F350 Visual Inspection System
OMRON Vision Language (OVL) V2

Reference Manual

Revised January 2001

OMRON Product References

All OMRON products are capitalized in this manual. The word “Unit” is also capitalized when it refers to an OMRON product, regardless of whether or not it appears in the proper name of the product.

Visual Aids

The following headings appear in the left column of the manual to help you locate different types of information.

Important Indicates information of importance that, if not heeded, could result in damage to the product, malfunction, or incorrect operation.

Note Indicates information of particular interest for efficient and convenient operation of the product.

1, 2, 3... 1. Indicates lists of one sort or another, such as procedures, checklists, etc.

TABLE OF CONTENTS

TABLE OF CONTENTS

About this Manual:

This manual describes the OMRON Vision Language (OVL) used with the F350 Visual Inspection System and includes the sections described below.

Please read this manual completely and be sure you understand the information provided before attempting to operate the F350 Visual Inspection System and use the OVL.

Section 1 provides an introduction to the OMRON Vision Language (OVL) and its use in the F350 Visual Inspection System. This section includes a table of related manuals and a description of OVL features.

Section 2 shows the basic and expanded system configuration, and explains basic operations such as starting up and exiting OVL. It also provides tables describing basic key operations.

Section 3 provides the basic syntax required before programming.

Section 4 provides a basic description of the F350 Visual Inspection System.

Section 5 provides detailed information on the commands and functions. Examples are also provided.

Section 6 provides the general-purpose subroutines used to simplify programming and reduce processing time.

Section 7 provides sample programs prepared mainly for measurement processing.

Section 8 provides an alphabetical list of error messages and their causes and remedies.

The **Appendices** provide lists of reserved words, induction functions, and characters and codes.

⚠WARNING Failure to read and understand the information provided in this manual may result in personal injury or death, damage to the product, or product failure. Please read each section in its entirety and be sure you understand the information provided in the section and related sections before attempting any of the procedures or operations given.

SECTION 1
Introduction

This section provides an introduction to the OMRON Vision Language (OVL) and its use in the F350 Visual Inspection System. This section includes a table of related manuals and a description of OVL features.

1-1 Overview

This manual describes the OMRON Vision Language (OVL) commands, functions, and programming methods required to operate the F350-L12E OVL Unit, which is used in an F350 Visual Inspection System. In an F350 Visual Inspection System, OVL serves as a specialized BASIC programming language. The F350-L12E OVL Unit can be used to carry out inspections that are optimal for specific environments and conditions. In order to use the F350 System safely and effectively, be sure to carefully read this and all other applicable manuals.

1-2 Applicable Manuals

The manuals applicable to the F350 Visual Inspection System are shown in the table below, according to the procedures used. There are three kinds of F350-series manuals:

- F350 Setup Menu Operation Manual: Included with the F350-C12E/C41E IMP Unit.
- F350 Application Software Operation Manual: Included with the F350-U□□□E Application Software.
- F350 OVL Reference Manual: Included with F350-L12E OVL Unit.

Procedure		Software	
		Application Menus	**OVL program**
System design	Consider the lighting environment, I/O devices, and so on, and arrange the system configuration. Design the system carefully, taking into account variations in conditions and the objects that are to be inspected.	F350-series catalog	
Assembly/Installation	Install the F350 Visual Inspection System by assembling the hardware and wiring the power supply and peripheral devices.	F350 Setup Menu Operation Manual	
Environmental settings	Start up the software and make the settings related to the F350 Visual Inspection System and the settings for starting, communicating with I/O devices, and so on.	Make the settings using the Setup Menu which is standard with F350-C12E/C41E IMP Unit. (Refer to the F350 Setup Menu Operation Manual.)	Mount the F350-L12E OVL Unit and program using OVL, a specialized BASIC programming language. (Refer to the F350 OVL Reference Manual.)
Inspection condition settings	Start up the software and make the settings related to inspection. Set the criteria for determining the inspection area and the acceptability of the inspected products.	Make the settings using the F350-U□□□E Application Menus. Do the actual testing according to the conditions that have been set. (Refer to the relevant F350 operation manual.)	Mount the F350-L12E OVL Unit and program using OVL, a specialized BASIC programming language. Do the actual testing according to the conditions that have been set. (Refer to the F350 OVL Reference Manual.)
Testing/Inspection	Do the actual testing according to the conditions that have been set. If adjustments are required, change the settings.		
Maintenance	Carry out periodic inspections. This is essential in order to maintain the F350 Visual Inspection System in optimum condition.	F350 Setup Menu Operation Manual	

1-3 Before Using this Manual

Intended Users

OVL is a programming language similar to standard BASIC, but with specialized commands and functions added for image processing. It is assumed that users of this manual will already be familiar with standard BASIC programming. Therefore this manual focuses mainly on operations and programming methods that are specific to OVL.

User Program Disclaimer

OVL is a programming language for making F350 operational settings. The user can create programs in response to particular inspection environments and conditions. OMRON accepts no responsibility, however, for any problems or damage that may occur as a result of any program created by the user.

1-4 Features

High Compatibility with BASIC

The programming language specifications are the same as those for BASIC on a personal computer, so there is no need to acquire a new language.

Wide Selection of Image Processing Commands

OVL adds commands and functions to standard BASIC which activate F350 operations. Using these commands and functions opens up the F350's fast and powerful capabilities.

High-speed Image Processing

Background processing of operations such as searches during arithmetic processing of numeric expressions is made possible by applying pipeline processing concepts such as sequence-type processing, thereby enabling high-speed real-time image processing.

Program Backups

Programs that have been created are backed up by the IMP Unit. The desired actions can be immediately carried out as soon as the power is turned on.

Multiple programs can be managed by using a memory card.

Help Function Support

Online help is provided, so there is no need to search through the manual when the format of a command or function is not understood. An explanation of the format and a list of commands and functions can be displayed on the screen.

Structured Programming Support

The use of GOTO in BASIC makes it difficult to follow the structure of a program, and this in turn makes maintenance quite difficult. OVL supports structured programs such as the one shown below, thereby making it easier to write programs that are simple to read and modify.

```
CALL
DEF FN ~ END DEF
DO ~ LOOP
EXIT
IF...THEN ~ ELSE ~ END IF
SELECT...CASE ~ CASE ~ END SELECT
SUB ~ END SUB
WHILE ~ WEND
```

Subroutine Support

Often-used subroutines are stored in advance in the F350, facilitating the processing for creating menus, setting areas, and so on.

Long Integer Support

The OVL now supports long integers, allowing a numeric range of -2,147,483,648 to 2,147,483,647 to be handled as integers. This surpasses the previous integer range of -32,768 to 32,767 that could be handled by BASIC, and manages the total F350 screen area (512 * 484 = 247,808).

Enhanced Editing Operations

The following editing operations are provided for easy program editing.

- Up and down list scrolling
- Cutting and pasting of character strings
- Searching and replacing of character strings

Error Messages in Either English or Japanese

Error messages can be displayed in either English or Japanese.

SECTION 2
System Setup and Basic Operations

This section shows the basic and expanded system configuration, and explains basic operations such as starting up and exiting OVL. It also provides tables describing basic key operations.

2-1 System Configuration

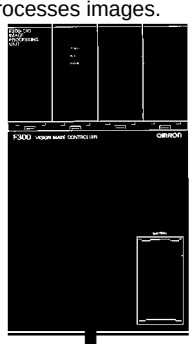
Check to be sure that the Units shown in the basic system configuration are included, as well as any peripheral devices that may be required.

Basic System Configuration (Must be used.)

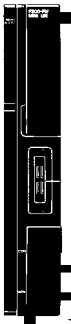
F300-P2 (100 to 120 VAC)
★ F300-P2E (200 to 240 VAC)
Power Supply Unit
Supplies power to the entire system.

(Note: The F200-P Power Supply Unit cannot be mounted.)

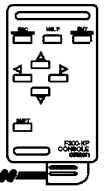
★F350-C12E
F350-C41E
IMP Unit
Processes images.



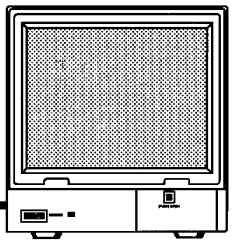
★F300-FM2 MMI Unit
Provides Memory Card functions and menu operations via a Video Monitor and Console.



★F300-KP Console



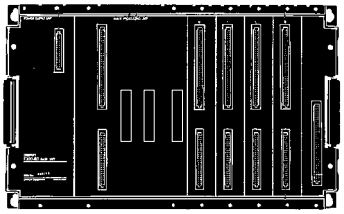
F300-M09 Video Monitor (100 VAC)



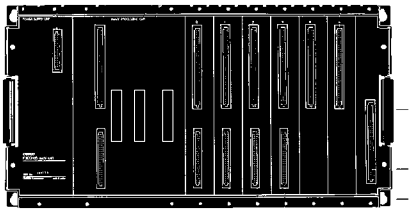
★F309-VM Monitor Cable

Standard Base Units
Available with either 3 slots or 5 slots. Select a model according to the number of Interface Units required.

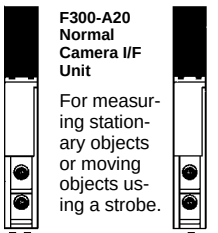
★F300-B32 (3 slots)



★F300-B52 (5 slots)



Camera I/F Units (Use at least one.)



F300-A20 Normal Camera I/F Unit
For measuring stationary objects or moving objects using a strobe.

★ F300-A22S Normal Simultaneously Camera I/F Unit
For measuring stationary objects or moving objects using a strobe, and for measuring two places simultaneously using two cameras.

F300-A20R Shutter Camera I/F Unit
For measuring objects moving at high speed.

★ F300-A22RS Shutter Simultaneously Camera I/F Unit
For measuring objects moving at high speed, and for measuring two places simultaneously using two cameras.

F300-A23RS Frame Shutter Camera I/F Unit
This I/F Unit is used exclusively with the F300-S4R.
For measuring objects moving at high speed, and for measuring two places simultaneously using two cameras.

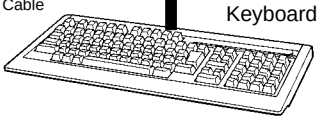
★F350-L12E OVL Unit
For writing and editing OVL programs.

Note: The F300-L12E OVL Unit cannot be used. It does not run F350 OVL.

★ F309-VSR2 Camera Cable

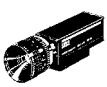
★ F309-VSR2 Camera Cable

★ F309-VSR2 Camera Cable



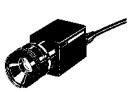
F300-K Keyboard

F200-S Camera



★ F300-S Camera

★ F300-S2R Shutter Camera



F300-S3DR Shutter Camera

F300-S4R Frame Shutter Camera

Cameras (Connect up to 2 Cameras per Unit.)

Note: Different types of cameras and Camera I/F Units cannot be used together. Using them together can cause a faulty image to be captured.

•Lens •Light

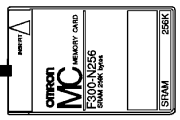
Some of the products listed may not be available overseas. Please contact your nearest OMRON sales office by referring to the addresses provided at the back of this manual.

Important An "★" before the model number indicates conformance to EC Directives. Use only these Units when constructing a system that must conform to EC Directives.
Refer to Appendix A in the Setup Manual for a complete list of the Units that conform to EC Directives.

F350-U□□□E Software
Select the proper application software for the desired application.

Peripheral Devices

★F300-N256/N512/N2M Memory Card



Used for storing scene data, system data, and output data (e.g., measurement values, judgement results, etc.).

I/F Units (Use one that matches the peripheral devices connected.)

★F300-E2 RS-232C I/F Unit
For saving scene and system data and carrying out menu operations via RS-232C interface.

★F300-D2 Terminal Block Unit
For inputting measurement instructions and outputting judgement results via a terminal block.

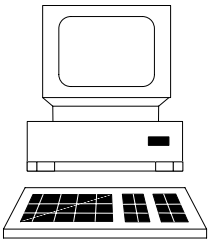
★F300-DC2 Parallel I/O Unit
For inputting measurement instructions and outputting measurement values and judgement results via parallel I/O.

F300-FS Strobe I/F Unit
For flashing the strobe while taking images.

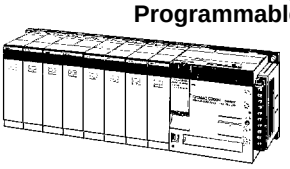
★F300-G Dummy Unit
For inserting into empty slots to protect and strengthen connectors.

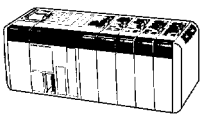
★F309-VR
RS-232C Cable

IBM PC/AT or Compatible

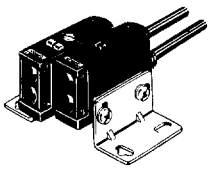


C200HS or CQM1 Programmable Controller



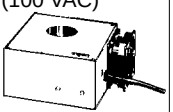


Synchronization Sensor



F309-VFS
Strobe Cable

3Z4S-LT DSX-240 Strobe Device
(100 VAC)



2-2 Starting Up and Exiting the OVL System

2-2-1 Starting Up

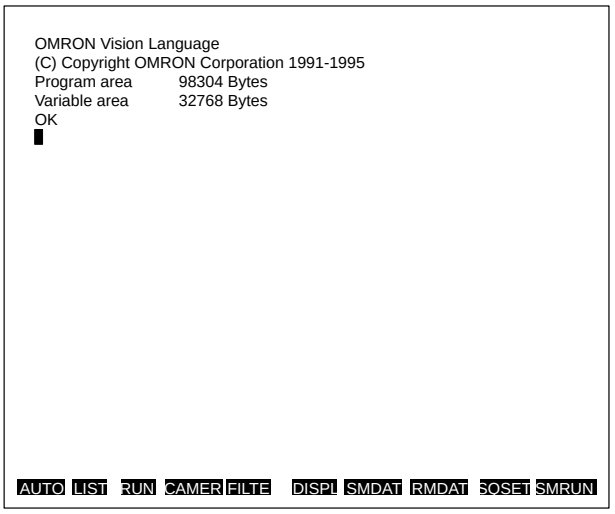
Before starting up the OVL System, check to be sure that the F350-L12E OVL Unit is mounted. If it is not mounted, OVL programs cannot be created or edited.

If the Setup Menu or Application Menu is already started, first check to be sure that “K:Environment/M:Initial mode” is set to “OVL prompt,” and then turn off the power supply switch. For details refer to *5-2-1 Designating Startup Operations: M.Initial mode* in the *F350 Setup Menu Operation Manual*.

Procedure

1, 2, 3...

1. Turn on the power to the Video Monitor.
2. Turn on the power to the F350. The OVL System will start up and the prompt will be displayed. It will then be possible to start programming.



2-2-2 Exiting

Before exiting the OVL System, stop the OVL program by pressing STOP.

Procedure

1, 2, 3...

1. Turn off the power to the F350. (The program will be retained even when the power is turned off.
2. Turn on the power to the Video Monitor.

Note The Setup Menu and Application Menu cannot be started from the OVL Menu.

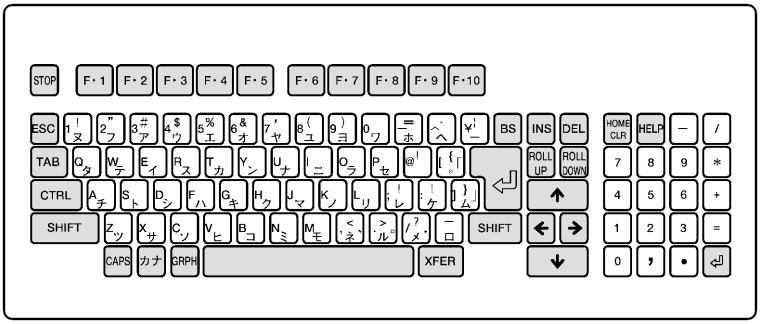
To start the Setup Menu, press “ENT” and then turn on the power. Refer to *3-1 Starting the Setup Menu* in the *F350 Setup Menu Operation Manual*.

To start the Application Menu, start the Setup Menu and then install and start the Application Menu. For details, refer to *5-1-1 Installing the Application Program: M.Application program* in the *F350 Setup Menu Operation Manual*.

2-3 Basic Operations

Key Arrangement

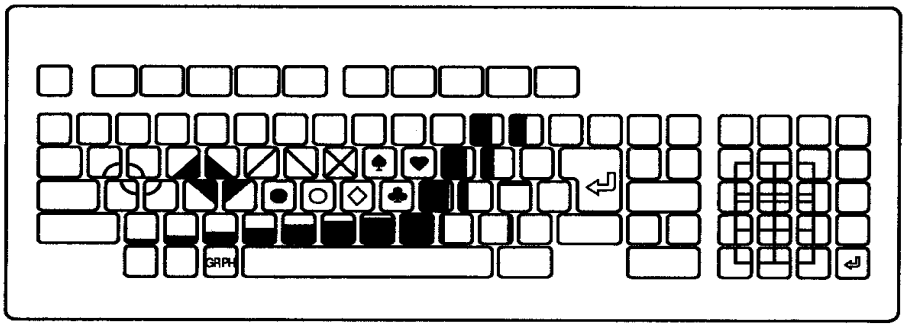
The F300-K Keyboard has keys for inputting alphanumeric characters and symbols, in addition to keys with specialized functions. In the following illustration, the shaded keys are the ones with specialized functions.



Key Combinations

A single character key can input several kinds of characters by being used in combination with various specialized keys.

Specialized Keys

Key	Operation
HELP	Displays the last correctly input line. Also displays the command and location when an error occurs during program execution.
STOP	Forcibly stops program execution.
BS	Deletes one character to the left of the cursor.
TAB	Moves the cursor eight digits to the right. Used to line up the left sides of a column.
CAPS	Locks the characters into upper-case mode. (A red indicator lights.) Pressing the key again clears the function so that lower case can be input again. In addition, pressing this key in conjunction with the Shift Key reverses the relationship between upper and lower case.
GRPH	Graphic characters can be entered in graphic character mode by pressing character keys while holding this key down. (Use the CONSOLE command to go into graphic character mode.)
Graphic Character Key Arrangement 	
CTRL	Special codes can be input by combining this key with other keys. (Refer to the following table for control key combinations.)
SHIFT	When a character key is pressed while this key is held down, either that character will be entered in upper case or the symbol shown above the character on the key will be entered.
PgDn, PgUp	Scrolls the text screen up or down.
HOME CLR	Clears the text from the screen and moves the cursor to the home position (i.e., to the upper left corner of the screen).
Return	Ends a line and moves the cursor to the beginning of the next line.
INS	Puts the program into Insert Mode. To return to Overwrite Mode, either press this key again or press a direction key followed by the Return Key.
DEL	Deletes one character to the right of the cursor.
→←↑↓	These direction keys move the cursor in the respective directions.
F1 to F10	These are function keys. The KEY command can be used to set a character string to a particular function key, and the function key can subsequently be used as a shortcut for entering that character string. The following character strings are the default settings. F1 AUTO F2 LIST F3 RUN CR F4 CAMERA F5 FILTERIN F6 DISPLAY F7 SMDATA F8 RMDATA F9 SQSET F10 SMRUN

Control Key

Combining the Control Key with other keys makes possible the operations shown in the following table. They can be used for creating and editing programs.

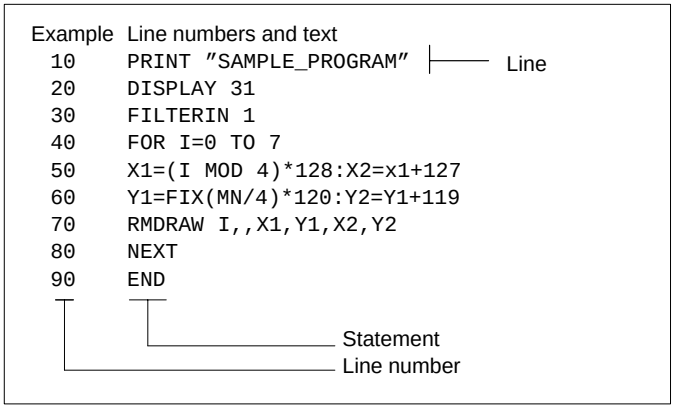
Key	Operation
CTRL+A	Performs the same function as HELP. It displays the last correctly input line and also displays the command and location when an error occurs during program execution.
CTRL+B	Moves the cursor to the beginning of the word to the left. A “word” is a text string separated by a symbol such as a space, comma, or colon.
CTRL+C	Performs the same function as STOP. It forcibly stops program execution.
CTRL+D	Deletes from the cursor position to the end of that word. A “word” is a text string separated by a symbol such as a space, comma, or colon.
CTRL+E	Deletes from the cursor position to the end of that line.
CTRL+F	Moves the cursor to the beginning of the word to the right. A “word” is a text string separated by a symbol such as a space, comma, or colon.
CTRL+G	Makes a beep sound.
CTRL+H	Performs the same function as BS. It deletes one character to the left of the cursor.
CTRL+I	Performs the same function as TAB. It moves the cursor eight digits to the right, and is used to line up the left sides of a column.
CTRL+J	Performs a line feed (LF).
CTRL+K	Performs the same function as SHIFT+HOME CLR. It moves the cursor to the home position (i.e., to the upper left corner of the screen) without changing the screen display.
CTRL+L	Performs the same function as HOME CLR. It clears the text from the screen and moves the cursor to the home position (i.e., to the upper left corner of the screen).
CTRL+M	Performs the same function as the Return Key (CR). It ends a line and moves the cursor to the beginning of the next line.
CTRL+P	Outputs (i.e., pastes) to the screen the contents that have previously been written to the paste buffer by CTRL+U.
CTRL+R	Performs the same function as INS. It puts the program into Insert Mode. To return to Overwrite Mode, either press this key again or press a direction key followed by the Return Key.
CTRL+S	Temporarily stops program and LIST command execution. To restart execution, enter anything other than STOP, SHIFT, GRAPH, or CTRL+C.
CTRL+U	Writes to the paste buffer the line where the cursor is located, and clears it from the screen.
CTRL+V	Deletes all lines onwards from the line where the cursor is located.
CTRL+X	Moves the cursor to the end of the line.
CTRL+Y	Marks the line (i.e., the line number) where the cursor is located.
CTRL+Z	Displays on the screen the lines (i.e., the line numbers) that have been marked by CTRL+Y. The display method is the same as for the EDIT command.

SECTION 3
Syntax

In order to use OVL to carry out an inspection process, it is necessary to create a program incorporating several commands. This section provides the basic syntax required before programming.

3-1 Lines

An OVL program is composed of lines. Each line consists of a line number and a statement.



Each line, including the line number, spaces, and the instruction may be up to 255 bytes in length.

It is possible to include multiple statements in a line. Multiple statements in a single line are delimited by colons (:).

3-1-1 Line Numbers

The line numbers are positioned at the start of a line and are integers applied in ascending order between 1 and 65535.

Lines assigned with line numbers are stored as part of the program when the Return Key is pressed. A line with no line number is not stored as part of the program but is executed immediately when the Return Key is pressed.

The program is executed in the order of the line numbers, except where branching occurs.

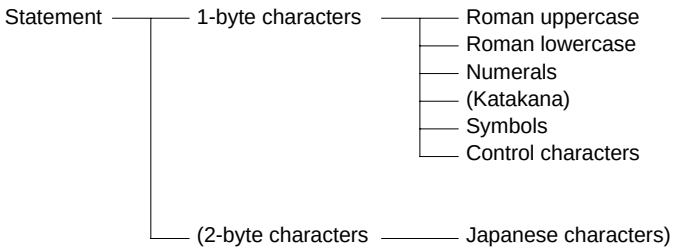
3-1-2 Statements

A statement is the smallest processing unit.

Statements include executable statements which declare and execute commands and functions, non-executable statements which provide program comments, and labels which define jump destinations.

