VarList on page 8-353

SetSystemData

Sets the system data or system variable.

Format:

SetSystemData <dataIdent0>, <dataIdent1>, <data>

Parameter:

Parameter name	Data type	Description
<dataIdent0>	Character string	Data identification name of identification information 0 of system data to be set.
<dataIdent1>	Character string	Data identification name of identification information 1 of system data to be set.
<data>	Integer Real number Character string	Value of the system data to set

Return value:

- Execute this macro function when the BUSY signal or other measurement in progress signal is ON and measurement is prohibited. (For details, refer to *5-1-4 State Transitions and Execution Timing* on page 5-13.)

Description:

- When setting values to the system data:
  Sets the value specified in the <data> parameter in the system data of identification information 1 specified in the <dataIdent1> parameter, which belongs to identification information 0 specified in the <dataIdent0> parameter. For the identification information list, refer to the system data list. (Refe to *9-3 System Data List* on page 9-10.)
- When setting values to system variables:
  Set a character string for "macro_sy" to <dataident0>, and the variable name following "SY." displayed on the system variable registration screen of TDM editor to <dataident1>.

If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If an identification name that does not exist is specified as the parameter, an "Illegal function call" error will occur.
If a character string longer than 255 characters is specified in the <dataIdent1> parameter, a "String too long" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:

None.

Example:

Set an arbitrary value to a system variable.

```
Rem Set an integer value of 123 when a variable of SY.ABC& is registered in the sy
stem variable.
SetSystemData "macro_sy", "ABC&", 123

Rem When an array variable of SY.ASDF&() is registered in the system variable,
Rem set an integer value of 123 to the first array element.
SetSystemData "macro_sy", "ASDF&(0)", 123
```

Sets the value "E:\temp\bmp" in the screen capture destination folder of identification information 1, "captureDirectory", which belongs to the measurement control settings of identification information 0, "Measure".

```
Rem Get the destination path to be set as the screen capture destination folder.
DIRNAME$ = "E:\temp\bmp"

Rem Set the screen capture destination folder that belongs to the measurement cont
rol settings.
SetSystemData "Measure", "captureDirectory", DIRNAME$
```

Usable Modules:

Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:

Version 3.50 or later
(For scene variables, Version 6.30 or later)

Related Items:

AddSystemData on page 8-27
GetSystemData on page 8-148
SetGlobalData on page 8-275
SetUnitData on page 8-300

SetTextStyle

Set the draw attributes of the character string.

Format:

SetTextStyle <fontSize>, <align>, <color>, <angle>, <style>

Parameter:

Parameter name	Data type	Description
<fontSize>	Integer	Font size of the drawn character string
<align>	Integer	Alignment of the drawn character string TA_BASELINE: Align baseline TA_BOTTOM: Align lower end TA_TOP: Align top TA_CENTER: Align horizontal center TA_LEFT: Align left TA_RIGHT: Align right TA_NOUPDATECP: Current position not update TA_RTLRENDING: Right to left TA_UPDATECP Current position update
<color>	Integer	Color value of the character string color to be drawn JUDGE_NC: Unmeasured color (Grey) JUDGE_OK: OK judgement color (Green) JUDGE_NG: NG judgement color (Red) RGB Function: Any color
<angle>	Integer	Rotation angle (0 to 359) of the drawn character string
<style>	Integer	Font style of the drawn character string FONTSTYLE_NORMAL: Normal FONTSTYLE_BOLD: Bold FONTSTYLE_ITALIC: Italic FONTSTYLE_UNDERLINE: Underline FONTSTYLE_STRIKEOUT: Strike-through

Return value:

None.

Description:

Sets the following parameters as the drawing attributes: specified font size by the <fontSize> parameter, the specified string alignment by the <align> parameter, the specified string color by the <color> parameter, the specified string rotation angle by the <angle> parameter, and the specified string style by the <style> parameter. Before executing the DrawTextG image screen window control function, execute this macro function to draw the graphic figure using the set drawing attribute. Use the SetDRAWStyle function to set the drawing attribute used for macro functions that draw graphic figures. (Refer to *SetDrawStyle* on page 8-272.)

Disjunctive specification of TA_BOTTOM, TA_TOP, TA_LEFT, TA_CENTER, and TA_RIGHT in the <align> parameter is possible.

The gotten color value by the RGB function can be set for the <color> parameter. (Refer to *RGB* on page 8-242.)

If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.

If a value outside the range -2147483648 to 2147483647 is specified as an integer parameter, an "Overflow" error will occur.

If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:

- This macro function can only be used in the *MEASUREDISPI subroutine or the *MEASUREDISPG subroutine. If used in another subroutine, an "Illegal function call" error will occur.

Example:

Uses the *MEASUREDISPG subroutine of the Unit Macro processing unit to display a character string with its font size of 20, aligned to bottom and horizontally centered, colored by the "OK" judgement color, and inclined by 90 degrees.

```
*MEASUREDISPG

   Rem Set the drawing attribute.
   SetTextStyle 20, TA_BOTTOM OR TA_CENTER, JUDGE_OK, 90, FONTSTYLE_ITALIC

   Rem Draw the image.
   DrawTextG "Measurement OK", 100, 100, 0, UnitNo

Return
```

Usable Modules:

Unit Macro

Supported Versions:

Version 3.50 or later

Related Items:

DrawTextG on page 8-111
SetDrawStyle on page 8-272
RGB on page 8-242

SetTextWindow

Set the draw attributes of the character string.

Format:

SetTextWindow <unitNo>, <subNo>, <update>, <visible>

Parameter:

Parameter name	Data type	Description
<unitNo>	Integer	Processing unit number of the target processing unit to display
<subNo>	Integer	Sub number of the target image to display
<update>	Integer	Updated timing (always 0)
<visible>	Integer	Setting of whether to display 0: Window invisible 1: Window visible (This parameter is always invalid i.e., the entered value is always ignored).

Return value:

None.

Description:

Sets the state of the text window.
Specify the value of the displayed processing unit number in the <unitNo> parameter. To link the processing unit displayed in the text window to the flow display, specify -1.
In the <subNo> parameter, 0 should normally be specified for the value of the displayed sub image number. When used with the unit macro, the text window state of a sub image number other than 0 can be set by including the DisplaySubNo macro function in the MEASUREDISPT subroutine.
In the <update> parameter, 0 should always be specified for the update timing.
The value specified in the <visible> parameter is not reflected to the setting.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:

- Execute this macro function when the BUSY signal or other measurement in progress signal is ON and measurement is prohibited. (For details, refer to *5-1-4 State Transitions and Execution Timing* on page 5-13.)

Example:

In the communication command macro, changes the processing unit number of the processing unit displayed in the text display window to the number specified in the communication command argument.

```
Rem Get the state of the text window.
GetTextWindow UNITNO&, SUBNO&, UPDATE&, VISIBLE&

Rem Set the number specified in the command argument in the processing unit number
 that is displayed.
SetTextWindow argumentValue#(0), SUBNO&, UPDATE&, VISIBLE&
```

Usable Modules:

Scene Control Macro / Communication Command Macro

Supported Versions:
Version 3.50 or later

Related Items:

DisplaySubNo on page 8-75
DisplayUnitNo on page 8-76
GetTextWindow on page 8-152
RefreshTextWindow on page 8-236
SetDisplayUnitNo on page 8-271
SetImageWindow on page 8-276
UnitNo on page 8-349
Ut on page 8-351

SetUnitData

Sets the data of a processing unit.

Format:
SetUnitData <unitNo>, <dataNo>, <data>
SetUnitData <unitNo>, <dataIdent>, <data>

Parameter:

Parameter name	Data type	Description
<unitNo>	Integer	Processing unit number (0 to (the number of registered processing units in the current scene minus one))
<dataNo>	Integer	External reference data of the processing unit data to get (For details, refer to *Vision System FH/FHV Series Processing Item Function Reference Manual (Cat. No. Z341)*.)
<dataIdent>	Character string	Data identification name of processing unit data to be gotten.
<data>	Integer Real number Character string	Gotten processing unit data

Return value:
None.

Description:
Sets the data specified in the <data> parameter in the data of the external reference data number specified in the <dataNo> parameter, held by the processing unit specified in the <unitNo> parameter. The data can also be gotten by specifying the <dataIdent> parameter instead of the <dataNo> parameter.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
- Execute this macro function when the BUSY signal or other measurement in progress signal is ON and measurement is prohibited. (For details, refer to *5-1-4 State Transitions and Execution Timing* on page 5-13.)

- Use this macro function with the measurement image displayed after one or more measurements, or after the image file is specified and re-measured.
- This macro function automatically converts character strings to numeric values when that is the required input format. Therefore, it recognizes and sets the available range as values from the head of strings when you set the character strings including the values.

Example:
Sets *Reflect to overall judgement* of the search processing unit of Processing Unit number 2 to *OFF*. *Reflect to overall judgement* is external reference data number 103 and external reference data identification name "overallJudge".

```
SetUnitData 2, 103, 1

Rem The same result will be gotten if "overallJudge" is specified instead of 103.
SetUnitData 2, "overallJudge", 1
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

GetUnitData on page 8-153
GetUnitFigure on page 8-154
MeasureStart on page 8-200
MeasureStop on page 8-201
UnitNo on page 8-349
Ut on page 8-351

SetUnitFigure

Sets the figure data of the processing unit.

Format:
SetUnitFigure <unitNo>, <figureNo>, <figure()>

Parameter:

Parameter name	Data type	Description
<unitNo>	Integer	Processing unit number (0 to (the number of registered processing units in the current scene minus one))
<figureNo>	Integer	Figure number to be set (Refer to *9-7 List of Figure Numbers* on page 9-73.)
<figure()>	Integer array	Figure data to be set (Refer to *9-7 List of Figure Numbers* on page 9-73.)

Return value:
None.

Description:
Sets the figure data specified in the <figure()> parameter in the figure specified in the <figureNo> parameter of the processing unit specified in the <unitNo> parameter. If the model figure is set in figure data, register the model again.

In the <figure()> parameter, specify the 1D integer array variable that will hold the figure data by adding only () without specifying an element number.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:

- Execute this macro function when the BUSY signal or other measurement in progress signal is ON and measurement is prohibited. (For details, refer to *5-1-4 State Transitions and Execution Timing* on page 5-13.)
- Use this macro function with the measurement image displayed after one or more measurements, or after the image file is specified and re-measured.
- Set the figure data so that pixels from outside the image are not included in the figure.

Example:

Changes the position of the region figure (rectangle) of the search processing unit of Processing Unit number 2.

```
Dim FIGURE&(5)

Rem Get the figure data of the processing unit.
GetUnitFigure 2, 1, FIGURE&()

Rem Based on the gotten figure data, change the values of the upper left point XY
coordinates and lower right point XY coordinates of the region figure.
FIGURE&(2) = 100
FIGURE&(3) = 100
FIGURE&(4) = 300
FIGURE&(5) = 300

Rem Set the figure data in which the position of the region figure has been change
d in the processing unit.
SetUnitFigure 2, 1, FIGURE&()
```

Usable Modules:

Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:

Version 3.50 or later

Related Items:

CopyUnitFigure on page 8-59
FigureType on page 8-131
GetUnitData on page 8-153
GetUnitFigure on page 8-154
MeasureStart on page 8-200
MeasureStop on page 8-201
SetUnitData on page 8-300
UnitNo on page 8-349
Ut on page 8-351

SetUnitJudge

Sets the judgement result of a processing unit.

Format:

SetUnitJudge <unitNo>, <judge>[, <totalJudgeRefrect>]

Parameter:

Parameter name	Data type	Description
<unitNo>	Integer	Processing unit number (0 to (the number of registered processing units in the current scene minus one))
<judge>	Integer	Judgement result to set JUDGE_NC: No judgement (unmeasured) JUDGE_OK: Judgement result OK JUDGE_NG: Judgement result NG JUDGE_IMAGEERROR: Judgement result error (image format mismatch) JUDGE_MODELERROR: Judgement result error (unregistered model) JUDGE_MEMORYERROR: Judgement result error (insufficient memory) JUDGE_ERROR: Judgement result error (other error)
<totalJudgeRefrect>	Integer	Setting of whether to reflect the overall judgement result False: not reflected True: reflected

Return value:

None.

Description:

Sets the judgement result specified in the <judge> parameter in the judgement result of the processing unit specified in the <unitNo> parameter. If the <totalJudgeRefrect> parameter is omitted, the specified judgement is applied to the overall judgement.

When this macro function is used to set the judgement result of another processing unit, operation is as follows:

- If measurement processing is executed on the processing unit after the judgement result for the first measurement is set with this macro function, the judgement result set with this macro function is overwritten with the measurement processing result of the processing unit.
- If the judgement result is set with this macro function after measurement processing is executed on the processing unit for the first measurement, the judgement result of the processing unit is overwritten by the judgement result set with this macro function.

In both cases, the judgement result can be reflected in the overall judgement prior to overwriting by specifying True with the <totalJudgeRefrect> parameter.

If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.

If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.

If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:

- If an error occurs during the execution of the program in unit macro, the judgment result of the processing unit is NG regardless of the setting of SetUnitJudge.
- The judgment result of the unit operation macro itself is determined by the judgment condition of the operation expression of the unit operation macro. It cannot be set in SetUnitJudge.

Example:
Sets the judgement result of the search processing unit of Processing Unit number 2 in Judgement Result OK.

```
SetUnitJudge 2, JUDGE_OK, True
```

Usable Modules:
Unit Calculation Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

SetUnitData on page 8-300
UnitJudge on page 8-348
Ut on page 8-351
TotalJudge on page 8-328
UnitNo on page 8-349

SetUnitTitle

Sets the title of a processing unit.

Format:
SetUnitTitle <unitNo>, <title>

Parameter:

Parameter name	Data type	Description
<unitNo>	Integer	Processing unit number (0 to (the number of registered processing units in the current scene minus one))
<title>	Character string	Title of processing unit to be set (31 characters max.)

Return value:
None.

Description:
Sets the title specified in the <title> parameter in the title name of the processing unit specified in the <unitNo> parameter. If a title with 32 or more characters is specified in the <title> parameter, the first 31 characters are set in the title.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
An error of "String too long" will be occurred when you specify the character strings as character strings type exceeds 255 characters.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:

- Execute this macro function when the BUSY signal or other measurement in progress signal is ON and measurement is prohibited. (For details, refer to *5-1-4 State Transitions and Execution Timing* on page 5-13.)

Example:
Sets "bolt search" as the search name in the title of the search processing unit of Processing Unit number 2.

```
SetUnitTitle 2, "bolt search"
```

Usable Modules:
Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

MeasureStart on page 8-200
MeasureStop on page 8-201
UnitNo on page 8-349
UnitTitle$ on page 8-350
Ut on page 8-351

SetUserSubroutine

Register a user-defined function that has been defined in the external DDL file.

Format:
SetUserSubroutine <subroutineIdent>, <dllFileName>, <functionName>

Parameter:

Parameter name	Data type	Description
<subroutineIdent>	Character string	Identification name to the user-defined function to be registered
<dllFileName>	Character string	DLL file name
<functionName>	Character string	User-defined function name to be registered

Return value:
None.

Description:
Registers the function that is defined in the specified DLL file by the <dllFileName> and has a specified function name by the <functionName> parameter as the specified identification name by the <subroutineIdent> parameter. If registered by this macro function, the user-defined functions can be called using the Call function and specify the identification names.
In the <dllFileName> parameter, specify the file name without the file extension ".dll".
Normally use the SetUserSubroutine statement in the *MCRINIT subroutine and execute the subroutine to perform this macro function, or execute this macro function before executing the Call function.
User-defined functions can only be registered by executing this the SetUserSubroutine function using the macro customize function. Execute this command on all processing units, all scene control macros in all scene, and all communications commands in all communications command macros that call and execute user-defined functions.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.

If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If a character string longer than 255 characters is specified in the <subroutineIdent> parameter or the <functionName> parameter, a "String too long" error will occur.
If a character string of a file name (including the absolute path name) longer than 255 characters is specified in the <dllFileName> parameter, a "String too long" error will occur.
If a registered user-defined function by this macro function has not been programmed with the supported interfaces, an error will not occur. In this case, an error will occur at the execution of the user-defined function processing with this Call function.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:

- Only the user-defined functions that have been defined in programmed DLL files by the supported interfaces are accepted to This macro function only accepts the user-defined function that have been defined in programmed DLL files by the supported interfaces. Also, the DLL files must be saved in the same directory as the FZ-CoreRA.exe For user-defined functions creation, use the FH-AP1.

Example:

With identification name "USR", registers a user-defined function "UserProc0" that has been defined in MacroUserProc.dll. Then, specifies the identification name to call the user-defined function and executes it.

```
Rem Register the user-defined function so that the function can be used in this pr
ogram
SetUserSubroutine "USR", "MacroUserProc", "UserProc0"

Rem Call the registered user-defined function and execute it
Call "USR", 0
```

Usable Modules:

Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:

Version 5.20 or later

Related Items:

Call (User-defined Function Execution) on page 8-40

SetVar

Sets all variables with the specified variable names.

Format:
SetVar <variableName>, <value>

Parameter:

Parameter name	Data type	Description
<variableName>	Character string	Name of variable to be set
<value>	Integer Real number Character string	Value to be set

Return value:
None.

Description:
This sets the value specified in the <value> parameter in the variable specified in the <variableName> parameter. Set a value of the same data type in the <value> parameter as the variable specified in the <variableName> parameter.
Characters * (character string wildcard operator) and ? (single character wildcard operator) can be used as wildcards for the <variableName> specification. When using a wildcard to specify multiple variables in the <variableName> parameter, make sure that the specified variables are the same data type.
Wildcards can be used to specify file names in the following manner.

*	Specify all variables.
???	Specify variables with a 3-character variable name.
A*	Specify variables with a variable name that starts with "A".
*A*A*A*	Specify variables with a variable name that includes at least three "A" letters.
????*	Specify variables with a variable name that consists of four or more characters.

If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
Type mismatch error is occurred when wrong data is specified as parameter. Illegal function call error is not occurred even if non-exist number, values, combination of data or values.
If a character string longer than 255 characters is specified in the <variableName> parameter, a "String too long" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
- Execute this macro function when the BUSY signal or other measurement in progress signal is ON and measurement is prohibited. (For details, refer to *5-1-4 State Transitions and Execution Timing* on page 5-13.)

Example:
Among the variables registered as reference variables of the Unit Macro processing unit of Processing Unit number 1, i.e., A01@, A02@, AA01@, AB01@, B01@, and B02@, sets the values of all reference variables that start with "A", i.e., A01@, A02@, AA01@, and AB01@.

```
A01@ = 100
A02@ = 100
AA01@ = 100
AB01@ = 100
B01@ = 100
B02@ = 100

Rem Set "123" only to the variables whose name start with "A".
SetVar "A*@", 123
```

The result is shown below.

```
A01@ = 123
A02@ = 123
AA01@ = 123
AB01@ = 123
B01@ = 100
B02@ = 100
```

Usable Modules:

Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:

Version 5.20 or later

Related Items:

Cont on page 8-53
Debug on page 8-68
DebugPrint on page 8-69
List on page 8-184
Print on page 8-221
SetStop on page 8-293
Stop on page 8-312
VarList on page 8-353

Sin

Gets the sine of the specified expression.

Format:
Sin(<expression>)

Parameter:

Parameter name	Data type	Description
<expression>	Integer Real number	Expression to get the sine

Return value:
Returns the sine as a real number value in the range -1 to 1.

Description:
Gets the sine of the expression specified in the <expression> parameter.
To convert the gotten value to an angle, multiply by π/180.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Gets the sine of 30°.

```
DATA# = Sin(30/180*3.141592)
```

The result is shown below.

```
DATA# = 0.5
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

Atn on page 8-36
GetUnitData on page 8-153
UnitData on page 8-341
Cos on page 8-64
Tan on page 8-323

Sqr

Determines the square root.

Format:
Sqr(<expression>)

Parameter:

Parameter name	Data type	Description
<expression>	Integer Real number	Expression to get the square root

Return value:
Returns the double precision real square root value.

Description:
Gets the square root of the expression specified in the <expression> parameter.
Specify 0 or positive number for the <expression> parameter.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Gets the square root of 256.

```
DATA# = Sqr(256)
```

The result is shown below.

```
DATA# = 16
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

GetUnitData on page 8-153
UnitData on page 8-341

StartTimer

Starts the elapsed time measurement.

Format:
StartTimer

Parameter:
None.

Return value:
Returns a real number type value for specifying the <start> parameter of the Timer function.

Description:
Starts the elapsed time measurement.
After starting measurement of elapsed time with this macro function, get the elapsed time by executing the Timer function.
Execution of the ElapsedTime function is valid only for the processing units such as the unit calculation macro and the Unit Macro processing units. In contrast, executions of this macro function and the Timer function are valid for all macro customize functions.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
The processing time of the processing to be measured is measured four times and displayed.

```
Dim TIME&(3)


For NUM& = 0 To 3
T# = StartTimer
Rem Execute the process to be measured

Rem StartTimer Get the elapsed time using the return value of the function
TIME&(NUM&) = Timer(T#, 0)
Next

For NUM& = 0 To 3
Print(TIME&(NUM&))
Next
```

Do not put anything other than the measurement target, such as a Print function, between the StartTimer function and the Timer function.

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

ElapsedTime on page 8-115
Wait on page 8-362
Timer on page 8-327

Stop

Stops program execution.

Format:
Stop [<string>]

Parameter:

Parameter name	Data type	Description
<string>	Character string	Execution stop condition label This parameter can be omitted. Without If the parameter is not specified, program execution stops at the point that this function is executed.

Return value:
None.

Description:
When the character string specified in the <string> parameter matches the character string set as the execution stop condition by executing the SetStop function, program execution stops.
To resume the stopped program, use the Cont function in the system status console window.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
An error of "String too long" will be occurred when you specify the character strings as character strings type exceeds 255 characters.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
- This macro function is only enabled when specified in debug mode with the Debug function. Specifying other values than the range above will treat the statement with this function in the same manner with the Rem function (i.e., ignores the statement). (For details, refer to *6-1 How to Use the Debug Function* on page 6-2.)
- After the data output to the system status console window, the window is displayed on top of the sensor controller main screen. To display the system status console window on top of the main screen, click [_] on the upper-right of the system status console window or press [Alt] + [Tab] on the connected USB keyboard to the sensor controller.

Example:
Stops the program execution using the specified condition using the SetStop function in debug mode.

```
Rem Set the execution form to debug mode.
Debug 18
SetStop "ABC"

Rem If character string "ABC" specified in the Stop function argument matches patt
ern "ABC", the program stops.
Stop "ABC"

Rem If character string "ABCD" specified in the Stop function argument does not ma
tch pattern "ABC", the program does not stop.
Stop "ABCD"

Rem If the parameter is not specified in the Stop function, program execution stop
s at the point that the Stop function is executed.
Stop

Rem Set the execution form to release mode.
Debug 1
```

Usable Modules:

Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:

Version 5.20 or later

Related Items:

Cont on page 8-53
DebugPrint on page 8-69
Print on page 8-221
SetVar on page 8-307
Debug on page 8-68
List on page 8-184
SetStop on page 8-293
VarList on page 8-353

Str$

Converts a numeric value in the numeric character string.

Format:
Str$(<expression>)

Parameter:

Parameter name	Data type	Description
<expression>	Integer Real number	Expression converted to a numerical character string

Return value:
Returns the character string type numeric value.

Description:
Converts the specified expression in the <expression> parameter to the numeric character string.
Str$ is the inverse function of Val. Val converts the specified numeric character string in numeric value.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a value outside the range -1.0E30 to 1.0E30 is specified for a real number parameter, an "Overflow" error might occur.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Outputs the gotten judgement result and measured correlation value with the search processing unit (Processing Unit number 5) to the system status console window using the MEASUREPROC subroutine in the Unit Macro processing unit. The judgement result and the correlation value can be gotten with External Reference Data numbers 0 and 5 respectively

```
*MEASUREPROC

   Rem Get the measurement result of the processing unit.
   GetUnitData 5, 0, JG&
   GetUnitData 5, 5, CR#

   Rem Convert the gotten measurement result to the numeric character string and o
utput the character string to the system status console window using the Print fun
ction.
   Print Str$(JG&) +"," + Str$(CR#)

Return
```

The result is shown below.

```
1,98.4
```

Usable Modules:

Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

Asc on page 8-34
GetUnitData on page 8-153
LCase$ on page 8-179
Len on page 8-181
Piece$ on page 8-219
Right$ on page 8-243
UCase$ on page 8-339
Chr$ on page 8-44
Hex$ on page 8-158
Left$ on page 8-180
Mid$ on page 8-205
Print on page 8-221
Str2$ on page 8-315
Val on page 8-352

Str2$

Converts a value to a numeric character string in the specified formats.

Format:
Str2$(<expression>, <integral>, <fixed>, <zeroSuppression>, <negative>)

Parameter:

Parameter name	Data type	Description
<expression>	Integer Real number	Expression converted to a numerical character string
<integral>	Integer	Number of digits in integer part (0 to 15)
<fixed>	Integer	Number of digits in decimal part (0 to 15)
<zeroSuppression>	Integer	Inserting character to the unused integer places 0: Fill in with spaces 1: Fill in with zeros
<negative>	Integer	Negative number sign 0: - 1: 8

Return value:
Returns the character string type numeric value.

Description:
Converts the expression specified in the <expression> parameter to the numeric character string after the following processes: Adjust number of integer places to the <integral> parameter and number of decimal places to the <fixed> parameter, insert the character specified in the <zeroSuppression> parameter to the unused places, and replace the negative sign with the sign specified in the <negative> parameter.
Str$ is the inverse function of Val. There is no inverse function for Str2$. Val converts the specified numeric character string in numeric value.
If 0 is specified in the <integral> parameter, all digits in the integer part in the <expression> parameter are converted to a numeric character string.

If a smaller number than number of integer places in the <expression> parameter is specified in the <integral> parameter, a maximum positive number (or a minimum negative number) that can be expressed with the specified number of integer digits in the <expression> parameter is returned as a numeral character string.

- Example: Specify the following parameters: Number of digits in the integer part: 2, number of digits in the decimal part: 0
  When <expression> parameter is 179.099, the changed numerical character string is "99".
  If a larger value than the number of integer places in the <expression> parameter is specified in the <integral> parameter, either zeros or spaces (depending on the <zeroSuppression> parameter) are filled to the higher digit places of the converted numeral character string until the digit place number reaches to the <integral> parameter.
  Note that the negative number uses one extra digit in the integer part for the negative number sign (specified in the <negative> parameter).
- Example: Specify the following parameters: Number of digits in the integer part: 3, number of digits in the decimal part: 3
  When <expression> parameter is 999.999, the changed numerical character string is "999.999".
  When <expression> parameter is -999.999, the changed numerical character string is "-99.999".
  If 0 is specified in the <fixed> parameter, the expression in the <expression> parameter is rounded off to the nearest whole number and converted to the numeric character string.
  If a smaller number than number of decimal places in the <expression> parameter is specified in the <fixed> parameter, the expression in the <expression> parameter is rounded off to the nearest number that can be expressed with a <fixed> number and converted to the numeral character string.
- Example: Specify the following parameters: Number of digits in the integer part: 2, number of digits in the decimal part: 4
  When <expression> parameter is 10.12345, the changed numerical character string is "10.1235".
  If a larger number than number of decimal places in the <expression> parameter is specified in the <fixed> parameter, zeros are filled to the lower digit places of the converted numeral character string until the digit place number reaches to the <fixed> parameter.
- Example: Specify the following parameters: Number of digits in the integer part: 2, number of digits in the decimal part: 5
  When <expression> parameter is 10.123, the changed numerical character string is "10.12300".

If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If a value outside the range -2147483648 to 2147483647 is specified as an integer parameter, an "Overflow" error will occur.
If a value outside the range -1.0E30 to 1.0E30 is specified for a real number parameter, an "Overflow" error might occur.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:

Outputs the measured correlation value, measured position coordinates X and Y (Position X and Position Y) with the search processing unit (Processing Unit number 5) to the system status console window. The correlation value, measured position coordinates X and Y can be gotten with External Reference Data numbers 5, 6, and 7 respectively.

```
Rem Get the measurement result of the processing unit.

GetUnitData 5, 6, X#
GetUnitData 5, 7, Y#
GetUnitData 5, 5, CR#

Rem Convert the measurement results to the numeric character strings in the specif
ied format.
RESX$ = Str2$(X#, 3, 3, 0, 0)
RESY$ = Str2$(Y#, 3, 3, 0, 0)
RESCR$ = Str2$(CR#, 3, 0, 0, 0)

Rem Use the Print function to output the strings to the value to the system status
 console window.
Print RESX$ + "," + RESY$ + "," + RESCR$
```

The result is shown below.

```
150.000,359.000, 97
```

Usable Modules:

Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:

Version 3.50 or later

Related Items:

Asc on page 8-34
Hex$ on page 8-158
Left$ on page 8-180
Mid$ on page 8-205
Print on page 8-221
Str$ on page 8-314
Val on page 8-352
Chr$ on page 8-44
LCase$ on page 8-179
Len on page 8-181
Piece$ on page 8-219
Right$ on page 8-243
UCase$ on page 8-339

Sub - End Sub

Defines the subroutine.

Format:

Sub <label>([, <argument0>...<argumentN>])

…

End Sub

Parameter:

Parameter name	Data type	Description
<label>	Character string	Label name of the subroutine to be executed
<argument0>	Integer Real number Character string Array Variant type	Data to be received by the subroutine
:	:	:
<argumentN>	Integer	Data to be received by the subroutine

Return value:

None.

Description:

It is possible to define a subroutine that can be executed from another subroutine in the macro code. For the <argument0> to <argumentN> parameters, local variables or local array variables of any type can be specified. Changing the values of the <argument0> to <argumentN> parameters passed as arguments in this subroutine updates the values of the variables specified as arguments at the caller of the Call function. When using a local array variable as an argument, specify the entire array without specifying the array element number, such as %AA&().% It is not possible to specify one element of a local array variable, such as AA&(0), in <argument>. If a local array variable is passed as an argument, the array size can be changed by the ReDim command.

If a variable other than local variables or local array variables is specified for the <argument0> to <argumentN> parameters, a Syntax error will occur.

Usage Cautions:

None.

Example:

Executes the *ValueInit subroutine defined separately in the *MeasureProc subroutine of the unit macro.

```
*MCRINIT
  Dim X#(0)
  Dim Y#(0)
  Dim ANG#(0)
  Dim JG&
Return
*MEASUREPROC
  Rem Execute another subroutine.
  Call *ValueInit(X#(), Y#(), ANG#(), JG&)
Return
Rem Initialize various types of measured values.
Sub *ValueInit(%POSX#(), %POSY#(), %ANGLE#(), %JUDGE&)
  ReDim %POSX#(10)
  ReDim %POSY#(10)
  ReDim %ANGLE#(10)
  %JUDGE& = JUDGE_NC
End Sub
```

Usable Modules:

Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:

Version 6.50 or later

Related Items:

Call (Structured Subroutine Call) on page 8-38

SubList

Outputs a list of subroutines to the console window.

Format:
SubList

Parameter:
None.

Return value:
None.

Description:
Outputs a list of subroutines with arguments that are defined in the macro code to the System status monitoring console window.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Displays a list of subroutines with arguments that are defined in the macro code. The names and arguments of all defined subroutines will be output to the System status monitoring console window.

```
SubList
Sub *TESTSUB1(%AA&, %BB#)
Sub *TESTSUB2(%AA&)
Sub *TESTSUB3(%AA&)
Ok
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 6.50 or later

Related Items:

Cls on page 8-52
Cont on page 8-53
Debug on page 8-68
DebugPrint on page 8-69
List on page 8-184
Print on page 8-221
SetStop on page 8-293

SystemInfo

Gets information on the system.

Format:
SystemInfo(<kind>)

Parameter:

Parameter name	Data type	Description
<kind>	Integer	Type of acquired information 4: Controller model (0: FZ, 256: FH1xx0,FH3xx0, 257: FH-L550,FH-L551, 258: FH2xx0,FH5xx0,FH2xx1,FH5xx1, 259: FH2xx2,FH5xx2, 260: FH2xx3,FH5xx3, 261: FH-L552) 8: Presence or absence of EtherCAT (0: EtherCAT communication board not installed, 2: EtherCAT communication board installed (FH)) 10: bsp version 11: fpga version 20: OS major version 21: OS minor version 22: Number of OS bits (32: 32bit, 64: 64bit) 23: OS build version 100: Maximum number of camera connections

Return value:
Returns the acquired information (integer value).

Description:
Gets the information of the type specified by the <kind> parameter. No error will occur even if an undefined value is specified for the <kind> parameter.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Check if the program is running as simulation software.

```
If SystemInfo(4) > 0 Then
  Rem Describe offline re-measurement processing because the program is running as
 simulation software.
Else
  Rem Describe measurement processing by the camera.
EndIf
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 6.41 or later

Related Items:
None.

SystemReset

Reboots the sensor controller.

Format:
SystemReset

Parameter:
None.

Return value:
None.

Description:
Reboots the sensor controller.
If this command is executed on the Sysmac Studio FH Tools, no action is taken and the process ends.

Usage Cautions:
None.

Example:
After loading the system data, executes **Save data**. To apply the loaded settings in the sensor controller, restart the sensor controller.

```
Rem Load the file of the system data
LoadSystemData "E:\BACKDIR\backupsysset.ini"

Rem Save to the controller.
SaveData

Rem Reboot the sensor controller.
SystemReset
```

Usable Modules:
Communication Command Macro

Supported Versions:
Version 3.50 or later

Related Items:

Date$ on page 8-67
Left$ on page 8-180
SaveData on page 8-248
ExitFzProcess on page 8-127
LoadSystemData on page 8-188

Tan

Gets the tangent of the specified expression.

Format:
Tan(<expression>)

Parameter:

Parameter name	Data type	Description
<expression>	Integer Real number	Expression to get the tangent

Return value:
Returns the double precision real tangent value.

Description:
Gets the tangent of the expression specified in the <expression> parameter.
To convert the gotten value to an angle, multiply by π/180.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Gets the tangent of 45°.

```
DATA# = Tan(45/180*3.141592)
```

The result is shown below.

```
DATA# = 0.999999673205
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

Atn on page 8-36
GetUnitData on page 8-153
UnitData on page 8-341
Cos on page 8-64
Sin on page 8-309

TestMeasure

Executes the test measurement.

Format:
TestMeasure [<preImageNo> | <fileName>]

Parameter:

Parameter name	Data type	Description
<preImageNo>	Integer	The number of image logging for measurement test. (Counts of logging images have been done -1) -1: Latest input image 0: Latest logging image of sensor controller
<fileName>	Character string	Name of image file for measurement test

Return value:
None.

Description:
TestMeaure is a command for measuring the specified image as a measurement image. When you use this command, its measurement does not include a measurement result. This command is recommended when you want to avoid a situation that Image input is not executed after sensor controller startup or Scene switch.
Measurement test will be executed for images of the number of image logging specified <preImageNo> parameter or images of the name of image file specified <fileName> parameter.
In the <preImageNo> parameter, specify the number of the logging image already logged as a logging image in the main unit. (For details, refer to *Setting Logging Conditions [Logging Setting]* in the *Vision System FH/FHV Series User's Manual (Cat. No. Z365)*.)
Measurement test will be executed for latest input image when you specify -1 as <preImageNo> parameter.
Measurement test may be executed for latest sensor controller's logging image when you specify 0 as <preImageNo> parameter.
If you omitted augment, measurement test will be input an image from camera.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
Type mismatch error is occurred when wrong data is specified as parameter. Illegal function call error is not occurred even if non-exist number, values, combination of data or values.
If a value outside the range -2147483648 to 2147483647 is specified as an integer parameter, an "Overflow" error will occur.
An error of "String too long" will be occurred when you specify the character strings as character strings type exceeds 255 characters.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
- Execute this macro function when the BUSY signal or other measurement in progress signal is ON and measurement is prohibited. (For details, refer to *5-1-4 State Transitions and Execution Timing* on page 5-13.)
- Do not write in *MCRINIT.

Example:

Loads specified registered image as measurement image.

```
Rem Acquires the save direction of registered image.
USBPATH$ = ApplicationPath$(2)
filename$ = argumentstring$(0)
FilePATH$ = USBPATH$ + "RegisteredImage\" + filename$ + ".ifz"

Rem Loads as a measurement image.
Testmeasure FilePATH$
```

Usable Modules:

Scene Control Macro / Communication Command Macro

Supported Versions:

Version 5.40 or later

Related Items:

ApplicationPath$ on page 8-29
MeasureStart on page 8-200
Remeasure on page 8-239
Measure on page 8-193
MeasureStop on page 8-201

Time$

Reads out the clock time from the internal clock.

Format:
Time$

Parameter:
None.

Return value:
Returns the character string type time value.
The time value is a character string of the internal clock time whose hour (HH), minute (MM), and second (SS) separated by a colon (:). The ranges of the hour, minute, and second values are as follows.

- Hour (HH) : 00 to 23
- Minute (MM) : 00 to 59
- Second (SS) : 00 to 59

Description:
Reads the time from the internal clock and returns the time value (HH, MM, SS) in character string format.
The internal clock can be adjusted in Date-time Settings under System settings. (For details, refer to *Date-time setting [Other]* in the *Vision System FH/FHV Series User's Manual (Cat. No. Z365)).*
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Outputs to the read internal clock value to the system status console window.

```
Dim NOW$(2)

Rem Read out the clock time from the internal clock.
NOW$ = Time$

Rem Change the format of the read out clock time.
For I&=0 To 2
   NOW$(I&) = Piece$(NOW$, ":", I&+1, I&+1)
Next

Rem Output to the system status console window the time which the format have been
 changed previously.
Print NOW$(0) + ":" + NOW$(1) + ":" + NOW$(2)
```

The result is shown below.

```
01:23:45
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:

Version 3.50 or later

Related Items:

Date$ on page 8-67
Mid$ on page 8-205
Print on page 8-221
GetSystemData on page 8-148
Piece$ on page 8-219
SetSystemData on page 8-295

Timer

Gets the elapsed time.

Format:
Timer(<start>, <mode>)

Parameter:

Parameter name	Data type	Description
<start>	Real number	Rem Return value of the StartTimer function that started the measurement of the elapsed time.
<mode>	Integer	Unit of the elapse time to get 0: ms unit 1: μs unit

Return value:
Returns the elapsed time after the execution of the StartTimer function as an integer value gotten by rounding off digits to the right of the decimal point.

Description:
Gets the elapsed time after the execution of the StartTimer function with the unit specified in the <mode> parameter. (To use this function, specify the return value of the StartTimer function in the <start> parameter.)
Specify the return value of the StartTimer function to be executed prior to this macro function.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
- Depending on the processing time of this macro function and the StartTimer function itself, there may be an error in the elapsed time gotten.

Example:
The processing time of the processing to be measured is measured four times and displayed.

```
Dim TIME&(3)


For NUM& = 0 To 3
T# = StartTimer
Rem Execute the process to be measured

Rem StartTimer Get the elapsed time using the return value of the function
TIME&(NUM&) = Timer(T#, 0)
Next

For NUM& = 0 To 3
Print(TIME&(NUM&))
Next
```

Do not put anything other than the measurement target, such as a Print function, between the StartTimer function and the Timer function.

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

ElapsedTime on page 8-115
Wait on page 8-362
StartTimer on page 8-311

TotalJudge

Gets the total judgement result.

Format:
TotalJudge

Parameter:
None.

Return value:
Returns the overall judgement result as an integer value.
- 0: No judgement (unmeasured)
- 1: Judgement result OK
- -1: Judgement result NG

Description:
Gets the overall judgement result that is the result of execution of the measurement flow.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:

In the *MEASUREDISPG subroutine of the Unit Macro processing unit, gets the overall judgement result and displays a character string in the image window.

```
CHARSTRING$ = ""

Rem Get the total judgement result.
JG& = TotalJudge

Rem Change the displayed character string depending on the overall judgement resul
t.
If JG& = 1 Then
   CHARSTRING$ = "OK"
ElseIf JG& = -1 Then
   CHARSTRING$ = "NG"
ElseIf JG& = 0 Then
   CHARSTRING$ = "NC"
Endif

Rem Display a character string.
DrawTextG CHARSTRING$, 100, 100, 0
```

Usable Modules:

Unit Calculation Macro / Unit Macro

Supported Versions:

Version 3.50 or later

Related Items:

DrawTextG on page 8-111

SetUnitJudge on page 8-303

TransformAngle

Applies the calibration result and position correction amount in the angle value.

Format:
TransformAngle <unitNo>, <imageNo>, <mode>, <srcAngle>, <destAngle>

Parameter:

Parameter name	Data type	Description
<unitNo>	Integer	Processing unit number (0 to (number of processing units of current scene minus one)) of the processing unit that holds the data to be converted
<imageNo>	Integer	Measurement image number (always 0) of the processing unit that holds the data to be transformed
<mode>	Integer	Transformation mode 0: After image transformation. -> Before image transformation. 1: Before image transformation. -> After image transformation. 10: Camera coordinates -> After calibration
<srcAngle>	Real number	Pre-transformation angle gotten from the processing unit that holds the data to be transformed
<destAngle>	Real number	Angle after the transformation

Return value:
None.

Description:
Applies the transformation information specified in the <mode> parameter to the angle value specified in the <srcAngle> parameter on the image specified in the <imageNo> parameter of the processing unit specified in the <unitNo> parameter.
Specify 0, 1, or 10 in the <mode> parameter. If a value other than 0, 1, and 10 is specified, operation after the execution of this macro function will be undefined.
Specify the variable that will contain the transformed angle value with the <destAngle> parameter.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Transforms the value of the measurement angle before calibration measured with the search processing unit of Processing Unit number 2 is applied, to the angle value after calibration is applied.
Measurement angle is the external reference data identification name "angle".

```
Rem Get the measurement result.
GetUnitData 2, "angle", BEFOREANGLE#

Rem Transform to the value after calibration is applied.
TransformAngle 2, 0, 10, BEFOREANGLE#, AFTERANGLE#
```

Usable Modules:
Unit Calculation Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

GetUnitData on page 8-153
TransformDist on page 8-332
TransformXY on page 8-335
Ut on page 8-351
TransformArea on page 8-331
TransformLine on page 8-333
UnitNo on page 8-349

TransformArea

Applies the calibration result and position correction amount in the area value.

Format:
TransformArea <unitNo>, <imageNo>, <mode>, <srcArea>, <destArea>

Parameter:

Parameter name	Data type	Description
<unitNo>	Integer	Processing unit number (0 to (number of processing units of current scene minus one)) of the processing unit that holds the data to be converted
<imageNo>	Integer	Measurement image number (always 0) of the processing unit that holds the data to be transformed
<mode>	Integer	Transformation mode 0: After image transformation. -> Before image transformation. 1: Before image transformation. -> After image transformation. 10: Camera coordinates -> After calibration
<srcArea>	Real number	Pre-transformation area gotten from the processing unit that holds the data to be transformed
<destArea>	Real number	Area after the transformation

Return value:
None.

Description:
Applies the transformation information specified in the <mode> parameter to the area value specified in the <srcArea> parameter on the image specified in the <imageNo> parameter of the processing unit specified in the <unitNo> parameter.
Specify 0, 1, or 10 in the <mode> parameter. If a value other than 0, 1, and 10 is specified, operation after the execution of this macro function will be undefined.
Specify the variable that will contain the transformed area value with the <destArea> parameter.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:

None.

Example:

Transforms the value of the area before calibration measured with the labeling processing unit of Processing Unit number 2 is applied, to the area value after calibration is applied. *Area*is the external reference data identification name "area".

```
Rem Get the measurement result.
GetUnitData 2, "area", BEFOREAREA#

Rem Transform to the value after calibration is applied.
TransformArea 2, 0, 10, BEFOREAREA#, AFTERAREA#
```

Usable Modules:

Unit Calculation Macro / Unit Macro

Supported Versions:

Version 3.50 or later

Related Items:

GetUnitData on page 8-153
TransformDist on page 8-332
TransformXY on page 8-335
Ut on page 8-351
TransformAngle on page 8-330
TransformLine on page 8-333
UnitNo on page 8-349

TransformDist

Applies a calibration result and position correction amount to a distance value.

Format:

TransformDist <unitNo>, <imageNo>, <mode>, <srcDist>, <destDist>

Parameter:

Parameter name	Data type	Description
<unitNo>	Integer	Processing unit number (0 to (number of processing units of current scene minus one)) of the processing unit that holds the data to be converted
<imageNo>	Integer	Measurement image number (always 0) of the processing unit that holds the data to be transformed
<mode>	Integer	Transformation mode 0: After image transformation. -> Before image transformation. 1: Before image transformation. -> After image transformation. 10: Camera coordinates -> After calibration
<srcDist>	Real number	Pre-transformation distance gotten from the processing unit that holds the data to be transformed
<destDist>	Real number	Distance after the transformation

Return value:

None.

Description:

Applies the transformation information specified in the <mode> parameter to the distance value specified in the <srcDist> parameter on the image specified in the <imageNo> parameter, of the processing unit specified in the <unitNo> parameter.
Specify 0, 1, or 10 in the <mode> parameter. If a value other than 0, 1, and 10 is specified, operation after the execution of this macro function will be undefined.
In the <destDist> parameter, specify the variable that holds the transformed distance value.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Transforms the distance value of the average width before calibration is applied, which is measured with the scan edge width processing unit of Processing Unit number 2, to the distance value after calibration is applied. The *average width* is external reference data identification name "width ave".

```
Rem Get the measurement result.
GetUnitData 2, "width_ave", BEFOREDIST#

Rem Transform to the value after calibration is applied.
TransformDist 2, 0, 10, BEFOREDIST#, AFTERDIST#
```

Usable Modules:
Unit Calculation Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

GetUnitData on page 8-153
TransformArea on page 8-331
TransformXY on page 8-335
Ut on page 8-351
TransformAngle on page 8-330
TransformLine on page 8-333
UnitNo on page 8-349

TransformLine

Applies the calibration result and position correction amount to a line component value.

Format:
TransformLine <unitNo>, <imageNo>, <mode>, <srcA>, <srcB>, <srcC>, <destA>, <destB>, <destC>

Parameter:

Parameter name	Data type	Description
<unitNo>	Integer	Processing unit number (0 to (number of processing units of current scene minus one)) of the processing unit that holds the data to be converted

Parameter name	Data type	Description
<imageNo>	Integer	Measurement image number (always 0) of the processing unit that holds the data to be transformed
<mode>	Integer	Transformation mode 0: After image transformation. -> Before image transformation. 1: Before image transformation. -> After image transformation. 10: Camera coordinates -> After calibration
<srcA>	Real number	Pre-transformation line component A gotten from the processing unit that holds the data to be transformed
<srcB>	Real number	Pre-transformation line component B gotten from the processing unit that holds the data to be transformed.
<srcC>	Real number	Pre-transformation line component C gotten from the processing unit that holds the data to be transformed.
<destA>	Real number	Transformed line component A
<destB>	Real number	Transformed line component B
<destC>	Real number	Transformed line component C

Return value:

None.

Description:

Applies the transformation information specified in the <mode> parameter to the line components of the lines specified in the <srcA>, <srcB>, and <srcC> parameters on the image specified in the <imageNo> parameter, of the processing unit specified in the <unitNo> parameter.
Specify 0, 1, or 10 in the <mode> parameter. If a value other than 0, 1, and 10 is specified, operation after the execution of this macro function will be undefined.
In the <destA>, <destB>, and <destC> parameters, specify the variables that will hold the transformed line component values.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:

None.

Example:

Transforms the values of the line components measured with the scan edge position processing unit of Processing Unit number 2 before calibration is applied, to the values of the line components after calibration is applied. *Line component A* is the external reference parameter identification name "coefficientA", *Line component B* is the external reference parameter identification name "coefficientB", and *Line component C* is the external reference parameter identification name "coefficientC".

```
Rem Get the measurement result.
GetUnitData 2, "coefficientA", BEFOREA#
GetUnitData 2, "coefficientB", BEFOREB#
GetUnitData 2, "coefficientC", BEFOREC#

Rem Transform to the value after calibration is applied.
TransformLine 2, 0, 10, BEFOREA#, BEFOREB#, BEFOREC#, AFTERA#, AFTERB#, AFTERC#
```

Usable Modules:
Unit Calculation Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

GetUnitData on page 8-153
TransformAngle on page 8-330
TransformArea on page 8-331
TransformDist on page 8-332
TransformXY on page 8-335
UnitNo on page 8-349
Ut on page 8-351

TransformXY

Applies the calibration result and position correction amount to coordinate values.

Format:
TransformXY <unitNo>, <imageNo>, <mode>, <srcX>, <srcY>, <destX>, <destY>

Parameter:

Parameter name	Data type	Description
<unitNo>	Integer	Processing unit number (0 to (number of processing units of current scene minus one)) of the processing unit that holds the data to be converted
<imageNo>	Integer	Measurement image number (always 0) of the processing unit that holds the data to be transformed
<mode>	Integer	Transformation mode 0: After image transformation. -> Before image transformation. 1: Before image transformation. -> After image transformation. 10: Camera coordinates -> After calibration
<srcX>	Real number	Pre-transformation X coordinate gotten from the processing unit that holds the data to be transformed.
<srcY>	Real number	Pre-transformation Y coordinate gotten from the processing unit that holds the data to be transformed.
<destX>	Real number	Transformed X coordinate
<destY>	Real number	Transformed Y coordinate

Return value:
None.

Description:
Applies the transformation information specified in the <mode> parameter to the coordinate values specified in the <srcX> and <srcY> parameters on the image specified in the <imageNo> parameter, of the processing unit specified in the <unitNo> parameter.
Specify 0, 1, or 10 in the <mode> parameter. If a value other than 0, 1, and 10 is specified, operation after the execution of this macro function will be undefined.
In the <destX> and <destY> parameters, specify the variables that will store the transformed coordinate values.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.

If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Transforms the pre-calibration measurement coordinates measured with the search processing unit of Processing Unit 2 to the measurement coordinates after calibration is applied. *Measurement coordinate X* is the external reference data identification name "positionX", and *Measurement coordinate Y* is the external reference data identification name "positionY".

```
Rem Get the measurement result.
GetUnitData 2, "positionX", BEFOREX#
GetUnitData 2, "positionY", BEFOREY#

Rem Transform to the value after calibration is applied.
TransformXY 2, 0, 10, BEFOREX#, BEFOREY#, AFTERX#, AFTERY#
```

Usable Modules:
Unit Calculation Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

GetUnitData on page 8-153
TransformAngle on page 8-330
TransformArea on page 8-331
TransformDist on page 8-332
TransformLine on page 8-333
UnitNo on page 8-349
Ut on page 8-351

Try Catch End Try

Detects an error occurrence and executes an exception process.

Format:
Try
<statement>
Catch
<exceptionStatement>]
End Try

Parameter:

Parameter name	Data type	Description
<statement>	-	Statement that can make error occurs
<exceptionstate-ment>	-	Statement that is executed when an error occurred

Return value:
None.

Description:
Executes the specified Catch block statement in the <exceptionStatement> parameter if an error is occurred as a result of the Try block statement execution specified in the <statement> parameter.
If there is no error occurrence as a result of executing all statements in the Try block statement, the process execution ends without executing the Catch block statement.
Use the Errno function or the Errcmnd$ function in the Catch block statement to get a macro function name and occurred error number in the Try block statement. (Refer to *Errno* on page 8-120, *Errcmnd$* on page 8-119.)
Errors in the Catch block statement cannot be detected. If statements in the Catch block can cause an error, nest the Try Catch-End Try statement to detect the error occurrence.
If the program process is jumped into or out of the Try and Catch block statements using the Goto function in a statement, unexpected operation may occur.
If neither the Try statement nor the End Try statement is used, either the "CATCH without TRY" or "END TRY without TRY" error will occur depending on the statement that is used.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Uses the Try Catch-End Try statement in the *MEASUREPROC subroutine of the Unit Macro processing unit to detect the error occurrence and get the detected error number.

```
*MEASUREPROC

   Try
      WORK& = 0
      SUMM& = 100 + 200 + 300
      ANS& = SUMM& / WORK&
   Catch
      If Errno = 11 Then
         Rem Output the error number and the error content on the system status co
nsole window.
         Print "Error Number = " + Str$(Errno) + ", Division by Zero"
      Endif
   End Try

Return
```

Usable Modules:

Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:

Version 3.50 or later

Related Items:

Errcmnd$ on page 8-119
Errno on page 8-120
If Then Elseif Else EndIf on page 8-160
Print on page 8-221
Str$ on page 8-314

UCase$

Converts an lower case letter to a upper case letter.

Format:
UCase$(<string>)

Parameter:

Parameter name	Data type	Description
<string>	Character string	Character string contains an alphabet to be converted to upper case.

Return value:
Returns the case converted character string as a string type value.

Description:
Converts the lower case letters in the character strings specified in the <string> parameter to upper case.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a character string longer than 255 characters is specified for a character string parameter, the 255-character string before the 256th character is used for the macro function processing. Characters after the 256th character will be discarded.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Converts an lower case letter to a upper case letter.

```
CHARA1$ = "Measurement Result = 100.0(OK)"

Rem Convert the lower case letters in the character strings to upper case.
CHARA2$ = UCase$(CHARA1$)
```

The result is shown below.

```
CHARA2$ = "MEASUREMENT RESULT = 100.0(OK)"
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 4.20 or later

Related Items:

Asc on page 8-34
Chr$ on page 8-44
Hex$ on page 8-158
LCase$ on page 8-179
Left$ on page 8-180
Len on page 8-181
Mid$ on page 8-205
Piece$ on page 8-219

Right$ on page 8-243
Str$ on page 8-314
Str2$ on page 8-315
Val on page 8-352

UnitCount

Gets the number of registered processing units.

Format:
UnitCount

Parameter:
None.

Return value:
Returns the number of registered processing units as an integer value.

Description:
Gets the number of processing units registered in the current scene.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Adds the search processing unit to the end of the measurement flow.

```
Rem Get the number of processing units registered in the current flow.
UNUM& = UnitCount

Rem Specify the processing item identifier.
IDENT$ = "Search"

Rem Add the "Search" processing item to the end of the flow.
AssignUnit UNUM& , IDENT$
```

Usable Modules:
Scene Control Macro / Communication Command Macro

Supported Versions:
Version 3.50 or later

Related Items:

AssignUnit on page 8-35
CheckUnit on page 8-43
CopyUnit on page 8-58
DeleteUnit on page 8-72
InsertUnit on page 8-168
MeasureStart on page 8-200
MeasureStop on page 8-201
MoveUnit on page 8-209

UnitData

Gets the numerical data of a processing unit.

Format:
UnitData(<unitNo>, <dataNo>)
UnitData(<unitNo>, <dataIdent>)

Parameter:

Parameter name	Data type	Description
<unitNo>	Integer	Processing unit number (0 to (the number of registered processing units in the current scene minus one))
<dataNo>	Integer	External reference data of the processing unit data to get (For details, refer to *Vision System FH/FHV Series Processing Item Function Reference Manual (Cat. No. Z341)*.)
<dataIdent>	Character string	Data identification name of processing unit data to be gotten.

Return value:
Returns processing unit data as integer or double precision real values.
If non-numerical data is gotten, the data is converted to numerical values and returned.