3-2 Characters and Symbols

The following characters and symbols can be used with OVL programming.



Only lowercase characters defined in a character constant or character variable inside quotes (" ") are handled as lowercase characters. The system automatically converts other lowercase characters to uppercase characters.

Symbols

Period (.)

The period indicates the current line number (the last line number executed). It can replace the line number in the following commands: AUTO, EDIT, LIST, LLIST, RENUM.

Minus (–)

The minus specifies a range of lines.

Colon (:)

The colon delimits one statement from another.

Comma (,)

The comma delimits parameters in a series.

Semicolon (;)

The semicolon delimits parameters in an output statement.

Apostrophe (')

Identical in meaning to a REM statement.

Question Mark (?)

Simplifies input of the word “PRINT.” Immediately after input, the question mark is converted to “PRINT.”

Asterisk (*)

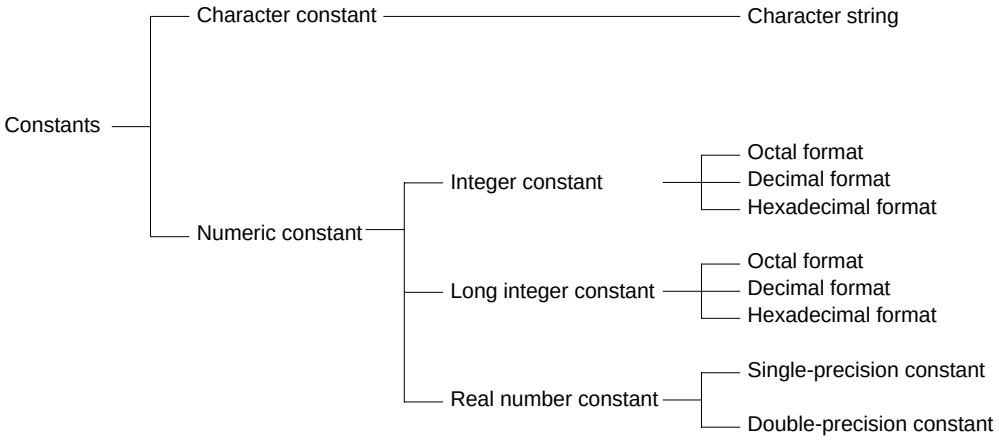
The asterisk indicates the start of a label name.

Space ( )

A space must be inserted between a command and the following parameter. Other spaces may be inserted anywhere except inside command names, variable names, or numeric values.

3-3 Constants

Values or character strings declared directly in the program are known as constants. A character constant is declared differently from a numeric constant.



3-3-1 Character Constants

A character constant is a character string enclosed in double quotation marks (") and can contain up to 255 characters.

The CHR$ function must be used to include double quotations inside a character string, as shown in one of the examples below.

A character string of zero length is known as a “null string.”

Character Strings

"1234567890" . . . . Arithmetic operations cannot be carried out on numbers in this form.

"NEW PLAN" . . . . This character string contains 8 characters, 7 letters and one space.

CHR$(34) . . . . . This character string represents a single “double quotation” (") character.

"" . . . . . This is a null string.

3-3-2 Numeric Constants

A numeric constant directly declares a value for operation by the arithmetic operators.

The plus (+) or minus (–) sign can be included before the value to indicate a positive or negative value, but the plus sign may be omitted.

Numeric constants can be declared as integers or real numbers.

Integer Constants

Decimal Format

Integers must lie in the range –32768 to +32767.

Adding a % sign to a real number in the above range rounds off the value to an integer constant.

Examples of decimal format integer constants:

32767

–123

+5%

321.01% (This is identical to 321.)

Octal Format

An octal integer constant is declared as an octal value (digits 0 to 7) preceded by the prefix &O. However, the “O” can be omitted if required.

Octal integer constants can be specified between &O0 and &O177777.

Examples of octal format integer constants:

&12345

&O77777

Hexadecimal Format

A hexadecimal integer constant is declared as a hexadecimal value (digits 0 to 9, A to F) preceded by the prefix &H.

Hexadecimal integer constants can be specified between &H0 and &HFFFF.

Examples of hexadecimal format integer constants:

&H100

&HFCA0

Long Integer Constants

Decimal Format

Integers must lie in the range –2147483648 to +2147483647.

Adding a & sign to a real number in the above range rounds off the value to a long integer constant.

Examples of decimal format long integer constants:

9500000&

–365432&

100.12& (This is identical to 100.)

Octal Format

An octal integer constant is declared as an octal value (digits 0 to 7) preceded by the prefix &O and followed by the type declarator (&).

Octal integer constants can be specified between &O0& and &O37777777777&.

Examples of octal format long integer constants:

&O12345&

&O112377777&

Hexadecimal Format

A hexadecimal integer constant is declared as a hexadecimal value (digits 0 to 9, A to F) preceded by the prefix &H and followed by the type declarator (&).

Hexadecimal integer constants can be specified between &H0& and &HFFFFFFFF&.

Examples of hexadecimal format long integer constants:

&H100&

&HFCA00&

Note Values input in octal and hexadecimal format are printed out in decimal format.

Real Number Constants

Single-precision Format

Real number data between –1.70141E+38 to +1.70141E+38 to 6 significant digits.

The following numbers are declared as single-precision real numbers:

- Values with a decimal point but not exceeding 6 significant digits
- Values with the single-precision declarator (!)
- Values in the single-precision E exponent format.

Examples of single-precision constants:

108.12

3.14!

–8.76E–6

Double-precision Format

Real number data between –1.701411834604692D+38 to +1.701411834604692D+38 to 16 significant digits.

The following numbers are declared as double-precision real numbers:

- Real numbers with 7 or more digits
- Values with the double-precision declarator (#)
- Values in the double-precision D exponent format.

Examples of double-precision constants:

108.123456

3.14#

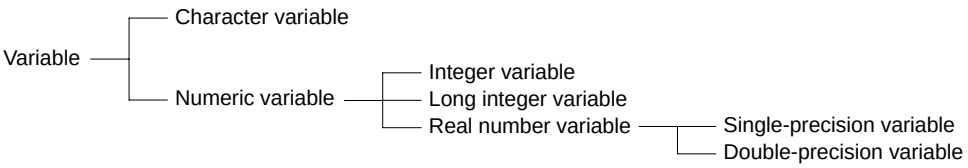
–8.76D–6

3-4 Variables

Named areas in the program in which values or character strings are stored are known as variables.

Before a value is allocated, a numeric variable contains 0 and a character variable contains a null string (” ”).

The user can declare both the name and type of a variable.



3-4-1 Type Declarators

The type of variable is declared with a declarator suffix to the variable name.

Variables with an identical variable name but different declarators are treated as separate variables.

Types of Declarator:

$ . . . . Character

% . . . . Integer

& . . . . Long integer

! . . . . . Single-precision

. . . . Double-precision

The type of variable can also be declared with a type statement. However, the type declared with the type declarator takes priority over the type statement.

Types of Type Statement:

`DEFSTR .....` Declares a character variable.

`DEFINT .....` Declares an integer variable.

`DEFLNG .....` Declares a long integer variable.

`DEFSNG .....` Declares a single-precision variable.

`DEFDBL .....` Declares a double-precision variable.

Variables not declared with a type declarator or type statement are treated as single-precision numeric variables.

Examples of type statements:

```
DEFSTR NAME$
DEFDBL AX#, AY#, AZ#
```

The $ and # can be omitted from these statements.

3-4-2 Variable Names

Variable names are limited to 40 characters, including the type declarator. Alphanumeric characters, the period, and type declarator are valid in a variable name.

The name must start with an alphabetic character. Uppercase and lowercase characters are not differentiated. The type declarator must be added to the end of the variable name.

Symbols cannot be used in variable names.

Reserved words cannot be used as variable names. However, variable names may contain reserved words.

Variable names cannot begin with the letters “FN.”

3-4-3 Reserved Words

Reserved words are character strings defined in the system, such as commands, functions, and operators. Users cannot use reserved words as variable names.

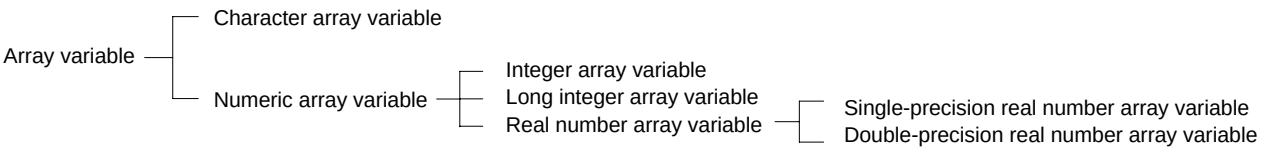
The reserved words are listed in *Appendix B, Reserved Words*.

3-4-4 Array Variables

A variable with a single variable name storing multiple values is known as an array variable.

Array variables storing character strings are classified as character array variables and array variables storing numeric values are classified as numeric array variables.

Numeric array variables are further subdivided according to the type of numeric values they store into integer array variables, long integer array variables, single-precision real number array variables, and double-precision real number array variables.



The dimension of the array variable and the subscript range are declared with the DIM statement.

It is unnecessary to use the DIM statement to declare subscripts up to 10.

Example:

```
DIM A (10, 10, 10) ..... may be omitted.
```

The dimensions of the array must be declared in a single program line (255 characters). The subscripts are limited by the amount of memory. Consequently, 4-dimensional, 5-dimensional, and 6-dimensional arrays may be declared but the number of elements are restricted.

Example:

```
DIM A(10) . ..... 1-dimensional array,
                  number of elements = 11
DIM TA(10, 50) .. 2-dimensional array,
                  number of elements = 11 x 51 = 561
DIM TTA$(2, 5, 3) 3-dimensional array,
                  number of elements = 3 x 6 x 4 = 72
```

Subscripts start from zero, so that the action number of elements is 1 plus the subscript.

3-5 Type Conversion

If necessary, the type of numeric data can be converted under the conditions described below.

It is not possible to convert between character and numeric data.

Condition 1

When data is assigned to a variable of a different type, the data is converted to the type declared with the variable type declarator.

Example of assigning a value to a variable of a different type:

```
10 A%=1.234 ..
```
Assigns the single-precision real number 1.234 to the integer variable A% (assigned as 1).

```
20 PRINT A% ..
```
Displays the value stored in the integer variable A% on the screen.

Result:

```
1 . ..... The integer 1 is displayed.
```

Condition 2

Before operations are carried out on values with different accuracies, all values are first converted to the accuracy of the highest accuracy value.

Example of operations on values of different accuracy:

```
10 A%=10%/3%
```

```
20 PRINT A% ..
```
The result of the operation on integers is displayed as an integer.

```
30 B!=10%/3!
```

```
40 PRINT B! ..
```
The integer 10% is converted to a single-precision real number, the operation is carried out on single-precision values, and the result displayed as a single-precision value.

```
50 C#=10%/3#
```

```
60 PRINT C# ..
```
The integer 10% is converted to a double-precision real number, the operation is carried out on double-precision values, and the result displayed as a double-precision value.

Results:

```
3 . ..... Result of execution of line 20 is an integer.
3.33333 . ..... Result of execution of line 40 is a single-precision real number.
3.333333333333333 ... Result of execution of line 60 is a double-precision real number
```

Condition 3

Before logical operations all values are converted into integers and the results are also integers.

Example of logical operations:

```
10 A=1.234  ..  Assigns the value 1.234 to the numeric variable A.
20 B=NOT A  ..  Converts the value 1.234 stored in the numeric variable A to the integer 1, executes the NOT logical operation, and assigns the result (–2) to the numeric variable B.
30 PRINT B, A   Displays the values stored in the numeric variables B and A on the screen.
```

Result:

```
–2 1.234 . .....  Result of execution of line 30 is displayed on the screen.
```

Condition 4

Real numbers are rounded off to an integer. An error occurs if the converted value lies outside integer range.

Example of conversion to an integer:

```
10 A%=1.45  ..  Assigns the value 1.45 rounded of to 1 to the integer variable A%.
20 B%=1.65  ..  Assigns the value 1.65 rounded of to 2 to the integer variable B%.
30 PRINT A%,B%  Displays the values stored in the integer variables A% and B% on the screen.
```

Result:

```
1 2 . ...........  Result of execution of line 30 is displayed on the screen.
```

Condition 5

When a double-precision variable is assigned to a single-precision variable, the variable is rounded off to 6 significant digits, with subsequent digits rounded off.

Example of conversion to a single-precision, real number variable:

```
10 A!=3.14159265358 .  Substitutes the effective six digits, 3.14159, for a single-precision, real number variable A!.
20 PRINT A! . .........  The value to be stored as the single-precision, real number variable A! will be displayed (the seventh digit will be rounded to the nearest whole number).
```

Result:

```
3.14159 . ...  Result of execution of line 20 is displayed on the screen.
```

Note

OVL does not convert between numeric and character data, except for special applications such as random access file I/O and conversion of number declaration character strings to numeric data.

Special functions are used to convert between numeric and character data, where required.

Random access file I/O:

CVI/CVL/CVS/CVD functions
MKI$/MKL$/MKS$/MKD$ functions

Conversion of a number declaration character string to numeric data:

VAL function

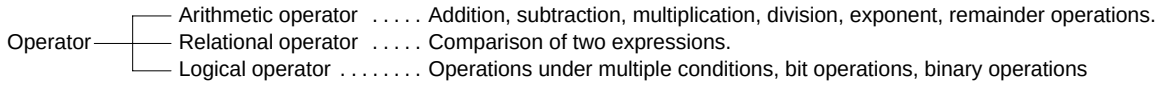
Conversion of numeric data to a character string:

STR$ function

3-6 Operations and Expressions

3-6-1 Operators

Three types of operator are used in OVL: arithmetic operators, relational operators, and logical operators.



Arithmetic Operators

Arithmetic operators link numeric constants or variables to carry out addition, subtraction, multiplication, division, exponent, and remainder operations.

Arithmetic operator	Description	Declaration example	Mathematical notation
+	Addition	A + B	A + B
–	Subtraction	A – B	A – B
*	Multiplication	A * B	A x B or AB
/	Real number division	A / B	A ÷ B or A/B
¥	Integer division	A ¥ B	[A/B] [ ] indicates Gauss' notation
^	Exponent operation	A ^ B	A^B
MOD	Remainder calculation	A MOD B	A – [A/B] x B [ ] indicates Gauss' notation

If either the divisor or dividend for integer division is a real number, the real number is rounded off to an integer before division. If the quotient contains decimal places, these are dropped.

Integer Division

Example of integer division:

```
123.4¥67.89      → 123¥68 →  1.808          →  1
Decimals rounded off → Division → Decimals dropped → Results
```

The decimal places are rounded off before the remainder calculation is carried out on real numbers. The result is the remainder from the integer division.

Remainder Calculation

Example of integer division:

Example of remainder calculation:

```
13.3 MOD 4 = 1          Remainder of 13 divided by 4.
25.68 MOD 6.99 = 5      Remainder of 26 divided by 7.
```

Digit Overflow

Digit overflow occurs if the result of a calculation exceeds the range of the value type. If the digits overflow, an error is output and the calculation continues with the maximum value the computer can handle.

Example of digit overflow:

```
A% = 32760 + 10
```

An error is output and the A% value becomes 32767.

If division by 0 is carried out during operation, an error is output and the calculation continues with the maximum value the computer can handle. The same applies if the exponent operation is carried out on 0 with a negative exponent.

Relational Operators

Relational operators compare two numeric data or 2 character data.

True (–1) is returned if the result of the compared data is the same or False (0) is returned if the result of the compared data is different.

Relational operator	Description	Declaration example
=	Equals	A = B
< >, > <	Not equals	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

Relational operators are used inside the IF statement to control the flow of program execution.

Examples of relational operators inside the IF statement:

`IF A<>B THEN 1000 .....` Jump to line 1000 if A is not equal to B (A < > B is true (–1)).

`IF A$ = "Y" THEN *PROCESS1` Jump to label *PROCESS1 if A$ is equal to " Y" (A$ = " Y" is true (–1)).

Logical Operators

Logical operators are used to investigate multiple conditions and carry out bit operations or binary (Boolean) operations on a specified value.

After numerals are converted to integers, the results are provided in bit units of 0 and 1. An error occurs if the allowable range is exceeded during conversion.

Logical operator	Description	Equivalence
NOT	Negation	NOT A
AND	Logical product	A AND B
OR	Logical sum	A OR B
XOR	Exclusive OR	A XOR B

Refer to page 23 for details of the results of logical operations.

3-6-2 Expressions

In an OVL program, an "expression" refers to constants, variables, functions, numeric constants and variables linked with arithmetic operators, and character constants and variables linked with plus signs (+).

Examples of expressions:

Numeric expressions	Character expressions	Function expressions	Logical expressions
A=20*15/3	A$="OVL"+"BASIC"	A=B	A AND B
S=P*D*D/4	A$=B$+C$	A<B	A OR B
2.7145	"OVL"	A<>B	A XOR B
A	A$		A<B AND A>C
SIN (X)	CHR$ (31)		A=B OR A<>C

Numeric Expressions

An expression returning a numeric value is known as a numeric expression.

Numeric expressions can be numeric constants, numeric variables, or functions returning numeric values linked by arithmetic or logical operators.

Multiple expressions contained within parentheses ( ) can be linked together.

Character Expressions

An expression returning a character string is known as a character expression.

Character expressions can be character constants, character variables, or functions returning character strings linked by plus signs.

Multiple expressions contained within parentheses ( ) can be linked together.

Relational Expressions

A pair of numeric expressions linked by a relational operator is known as a relational expression.

Relational expressions can be numeric constants, numeric variables, or functions returning numeric values linked by relational operators.

Logical Expressions

Multiple relational expressions linked by logical operators are known as a logical expression.

A logical expression is made up of several relational expressions linked by logical operators to execute bit or binary operations or to evaluate multiple conditions.

Logical expressions have a numeric value and may be used in any position as a numeric expression.

A logical expression is True if the result of the expression is other than 0 or False if the result is 0.

Logical expressions may include six types of logical operator: NOT, OR, AND, and XOR.

Functions

A function executes predefined operations on specified values (called arguments) and returns a numeric value or character string as the result. Although functions are handled as expressions, unlike numeric, character, relational, and logical expressions the function itself holds the result of the operations.

Some functions are automatically assigned values by the system. These functions are known as system variables.

System variables require no arguments. Values are assigned to some system variables under special system conditions, such as when an error or interrupt occurs, but some other system variables always hold values, such as the time or date.

OVL offers user-defined functions that the BASIC user can define as required. User-defined functions are handled inside the program in the same way as the system variables.

Elementary functions (such as the SIN function) match the precision of the argument. The function becomes double precision if the argument is a double-precision value or single-precision if the argument is an integer or a single-precision value.

3-6-3 Priority of Operations

Operations are executed in the order of priority shown below. Low numbers take priority over higher numbers. Operations with the same number are executed in the order in which they appear.

1, 2, 3...

1. Expressions enclosed in parenthesis
2. Functions
3. ^ (Exponents)
4. – (Minus signs: not preceded by a value or numeric expression)
5. *, / (multiplication and real number division)
6. ¥ (integer division)
7. MOD (remainder calculation)
8. +, – (addition, subtraction)
9. Relational operators (<, >, =, and combinations)
10. NOT
11. AND
12. OR
13. XOR

Example:

```
A% = 2 + 8 MOD 5 – 3
```

The above expression is interpreted as 2 + (8 MOD 5) – 3, so that A% = 2. Although the parentheses are not strictly necessary in this example, it is normal to include them to prevent confusion when reading the program.

3-7 Interrupts

OVL supports the interrupts listed below.

Stop Key	(ON STOP GOSUB)
Help Key	(ON HELP GOSUB)
Real-time timer	(ON TIME$ GOSUB)
Function Key	(ON KEY GOSUB)
RS-232C circuit	(ON COM GOSUB)
Processing error	(ON ERROR GOTO)
STEP signal	(ON INTR GOSUB)
Console Key	(ON CONKEY GOSUB)
Sequence measurement	(ON SQMEAS GOSUB)

3-8 Labels

Labels can be used instead of line numbers to control jump destinations in a program. Precede label names by an asterisk (*).

Define the name after the asterisk in alphabetic characters and the period character (.). Uppercase and lowercase characters are not differentiated.

Reserved words cannot be used as label names. However, label names may contain reserved words.

The length label name is restricted only by the number of characters in the program line (255 characters).

Label names must be positioned at the start of the program line.

Example:

```
100 IF A<>B GOTO *UNEQUAL
110 ....
120 ....
200 *UNEQUAL
210 ....
```

*UNEQUAL is interpreted in the GOTO statement as identical to 200. In this case, the program jumps to line 200 if A ≠ B.

3-9 Reserved Words

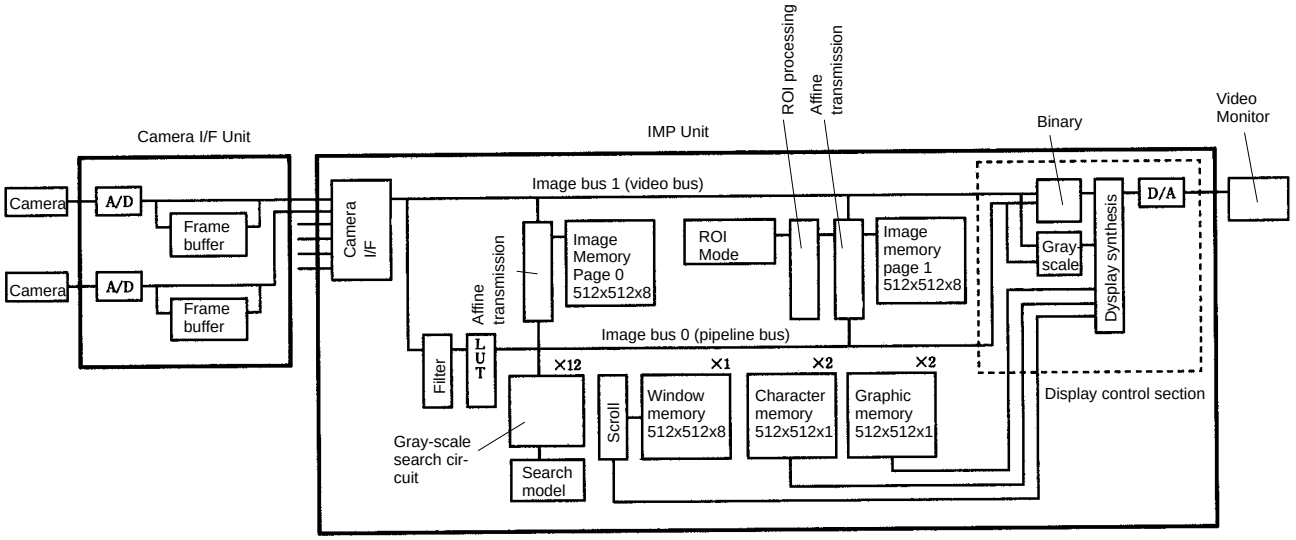
The character strings that have been defined in advance on the system side including commands, functions, and operators are called the ”reserved words.” These variables cannot be used. Refer to *Appendix A Reserved Words*.

SECTION 4
Basics of the F350 Visual Inspection System

This section provides a basic description of the F350 Visual Inspection System.

4-1 Hardware Configuration

It is essential to understand the hardware configuration of the F350 Visual Inspection System to make the best use of the F350 OVL Unit. Shown below is the hardware configuration of the F350-C12E/C41E.



Camera I/F Unit

The Camera I/F Unit performs A/D conversion on the video signals input via connected cameras and sends digitalized signals to the IMP Unit.

Some Camera I/F Units contain frame buffers. Those provided with frame buffers can store images in their memories.

Frame buffers are used for the following applications:

- To store in the memory images of moving objects at given times.
- To store in the memories all images from multiple cameras in synchronization with the STEP signals.
- To use buffers as a temporary memory area in case images stored in the IMP Unit are insufficient.