Description:
Gets the data of the external reference data number specified in the <dataNo> parameter, held by the processing unit specified in the <unitNo> parameter. The data can also be gotten by specifying the <dataIdent> parameter instead of the <dataNo> parameter.
To get data other than numerical data, use the UnitData$ function or the GetUnitData function.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Gets the value of the measurement X coordinate of the search processing unit of Processing Unit number 5. The measurement X coordinate is external reference data number 6 and external reference data identification name "X".

```
SEARCH# = UnitData(5, 6)

Rem The same result can be gotten by specifying "X" instead of 6.
SEARCH# = UnitData(5, "X")
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

GetUnitData on page 8-153
UnitData$ on page 8-342
UnitNo on page 8-349
SetUnitData on page 8-300
UnitData2 on page 8-343
Ut on page 8-351

UnitData$

Gets the character string data of the specified processing unit.

Format:
UnitData$(<unitNo>, <dataNo>)
UnitData$(<unitNo>, <dataIdent>)

Parameter:

Parameter name	Data type	Description
<unitNo>	Integer	Processing unit number (0 to (the number of registered processing units in the current scene minus one))
<dataNo>	Integer	External reference data of the processing unit data to get (For details, refer to *Vision System FH/FHV Series Processing Item Function Reference Manual (Cat. No. Z341)*.)
<dataIdent>	Character string	Data identification name of processing unit data to be gotten.

Return value:
Returns the character string data of the processing unit as a string type value.
If non character string data is gotten, the data is converted to a character string and returned.

Description:
Gets the data of the external reference data number specified in the <dataNo> parameter, held by the processing unit specified in the <unitNo> parameter. The data can also be gotten by specifying the <dataIdent> parameter instead of the <dataNo> parameter.
To get data other than character string data, use the UnitData function or the GetUnitData function.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Gets the decode character string of the 2D code processing unit of Processing Unit number 5. The decode character string is external reference data number 7 and external reference data identification name "decodeCharStr".

```
DECODECHAR$ = UnitData$(5, 7)

Rem The same result can be gotten by specifying "decodeCharStr" instead of 7.
DECODECHAR$ = UnitData$(5, "decodeCharStr")
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

GetUnitData on page 8-153
UnitData on page 8-341
UnitNo on page 8-349
SetUnitData on page 8-300
UnitData2 on page 8-343
Ut on page 8-351

UnitData2

Gets the drawing coordinate data of a processing unit.

Format:
UnitData2(<unitNo>, <dataNo>)
UnitData2(<unitNo>, <dataIdent>)

Parameter:

Parameter name	Data type	Description
<unitNo>	Integer	Processing unit number (0 to (the number of registered processing units in the current scene minus one))
<dataNo>	Integer	External reference data of the processing unit data to get (For details, refer to *Vision System FH/FHV Series Processing Item Function Reference Manual (Cat. No. Z341)*.)
<dataIdent>	Character string	Data identification name of processing unit data to be gotten.

Return value:
Returns processing unit data as integer or double precision real values.

Description:
Gets the data of the external reference data number specified in the <dataNo> parameter, held by the processing unit specified in the <unitNo> parameter. The data can also be gotten by specifying the <dataIdent> parameter instead of the <dataNo> parameter.
This macro function can be used to get measurement coordinate values prior to transformation by calibration or otherwise. Use to get drawing coordinates for the display of measurement results in the image window.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
In the *MEASUREDISPG subroutine of the Unit Macro processing unit, gets the measurement X and Y coordinate values of the search processing unit of Processing Unit number 5 and displaying the cursor in the image coordinates. The measurement X coordinate is the external reference data number 8 and the external reference data identification name "X", and the measurement Y coordinate is the external reference data number 7 and the external reference data identification name "Y".
Even when the *Calibration* setting of the search processing unit is *ON*this macro function can be used to get drawing coordinates without concern for the calibration settings.

```
SEARCHX& = UnitData2(5, 6)
SEARCHY& = UnitData2(5, 7)

Rem The same result can be gotten by specifying "X" instead of 6.
SEARCHX& = UnitData2(5, "X")
SEARCHY& = UnitData2(5, "Y")

Rem Display the cursor in the coordinates prior to position correction.
DrawCursor SEARCHX&, SEARCHY&, 0, UnitNo
```

Usable Modules:
Unit Calculation Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

GetUnitData on page 8-153
UnitData on page 8-341
UnitNo on page 8-349
SetUnitData on page 8-300
UnitData$ on page 8-342
Ut on page 8-351

UnitInfo

Gets the processing unit information.

Format:
UnitInfo(<unitNo>, <kind>)

Parameter:

Parameter name	Data type	Description
<unitNo>	Integer	Processing unit number (0 to (number of processing units of current scene minus one)) of the processing unit whose information is to be gotten.
<kind>	Integer	Type of information 0: Processing item type Number that indicates the processing item type. The values below can be gotten. • 0: Inspect and Measure (measurement) • 1: Input image (Image input) • 2: Compensate image (image correction) • 3: Support Inspection and Measurement (supplementary measurement) • 4: Branch (branch control) • 5: Output result (result output) • 6: Display result (result display) 1: Setting data structure size Size of setting data structure. Units are bytes. 2: Measurement data structure size Size of measurement data structure. Units are bytes. 3: Control data structure size Size of control data structure. Units are bytes. 4: The maximum number of figure data Maximum number of figure data items held by processing unit. 5: Maximum number of model data Maximum number of model data items held by processing unit. 6: Maximum number of image data Maximum number of image data items held by processing unit. 7: Maximum number of inner processing unit Maximum number of processing units held (incorporated) by processing unit. 8: Whether camera setting is effective or not Displays whether or not the processing unit updates the camera settings at measurement initialization and other times. "1" is returned when there is an image input processing unit for camera image input. • 0: Camera settings invalid • 1: Camera settings valid

Parameter name	Data type	Description
		9: Whether processing unit measure processing can parallel or not Displays whether or not parallel execution of processing units is possible in the measurement flow during measurement. "1" is returned when the processing units support parallel processing. • 0: Parallel processing disabled • 1: Parallel processing enabled For the parallel processing, refer to *Parallel Processing* in the *Vision System FH/FHV Series User's Manual (Cat. No. Z365)*.

Return value:

Returns processing unit information as an integer value. Returns -1 if information does not exist.

Description:

Gets the information specified in the <kind> parameter of the processing unit specified in the <unitNo> parameter.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:

None.

Example:

Checks if model registration is possible on the processing unit of Processing Unit number 2.

```
If UnitInfo(2, 5) > 0 Then
   Rem Write a model registration processing statements here.
Endif
```

Usable Modules:

Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:

Version 3.50 or later

Related Items:

UnitNo on page 8-349

Ut on page 8-351

UnitItemIdent$

Gets the processing item identification name of the specified processing unit.

Format:
UnitItemIdent$(<unitNo>)

Parameter:

Parameter name	Data type	Description
<unitNo>	Integer	Processing unit number (0 to (number of processing units of current scene minus one)) of processing unit whose processing item identification name*9-5 List for Processing Item Identifier* on page 9-67 is to be gotten.

Return value:
Returns the value of the processing item identification name as a character string.

Description:
Gets the processing item identification name of the processing unit specified in the <unitNo> parameter. If the specified processing unit is not registered on the measurement flow, the null character string ("") is returned.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
Type mismatch error is occurred when wrong data is specified as parameter. Illegal function call error is not occurred even if non-exist number, values, combination of data or values.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
In the scene macro, searches for the search processing unit registered in the measurement flow, and updating the correlation value lower limit, which is the judgement condition, to "70". The search correlation value lower limit is external reference number 143.

```
Rem Get the enrollment number of the processing unit.
COUNT& = UnitCount

Rem Search the search processing unit.
For I&=0 To COUNT&-1
   If UnitItemIdent$(I&) = "Search" Then
      Rem Update the correlation value lower limit, which is the judgement conditi
on of the search processing unit.
      SetUnitData I&, 143, 70
   Endif
Next
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:

Version 3.50 or later

Related Items:

SetUnitData on page 8-300
UnitNo on page 8-349
UnitCount on page 8-340
Ut on page 8-351

UnitJudge

Gets the judgement result of a processing unit.

Format:
UnitJudge(<unitNo>)

Parameter:

Parameter name	Data type	Description
<unitNo>	Integer	Processing unit number to get the judgement result of the processing unit (0 to (the number of registered processing units in the current scene minus one))

Return value:
Returns the judgement result as an integer value.
- 0: No judgement (unmeasured)
- 1: Judgement result OK
- -1: Judgement result NG
- -10: Judgement result error (image format mismatch)
- -11: Judgement result error (unregistered model)
- -12: Judgement result error (insufficient memory)
- -20: Judgement result error (other errors)

Description:
Gets the judgement result of the processing unit specified in the <unitNo> parameter.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Gets the judgement result of the processing unit of Processing Unit number 5.

```
JUDGE& = UnitJudge(5)
```

Usable Modules:
Unit Calculation Macro / Unit Macro

Supported Versions:

Version 3.50 or later

Related Items:

GetUnitData on page 8-153
TotalJudge on page 8-328
Ut on page 8-351
SetUnitJudge on page 8-303
UnitNo on page 8-349

UnitNo

Gets the processing unit number.

Format:
UnitNo

Parameter:
None.

Return value:
Returns the processing unit number as an integer value.

Description:
Gets the processing unit number of the processing unit.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Gets the judgement result of the processing unit registered immediately before the current processing unit.

```
Rem Get the processing unit number of this processing unit.
UNO& = UnitNo

Rem Get the judgement result of the processing unit registered immediately before
this processing unit.
JUDGE& = UnitJudge(UNO& - 1)
```

Usable Modules:
Unit Calculation Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

UnitJudge on page 8-348
Ut on page 8-351

UnitTitle$

Gets the title of a processing unit.

Format:
UnitTitle$(<unitNo>)

Parameter:

Parameter name	Data type	Description
<unitNo>	Integer	Processing unit number (0 to (number of processing units of current scene minus one)) of processing unit whose title is to be gotten.

Return value:
Returns the title as a character string.

Description:
Gets the title of the processing unit specified in the <unitNo> parameter.
The title can be gotten in a language based on the language setting.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
In the scene control macro, searches for the search processing unit registered in the measurement flow with the title "Bolt search", and updates the correlation value lower limit, which is the judgement condition, to "70". The search correlation value lower limit is external reference number 143.

```
Rem Get the enrollment number of the processing unit.
COUNT& = UnitCount

Rem Search for the processing unit with the title "Bolt search".
For I&=0 To COUNT&-1
   If UnitTitle$(I&) = "Bolt search" Then
      Rem Change the correlation value lower limit, which is the judgement conditi
on.
      SetUnitData I&, 143, 70
   Endif
Next
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

SetUnitData on page 8-300
SetUnitTitle on page 8-304
UnitCount on page 8-340
UnitNo on page 8-349
Ut on page 8-351

Ut

Gets a processing unit number based on the specified unit label.

Format:
Ut(<unitLabel>)

Parameter:

Parameter name	Data type	Description
<unitLabel>	Character string	Unit label of processing unit

Return value:
Returns the processing unit number as an integer value.

Description:
Gets the processing unit number of the processing unit that has the set unit label specified in the <unitLabel> parameter.
Knowing the unit label allows you to get the processing unit number, and thus even if the measurement flow is changed and the unit number changes, there is no need to change the program.
Set the unit label in advance with the scene control macro tool. (Refer to *3-1-4 Description of the Setting Screen of the Scene Control Macro Tool and How to Configure Settings* on page 3-13.)
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
In the scene control macro tool, gets the processing unit number of the unit label "Position Search" set in the search processing unit of processing unit number 10, and gets the judgement result.

```
Rem Specify the unit label that was set with the scene control macro tool and get
the processing unit number.
UNITNO& = Ut("PositionSearch")

Rem Using the gotten processing unit number, get the judgement result of the proce
ssing unit.
JG& = UnitJudge(UNITNO&)
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:

Version 5.20 or later

Related Items:

UnitJudge on page 8-348 *UnitNo* on page 8-349

Val

Converts a numeric character string to numeric value.

Format:
Val(<string>)

Parameter:

Parameter name	Data type	Description
<string>	Character string	Numeric character string converted to numeric value

Return value:
Returns the value as the double precision type real number.

Description:
Converts the specified numeric character string in the <string> parameter to the numeric value.
Val is the inverse function of Str$. Str$ converts the specified numeric value to the numeric character string.
Specify a character string starting with either "+", "-", ".", or half-width numbers "0" to "9" to the <string> parameter. If other characters than above is in the first character of the specified character string, 0 is returned.
If there are characters that cannot be converted to numeric values in the specified alphanumeric character string with the <string> parameter, characters from the beginning of the string to one character before the inconvertible character are converted to a numeric value.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
Even if a character string longer than 255 characters is specified for a character string parameter, an error will not occur.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Converts a numeric character string to numeric value.

```
VALUE1# = Val("123.456")
VALUE2# = Val("-123.456")
VALUE3# = Val(".123")
VALUE4# = Val("-.456")
VALUE5# = Val("123"+"."+"456")
VALUE6# = Val("123+456")
```

The result is shown below.

```
VALUE1# = 123.456
VALUE2# = -123.456
VALUE3# = 0.123
VALUE4# = -0.456
VALUE5# = 123.456
VALUE6# = 123
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

Asc on page 8-34
Chr$ on page 8-44
Hex$ on page 8-158
LCase$ on page 8-179
Left$ on page 8-180
Len on page 8-181
Mid$ on page 8-205
Piece$ on page 8-219
Right$ on page 8-243
Str$ on page 8-314
Str2$ on page 8-315
UCase$ on page 8-339

VarList

Outputs a list of the values of the specified variables in the system status console window.

Format:
VarList [<variableName>]

Parameter:

Parameter name	Data type	Description
<variableName>	Character string	Name of variable to be output This parameter can be omitted. When the parameter is not specified, a list of the values of the variables used in the current scope is output.

Return value:
None.

Description:
Outputs the values of the variables specified in the <variableName> parameter to the system status console window.
This macro function cannot output the values of the elements of an array variable. To output the value of each element of an array variable, specify each element of the array variable by entering a question mark (?), a half-width space ( ), and then the array variable and the element number (for example, "? AA&(5)").
Characters * (character string wildcard operator) and ? (single character wildcard operator) can be used as wildcards for the <variableName> specification.
Wildcards can be used to specify file names in the following manner.

*	Specify all variables.
???	Specify variables with a 3-character variable name.

A*	Specify variables with a variable name that starts with "A".
*A*A*A*	Specify variables with a variable name that includes at least three "A" letters.
????*	Specify variables with a variable name that consists of four or more characters.

If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
An error of "String too long" will be occurred when you specify the character strings as character strings type exceeds 255 characters.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:

- Use this macro function when program execution has been stopped by the Stop function.
- The timing of displaying variables by the VarList command changes depending on the use of the Option Explicit command.
  When the Option Explicit command is used, variables are not displayed by setting reference variables and executing the VarList command. They are displayed when the Dim function is used and the VarList command is executed.
  When the Option Explicit command is not used, variables are displayed when reference variables are set and the VarList command is executed.

Example:

Among the variables used by the Unit Macro processing unit of Processing Unit number 1 (AA&, AB#, BB&, ABC$, DEF#), outputs a list of the values of variables AA& and AB$, which start with "A" and consist of three characters including the type identifier.

```
Macro(U1)>VarList "A??"
AA&=123
AB#=123.456
```

Usable Modules:

Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:

Version 5.20 or later

Related Items:

Cls on page 8-52
Cont on page 8-53
Debug on page 8-68
DebugPrint on page 8-69
Dim on page 8-73
List on page 8-184
Option Explicit on page 8-217
Print on page 8-221
SetStop on page 8-293
SetVar on page 8-307
Stop on page 8-312

VarPop

Restores the value of the variables that are saved temporarily.

Format:
VarPop

Parameter:
None.

Return value:
None.

Description:
Restore the values of all variables that were saved by the most recent VarPush function.
If the VarPush function is executed more than once, the saved values are restored from the latest saved variable to the oldest saved variable.
If this function is executed before saving the values with the VarPush function, an "Internal error" will occur.

Usage Cautions:
None.

Example:
Uses the variables used in subroutines as local variables.

```
*EXPA

   Rem Display the current value of the variable.
   Print A&, B&, C&, D#, E#

   Rem Use the EXPA subroutine to save the current values of the variables in orde
r to prepare for treating the variable as local variables.
   VarPush A&, B&, C&, D#, E#

   Rem Use A&, B&, C&, D#, and E# freely.
   GetUnitData 2, "CR", A&
   GetUnitData 3, "CR", B&
   GetUnitData 4, "CR", C&
   GetUnitData 5, "X", D#
   GetUnitData 6, "Y", E#

   Rem Check the current values of the variables before calling the subroutine.
   Print A&, B&, C&, D#, E#

   Rem Variables named A&, B&, C& are used in the *EXPB subroutine.
   Rem These names are also used for variables in this *EXPA subroutine.
   Rem Although being nested with the Gosub statement in this program example,
   Rem saving and restoration of variable values are performed with the Varpush an
d Varpop functions within the *EXPB subroutine,
   Rem so as to prevent variable values from being unintentionally overwritten.
   Gosub *EXPB

   Rem Check the current value of the variables after calling the subroutine.
   Print A&, B&, C&, D#, E#

   Rem Restore the current value of the variables that were saved at the beginning
 of the subroutine EXPA.
   VarPop

Return
```

```
*EXPB

   Rem Use the EXPB subroutine to save the current values of the variables in orde
r to prepare for treating the variable as local variables.
   Rem Values in variables A&, B&, C&, D#, E# are saved in different areas from
   Rem where the Varpush statement in the earlier part of the *EXPA subroutine sav
es.
   Rem This prevents the previously saved values from being overwritten.
   Rem VarPush can be executed up to 16 times consecutively.
   VarPush A&, B&, C&, D#, E#

   Rem Use A&, B&, C&, D#, and E# freely.
   GetUnitData 2, "X", A&
   GetUnitData 3, "X", B&
   GetUnitData 4, "X", C&
   D# = 3
   E# = 100 / 512

   Rem Check the current values of the variables after change.
   Print A&, B&, C&, D#, E#

   Rem Restore the current value of the variables that were saved at the beginning
 of the subroutine EXPB.
   VarPop

Return
```

Usable Modules:

Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:

Version 3.50 or later

Related Items:

VarPush on page 8-358

VarPush

Saves the value of the variables that are saved temporarily.

Format:
VarPush <variable>[, <variable>[,..., <variable>]]

Parameter:

Parameter name	Data type	Description
<variable>	Character string	Variable name of the variable whose value is saved temporarily

Return value:
None.

Description:
Temporarily save the variable value specified in the <variable> parameter. Execute the Varpush function to restore the saved value. (Refer to *VarPop* on page 8-355.)
If this macro function is executed 17 times or more without executing the VarPop function, an "Internal error" will occur. Execute the VarPop function to restore the value so that this function execution count is equal to or less than the Varpop execution count + 16.
If variables except array variable are specified as parameter, a "Type mismatch" error will occur.

Usage Cautions:
None.

Example:
Uses the variables used in subroutines as local variables.

```
*EXPA

   Rem Display the current value of the variable.
   Print A&, B&, C&, D#, E#

   Rem Use the EXPA subroutine to save the current values of the variables in orde
r to prepare for treating the variable as local variables.
   VarPush A&, B&, C&, D#, E#

   Rem Use A&, B&, C&, D#, and E# freely.
   GetUnitData 2, "CR", A&
   GetUnitData 3, "CR", B&
   GetUnitData 4, "CR", C&
   GetUnitData 5, "X", D#
   GetUnitData 6, "Y", E#

   Rem Check the current values of the variables before calling the subroutine.
   Print A&, B&, C&, D#, E#

   Rem Variables named A&, B&, C& are used in the *EXPB subroutine.
   Rem These names are also used for variables in this *EXPA subroutine.
   Rem Although being nested with the Gosub statement in this program example,
   Rem saving and restoration of variable values are performed with the Varpush an
d Varpop functions within the *EXPB subroutine,
   Rem so as to prevent variable values from being unintentionally overwritten.
   Gosub *EXPB

   Rem Check the current value of the variables after calling the subroutine.
   Print A&, B&, C&, D#, E#

   Rem Restore the current value of the variables that were saved at the beginning
 of the subroutine EXPA.
   VarPop

Return
```

```
*EXPB

   Rem Use the EXPB subroutine to save the current values of the variables in orde
r to prepare for treating the variable as local variables.
   Rem Values in variables A&, B&, C&, D#, E# are saved in different areas from
   Rem where the Varpush statement in the earlier part of the *EXPA subroutine sav
es.
   Rem This prevents the previously saved values from being overwritten.
   Rem VarPush can be executed up to 16 times consecutively.
   VarPush A&, B&, C&, D#, E#

   Rem Use A&, B&, C&, D#, and E# freely.
   GetUnitData 2, "X", A&
   GetUnitData 3, "X", B&
   GetUnitData 4, "X", C&
   D# = 3
   E# = 100 / 512

   Rem Check the current values of the variables after change.
   Print A&, B&, C&, D#, E#

   Rem Restore the current value of the variables that were saved at the beginning
 of the subroutine EXPB.
   VarPop

Return
```

Usable Modules:

Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:

Version 3.50 or later

Related Items:

VarPop on page 8-355

VarSave

Saves the values of the variables in the scene data.

Format:
VarSave <variableName>

Parameter:

Parameter name	Data type	Description
<variableName>	Character string	Name of the variables to save

Return value:
None.

Description:
Saves the values of the variables specified in the <variableName> parameter to the scene data.
Characters * (character string wildcard operator) and ? (single character wildcard operator) can be used as wildcards for the <variableName> specification.
Wildcards can be used to specify file names in the following manner.

*	Specify all variables.
???	Specify variables with a 3-character variable name.
A*	Specify variables with a variable name that starts with "A".
*A*A*A*	Specify variables with a variable name that includes at least three "A" letters.
????*	Specify variables with a variable name that consists of four or more characters.

The variable value saved with this macro function will be read when the scene data is loaded.
If this macro function is executed multiple times, values are restored to the original variables in execution order of the VarSave statement (from the oldest saved variable value to the latest saved variable value) at the loading of the scene data.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
Type mismatch error is occurred when wrong data is specified as parameter. Illegal function call error is not occurred even if non-exist number, values, combination of data or values.
An error of "String too long" will be occurred when you specify the character strings as character strings type exceeds 255 characters.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
- This macro function can only be used in the *SAVEPROC subroutine. If used in another subroutine, an "Illegal function call" error will occur.

Example:
In SAVEPROC subroutine of scene control macro, saves the version information managed independently in scene data.

```
*SAVEPROC

   Rem Create a variable to store the version information and set 100.
   Version& = 100

   Rem Save the version information.
   VarSave "Version*"

Return
```

Usable Modules:
Scene Control Macro

Supported Versions:
Version 5.20 or later

Related Items:
None.

Wait

Pauses the program process for the specified amount of time elapses.

Format:
Wait <time>

Parameter:

Parameter name	Data type	Description
<time>	Integer	Standby time (ms)

Return value:
None.

Description:
Pauses the program process on the period of time specified in the <time> parameter.
When the process is performed in the background while waiting, the background process will be performed without waiting.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a value outside the range -2147483648 to 2147483647 is specified as an integer parameter, an "Overflow" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
- An error may occur between the waiting time specified as a parameter and the actual waiting time.

Example:
After switching to the scene 2 with the communication command macro, waits for 10 ms.

```
Rem Switch the scene.
ChangeScene 2

Rem Wait 10ms.
Wait 10
```

Usable Modules:
Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

ChangeScene on page 8-41
StartTimer on page 8-311
ElapsedTime on page 8-115
Timer on page 8-327

WritePlcMemory

Writes values in the PLC memory area.

Format:
WritePlcMemory <ioIdent>, <area>, <channelOffset>, <channelCount>, <writeData()>

Parameter:

Parameter name	Data type	Description
<ioIdent>	Character string	Identification name of the communication module to be used (Refer to *9-4 List of I/O Modules* on page 9-27.)
<area>	Integer	Area type number of data output area to be written to.
<channelOffset>	Integer	Offset from beginning of command area to address where writing is to start.
<channelCount>	Integer	Data size to write (channel unit)
<writeData()>	Integer array	Data to write

Return value:
None.

Description:
Writes the amount of data specified in the <channelCount> parameter from the address offset by the amount of the value specified in the <channelOffset> parameter, of the PLC area type specified in the <area> parameter by using the communication module specified in the <ioIdent> parameter.
Before using this macro function to write data to the PLC memory area, execute the SetPlcData function to set the data to be written.
In the <writeData()> parameter, specify the 1D integer array variable that stores the data to be written, without adding element numbers but adding () to the variables.
In the <area> parameter, specify the identification of the register that is set with the PLC link setting in the system settings. For Fieldbus, specify 0 because the identification cannot be specified.
In the <channelCount> parameter, specify the size in channel units. The size of one integer type data item is two channels (4 bytes), and thus to write one integer value, a one-element array should be prepared with the <writeData()> parameter, and 2 should be specified in the <channelCount> parameter.

If a size larger than the array size specified in the <writeData()> parameter is specified in the <channelCount> parameter, a "Subscript out of range" error will occur.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a non-existent number, numerical value, or combination of data types or values is specified for a parameter, an "Illegal function call" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.
Writes the data to the data output area. Note that if the data is written to the response area, the response data will be overwritten after the command processing.

Usage Cautions:

- Before using this macro function to write data, always use the SetPlcData function to set the data to be written. If the data is directly set in the <writeData()> parameter without using the SetPlcData function, the correct data may not be set.

Example:

In the communication macro, writes measurement coordinate X and measurement coordinate Y of the search processing unit of Processing Unit number 2 to the PLC connected by PLC link. Measurement coordinate X is external data number 6, and measurement coordinate Y is external data number 7.

```
IOMODULE$ = "UdpPlcLink"

Rem Get the measurement result.
GetUnitData 2, 6, X#
GetUnitData 2, 7, Y#

Rem Convert the real number value multiplied by 1,000 to the integer value.
VALUE0& = Int(X# * 1000)
VALUE1& = Int(Y# * 1000)

Rem Get the settings of the output data area.
GetSystemData IOMODULE$, "outputArea", AREA&
GetSystemData IOMODULE$, "outputMemoryAddress", ADDRESS&

Rem Store the data to be written in an integer array variable.
Dim DATA&(1)
SetPlcData IOMODULE$, DATA&(), 0, 4, VALUE0&
SetPlcData IOMODULE$, DATA&(), 4, 4, VALUE1&

Rem Write the data (4ch) in data output area.
WritePlcMemory IOMODULE$, AREA&, ADDRESS&, 4, DATA&()
```

Usable Modules:

Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:

Version 4.20 or later

Related Items:

GetPlcData on page 8-140
GetUnitData on page 8-153
ReadPlcMemory on page 8-229
GetSystemData on page 8-148
Int on page 8-169
SetPlcData on page 8-281

XOR

Gets the exclusive disjunction (XOR) of two expressions.

Format:
<expression1> XOR <expression2>

Parameter:

Parameter name	Data type	Description
<expression1>	Integer	Expression to get the exclusive disjunction
<expression2>	Integer	

Return value:
Returns the XOR value as an integer value.

Description:
Gets the XOR value of the expressions specified in the <expression 1> and <expression 2> parameters. (Each bit of the two expressions is computed separately.)
When the values of the <expression1> parameter and <expression2> parameter are double precision real values, the decimal part is rounded off.
If an incorrect data type is specified for a parameter, a "Type mismatch" error will occur.
If a value outside the range -2147483648 to 2147483647 is specified as an integer parameter, an "Overflow" error will occur.
If a value is assigned to the return value variable or the variable is not used in an expression, a "Syntax error" error will occur.
If the format is written incorrectly, such as writing the macro function name incorrectly, omitting a comma, or omitting a half-width space, a "Syntax error" error will occur.

Usage Cautions:
None.

Example:
Gets the XOR value of 12 and 31.

```
DATA1& = 12
DATA2& = 31

DATA3& = DATA1& XOR DATA2&
```

The result is shown below.

```
DATA3& = 19
```

Usable Modules:
Unit Calculation Macro / Scene Control Macro / Communication Command Macro / Unit Macro

Supported Versions:
Version 3.50 or later

Related Items:

AND on page 8-28
NOT on page 8-210
GetUnitData on page 8-153
OR on page 8-218

UnitData on page 8-341

9

Macro Reference List

9-1 Error List 9-2

9-2 List of Reserved Words 9-5
- 9-2-1 List of Reserved Words 9-5
- 9-2-2 List of Reserved Global Data Words 9-9

9-3 System Data List 9-10

9-4 List of I/O Modules 9-27
- 9-4-1 EtherCAT 9-28
- 9-4-2 EtherNetIP 9-31
- 9-4-3 PROFINET 9-34
- 9-4-4 ParallelIo 9-37
- 9-4-5 SerialNormal 9-39
- 9-4-6 SerialPlcLinkM 9-41
- 9-4-7 SerialPlcLink 9-43
- 9-4-8 TcpClient 9-46
- 9-4-9 TcpNormal 9-48
- 9-4-10 UdpNormal 9-50
- 9-4-11 UdpPlcLinkM 9-52
- 9-4-12 UdpPlcLinkY 9-55
- 9-4-13 UdpPlcLink 9-58
- 9-4-14 TcpPlcLinkM 9-61
- 9-4-15 TcpPlcLink 9-64

9-5 List for Processing Item Identifier 9-67
- 9-5-1 Measurement 9-67
- 9-5-2 Input image 9-67
- 9-5-3 Compensate image 9-68
- 9-5-4 Support measurement 9-68
- 9-5-5 Branch 9-69
- 9-5-6 Output result 9-69
- 9-5-7 Display result 9-69

9-6 Figure Data List 9-70
- 9-6-1 Figure Data Structure List 9-70
- 9-6-2 List of Figure Data Array Elements 9-71

9-7 List of Figure Numbers 9-73

9-8 Model Number List 9-76

9-9 Image Number List 9-78

9-10 List of Sub-Image Numbers 9-82

9-1 Error List

If an error occurs during execution of the program of a macro customize function, you can identify the cause of the error from the error number (ErrNo) and error message that are displayed. The error messages that may appear are described below.

Precautions for Correct Use

When an error occurs in a process other than a communication command macro or Try - Catch - End Try, the error number cannot be used to check the error.
Determine the nature of the error from the error message that appears in the system status console window.

Additional Information

Error information appears in the system status console window.

No.	Error Message	Explanation	Action
1	NEXT without FOR	A Next statement occurs without a corresponding For statement.	Write For properly in a For - Next statement.
2	Syntax error	Incorrect function format or spelling, or a function is not used in accordance with the rules.	Check the macro function reference and correct the function format or spelling.
3	RETURN without GOSUB	A Return statement occurs without a corresponding Gosub statement.	A Gosub function does not exist or is not written correctly. Write the Gosub function correctly.
5	Illegal function call	The argument specified for a function is not within the allowed range, or an error occurred when the function was executed.	Occurs when a macro function argument is not within the allowed range. Check the macro function reference and set the macro function argument appropriately. If the macro function is executed from an inappropriate subroutine, execute from an appropriate subroutine.
6	Overflow	The calculation result or entered value is not within the allowed range for real numbers, or the character string length is over the allowed range.	Occurs when a real number value is not within the range -1.0E30 to 1.0E30. Correct the process so that the value is within the real number range. Occurs when the length of a character string variable or constant is over 255 characters. Shorten the character string length.
7	Out of memory	Insufficient sensor controller working memory, or a loop process is nested too deep.	Occurs when too many character strings or array variables are used in a program. Decrease the number used until the error no longer occurs. Occurs when the number of nested levels in a loop process is too large. Decrease the number of nested levels until the error no longer occurs.

No.	Error Message	Explanation	Action
8	Undefined line number	The program branches to an undefined line number.	Occurs when a Goto or other function branches to a non-existent line number. Specify an existing line number, or use a <label> instead of a line number to specify the branch destination.
9	Subscript out of range	An attempt was made to use an arrayelement with an index over the declared maximum.	Occurs when an attempt is made to access an array variable over the number of array variable elements declared in the Dim instruction. Change the element number to a number equal to or less than the number of elements, or change the number of declared elements.
11	Division by Zero	Division by zero was attempted.	Division by 0 is not possible. Divide by a value other than 0.
13	Type mismatch	Variable types do not match, such as on the left or right side of an expression, or in the argument of a function.	Occur when variable types do not match on the left or right side of an expression, or in the argument of a function. Specify the correct variable type or array variable. Occurs when a character string is assigned to a numerical value, or a numerical value is assigned to a character string. Assign a numerical value to a numerical value, and a character string to a character string.
15	String too long	The number of characters in a character string assigned to a character string variable is over 255.	If the character string variable is over 255 characters, break the string into two variables.
18	Undefined array	An undefined array is used.	Declare the array variable with a Dim instruction before using the array.
23	Line buffer overflow	An attempt was made to input more than the allowed character limit (255 bytes) on one line.	Generally occurs when one line of data is received by serial communication or from a memory card. Use the Input$ function to input the required number of bytes.
26	FOR without NEXT	A For statement occurs without a corresponding Next statement.	Occurs when a Next instruction does not exist or is not written correctly in a For - Next statement. Write the Next instruction correctly.
32	Undefined label	An attempt was made to access an undefined label.	Occurs when the referencing and referenced label names do not match. Use a defined label name.
121	CASE without SELECT	A Case statement occurs without a corresponding Select statement.	Occurs when a Select instruction does not exist or is not written correctly in a Select - Case statement. Write the Select instruction correctly.
122	END SELECT without SELECT	An End Select statement occurs without a corresponding Select statement.	
123	SELECT without END SELECT	A Select statement occurs without a corresponding End Select statement	Occurs when an End Select instruction does not exist or is not written correctly in a Select - Case statement. Write the End - Select instruction correctly.
124	CASE without END SELECT	A Case statement occurs without a corresponding End Select statement.	

No.	Error Message	Explanation	Action
125	ELSEIF without IF	An ElseIf statement occurs without a corresponding If statement.	Occurs when an If instruction does not exist or is not written correctly in an If Else statement. Write the If instruction correctly.
126	ELSE without IF	Else statement occurs without a corresponding If statement.	
127	ENDIF without IF	Endif statement occurs without a corresponding If statement.	Occurs when an If instruction does not exist or is not written correctly in an If - EndIf statement. Write the If instruction correctly.
128	IF without ENDIF	If statement occurs without a corresponding EndIf statement.	Occurs when an EndIf instruction does not exist or is not written correctly in an If - EndIf statement. Write the EndIf instruction correctly.
129	ELSEIF without ENDIF	ElseIf statement occurs without a corresponding EndIf statement.	
130	ELSE without ENDIF	Else statement occurs without a corresponding Endif statement.	
135	DO without LOOP	Do statement occurs without a corresponding Loop statement.	Occurs when an Loop instruction does not exist or is not written correctly in a Do Loop While statement. Write the Loop instruction correctly.
136	LOOP without DO	Loop statement occurs without a corresponding Do statement.	Occurs when an Do instruction does not exist or is not written correctly in a Do Loop While statement. Write the Do instruction correctly.
140	EXIT without FOR	An Exit Do statement occurs without acorresponding For statement.	Occurs when a For instruction does not exist or is not written correctly in a For Next statement. Write the For instruction correctly.
141	EXIT without DO	Exit Do statement occurs without a corresponding Do statement.	Occurs when an Do instruction does not exist or is not written correctly in a Do Loop While statement. Write the Do instruction correctly.

9-2 List of Reserved Words

Macro functions, operators, and other character strings predefined in the system are referred to as "reserved words". Do not use the reserved words for Macro customize function.

9-2-1 List of Reserved Words

Reserved words that are defined in the FH series are shown below.

A

Abs	AbsolutePath$	Active&
ActiveTable&	AddGlobalData	AddSystemData
And	Append	ApplicationPath$
ApplicationVersion$	ApproximationCircle	ArgmentCountMax&
ArgmentsLength&	ArgmentsLine$	ArgmentString$
ArgmentValue#	ArgumentsLength&	ArgumentString$
ArgumentValue#	ArrayDims	ArrayLen
As	Asc	AssignUnit
Atn		

B

BusyOut	BITPATTERN&	

C

Calculate?*1	Call	Case
Catch	ChangeScene	ChangeSceneGroup
CheckUnit	Chr$	Clear
ClearMeasureData	ClearScene	ClearSceneGroup
Close	CloseTextData	Cls
CMD$	CmdNo&	CommandArea&
CommandCode&	CommandCountMax&	CommandData&
CommandDelimiter$	CommandDelimiter1$	CommandExecute&
CommandLine$	CommandMemoryAddress&	CommandResponse&
CommandString$	COMMENTSTRING$	Cont
ContinuousMeasure	CopyMeasureImage	CopyScene
CopySceneGroup	CopyUnit	CopyUnitFigure
CopyUnitImage	CopyUnitModel	Cos
CR	CR$	CR_LF$
CreateDialogData	Crspoint	

*1. "?" represents numbers.

D

DATACOUNT&	Date$	DBL_MAX
DBL_MIN	Debug	DebugPrint
Delete	DeleteDialogData	DeleteProfileData
DeleteUnit	DelimiterCount&	Delimiter1Count&

Dim	DisableUnit	DisplaySubNo
DISPLAYTEXT$	DisplayUnitNo	DISPNAMESTRING$
Do	Dposline	DrawArc
DrawArcW	DrawBox	DrawCircle
DrawCircleW	DrawCursor	DrawEllipse
DrawFigure	DrawFillImage	DrawJudgeText
DrawLine	DrawLineW	DrawMeasureImage
DrawPoint	DrawPolygon	DrawSearchFigure
DrawText	DrawTextG	DrawUnitImage
Dskf		

E

ElapsedTime	Else	Elseif
End	EndIf	Eof
Erase	Erl	Err
ERRCMND$	Errno	Error
ErrorOut	Executable	ExecuteErrorProc
ExecuteImageLogging	Exit	ExitFzProcess
Exp	Explicit	

F

False	Fcopy	FieldbusCommand
FigureType	Files	Fix
FONTSIZE_NORMAL	FONTSTYLE_BOLD	FONTSTYLE_ITALIC
FONTSTYLE_NORMAL	FONTSTYLE_STRIKEOUT	FONTSTYLE_UNDERLINE
For	FunctionCall	

G

GetAll	GetCommandNo	GetDelimiter1Count
GetDelimiterCount	GetGlobalData	GetDialogData
GetImageSize	GetImageWindow	GetMeasureOut
GetPlcData	GetPollingState	GetPort
GetProfileData	GetSceneData	GetSceneGroupData
GetSystemData	GetText$	GetTextWindow
GetUnitData	GetUnitFigure	Global
Gosub	Goto	

H

Hex$	HighBytes&	

I

If	ImageFormat	ImageUpdate
Include	Initialize	InitSetting
Input	Input$	InsertUnit
Int	INT_MAX	INT_MIN
IoIdent$	IsFile	IsEndFlg&
ItemCount	ItemIdent$	ItemInfo

ItemTitle$		

J

JGINDEX&	JUDGE&	JUDGE_ACTIVATEERROR
JUDGE_ERROR	JUDGE_IMAGEERROR	JUDGE_MEMORYERROR
JUDGE_MODELERROR	JUDGE_NC	JUDGE_NG
JUDGE_OK	JUDGELOWER#	JUDGEMACROFLAG&
JudgeOut	JUDGEUPPER#	

K

Keyword	Kill	

L

LanguageString$	LCase$	Left$
Len	LF	LF$
Line	List	Load
LoadBackupData	LoadScene	LoadSceneGroup
LoadSystemData	LoadUnitData	Local
Log	Loop	LowBytes&
Lsqumeth		

M

M_PI	MacroCall	MacroClose
MacroDirect	MacroGosub	MacroOpen
MCRINIT	Measure	MeasureDispG
MeasureDispI	MeasureDispT	MeasureId$
MeasureInit	MeasureOut	MeasureProc
MeasureStart	MeasureStop	MeasureStopTable&
MemoryInfo	MessageBox	Mid$
Mkdir	Mod	MoveUnit

N

New	Next	Not

O

On	Open	OpenTextData
Option	Or	Output

P

ParallelCommand	ParallelExecute	ParaOffset&
Piece$	PlcLinkCommand	PlcRcvData&
PlcRcvStrData$	PlcSndData&	Print
PS_DASH	PS_DASHDOT	PS_DASHDOTDOT
PS_DOT	PS_INSIDEFRAME	PS_NULL
PS_SOLID	PutAll	PutPort

R

RaiseErrorProcEvent	RaiseOptionEvent	ReadPlcMemory
ReceiveData	ReDim	RefreshImageWindow
RefreshJudgeWindow	RefreshTextWindow	RefreshTimeWindow
Rem	Remeasure	RENUMPROC
RenumUnitNo	ResponseArea&	ResponseCode&
ResponseMemoryAddress&	ResponseString$	ResponseValue&
RESULTDATA#	RESULTJUDGE&	Return
RGB	Right$	Rmdir
Run	RunOut	

S

Save	SaveBackupData	SaveData
SaveImage	SaveMeasureImage	SaveScene
SaveSceneGroup	SaveSystemData	SaveUnitData
SceneCount	SceneDescription$	SceneGroupCount
SceneGroupNo	SceneGroupTitle$	SceneMaker$
SceneNo	SceneTitle$	ScreenCapture
Select	SendData	SendString
SetArgument	SetDialogData	SetDisplayUnitNo
SetDrawStyle	SetForegroundLine	SetGlobalData
SetImageWindow	SETJUDGE?*1	SETJUDGESHARED
SetJudgeWindow	SetMeasureImage	SetMeasureOut
SetPlcData	SetPollingState	SetProfileData
SetSceneData	SetSceneDescription	SetSceneGroupData
SetSceneGroupTitle	SetSceneMaker	SetSceneTitle
SetStop	SetSystemData	SetTextStyle
SetTextWindow	SetTimeWindow	SetUnitData
SetUnitFigure	SetUnitJudge	SetUnitTitle
SetupDisp	SetUserSubroutine	SetVar
Shell	Sin	Sqr
StartTimer	Step	Stop
Str$	Str2$	StrCount&
StringCommand	Stupproc	Sub
SubArgmentsLine$	SubList	SystemInfo
SystemReset		

*1. "?" represents numbers.

T

TA_BASELINE	TA_BOTTOM	TA_CENTER
TA_LEFT	TA_NOUPDATECP	TA_RIGHT
TA_RTLREADING	TA_TOP	TA_UPDATECP
TAB	TAB$	Tan
Task	TEMPLATEVERSION&	TestMeasure
Then	Time$	Timer
TJGFLAG&	TmpSubArgmentsLine$	TMPSTR$
TMPSTROLD$	To	TotalJudge

TransformAngle	TransformArea	TransformDist
TransformLine	TransformXY	True
Try		

U

UCase$	UnitCount	UnitData
UnitData$	UnitData2	UnitInfo
UnitItemIdent$	UnitJudge	UnitNo
UnitTitle$	UserDialog	Ut

V

Val	VarList	VarPop
VarPtr	VarPush	VarSave

W

Wait	While	WritePlcMemory

X

Xor		

9-2-2 List of Reserved Global Data Words

Global data that is reserved in the FH series is shown below.

Precautions for Correct Use

Global data cannot be obtained from macros when the sensor controller is startup.

ControlFlowParallelCommand_status	editScnGroupNo	editScnNo
editState	mageWindowOrigin_IoCommand	ImageWindowSubNoMax_IoCommand
LayoutNoLocal_IoCommand	LayoutNoRemote_IoCommand	LineNoLocal_IoCommand
LineNoRemote_IoCommand	ParallelDIOffset	RemoteOperationStatus_IoCommand

9-3 System Data List

The ID information and ID names required to set or acquire system data are indicated below.

Precautions for Correct Use

Do not edit settings data not listed in this manual. Doing so could cause the system to not operate correctly.

Identifier information 0	Identifier information 1	Data
Configuration (General)	language	Language Sets the display language. chs: Simplified Chinese, cht: Traditional Chinese, deu: German, eng: English, esp: Spanish, fra: French, ita: Italian, jpn: Japanese, kor: Korean, pol: Polish, vie: Vietnamese
	initialSceneGroupNo	Initial scene group number
	initialSceneNo	Initial scene number
	initialMeasureOut	External output setting 0: Not output externally 1: Output externally
	operationPriority	Operation priority 0: Measurement result priority 1: Menu operation priority
	measureInitPriority	Measurement initialization priority 0: Measurement trigger receipt priority 1: Processing of re-drawing on screen priority
	measureProcPriority	Measurement process priority 0: Performing measurement processing with highest priority 1: Lowering priority for measurement processing
	operatingMode	Operation Mode 0: Standard (Operation Mode) 1: Double Speed Multi-input 2: Multi-line Random-trigger 3: Non-stop Adjustment
	parallelExecute	Parallel execution 0: OFF 1: ON
	sceneCount	Number of scenes
	sceneGroupCount	Number of scene groups
	unitDataAllocator	Memory management setting 0: Usable memory maximum size set, free memory reallocated 1: Usable memory maximum size set, free memory not reallocated 2: Usable memory maximum size not set, free memory not re-allocated
	imageDataAllocator	

Identifier information 0	Identifier information 1	Data
ErrorProc (Error process)	errorKind00000	Error operation settings when connected camera is changed Set with the string "n1,n2". • n1: Error output 0: Error output disable 1: Error output enable • n2: How to display the Error output on the UI. 0: Don't display the error. 1: Informs the error contents on the information display window. Note that displays the error dialog except Logging error and PLC Link communication error. 2: Displays the error dialog. 3: Displays the error dialog and clears the error at close the dialog.
	errorKind00001	Error handling for when a system malfunction (Fan / Voltage error) occurs Set with the string "n1,n2". The setting is same as the data value of errorKind00000. Refer to data of errorKind00000.
	errorKind00003	Error handling for when a battery error occurs Set with the string "n1,n2". The setting is same as the data value of errorKind00000. Refer to data of errorKind00000.
	errorKind00004	Error handling for when a UWF overlay depletion occurs Set with the string "n1,n2". The setting is same as the data value of errorKind00000. Refer to data of errorKind00000.
	errorKind00010	Error handling for when a camera connection error occurs Set with the string "n1,n2". The setting is same as the data value of errorKind00000. Refer to data of errorKind00000.
	errorKind00011	Error handling for when a system malfunction occurs Set with the string "n1,n2". The setting is same as the data value of errorKind00000. Refer to data of errorKind00000.
	errorKind00012	Error handling for when a Camera over-current condition is detected Set with the string "n1,n2". The setting is same as the data value of errorKind00000. Refer to data of errorKind00000.
	errorKind00013	Error handling for when a lighting connection, or configuration error occurs Set with the string "n1,n2". The setting is same as the data value of errorKind00000. Refer to data of errorKind00000.
	errorKind00020	Error operation setting for when image logging disk write error occurs Set with the string "n1,n2". The setting is same as the data value of errorKind00000. Refer to data of errorKind00000.

Identifier information 0	Identifier information 1	Data
	errorKind00030	Error operation setting for parallel output time-out Set with the string "n1,n2". The setting is same as the data value of errorKind00000. Refer to data of errorKind00000.
	errorKind00031	Error operation setting when the PLC Link communication error occurs Set with the string "n1,n2". The setting is same as the data value of errorKind00000. Refer to data of errorKind00000.
	errorKind00032	Error handling for when a Parallel I/O over-current condition is detected Set with the string "n1,n2". The setting is same as the data value of errorKind00000. Refer to data of errorKind00000.
	errorKind00040	Error operation setting for data load error Set with the string "n1,n2". The setting is same as the data value of errorKind00000. Refer to data of errorKind00000.
	errorKind00041	Error operation setting for data transfer error Set with the string "n1,n2". The setting is same as the data value of errorKind00000. Refer to data of errorKind00000.
	errorKind00042	Error operation setting for invalid Initial Scene group number error Set with the string "n1,n2". The setting is same as the data value of errorKind00000. Refer to data of errorKind00000.
	errorKind00043	Error operation setting for invalid Initial Scene number error Set with the string "n1,n2". The setting is same as the data value of errorKind00000. Refer to data of errorKind00000.
IoModule (Communication module)	ioIdent2	Serial (Ethernet) UdpNormal: Normal (UDP) TcpNormal: Normal (TCP) TcpClient: Normal (TCP Client) UdpNormal2: Normal (UDP) (Fxxx series method) UdpPlcLink: PLC Link (SYSMAC CS/CJ/CP/One)(UDP) TcpPlcLink: PLC Link (SYSMAC CS/CJ/CP/One)(TCP) UdpPlcLinkM: PLC Link (MELSEC QnU/Q/QnAS)(UDP) TcpPlcLinkM: PLC Link (MELSEC QnU/Q/QnAS)(TCP) UdpPlcLinkY: PLC Link (JEPMC MP)
	ioIdent1	Serial (RS-232C/422) SerialNormal: Normal SerialNormal2: Normal (Fxxx series method) SerialPlcLink: PLC Link (SYSMAC CS/CJ/CP/One) SerialPlcLinkM: PLC Link (MELSEC QnU/Q/QnAS series)
	ioIdent0	Parallel ParallelIo: Standard Parallel I/O

Identifier information 0	Identifier information 1	Data
	ioIdent3	Fieldbus Blank character: None EtherCAT: EtherCAT EtherNet/IP: EtherNet/IP
	ioIdent4	Remote Operation RemoteServer: Yes Blank character: None
MultiLineRandom (Multi line Random trigger mode settings)	lineCount	Number of lines (2 to 8)
	physicalCameraMask0	Camera - line assignment 0
	physicalCameraMask1	Camera - line assignment 1
	physicalCameraMask2	Camera - line assignment 2
	physicalCameraMask3	Camera - line assignment 3
	physicalCameraMask4	Camera - line assignment 4
	physicalCameraMask5	Camera - line assignment 5
	physicalCameraMask6	Camera - line assignment 6
	physicalCameraMask7	Camera - line assignment 7
CameraControl (Camera settings)	cameraDelay0	Camera0 exposure start additive time (Ver.6.55 or earlier: STEP - camera delay)
	cameraDelay1	Camera1 exposure start additive time (Ver.6.55 or earlier: STEP - camera delay)
	cameraDelay2	Camera2 exposure start additive time (Ver.6.55 or earlier: STEP - camera delay)
	cameraDelay3	Camera3 exposure start additive time (Ver.6.55 or earlier: STEP - camera delay)
	cameraDelay4	Camera4 exposure start additive time (Ver.6.55 or earlier: STEP - camera delay)
	cameraDelay5	Camera5 exposure start additive time (Ver.6.55 or earlier: STEP - camera delay)
	cameraDelay6	Camera6 exposure start additive time (Ver.6.55 or earlier: STEP - camera delay)
	cameraDelay7	Camera7 exposure start additive time (Ver.6.55 or earlier: STEP - camera delay)
	transferRate0	Camera0 transmission speed 0: Normal 2: Fast
	transferRate1	Camera1 transmission speed 0: Normal 2: Fast
	transferRate2	Camera2 transmission speed 0: Normal 2: Fast
	transferRate3	Camera3 transmission speed 0: Normal 2: Fast
	transferRate4	Camera4 transmission speed 0: Normal 2: Fast
	transferRate5	Camera5 transmission speed 0: Normal 2: Fast

Identifier information 0	Identifier information 1	Data
	transferRate6	Camera6 transmission speed 0: Normal 2: Fast
	transferRate7	Camera7 transmission speed 0: Normal 2: Fast
	timingSignal	Output signal setting 0: STGOUT 1: SHTOUT
	stgoutCamera	SGTOUT output when camera is not connected. 0: STGOUT is not output. 1: STGOUT is output.
	shtoutDelay	SHTOUT signal output additive time (Ver.6.55 or earlier: SHTOUT signal delay)
	shtoutWidth	SHTOUT signal pulse width
	shtoutPolarity	SHTOUT signal pulse polarity 0: Negative polarity 1: Positive polarity
CameraInfo (Camera information)	cameraModel0	Connected camera type
	cameraModel1	
	cameraModel2	
	cameraModel3	
	cameraModel4	
	cameraModel5	
	cameraModel6	
	cameraModel7	
ParallelIo (Parallel)	polarity	Output polarity 0: ON at NG 1: ON at OK
	handshake	Output control 0: OFF 1: Handshaking 2: Synchronization output
	cycleTime	Output cycle
	riseTime	Gate ON delay
	outputTime	Output time
	timeout	Timeout
	delayCount	Number of delay
	orOutMode	One-shot OR signal 0: OFF 1: ON
	orOutputTime	OR signal output time
SerialNormal (Non-protocol)	timeout	Timeout
	rsMode	Interface 0: RS-232C
	baudRate	Baud rate [bps] (2400/4800/9600/19200/38400/57600/115200)
	byteSize	Data length [bit] (7 or 8)

Identifier information 0	Identifier information 1	Data
	parity	Parity 0: OFF 1: Odd 2: Even
	stopBits	Stop bits [bit] 0: 1 1: 2
	softFlow	Flow control 0: OFF 1: ON (Xon/Xoff)
	delimiter	Delimiter 0: CR 1: LF 2: CR + LF
SerialNormal2 (Non-protocol for Fxxx series method)	timeout	Timeout
	rsMode	Interface 0: RS-232C
	baudRate	Baud rate [bps] (2400/4800/9600/19200/38400/57600/115200)
	byteSize	Data length [bit] (7 or 8)
	parity	Parity 0: OFF 1: Odd 2: Even
	stopBits	Stop bits [bit] 0: 1 1: 2
	softFlow	Flow control 0: OFF 1: ON (Xon/Xoff)
	delimiter	Delimiter 0: CR 1: LF 2: CR + LF
SerialPlcLink (RS-232C/422 PLC Link (SYSMAC CS/CJ/CP/One))	handshake	Output control 0: OFF 1: Handshaking
	timeout	Timeout
	rsMode	Interface 0: RS-232C
	baudRate	Baud rate [bps] (2400/4800/9600/19200/38400/57600/115200)
	byteSize	Data length [bit] (7 or 8)
	parity	Parity 0: OFF 1: Odd 2: Even
	stopBits	Stop bits [bit] 0: 1 1: 2

Identifier information 0	Identifier information 1	Data
	softFlow	Flow control 0: OFF 1: ON (Xon/Xoff)
	commandArea	Command Area type
	commandMemoryAddress	Command Area address
	responseArea	Response Area type
	responseMemoryAddress	Response Area address
	outputArea	Data Output Area type
	outputMemoryAddress	Data Output Area address
	outputBuffering	Asynchronous output 0: OFF 1: ON
	responseTimeout	Retry interval
	pollingMinCycle	Polling cycle
SerialPlcLinkM (RS-232C/422 PLC Link (SYSMAC CS/CJ/CP/One))	handshake	Output control 0: OFF 1: Handshaking
	timeout	Timeout
	rsMode	Interface 0: RS-232C
	baudRate	Baud rate [bps] (2400/4800/9600/19200/38400/57600/115200)
	byteSize	Data length [bit] (7 or 8)
	parity	Parity 0: OFF 1: Odd 2: Even
	stopBits	Stop bits [bit] 0: 1 1: 2
	softFlow	Flow control 0: OFF 1: ON (Xon/Xoff)
	commandArea	Command Area type
	commandMemoryAddress	Command Area address
	responseArea	Response Area type
	responseMemoryAddress	Response Area address
	outputArea	Data Output Area type
	outputMemoryAddress	Data Output Area address
	outputBuffering	Asynchronous output 0: OFF 1: ON
	responseTimeout	Retry interval
	pollingMinCycle	Polling cycle
UdpPlcLink (Ethernet PLC Link (SYSMAC CS/CJ/CP/One) (UDP))	handshake	Output control 0: OFF 1: Handshaking
	commandArea	Command Area type
	commandMemoryAddress	Command Area address

Identifier information 0	Identifier information 1	Data
	responseArea	Response Area type
	responseMemoryAddress	Response Area address
	outputArea	Data Output Area type
	outputMemoryAddress	Data Output Area address
	outputBuffering	Asynchronous output 0: OFF 1: ON
	responseTimeout	Retry interval
	responseTimeout2	Retry interval2
	pollingMinCycle	Polling cycle
	enableDhcp	Automatic 0: OFF 1: ON
	ipAddress	IP address
	subnetMask	Subnet mask
	defaultGateway	Default gateway
	enableDhcp2	Automatic 2 0: OFF 1: ON
	ipAddress2	IP address2
	subnetMask2	Subnet mask2
	defaultGateway2	Default gateway2
	dns	DNS server
	dns2	DNS server2
	wins	WINS server
	wins2	WINS server2
	destIpAddress	Output IP address
	portNo	Port number
UdpPlcLinkM (Ethernet PLC Link (MELSEC QnU/Q/QnAS)(UDP))	handshake	Output control 0: OFF 1: Handshaking
	commandArea	Command Area type
	commandMemoryAddress	Command Area address
	responseArea	Response Area type
	responseMemoryAddress	Response Area address
	outputArea	Data Output Area type
	outputMemoryAddress	Data Output Area address
	outputBuffering	Asynchronous output 0: OFF 1: ON
	responseTimeout	Retry interval
	responseTimeout2	Retry interval2
	pollingMinCycle	Polling cycle
	enableDhcp	Automatic 0: OFF 1: ON
	ipAddress	IP address
	subnetMask	Subnet mask