For taking images into the frame buffers in the Camera I/F Unit, use the FLASH, MEASURE, SMGRUN, SMRUN, or SQRUN command.

Use the CAMMODE command to determine whether the images output from the Camera I/F Unit should be sent directly to the IMP Unit or through the frame buffers.

Model	Name	Number of frame buffers
F300-A20	Normal Camera I/F Unit	0
F300-A22S	Normal Simultaneously Camera I/F Unit	2
F300-A20R	Shutter Camera I/F Unit	1
F300-A22RS	Shutter Simultaneously Camera I/F Unit	2
F300-A23RS	Frame Shutter Camera I/F Unit	2

Camera I/F

One camera can be selected from a maximum of 8 cameras and its images can be sent to the IMP Unit.

Camera selection can be made using the CAMERA command.

Although an externally synchronized camera requires no time for camera switching, an internally synchronized camera requires numerous milliseconds from the moment the camera is selected until stable, synchronized camera images are obtained.

The IMP Unit cannot simultaneously process signals from multiple cameras.

Image Bus

The IMP Unit contains two kinds of buses, namely, an image bus 0 (pipeline bus) and an image bus 1 (video bus).

8 bits (256 gradation) images pass through each bus at the video rate.

After being filtered through the filter and LUT, the images that pass through the image bus 0 (pipeline bus) are processed in real-time.

The images that pass through the image bus 1 (video bus) are mainly used for display.

The image bus 1 (video bus) allows a flow of input images from the camera and either the images from the image memory page 0 or image memory page 1. Use the FILTERIN command to designate which image to pass.

The images flowing into the image bus 1 (video bus) pass through the filter and LUT where the images are filtered and then flow to the image bus 0 (pipeline bus).

Filter

The filter is used to perform real-time filtering on the images it receives.

The filtering of images are effective for performing highly accurate and stable inspections.

The filter is treated with a 3 x 3 mask.

Filtered images can be easily selected using the FILTSEL command.

Fine filtering setting can be selected using the FILTER and FILTDATA commands.

LUT

The LUT is designed to change the input image gradation from 512 (–256 to 255) to 256 gradation.

Although it is possible to generate binary images using the LUT, the F350 System is designed to perform gray processing and normally doesn't require the LUT.

Image Memory (Frame Type)

The F350-C12E/C41E has two built-in image memory cards with the capacity of 512 pixels x 512 pixels x 8 bits.

The image memories are useful for filtering images more than one time or for temporarily storing images.

Images can be input to the image memory from the image bus 0 (pipeline bus) or image bus 1 (video bus).

Graphic patterns can be drawn on the image memory using the graphic command.

Data on the image memory is output to the image bus 1 (video bus). The data cannot be output to the image bus 0 (pipeline bus).

When the data on the image memory is output to the image bus 1 (video bus), search processing or binary raster processing can be performed on the output data.

The output data can be scrolled or zoomed in real-time using the affine transformation.

Gray-scale Search

Searching and locating a registered model in picture images is called a search processing.

Object positions can be found at high speed using the search processing of the F350 System.

For the images flowing through the image bus 0 (pipeline), 12 models can be searched at 16.7 ms or 33.3 ms in real-time. No need to store images in the image memory.

The maximum of 436 search models can be registered and these remain backed up even after the power switch is turned off.

ROI Processing

Processing only the specified region of the image memory is called the ROI (Region Of Interest) processing.

For executing the ROI processing, images need to be input in advance in page 1 of the image memory.

The following feature values can be obtained from the ROI processing.

- Binary area and center of gravity in the region,
- Gray-scale area and center of gravity in the region
- Binary weighted correlation in the region
- Gray-scale correlation with the registered model
- Mean density and density deviation in the region

Inspections or measurements of more than one places in the screen can be easily performed using the ROI processing function.

For executing the ROI processing, ROI models need to be registered in advance. The maximum of 1,024 ROI models can be registered and these remain backed up even after the power switch is turned off.

Window Memory (Frame Type)

The F350-C12E/C41E has a built-in window memory with the capacity of 512 pixels x 512 pixels x 8 bits.

The data in the window memory can be displayed together with the data from the image bus 0 (pipeline bus) or image bus 1 (video bus). It is useful when specifying a search processing region.

The window memory data can be scrolled or zoomed when displayed.

Character Memory (Plane Type)

The 512 pixels x 512 pixels x 1 bit character memory is mainly used for displaying characters.

Graphic patterns can be drawn on the character memory.

The character memory data doesn't affect measurement results.

The character memory has two pages. Normally page 0 is used.

Graphic Memory (Plane Type)

The 512 pixels x 512 pixels x 1 bit graphic memory is mainly used for displaying graphic patters.

The graphic memory is used for displaying the cursor or model frames.

The graphic memory data doesn't affect measurement results.

The character memory has two pages. Normally page 0 is used.

Display Control

Data from the image bus 0 (pipeline bus), image bus 1 (video bus), window memory, character memory, and graphic memory are synthesized to display images on the video monitor.

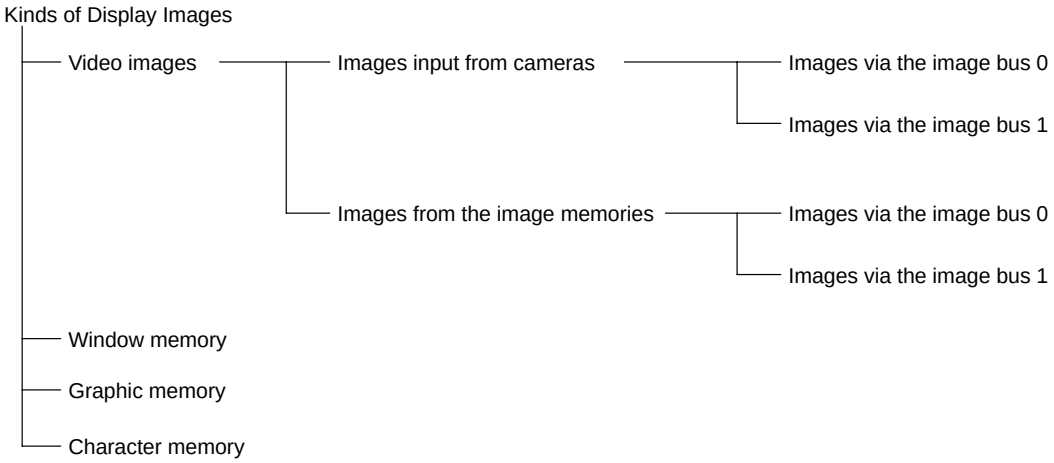
It is possible to set the gradation of displayed characters or graphics or to display filtered images only in the window.

4-2 Display

4-2-1 Display Images

The F350 System is designed to synthesize images input from cameras and data contained in the memories and to display them on the Video Monitor.

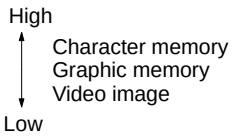
The following images can be displayed on the Video Monitor.



Images to be displayed can be selected using the DISPLAY command. Normally these can be displayed together in layers by giving the following command; `DISPLAY 31`.

4-2-2 Display Priority and Gradation

The display priority in layered images is defined as follows:



If a region of the character memory is filled, graphic memory data and video memory data in the region are not displayed.

If a region of the graphic memory is filled, video memory data in the region is not displayed.

If a region of the window memory is filled, video image in the region is brightened. The brightness can be varied by changing the setting of the SETDLVL command.

The display gradation of the character memory, graphic memory, or window memory can be set with the SETDLVL command. It is initially set to a value shown in the following table when the OVL is started up.

Kinds of images	Initial gradation
Character memory	255
Graphic memory	192
Mask image	0
Binary image, white	128
Binary image, black	0
Window memory, increments	48
Paint window memory, increments	32
Outside the image memory region	0

When the OVL is started up, video images are displayed in 0 to 192 gradation, which is changed from 0 to 255 gradation. The gradation change can be set using the SETDLUT command.

4-2-3 Control of Video Images

The video image mean a gray-scale image with normal gradation of 0 to 255 that passes through the image bus 0 (pipeline bus) or image bus 1 (video image). Depending on the setting of the LUT, it can be a binary image with eight sections.

Images that pass through the image bus could be gray-scale images or binary images depending on the commands or programs. It is important to know which type of images are passing through the image bus for performing proper display control.

Display control must be performed according to the type of images passing through he image bus, otherwise desired images won't be displayed. Use the DISPLAY command to control images to be displayed.

Command for displaying images that pass through the image bus 0 (pipeline bus): `DISPLAY 31, 0`
Command for displaying images that pass through the image bus 1 (video bus): `DISPLAY 31, 3`

4-2-4 Display of Camera Images

Display of Images without Filtering

Give the following commands to display the camera images that have not been filtered.

`DISPLAY 31, 3` : Character, graphic, and video images are displayed. Images from the image bus 1 (video bus) are displayed.
`CAMERA 0` : Input camera is designated.
`FILTERIN 0` : Camera images are sent to the image bus 1 (video bus).

Display of Filtered Images

Give the following commands to display the camera images that have been filtered.

`DISPLAY 31, 0` : Character, graphic, and video images are displayed. Images from the image bus 0 (pipeline bus) are displayed.
`CAMERA 0` : Input camera is designated.
`FILTERIN 0` : Camera images are sent to the image bus 1 (video bus).
`FILTSEL 11, 0` : Image filtering mode is selected.

4-2-5 Display of Image Memory Data

Display of Images without Filtering

Give the following commands to display the image memory data that has not been filtered.

`DISPLAY 31, 3` : Character, graphic, and video images are displayed. Images from the image bus 1 (video bus) are displayed.
`CAMERA 0` : Input camera is designated.
`FILTERIN 1, 0` : Images on the image memory page 0 are sent to the image bus 1 (video bus).

Display of Filtered Images

Give the following commands to display the image memory data that has been filtered.

`DISPLAY 31, 0` : Character, graphic, and video images are displayed. Images from the image bus 0 (pipeline bus) are displayed.
`CAMERA 0` : Input camera is designated.
`FILTERIN 1, 0` : Images on the image memory page 0 are sent to the image bus 1 (video bus).
`FILTSEL 11, 0` : Image filtering mode is selected.

4-3 Image Input

4-3-1 Image Input Function

The F350 System has two image memory cards to which images from cameras can be input.

Different images can be input to each memory card. Also the same images can be input simultaneously to both memory cards.

There are two image input modes: frame mode and field mode. The frame mode requires 33.3 ms for inputting images, while the filed mode requires only 16.7 ms.

Once the image input is started, it is completed by the hardware control. Therefore, the OVL Unit can be used for other processing.

Data in one image memory can be copied to the other memory.

Normally, the VIDEOIN command is used for inputting images. When the MEASURE, SMGRUN, or SMRUN command is used, images can be input while performing measurements.

4-3-2 Inputting Images from Cameras

Inputting Images without Filtering

Give the following commands when inputting camera images that have not been filtered into the image memory.

`CAMERA 0` : Input camera is designated.

`FILTERIN 0` : Camera images are sent to the image bus 1 (video bus).

`VIDEOIN –1, 0, 0` : Images from the image bus 1 (video bus) are input in the frame mode into all the pages of the image memory.

As the input path is set to 0 in the VIDEOIN command, images from the image bus 1 (video bus) are input.

As the page number is set to –1 in the VIDEOIN command, the same images are input into both page 0 and page 1.

Inputting Filtered Images

Give the following commands when inputting camera images that have been filtered into the image memory.

`CAMERA 0` : Input camera is designated.

`FILTERIN 0` : Camera images are sent to the image bus 1 (video bus).

`FILTSEL 11, 0` : Image filtering mode is selected.

`VIDEOIN –1, 1, 0` : Images from the image bus 0 (pipeline bus) are input in the frame mode into all the pages of the image memory.

As the input path is set to 1 in the VIDEOIN command, images from the image bus 0 (pipeline bus) are input.

As the page number is set to –1 in the VIDEOIN command, the same images are input into both page 0 and page 1.

4-3-3 Copying Images between Image Memory Pages

Copying Images without Filtering

Give the following commands to copy, without filtering, the images in the image memory page 0 into the image memory page 1.

`FILTERIN 1, 0` : Images in the image memory page 0 are sent to the image bus 1 (video bus).

`VIDEOIN 1, 0, 0` : Images from the image bus 1 (video bus) are input into the image memory page 1.

As the input path is set to 0 in the VIDEOIN command, images from the image bus 1 (video bus) are input.

As the page number is set to 1 in the VIDEOIN command, the images are input only into page 1.

Copying Filtered Images

Give the following commands to copy, after filtering, the images in the image memory page 0 into the image memory page 1.

`FILTERIN 1, 0`: Images in the image memory page 0 are sent to the image bus 1 (video bus).

`FILTSEL 11, 0`: Image filtering mode is selected.

`VIDEOIN 1, 1, 0`: Images from the image bus 1 (video bus) are input into the image memory page 1.

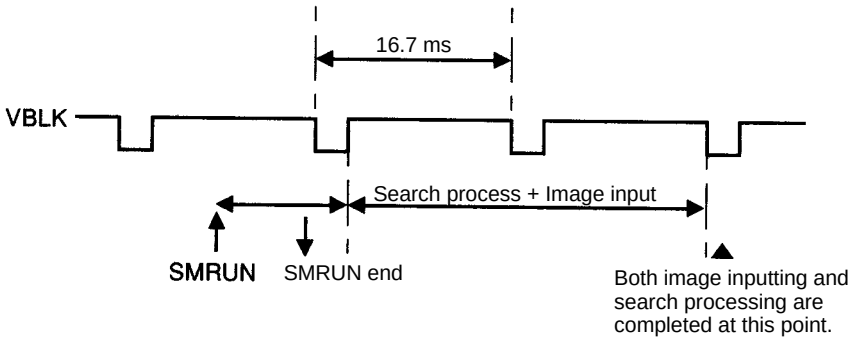
As the input path is set to 1 in the VIDEOIN command, images from the image bus 0 (pipeline bus) are input.

As the page number is set to 1 in the VIDEOIN command, the images are input only into page 1.

4-3-4 Inputting Images Using Commands Other than VIDEOIN

Images are normally input using the VIDEOIN command. When the (MEASURE, see note), SMGRUN, or SMRUN command is used for inputting images, measurements can be performed while inputting images.

By designating parameters of these commands, measurement and image input can be simultaneously executed, largely reducing the processing time.



4-3-5 Inputting Still Images

While inputting still images of a moving object using strobes or shutter cameras, the image input timing needs to be synchronized with the strobe flash or shutter timing.

The strobe flash or shutter timing can be synchronized with the image input timing by designating the trigger mode of the VIDEOIN, (MEASURE), SMGRUN, or SMRUN command.

Give the following commands to flash strobes and to input images by an input from a Console key.

```
IF KEYIN(0) THEN VIDEOIN –1,0,0,4
```

Give the following commands to flash strobes and to input images by an input of the STEP signal. However, once the STEP signal is input, the STEP synchronization is disabled. Therefore, it is necessary to execute the VIDEOIN command again to synchronize with the STEP signal.

```
VIDEOIN –1,0,0,2
```

4-3-6 Simultaneous Imaging

Images taken by multiple cameras can be stored all together by using a Camera I/F Unit that has simultaneous imaging function.

The Simultaneously Camera I/F Unit has a frame buffer where images for one screen can be stored for each camera.

When strobes are flashed, images are input into this frame buffer.

Strobes can be flashed by giving the FLASH command or by using the strobe flashing function with the VIDEOIN, (MEASURE), SMGRUN, or SMRUN command.

Images in the frame buffer can be sent to the IMP Unit using the CAMMODE command.

Give the following commands to flash strobes every 1/15 second and to display the images.

```
DISPLAY 31, 3 : Images from the image bus 1 (video bus) are displayed.
FILTERIN 0 :    Camera images are sent to the image bus 1 (video bus).
CAMMODE 1 :     Data in the frame buffer is sent to the IMP Unit.
```

4-4 Graphic Function

Graphic Functions

The following graphic patterns can be drawn on the character memory, graphic memory, window memory, image memory, and shading memory. The shading memory cannot be used with the F350-C12E/C41E.

ARC:	Arc
BOX:	Rectangle
CIRCLE:	Circle
CURSOR:	Cross cursor
ELLIPSE:	Ellipse
LINE:	Line
POLYGON:	Polygon
POLYLINE:	Polygonal line
PSET:	Point
SPCLOSE:	Close spline curve
SPLINE:	Spline curve

Graphic patterns or images can be copied to another memory using the following commands.

BCOPY:	Copy binary images or graphics
BCOPY2:	Copy enlarged or reduced images or graphics
GCOPY:	Copy gray-scale images or graphics
GCOPY2:	Copy enlarged or reduced gray-scale images or graphics

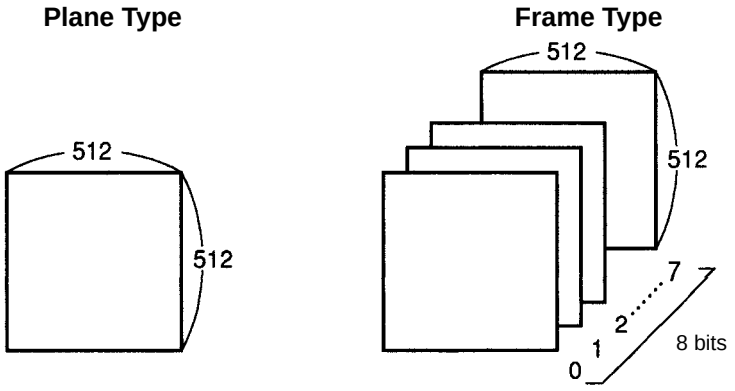
Memories

There are two kinds of memories that allow graphic drawing: plane memory and a frame memory.

The plane memory is composed of 512 pixels x 512 pixels x 1 bit. Only 0 (black) or 1 (white) graphic patterns can be drawn on this memory.

The frame memory is composed of 512 pixels x 512 pixels x 8 bits. Graphic patterns with 0 to 255 gradation can be drawn on this memory.

The frame memory can also be regarded as consisting of 512 pixels x 512 pixels x 1 bit x 8 screens. Each one of these screens is sometimes called as the binary image plane.



Type	Memory	Gradation
Plane	Character memory	0, 1
	Graphic memory	
Frame	Window memory	0 to 255
	Image memory	
	Shading memory (see note)	

Drawing Density and Drawing Mode

When drawing graphic patterns in the memories using the graphic function, “drawing density” and “drawing mode” must be designated for each graphic command.

Set the drawing density to designate the density of a graphic pattern to be drawn. For the plane memory, set to “0” or “Other than 0.” In case of the frame memory, set to a number between 0 and 255.

For example, if a graphic pattern is drawn at the drawing density of 100 in the frame memory, “1” is written in the binary planes 6, 5, and 2, and “0” is written in the rest of the binary planes.

100 = 01100100 (2)

Binary Plane Number	7	6	5	4	3	2	1	0
	0	1	1	0	0	1	0	0

Designate OR, XOR, or NOT in the drawing mode.

For the plane memory,
OR: “1” is written in the specified graphic pattern region.
XOR: Data in the specified graphic region is reversed.
NOT: “0” is written in the specified graphic pattern region.

For the frame memory,
OR: “255” is written in the specified graphic pattern region.
XOR: Data in the specified graphic region is reversed.
NOT: “0” is written in the specified graphic pattern region.

Designate NOT in the drawing mode to clear a specific graphic pattern.

Designate XOR in the drawing mode to delete the graphic patterns that have been drawn so far.

The drawing density and drawing mode cannot be designated at the same time.

Graphic Coordinates

Graphic patterns can be drawn using the graphic function in the space defined with coordinates (–32768, –32768) – (32767, 32767).

Of this space, the region that actually permits drawing is limited to (0, 0)–(511, 511) area and the region that is displayed on the screen is limited to (0, 0)–(511, 483) area.

For example, if the following commands are given, the center of a circle can be designated outside the screen to draw a part of the circle.

```
CIRCLE –100, –100, 200, 0
```

Mask Bit

Use the mask bit function when drawing graphic patterns only in a specific plane of the frame memory.

If the mask bit function is used, graphic patterns can be drawn only in a specific plane without destroying data in other planes.

The mask bit function can be designated using the MASKBIT command.

For example, give the following commands to draw graphic patterns only in the plane 7 of the window memory.

`MASKBIT 2, &H7F` : Masks all the planes except plane 7 of the window memory.

`CIRCLE 100, 100, 100, 2` : A circle is drawn on the window memory.

In the example, as the drawing gradation in the CIRCLE command is omitted, 255 (1 in all the planes) is to be drawn. However, planes 0 to 6 are write-protected by the MASKBIT command and, therefore, a circle is drawn only on plane 7.

4-5 Search Processing

4-5-1 Outline of Search Processing

Definition of Search Processing

Search processing refers to a series of processing actions by which the location (search location) of a pre-registered image, called the model image, is found in the input image.

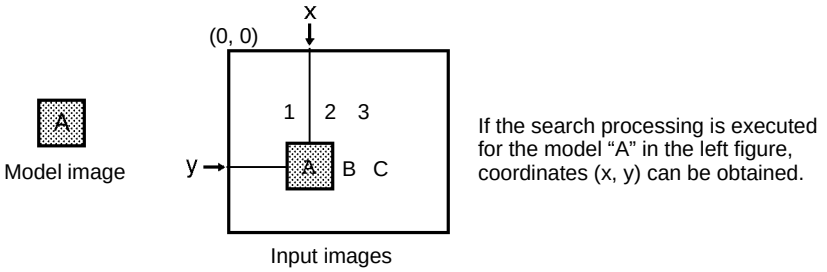
The search processing with the F350 System is executed in real-time (video rate). That is, the search location for a specified model can be found at a scan of 33.3 ms or 16.7 ms (fast mode).

This real-time search doesn't require images to be input in the image memory. Besides, the search processing can be executed simultaneously with image inputting process.

By using this search processing function, positional deviation and absolute position of an object can be easily found at a high speed. Furthermore, there is no need to set the window parameters as required with the previous models.

The search position is indicated in absolute coordinates with the origin (0, 0) at the top-left of the screen.

For executing the search processing function, it is necessary to set conditions including the search region and evaluation feature amount in addition to registration of a model image. The word "model" hereinafter refers to a model image together with these conditions.



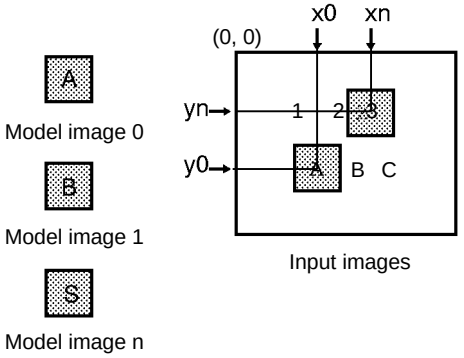
Simultaneous Search of Multiple Models

It is possible to execute the search processing function on multiple models with the F350 System. In this case, the search processing time for one model is the same as that for 10 models.

The F350 System is capable of performing the search processing on the maximum of 12 models in the same processing time (33.3 ms or 16.7 ms.)

The search processing function can be executed simultaneously on more than 12 models, but the processing time increases as shown in the following equation.

Search Processing Time = INT((Search model quantity + 11)/12)*(33.3 or 16.7) ms



If the search processing function is executed on models “A” to “3” in the figure, respective coordinate positions (x0, y0) to (xn, yn) can be obtained simultaneously.

Designate the starting model number and ending model number in the SMRUN command to designate which models are to be searched. If the group function described in the following section, it will be easier to search multiple models simultaneously.

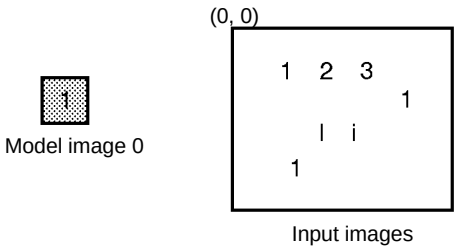
Detecting Multiple Positions

If multiple image patterns like a model image exist in the input screen, multiple positions of the respective image pattern can be detected.

The detected positions are referred to as the “candidate positions.”

The number of candidate positions to be detected can be designated for each model.

By using this function, the number of objects registered as the model image can be easily detected at a high speed.



In case of this figure, “1” at three positions and “i” and “l” each at one position respectively can be detected as candidate positions. It is possible to detect only “1” as the candidate position.

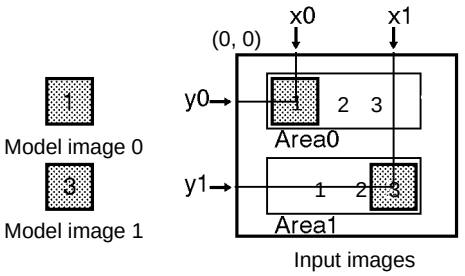
The number of candidate positions to be detected can be designated in the search quantity of the SMMODE or SMGMODE command.

Search Region

If multiple image patterns exist in one input screen and if a specific search position needs to be obtained, define the region where the search processing is to be performed. This region is called the “search region.”

Only one search region can be specified for one model.

If the approximate workpiece position is predetermined, use this function to avoid searching unnecessary positions.



Search command can be given in such a way that search processing is performed only in the Area A for the model image 0 and in the Area 1 for the model image 1. (If search processing is performed for the whole screen as shown in the figure, two positions will be obtained for each model.)

Use the SMAREA or SMGAREA command to designate search regions.

Evaluation Feature

The algorithm designed to locate search positions performs calculation for each pixel from the top-left to the bottom-right of the search region and evaluate the size of the calculation results to find locations. The calculation results used for evaluation are called the evaluation features.

The evaluation features include the following three:

Gray-scale Correlation: Using the image processing method called normalized correlation, the level of matching between model images and Input images is evaluated. This correlation feature allows stable search processing for variation in the brightness of input images and is the most commonly used evaluation feature in search processing.

Mean Density: The mean brightness in a specified region is evaluated. This feature is useful for finding the brightest location in a projected pattern of light beam at a high speed.

Density Deviation: The density deviation in a specified region is evaluated. The density deviation is a value that indicates the level of deviation in density in a region. This feature is useful for finding flaws on a flat plate at high-speed.

The following calculations are made by each evaluation feature.

Gray-scale correlation = sign (A) A x A / (B x C)

Where, $A = N\Sigma (I \times M) - (\Sigma I) \times (\Sigma M)$
$B = N\Sigma (I \times I) - (\Sigma I) \times (\Sigma I)$
$C = (M \times M) - (\Sigma M) \times (\Sigma M)$
N: Area of the model image
M: Density of the model image
I: Density of the input image

Density average = $(\Sigma I)/N$

Where, N: Area of the model image
I: Density of the input image

Density deviation = $\sqrt{((N\Sigma (I \times I) - (\Sigma I) \times (\Sigma I))/(N \times N))}$

Where, N: Area of the model image
I: Density of the input image

One of the above three evaluation features can be assigned to each model. Designate the one to be used in the evaluation feature of the SMMODE or SMGMODE command.

Evaluation Level

If the search function is executed for an input image where multiple image patterns like the model image exist, many candidate points will be found, including those that are not the ones to be searched.

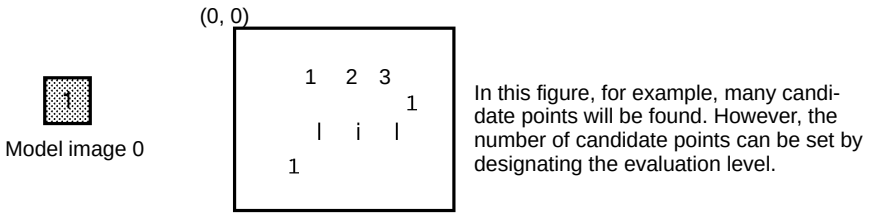
To minimize such findings, the evaluation level can be specified.

Once the evaluation level is specified in the search processing, only those positions that have higher values than the evaluation level are found as candidate points.

The evaluation level for each feature is as follows. The larger the value is, the less the number of candidate points, i.e., more detailed search processing is performed.

Gray-scale Correlation:	0.00 to 100.00
Mean Density:	0.00 to 255.00
Density Deviation:	0.00 to 127.50

The evaluation level can be specified for each model.

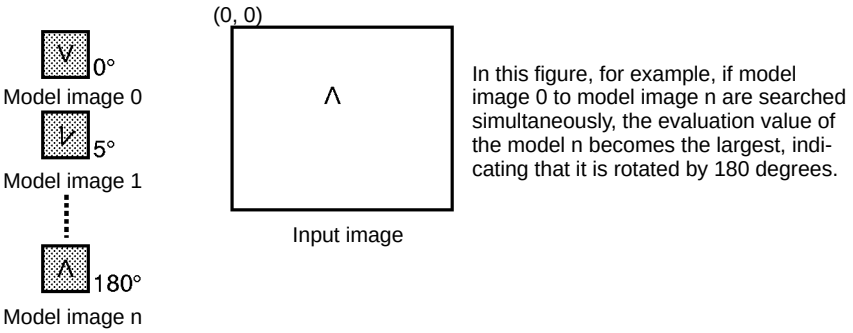


Designate the level of evaluation by specifying the evaluation level of the SMMODE or SMGMODE command.

Registration of Rotated Models

This function is to register multiple models of one image rotated gradually by a small angle.

By registering rotated models, positions of inclined objects can be searched and their inclination angles can be obtained.



Model images from model image 0 up to model image “n” are registered for the same model image 0.

Rotated models can be automatically created by using the SMROTATE command.

The rotation angle can be obtained by finding a model that has the largest evaluation level among those rotated models.

Group Registration

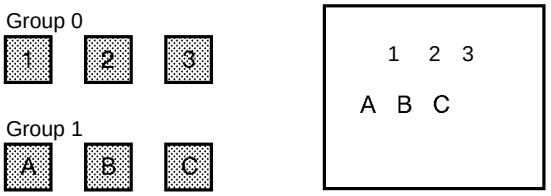
When executing search processing simultaneously on multiple models, it is troublesome to select models every time search is made. Also, it is convenient to execute a word search for a string of words on a group of models.

The group registration function that groups multiple models together as a unit is useful for such cases.

In the sequential processing, the rate of deviation can be detected for each group. This detection mode, when used together with the ROI model group, positional deviation can be easily corrected.

Search region and evaluation feature can be designated for a group. This means that settings which need to be repeated for multiple models can be executed at one time.

Use the SMGGROUP command to register search groups.



Input image

In the figure, for example, the position of the upper-string of words can be detected when the search is executed for group 0 and the position of the lower-string of words can be detected when the search is executed for group 1.

Sizing the Optimum Search Model

It is necessary to register the optimum search model to achieve more accurate search processing.

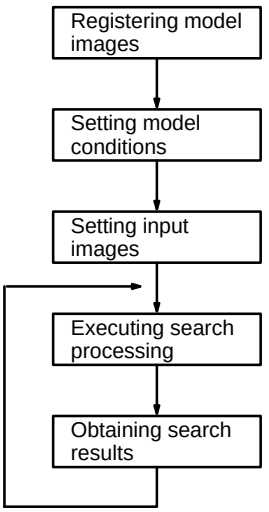
F350 System requires the size of a search model to be smaller than 71 pixels x 67 pixels. Any larger-size model must be sized to this scale or smaller to be registered as a search model. Normally large-size models are not required to be registered for search processing.

Use the SMSELECT command to select an optimum search model out of a designated large region.

4-5-2 Search Processing Flow

The basic flow of single search processing is as shown below.

Single processing refers to a process that is performed for each command. Another processing type is called the sequential processing, which is described in detail in a later section.



Single Processing Flow

Registration of Model Images

Before registering model images, input images into the image memory using the VIDEOIN or other relevant commands.

Register model images in advance to be used for search processing. Normally use the SMPUT command for registering model images.

Model images must be registered in advance before executing search processing.

As the model images, once registered, are not deleted even if the power is turned off, no repetitive registration is required every time search processing is performed.

Setting Model Conditions

Set conditions including the area to be searched, the number of candidate points to be searched, etc.

Use the SMAREA or SMGAREA command to set the search area. If model conditions are not registered, the whole screen becomes the search area.

Use the SMMODE or SMGMODE command to set the number of candidate points to be searched. If model conditions are not registered, only one point where the evaluation value is largest is searched.

The search area and the number of candidate points to be searched require no further setting if the default values are acceptable.

As the model conditions, once registered, are not deleted even if the power is turned off, no repetitive registration is required every time search processing is performed.

Setting Input Images

Designate the type of input images to be searched.

Basically the search processing is executed on gray-scale images. It is set to be executed on unfiltered gray-scale images when the OVL Unit is started up. Unless the type of input images needs to be changed, no particular settings are necessary.

For setting input images, change the image bus settings and filter settings.

Executing Search Processing

The search processing is executed according to the conditions set in advance. Use the SMRUN or SMGRUN command to execute the search processing.

As the search processing is executed in the background, other processing can be executed during the search processing.

The search processing can be executed for one model or for multiple models simultaneously.

The search processing time varies depending on the number of search models to be processed at a time.

As the model images and model conditions are retained even when the power is turned off, search processing can be executed right after the OVL Unit is started up provided that search models have been registered.

If images are set to be input by the argument of the SMRUN or SMGRUN command, images can be input while search processing is executed.

Search processing can be executed at a faster speed in the field mode.

If the processing mode of the SMRUN or SMGRUN command is designated before executing search processing, the candidate points can be sorted in the specified order, facilitating an easy access to the search processing results.

Obtaining Search Processing Results

As a result of search processing, the following data is stored in one model.

Search Result of Search Model

Number of candidate points stored (n) Total number of candidate points X coordinate of the point where the evaluation value is largest Y coordinate of the point where the evaluation value is largest Evaluation value of the point where the evaluation value is largest X coordinate of the point where the evaluation value is smallest Y coordinate of the point where the evaluation value is smallest Evaluation value of the point where the evaluation value is smallest Error information
X coordinate of the candidate point 0 Y coordinate of the candidate point 0 Evaluation value of the candidate point 0
X coordinate of the candidate point 1 Y coordinate of the candidate point 1 Evaluation value of the candidate point 1
X coordinate of the candidate point n–1 Y coordinate of the candidate point n–1 Evaluation value of the candidate point n–1

The “number of stored candidate points” refers to the number of candidate points stored in the result area. Normally, it means the number of candidate points designated in the search number of SMMODE or SMGMODE starting from the largest evaluation value.

The “total number of candidate points” refers to the number of candidate points in the screen that have higher evaluation values than the evaluation level set in SMMODE or SMGMODE mode.

If only one position is searched, the candidate point is at the coordinates where the evaluation value is largest. The data of candidate points becomes meaningful only when searching more than one position for one model.

Use the SMDATA function, SMGDATA, or SMMDATA command to obtain search results.

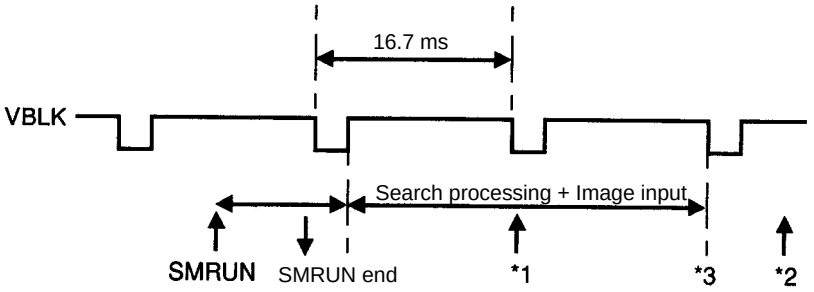
SMDATA: To obtain search results for one search model

SMGDATA: To store in the array all search results that belong to the search group

SMMDATA: To store in the array all search results for one search model

If the SMGDATA command is used, the positions searched for the search models in the search group can be obtained as sorted data.

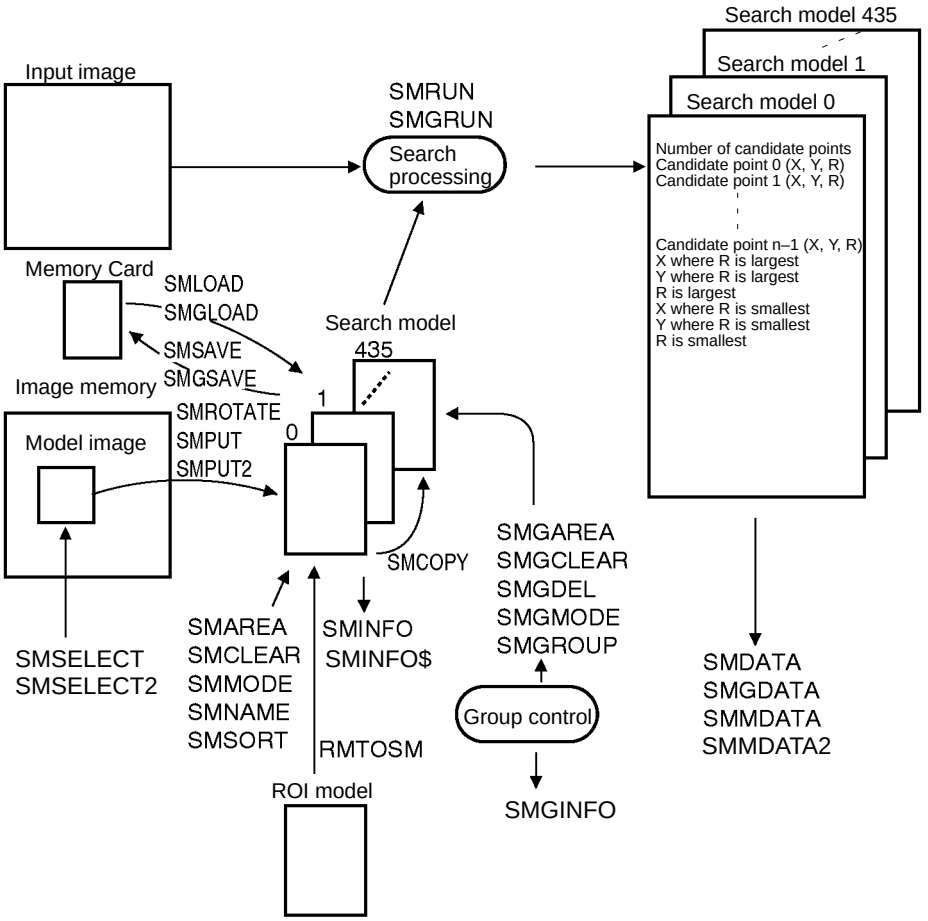
If search processing is still under execution when the command or function is executed to obtain search results, the command or function is kept waiting internally until the search processing is completed. Therefore, the search results can always be obtained for the last search processing irrespective of the result retrieving timing.



- When an attempt is made to obtain the search results at point *1, a wait time is required until the search processing is completed, and the results can be obtained only at point *3.
- When an attempt is made to obtain the search results at point *2 where the search processing has been completed, the results for the last search processing can be obtained.

4-5-3 Search Processing Conceptual Diagram

The following diagram shows configuration of data and commands used in the search processing



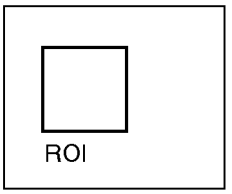
Conceptual Diagram of Search Processing

4-6 ROI Processing

4-6-1 Outline of ROI Processing

Definition of ROI Processing

ROI is an abbreviation of Region Of Interest. ROI processing means processing only the desired region in a screen, not the whole screen. If this function is used, processed results can be obtained in a short time.



Processing is applied only to the area marked in bold lines.

Image memory 1

Before executing ROI processing with the F350 System, images must be input in advance in image memory page 1. ROI processing cannot be executed on the image on the image memory page 0.

For executing ROI processing, processing conditions such as the processing area, type of processing, etc. need to be set. The whole set of these conditions is called the ROI model.

The ROI processing time is proportional to the size of the specified processing area and can be obtained from the following formula. This doesn't include the software processing time.

In case of high-precision search processing in the specified area, more time is required because ROI processing is repeated internally.

$$\text{ROI processing time} = \frac{80\ \text{ns}}{\text{Processing time per one pixel}} \times \text{ROI processing area (pixels)}$$

Measurement Feature

In the ROI processing, one of the following measurement features can be selected for one model.

- Binary feature
- Gray-scale feature
- Binary incremental correlation value
- Gray-scale correlation value

Measurement features that can be used in ROI processing are described below.

- Binary Feature:

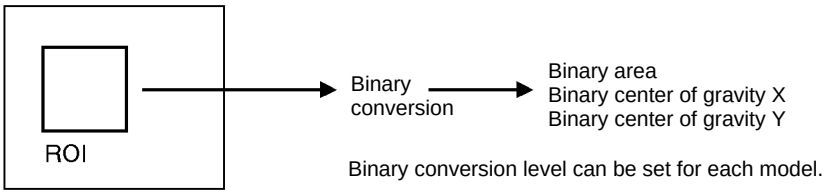


Image memory 1

- Gray-scale Feature:

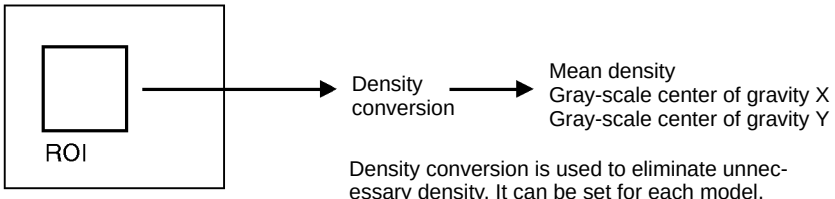


Image memory 1

- Binary Correlation:
  Binary correlation is useful for inspecting characters.

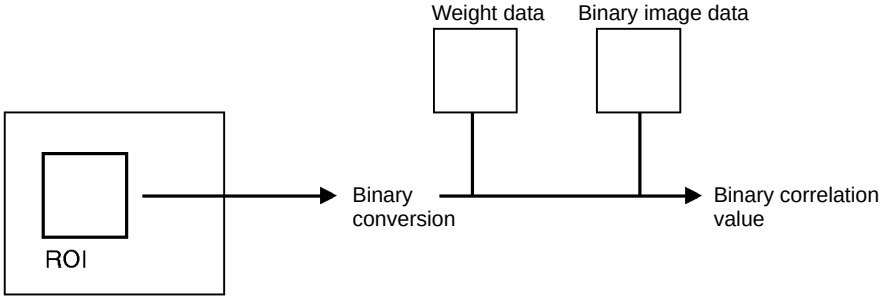
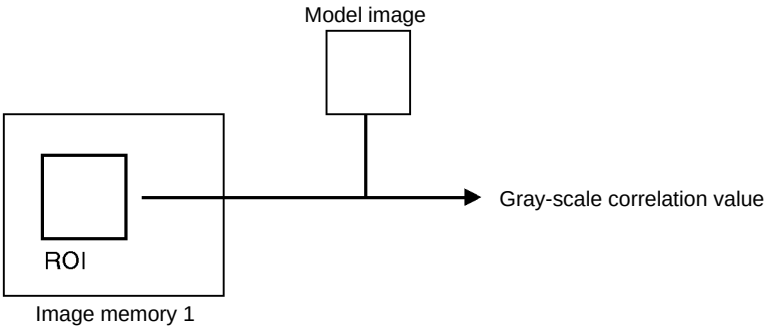


Image memory 1

• Gray-scale Correlation:

The gray-scale correlation value can be calculated using the same method as for the search processing.

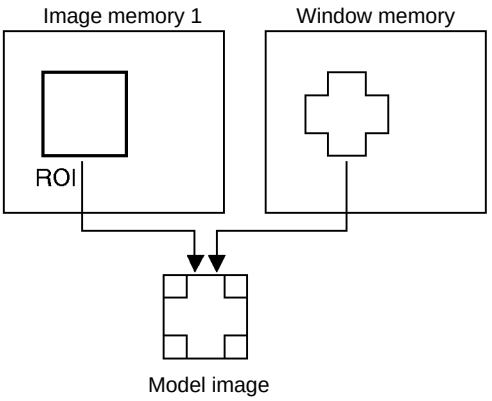


Designate the measurement feature in the measurement feature of the RMMODE or RMGMODE command.

Optional Area Processing

The ROI area is designated with a rectangle. Within the rectangle, any optional area for further processing can be set. Mask area can be designated.

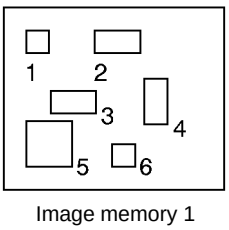
For setting an optional area, the area needs to be drawn on the window memory at the time of registering a model. The area where the window memory bits are set to ON is considered the processing area.



The model image is registered using the RMPUT command. During this registration, the processing area can also be registered in the window function.

Binary and Gradation Conversion Settings

When obtaining the binary feature, binary conversion level can be designated for each ROI model.



In the figure, for example, ROI processing can be executed for the binary feature at a different binary conversion level for each ROI model 1 to 6.

When obtaining the gray-scale feature, gray-scale gradation can be designated for each ROI model. Normally, processing for the gray-scale feature is executed on the images with 0 to 255 gradation. With the F350 System, it is also possible to execute processing only on the images with 50 to 150 gradation.

Processing with reversed gradation is also possible.

Use the density lower limit, density upper limit, or absolute value mode of the RMMODE or RMGMODE command to set the binary conversion level, gradation conversion, or image reversal.

High Precision Search

This function finds the position which has the largest evaluation value to a sub-pixel precision and sub-angle by repeatedly executing the ROI processing around the designated area.

Find the approximate position through the search processing and then repeat the ROI processing around that position to locate the precise position.

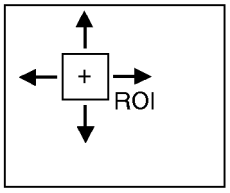


Image memory 1

Before executing the high-precision search, the gray-scale correlation needs to be designated in the measurement feature using the RMMODE command. Once the gray-scale correlation is set, set the processing mode of the RMMODE 2 command to the high-precision search mode.

Defect Detection Inspection

Any breakage, burrs, flaws, stains, etc. within the designated area can be inspected using the ROI processing function.

It is possible to detect even a small breakage or light stain using this function.

There are following four kinds of optional region processing.

- Linear region processing
- Arc region processing
- Circumferential region processing
- Optional region processing

For executing the optional region processing, a special ROI model needs to be registered in advance using the RMPUT2 command. Then designate the gray-scale feature in the measurement feature of the RMMODE command and specify the processing mode of the RMMODE2 command.

Judgment Processing

Judgment criteria for the measurement feature can be specified in the ROI model to obtain the results for the measurement feature through the ROI processing and at the same time to execute the judgment processing.

As the judgment is made in the ROI processing, there is no need judge measured results using the IF statements, and the total processing time can be reduced.

Set the judgment criteria using the RMJUDGE or RMGJUDGE command.

Group Registration

When executing the ROI processing for multiple models, it is cumbersome to designate which models have to be processed for every processing. For inspecting characters, it will be convenient if the processing can be executed for a string of characters.

This can be done by using the group registration function that is designed to group multiple models into one unit.

In the sequential processing, the position displacement detection mode can be set for every group. When this setting is correlated with the search model, position (displacement) compensation can be easily made.