Identifier information 0	Identifier information 1	Data
	defaultGateway	Default gateway
	enableDhcp2	Automatic 2 0: OFF 1: ON
	ipAddress2	IP address2
	subnetMask2	Subnet mask2
	defaultGateway2	Default gateway2
	dns	DNS server
	dns2	DNS server2
	wins	WINS server
	wins2	WINS server2
	destIpAddress	Output IP address
	portNo	Port number
TcpPlcLink (Ethernet PLC Link (SYSMAC CS/CJ/CP/One) (TCP)	handshake	Output control 0: OFF 1: Handshaking
	commandArea	Command Area type
	commandMemoryAddress	Command Area address
	responseArea	Response Area type
	responseMemoryAddress	Response Area address
	outputArea	Data Output Area type
	outputMemoryAddress	Data Output Area address
	outputBuffering	Asynchronous output 0: OFF 1: ON
	responseTimeout	Retry interval
	responseTimeout2	Retry interval2
	pollingMinCycle	Polling cycle
	enableDhcp	Automatic 0: OFF 1: ON
	ipAddress	IP address
	subnetMask	Subnet mask
	defaultGateway	Default gateway
	enableDhcp2	Automatic 2 0: OFF 1: ON
	ipAddress2	IP address2
	subnetMask2	Subnet mask2
	defaultGateway2	Default gateway2
	dns	DNS server
	dns2	DNS server2
	wins	WINS server
	wins2	WINS server2
	serverIpAddress	Server IP address
	portNo	Port number

Identifier information 0	Identifier information 1	Data
TcpPlcLinkM (Ethernet PLC Link (MELSEC QnU/Q/QnAS) (TCP))	handshake	Output control 0: OFF 1: Handshaking
	commandArea	Command Area type
	commandMemoryAddress	Command Area address
	responseArea	Response Area type
	responseMemoryAddress	Response Area address
	outputArea	Data Output Area type
	outputMemoryAddress	Data Output Area address
	outputBuffering	Asynchronous output 0: OFF 1: ON
	responseTimeout	Retry interval
	responseTimeout2	Retry interval2
	pollingMinCycle	Polling cycle
	enableDhcp	Automatic 0: OFF 1: ON
	ipAddress	IP address
	subnetMask	Subnet mask
	defaultGateway	Default gateway
	enableDhcp2	Automatic 2 0: OFF 1: ON
	ipAddress2	IP address2
	subnetMask2	Subnet mask2
	defaultGateway2	Default gateway2
	dns	DNS server
	dns2	DNS server2
	wins	WINS server
	wins2	WINS server2
	serverIpAddress	Server IP address
	portNo	Port number
UdpNormal (Non-protocol (UDP))	enableDhcp	Automatic 0: OFF 1: ON
	ipAddress	IP address
	subnetMask	Subnet mask
	defaultGateway	Default gateway
	enableDhcp2	Automatic 2 0: OFF 1: ON
	ipAddress2	IP address2
	subnetMask2	Subnet mask2
	defaultGateway2	Default gateway2
	dns	DNS server
	dns2	DNS server2
	wins	WINS server

Identifier information 0	Identifier information 1	Data
	wins2	WINS server2
	destIpAddress	Output IP address
	portNo	Input port number
	portNo2	Output port number
TcpNormal (Non-protocol (TCP))	enableDhcp	Automatic 0: OFF 1: ON
	ipAddress	IP address
	subnetMask	Subnet mask
	defaultGateway	Default gateway
	enableDhcp2	Automatic 2 0: OFF 1: ON
	ipAddress2	IP address2
	subnetMask2	Subnet mask2
	defaultGateway2	Default gateway2
	dns	DNS server
	dns2	DNS server2
	wins	WINS server
	wins2	WINS server2
	portNo	Port number
TcpClient (Non-protocol (TCP Client))	enableDhcp	Automatic 0: OFF 1: ON
	ipAddress	IP address
	subnetMask	Subnet mask
	defaultGateway	Default gateway
	enableDhcp2	Automatic 2 0: OFF 1: ON
	ipAddress2	IP address2
	subnetMask2	Subnet mask2
	defaultGateway2	Default gateway2
	dns	DNS server
	dns2	DNS server2
	wins	WINS server
	wins2	WINS server2
	portNo	Port number
	serverIpAddress	Server IP address
UdpNormal2 (Non-protocol(UDP Fxxx series method))	enableDhcp	Automatic 0: OFF 1: ON
	ipAddress	IP address
	subnetMask	Subnet mask
	defaultGateway	Default gateway
	enableDhcp2	Automatic 2 0: OFF 1: ON

Identifier information 0	Identifier information 1	Data
	ipAddress2	IP address2
	subnetMask2	Subnet mask2
	defaultGateway2	Default gateway2
	dns	DNS server
	dns2	DNS server2
	wins	WINS server
	wins2	WINS server2
	destIpAddress	Output IP address
	portNo	Input port number
	portNo2	Output port number
UdpPlcLinkY (Ethernet PLC Link (JEPMC MP)	handshake	Output control 0: OFF 1: Handshaking
	commandArea	Command Area type
	commandMemoryAddress	Command Area address
	responseArea	Response Area type
	responseMemoryAddress	Response Area address
	outputArea	Data Output Area type
	outputMemoryAddress	Data Output Area address
	outputBuffering	Asynchronous output 0: OFF 1: ON
	responseTimeout	Retry interval
	responseTimeout2	Retry interval2
	pollingMinCycle	Polling cycle
	enableDhcp	Automatic 0: OFF 1: ON
	ipAddress	IP address
	subnetMask	Subnet mask
	defaultGateway	Default gateway
	enableDhcp2	Automatic 2 0: OFF 1: ON
	ipAddress2	IP address2
	subnetMask2	Subnet mask2
	defaultGateway2	Default gateway2
	dns	DNS server
	dns2	DNS server2
	wins	WINS server
	wins2	WINS server2
	portNo	Input/Output IP address
	destIpAddress	Output IP address
EtherCAT (EtherCAT)	handshake	Output control 0: OFF 1: Handshaking
	cycleTime	Output cycle
	outputTime	Output time

Identifier information 0	Identifier information 1	Data
	timeout	Timeout
	outputDataSize0	Data output number for line 0 (number of data output items for line 0) 32: Result Data Format 0 (8 DINT) 64: Result Data Format 1 (16 DINT) 128: Result Data Format 2 (32 DINT) 256: Result Data Format 3 (64 DINT) 2097152: Result Data Format 4 (4 LREAL) 4194304: Result Data Format 5 (8 LREAL) 8388608: Result Data Format 6 (16 LREAL) 16777216: Result Data Format 7 (32 LREAL) 1572872: Result Data Format 8 (2 DINT + 3 LREAL) 3145744: Result Data Format 9 (4 DINT + 6 LREAL) 6291488: Result Data Format 10 (8 DINT + 12 LREAL) 12582976: Result Data Format 11 (16 DINT + 24 LREAL)
	outputDataSize1	Data output number for line 1 (number of data output items for line 1) The setting is same as the data value of outputDataSize0. Refer to data of outputDataSize0.
	outputDataSize2	Data output number for line 2 (number of data output items for line 2) The setting is same as the data value of outputDataSize0. Refer to data of outputDataSize0.
	outputDataSize3	Data output number for line 3 (number of data output items for line 3) The setting is same as the data value of outputDataSize0. Refer to data of outputDataSize0.
	outputDataSize4	Data output number for line 4 (number of data output items for line 4) The setting is same as the data value of outputDataSize0. Refer to data of outputDataSize0.
	outputDataSize5	Data output number for line 5 (number of data output items for line 5) The setting is same as the data value of outputDataSize0. Refer to data of outputDataSize0.
	outputDataSize6	Data output number for line 6 (number of data output items for line 6) The setting is same as the data value of outputDataSize0. Refer to data of outputDataSize0.
	outputDataSize7	Data output number for line 7 (number of data output items for line 7) The setting is same as the data value of outputDataSize0. Refer to data of outputDataSize0.
	userOutputDataSize0	Line 0 User Output Data Size
	userOutputDataSize1	Line 1 User Output Data Size

Identifier information 0	Identifier information 1	Data
	userOutputDataSize2	Line 2 User Output Data Size
	userOutputDataSize3	Line 3 User Output Data Size
	userOutputDataSize4	Line 4 User Output Data Size
	userOutputDataSize5	Line 5 User Output Data Size
	userOutputDataSize6	Line 6 User Output Data Size
	userOutputDataSize7	Line 7 User Output Data Size
	userInputDataSize0	Line 0 User Input Data Size
	userInputDataSize1	Line 1 User Input Data Size
	userInputDataSize2	Line 2 User Input Data Size
	userInputDataSize3	Line 3 User Input Data Size
	userInputDataSize4	Line 4 User Input Data Size
	userInputDataSize5	Line 5 User Input Data Size
	userInputDataSize6	Line 6 User Input Data Size
	userInputDataSize7	Line 7 User Input Data Size
EtherNet/IP (EtherNetIP)	handshake	Output control 0: OFF 1: Handshaking
	cycleTime	Output cycle
	outputTime	Output time
	timeout	Timeout
	outputDataSize	Output data size
	userOutputDataSize	User data area output data size 0: [OFF] 32: ON
	userInputDataSize	User data area input data size 0: [OFF] 32: ON
Measure (Measurement)	fanControl	Fan control setting 0: Slow rotation 1: Fast rotation
	captureDirectory	Destination folder of Image Capture
	stepError	STEP in measure 0: ERROR signal output OFF for STEP input during measurement 1: ERROR signal output ON for STEP input during measurement
	sceneGroupSave	Scene group switch 0: Do not save when changing scene group 1: Save when changing scene group
	sceneWaitTime	Scene switch time
Trigger (Other)	digitalFilter	STEP signal filter width 5: 100 11: 200 18: 300 24: 400 30: 500
	encoderEnabled	Use Encoder trigger 0: Not used 1: Used

Identifier information 0	Identifier information 1	Data
	encoderInputPulse	Encoder input selection 0: Open collector 1: Line driver
	encoderResolutionA	Phase A resolution
	encoderZ	Count rotations with phase Z 0: Do not count 1: Count
	encoderStart	Signal active timing 0: ON 1: ON from STEP input
	encoderCount	Trigger input signal and reset timing 0: Trigger signal: Phase A, Reset timing: By trigger 1: Trigger signal: Phase A, Reset timing: Count rotation 2: Phase Z
	encoderTriggerAngleA0	Pulse A 0 to 5 (When the pulse reset timing is "Every rotation (PulseZ)")
	encoderTriggerAngleA1	
	encoderTriggerAngleA2	
	encoderTriggerAngleA3	
	encoderTriggerAngleA4	
	encoderTriggerAngleA5	
	encoderBacklash	Support backlashing (phase B) 0: Do not support 1: Support
	encoderBacklashTrigger	Trigger in backlashing 0: Do not generate 1: Generate
	encoderTriggerCountA	Pulse A (When the pulse reset timing is "Every trigger pulse")
	encoderTriggerCountZ	Pulse Z
NetworkDrive (Network drive setting)	drive0	S
	drive1	T
	drive2	U
	drive3	V
	drive4	W
	drive5	X
	drive6	Y
	drive7	Z
	passwordCrypt0	Network drive encryption password
	passwordCrypt1	
	passwordCrypt2	
	passwordCrypt3	
	passwordCrypt4	
	passwordCrypt5	
	passwordCrypt6	
	passwordCrypt7	

Identifier information 0	Identifier information 1	Data
Logging (Logging setting)	imageLogging	Image Logging 0: None 1: Save for only NG 2: All
	imageLoggingDirectory	Destination Specify the image file save destination (RAMDisk or the external storage, i.e. USB memory).
	imageLoggingHeader	Prefix
	imageLoggingFooter	Suffix
	imageLoggingUseMeasur-eID	Show measurement ID 0: Disabled 1: Enabled
	imageLoggingDistribution-Mode	Manage folders 0: Manual configuration 1: Auto configuration
	imageLoggingDate	Sort by date 0: Disabled 1: Enabled
	imageLoggingScene	Switch saving folder by scene 0: Disabled 1: Enabled 2: Enabled (Use scene name as folder names.)
	imageLoggingJudge	Switch saving folder by judge 0: Disabled 1: Enabled
	imageLoggingSubDirectory	Subfolder name
	imageLoggingType	Number of images for Image logging single: OFF multiple: ON
	imageLoggingFormat	Image file type ifz bfz jfz pfz
	imageLoggingPriority	Logging priority 0: Give priority to logging 1: Give priority to measurement takt
	dataLogging	Data Logging 0: None 1: Save for only NG 2: All
	dataLoggingDirectory	Destination folder of Data Logging

Identifier information 0	Identifier information 1	Data
FtpImageLogging (FTP client image logging setting)	enabled	Save to controller memory + FTP 0: No logging 1: Logging to FTP client
	serverIpAddress	Server IP address
	userName	Username
	password	Password
	passwordCrypt	FTP server encryption password
	portNo	Port No.
	passiveMode	Connection Method 0: Active 1: Passive
	targetDirectory	Folder name
OperationLog (Operation logging setting)	targetDirectory	Save destination
	autoEnabled	Start at launch 0: Don't start 1: Start
PanDA (User definition)	PanDA	-

9-4 List of I/O Modules

I/O module settings required for communication with external devices are indicated below.

Identification name	IO module name	References
EtherCAT	EtherCAT communication	page 9-28
EtherNetIP	EtherNet/IP Interface communication	page 9-31
PROFINET	PROFINET Interface communication	page 9-34
ParallelIo	Parallel Interface communication	page 9-37
SerialNormal SerialNormal2 (Fxxx series method)	Serial Interface Normal communication	page 9-39
SerialPlcLinkM	Serial Interface PLC Link (MELSEC QnU/Q/QnAS) communication	page 9-41
SerialPlcLink	Serial Interface PLC Link (SYSMAC CS/CJ/CP/One) communication	page 9-43
TcpClient	TCP Client Normal communication	page 9-46
TcpNormal	TCP Normal communication	page 9-48
UdpNormal UdpNormal2 (Fxxx series method)	UDP Normal communication	page 9-50
UdpPlcLinkM	PLC Link (MELSEC QnU/Q/QnAS) (UDP)	page 9-52
UdpPlcLinkY	PLC Link (JEPMC MP) communication	page 9-55
UdpPlcLink	PLC Link (SYSMAC CS/CJ/CP/One) (UDP)	page 9-58
TcpPlcLinkM	PLC Link (MELSEC QnU/Q/QnAS) (TCP)	page 9-61
TcpPlcLink	PLC Link(SYSMAC CS/CJ/CP/One) (TCP)	page 9-64

9-4-1 EtherCAT

EtherCAT communication

IoModule identification name:

EtherCAT

Overview:

This is a module for sending and receiving commands and data by EtherCAT protocol.

System data:

Identification name	Meaning	Initial value
handshake	Output control 0: OFF 1: Handshaking	0
cycleTime	Output cycle [EtherCAT communication cycle]	2
outputTime	Output time [EtherCAT communication cycle]	1
timeout	Timeout [0.1s]	100
outputDataSize0 to 7	Output area data formatting setting	32
userOutputDataSize0 to 7	User output formatting setting	0
userInputDataSize0 to 7	User input formatting setting	0

Supported functions:

IoInitialize	OK	-
GetPort	-	-
PutPort	-	-
GetAll	-	-
PutAll	-	-
BusyOut	OK	Sets the output state of the processing busy signal. (Refer to *BusyOut* on page 8-37.)
JudgeOut	OK	Sets the output state of the overall judgement signal.. (Refer to *JudgeOut* on page 8-175.)
RunOut	OK	Sets the output state of the RUN signal. (Refer to *RunOut* on page 8-245.)
ReceiveData	-	-
SendData	OK	Sends data.(Refer to *SendData* on page 8-268.)
SendString	-	-
ReadPlcMemory	OK	Reads a value from the PLC memory area.(Refer to *ReadPlcMemory* on page 8-229.)
WritePlcMemory	OK	Writes values in the PLC memory area.(Refer to *WritePlcMemory* on page 8-363.)
SetPlcData	OK	Creates the data that is written with the WritePlcMemory function.(Refer to *SetPlcData* on page 8-281.)
GetPlcData	OK	Gets data read with the ReadPlcMemory function.(Refer to *GetPlcData* on page 8-140.)
SetPollingState	OK	Sets the execution status of the communication module.(Refer to *SetPollingState* on page 8-283.)
GetPollingState	OK	Gets the polling state of the communication module.(Refer to *GetPollingState* on page 8-142.)

ErrorOut	OK	Set the output status of error ERROR signal. (Refer to *ErrorOut* on page 8-121.)

Example:

Send data using SendData.

Parameter and parameter size for SendData are not required.

Use output data according to the EtherCAT output count setting under System Setting.

Set the <channelOffset> of WritePlcMemory to *8 + (output data size (bytes)) / 2*.

```
Dim DATA&(7)
DATA&(0) = 100
DATA&(1) = 200
DATA&(2) = 300
DATA&(3) = 400
DATA&(4) = 500
DATA&(5) = 600
DATA&(6) = 700
DATA&(7) = 800

'Outputs eight integer type data.
SendData "EtherCAT", data&(), 4*8
```

You can load and write using User Area.

Loading from User Input Area:

```
Dim BUF&(7)
Dim DATA_DINT&(3)
Dim DATA_LREAL#(1)

Rem Loads EtherCAT memory.
ReadPlcMemory "EtherCAT", 0, 10, 16, BUF&()

Rem Gets DINT type data and LREAL type data from buffer.
GetPlcData "EtherCAT", BUF&(), 0, 4, DATA_DINT&(0)
GetPlcData "EtherCAT", BUF&(), 4, 4, DATA_DINT&(1)
GetPlcData "EtherCAT", BUF&(), 8, 4, DATA_DINT&(2)
GetPlcData "EtherCAT", BUF&(), 12, 4, DATA_DINT&(3)
GetPlcData "EtherCAT", BUF&(), 16, 8, DATA_LREAL#(0)
GetPlcData "EtherCAT", BUF&(), 24, 8, DATA_LREAL#(1)
```

Writing to User Output Area:

```
Dim BUF&(7)
Dim DATA_DINT&(3)
Dim DATA_LREAL#(1)

Rem Set you are writing.
DATA_DINT&(0) = 100
DATA_DINT&(1) = 200
DATA_DINT&(2) = 300
DATA_DINT&(3) = 400
DATA_LREAL#(0) = 12.34
DATA_LREAL#(1) = 56.78

Rem Sets DINT type and LREAL type to buffer.
SetPlcData "EtherCAT", BUF&(), 0, 4, DATA_DINT&(0)
SetPlcData "EtherCAT", BUF&(), 4, 4, DATA_DINT&(1)
SetPlcData "EtherCAT", BUF&(), 8, 4, DATA_DINT&(2)
SetPlcData "EtherCAT", BUF&(), 12, 4, DATA_DINT&(3)
SetPlcData "EtherCAT", BUF&(), 16, 8, DATA_LREAL#(0)
SetPlcData "EtherCAT", BUF&(), 24, 8, DATA_LREAL#(1)

Rem Writes EtherCAT memory.
WritePlcMemory "EtherCAT", 0, 24, 16, BUF&()
```

9-4-2 EtherNetIP

EtherNet/IP interface communication

IoModule identification name:

EtherNetIP

Overview:

This is a module for sending and receiving commands and data by Ethernet/IP protocol.

System data:

Identification name	Meaning	Initial value
handshake	Output control 0: OFF 1: Handshaking	0
cycleTime	Output cycle [ms]	100
outputTime	Output time [ms]	50
timeout	Timeout [0.1s]	100

Supported functions:

IoInitialize	OK	-
GetPort	-	-
PutPort	-	-
GetAll	-	-
PutAll	-	-
BusyOut	-	-
JudgeOut	OK	Sets the output state of the overall judgement signal.. (Refer to *Judge-Out* on page 8-175.)
RunOut	OK	Sets the output state of the RUN signal. (Refer to *RunOut* on page 8-245.)
ReceiveData	-	-
SendData	OK	Sends data.(Refer to *SendData* on page 8-268.)
SendString	-	-
ReadPlcMemory	OK	Reads a value from the PLC memory area.(Refer to *ReadPlcMemory* on page 8-229.)
WritePlcMemory	OK	Writes values in the PLC memory area.(Refer to *WritePlcMemory* on page 8-363.)
SetPlcData	OK	Creates the data that is written with the WritePlcMemory function.(Refer to *SetPlcData* on page 8-281.)
GetPlcData	OK	Gets data read with the ReadPlcMemory function.(Refer to *GetPlcData* on page 8-140.)
SetPollingState	OK	Sets the execution status of the communication module.(Refer to *Set-PollingState* on page 8-283.)
GetPollingState	OK	Gets the polling state of the communication module.(Refer to *GetPol-lingState* on page 8-142.)
ErrorOut	OK	Set the output status of error ERROR signal. (Refer to *ErrorOut* on page 8-121.)

Example:

Send data using SendData.

Set an IP address and parameter size (*4 integer type domain) in a parameter to use an Ethernet.

```
Dim DATA&(4)
DATA&(0)=111
DATA&(1)=222
DATA&(2)=333
DATA&(3)=444
DATA&(4)=555

'Outputs five integer type data.
SendData "EtherNetIp", data&(), 4*5
```

Gets 7ch data from the Command area using ReadPlcMemory.

```
Dim DATA&(7)
ReadPlcMemory "EtherNetIP", 0, 0, 16, DATA&()
For I&= 0 To 7
   Print DATA&(I&)
Next
```

Output 4ch data to the Data Output area using WritePlcMemory.

```
Dim BUF&(7)
Dim DATA_DINT&(3)

Rem Set you are writing.
DATA_DINT&(0) = 100
DATA_DINT&(1) = 200
DATA_DINT&(2) = 300
DATA_DINT&(3) = 400

Rem Sets DINT type to buffer.
SetPlcData "EtherNetIP", BUF&(), 0, 4, DATA_DINT&(0)
SetPlcData "EtherNetIP", BUF&(), 4, 4, DATA_DINT&(1)
SetPlcData "EtherNetIP", BUF&(), 8, 4, DATA_DINT&(2)
SetPlcData "EtherNetIP", BUF&(), 12, 4, DATA_DINT&(3)

WritePlcMemory "EtherNetIP", 0, 8, 16, BUF&()
```

You can load and write using User Area.

Loading from User Input Area:

```
Dim BUF&(7)
Dim DATA_DINT&(3)
Dim DATA_LREAL#(1)

Rem Loads EtherNet/IP memory.
ReadPlcMemory "EtherNetIP", 0, 10, 16, BUF&()

Rem Gets DINT type data and LREAL type data from buffer.
GetPlcData "EtherNetIP", BUF&(), 0, 4, DATA_DINT&(0)
GetPlcData "EtherNetIP", BUF&(), 4, 4, DATA_DINT&(1)
GetPlcData "EtherNetIP", BUF&(), 8, 4, DATA_DINT&(2)
GetPlcData "EtherNetIP", BUF&(), 12, 4, DATA_DINT&(3)
GetPlcData "EtherNetIP", BUF&(), 16, 8, DATA_LREAL#(0)
GetPlcData "EtherNetIP", BUF&(), 24, 8, DATA_LREAL#(1)
```

Writing to User Output Area:

```
Dim BUF&(7)
Dim DATA_DINT&(3)
Dim DATA_LREAL#(1)

Rem Set you are writing.
DATA_DINT&(0) = 100
DATA_DINT&(1) = 200
DATA_DINT&(2) = 300
DATA_DINT&(3) = 400
DATA_LREAL#(0) = 12.34
DATA_LREAL#(1) = 56.78

Rem Sets DINT type and LREAL type to buffer.
SetPlcData "EtherNetIP", BUF&(), 0, 4, DATA_DINT&(0)
SetPlcData "EtherNetIP", BUF&(), 4, 4, DATA_DINT&(1)
SetPlcData "EtherNetIP", BUF&(), 8, 4, DATA_DINT&(2)
SetPlcData "EtherNetIP", BUF&(), 12, 4, DATA_DINT&(3)
SetPlcData "EtherNetIP", BUF&(), 16, 8, DATA_LREAL#(0)
SetPlcData "EtherNetIP", BUF&(), 24, 8, DATA_LREAL#(1)

Rem Writes EtherNet/IP memory.
WritePlcMemory "EtherNetIP", 0, 24, 16, BUF&()
```

9-4-3 PROFINET

PROFINET interface communication

IoModule identification name:

PROFINET

Overview:

This is a module for sending and receiving commands and data by PROFINET protocol.

System data:

Identification name	Meaning	Initial value
handshake	Output control 0: OFF 1: Handshaking	0
cycleTime	Output cycle [ms]	100
outputTime	Output time [ms]	50
timeout	Timeout [0.1s]	100

Supported functions:

IoInitialize	OK	-
GetPort	-	-
PutPort	-	-
GetAll	-	-
PutAll	-	-
BusyOut	-	-
JudgeOut	OK	Sets the output state of the overall judgement signal.. (Refer to *JudgeOut* on page 8-175.)
RunOut	OK	Sets the output state of the RUN signal. (Refer to *RunOut* on page 8-245.)
ReceiveData	-	-
SendData	OK	Sends data.(Refer to *SendData* on page 8-268.)
SendString	-	-
ReadPlcMemory	OK	Reads a value from the PLC memory area.(Refer to *ReadPlcMemory* on page 8-229.)
WritePlcMemory	OK	Writes values in the PLC memory area.(Refer to *WritePlcMemory* on page 8-363.)
SetPlcData	OK	Creates the data that is written with the WritePlcMemory function.(Refer to *SetPlcData* on page 8-281.)
GetPlcData	OK	Gets data read with the ReadPlcMemory function.(Refer to *GetPlcData* on page 8-140.)
SetPollingState	OK	Sets the execution status of the communication module.(Refer to *SetPollingState* on page 8-283.)
GetPollingState	OK	Gets the polling state of the communication module.(Refer to *GetPollingState* on page 8-142.)
ErrorOut	OK	Set the output status of error ERROR signal. (Refer to *ErrorOut* on page 8-121.)

Example:

Send data using SendData.

Set an IP address and parameter size (*4 integer type domain) in a parameter to use an Ethernet.

```
Dim DATA&(4)
DATA&(0)=111
DATA&(1)=222
DATA&(2)=333
DATA&(3)=444
DATA&(4)=555

'Outputs five integer type data.
SendData "PROFINET", data&(), 4*5
```

Gets 7ch data from the Command area using ReadPlcMemory.

```
Dim DATA&(7)
ReadPlcMemory "PROFINET", 0, 0, 16, DATA&()
For I&= 0 To 7
   Print DATA&(I&)
Next
```

Output 4ch data to the Data Output area using WritePlcMemory.

```
Dim BUF&(7)
Dim DATA_DINT&(3)

Rem Set you are writing.
DATA_DINT&(0) = 100
DATA_DINT&(1) = 200
DATA_DINT&(2) = 300
DATA_DINT&(3) = 400

Rem Sets DINT type to buffer.
SetPlcData "PROFINET", BUF&(), 0, 4, DATA_DINT&(0)
SetPlcData "PROFINET", BUF&(), 4, 4, DATA_DINT&(1)
SetPlcData "PROFINET", BUF&(), 8, 4, DATA_DINT&(2)
SetPlcData "PROFINET", BUF&(), 12, 4, DATA_DINT&(3)

WritePlcMemory "PROFINET", 0, 8, 16, BUF&()
```

You can load and write using User Area.