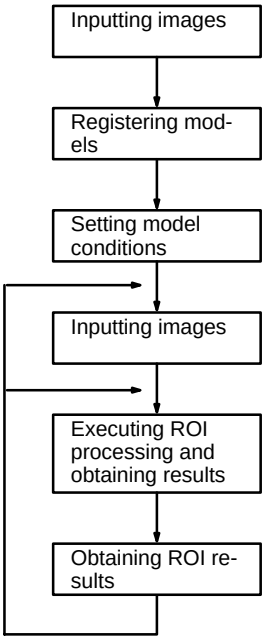
As the measurement feature and evaluation criteria can be set for a group at one time, just one setting procedure replaces several setting steps required for multiple models.

Use the RMGROUP command for registering the ROI group.

4-6-2 ROI Processing Flow

The following figure shows a basic ROI processing flow when it is being executed individually, i.e., single processing.

The single processing means a processing is performed for every command. The other type of processing is called the sequential processing, which is described in the following section.



Single ROI Processing Flow

Inputting Images

For registering images, images must be input in advance using the VIDEOIN command, etc.

Registering Images

Determine the model image and model size for ROI processing, and using the RMPUT command, register the model image in advance. Strictly speaking, the measurement feature is set to the binary feature or gray-scale feature, only the model size is required; no image data is necessary.

A model image must be registered before executing the ROI processing.

There is no need to renew the registered image for every execution of the ROI processing.

Setting Model Conditions

Set model conditions including the kind of feature for the region where the ROI processing is to be executed.

The conditions to be set in the ROI model include the measurement feature that specifies the items to be measured and the upper- and lower-density limits for density conversion. Set these conditions using the RMMODE or RMGMODE command.

Inputting Images

For executing the ROI processing, images to be processed need to be input in advance using the VIDEOIN command.

The ROI processing cannot be executed while images are being input. To wait until the images are completely input, execute the VDWAIT3 after the VIDEOIN command.

Executing ROI Processing and Obtaining Results

Use the RMRUN or RMGRUN command to execute the ROI processing.

The RMRUN command is used for processing an individual ROI model, while the RMGRUN command is used for processing multiple ROI models registered in the ROI group at one time.

The ROI processing results can be obtained as soon as the RMRUN or RMGRUN command is executed.

Unlike search processing, the RMRUN or RMGRUN command doesn't allow an exit from the command until the ROI processing is completely finished.

Obtaining ROI Results

Use the RMDATA or RMMDATA command to obtain ROI processing results.

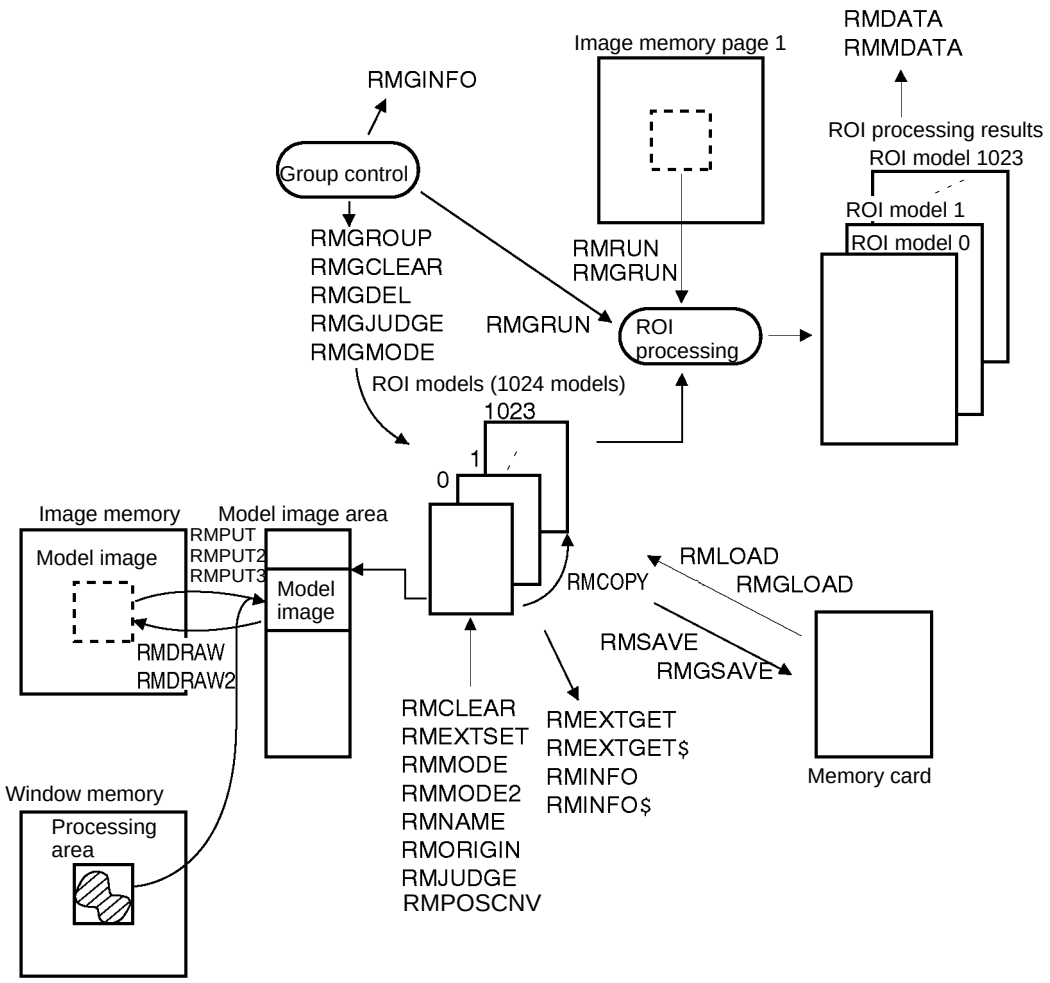
The RMDATA command stores in the array all the ROI processing results for one ROI model.

The RMMDATA command stores in the array all the ROI processing results for the designated number of ROI models.

It must be noted that, in storing processing results in the array using the RMDTA or RMMDATA, the storing method varies depending on the processing mode designated in the RMMODE or RMMODE2 command.

4-6-3 Conceptual Diagram of ROI Processing

The following diagram shows a configuration of data subject to ROI processing in relation to associated commands.



ROI Processing Conceptual Diagram

4-7 Sequential Processing

4-7-1 Outline of Sequential Processing

Definition of Sequential Processing

Sequential processing is a function that is designed to process a series of image processings at high speed including image inputting, position (displacement) compensation, and ROI processing.

Use the sequential processing to realize high-speed applications using the OVL Unit and utilizing the hardware features of the F350.

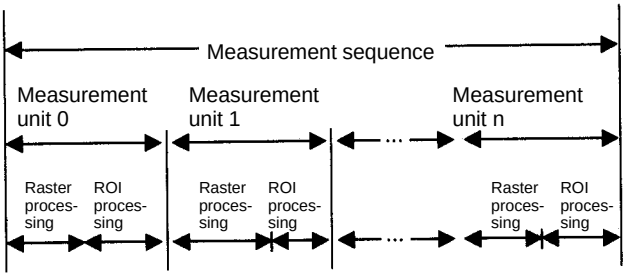
Before executing sequential processing, define which processing should be executed and when each one of them should be processed. All the specified processings are executed at one time when a measurement trigger is input.

The other processing type is called single processing. In single processing, each command is executed one by one. Therefore, high speed processing is not possible with this type due to overhead time of the OVL Unit.

Timing of Sequential Processing

For executing sequential processing, use the SQSET command and define the kind of processing to be executed according to the video signal timing.

Shown below are the basic timing patterns of sequential processing.



(0 ≤ n ≤ 15)

The measurement sequence consists of the maximum of 16 measurement units.

Each measurement unit consists of one raster processing and one ROI processing.

The raster processing means search processing, binary raster processing or image inputting. Note that the binary raster processing cannot be used with the F350-C12E/C41E.

Each measurement unit must include the raster processing, but the ROI processing is not necessarily required.

The raster processing time is always indicated as follows: 16.7 ms x n.

The ROI processing time depends on the content to be processed.

Information Necessary for Sequential Processing

Sequential processing can be executed only if desired kind of processing have been set in advance. Use the SQSET command for the settings.

Settings must be made for each measurement unit.

These settings include the following items:

- Which image (camera or image memory) must be processed?
- What kind of image filtering must be performed?
- Which search group must be used?
- What kind of position (displacement) compensation must be performed?
- Relation between the ROI model or ROI group processing and the search model or search group position?
- Do strobes need to be flashed?
- Interrupt processing must be performed for every measurement unit?

All the above items are set to the default settings when the OVL Unit is started up. Change only the settings of the items that need to be changed using the SQSET command.

Before executing sequential processing, set the conditions including the search model, search group, ROI model, and ROI group using the same commands as for single processing.

Using the Search Processing and ROI Processing

Use the search processing and ROI processing as shown below.

Find the object position using the search processing.

↓

Obtain the amount of position displacement.

↓

Scroll the image by the amount of position displacement.

↓

Inspect using the ROI processing.

The method of obtaining the amount of position displacement for the search processing results and also the use of ROI processing can be designated for every measurement unit using the SQMODE command.

Positioning mode can be correlated with the search mode and ROI processing as shown below.

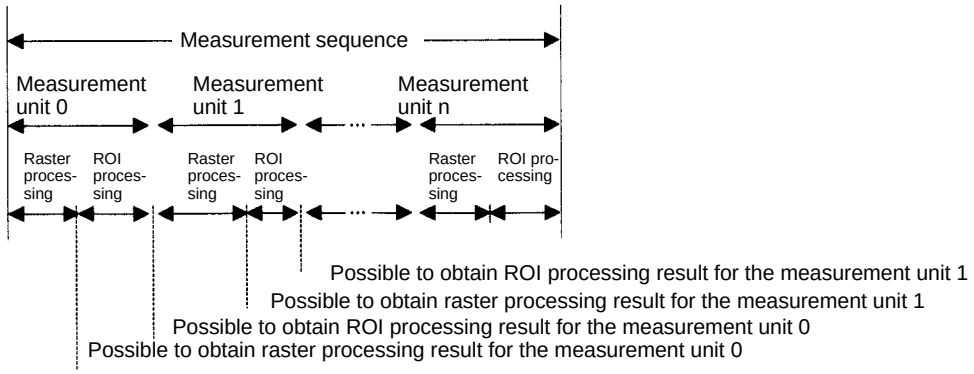
- No positioning (image scroll)
- 1 model positioning (not rotated)
- 1 model positioning (rotated)
- 2 model positioning (not rotated)
- 2 model positioning (rotated)
- 2 model positioning 2 (not rotated)
- 2 model positioning 2 (rotated)
- Circular positioning (not rotated)
- Circular positioning (rotated)
- Point where the evaluation value for the search model is largest → ROI model processing
- All candidate points for the search model → ROI model processing
- Point where the evaluation value for the search group is largest → ROI model processing
- All candidate points for the search group → ROI model processing
- Point where the evaluation value for the search model is largest → ROI group processing
- All candidate points for the search model → ROI group processing
- Point where the evaluation value for the search group is largest → ROI group processing
- All candidate points for the search group → ROI group processing

Obtaining Results from Sequential Measurement Processing

Sequential processing results can be obtained using the same commands and functions (SMDATA, SMGDATA, SMMDATA, RMDATA, RMMDATA) as those for single processing.

When one search model or ROI model is processed using more than one measurement unit, only the result from the last processing can be obtained.

The result can be obtained even if processing of all the measurement units have not been completed.



Obtaining the Processing Time

The time required for sequential processing can be obtained to check if the sequential processing time falls within the processing time required by the application.

The following times can be obtained:

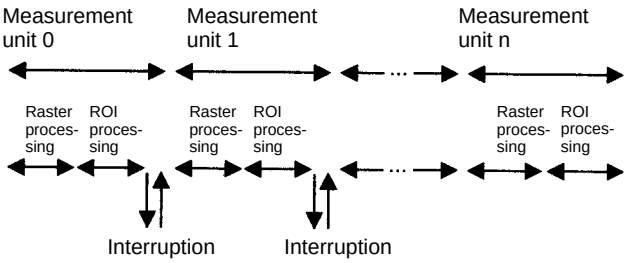
- Total processing time of sequential processing
- Time required for the designated measurement unit
- Raster processing time of the designated measurement unit

The processing time can be obtained using the SQTIME function.

Interrupting by Measurement Unit

The OVL program can be interrupted when the raster processing is completed.

The interruption can be specified for every measurement unit.

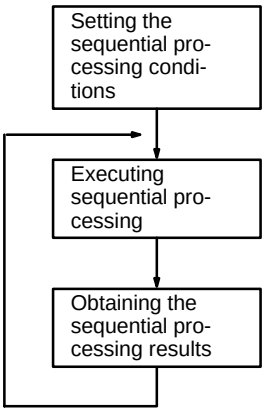


If this function is used effectively, it is possible to process complicated ROI processing or to change processing conditions of the next measurement unit.

Use the ON SQMEAS GOSUB command to specify the interrupt function in the sequential processing.

4-7-2 Sequential Processing Flow

The following figure shows the basic flow of sequential processing.



Setting the Sequential Processing Conditions

Use the SQSET command and define what kinds of processing should be executed in sequential measurement.

Executing the Sequential Processing

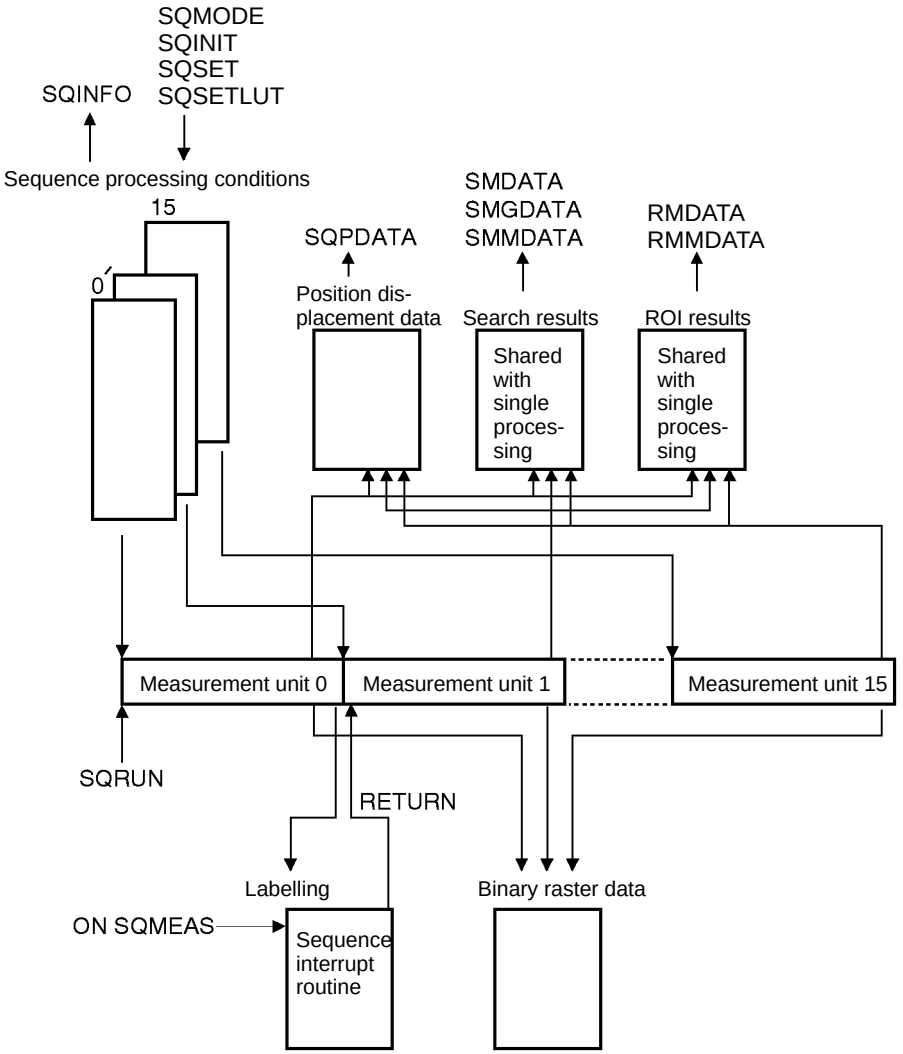
Sequential measurement processing is executed according the conditions set in advance.

Use the SQRUN command to execute sequential processing.

Obtaining Sequential Processing Results

Measurement results of sequential processing can be can be obtained by using the same command and functions as for single processing.

4-7-3 Conceptual Diagram of Sequential Processing



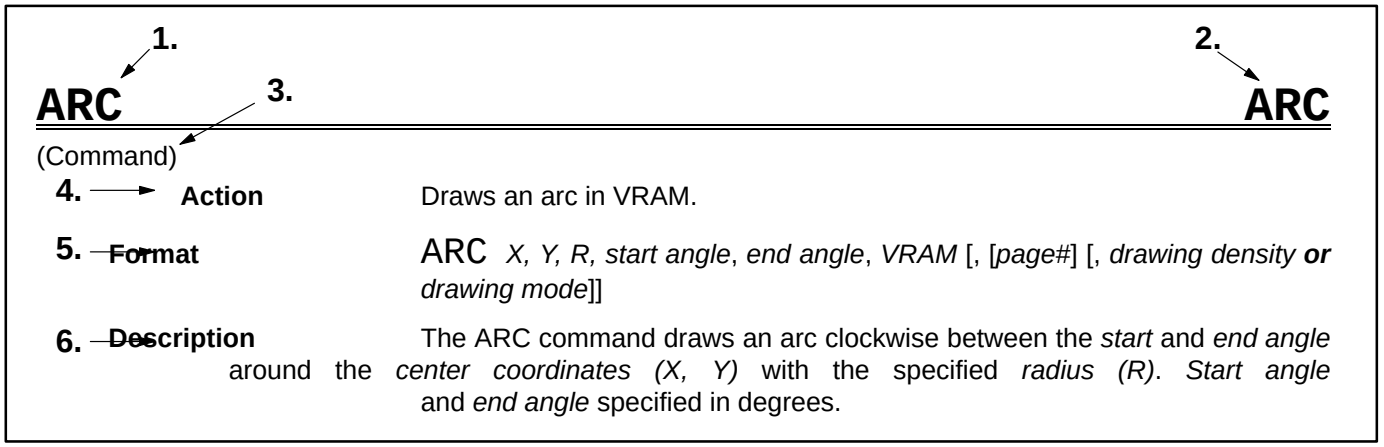
Conceptual Diagram of Sequential Processing

SECTION 5
Reference

This section provides detailed information on the commands and functions. Examples are also provided.

Introduction

The structure of the explanations given in this section are explained below.



1, 2, 3...

1. **Command or Function Mnemonic**

2. **Command or Function Name**

3. **Command or Function Classification**

4. **Outline of the Action Performed by the Command or Function**

5. **Format of the Command or Function**

Symbols used in the format are as follows:

- The mnemonic consists of one-byte alphabet characters, and is not case sensitive. (OVL will automatically convert to uppercase characters.) Any characters enclosed in double quotes or characters in DATA lines, however, are case sensitive.
- The parameters following the command or function are specified by the programmer and are described individually for each command or function.
- Any parameter enclosed in brackets "[ ]" are optional and may be omitted. Default values will be used for all omitted parameters. Refer to the explanations of individual command and functions for details.
- Any parameter followed by dots "..." may be input as many times as desired as long as the length of the line does not exceed 255 characters.

Example: The following may be input for

```
DATA constant [, constant [,constant]...]
DATA 10,15,20,25,30
```

6. **Details of the Action Performed and Parameter Specifications**

ABS — ABSolute

(Function)

Action Determines an absolute value.

Format ABS (*numeric expression*)

Description Determines the absolute value of the specified *numeric expression*.

The *numeric expression* may be in the form of an integer, long integer, or a single or double-precision real number.

AKCNV$ — Ank Kanji CoNVert $

(Function)

Action Converts 1-byte characters in a character string to 2-byte characters.

Format AKCNV$ (*character string*)

Description The AKCNV$ function converts all 1-byte characters in a *character string* to a 2-byte equivalent. The AKCNV$ function returns a character string containing 2-byte characters only.

Specify a *character string* to be converted to a 2-byte character string.

The KACNV$ function is the opposite of the AKCNV$ function. KACNV$ converts 2-byte characters in a *character string* to a 1-byte equivalent.

ANDOUT — AND OUT

(Command)

Action Controls the AND terminal.

Format ANDOUT *switch*

Description The ANDOUT command turns the Terminal Block Unit's AND terminal ON and OFF.

Specify the *switch* parameter as follows.

0: Turn OFF the AND terminal.
Other than 0: Turn ON the AND terminal.

When two or more Terminal Block Units are connected, the ANDOUT command controls the AND terminals on all of these Units.

ARC — ARC

(Command)

Action Draws an arc in VRAM.

Format ARC *X, Y, R, start angle*, *end angle*, *VRAM* [, [*page#*] [, *drawing density* ***or*** *drawing mode*]]

Description The ARC command draws an arc clockwise between the *start* and *end angle* around the *center coordinates (X, Y)* with the specified *radius (R)*. *Start angle* and *end angle* specified in degrees.

Specify the *VRAM* where the arc is drawn with a number, as follows.

0: Character memory
1: Graphic memory
2: Window memory
3: Image memory
4: Shading memory (Cannot be used in the F350-C12E/C41E.)

Set the page number to 0 (when omitting) or to 1.

Specify the drawing method with the *drawing density* or the *drawing mode*. The default value is *drawing mode*, OR.

The *drawing density* parameter specifies the density between 0 and 255 when drawing to the window, image, or shading memory. The *drawing density* parameter has the following effect when set for the character or graphic memory.

0 : 0 written to memory

Other than 0: 1 written to memory

The *drawing mode* settings operate as follows:

OR: The current contents of the VRAM ORed with 255 are written to memory.

NOT: 0 is written to memory

XOR: The current contents of the image memory are inverted.

When writing to a frame memory, the contents of planes write protected by the MASKBIT command remain unchanged.

ARRYFUNC ARRaY FUNCtion

(Command)

Action Executes operations between array variables.

Format ARRYFUNC*, operation, data number, array 1, array 2* [,array 3 [, array 4]]

Description Executes one operation between array variables.

Array operations can be executed at higher speed by means of processing such as FOR to NEXT.

Measurement results can be used effectively for comparisons with upper and lower limits, and so on.

For the *data number*, specify the number of array elements for executing operations.

Array variables must be declared in advance by means of the DIM command. Array variable names do not require accompanying characters and parentheses.

The number of array variable elements must be greater than the *data number*.

Faster processing can be achieved by using integer and long integer arrays when no decimal points are involved

Specify one of the following 13 types of operations. In these equations, A1 represents array 1, A2 represents array 2, and so on.

0: A1=A2 AND A3

1: A1=A2 OR A3

2: A1=A2 XOR A3

3: A1=A2 + A3

4: A1=A2 – A3

5: A1=A2 * A3

6: A1=A2 / A3

7: A1=–1 if (A2=A3)
0 if (A2A3)

8: A1=–1 if (A2 A3)
0 if (A2A3)

9: A1=–1 if (A2 A3)
0 if (A2A3)

10: A1=–1 if (A2 A3) AND (A2 A4)
0 if (A2A3) OR (A4A2)

11: The result of operation 10 above is substituted for A1 in bit strings. Array 1 must be in integer and long integer format.

12: A1=A2

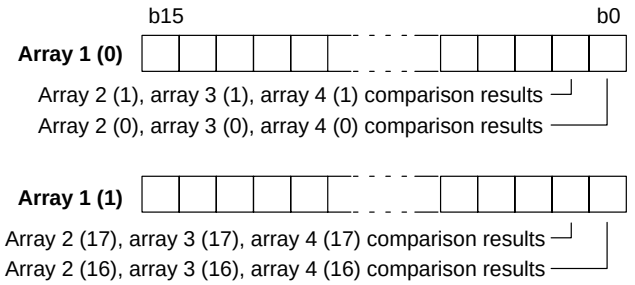
Dividing by zero yields a result of zero.

Overflow processing is not carried out.

Operation 11 above can be explained more fully as follows:

- The right side (i.e., the A2, A3, and A4 comparison results) is stored in order from the rightmost to leftmost bits of the array variable of the left side.
- If the right side result is true, the bit status will be "1." If the result is false, the bit status will be "0."
- If the left side array variable is an integer, then the results of right side element numbers 0 to 15 will be entered in element #0, and the results of right side element numbers 15 to 31 will be entered in element #1.
- The process is subsequently repeated. If the left side array variable is a long integer, the results of element numbers 0 to 31 is entered in element #0.

When array 1 is an integer-type array:



ASC — ASCii

(Function)

Action Determines the character code of the first character in a character string.

Format ASC (*character string*)

Description Returns the decimal character code of the first character in the specified *character string*.

The CHR$ function is the opposite of the ASC function. CHR$ returns the character corresponding to the specified character code.

ATN — Arc TaNgent

(Command)

Action Determines the arctangent (arctan) of a value.

Format ATN (*numeric expression*)

Description The ATN function returns the arctangent value in radians between –π/2 and π/2. Multiply this value by 180/π to convert the returned value to degrees.

The *numeric expression* may be in the form of an integer, long integer, or a single or double-precision real number.

ATTR$ ATTRibute $

(Function)

Action Determines the write protect attribute of the specified file.

Format ATTR$ (*filename* ***or*** *file#*)

Description Determines if the file specified with a *filename* or *file#* is write protected or write enabled and returns a letter indicating the status.

Specify the *filename* stored in a memory card as a character string. Specify the *file#* as the number in which the file was opened with the OPEN statement.

The ATTR$ function returns a 2-character character string (attribute code) corresponding to the write protect attribute assigned to the file specified with the *filename* or *file#*. The attribute codes are described in the table below.

Attribute code	Meaning
"__"	The file has no write protect attribute allocated. The file specified with the *filename* or *file#* is write enabled.
"_P"	The file specified with the *filename* or *file#* is write protected.
"E_"	The file specified with the *filename* or *file#* is coded.

Note _ indicates a blank

AUTO AUTOmatic numbering

(Command)

Action Automatically generates line numbers when the program is keyed in.

Format AUTO [*initial value*] [,*increment*]

Description When the AUTO command is input, the line number *initial value* is displayed at the start of the next line and the cursor stops to the right of this line number. Subsequently, each time the Return Key is pressed to input a line, the present line number plus the specified *increment* is displayed at the start of the next line.

The default value of the *initial value* is 10. The default value of the *increment* is 10.

If a line number already exists, an asterisk (*) is displayed after the line number.

Press the CTRL+C or STOP Keys to end automatic line number generation.

AUTOLVL AUTO LeVeL

(Function)

Action Determines the ideal binary level from density histogram data.

Format AUTOLVL (*algorithm, array name, subscript*)

Description The AUTOLVL determines the ideal binary level from the array data, which is specified with the *array name*, where the histogram data obtained with the HISTGRAM command is stored.

Always set the *algorithm* to 0. When the *algorithm* is set to 0, the binary level is calculated from the histogram using the discrimination analysis method.

The *subscript* specifies the first element used in the histogram data array. Normally set to 0.

BACKDISP BACKground DISPlay

(Command)

Action Sets the type of display for the image outside the window.

Format BACKDISP *display image* [, [*binary image plane#*] [, *binary reverse*]]

Description BACKDISP sets how the image outside the window is displayed.

Set the *display image* to one of the following values.

0: Raw image (displays the same image inside or outside window)
1: Binary image (displays image outside window)
2: Mask image (does not display image outside window)

The raw image is displayed outside the window when the OVL is started up.

If a binary image is specified as the *display image*, set the *binary image plane#* between 0 and 7 to specify the plane for binary conversion. The binary image corresponding to the specified binary image plane# is displayed outside the window. The default value is 0.

The binary reverse value can be specified if a binary image is selected, as follows.

0: Normal monochrome display (when omitted)
Other than 0: Reversed monochrome display

The default value is 0.

The BACKDISP command sets the display only and does not affect the measurements.

BATCHK BATtery CHecK

(Function)

Action Checks the battery voltage of the memory card.

Format BATCHK (*drive designation*)

Description BATCHK finds the battery voltage status of the memory card.

The BATCHK function returns the following values:

0: Battery voltage low (battery must be replaced)
1: Battery voltage unsatisfactory (battery should be replaced)
2: Battery voltage satisfactory (battery need be replaced)

Specify the “C:” drive name as the *drive designation*.

If a drive other than the memory card is designated, an error will be generated.

BCDTOBIN BCD TO BINary

(Function)

Action Converts BCD data to binary data.

Format BCDTOBIN (*numeric expression*)

Description BCDTOBIN converts BCD data values to binary. It can be used for operations such as inputting BCD data that has been input to a Terminal I/O Units or Parallel I/O Unit.

The range that can be designated for the *numeric expression* is 0 to &H99999999& in BCD.

BCOPY Binary COPY

(Command)

Action Copies a binary image in VRAM.

Format BCOPY *VRAM1*, [*page# 1*], [*plane# 1*], *VRAM2*, [*page# 2*], [*plane# 2*] [, [*X1*] [, [*Y1*] [, [*X2*] [, [*Y2*] [, [*X*] [, *Y*]]]]]]

Description BCOPY copies a binary image between VRAMs.

Specify the numbers of the copy source and destination VRAMs, with *VRAM1* and *VRAM2*, as follows:

0: Character memory
1: Graphic memory
2: Window memory
3: Image memory
4: Shading memory (Cannot be used in the F350-C12E/C41E.)

Set *page# 1* and *page# 2* to 0 (when omitted) or to 1.

Specify the plane number (0 to 7) of the copy source and destination VRAMs, with *plane# 1* and *plane# 2*.

To enclose the copy source image in a rectangular region, specify the top-left and bottom-right coordinates as *X1*, *Y1* and *X2*, *Y2*, respectively. The default value of *X1* and *Y1* is 0 and the default value of *X2* and *Y2* is 511.

Specify the top-left coordinates of the copy destination copy start region with *X*, *Y*. The default values of *X*, *Y* are 0,0.

When copying from a plane VRAM to a frame VRAM, if the plane# of the copy destination (*plane# 2*) is specified, only the specified plane is copied, otherwise all planes are copied.

The plane# of the copy source VRAM (*plane# 1*) must be specified when copying from a frame VRAM to a plane VRAM. Data in planes write protected with the MASKBIT command remains unchanged when data is copied to a frame VRAM.

BCOPY2 Binary COPY 2

(Command)

Action Makes enlarged and reduced copies of binary images in VRAM.

Format BCOPY2 *VRAM1*, [*page# 1*], [*plane# 1*], *VRAM2*, [*page# 2*], [*plane# 2*] [, [*XS1*] [, [*YS1*] [, [*XS2*] [, [*YS2*] [, [*XD1*] [, [*YD1*] [, [*XD2*] [, *YD2*]]]]]]]]

Description BCOPY2 makes enlarged and reduced copies of binary images between rectangular regions in VRAM.

Specify the numbers of the copy source and destination VRAMs, with *VRAM1* and *VRAM2*, as follows.

0: Character memory
1: Graphic memory
2: Window memory
3: Image memory
4: Shading memory (Cannot be used in the F350-C12E/C41E.)

Omit the setting for the *page# 1* and *page# 2* or set them to 0.

Specify the plane number (0 to 7) of the copy source and destination VRAMs, with *plane# 1* and *plane# 2*.

Specify the top-left and bottom-right coordinates of the copy source rectangular region as *XS1*, *YS1* and *XS2*, *YS2*, respectively. The default value of *XS1* and *YS1* is 0 and the default value of *XS2* and *YS2* is 511.

Specify the top-left and bottom-right coordinates of the copy destination rectangular region as *XD1*, *YD1* and *XD2*, *YD2*, respectively. The default value of *XD1* and *YD1* is 0 and the default value of *XD2* and *YD2* is 511.

When copying from a plane VRAM to a frame VRAM, if the plane# of the copy destination (*plane# 2*) is specified, only the specified plane is copied, otherwise all planes are copied.

The plane# of the copy source VRAM (*plane# 1*) must be specified when copying from a frame VRAM to a plane VRAM. Data in planes write protected with the MASKBIT command remains unchanged when data is copied to a frame VRAM.

BEDGE — Binary EDGE

(Command)

Action Detects the edge of a binary image.

Format BEDGE [*page#*], *binary image plane#* [,[*X1*] [,[*Y1*] [,[*X2*] [,[*Y2*] [,*number of connections*]]]]]

Description BEDGE detects the edge of a binary image for (*X1*, *Y1*) – (*X2*, *Y2*) of the image memory. It can be used for functions such as detecting the contours of a work-piece.

Specify the image memory page number (0 or 1) as the *page#*. The default value is 0.

For the *binary image plane#*, specify the binary image plane number (0 to 7) for the image memory that is to be detected.

Specify for *X1*, *Y1*, *X2*, and *Y2* the starting and ending points for the rectangle that is to be processed. If this setting is omitted, it will be regarded as (1, 1) – (510, 510).

Specify the *number of connections* as follows:

0: 8 (Default)
Other than 0: 4

Edge detection processing leaves any pixel value 1 adjoining pixel value 0 and changes the other pixel values to 0.

BEEP — BEEP

(Command)

Action Turns the buzzer on and off.

Format BEEP [*switch*]

Description The BEEP command turns the buzzer on and off.

Specify the *switch* parameter as 0 or 1, as follows.

0: Stop the buzzer.
1: Continuously sound the buzzer.

The buzzer sounds for a fixed time if the parameter is omitted.

After the buzzer is turned on, it can be turned off again only with the BEEP command. The buzzer remains on if program operation stops.

BINFO Binary file load INFOrmation

(Function)

Action Gets binary file information.

Format BINFO (*data type*)

Description BINFO gets information on the BLOAD command that was executed last.

Specify one of the following values (0, 1, 2, or 3) for the *data type* parameter.

Data type	Function
0	Returns the leading address specified by the BLOAD command. The default value is the first address of the user area.
1	Returns the size of the file loaded with the BLOAD command. The default value is 0.
2	Returns the last address (4-byte units) of the file specified by the BLOAD command. The default value is the first address of the user area.
3	Returns the first address of the user area.

The information will be retained even if the power is turned off.

BINTOBCD BINary TO BCD

(Function)

Action Converts binary data to BCD data.

Format BINTOBCD (*numeric expression*)

Description BINTOBCD converts binary data values to BCD. It can be used for operations such as outputting measurement data to a Terminal I/O Unit or Parallel I/O Unit.

BLOAD Binary file LOAD

(Command)

Action Loads the specified binary file into memory.

Format BLOAD *filename, address* [, *R*]

Description The BLOAD command files binary data in system memory.

The *filename* parameter specifies the name of the file to be loaded. Data can be loaded from the RS-232C port by specifying “COM: ” in the filename.

The *address* parameter specifies the leading address (absolute address) in memory where the file will be stored. The address must be specified in the long integer format.

When the *R* option is included, the program will be called from the specified address after the file is loaded.

System operation may be jeopardized when the binary data is loaded in addresses used by the system.

The system must be informed that the data is binary in order to handle binary data.

BMFUNC Binary Memory image FUNCtion

(Command)

Action Executes operations between binary images.

Format BMFUNC *operation,* [*page# 1*], *plane# 1,* [*page# 2*], *plane# 2* [,[*X1*] [,[*Y1*] [,[*X2*] [,[*Y2*]]]]

Description BMFUNC performs a binary image operation on the region defined by (*X1*, *Y1*) – (*X2*, *Y2*) in the two specified planes. The result of the operation is stored in plane# 2 of page# 2.

Specify one of the following 3 operations.

0: AND
1: OR
2: XOR

Specify the page numbers in image memory in the *page# 1* and *page# 2* parameters. The default value for the page numbers is 0.

Specify the top-left and bottom-right coordinates of the rectangular region with the *(X1*, *Y1)* and *(X2*, *Y2)* parameters. The default value of *(X1, Y1)* is (0, 0) and the default value of *(X2, Y2)* is (511, 511).

BOX BOX

(Command)

Action Draws a rectangle in VRAM.

Format BOX *X1, Y1, X2, Y2, VRAM* [, [*page#*] [,[*drawing density* ***or*** *drawing mode*] [, *lineart*]]]

Description The BOX command draws a rectangle between the opposing corner coordinates (*X1*, *Y1*) and (*X2*, *Y2*).

Specify the VRAM where the rectangle is drawn with a number, as follows:

0: Character memory
1: Graphic memory
2: Window memory
3: Image memory
4: Shading memory (Cannot be used in the F350-C12E/C41E.)

Either omit the *page#* (default setting 0) or set it to 1.

Specify the drawing method with the *drawing density* or the *drawing mode*. The default value is *drawing mode*, OR.

The *drawing density* parameter specifies the density between 0 and 255 when drawing to the window, image, or shading memory. The default value is 255. The *drawing density* parameter has the following effect when set for the character or graphic memory:

0 : 0 written to memory
Other than 0: 1 written to memory

The *drawing mode* settings operate as follows. (OR is the default setting.)

OR: The current contents of VRAM ORed with 255 are written to memory.

NOT: 0 is written to memory

XOR: The current contents of VRAM are inverted.

Specify with the *lineart* parameter if the rectangle is an outline only or filled. (0 is the default setting.)

0: Filled rectangle
1: Rectangle outline only

When writing to a frame memory, the contents of planes write protected with the MASKBIT command remain unchanged.

BSAVE Binary file SAVE

(Command)

Action Saves the data in the memory into a file.

Format BSAVE *filename, address, length*

Description The BSAVE command saves the data in the memory of the system into a file.

Specify the name of the file to be saved into the filename.

When "COM:" is specified in the filename, the file can be output to the RS-232C port.

In the address, designate the absolute address where the data is to be saved. The address must be specified in a long integer.

In the length, designate the length of the data to be saved in bytes.

Sufficient knowledge of the system is required to process the binary data.

BUSY BUSY

(Command)

Action Controls the BUSY signal output status.

Format BUSY *switch*

Description The BUSY command turns the BUSY signal output ON and OFF for the Terminal Block and Parallel I/O Units.

Set the *switch* parameter to 0 or 1, as follows.

0: Turns OFF the BUSY signal.
Other than 0: Turns ON the BUSY signal.

When multiple Terminal Block or Parallel I/O Units are connected, the BUSY signals are controlled simultaneously for all Units.

CALL CALL

(Command)

Action Calls a structural subroutine defined between the SUB and END SUB commands.

Format CALL *label* (*argument* [, *argument*...])

Description The CALL command calls structural subroutines in which local variables are used or general-purpose structural subroutines provided in the OVL System. Refer to *Section 6 General-purpose Structural Subroutines* for details on these subroutines.

The *label* parameter specifies a *label* designated with the SUB command.

The *arguments* specify data to be transferred to the subroutine. The *arguments* can be any type of variable, constant, or expression, but the argument type must match the type of *argument* used with the SUB command.

If a variable specified as an *argument* is changed in the subroutine, the value of the *argument* changes simultaneously. No logical limitation is placed on the number of *arguments*. However, the command line can physically accommodate up to 255 characters only.

An entire array cannot be used as an *argument*. Data from an array can be used as an *argument* by specifying individual elements of the array in the form: array name (subscript). If multiple *arguments* are specified with the CALL command, no variable name can be used more than once.

CALL CALL

(Command)

Action Calls a machine language subroutine.

Format CALL *variable name* (*argument* [, *argument*...])

Description The CALL command calls machine language subroutines loaded in the memory of the system.

In the variable name, specify a variable name substituted by the absolute address where the machine language subroutines are stored. A long integer is normally used.

In the argument, specify the variable name where data is to be passed to the machine language subroutines.

Any constant or formula cannot be specified in the argument.

Sufficient knowledge of the system is required to process the machine language subroutines.

CAMCHK CAMera CHecK

(Function)

Action Determines the connection status of the camera specified with the camera number.

Format CAMCHK (*camera#*)

Description The CAMCHK function determines the status of the camera specified with the *camera#* (0 to 7), and returns a value as follows:

0: Connected
–1: Not connected

The camera connection is confirmed by the presence of a proper synchronization signal being input from the camera.

CAMERA CAMERA

(Command)

Action Switches cameras.

Format CAMERA *camera#*

Description The CAMERA commands switches to the camera specified by the *camera#* (0 to 7).

The F350 cannot simultaneously receive inputs from multiple cameras. Only the image from the camera selected with the CAMERA command is processed.

With an internal synchronization camera, it takes several tens of ms for the image to be stabilized and input after the camera is switched.

CAMMODE CAMera MODE

(Command)

Action Selects the image output from a Camera I/F Unit.

Format CAMMODE *output image*

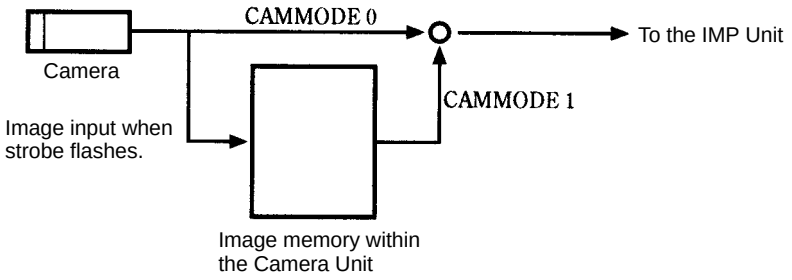
Description The CAMMODE command is valid for the Camera I/F Unit (normal/simultaneously) and Camera I/F Unit (shutter/simultaneously) only.
Both the Camera I/F Unit (normal/simultaneously) and Camera I/F Unit (shutter/simultaneously) have a built-in image memory able to hold one image screen from a single camera. The CAMMODE command selects if the image output from the Camera I/F Unit to the IMP Unit is the image output directly from the camera or the image from the Camera I/F Unit internal video memory.

The parameter of the CAMMODE command selects the *output image*, as follows:

0: Output the camera image directly.
Other than 0: Output the image from the internal video memory.

Images are written to the video memory with a command such as FLASH, SMRUN, SMGRUN, SQRUN, VIDEOIN, or (MEASURE). These commands input the image when a strobe emission signal or shutter trigger signal is received, so they are useful for capturing still images of moving objects.

If multiple Camera Units with internal memory are connected, the CAMMODE command applies to all Units.



CAMSYNC CAMera SYNChronous

(Command)

Action Sets the camera synchronization method.

Format CAMSYNC *sync method*

Description Specify the synchronization method with the *sync method* parameter, as follows:

0: External synchronization
Other than 0: Internal synchronization

The synchronization method applies to all connected cameras. Separate synchronization settings can't be made for different cameras.

The setting for the synchronization method at OVL startup is determined by the synchronization method set in the Setup menu.

CANCEL — CANCEL

(Command)

Action Clears the coded status set when the file was saved with the “P” option.

Format CANCEL *filename*

Description Cancels the coded (P option) status of the specified file. Specify a file on the memory card in the *filename* parameter.

CDBL — Convert DouBLe

(Function)

Action Converts a value to a double-precision value.

Format CDBL (*numeric expression*)

Description The CDBL function returns the specified numeric expression converted to a double-precision real number.

The CDBL function converts values to a double-precision real value. It does not change the number of significant digits of the returned value. Consequently, the precision of the returned value is the same as the precision of the value before conversion.

CGPAGE — Character & Graphic PAGE

(Command)

Action Switches the page in character memory and graphic memory.

Format CGPAGE *display page#* [*, active page#*]

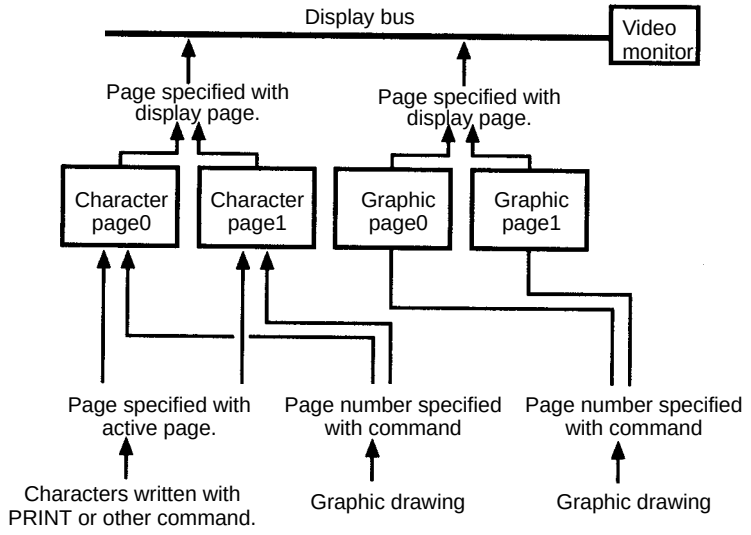
Description Switches the display page and the active page in character memory and graphic memory. In the F350, there are two pages each for character memory and graphic memory. The CGPAGE command controls the page in which characters are written and which page is displayed.

The active page is the page in which characters are drawn by commands such as the PRINT command. The active page isn’t related to the page# in graphics commands.

The display page is the page that is output to the video monitor. The *display page#* parameter specifies the page (0 or 1) that is displayed on the video monitor. The same setting applies to both character memory and graphic memory.

The *active page#* parameter specifies the page (0 or 1) in which character strings are drawn by commands such as the PRINT command. The default setting is page0. The same setting applies to both character memory and graphic memory.

The CGPAGE command can be used to switch quickly to a page that has been prepared in advance.



CHAIN CHAIN

(Command)

Action Calls and transfers control to a specified program.

Format

1. CHAIN *"filename"* [, *line#*] [, *ALL*]
2. CHAIN MERGE *"filename"* [*FOR APPEND*] [, *line#*] [, *ALL*] [, *DELETE line# 1 – line# 2*] [, *line# increment*]

Description The CHAIN command reads the program with the specified *filename* from the memory card and executes the program from the specified *line#*.

If no *line#* parameter is specified, the program is executed from the beginning.

If the command is specified in format 1, the current program is cleared before the specified program is read.

If the command is specified in format 2, the current program is merged with the new program. All variables remain unchanged if the *ALL* parameter is specified. If *ALL* is not specified, all numeric variables become 0 and character variables become blank (null). Use the COMMON command to leave some of the variables unchanged.

If the *DELETE* parameter is used, the program lines between the specified *line# 1* and *line# 2* are deleted before the programs are merged. The *line#* and *ALL* parameters can't be omitted.

Labels can be used instead of numbers for the *line#*, *line# 1*, and *line# 2* parameters.

If the *FOR APPEND* parameter is used, the program is merged from the end of the current program. The *line# increment* parameter is effective only when the *FOR APPEND* parameter is used.

The *line#* and *ALL* parameters can't be omitted when a *line # increment* is specified.

When the *FOR APPEND* parameter is used, processing will stop when the text's line number exceeds 65,535.

If a program with a "GOTO line#" command is merged with the *FOR APPEND* specification, line number in the GOTO command will not be updated.

CHDIR CHange DIRectory

(Command)

Action Changes the current directory.

Format CHDIR *directory name*

Description Specify the directory name parameter as a character string. Specify the name of a directory existing on the memory card. An error will occur if the specified directory does not exist on the memory card.

To specify an absolute path name for the directory, separate directory names with \ delimiters. The total length, including the \ delimiters, must not exceed 63 characters.

Specify the *directory name* as [".."] to switch the current directory up to the next hierarchy.

CHR$ CHaRacter $

(Function)

Action Determines the character corresponding to a character code.

Format CHR$ (*numeric expression*)

Description The CHR$ functions returns the 1-byte character or the control code corresponding to the character code.

Specify the *numeric expression* as an integer between 0 and 255.

The ASC function is the opposite of the CHR$ function. ASC returns the decimal character code corresponding to a character.