Loading from User Input Area:

```
Dim BUF&(7)
Dim DATA_DINT&(3)
Dim DATA_LREAL#(1)

Rem Loads PROFINET memory.
ReadPlcMemory "PROFINET", 0, 10, 16, BUF&()

Rem Gets DINT type data and LREAL type data from buffer.
GetPlcData "PROFINET", BUF&(), 0, 4, DATA_DINT&(0)
GetPlcData "PROFINET", BUF&(), 4, 4, DATA_DINT&(1)
GetPlcData "PROFINET", BUF&(), 8, 4, DATA_DINT&(2)
GetPlcData "PROFINET", BUF&(), 12, 4, DATA_DINT&(3)
GetPlcData "PROFINET", BUF&(), 16, 8, DATA_LREAL#(0)
GetPlcData "PROFINET", BUF&(), 24, 8, DATA_LREAL#(1)
```

Writing to User Output Area:

```
Dim BUF&(7)
Dim DATA_DINT&(3)
Dim DATA_LREAL#(1)

Rem Set you are writing.
DATA_DINT&(0) = 100
DATA_DINT&(1) = 200
DATA_DINT&(2) = 300
DATA_DINT&(3) = 400
DATA_LREAL#(0) = 12.34
DATA_LREAL#(1) = 56.78

Rem Sets DINT type and LREAL type to buffer.
SetPlcData "PROFINET", BUF&(), 0, 4, DATA_DINT&(0)
SetPlcData "PROFINET", BUF&(), 4, 4, DATA_DINT&(1)
SetPlcData "PROFINET", BUF&(), 8, 4, DATA_DINT&(2)
SetPlcData "PROFINET", BUF&(), 12, 4, DATA_DINT&(3)
SetPlcData "PROFINET", BUF&(), 16, 8, DATA_LREAL#(0)
SetPlcData "PROFINET", BUF&(), 24, 8, DATA_LREAL#(1)

Rem Writes PROFINET memory.
WritePlcMemory "PROFINET", 0, 24, 16, BUF&()
```

9-4-4 ParallelIo

Parallel Interface communication

IoModule identification name:

ParallelIo

Overview:

This is a module is for sending and receiving commands and data via the Parallel interface.

System data:

Identification name	Meaning	Initial value
polarity	Output polarity 0: ON at NG 1: ON at OK	0
handshake	Output control 0: OFF 1: Handshaking 2: Synchronization output	0
cycleTime	Output cycle [0.1ms]	100
riseTime	Gate ON delay [0.1ms]	10
outputTime	Output time [0.1ms]	50
timeout	Timeout [0.1ms]	100
delayCount	Number of delay	1
orOutMode	One-shot OR signal 0: OFF 1: ON	0
orOutputTime	Time of one-shot output when OK signal [0.1ms]	50

Supported functions:

IoInitialize	OK	-
GetPort	OK	Gets the input state of the specified input port. (Refer to *GetPort* on page 8-144.)
PutPort	OK	Sets the output state of the specified output terminal. (Refer to *PutPort* on page 8-225.)
GetAll	OK	Gets the input states of all input terminals. (Refer to *GetAll* on page 8-134.)
PutAll	OK	Sets the output state of all output terminals. (Refer to *PutAll* on page 8-223.)
BusyOut	OK	Sets the output state of the processing busy signal. (Refer to *BusyOut* on page 8-37.)
JudgeOut	OK	Sets the output state of the overall judgement signal.. (Refer to *Judge-Out* on page 8-175.)
RunOut	OK	Sets the output state of the RUN signal. (Refer to *RunOut* on page 8-245.)
ReceiveData	-	-
SendData	OK	Sends data.(Refer to *SendData* on page 8-268.)
SendString	-	-
ReadPlcMemory	-	-
WritePlcMemory	-	-
SetPlcData	-	-

GetPlcData	-	-
SetPollingState	OK	Sets the execution status of the communication module.(Refer to *SetPollingState* on page 8-283.)
GetPollingState	OK	Gets the polling state of the communication module.(Refer to *GetPollingState* on page 8-142.)
ErrorOut	OK	Set the output status of error ERROR signal. (Refer to *ErrorOut* on page 8-121.)

Example:

Send data using SendData.

```
Dim data&(256)

'Outputs five integer type data.
SendData "ParallelIo", data&(), 4*5
```

9-4-5 SerialNormal

Serial Interface Normal communication

IoModule identification name:

SerialNormal

SerialNormal2 (Fxxx series method)

Overview:

This is a module is for sending and receiving commands and data via the serial interface.

System data:

Identification name	Meaning	Initial value
rsMode	Interface 0: RS-232C	0
baudRate	Baud rate [bps]	38400
byteSize	Data length [bit] (7 or 8)	8
parity	Parity 0: OFF 1: Odd 2: Even	0
stopBits	Stop bits [bit] 0: 1 1: 2	0
softFlow	Flow control 0: OFF 1: ON (Xon/Xoff)	0
delimiter	Delimiter 0: CR 1: LF 2: CR + LF	0
timeout	Timeout [s]	5

Supported functions:

IoInitialize	OK	-
GetPort	-	-
PutPort	-	-
GetAll	-	-
PutAll	-	-
BusyOut	-	-
JudgeOut	-	-
RunOut	-	-
ReceiveData	OK	Receives data.(Refer to *ReceiveData* on page 8-231.)
SendData	OK	Sends data.(Refer to *SendData* on page 8-268.)
SendString	OK	Sends the character string data.(Refer to *SendString* on page 8-270.)
ReadPlcMemory	-	-
WritePlcMemory	-	-
SetPlcData	-	-
GetPlcData	-	-

SetPollingState	OK	Sets the execution status of the communication module.(Refer to *SetPollingState* on page 8-283.)
GetPollingState	OK	Gets the polling state of the communication module.(Refer to *GetPollingState* on page 8-142.)
ErrorOut	-	-

Example:

Receive data using ReceiveData.

A parameter of ReceiveData, the parameter size are unnecessary.

```
Dim data&(256)

'Gets the five integer type data.
ReceiveData "SerialNormal", data&(), 4*5, size&
```

Send data using SendData.

```
Dim data&(256)

'Outputs five integer type data.
SendData "SerialNormal", data&(), 4*5
```

9-4-6 SerialPlcLinkM

Serial Interface PLC Link (MELSEC QnU/Q/QnAS) communication

IoModule identification name:

SerialPlcLinkM

Overview:

This is a module is for sending and receiving commands and data via the serial PLC Link interface.

System data:

Identification name	Meaning	Initial value
rsMode	Interface 0: RS-232C	0
baudRate	Baud rate [bps]	9600
byteSize	Data length [bit] (7 or 8)	7
parity	Parity 0: OFF 1: Odd 2: Even	2
stopBits	Stop bits [bit] 0: 1 1: 2	1
softFlow	Flow control 0: OFF 1: ON (Xon/Xoff)	0
timeout	Timeout [s]	5

PLC Link data:

Identification name	Meaning	Initial value
commandArea	Command Area type	CIO Area (CIO)
commandMemoryAddress	Command Area address	0
responseArea	Response Area type	CIO Area (CIO)
responseMemoryAddress	Response Area address	100
outputArea	Data Output Area type	CIO Area (CIO)
outputMemoryAddress	Data Output Area address	200
handshake	Output control 0: OFF 1: Handshaking	1
responseTimeout	Retry interval[ms]	10000

Area classification:

Area classification name	Area classification number
Data register	168
File register	175
Link register	180

Supported functions:

IoInitialize	OK	-

GetPort	-	-
PutPort	-	-
GetAll	-	-
PutAll	-	-
BusyOut	-	-
JudgeOut	-	-
RunOut	-	-
ReceiveData	-	-
SendData	OK	Sends data.(Refer to *SendData* on page 8-268.)
SendString	-	-
ReadPlcMemory	OK	Reads a value from the PLC memory area.(Refer to *ReadPlcMemory* on page 8-229.)
WritePlcMemory	OK	Writes values in the PLC memory area.(Refer to *WritePlcMemory* on page 8-363.)
SetPlcData	OK	Creates the data that is written with the WritePlcMemory function.(Refer to *SetPlcData* on page 8-281.)
GetPlcData	OK	Gets data read with the ReadPlcMemory function.(Refer to *GetPlcData* on page 8-140.)
SetPollingState	OK	Sets the execution status of the communication module.(Refer to *SetPollingState* on page 8-283.)
GetPollingState	OK	Gets the polling state of the communication module.(Refer to *GetPollingState* on page 8-142.)
ErrorOut	-	-

Example:

Send data using SendData.

```
Dim data&(256)

'Outputs five integer type data.
SendData "SerialPlcLinkM", data&(), 4*5
```

9-4-7 SerialPlcLink

Serial Interface PLC Link (SYSMAC CS/CJ/CP/One) communication

IoModule identification name:

SerialPlcLink

Overview:

This is a module is for sending and receiving commands and data via the serial PLC Link interface.

System data:

Identification name	Meaning	Initial value
rsMode	Interface 0: RS-232C	0
baudRate	Baud rate [bps]	9600
byteSize	Data length [bit] (7 or 8)	7
parity	Parity 0: OFF 1: Odd 2: Even	2
stopBits	Stop bits [bit] 0: 1 1: 2	1
softFlow	Flow control 0: OFF 1: ON (Xon/Xoff)	0
timeout	Timeout [s]	5

PLC Link data:

Identification name	Meaning	Initial value
commandArea	Command Area type	CIO Area (CIO)
commandMemoryAddress	Command Area address	0
responseArea	Response Area type	CIO Area (CIO)
responseMemoryAddress	Response Area address	100
outputArea	Data Output Area type	CIO Area (CIO)
outputMemoryAddress	Data Output Area address	200
handshake	Output control 0: OFF 1: Handshaking	1
responseTimeout	Retry interval[ms]	10000

Area classification:

Area classification name	Area classification number
CIO Area (CIO)	176
Work Area (WR)	177
Holding Bit Area (HR)	178
Auxiliary Bit Area (AR)	179
DM Area (DM)	130
EM Area (EM0)	160
EM Area (EM1)	161

Area classification name	Area classification number
EM Area (EM2)	162
EM Area (EM3)	163
EM Area (EM4)	164
EM Area (EM5)	165
EM Area (EM6)	166
EM Area (EM7)	167
EM Area (EM8)	168
EM Area (EM9)	169
EM Area (EMA)	170
EM Area (EMB)	171
EM Area (EMC)	172

Supported functions:

IoInitialize	OK	-
GetPort	-	-
PutPort	-	-
GetAll	-	-
PutAll	-	-
BusyOut	-	-
JudgeOut	-	-
RunOut	-	-
ReceiveData	-	-
SendData	OK	Sends data.(Refer to *SendData* on page 8-268.)
SendString	-	-
ReadPlcMemory	OK	Reads a value from the PLC memory area.(Refer to *ReadPlcMemory* on page 8-229.)
WritePlcMemory	OK	Writes values in the PLC memory area.(Refer to *WritePlcMemory* on page 8-363.)
SetPlcData	OK	Creates the data that is written with the WritePlcMemory function.(Refer to *SetPlcData* on page 8-281.)
GetPlcData	OK	Gets data read with the ReadPlcMemory function.(Refer to *GetPlcData* on page 8-140.)
SetPollingState	OK	Sets the execution status of the communication module.(Refer to *SetPollingState* on page 8-283.)
GetPollingState	OK	Gets the polling state of the communication module.(Refer to *GetPollingState* on page 8-142.)
ErrorOut	-	-

Example:

Send data using SendData.

```
Dim data&(256)

'Outputs five integer type data.
SendData "SerialPlcLink", data&(), 4*5
```

Gets 7ch data from 10ch of the DM area using ReadPlcMemory.
Gets the data from readData() using GetPlcData.

```
Dim readData&(256)
Dim data3$(21)

'Gets the data from PLC.
ReadPlcMemory "SerialPlcLink", 130, 10, 7, readData&()

'Gets the data of the real number type.
GetPlcData "SerialPlcLink", readData&(), 0, 8, data1#
'Gets the data of the integer type.
GetPlcData "SerialPlcLink", readData&(), 8, 4, data2&
'Gets the data of the character type.
GetPlcData "SerialPlcLink", readData&(), 12, 5, data2&
```

Write in data for 7ch from 10ch of the DM area using WritePlcMemory.
Set data to writeData() using SetPlcData.

```
Dim writeData&(256)

'Set the writeData (123.45) of the real number type.
SetPlcData "SerialPlcLink", writeData&(), 0, 8, 123.45
Set the writeData (20) of the integer type.
SetPlcData "SerialPlcLink", writeData&(), 32, 4, 20
'Set the writeData (OMRON) of the character type.
SetPlcData "SerialPlcLink", writeData&(), 36, 5, "OMRON"

'Write in data for 7ch from 10ch of the DM area.
WritePlcMemory "SerialPlcLink", 130, 10, 7, writeData&()
```

9-4-8 TcpClient

TCP Client Normal communication

IoModule identification name:

TcpClient

Overview:

This is a module is for sending and receiving commands and data by Ethernet TCP Client protocol.

System data:

Identification name	Meaning	Initial value
enableDhcp	Enable DHCP 0: Disabled (Use the next IP Address.) 1: Enabled (Acquires the IP Address automatically.)	0
ipAddress	IP address	10.5.5.100
subnetMask	Subnet mask	255.255.255.0
defaultGateway	Default gateway	10.5.5.110
dns	DNS server	10.5.5.1
serverIpAddress	Server IP address	10.5.5.101
portNo	Port number to receive commands	9600

Supported functions:

IoInitialize	OK	-
GetPort	-	-
PutPort	-	-
GetAll	-	-
PutAll	-	-
BusyOut	-	-
JudgeOut	-	-
RunOut	-	-
ReceiveData	OK	Receives data.(Refer to *ReceiveData* on page 8-231.)
SendData	OK	Sends data.(Refer to *SendData* on page 8-268.)
SendString	OK	Sends the character string data.(Refer to *SendString* on page 8-270.)
ReadPlcMemory	-	-
WritePlcMemory	-	-
SetPlcData	-	-
GetPlcData	-	-
SetPollingState	OK	Sets the execution status of the communication module.(Refer to *SetPollingState* on page 8-283.)
GetPollingState	OK	Gets the polling state of the communication module.(Refer to *GetPollingState* on page 8-142.)
ErrorOut	-	-

Example:

Receive data using ReceiveData.

Set an IP address and parameter size (*4 integer type domain) in a parameter to use an Ethernet.

```
Dim data&(256)
Dim ipaddr&(4)

'Set the IP address of the destination.
ipaddr&(0) = 10
ipaddr&(1) = 1
ipaddr&(2) = 1
ipaddr&(3) = 101

'Gets the five integer type data.
ReceiveData "TcpClient", data&(), 4*5, size&, ipaddr&(), 4*4
```

Send data using SendData.
Set an IP address and parameter size (*4 integer type domain) in a parameter to use an Ethernet.

```
Dim data&(256)
Dim ipaddr&(4)

'Set the IP address of the destination.
ipaddr&(0) = 10
ipaddr&(1) = 1
ipaddr&(2) = 1
ipaddr&(3) = 101

'Outputs five integer type data.
SendData "TcpClient", data&(), 4*5, ipaddr&(), 4*4
```

9-4-9 TcpNormal

TCP Normal communication

IoModule identification name:

TcpNormal

Overview:

This is a module is for sending and receiving commands and data by Ethernet TCP server.

System data:

Identification name	Meaning	Initial value
enableDhcp	Enable DHCP 0: Disabled (Use the next IP Address.) 1: Enabled (Acquires the IP Address automatically.)	0
ipAddress	IP address	10.5.5.100
subnetMask	Subnet mask	255.255.255.0
defaultGateway	Default gateway	10.5.5.110
dns	DNS server	10.5.5.1
portNo	Port number to receive commands	9600

Supported functions:

IoInitialize	OK	-
GetPort	-	-
PutPort	-	-
GetAll	-	-
PutAll	-	-
BusyOut	-	-
JudgeOut	-	-
RunOut	-	-
ReceiveData	OK	Receives data.(Refer to *ReceiveData* on page 8-231.)
SendData	OK	Sends data.(Refer to *SendData* on page 8-268.)
SendString	OK	Sends the character string data.(Refer to *SendString* on page 8-270.)
ReadPlcMemory	-	-
WritePlcMemory	-	-
SetPlcData	-	-
GetPlcData	-	-
SetPollingState	OK	Sets the execution status of the communication module.(Refer to *SetPollingState* on page 8-283.)
GetPollingState	OK	Gets the polling state of the communication module.(Refer to *GetPollingState* on page 8-142.)
ErrorOut	-	-

Example:

Receive data using ReceiveData.

Set an IP address and parameter size (*4 integer type domain) in a parameter to use an Ethernet.

```
Dim data&(256)
Dim ipaddr&(4)

'Set the IP address of the destination.
ipaddr&(0) = 10
ipaddr&(1) = 1
ipaddr&(2) = 1
ipaddr&(3) = 101

'Gets the five integer type data.
ReceiveData "TcpNormal", data&(), 4*5, size&, ipaddr&(), 4*4
```

Send data using SendData.
Set an IP address and parameter size (*4 integer type domain) in a parameter to use an Ethernet.

```
Dim data&(256)
Dim ipaddr&(4)

'Set the IP address of the destination.
ipaddr&(0) = 10
ipaddr&(1) = 1
ipaddr&(2) = 1
ipaddr&(3) = 101

'Outputs five integer type data.
SendData "TcpNormal", data&(), 4*5, ipaddr&(), 4*4
```

9-4-10 UdpNormal

UDP Normal communication

IoModule identification name:

UdpNormal

UdpNormal2 (Fxxx series method)

Overview:

This is a module is for sending and receiving commands and data by Ethernet UDP protocol.

System data:

Identification name	Meaning	Initial value
enableDhcp	Enable DHCP 0: Disabled (Use the next IP Address.) 1: Enabled (Acquires the IP Address automatically.)	0
ipAddress	IP address	10.5.5.100
subnetMask	Subnet mask	255.255.255.0
defaultGateway	Default gateway	10.5.5.110
dns	DNS server	10.5.5.1
destIpAddress	Output IP address	0.0.0.0
portNo	Input port number	9600
portNo2	Output port number*1	-1

*1. If the input port number and the output port number are the same setting, set the output port number to -1.

Supported functions:

IoInitialize	OK	-
GetPort	-	-
PutPort	-	-
GetAll	-	-
PutAll	-	-
BusyOut	-	-
JudgeOut	-	-
RunOut	-	-
ReceiveData	OK	Receives data.(Refer to *ReceiveData* on page 8-231.)
SendData	OK	Sends data.(Refer to *SendData* on page 8-268.)
SendString	OK	Sends the character string data.(Refer to *SendString* on page 8-270.)
ReadPlcMemory	-	-
WritePlcMemory	-	-
SetPlcData	-	-
GetPlcData	-	-
SetPollingState	OK	Sets the execution status of the communication module.(Refer to *SetPollingState* on page 8-283.)
GetPollingState	OK	Gets the polling state of the communication module.(Refer to *GetPollingState* on page 8-142.)
ErrorOut	-	-

Example:

Receive data using ReceiveData.

Set an IP address and parameter size (*4 integer type domain) in a parameter to use an Ethernet.

```
Dim data&(256)
Dim ipaddr&(4)

'Set the IP address of the destination.
ipaddr&(0) = 10
ipaddr&(1) = 1
ipaddr&(2) = 1
ipaddr&(3) = 101

'Gets the five integer type data.
ReceiveData "UdpNormal", data&(), 4*5, size&, ipaddr&(), 4*4
```

Send data using SendData.
Set an IP address and parameter size (*4 integer type domain) in a parameter to use an Ethernet.

```
Dim data&(256)
Dim ipaddr&(4)

'Set the IP address of the destination.
ipaddr&(0) = 10
ipaddr&(1) = 1
ipaddr&(2) = 1
ipaddr&(3) = 101

'Outputs five integer type data.
SendData "UdpNormal", data&(), 4*5, ipaddr&(), 4*4
```

9-4-11 UdpPlcLinkM

PLC Link (MELSEC QnU/Q/QnAS) (UDP)

IoModule identification name:

UdpPlcLinkM

Overview:

This is a module is for sending and receiving commands and data by Ethernet PLC Link protocol.

System data:

Identification name	Meaning	Initial value
enableDhcp	Enable DHCP 0: Disabled (Use the next IP Address.) 1: Enabled (Acquires the IP Address automatically.)	0
ipAddress	IP address	10.5.5.100
subnetMask	Subnet mask	255.255.255.0
defaultGateway	Default gateway	10.5.5.110
dns	DNS server	10.5.5.1
destIpAddress	Output IP address	0.0.0.0
portNo	Port number	9600

PLC Link data:

Identification name	Meaning	Initial value
commandArea	Command Area type	Data register
commandMemoryAddress	Command Area address	0
responseArea	Response Area type	Data register
responseMemoryAddress	Response Area address	100
outputArea	Data Output Area type	Data register
outputMemoryAddress	Data Output Area address	200
handshake	Output control 0: OFF 1: Handshaking	1
responseTimeout	Retry interval[ms]	10000
responseTimeout2	Retry interval2[ms]	1000

Area classification:

Area classification name	Area classification number
Data register	168
File register	175
Link register	180

Supported functions:

IoInitialize	OK	-
GetPort	-	-
PutPort	-	-
GetAll	-	-
PutAll	-	-

BusyOut	-	-
JudgeOut	-	-
RunOut	-	-
ReceiveData	-	-
SendData	OK	Sends data.(Refer to *SendData* on page 8-268.)
SendString	-	-
ReadPlcMemory	OK	Reads a value from the PLC memory area.(Refer to *ReadPlcMemory* on page 8-229.)
WritePlcMemory	OK	Writes values in the PLC memory area.(Refer to *WritePlcMemory* on page 8-363.)
SetPlcData	OK	Creates the data that is written with the WritePlcMemory function.(Refer to *SetPlcData* on page 8-281.)
GetPlcData	OK	Gets data read with the ReadPlcMemory function.(Refer to *GetPlcData* on page 8-140.)
SetPollingState	OK	Sets the execution status of the communication module.(Refer to *SetPollingState* on page 8-283.)
GetPollingState	OK	Gets the polling state of the communication module.(Refer to *GetPollingState* on page 8-142.)
ErrorOut	-	-

Example:

Send data using SendData.

Set an IP address and parameter size (*4 integer type domain) in a parameter to use an Ethernet.

```
Dim data&(256)
Dim ipaddr&(4)

'Set the IP address of the destination.
ipaddr&(0) = 10
ipaddr&(1) = 1
ipaddr&(2) = 1
ipaddr&(3) = 101

'Outputs five integer type data.
SendData "UdpPlcLinkM", data&(), 4*5, ipaddr&(), 4*4
```

Gets 7ch data from 10ch of the Data register area using ReadPlcMemory.

Gets the data from readData() using GetPlcData.

```
Dim readData&(256)
Dim data3$(21)
'Gets the data from PLC.
ReadPlcMemory "UdpPlcLinkM", 176, 10, 7, readData&()

'Gets the data of the real number type.
GetPlcData "UdpPlcLinkM", readData&(), 0, 8, data1#
'Gets the data of the integer type.
GetPlcData "UdpPlcLinkM", readData&(), 8, 4, data2&
'Gets the data of the character type.
GetPlcData "UdpPlcLinkM", readData&(), 12, 5, data2&
```

Write in data for 7ch from 10ch of the Data register area using WritePlcMemory.

Set data to writeData() using SetPlcData.

```
Dim writeData&(256)

'Set the writeData (123.45) of the real number type.
SetPlcData "UdpPlcLinkM", writeData&(), 0, 8, 123.45
Set the writeData (20) of the integer type.
SetPlcData "UdpPlcLinkM", writeData&(), 32, 4, 20
'Set the writeData (OMRON) of the character type.
SetPlcData "UdpPlcLinkM", writeData&(), 36, 5, "OMRON"

'Write in data for 7ch from 10ch of the Data register area.
WritePlcMemory "UdpPlcLinkM", 168, 10, 7, writeData&()
```

9-4-12 UdpPlcLinkY

PLC Link (JEPMC MP) communication

IoModule identification name:

UdpPlcLinkY

Overview:

This is a module is for sending and receiving commands and data by Ethernet PLC Link protocol.

System data:

Identification name	Meaning	Initial value
enableDhcp	Enable DHCP 0: Disabled (Use the next IP Address.) 1: Enabled (Acquires the IP Address automatically.)	0
ipAddress	IP address	10.5.5.100
subnetMask	Subnet mask	255.255.255.0
defaultGateway	Default gateway	10.5.5.110
dns	DNS server	10.5.5.1
destIpAddress	Output IP address	0.0.0.0
portNo	Port number	9600

PLC Link data:

Identification name	Meaning	Initial value
commandArea	Command Area type	Data register
commandMemoryAddress	Command Area address	0
responseArea	Response Area type	Data register
responseMemoryAddress	Response Area address	100
outputArea	Data Output Area type	Data register
outputMemoryAddress	Data Output Area address	200
handshake	Output control 0: OFF 1: Handshaking	1
responseTimeout	Retry interval[ms]	10000
responseTimeout2	Retry interval2[ms]	1000

Area classification:

Area classification name	Area classification number
Data register	176

Supported functions:

IoInitialize	OK	-
GetPort	-	-
PutPort	-	-
GetAll	-	-
PutAll	-	-
BusyOut	-	-
JudgeOut	-	-

RunOut	-	-
ReceiveData	-	-
SendData	OK	Sends data.(Refer to *SendData* on page 8-268.)
SendString	-	-
ReadPlcMemory	OK	Reads a value from the PLC memory area.(Refer to *ReadPlcMemory* on page 8-229.)
WritePlcMemory	OK	Writes values in the PLC memory area.(Refer to *WritePlcMemory* on page 8-363.)
SetPlcData	OK	Creates the data that is written with the WritePlcMemory function.(Refer to *SetPlcData* on page 8-281.)
GetPlcData	OK	Gets data read with the ReadPlcMemory function.(Refer to *GetPlcData* on page 8-140.)
SetPollingState	OK	Sets the execution status of the communication module.(Refer to *SetPollingState* on page 8-283.)
GetPollingState	OK	Gets the polling state of the communication module.(Refer to *GetPollingState* on page 8-142.)
ErrorOut	-	-

Example:

Send data using SendData.

Set an IP address and parameter size (*4 integer type domain) in a parameter to use an Ethernet.

```
Dim data&(256)
Dim ipaddr&(4)

'Set the IP address of the destination.
ipaddr&(0) = 10
ipaddr&(1) = 1
ipaddr&(2) = 1
ipaddr&(3) = 101

'Outputs five integer type data.
SendData "UdpPlcLinkY", data&(), 4*5, ipaddr&(), 4*4
```

Gets 7ch data from 10ch of the Data register area using ReadPlcMemory.

Gets the data from readData() using GetPlcData.

```
Dim readData&(256)
Dim data3$(21)

'Gets the data from PLC.
ReadPlcMemory "UdpPlcLinkY", 168, 10, 7, readData&()

'Gets the data of the real number type.
GetPlcData "UdpPlcLinkY", readData&(), 0, 8, data1#
'Gets the data of the integer type.
GetPlcData "UdpPlcLinkY", readData&(), 8, 4, data2&
'Gets the data of the character type.
GetPlcData "UdpPlcLinkY", readData&(), 12, 5, data2&
```

Write in data for 7ch from 10ch of the Data register area using WritePlcMemory.

Set data to writeData() using SetPlcData.

```
Dim writeData&(256)

'Set the writeData (123.45) of the real number type.
SetPlcData "UdpPlcLinkY", writeData&(), 0, 8, 123.45
Set the writeData (20) of the integer type.
SetPlcData "UdpPlcLinkY", writeData&(), 32, 4, 20
'Set the writeData (OMRON) of the character type.
SetPlcData "UdpPlcLinkY", writeData&(), 36, 5, "OMRON"

'Write in data for 7ch from 10ch of the Data register area.
WritePlcMemory "UdpPlcLinkY", 176, 10, 7, writeData&()
```

9-4-13 UdpPlcLink

PLC Link (SYSMAC CS/CJ/CP/One) (UDP)

IoModule identification name:

UdpPlcLink

Overview:

This is a module is for sending and receiving commands and data by Ethernet PLC Link protocol.

System data:

Identification name	Meaning	Initial value
enableDhcp	Enable DHCP 0: Disabled (Use the next IP Address.) 1: Enabled (Acquires the IP Address automatically.)	0
ipAddress	IP address	10.5.5.100
subnetMask	Subnet mask	255.255.255.0
defaultGateway	Default gateway	10.5.5.110
dns	DNS server	10.5.5.1
destIpAddress	Output IP address	0.0.0.0
portNo	Port number	9600

PLC Link data:

Identification name	Meaning	Initial value
commandArea	Command Area type	CIO Area (CIO)
commandMemoryAddress	Command Area address	0
responseArea	Response Area type	CIO Area (CIO)
responseMemoryAddress	Response Area address	100
outputArea	Data Output Area type	CIO Area (CIO)
outputMemoryAddress	Data Output Area address	200
handshake	Output control 0: OFF 1: Handshaking	1
responseTimeout	Retry interval[ms]	10000
responseTimeout2	Retry interval2[ms]	1000

Area classification:

Area classification name	Area classification number
CIO Area (CIO)	176
Work Area (WR)	177
Holding Bit Area (HR)	178
Auxiliary Bit Area (AR)	179
DM Area (DM)	130
EM Area (EM0)	160
EM Area (EM1)	161
EM Area (EM2)	162
EM Area (EM3)	163
EM Area (EM4)	164

Area classification name	Area classification number
EM Area (EM5)	165
EM Area (EM6)	166
EM Area (EM7)	167
EM Area (EM8)	168
EM Area (EM9)	169
EM Area (EMA)	170
EM Area (EMB)	171
EM Area (EMC)	172

Supported functions:

IoInitialize	OK	-
GetPort	-	-
PutPort	-	-
GetAll	-	-
PutAll	-	-
BusyOut	-	-
JudgeOut	-	-
RunOut	-	-
ReceiveData	-	-
SendData	OK	Sends data.(Refer to *SendData* on page 8-268.)
SendString	-	-
ReadPlcMemory	OK	Reads a value from the PLC memory area.(Refer to *ReadPlcMemory* on page 8-229.)
WritePlcMemory	OK	Writes values in the PLC memory area.(Refer to *WritePlcMemory* on page 8-363.)
SetPlcData	OK	Creates the data that is written with the WritePlcMemory function.(Refer to *SetPlcData* on page 8-281.)
GetPlcData	OK	Gets data read with the ReadPlcMemory function.(Refer to *GetPlcData* on page 8-140.)
SetPollingState	OK	Sets the execution status of the communication module.(Refer to *SetPollingState* on page 8-283.)
GetPollingState	OK	Gets the polling state of the communication module.(Refer to *GetPollingState* on page 8-142.)
ErrorOut	-	-

Example:

Send data using SendData.

Set an IP address and parameter size (*4 integer type domain) in a parameter to use an Ethernet.

```
Dim data&(256)
Dim ipaddr&(4)

'Set the IP address of the destination.
ipaddr&(0) = 10
ipaddr&(1) = 1
ipaddr&(2) = 1
ipaddr&(3) = 101

'Outputs five integer type data.
SendData "UdpPlcLink", data&(), 4*5, ipaddr&(), 4*4
```

Gets 7ch data from 10ch of the DM area using ReadPlcMemory.
Gets the data from readData() using GetPlcData.

```
Dim readData&(256)
Dim data3$(21)

'Gets the data from PLC.
ReadPlcMemory "UdpPlcLink", 130, 10, 7, readData&()

'Gets the data of the real number type.
GetPlcData "UdpPlcLink", readData&(), 0, 8, data1#
'Gets the data of the integer type.
GetPlcData "UdpPlcLink", readData&(), 8, 4, data2&
''Gets the data of the character type.
GetPlcData "UdpPlcLink", readData&(), 12, 5, data2&
```

Write in data for 7ch from 10ch of the DM area using WritePlcMemory.
Set data to writeData() using SetPlcData.

```
Dim writeData&(256)

'Set the writeData (123.45) of the real number type.
SetPlcData "UdpPlcLink", writeData&(), 0, 8, 123.45
Set the writeData (20) of the integer type.
SetPlcData "UdpPlcLink", writeData&(), 32, 4, 20
'Set the writeData (OMRON) of the character type.
SetPlcData "UdpPlcLink", writeData&(), 36, 5, "OMRON"

'Write in data for 7ch from 10ch of the DM area.
WritePlcMemory "UdpPlcLink", 130, 10, 7, writeData&()
```

9-4-14 TcpPlcLinkM

PLC Link (MELSEC QnU/Q/QnAS) (TCP)

IoModule identification name:

TcpPlcLinkM

Overview:

This is a module is for sending and receiving commands and data by Ethernet PLC Link protocol.

System data:

Identification name	Meaning	Initial value
enableDhcp	Enable DHCP 0: Disabled (Use the next IP Address.) 1: Enabled (Acquires the IP Address automatically.)	0
ipAddress	IP address	10.5.5.100
subnetMask	Subnet mask	255.255.255.0
defaultGateway	Default gateway	10.5.5.110
dns	DNS server	10.5.5.1
wins	WINS server	0.0.0.0, 0.0.0.0*1
enableDhcp2	Enable DHCP 0: Disabled (Use the next IP Address.) 1: Enabled (Acquires the IP Address automatically.)	0
ipAddress2	IP address2	10.5.6.100
subnetMask2	Subnet mask2	255.255.255.0
defaultGateway2	Default gateway2	10.5.6.100
dns2	DNS server2	10.5.6.100
wins2	WINS server2	0.0.0.0, 0.0.0.0*1
destIpAddress	Output IP address	10.5.5.101
portNo	Port number	9876

*1. This value is displayed *0.0.0.0* on the **Communication** setting window of **System Settings** in **Tool**, it means an empty data.

PLC Link data:

Identification name	Meaning	Initial value
commandArea	Command Area type	Data register
commandMemoryAddress	Command Area address	0
responseArea	Response Area type	Data register
responseMemoryAddress	Response Area address	100
outputArea	Data Output Area type	Data register
outputMemoryAddress	Data Output Area address	200
handshake	Output control 0: OFF 1: Handshaking	1
responseTimeout	Retry interval[ms]	10000
outputBuffering	Asynchronous output 0: OFF 1: ON	0

Identification name	Meaning	Initial value
	Polling cycle[ms]	0

Area classification:

Area classification name	Area classification number
Data register	168
File register	175
Link register	180

Supported functions:

IoInitialize	OK	-
GetPort	OK	Gets the input state of the specified input port. (Refer to *GetPort* on page 8-144.)
PutPort	OK	Sets the output state of the specified output terminal. (Refer to *PutPort* on page 8-225.)
GetAll	OK	Gets the input states of all input terminals. (Refer to *GetAll* on page 8-134.)
PutAll	OK	Sets the output state of all output terminals. (Refer to *PutAll* on page 8-223.)
BusyOut	-	-
JudgeOut	-	-
RunOut	-	-
ReceiveData	-	-
SendData	OK	Sends data.(Refer to *SendData* on page 8-268.)
SendString	-	-
ReadPlcMemory	OK	Reads a value from the PLC memory area.(Refer to *ReadPlcMemory* on page 8-229.)
WritePlcMemory	OK	Writes values in the PLC memory area.(Refer to *WritePlcMemory* on page 8-363.)
SetPlcData	OK	Creates the data that is written with the WritePlcMemory function.(Refer to *SetPlcData* on page 8-281.)
GetPlcData	OK	Gets data read with the ReadPlcMemory function.(Refer to *GetPlcData* on page 8-140.)
SetPollingState	OK	Sets the execution status of the communication module.(Refer to *SetPollingState* on page 8-283.)
GetPollingState	OK	Gets the polling state of the communication module.(Refer to *GetPollingState* on page 8-142.)
ErrorOut	-	-

Example:

Send data using SendData.

Set an IP address and parameter size (*4 integer type domain) in a parameter to use an Ethernet.

```
Dim data&(256)
Dim ipaddr&(4)

'Set the IP address of the destination.
ipaddr&(0) = 10
ipaddr&(1) = 1
ipaddr&(2) = 1
ipaddr&(3) = 101

'Outputs five integer type data.
SendData "TcpPlcLinkM", data&(), 4*5, ipaddr&(), 4*4
```

Gets 7ch data from 10ch of the Data register area using ReadPlcMemory.
Gets the data from readData() using GetPlcData.

```
Dim readData&(256)
Dim data3$

'Gets the data from PLC.
ReadPlcMemory "TcpPlcLinkM", 168, 10, 7, readData&()

'Gets the data of the real number type.
GetPlcData "TcpPlcLinkM", readData&(), 0, 8, data1#
'Gets the data of the integer type.
GetPlcData "TcpPlcLinkM", readData&(), 8, 4, data2&
'Gets the data of the character type.
GetPlcData "TcpPlcLinkM", readData&(), 12, 5, data3$
```

Write in data for 7ch from 10ch of the Data register area using WritePlcMemory.
Set data to writeData() using SetPlcData.

```
Dim writeData&(256)

'Set the writeData (123.45) of the real number type.
SetPlcData "TcpPlcLinkM", writeData&(), 0, 8, 123.45
Set the writeData (20) of the integer type.
SetPlcData "TcpPlcLinkM", writeData&(), 32, 4, 20
'Set the writeData (OMRON) of the character type.
SetPlcData "TcpPlcLinkM", writeData&(), 36, 5, "OMRON"

'Write in data for 7ch from 10ch of the Data register area.
WritePlcMemory "TcpPlcLinkM", 168, 10, 7, writeData&()
```

9-4-15 TcpPlcLink

PLC Link (SYSMAC CS/CJ/CP/One) (TCP)

IoModule identification name:

TcpPlcLink

Overview:

This is a module is for sending and receiving commands and data by Ethernet PLC Link protocol.

System data:

Identification name	Meaning	Initial value
enableDhcp	Enable DHCP 0: Disabled (Use the next IP Address.) 1: Enabled (Acquires the IP Address automatically.)	0
ipAddress	IP address	10.5.5.100
subnetMask	Subnet mask	255.255.255.0
defaultGateway	Default gateway	10.5.5.110
dns	DNS server	10.5.5.1
wins	WINS server	0.0.0.0, 0.0.0.0*1
enableDhcp2	Enable DHCP 0: Disabled (Use the next IP Address.) 1: Enabled (Acquires the IP Address automatically.)	0
ipAddress2	IP address2	10.5.6.100
subnetMask2	Subnet mask2	255.255.255.0
defaultGateway2	Default gateway2	10.5.6.100
dns2	DNS server2	10.5.6.100
wins2	WINS server2	0.0.0.0, 0.0.0.0*1
destIpAddress	Output IP address	10.5.5.101
portNo	Port number	9876

*1. This value is displayed *0.0.0.0* on the **Communication** setting window of **System Settings** in **Tool**, it means an empty data.

PLC Link data:

Identification name	Meaning	Initial value
commandArea	Command Area type	CIO Area (CIO)
commandMemoryAddress	Command Area address	0
responseArea	Response Area type	CIO Area (CIO)
responseMemoryAddress	Response Area address	100
outputArea	Data Output Area type	CIO Area (CIO)
outputMemoryAddress	Data Output Area address	200
handshake	Output control 0: OFF 1: Handshaking	1
responseTimeout	Retry interval[ms]	10000
outputBuffering	Asynchronous output 0: OFF 1: ON	0

Identification name	Meaning	Initial value
pollingMinCycle	Polling cycle[ms]	0

Area classification:

Area classification name	Area classification number
CIO Area (CIO)	176
Work Area (WR)	177
Holding Bit Area (HR)	178
Auxiliary Bit Area (AR)	179
DM Area (DM)	130
EM Area (EM0)	160
EM Area (EM1)	161
EM Area (EM2)	162
EM Area (EM3)	163
EM Area (EM4)	164
EM Area (EM5)	165
EM Area (EM6)	166
EM Area (EM7)	167
EM Area (EM8)	168
EM Area (EM9)	169
EM Area (EMA)	170
EM Area (EMB)	171
EM Area (EMC)	172

Supported functions:

IoInitialize	OK	-
GetPort	OK	Gets the input state of the specified input port. (Refer to *GetPort* on page 8-144.)
PutPort	OK	Sets the output state of the specified output terminal. (Refer to *PutPort* on page 8-225.)
GetAll	OK	Gets the input states of all input terminals. (Refer to *GetAll* on page 8-134.)
PutAll	OK	Sets the output state of all output terminals. (Refer to *PutAll* on page 8-223.)
BusyOut	-	-
JudgeOut	-	-
RunOut	-	-
ReceiveData	-	-
SendData	OK	Sends data.(Refer to *SendData* on page 8-268.)
SendString	-	-
ReadPlcMemory	OK	Reads a value from the PLC memory area.(Refer to *ReadPlcMemory* on page 8-229.)
WritePlcMemory	OK	Writes values in the PLC memory area.(Refer to *WritePlcMemory* on page 8-363.)
SetPlcData	OK	Creates the data that is written with the WritePlcMemory function.(Refer to *SetPlcData* on page 8-281.)
GetPlcData	OK	Gets data read with the ReadPlcMemory function.(Refer to *GetPlcData* on page 8-140.)

SetPollingState	OK	Sets the execution status of the communication module.(Refer to *SetPollingState* on page 8-283.)
GetPollingState	OK	Gets the polling state of the communication module.(Refer to *GetPollingState* on page 8-142.)
ErrorOut	-	-

Example:

Send data using SendData.

Set an IP address and parameter size (*4 integer type domain) in a parameter to use an Ethernet.

```
Dim data&(256)
Dim ipaddr&(4)

'Set the IP address of the destination.
ipaddr&(0) = 10
ipaddr&(1) = 1
ipaddr&(2) = 1
ipaddr&(3) = 101

'Outputs five integer type data.
SendData "TcpPlcLink", data&(), 4*5, ipaddr&(), 4*4
```

Gets 7ch data from 10ch of the DM area using ReadPlcMemory.

Gets the data from readData() using GetPlcData.

```
Dim readData&(256)
Dim data3$

'Gets the data from PLC.
ReadPlcMemory "TcpPlcLink", 130, 10, 7, readData&()
'Gets the data of the real number type.
GetPlcData "TcpPlcLink", readData&(), 0, 8, data1#
'Gets the data of the integer type.
GetPlcData "TcpPlcLink", readData&(), 8, 4, data2&
'Gets the data of the character type.
GetPlcData "TcpPlcLink", readData&(), 12, 5, data3$
```

Write in data for 7ch from 10ch of the DM area using WritePlcMemory.

Set data to writeData() using SetPlcData.

```
Dim writeData&(256)

'Set the writeData (123.45) of the real number type.
SetPlcData "TcpPlcLink", writeData&(), 0, 8, 123.45
Set the writeData (20) of the integer type.
SetPlcData "TcpPlcLink", writeData&(), 32, 4, 20
'Set the writeData (OMRON) of the character type.
SetPlcData "TcpPlcLink", writeData&(), 36, 5, "OMRON"

'Write in data for 7ch from 10ch of the DM area.
WritePlcMemory "TcpPlcLink", 130, 10, 7, writeData&()
```

9-5 List for Processing Item Identifier

9-5-1 Measurement

Processing item	Identifier	Processing item	Identifier
Search	Search	Flexible Search	FlexibleSearch
Sensitive Search	SensitiveSearch	ECM Search	EcmSearch
Shape Search II	ShapeSearch2	Shape Search III	ShapeSearch3
EC Corner	EcCorner	EC Cross	EcCross
Classification	Classification	Edge Position	EdgePosition
Edge Pitch	EdgePitch	Scan Edge Position	ScanEdgePosition
Scan Edge Width	ScanEdgeWidth	Circular Scan Edge Position	CircularScanEdgePosition
Circular Scan Edge Width	CircularScanEdgeWidth	Intersection	Intersection
Color Data	ColorData	Gravity and Area	ColorArea
Labeling	ColorLabeling	Label Data	LabelData
Defect	Defect	Precise Defect	PreciseDefect
Fine Matching	FineMatching	Character Inspection	CharacterInspect
Date Verification	DateVerification	Model Dictionary	ModelDictionary
2DCode II	2dcr2	2DCode	2dcrV400
Barcode	BarcodeCC	OCR User Dictionary	CharacterDictionary
OCR	CharacterRecognition	Circle Angle	CircleAngle
Glue Bead Inspection	EmbrocationTrack	Search II	Search2
AI Fine Matching	FineMatchingAnalysisAI		

9-5-2 Input image

Processing item	Identifier	Processing item	Identifier
Camera Image Input	CameraImage	Camera Image Input FH	CameraImageFH
Camera Image Input FHV	CameraImageFHV	Camera Image Input HDR	CameraImageHDR
Camera Image Input HDR Lite	CameraImageHDRLite	Photometric Stereo Image Input	CameraImagePS
Camera Switching	CameraSwitch	Measurement Image Switching	MeasureImageSwitch
Multi-Trigger Imaging	MultiImaging	*Multi-Trigger Imaging End	MultiImagingEnd
Multi-Trigger Imaging Task	MultiImagingTask		

Note: An item with * in the head is not displayed.

9-5-3 Compensate image

Processing item	Identifier	Processing item	Identifier
Position Compensation	Scroll	Filtering	Filtering
Background Suppression	FilterHighContrast	Brightness Correct Filter	FilterShading
Color Gray Filter	FilterColorGray	Extract Color Filter	FilterExtractColor
Anti Color Shading	FilterEven	Stripes Removal Filter II	FilterRank
Polar Transformation	TransPolar	Trapezoidal Correction	TrapezoidalCorrection2
Machine Simulator	MachineSimulator	Image Subtraction	DifferenceExtract
Advanced filter	FilterAdvanced	Panorama	Panorama2
Trim and Resize	TrimAndResize	Image Stitching	Stitch

9-5-4 Support measurement

Processing item	Identifier	Processing item	Identifier
Unit Macro	Macro	Unit Calculation Macro	MacroCalc
Calculation	Calculation	Line Regression	ApproximateLine
Circle Regression	ApproximateCircle	Precise Calibration	CalibrationP
User Data	UserData	Set Unit Data	SetUnitData
Get Unit Data	GetUnitData	Set Unit Figure	SetUnitFigure
Get Unit Figure	GetUnitFigure	Trend Monitor	TrendMonitor
Image Logging	ImageLogging	Image Conversion Logging	MeasureImageSave
Data Logging	DataLogging	Elapsed Time	ElapsedTime
Wait	Wait	Focus	SettingFocus
Iris	SettingBright	Parallelize	Parallelize
*Parallelize End	ParallelizeEnd	Parallelize Task	ParallelizeTask
Statistics	Statistics	Calibration Data Reference	ReferenceCalibData
Position Data Calculation	SetPositionData	Stage Data	StageData
Robot Data	RobotData	Vision Master Calibration	AlignmentCalibration
PLC Master Calibration	AlignmentCalibrationPLC	Transfer Position Data	ScrollPositionData
Calc Axis Move	ScrollCalculation	Calc Axis Move by Multi-point	ScrollCalculationMulti
Detection Point	DetectionPoint	Manual Position Setting	ForcePositionSetting
Camera Calibration	CameraCalibration	Data Save	DataSave
Conveyor Calibration	ConveyorCalibration	Scene	Scene
System information	SystemInfo		

Note: An item with * in the head is not displayed.

9-5-5 Branch

Processing item	Identifier	Processing item	Identifier
Conditional Branch	Branch	End	End
DI Branch	InputBranch	Control Flow Normal	ControlFlowStringCommand
Control Flow PLC Link	ControlFlowPlcLinkCommand	Control Flow Parallel	ControlFlowParallelCommand
Control Flow Fieldbus	ControlFlowFieldbusCommand	Conditional execution (If)	If
Conditional execution (Else)	IfElse	*Condition execution end	IfEnd
Loop	Loop	Loop suspension	LoopExit
*End loop	LoopEnd	Select execution (Select)	Select
Select execution (Case)	SelectCase	*End selection execution	SelectEnd

Note: An item with * in the head is not displayed.

9-5-6 Output result

Processing item	Identifier	Processing item	Identifier
Result output (I / O)	OutputIO	Result output (Message)	OutputString
Data Output	NormalData	Parallel Data Output	ParallelData
Parallel Judgement Output	ParallelJudge	Fieldbus Data Output	FieldbusData
Result output (Parallel I / O)	OutputParallelIo		

9-5-7 Display result

Processing item	Identifier	Processing item	Identifier
Result Display	ResultDisplay	Display Image File	DisplayImageFile
Display Last NG Image	DisplayLastNG	Conveyor Panorama Display	ConveyorPanorama
Display image hold	DisplayImage		

9-6 Figure Data List

To set or acquire a model figure or region figure held in a processing unit, use an array to specify the figure data to be set or acquired.

9-6-1 Figure Data Structure List

The structure of figure data is indicated below.

Array element	Description	
figure(0)	Figure data header information	This is figure data header information. Includes the number of figures and figure data size information. Upper 16 bits: Number of figures Lower 16 bits: Number of bytes of figure data size (figure array length x 4) Figure data header information = Number of bytes of figure data size + Number of figure data x 65536 Number of figures: Sets the number of figures included in the figure data. Normally 1 should be set. If you are combining multiple figures, set the number of figures that are combined (2 or more). Number of bytes of figure data size: Set the figure data size converted into bytes. Set the value that is 4 times the number of array elements. Example: One rectangle Array length = 5, Number of figure data items = 1 Number of bytes of figure data size = 5 x 4 = 20 Figure data header information = 20 + 1 x 65536 = 65556
figure(1)	Figure 0 type information	Type information of figure 0 data. Includes drawing mode and figure type information. Upper 16 bits: Drawing mode Lower 16 bits: Figure type Figure type information = Figure type + Drawing mode x 65536 Drawing mode: Set whether the figure drawing mode is OR mode or NOTE mode. Normally 0 (OR mode) should be set. When multiple figures are used and you want to exclude some of the figures, set 1 (NOT mode) for the 2nd or later figures. Figure type: Set the figure type (line, rectangle, etc.). Example: One rectangle (drawing mode is OR) Figure type = 8, Drawing mode = 0 Figure type information = 8 + 0 x 65536 = 8
figure(2)	Figure 0 data	Figure data of figure 0. The size and content depends on the figure type.
figure(M)	Figure 1 type information	Type information of figure 1 data.
figure(M+1)	Figure 1 data	Figure data of figure 1. The size and content depends on the figure type.
:	:	:
figure(N*M)	Figure N type information	Type information of figure N data.
figure(N*M+1)	Figure N data	Figure data of figure N. The size and content depends on the figure type.

9-6-2 List of Figure Data Array Elements

The array elements of each figure and corresponding settings are shown below. The information shown in the table is for a figure count of 1.

Figure	Figure kind	Array element	Description
Point	1	figure(0)	Figure data header information
		figure(1)	Figure type information
		figure(2)	X-coordinate
		figure(3)	Y-coordinate
Line	2	figure(0)	Figure data header information
		figure(1)	Figure type information
		figure(2)	First point X
		figure(3)	First point Y
		figure(4)	Second point X
		figure(5)	Second point Y
Wide line	4	figure(0)	Figure data header information
		figure(1)	Figure type information
		figure(2)	First point X
		figure(3)	First point Y
		figure(4)	Second point X
		figure(5)	Second point Y
		figure(6)	Width
Rectangle	8	figure(0)	Figure data header information
		figure(1)	Figure type information
		figure(2)	Upper left X
		figure(3)	Upper left Y
		figure(4)	Lower right X
		figure(5)	Lower right Y
Ellipse	16	figure(0)	Figure data header information
		figure(1)	Figure type information
		figure(2)	X-coordinate of center
		figure(3)	Y-coordinate of center
		figure(4)	X-direction radius
		figure(5)	Y-direction radius
Circle	32	figure(0)	Figure data header information
		figure(1)	Figure type information
		figure(2)	X-coordinate of center
		figure(3)	Y-coordinate of center
		figure(4)	Radius
Wide circle	64	figure(0)	Figure data header information
		figure(1)	Figure type information
		figure(2)	X-coordinate of center
		figure(3)	Y-coordinate of center
		figure(4)	Radius
		figure(5)	Width

Figure	Figure kind	Array element	Description
Arc	128	figure(0)	Figure data header information
		figure(1)	Figure type information
		figure(2)	X-coordinate of center
		figure(3)	Y-coordinate of center
		figure(4)	Radius
		figure(5)	Start angle
		figure(6)	End angle
Wide arc	256	figure(0)	Figure data header information
		figure(1)	Figure type information
		figure(2)	X-coordinate of center
		figure(3)	Y-coordinate of center
		figure(4)	Radius
		figure(5)	Start angle
		figure(6)	End angle
		figure(7)	Width
Polygon	512	figure(0)	Figure data header information
		figure(1)	Figure type information
		figure(2)	Number of vertices
		figure(3)	First point X
		figure(4)	First point Y
		figure(5)	Second point X
		figure(6)	Second point Y
		:	:
		figure(19)	9th point X
		figure(20)	9th point Y
		figure(21)	10th point X
		figure(22)	10th point Y

9-7 List of Figure Numbers

To set or acquire a model figure or region figure held in a processing unit, specify the number of the figure to be set or acquired. The figure numbers and figures of each processing item are shown below.

Item	Figure number	Description
Search	0	Model registration figure (model registration)
	1	Measurement region figure (region setting)
Search II	0	Model registration figure (model registration)
	1	Measurement region figure (region setting)
Flexible Search	0	Model registration figure for model 0 (model registration)
	1	Model registration figure for model 1 (model registration)
	2	Model registration figure for model 2 (model registration)
	3	Model registration figure for model 3 (model registration)
	4	Model registration figure for model 4 (model registration)
	5	Model registration figure for model 5 (model registration)
Sensitive Search	0	Model registration figure (model registration)
	1	Measurement region figure (region setting)
ECM Search	0	Model registration figure (model registration)
	1	Measurement region figure (region setting)
	2	Mask registration figure (model registration)
	3	Model registration figure (error model registration)
	4	Mask registration figure(error model registration)
EC Circle Search	0	Model registration figure (model registration)
	1	Measurement region figure (region setting)
Shape Search II	0	Model registration figure (model registration)
	1	Measurement region figure (region setting)
Shape Search III	0	Model registration figure (model registration)
	1	Measurement region figure (region setting)
Ec Corner	0	Measurement region figure (region setting)
Ec Cross	0	Measurement region figure (region setting)
Classification	0	Index 0, Model registration figure for model 0 (model registration)
	1	Index 0, Model registration figure for model 1 (model registration)
	2	Index 0, Model registration figure for model 2 (model registration)
	3	Index 0, Model registration figure for model 3 (model registration)
	4	Index 0, Model registration figure for model 4 (model registration)
	5	Index 1, Model registration figure for model 0 (model registration)
	6	Index 1, Model registration figure for model 1 (model registration)
	:	:

Item	Figure number	Description
	178	Index 35, Model registration figure for model 3 (model registration)
	179	Index 35, Model registration figure for model 4 (model registration)
	180	Measurement region figure (region setting)
	181	Index 36, Model registration figure for model 0 (model registration)
	182	Index 36, Model registration figure for model 1 (model registration)
	:	:
	999	Index 199, Model registration figure for model 3 (model registration)
	1000	Index 199, Model registration figure for model 4 (model registration)
Edge Position	0	Measurement region figure (region setting)
Edge Pitch	0	Measurement region figure (region setting)
Scan Edge Position	0	Measurement region figure (region setting)
Scan Edge Width	0	Measurement region figure (region setting)
Circular Scan Edge Position	0	Measurement region figure (region setting)
Circular Scan Edge Width	0	Measurement region figure (region setting)
Intersection	0	Measurement region figure for straight line 0 (region setting)
	2	Measurement region figure for straight line 1 (region setting)
Color Data	0	Measurement region figure (region setting)
Gravity and Area	0	Measurement region figure (region setting)
Labeling	0	Measurement region figure (region setting)
Label Data	-	No figure
Defect	0	Measurement region figure (region setting)
Precise Defect	0	Measurement region figure (region setting)
Fine Matching	0	Model registration figure (model registration)
Character inspection	0	Measurement region figure (region setting)
Date Verification	-	No figure
Model Dictionary	0	Index 0, Model registration figure for model 0 (model registration)
	1	Index 0, Model registration figure for model 1 (model registration)
	2	Index 0, Model registration figure for model 2 (model registration)
	3	Index 0, Model registration figure for model 3 (model registration)
	4	Index 0, Model registration figure for model 4 (model registration)
	5	Index 1,
	6	Index 1, Model registration figure for model 1 (model registration)
	:	:
	178	Index 35, Model registration figure for model 3 (model registration)

Item	Figure number	Description
	179	Index 35, Model registration figure for model 4 (model registration)
	180	Measurement region figure (region setting)
2DCode	0	Measurement region figure (region setting)
2DCode II	0	Measurement region figure (region setting)
Barcode	0	Measurement region figure (region setting)
Circle Angle	0	Measurement region figure (region setting)
Glue Bead Inspection	-	No figure
AI Fine Matching	-	No figure
Position Compensation	0	Measurement region figure (region setting)
Filtering	0	Measurement region figure (region setting)
Background Suppression	0	Measurement region figure (region setting)
Brightness Correct Filter	0	Measurement region figure (region setting)
Color Gray Filter	0	Measurement region figure (region setting)
Extract Color Filter	0	Measurement region figure (region setting)
Anti Color Shading	0	Measurement region figure (region setting)
Stripes Removal Filter II	0	Measurement region figure (region setting)
Polar Transformation	0	Measurement region figure (region setting)
Trapezoidal Correction	0	Measurement region figure (region setting)
Machine Simulator	-	No figure
Image Subtraction	0	Model registration figure (model registration)
Advanced filter	0	Measurement region figure (region setting)
Panorama	-	No figure
Trim and Resize	0	Measurement region figure (Trim)
Image Stitching	-	No figure

9-8 Model Number List

When you need to re-register the processing unit model, specify the number of the model to be re-registered. The model numbers and models of each processing item are shown below.