CHRCUT CHaRacter CUT

(Command)

Action Cuts characters. (Only one line)

Format CHRCUT *cut direction, X1, Y1, X2, Y2, dispersion level,* [*noise X*], [*noise Y*], [*integration width X*], [*integration width Y*], [*mask X*], [*mask Y*], [*margin X*], [*margin Y*], [*min. size X*], [*min. size Y*], *X1 array, Y1 array, X2 array, Y2 array, number of characters array*

Description When a rectangular region and operating parameters have been provided, the CHRCUT command outputs the coordinates of each character that was cut and the number of characters that was cut.

The *cut direction* parameter specifies the direction in which the characters are cut.

0: Y direction → X direction (Cuts characters written horizontally.)
(eg. AB)
1: X direction → Y direction (Cuts characters written vertically.)
(eg. AD)
Example: A B
D

Specify the upper-left and lower-right corners of the rectangular region in which the cut operation will be performed with parameters *X1*, *Y1, X2*, *Y2*. The setting range for these parameters is (0, 0) to (511, 483). Characters that overlap the rectangular region won't be cut.

Specify the dispersion evaluation level (0 to 100) in the *dispersion level* parameter.

Specify the noise-removal size in the X and Y directions with the *noise X* and *noise Y* parameters. The default value for these parameters is 0. If the width or height of a cut region is less than the noise-removal size, that region won't be stored as a cut result.

Specify the region integration spacing with the *integration width X* and *integration width Y* parameters. The default value for these parameters is 0.

If the space between two cut regions is less than the region integration spacing, two or more regions will be cut as a single region.

Specify the cut mask margin with the *mask X* and *mask Y* parameters. The default value for these parameters is 0.

Specify the size of the margins added to the cut region with the *margin X* and *margin Y* parameters. The default value for these parameters is 0. These are model width margins, so the characters won't be cut if they extend out of the rectangular region.

Specify the minimum size of the cut region with the *min. size X* and *min. size Y* parameters. The default value for these parameters is 0. If the cut size is less than the minimum model size, the region will be expanded to the minimum size and then cut.

Specify the names of the one-dimensional variable arrays that will contain the coordinates of the cut regions with the *X1 array*, *Y1 array*, *X2 array*, and *Y2 array* parameters. Coordinate values are saved in the order the characters are cut.

Specify the name of the one-dimensional variable array that will contain the number of cut characters with the *number of characters array* parameter. The number of cut characters will be 0 if nothing was cut at all. The total number of cut characters will be saved at the beginning of the array.

CHRCUTM CHaRacter CUT into Multiline

(Command)

Action Cuts characters. (Multiple lines)

Format CHRCUTM *cut direction, X1, Y1, X2, Y2, dispersion level,* [*noise X*], [*noise Y*], [*integration width X*], [*integration width Y*], [*mask X*], [*mask Y*], [*margin X*], [*margin Y*], [*min. size X*], [*min. size Y*], *X1 array, Y1 array, X2 array, Y2 array, number of characters array, number of lines array, number of characters/lines array*

Description When the rectangular region and each processing parameter have been provided, the CHRCUTM command outputs the coordinates of each character that was cut, number of characters cut, number of lines cut, and number of characters cut in each line.

The *cut direction* parameter specifies the direction in which the characters are cut.

0: Y direction → X direction (Cuts characters written horizontally.)
(eg. ABCDEF)
1: X direction → Y direction (Cuts characters written vertically.)
(eg. ADBECF)
Example: A B C
D E F

Specify the upper-left and lower-right corners of the rectangular region in which the cut operation will be performed with parameters *X1*, *Y1, X2*, *Y2*. The setting range for these parameters is (0, 0) to (511, 483). Characters that overlap the rectangular region won't be cut.

Specify the dispersion evaluation level (0 to 100) in the *dispersion level* parameter.

Specify the noise-removal size in the X and Y directions with the *noise X* and *noise Y* parameters. The default value for these parameters is 0. If the width or height of a cut region is less than the noise-removal size, that region won't be stored as a cut result.

If the space between two cut regions is less than the region integration spacing, two or more regions will be cut as a single region.

Specify the region integration spacing with the *integration width X* and *integration width Y* parameters. The default value for these parameters is 0.

Specify the cut mask margin with the *mask X* and *mask Y* parameters. The default value for these parameters is 0.

Specify the size of the margins added to the cut region with the *margin X* and *margin Y* parameters. The default value for these parameters is 0. These are model width margins, so the characters won't be cut if they extend out of the rectangular region.

Specify the minimum size of the cut region with the *min. size X* and *min. size Y* parameters. The default value for these parameters is 0. If the cut size is less than the minimum model size, the region will be expanded to the minimum size and then cut.

Specify the names of the one-dimensional arrays that will contain the coordinates of the cut regions with the *X1 array*, *Y1 array*, *X2 array*, and *Y2 array* parameters. Coordinate values will be saved in the order the characters are cut.

Specify the name of the one-dimensional variable array that will contain the number of cut characters with the *number of characters array* parameter. The total number of cut characters will be saved at the beginning of the array.

Specify the name of the one-dimensional variable array that will contain the number of cut lines with the *number of lines array* parameter. The total number of cut lines will be saved at the beginning of the array.

Specify the name of the one-dimensional variable array that will contain the number of characters cut in each line with the *number of characters/line array* parameter. When the cutting direction is set to 0 (Y direction to X direction), the number of characters cut for each line will be saved. When the cutting direction is set to 1 (X direction to Y direction), the number of characters cut for each row will be saved.

The number of cut characters will be 0 if nothing was cut at all.

CINT — Convert INTeger

(Function)

Action Converts a value to an integer.

Format `CINT` (*numeric expression*)

Description The CINT function rounds off the decimal places of the specified *numeric expression* and returns an integer.

An error will occur if the integer returned by the CINT function isn't within the range –32768 to 32767.

The FIX and INT functions are similar to the CINT function. However, the FIX function simply cuts off (rounds down) the decimal places from the specified value. The INT function also rounds down the specified value, but this function never returns a value larger than the value specified with the numeric expression.

CIRCLE CIRCLE

(Command)

Action Draws a circle in VRAM.

Format CIRCLE *X, Y, R, VRAM* , [, [*page#*] [, [*drawing density* ***or*** *drawing mode*] [, *lineart*]]]

Description The CIRCLE command draws a circle around the center coordinates (X,Y) with the specified radius (R).

Specify the *VRAM* where the circle is drawn with a number, as follows:

0: Character memory

1: Graphic memory

2: Window memory

3: Image memory

4: Shading memory (Cannot be used in the F350-C12E/C41E.)

Either omit the *page#* (default setting 0) or set it to 1.

Specify the drawing method with the *drawing density* or the *drawing mode*. The default value is drawing mode, OR.

The *drawing density* parameter specifies the density between 0 and 255 when drawing to the window, image, or shading memory. The default value is 255. The *drawing density* parameter has the following effect when set for the character or graphic memory:

0 : 0 written to memory

Other than 0: 1 written to memory

The *drawing mode* settings operate as follows. (OR is the default setting.)

OR: The current contents of VRAM ORed with 255 are written to memory.

NOT: 0 is written to memory

XOR: The current contents of VRAM are inverted.

The default value for the *drawing mode* is OR.

Specify with the *lineart* parameter if the circle is an outline only or filled. (The default setting is 0.)

0: Filled circle

1: Circle outline only

When writing to a frame memory, the contents of planes write protected with the MASKBIT command remain unchanged.

CLEAR CLEAR variable

(Command)

Action Initializes variables.

Format CLEAR (*mode*)

Description The *mode* parameter determines which variables will be initialized. (The default setting is 0.)

0 : All variables are initialized.

1: All variables except backup variables are initialized.

When the *mode* is set to 1, CLEAR will initialize all variables that don't start with characters specified by the VARBACK command.

CLNG Change to LoNG integer

(Function)

Action Converts a value to a long integer.

Format CLNG (*numeric expression*)

Description The CLNG function rounds off the decimal places of the specified numeric expression and returns a long integer.

An error will occur if the long integer returned by the CLNG function isn't within the range –2147483648 to 2147483647.

CLOSE CLOSE

(Command)

Action Closes an open file.

Format CLOSE [*file#* [, *file#*] ...]

Description The CLOSE command closes a file previously opened for data I/O.

Specify the same *file#* used when the file was opened with the OPEN command.

After a file is closed with the CLOSE command, data I/O operations on the file are not possible until the file is reopened with the OPEN command.

After a file is closed with the CLOSE command, the same *file#* can be used to open a different file with the OPEN command. Alternatively, the closed file can be reopened with the original *file#*.

Multiple file numbers (*file#*) can be specified to close multiple files simultaneously with a single CLOSE command. All open files are closed if the *file#* is omitted. Similarly, all files are closed automatically when the END, NEW, or STOP command is executed.

When a file opened for data output is closed, all data remaining in the file buffer is written to the file before it is closed. The CLOSE command must be used to ensure this data is correctly written to the file.

CLRUINFO CLeaR User INFOrmation

(Command)

Action Clears information stored in the user information area.

Format CLRUINFO

Description The CLRUINFO command clears the user information that was set with the SETUINFO command.

CLS CLear Screen

(Command)

Action Clears the specified VRAM.

Format CLS [*VRAM* [, [*page#*] [, *plane#*]]]

Description The CLS command clears the specified VRAM.

Specify the *VRAM* to be cleared with a number, as follows. (The default setting is 0.)

0: Character memory

1: Graphic memory

2: Window memory

3: Image memory

4: Shading memory (Cannot be used in the F350-C12E/C41E.)

Either omit the *page#* (default setting 0) or set it to 1.

Specify the plane to be cleared with the *plane#* parameter. All planes are cleared if this parameter is omitted. When a frame memory is specified, the contents of planes write protected with the MASKBIT command remain unchanged.

COLOR COLOR

(Command)

Action Changes the text display, character attribute.

Format COLOR *attribute code*

Description The COLOR command changes the character attribute to the specified attribute.

Specify the attribute with a number, as follows.

0: Clear attribute

2: Overline

4: Underline

7: Reverse

After the COLOR command is executed, the specified attribute is added to character displayed by commands such as the PRINT command.

COLOR@ COLOR @

(Command)

Action Changes the character attribute within a rectangular region of the text display.

Format COLOR@ *(X1, Y1) – (X2, Y2)* [,*attribute code*]

Description The COLOR@ command changes the character attribute to the specified attribute within the rectangular region between the opposing corners specified in character coordinates (*X1*, *Y1*) and (*X2*, *Y2*).

Specify the horizontal character coordinates *X1* and *X2* as an integer between 0 and 63 and the vertical character coordinates *Y1* and *Y2* between as an integer between 0 and 24.

Specify the *attribute code* with a number, as follows. (The default setting is 7.)

0: Clear attribute

2: Overline

4: Underline

7: Reverse

COM ON/OFF/STOP COMmunication ON/OFF/STOP

(Command)

Action Enables, disables, or stops interrupts from the RS-232C.

Format COM [(*RS-232C port#*)] *ON* ***or*** *OFF* ***or*** *STOP*

Description Specify the *port#* as 1 or 2. The default value is 1. The *port#* is represented by n in the description below.

COM(n) ON — Branches to the interrupt subroutine at the specified line# or label when input data is received via the specified *port#*. The line# or label must be specified with the ON COM GOSUB statement before the COM(n) ON command is executed. In addition, use of the RS-232C communication port must be declared with the OPEN command before the COM(n) ON command is executed.

COM (n) OFF — Disables branching to the interrupt routine when input data is received via the specified *port#*.

COM (n) STOP — Temporarily stops operation when input data is received via the specified *port#*. Operation does not branch to the interrupt subroutine. Operation will branch to the interrupt subroutine after the COM (n) ON command is executed.

The COM (n) OFF status must be set before the program execution ends.

COMMON COMMON

(Command)

Action Transfers variables to a program linked with the CHAIN command.

Format COMMON *variable name* [, *variable name* ...]

Description The COMMON command transfers variables to another program. The COMMON command is used as a pair with the CHAIN command. The COMMON statement must be declared before the CHAIN statement, that is, at the start of the program.

Add parentheses to declare array variables, e.g., B( ).

Use the *ALL* parameter in the CHAIN statement to transfer all variables to the linked program.

CONKEY ON/OFF/STOP CONsole KEY ON/OFF/STOP

(Command)

Action Enables, disables, or stops interrupts from the keyboard.

Format CONKEY *ON* ***or*** *OFF* ***or*** *STOP*

Description The CONKEY command controls branching to interrupt routines that occurs when a key is pressed.

CONKEY ON Enables interrupts, so operation will branch to the interrupt routine defined by the ON CONKEY GOSUB command each time that a console key is pressed.

CONKEY OFF Disables branching to the interrupt routine.

CONKEY STOP After CONKEY STOP is executed, console key inputs will be recorded, but operation won't branch to the interrupt subroutine. Operation will branch to the interrupt subroutine defined by the ON CONKEY GOSUB command after the CONKEY ON command is executed.

CONSOLE CONSOLE

(Command)

Action Sets the text display mode.

Format CONSOLE [*scroll start line*] [*,* [*number of scroll lines*] [*,* [*function key display switch*] [*, character mode*]]]

Description The CONSOLE command sets the text display mode.

The area designated by the scroll start line and number of scroll lines parameters. The screen clear operation acts on this specified scroll area. The default value for the scroll start line is 0. If the number of scroll lines is not specified, the previous setting is maintained.

The function key display switch setting specifies whether the function key menu is displayed on the bottom line of the screen. The function key menu is not displayed if this parameter is set to 0. If the function key display switch is not specified, the previous setting is maintained.

Set the character mode to 1 to select the graphic character mode. This mode permits display of the graphic characters (character codes &H80 to &H9F and &HE0 to &HF7) but disables the Japanese character display.

Set the character mode to 0 to enable the Japanese character display but disable display of the graphic characters. If no setting is specified, the previous setting is maintained. A "g" will be displayed at the bottom of the screen when graphic character mode is set.

CONT CONTinue

(Command)

Action Continues execution of a stopped user program.

Format CONT

Description The CONT command is a direct command to continue execution of a user program stopped by pressing the STOP or CTRL+C Keys or by executing the STOP command.

Operations such as printing the variable names in the direct mode are possible while the program is stopped. However, a program cannot be continued if it was modified while execution was stopped.

COS COSine

(Function)

Action Determines the cosine of an angle.

Format COS (*numeric expression*)

Description Returns the cosine as a value between –1 and 1.

The angle in the *numeric expression* is set in radians.

The *numeric expression* may be in the form of an integer, long integer, or a single or double-precision real number.

CSNG Convert SiNGle

(Function)

Action Converts a value to a single-precision value.

Format CSNG (*numeric expression*)

Description The CSNG function returns the specified *numeric expression* converted to a single-precision real number.

The CSNG function converts values to a single-precision real value between –1.70141E+38 and –1.70141E+38. An error will occur if the value lies outside this range.

CSRLIN CurSoR LINe

(Function)

Action The CSRLIN function determines the number of the line where the character cursor is positioned.

Format CSRLIN

Description Returns the line position of the character cursor (the Y-axis character coordinate).

The value returned is between 0 and 24.

Use the POS function to determine the cursor character position along the line.

CURSOR CURSOR

(Command)

Action Draws the cross cursor in VRAM.

Format CURSOR *X*, *Y*, *T*, *angle*, *VRAM* [,[*page#*][,*drawing density* ***or*** *drawing mode*]]

Description The CURSOR command draws a cursor at coordinate position (*X*,*Y*) at the specified angle.

The angle parameter specifies the tilt angle of the cross cursor in degrees.

Specify the *VRAM* where the cursor is drawn with a number, as follows.

0: Character memory

1: Graphic memory

2: Window memory

3: Image memory

4: Shading memory (Cannot be used in the F350-C12E/C41E.)

Either omit the *page#* (default setting 0) or set it to 1.

Specify the drawing method with the *drawing density* or the *drawing mode*. The default value is *drawing mode*, OR.

The *drawing density* parameter specifies the density between 0 and 255 when drawing to the window, image, or shading memory. The default value is 255. The *drawing density* parameter has the following effect when set for the character or graphic memory:

0 : 0 written to memory

Other than 0: 1 written to memory

The *drawing mode* settings operate as follows. (OR is the default setting.)

OR: The current contents of VRAM ORed with 255 are written to memory.

NOT: 0 is written to memory

XOR: The current contents of VRAM are inverted.

When writing to a frame memory, the contents of planes write protected with the MASKBIT command remain unchanged.

CVD ConVert to Double

(Function)

Action Converts an 8-byte character string to a double-precision real number.

Format CVD (*8-byte character string*)

Description The CVD function returns the value of an *8-byte character string* converted to a double-precision real number.

Because numeric data cannot be handled in a random access file, a a double-precision real number must be converted to a character-type numeric data variable (character string) with the MKD$ function before it is written to a random access file. The CVD function converts character-type numeric data read from a random access file back to a double-precision real number.

The CVD function is the opposite of the MKD$ function. The MKD$ function converts numeric data to a character string.

Character-type numeric data is read from a random access file with the GET# command to the variable areas defined with the FIELD# command in the file buffer.

CVI ConVert to Integer

(Function)

Action Converts a 2-byte character string to an integer.

Format CVI (*2-byte character string*)

Description The CVI function returns the value of a *2-byte character string* converted to an integer.

Because numeric data cannot be handled in a random access file, an integer must be converted to a character-type numeric data variable (character string) with the MKI$ function before it is written to a random access file. The CVI function converts character-type numeric data read from a random access file back to an integer.

The CVI function is the opposite of the MKI$ function. The MKI$ function converts numeric data to a character string.

Character-type numeric data is read from a random access file with the GET# command to the variable areas defined with the FIELD# command in the file buffer.

CVL ConVert Long integer

(Function)

Action Converts a 4-byte character string to a long integer.

Format CVL (*4-byte character string*)

Description The CVL function returns the value of a *4-byte character string* converted to a long integer.

Because numeric data cannot be handled in a random access file, a long integer must be converted to a character-type numeric data variable (character string) with the MKL$ function before it is written to a random access file. The CVL function converts character-type numeric data read from a random access file back to long integer data.

The CVL function is the opposite of the MKL$ function. The MKL$ function converts numeric data to a character string.

Character-type numeric data is read from a random access file with the GET# command to the variable areas defined with the FIELD# command in the file buffer.

CVS ConVert to Single

(Function)

Action Converts an 4-byte character string to a single-precision real number.

Format CVS (*4-byte character string*)

Description The CVS function returns the value of an *4-byte character string* converted to a single-precision real number.

Because numeric data cannot be handled in a random access file, a a single-precision real number must be converted to a character-type numeric data variable (character string) with the MKS$ function before it is written to a random access file. The CVS function converts character-type numeric data read from a random access file back to a single-precision real number.

The CVS function is the opposite of the MKS$ function. The MKS$ function converts numeric data to a character string.

Character-type numeric data is read from a random access file with the GET# command to the variable areas defined with the FIELD# command in the file buffer.

DATA DATA

(Command)

Action Defines the integer constants and character constants read with the READ command.

Format DATA *constant* [, *constant* [, *constant*...]]

Description The DATA statement is not executable. It can be declared anywhere in the program.

The *constants* (numeric and character) are delimited by commas (,).

A *constant* cannot be defined as a constant expression.

When one of the character strings described below is declared as a *constant*, it must be enclosed in double quotations (”).

- A character string containing a meaningful symbol such as a comma (,), colon (:), or period (.)
- A character string starting or ending with a required space.

The constants defined with the DATA command are read sequentially to the variables defined with the READ command.

Normally the DATA statement constants and the READ statement variables have the same format. However, it is possible to specify character variables for the READ statement which are read as the character-type numeric variables in the corresponding DATA statement.

A RESTORE command before the READ command specifies the line containing the DATA statement to read. If the DATA statement is not specified by a RESTORE command, the position of the data read depends on the execution status of the READ command.

DATE$ DATE $

(Command-Function)

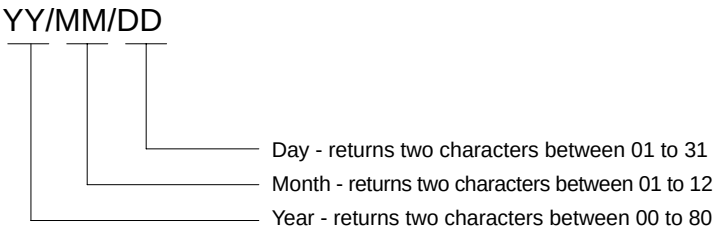
Action Displays and sets the date in the internal clock.

Format Format 1: DATE$

Format 2: DATE$="*yy/mm/dd*"

Description Format 1 is used to read the date from the F350's system clock.

A character string with the year, month, and day (yy/mm/dd) delimited by slashes is returned when the DATE$ function is executed. The format is shown below. The last two digits of the year are returned.



Format 2 is used to set the date in the F350's system clock. The system clock handles dates from Jan. 1, 1980 (80/01/01) to Dec. 31, 2079 (79/12/31).

DBOUT Data Block OUT

(Command)

Action Outputs a data sequence to a parallel port.

Format DBOUT *data#, array,* [, [*size*] [, [*handshake*] [, [*output period*] [, [*GATE rise time*] [, [*GATE ON time*] [, *timeout*]]]]]]

Description The DBOUT command outputs data consecutively to the output port of the Terminal Block Unit or Parallel I/O Unit.

Specify the number of data elements to be output with the *data#* parameter. The specified number of data elements will be output from the beginning of the *array*. If the *data#* parameter is set to 0, the data stored in the buffer will be cleared and the data output will stop.

Specify the name of the one-dimensional array containing the data with the *array* parameter.

Specify the number of data bits to be output (normally 8, 16, or 32) with the *size* parameter.

The output data is rounded to the nearest integer and the integer part is output in the 2's complement format.

Specify whether there is a handshake or not with the *handshake* parameter. Set the handshake to 0 to indicate no handshake, 1 to indicate a handshake. When there is no handshake, specify the data output interval in ms (0.1 to 6553.5 ms) with the *output period* parameter.

Specify the time it takes for the GATE signal to go ON after the data is output in ms (0.3 to 6553.5 ms) with the *gate rise time* parameter. Specify the time that the GATE signal will remain ON in ms (0.1 to 6553.5 ms) with the *gate ON time* parameter.

This command just transfers the data to the system's internal buffer; the output to the parallel port is performed in the background. When the specified number of elements of data can't be stored in the buffer, the command waits until all of the data is stored in the buffer.

When any parameters are omitted, the setting set in the Setup menu will be used.

An error will occur if the output period □ gate rise time + gate ON time.

An error will occur if neither a Terminal Block Unit nor a Parallel I/O Unit is connected.

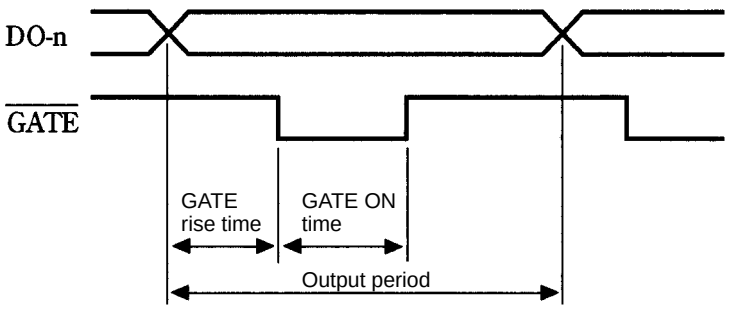
The output timing might be disrupted if a GATE command is executed while data is being output with the DBOUT command, or vice-versa.

An error will occur if the number of bits output is greater than the number specified in the *size* parameter. When the *size* parameter is set to 8, 255 will be output.