Item	Model number	Description
Search	0	Search model
Search II	0	Search model
Flexible Search	0	Index 0, model 0 search model
	1	Index 0, model 1 search model
	2	Index 0, model 2 search model
	3	Index 0, model 3 search model
	4	Index 0, model 4 search model
Sensitive Search	0	Search model
ECM Search	0	Search model
EC Circle Search	0	Search model
Shape Search II	0	Search model
Shape Search III	0	Search model
Ec Corner	0	Reference position
Ec Cross	0	Reference position
Classification	0	Index 0, model 0 search model
	1	Index 0, model 1 search model
	2	Index 0, model 2 search model
	3	Index 0, model 3 search model
	4	Index 0, model 4 search model
	5	Index 1, model 0 search model
	6	Index 1, model 1 search model
	:	:
	178	Index 35, model 3 search model
	179	Index 35, model 4 search model
	180	Index 36, model 0 search model
	181	Index 36, model 1 search model
	:	:
	998	Index 199, model 3 search model
	999	Index 199, model 4 search model
Edge Position	0	Edge model
Edge Pitch	0	Edge model
Scan Edge Position	0	Edge model
Scan Edge Width	0	Edge model
Circular Scan Edge Position	0	Edge model
Circular Scan Edge Width	0	Edge model
Intersection	0	Edge model
Color Data	0	Color Data model
Gravity and Area	0	Gravity and Area model
Labeling	0	Labeling model
Label Data	-	No model

Item	Model number	Description
Defect	0	Defect model
Precise Defect	0	Precise Defect model
Fine Matching	0	Fine Matching model
Character inspection	-	No model
Date Verification	-	No model
Model Dictionary	0	Index 0, model 0 search model
	1	Index 0, model 1 search model
	2	Index 0, model 2 search model
	3	Index 0, model 3 search model
	4	Index 0, model 4 search model
	5	Index 1, model 0 search model
	6	Index 1, model 1 search model
	:	:
	178	Index 35, model 3 search model
	179	Index 35, model 4 search model
2DCode	-	No model
2DCode II	-	No model
Barcode	-	No model
Circle Angle	-	No model
Glue Bead Inspection	-	No model
AI Fine Matching	-	No model

9-9 Image Number List

When you need to access image data in a processing unit, specify the number of the image that you want to access. The image numbers and images of each processing item are shown below.

Item	Image number	Description
Search	-	No image
Search II	-	No image
Flexible Search	-	No image
Sensitive Search	-	No image
ECM Search	-	No image
EC Circle Search	-	No image
Shape Search II	-	No image
Shape Search III	-	No image
Ec Corner	-	No image
Ec Cross	-	No image
Classification	-	No image
Edge Position	-	No image
Edge Pitch	-	No image
Scan Edge Position	-	No image
Scan Edge Width	-	No image
Circular Scan Edge Position	-	No image
Circular Scan Edge Width	-	No image
Intersection	-	No image
Color Data	-	No image
Gravity and Area	-	No image
Labeling	-	No image
Label Data	-	No image
Defect	-	No image
Precise Defect	-	No image
Fine Matching	-	No image
Character inspection	-	No image
Date Verification	-	No image
Model Dictionary	-	No image
2DCode	-	No image
2DCode II	-	No image
Barcode	-	No image
OCR	-	No image
OCR User Dictionary	-	No image
Circle Angle	-	No image
Glue Bead Inspection	-	No image
AI Fine Matching	0	Binary difference image
	1	Grayscale difference image
	2	Restored image
	3	Labeling binary image
Camera Image Input	0	Camera image

Item	Image number	Description
Camera Image Input FH	0	Output image0
	:	:
	7	Output image7
Camera Image Input HDR	0	Camera image
Camera Image Input HDR Lite	0	Camera image
Photometric Stereo Image Input	0	Output image
	1	Shape image 1
	2	Shape image 2
	3	Shape image 3
	4	Texture image
	16	Captured image 1
	:	:
	23	Captured image 8
Camera Switching	0	Camera image
Measurement Image Switching	-	No image
Position Compensation	0	Position compensated image
Filtering	0	Filtered image
Background Suppression	0	Background suppressed image
Brightness Correct Filter	0	Brightness corrected image
Color Gray Filter	0	Color gray image
Extract Color Filter	0	Color extracted image
Anti Color Shading	0	Anti color shading image
Stripes Removal Filter II	0	Stripes removed image
Polar Transformation	0	Polar transformed image
Trapezoidal Correction	0	Trapezoidal corrected image
Machine Simulator	0	Axis shifted image
Image Subtraction	0	Subtraction image
Advanced filter	0	Output image 0
	1	Output image 1
	2	Output image 2
	3	Output image 3
Panorama	0	Panorama image
Trim and Resize	0	Output image
Image Stitching	0	Stitched image
Unit Macro	0	Processing unit image 0
	1	Processing unit image 1
	:	:
	30	Processing unit image 30
	31	Processing unit image 31
Unit Calculation Macro	-	No image
Calculation	-	No image
Line Regression	-	No image
Circle Regression	-	No image
Precise Calibration	0	Corrected image
User Data	-	No image

Item	Image number	Description
Set Unit Data	-	No image
Get Unit Data	-	No image
Set Unit Figure	-	No image
Get Unit Figure	-	No image
Trend Monitor	-	No image
Image Logging	-	No image
Image Conversion Logging	-	No image
Data Logging	-	No image
Elapsed Time	-	No image
Wait	-	No image
Focus	-	No image
Iris	-	No image
Parallelize	-	No image
Parallelize Task	-	No image
Statistics	-	No image
Reference Calib Data	0	Corrected image
Position Data Calculation	-	No image
Stage Data	-	No image
Robot Data	-	No image
Vision Master Calibration	-	No image
PLC Master Calibration	-	No image
Convert Position Data	-	No image
Movement Single Position	-	No image
Movement Multi Points	-	No image
Detection Point	-	No image
Camera Calibration	-	No image
Data Save	-	No image
Conditional Branch	-	No image
End	-	No image
DI Branch	-	No image
Control Flow Normal	-	No image
Control Flow PLC Link	-	No image
Control Flow Parallel	-	No image
Control Flow Fieldbus	-	No image
Selective Branch	-	No image
Data Output	-	No image
Parallel Data Output	-	No image
Parallel Judgement Output	-	No image
Fieldbus Data Output	-	No image
Result Display	-	No image
Display Image File	0	File display image 0
	1	File display image 1
	2	File display image 2
	3	File display image 3

Item	Image number	Description
Display Last NG Image	0	NG image 0
	1	NG image 1
	2	NG image 2
	3	NG image 3

9-10 List of Sub-Image Numbers

In addition to measurement images, the processing unit also contains images being processed and images that have been processed, and these can be displayed in the image window as sub-images. To display a sub-image, specify the number of the sub-image. The sub-image numbers and sub-images of each processing item are shown below.

Item	Sub-image	Description
Search	0	Measurement image
Search II	0	Measurement image
Flexible Search	0	Measurement image
Sensitive Search	0	Measurement image
ECM Search	0	Measurement image
	1	Image with the matched model overlaid on the measurement image Green: Matched model points Red: Unmatched model points
EC Circle Search	0	Measurement image
Shape Search II	0	Measurement image
Shape Search III	0	Measurement image
	1	Display of measurement image with detection results overlaid.
	2	Edge image
	3	Display of edge image with detection results overlaid.
Ec Corner	0	Measurement image
Ec Cross	0	Measurement image
Classification	0	Measurement image
Edge Position	0	Measurement image
	1	Profile image
Edge Pitch	0	Measurement image
	1	Profile image
Scan Edge Position	0	Measurement image
	1	Scan image
Scan Edge Width	0	Measurement image
	1	Scan image
Circular Scan Edge Position	0	Measurement image
	1	Overlay the scan area on the measurement image.
Circular Scan Edge Width	0	Measurement image
	1	Overlay the scan area on the measurement image.
Intersection	0	Measurement image
Color Data	0	Measurement image
	1	Masked image
Gravity and Area	0	Measurement image
	1	Color extraction image (when measuring a color image) Binary image (when measuring a monochrome image)
Labeling	0	Measurement image
	1	Color extraction image (when measuring a color image) Binary image (when measuring a monochrome image)
Label Data	0	Measurement image

Item	Sub-image	Description
Defect	0	Measurement image
	1	Masked defect profile image (with area measurement)
Precise Defect	0	Measurement image
	1	Masked defect profile image (with area measurement) Defect area image (without area measurement)
Fine Matching	0	Measurement image
	1	Difference image
Character inspection	0	Measurement image
Date Verification	0	Measurement image
Model Dictionary	0	Measurement image
2DCode	0	Measurement image
2DCode II	0	Measurement image
Barcode	0	Measurement image
OCR	0	Measurement image
OCR User Dictionary	0	Measurement image
Circle Angle	0	Measurement image
Glue Bead Inspection	0	Measurement image
	1	Color extraction image (when measuring a color image) Binary image (when measuring a monochrome image)
	3	Route (Reference path) image
	4	Binary image
AI Fine Matching	0	Measurement image + Label No. of all Defect position
	1	Binary difference image + Label No. of all Defect position
	2	Grayscale difference image + Label No. of all Defect position
	3	Restored image + Label No. of all Defect position
	4	Labeling binary image + Label No. of all Defect position
	5	Defect overlay image + Label No. of all Defect position
	20	Measurement image + Circle around Defect position
	21	Binary difference image + Circle around Defect position
	22	Grayscale difference image + Circle around Defect position
	23	Restored image + Circle around Defect position
	24	Labeling binary image + Circle around Defect position
	25	Defect overlay image + Circle around Defect position
	30	Measurement image + Circle around all Defect position
	31	Binary difference image + Circle around all Defect position
	32	Grayscale difference image + Circle around all Defect position
	33	Restored image + Circle around all Defect position
	34	Labeling binary image + Circle around all Defect position
	35	Defect overlay image + Circle around all Defect position
Camera Image Input	0	Camera image
Camera Image Input FH	0	Camera image
Camera Image Input HDR	0	Camera image
Camera Image Input HDR Lite	0	Camera image
Camera Switching	0	Camera image
Measurement Image Switching	0	Image after switching
	1	Image before switching

Item	Sub-image	Description
Position Compensation	0	After scroll
	1	Before scroll
Filtering	0	Filtered image
Background Suppression	0	Background suppressed image
Brightness Correct Filter	0	Brightness corrected image
Color Gray Filter	0	Color gray image
Extract Color Filter	0	Color extracted image
	1	Measurement image
Anti Color Shading	0	Anti color shading image
Stripes Removal Filter II	0	Stripes removed image
Polar Transformation	0	Polar transformed image
	1	Measurement image
Trapezoidal Correction	0	Trapezoidal corrected image
Machine Simulator	0	Measurement image
	1	Image before measurement
Image Subtraction	0	Subtraction image
	1	Measurement image
Advanced filter	0	Advanced filter image
	1	Output image 0
	2	Output image 1
	3	Output image 2
	4	Output image 3
	5	Measurement image
Panorama	0	Panorama image
Trim and Resize	0	Output image
	1	Measurement image
Image Stitching	0	Stitched image
	1	Measurement image
Unit Macro	0 to 100	Can defined as desired
Unit Calculation Macro	0	Measurement image
Calculation	0	Measurement image
Line Regression	0	Measurement image
Circle Regression	0	Measurement image
Precise Calibration	0	High-precision calibration image
User Data	0	Measurement image
Set Unit Data	0	Measurement image
Get Unit Data	0	Measurement image
Set Unit Figure	0	Measurement image
Get Unit Figure	0	Measurement image
Trend Monitor	0	Trend graph display
	1	Histogram display
Image Logging	0	Measurement image
Image Conversion Logging	0	Measurement image
Data Logging	0	Measurement image
Elapsed Time	0	Measurement image
Wait	0	Measurement image
Focus	0	Measurement image

Item	Sub-image	Description
Iris	0	Measurement image
Parallelize	0	Measurement image
Parallelize Task	0	Measurement image
Statistics	0	Graph display corresponding to *Data number* in setting data
	1	Data graph display of data 0
	2	Data graph display of data 1
	3	Data graph display of data 2
	4	Data graph display of data 3
	5	Data graph display of data 4
	6	Data graph display of data 5
	7	Data graph display of data 6
	8	Data graph display of data 7
Reference Calib Data	0	Calibration reference image
Position Data Calculation	0	Measurement image
Stage Data	0	Measurement image
Robot Data	0	Measurement image
Vision Master Calibration	0	Measurement image + Calibration progress display
	1	Measurement image
PLC Master Calibration	0	Measurement image + Calibration progress display
	1	Measurement image
Convert Position Data	0	Measurement image
Movement Single Position	0	Measurement image
Movement Multi Points	0	Measurement image
Detection Point	0	Measurement image
Manual Posiotion Setting	0	Measurement image
Camera Calibration	0	Measurement image
Data Save	0	Measurement image
Conditional Branch	0	Measurement image
End	0	Measurement image
DI Branch	0	Measurement image
Control Flow Normal	0	Measurement image
Control Flow PLC Link	0	Measurement image
Control Flow Parallel	0	Measurement image
Control Flow Fieldbus	0	Measurement image
Selective Branch	0	Measurement image
Data Output	0	Measurement image
Parallel Data Output	0	Measurement image
Parallel Judgement Output	0	Measurement image
Fieldbus Data Output	0	Measurement image
Result Display	0	Result image
Display Image File	0	Image 0
	1	Image 1
	2	Image 2
	3	Image 3

Item	Sub-image	Description
Display Last NG Image	0	Last NG
	1	1st preceding NG image (when two or more images are saved) Most recent NG image (when less than two images are saved)
	2	2nd preceding NG image (when three or more images are saved) Most recent NG image (when less than three images are saved)
	3	3rd preceding NG image (when four or more images are saved) Most recent NG image (when less than four images are saved)

I

Index

Index

Numerics

(,) Comma....4-3
(;) Semicolon 4-3
(:) Colon....4-3
(") Double Quotation Marks.... 4-3
( ) Space.... 4-3
(@) "At" mark....4-3
($) Dollar Mark....4-3
(*) Asterisk.... 4-3
(&) Ampersand.... 4-3
(#) Pound Sign....4-3
(‘) Apostrophe.... 4-3
*CLEARMEASUREDATA.... 5-20
*MCRINIT.... 5-18
*MEASUREDISPG.... 5-19
*MEASUREDISPI.... 5-19
*MEASUREDISPT....5-19
*MEASUREINIT....5-18
*MEASUREPROC.... 5-18
*RENUMPROC....5-19

A

Abs.... 8-24
AbsolutePath$.... 8-25
AddGlobalData.... 8-26
AddSystemData....8-27
AND.... 8-28
ApplicationPath$....8-29
ApplicationVersion$.... 8-30
ApproximationCircle.... 8-31
Arithmetic Operator.... 4-12
ArrayDims....8-32
ArrayLen.... 8-33
Array Variable.... 4-8
Asc....8-34
Assignment Operator....4-12
AssignUnit.... 8-35
Atn.... 8-36

B

Basic Syntax....4-2
Binary.... 4-5
BusyOut....8-37

C

Call (Structured Subroutine Call)....8-38
Call (User-defined Function Execution)....8-40
ChangeScene....8-41
ChangeSceneGroup....8-42
Character....4-3
Character expression.... 4-14
Character string....4-5, 4-7
CheckUnit.... 8-43
Chr$....8-44
ClearMeasureData.... 8-46
ClearScene....8-47
ClearSceneGroup....8-48
Close.... 8-49
CloseTextData.... 8-50
Cls.... 8-52
Comment....4-4
Communication Command Macro tool.... 1-3
Components of the Program Editing Screen....3-2
Components of the Setting Screen of the Communication Command Macro Tool and How to Configure Settings 3-15
Constant.... 4-5
Cont....8-53
ContinuousMeasure.... 8-54
CopyMeasureImage.... 8-55
CopyScene....8-56
CopySceneGroup....8-57
CopyUnit....8-58
CopyUnitFigure....8-59
CopyUnitImage....8-61
CopyUnitModel....8-62
Cos.... 8-64
Crspoint.... 8-65

D

Data Save Area....5-11
Data Scope....5-10
Date$....8-67
Debug....8-68
Debugging mode.... 6-2
DebugPrint....8-69
Decimal....4-5
DeleteProfileData.... 8-71
DeleteUnit....8-72
Description of the Setting Screen for the Unit Calculation Macro Processing Item and How to Configure Settings 3-8
Description of the Setting Screen of the Scene Control Macro Tool and How to Configure Settings.... 3-13
Description of the Setting Screen of the Unit Macro Processing Item and How to Configure Settings.... 3-23
Description of the System Status Console Window.... 3-7
Dim.... 8-73
DisableUnit.... 8-74
DisplaySubNo....8-75
DisplayUnitNo....8-76
Do Loop While....8-77
Double precision real number.... 4-5, 4-7
Dposline....8-78
DrawArc....8-80
DrawArcW.... 8-82
DrawBox....8-84
DrawCircle.... 8-86

DrawCircleW....8-88
DrawCursor.... 8-90
DrawEllipse....8-92
DrawFigure.... 8-94
DrawFillImage....8-96
DrawJudgeText....8-97
DrawLine.... 8-98
DrawLineW.... 8-100
DrawMeasureImage.... 8-102
DrawPoint.... 8-103
DrawPolygon.... 8-105
DrawSearchFigure....8-107
DrawText....8-109
DrawTextG.... 8-111
DrawUnitImage.... 8-113
Dskf....8-114

E

ElapsedTime.... 8-115
Eof....8-116
Erase....8-118
Errcmnd$.... 8-119
Errno....8-120
Error List.... 9-2
ErrorOut....8-121
EtherCAT.... 9-28
EtherNetIP.... 9-31
Exclusive Control in a Communication Process.... 5-21
Exclusive Control in a Measurement Process.... 5-20
ExecuteErrorProc.... 8-123
ExecuteImageLogging....8-125
Execution statement....4-4
ExitFzProcess....8-127
Exp.... 8-128
Expression....4-14
External reference data.... 5-2

F

Fcopy....8-129
Figure data.... 5-3
Figure Data List.... 9-70
FigureType....8-131
Fix....8-130
For To Step Next....8-132
Function....4-15

G

GetAll....8-134
GetGlobalData....8-135
GetImageSize....8-136
GetImageWindow.... 8-137
GetMeasureOut.... 8-139
GetPlcData.... 8-140
GetPollingState....8-142
GetPort.... 8-144
GetProfileData.... 8-145
GetSceneData.... 8-146
GetSceneGroupData.... 8-147
GetSystemData.... 8-148
GetText$.... 8-150
GetTextWindow.... 8-152
GetUnitData....8-153
GetUnitFigure.... 8-154
Global Data....5-8
Gosub....8-156
Goto....8-157
Graphic display (*MEASUREDISPG).... 5-19

H

Hex$.... 8-158
Hexadecimal....4-5
How to Use the Debug Function....6-2

I

If Then Else.... 8-159
If Then Elseif Else EndIf.... 8-160
Image data....5-6
Image display (*MEASUREDISPI).... 5-19
ImageFormat.... 8-162
Image Number List.... 9-78
ImageUpdate....8-164
Input$....8-166
Input#....8-165
InsertUnit.... 8-168
Int....8-169
Integer.... 4-5, 4-7
Isfile.... 8-170
ItemCount....8-171
ItemIdent$....8-172
ItemInfo....8-173
ItemTitle$....8-174

J

JudgeOut....8-175

K

Kill....8-176

L

Label....4-4
LanguageString$.... 8-178
LCase$.... 8-179
Left$....8-180
Len....8-181
Line....4-3
Line Input#....8-182
Line number....4-3
List....8-184
List of Figure Numbers.... 9-73
List of I/O Modules....9-27
List of Reserved Words.... 9-5
List of Sub-Image Numbers....9-82

LoadBackupData....8-185
LoadScene....8-186
LoadSceneGroup....8-187
LoadSystemData....8-188
LoadUnitData....8-189
Log....8-191
Logical Expression....4-15
Logic Operator....4-13
Lsqumeth....8-192

M

Macro(IO)>....6-5
Macro(SC)>....6-5
Macro(U**)>....6-5
Measure....8-193
MeasureDispG....8-195
MeasureDispI....8-196
MeasureDispT....8-197
MeasureId$....8-198
Measurement acceptance state....5-14
Measurement Execution (*MEASUREPROC)....5-18
Measurement Flow Update (*RENUMPROC)....5-19
Measurement Initialization (*MCRINIT)....5-18
Measurement Initialization (*MEASUREINIT)....5-18
Measurement prohibited state....5-14
Measurement result clear (*CLEARMEASUREDATA)....5-20
MeasureProc....8-199
MeasureStart....8-200
MeasureStop....8-201
MemoryInfo....8-203
MessageBox....8-204
Mid$....8-205
Mkdir....8-207
MOD....8-208
Model data....5-5
Model Number List....9-76
MoveUnit....8-209

N

Non-execution statement....4-4
NOT....8-210
Numerical expression....4-14

O

Octal....4-5
Open For Append As#....8-211
Open For Input As#....8-212
Open For Output As#....8-213
OpenTextData....8-215
Operator....4-12
Option Explicit....8-217
OR....8-218

P

Parallel Interface communication....9-37
ParallelIo....9-37
Piece$....8-219
PLC Link (JEPMC MP) communication....9-55
PLC Link (MELSEC QnU/Q/QnAS) (TCP)....9-61
PLC Link (MELSEC QnU/Q/QnAS) (UDP)....9-52
PLC Link (SYSMAC CS/CJ/CP/One) (TCP)....9-64
PLC Link (SYSMAC CS/CJ/CP/One) (UDP)....9-58
Print....8-221
Print#....8-222
PROFINET....9-34
PutAll....8-223
PutPort....8-225

R

RaiseErrorProcEvent....8-226
RaiseOptionEvent....8-228
ReadPlcMemory....8-229
ReceiveData....8-231
ReDim....8-233
Reference Variables....4-9
RefreshImageWindow....8-234
RefreshJudgeWindow....8-235
RefreshTextWindow....8-236
RefreshTimeWindow....8-237
Relational expression....4-14
Relational Operator....4-13
Release mode....6-2
Rem....8-238
Remeasure....8-239
RenumUnitNo....8-241
RGB....8-242
Right$....8-243
Rmdir....8-244
RunOut....8-245

S

SaveBackupData....8-247
SaveData....8-248
SaveImage....8-249
SaveMeasureImage....8-250
SaveScene....8-252
SaveSceneGroup....8-253
SaveSystemData....8-254
SaveUnitData....8-255
Saving and Loading Programs in the Communication Command Macro....3-27
Saving and Loading Programs in the Scene Control Macro....3-27
Saving and Loading Programs in the Unit Calculation Macro....3-27
Saving and Loading Programs in the Unit Macro....3-28
Scene Control Macro tool....1-3
SceneCount....8-257
SceneDescription$....8-258
SceneGroupCount....8-259
SceneGroupNo....8-260
SceneGroupTitle$....8-261
SceneMaker$....8-262
SceneNo....8-263

SceneTitle$....8-264
Scene variable....4-11
ScreenCapture....8-265
Select Case Case Else End Select....8-266
SendData....8-268
SendString....8-270
Serial Interface Normal communication....9-39
Serial Interface PLC Link (MELSEC QnU/Q/QnAS) communication)....9-41
Serial Interface PLC Link (SYSMAC CS/CJ/CP/One) communication....9-43
SerialNormal....9-39
SerialPlcLink....9-43
SerialPlcLinkM....9-41
SetDisplayUnitNo....8-271
SetDrawStyle....8-272
SetForegroundLine....8-274
SetGlobalData....8-275
SetImageWindow....8-276
SetMeasureImage....8-279
SetMeasureOut....8-280
SetPlcData....8-281
SetPollingState....8-283
SetProfileData....8-285
SetSceneData....8-286
SetSceneDescription....8-288
SetSceneGroupData....8-289
SetSceneGroupTitle....8-290
SetSceneMaker....8-291
SetSceneTitle....8-292
SetStop....8-293
SetSystemData....8-295
SetTextStyle....8-297
SetTextWindow....8-299
SetUnitData....8-300
SetUnitFigure....8-301
SetUnitJudge....8-303
SetUnitTitle....8-304
SetUserSubroutine....8-305
SetVar....8-307
Sin....8-309
Sqr....8-310
StartTimer....8-311
Statement....4-3
Stop....8-312
Str$....8-314
Str2$....8-315
Sub - End Sub....8-318
SubList....8-320
Subroutine....4-4
System data....5-9
System Data List....9-10
SystemInfo....8-321
SystemReset....8-322
System variable....4-11

T

Tan....8-323
TcpClient....9-46
TCP Client Normal communication....9-46
TcpNormal....9-48
TCP Normal communication....9-48
TcpPlcLink....9-64
TcpPlcLinkM....9-61
TDM Editor....4-11
Temporary Variable....4-8
TestMeasure....8-324
Text display (*MEASUREDISPT)....5-19
Time$....8-326
Timer....8-327
TotalJudge....8-328
TransformAngle....8-330
TransformArea....8-331
TransformDist....8-332
TransformLine....8-333
TransformXY....8-335
Troubleshooting....7-2
Try Catch End Try....8-337

U

UCase$....8-339
UdpNormal....9-50
UDP Normal communication....9-50
UdpPlcLink....9-58
UdpPlcLinkM....9-52
UdpPlcLinkY....9-55
Unit Calculation Macro processing item....1-3
UnitCount....8-340
UnitData....8-341
UnitData$....8-342
UnitData2....8-343
UnitInfo....8-345
UnitItemIdent$....8-347
UnitJudge....8-348
Unit Macro processing item....1-3
UnitNo....8-349
UnitTitle$....8-350
User Data....5-8
Ut....8-351

V

Val....8-352
Variable....4-6
VarList....8-353
VarPop....8-355
VarPush....8-358
VarSave....8-361

W

Wait....8-362
WritePlcMemory....8-363

X

XOR....8-365

OMRON Corporation **Industrial Automation Company**

Kyoto, JAPAN **Contact : www.ia.omron.com**

Regional Headquarters

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

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

OMRON ASIA PACIFIC PTE. LTD.
438B Alexandra Road, #08-01/02 Alexandra
Technopark, Singapore 119968
Tel: (65) 6835-3011 Fax: (65) 6835-3011

OMRON (CHINA) CO., LTD.
Room 2211, Bank of China Tower,
200 Yin Cheng Zhong Road,
PuDong New Area, Shanghai, 200120, China
Tel: (86) 21-6023-0333 Fax: (86) 21-5037-2388

Authorized Distributor:

©OMRON Corporation 2016-2026 All Rights Reserved.
In the interest of product improvement,
specifications are subject to change without notice.

Cat. No. Z367-E1-16 0926