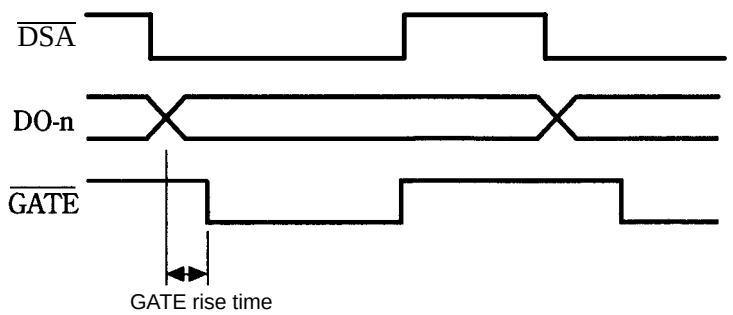
An error will occur when data is being output with a handshake and the timeout time elapses before a DSA signal is input.

If the *size* parameter setting is different from the previous setting, any data remaining in the output buffer will be cleared.

The following diagram shows output timing without a handshake.



The following diagram shows output timing with a handshake.



DEF FN DEFine FuNction

(Command)

Action Defines a user function.

Format DEF FN *name* [(*parameter* [, *parameter*...])] = *expression*

Description The DEF function defines a function and its name.

Function names are subject to the same restrictions as variable names.

The *parameters* have the same names as the variables used inside the function. These variable names are used only for evaluation of the expression inside the function. The same variable names can be used elsewhere inside the program.

The expression declares the operation of the function. It must not exceed one line.

The function is called in the format: FN *name* (variable). It is not necessary for the variable name used to match the name used in the expression declaring the operation of the function, however the variable format must be the same.

The function must be declared before it can be called.

DEF FN...END DEF DEFine FuNction END DEFine

(Command)

Action Defines a user function block.

Format DEF FN *name* [(*parameter* [, *parameter...*])]
statement in DEF FN block
END DEF

Description Defines a user function block.

The function names must conform to restrictions covering variable names.

The *parameters* have the same names as the variables used inside the function. These variable names are used only for evaluation of the expression inside the function. The same variable names can be used elsewhere inside the program.

It is possible to describe operation expressions as statements in more than one line in a DEF FN statement block.

Use the format FN *name (variable)* to have access to a function. A DEF FN statement must exist before the above format.

No further DEF FN ... END DEF declaration can be nested inside a DEF FN statement block. The GOTO command must not be used to jump into or out of a DEF FN statement block.

Use the EXIT DEF command to get out of a DEF FN statement block.

DEFDBL DEFine DouBLe

(Command)

Action Declares variables as double-precision real number variables.

Format DEFDBL *character* [–*character*] [,*character* [–*character*]...]

Description Declares variables that begin with the character specified in the *character* parameter as double-precision real number variables.

Several characters can be specified in *character* parameters separated by commas. A range of consecutive characters can be specified by using a hyphen. For example, the parameter *E–G* would declare all variables beginning with E, F, or G as double-precision real number variables.

The type statements (%, &, !, #, $) take priority over type declarations with this command.

DEFGLOBAL DEFine GLOBAL variable

(Command)

Action Declares global variables to be used in a structural subroutine.

Format DEFGLOBAL *character* [–*character*] [,*character* [–*character*]...]

Description Among the variables that appear before the end of the structural subroutine, any variables that begin with the character specified in the *character* parameter will be treated as global variables.

It is also possible to create new global variables within the structural subroutine. In this case, the variables will retain their values after the end of the structural subroutine.

The DEFGLOBAL command is effective from the line where it is executed until the end of structural subroutine.

When a variable is treated as a global variable in another structural subroutine, it must be declared by a DEFGLOBAL command in each subroutine.

DEFINT DEFine INTeger

(Command)

Action Declares variables as integers.

Format DEFINT *character* [–*character*] [,*character* [–*character*]...]

Description Declares variables that begin with the character specified in the *character* parameter as integer variables.

Several characters can be specified in *character* parameters separated by commas. A range of consecutive characters can be specified by using a hyphen. For example, the parameter *E–G* would declare all variables beginning with E, F, or G as integer variables.

The type statements (%, &, !, #, $) take priority over type declarations with this command.

DEFLNG DEFine LoNG

(Command)

Action Declares variables as long integers.

Format DEFLNG *character* [–*character*] [,*character* [–*character*]...]

Description Declares variables that begin with the character specified in the *character* parameter as long integer variables.

Several characters can be specified in *character* parameters separated by commas. A range of consecutive characters can be specified by using a hyphen. For example, the parameter *E–G* would declare all variables beginning with E, F, or G as long integer variables.

The type statements (%, &, !, #, $) take priority over type declarations with this command.

DEFSNG DEFine SiNGle

(Command)

Action Declares variables as single-precision real number variables.

Format DEFSNG *character* [–*character*] [,*character* [–*character*]...]

Description Declares variables that begin with the character specified in the *character* parameter as single-precision real number variables.

Several characters can be specified in *character* parameters separated by commas. A range of consecutive characters can be specified by using a hyphen. For example, the parameter *E–G* would declare all variables beginning with E, F, or G as single-precision real number variables.

The type statements (%, &, !, #, $) take priority over type declarations with this command.

DEFSTR DEFine STRing

(Command)

Action Declares variables as string variables.

Format DEFSTR *character* [–*character*] [,*character* [–*character*]...]

Description Declares variables that begin with the character specified in the *character* parameter as character string variables.

Several characters can be specified in *character* parameters separated by commas. A range of consecutive characters can be specified by using a hyphen. For example, the parameter *E–G* would declare all variables beginning with E, F, or G as string variables.

The type statements (%, &, !, #, $) take priority over type declarations with this command.

DELETE DELETE

(Command)

Action Deletes all lines in a specified range of the program.

Format DELETE [*line# 1*] [–*line# 2*]

Description Deletes all lines between *line# 1* and *line# 2*.

Only *line# 1* is deleted if *line# 2* is not specified.

Lines from the start of the program to line# 2 are deleted if *line# 1* is not specified.

Use a period (.) instead of *line# 1* or *line# 2* to specify the current program line. The current line is indicated by the pointer. The current line is the last line input during program creation or the last line displayed with the LIST command.

It is not possible to omit both the *line# 1* and *line# 2* parameters.

DEVICE DEVICE

(Command)

Action Specifies the standard OVL input and output devices.

Format DEVICE [*"KYBD:"* ***or*** *"COM:"*] [,*"SCRN:"* ***or*** *"COM:"*]

Description The input device can be selected as the keyboard or RS-232C port. The output device can be selected as the video monitor or the RS-232C port.

When the F350 is connected to a computer via the RS-232C port, the program can be created using the computer as a terminal.

The first parameter specifies the input device, as one of the following.

KYBD: Keyboard

COM: RS-232C

Input from the keyboard is disabled if the input device is set to RS-232C. Part of the screen edit function can't be used if the input device is set to RS-232C.

The second parameter specifies the input device, as one of the following.

SCRN: Video monitor

COM: RS-232C

Subsequent characters are output to the RS-232C port if the output device is set to RS-232C. Screen control can't be used if the output device is set to RS-232C.

The previous setting is used if either parameter is omitted; both parameters can't be omitted.

DILA DILAte

(Command)

Action Expands a binary image.

Format DILA [*page#*], *binary image plane#* [, [*X1*] [, [*Y1*] [, [*X2*] [, [*Y2*] [, [*number of connections*] [, *repetitions*]]]]]]

Description DILA expands the binary image in the region defined by (*X1*, *Y1*) and (*X2*, *Y2*) in image memory.

Specify the image memory page number (0 or 1) with the *page#* parameter. (default is #0)

Specify the plane in image memory in which the expansion will be performed with the *binary image plane#* parameter.

Specify the upper-left and lower-right corners of the rectangular region with *(X1*, *Y1)* and *(X2*, *Y2)*. The default settings are (1, 1) and (510, 510).

Specify the *number of connections* as follows. (The default setting is 0.)

0: 8
Other than 0: 4

Specify the number of times that the processing will be performed with the *repetitions* parameter.

If adjacent pixels in the rectangular region have pixel values of 0 and 1, the processing will produce a pixel value of 1.

Expansion can't be performed beyond the specified rectangular region. Processing will end before the specified number of repetitions if another expansion can't be performed.

DIM DIMension

(Command)

Action Defines an array variable.

Format DIM *variable name* (*subscript max. value* [, *subscript max. value* ...]) [, *variable name* (*subscript max. value* [, *subscript max. value*...])]

Description Declares the variable format, number of dimensions, and *subscript maximum values* of an array variable and assigns the array variable to an area of memory.

The array variable format is specified with the type statements (%, !, #, $, &). The default type if the type statement is omitted is a single-precision real number variable.

The number of *subscript maximum values* indicates the number of dimensions of the array. If multiple dimensions are specified, the *subscript maximum values* are delimited by commas (,).

The *subscript maximum value* parameters specify the maximum value a subscript can have. The subscript minimum value can be specified as 0 or 1 with the OPTION BASE statement.

The default value is 0 if no OPTION BASE statement is specified. Note that executing the command DIM A(20) if no OPTION BASE statement is specified in the program results in an array with elements from 0 to 20, that is, 21 elements.

DIN Data IN

(Function)

Action Reads the status of the Terminal Block Unit or Parallel I/O Unit input port.

Format DIN (*bit address* [, *size*])

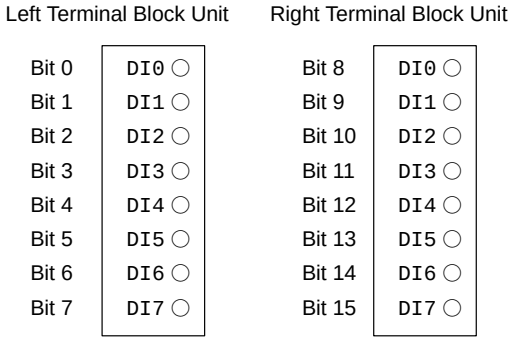
Description The DIN function reads the data input to the Terminal Block Unit or Parallel I/O Unit input ports.

Specify the number of input data bits with the size parameter. The default value is 8. The data input start position can be specified with the bit address parameter.

Data is input as signed integers (2's complement format).

If both Terminal Block Units and Parallel I/O Units are connected, the DIN function does not differentiate between the two. The maximum number of input bits becomes the total number of the input bits for all Units.

Bit addresses are allocated in order from the left Unit, as shown in the following diagram. In this example, the status of all 16 terminals can be read with a DIN(0,16) command.



DISPLAY DISPLAY

(Command)

Action Sets the display image.

Format DISPLAY *display image* [*, image pathway*]

Description The DISPLAY command controls the overlapping of images displayed on the video monitor.

Set the *display image* a decimal number between 0 and 31. This number is converted to binary and the image is displayed if its corresponding bit is set to 1. The relationship between the bits and displayed images is shown below.

Bit 0: Camera image

Bit 1: Paint window memory/pattern matching memory image

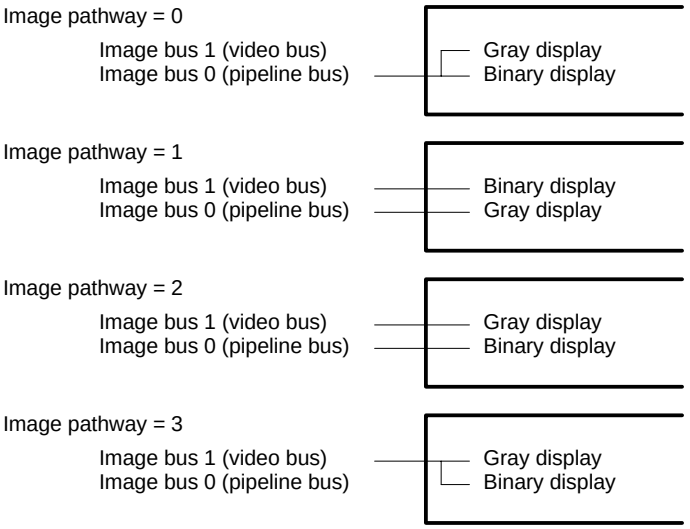
Bit 2: Window memory image

Bit 3: Graphic memory image

Bit 4: Character memory image

Set bit 0 to 1 when displaying window memory.

The *image pathway* parameter specifies the image bus taken by the displayed image. The default value is 2.



DO–LOOP REPEAT

(Command)

Action Repeats the statement between DO and LOOP the specified number of repetitions.

Format

DO

Statements in DO block

LOOP REPEAT *number of times*

Description The DO–LOOP REPEAT command executes the statement between DO and LOOP the specified *number of times*.

The *statements in the DO block* are executed the *number of times* specified by the (*number of times* parameter + 1).

The GOTO command must not be used to jump into or out of a DO block. Use the EXIT DO command to get out of a DO block.

DO–LOOP UNTIL

(Command)

Action Repeats the statements between DO and LOOP until the condition is fulfilled.

Format

DO

Statements in DO block

LOOP UNTIL *conditional expression*

Description The DO–LOOP UNTIL command executes the statement between DO and LOOP until the condition is fulfilled.

The statements in the DO block are executed repeatedly while the conditional expression is false.

The GOTO command must not be used to jump into or out of a DO block. Use the EXIT DO command to get out of a DO block.

DO–LOOP WHILE DO–LOOP WHILE

(Command)

Action Repeats the statements between DO and LOOP while the condition is fulfilled.

Format

DO

Statements in DO block

LOOP WHILE *logical expression*

Description The DO–LOOP WHILE command executes the statement between DO and LOOP while the condition is fulfilled.

The statements in the DO block are executed repeatedly while the logical expression is true.

The GOTO command must not be used to jump into or out of a DO block. Use the EXIT DO command to get out of a DO block.

DO REPEAT–LOOP DO REPEAT–LOOP

(Command)

Action Repeats the statements between DO and LOOP the specified number of times.

Format

DO REPEAT *number of times*

Statements in DO block

LOOP

Description The DO REPEAT and LOOP command executes the statement between DO and LOOP the specified *number of times*.

The statements in the DO block are executed the specified number of times.

The GOTO command must not be used to jump into or out of a DO block. Use the EXIT DO command to get out of a DO block.

DO UNTIL–LOOP DO UNTIL–LOOP

(Command)

Action Repeats the statements between DO and LOOP until the condition is fulfilled.

Format

DO UNTIL *logical expression*

Statements in DO block

LOOP

Description The DO UNTIL–LOOP command executes the statement between DO and LOOP until the condition is fulfilled.

The statements in the DO block are executed repeatedly while the logical expression is false.

The GOTO command must not be used to jump into or out of a DO block. Use the EXIT DO command to get out of a DO block.

DO WHILE–LOOP DO WHILE–LOOP

(Command)

Action Repeats the statements between DO and LOOP while the condition is fulfilled.

Format

DO WHILE *logical expression*

Statements in DO block

LOOP

Description The DO WHILE–LOOP command executes the statement between DO and LOOP while the condition is fulfilled.

The statements in the DO block are executed repeatedly while the logical expression is true.

The GOTO command must not be used to jump into or out of a DO block. Use the EXIT DO command to get out of a DO block.

DOUT Data OUTput

(Command)

Action Outputs data to a Terminal Block Unit or Parallel I/O Unit.

Format DOUT *output data* [, [*bit address*] [, *size*]]

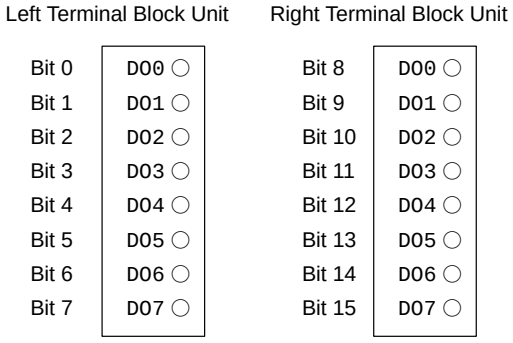
Description The DOUT function outputs the data specified with the *output data* parameter to the output port of the Terminal Block Unit or Parallel I/O Unit.

If both Terminal Block Units and Parallel I/O Units are connected, the DOUT function does not differentiate between the two. The maximum number of output bits becomes the total number of the output bits for all units.

The data output start position (0 to (max. no. of output bits –1)) can be specified with the *bit address* parameter. The default value is 0.

Specify the number of data bits to define the output data with the *size* parameter. The default value is 16.

Bit addresses are allocated in order from the left Unit, as shown below.



DSA Data Set Acknowledge

(Function)

Action Reads the status of the DSA signal.

Format DSA

Description The DSA function reads the status of the Terminal Block Unit or Parallel I/O Unit DSA signal, and returns a value as follows:

1: DSA signal is ON.
0: DSA signal is OFF.

When multiple I/O units are used simultaneously, this function reads the DSA signal status from the unit with the lowest slot address.

DSKF DiSK Function

(Function)

Action Determines the free space remaining in the memory card.

Format DSKF (*drive*)

Description Determines the number of free bytes remaining in the disk specified by the drive parameter.

Specify the memory card drive name (C:) in the *drive* parameter.

EDGRJECT EDGe ReJECT

(Command)

Action Removes the edge surrounding a binary image.

Format EDGRJECT [*page#*], *binary image plane#* [,[*X1*] [,[*Y1*] [,[*X2*] [,[*Y2*] [,*number of connections*]]]]]

Description EDGRJECT removes the edge around the binary image in the rectangular region defined by points (*X1*, *Y1*) and (*X2*, *Y2*) in image memory.

Specify the image memory page number (0 or 1) in the *page#* parameter. The default setting is 0.

The *binary image plane#* is used to specify the binary image plane number for the image memory that is to be processed.

Specify the upper-left and lower-right corners of the rectangular region in points *(X1*, *Y1)* and *(X2*, *Y2)*. The default settings are (1, 1) and (510, 510).

Specify the *number of connections* as follows. The default setting is 0.

0: 8
Other than 0: 4

A "Vision error" will occur if the number of elements in the graphic's outline exceeds 255. A "Vision error" will also occur if the length of the graphic's outline exceeds 4,096.

EDIT EDIT

(Command)

Action Selects the mode to edit lines of the program.

Format EDIT *line number* ***or*** *label* ***or*** .

Description Displays the line with the specified line number and selects the Edit mode.

In the Edit mode, the lines are edited with the ROLL-UP, ROLL-DOWN, and other Keys. Use a period (.) to specify the current program line.

The current line is indicated by the pointer. The current line is the last line input during program creation or the last line displayed with the LIST command.

Program contents can't be edited with the EDIT command if a password has been set with the PASSWD command.

ELIM ELIMinate

(Command)

Action Removes isolated points from a binary image.

Format ELIM [*page#*], *binary image plane#* [,[*X1*] [,[*Y1*] [,[*X2*] [,[*Y2*] [,*number of connections*]]]]]

Description ELIM eliminates isolated points from the binary image in the rectangular region defined by points (*X1*, *Y1*) and (*X2*, *Y2*) in image memory.

Specify the image memory page number (0 or 1) in the *page#* parameter. The default setting is 0.

The *binary image plane#* is used to specify the binary image plane number for the image memory that is to be processed.

Specify the upper-left and lower-right corners of the rectangular region in points *(X1*, *Y1)* and *(X2*, *Y2)*. The default settings are (1, 1) and (510, 510).

Specify the *number of connections* as follows. The default setting is 0.

0: 8
Other than 0: 4

ELLIPSE ELLIPSE

(Command)

Action Draws an ellipse in VRAM.

Format ELLIPSE *X, Y, XR, YR, VRAM* [, [*page#*][, [*drawing density* ***or*** *drawing mode*] [*, lineart*]]

Description The ELLIPSE command draws an ellipse around the center coordinates (X,Y) with the specified horizontal axis (XR), and vertical axis (YR).

Specify the *VRAM* where the ellipse is drawn with a number, as follows:

0: Character memory
1: Graphic memory
2: Window memory
3: Image memory
4: Shading memory (Cannot be used in the F350-C12E/C41E.)

Either omit the *page#* (default setting 0) or set it to 1.

Specify the drawing method with the *drawing density* or the *drawing mode*. The default value is *drawing mode*, OR.

The *drawing density* parameter specifies the density between 0 and 255 when drawing to the window, image, or shading memory. The default value is 255. The *drawing density* parameter has the following effect when set for the character or graphic memory:

0 : 0 written to memory
Other than 0: 1 written to memory

The *drawing mode* settings operate as follows. (The default setting is OR.)

OR: The current contents of VRAM ORed with 255 are written to memory.

NOT: 0 is written to memory

XOR: The current contents of VRAM are inverted.

Specify with the *lineart* parameter if the ellipse is an outline only or filled. The default setting is 0 (filled).

0: Filled ellipse
Other than 0: Ellipse outline only

When writing to a frame memory, the contents of planes write protected with the MASKBIT command remain unchanged.

END END

(Command)

Action Stops program execution.

Format END

Description Stops program execution.

All files opened during program execution are closed when the END command is executed. Even if no END command is executed, program execution stops when the last line of the program has been executed.

ENHANCE ENHANCE

(Command)

Action Creates LUT data for contrast modification from histogram array data.

Format ENHANCE *array*, *modification array*

Description The ENHANCE command creates the original contrast-modified histogram from the histogram data obtained with the HISTGRAM command. This contrast-modified histogram is set into a LUT (look-up table) for binary conversion with the SETLUT command. After the camera image is selected with the FILTERIN command and the image is input to the video memory with the VIDEOIN command, the contrast of the image input from the camera is improved before the image is input to the image memory.

Use the *array* parameter to specify the name of the array containing the histogram obtained with the HISTGRAM command.

Specify the name of the array to store the converted data with the *modification array* parameter.

The array to store the modified data must contain at least 256 elements.

EOF End Of File

(Function)

Action Determines if the file is at the end of the file.

Format EOF (*file#*)

Description Determines whether the end of the file specified by *file#* has been reached.

The EOF function returns 0 or –1.

0 : The end of the file hasn't been reached.
–1 : The end of the file has been reached.

The EOF function can be used for sequential access files and communication ports. Sequential access files on the memory card must be opened in the INPUT mode.

Specify the number of the file opened with the OPEN command in the *file#* parameter.

ERASE ERASE

(Command)

Action Deletes an array defined with the DIM command.

Format ERASE *array* [, *array...*]

Description The DIM command deletes the array with the name specified by the *array* parameter. The amount of memory equivalent to the size of the file is then available to store array or string variables.

After an array is deleted, another array with the same name can be created. This allows the size of an array to be changed.

An error will occur if an array with the same name as an existing array is created without first deleting the original array with the ERASE command.

ERL — ERror Line

(Function)

Action Determines the line number where an error occurred.

Format ERL

Description The ERL function returns the line number where an error occurred.

The ERL function is normally used inside an error processing routine in combination with program flow control statements. It is used for flow control in error recovery processing and for recovery after the error is reset.

EROS — EROSion

(Command)

Action Contracts a binary image.

Format EROS [*page#*], *binary image plane#*, [, [*X1*] [, [*Y1*] [, [*X2*] [, [*Y2*] [, [*number of connections*] [, *repetitions*]]]]]]

Description EROS contracts the binary image in the region defined by (*X1*, *Y1*) and (*X2*, *Y2*) in image memory.

Specify the image memory page number (0 or 1) with the *page#* parameter.

Specify the plane in image memory in which the contraction will be performed with the *binary image plane#* parameter.

Specify the upper-left and lower-right corners of the rectangular region with *(X1*, *Y1)* and *(X2*, *Y2)*. The default settings are (1, 1) and (510, 510).

Specify the *number of connections* as follows. (The default setting is 0.)

0: 8
Other than 0: 4

Specify the number of times that the processing will be performed with the *repetitions* parameter.

If adjacent pixels in the rectangular region have pixel values of 0 and 1, the processing will produce a pixel value of 0.

Processing will end before the specified number of repetitions if another contraction can't be performed.

ERR — ERRor code

(Function)

Action Determines the error code after an error occurs.

Format ERR

Description The ERR function returns the error code.

The ERR function is normally used inside an error processing routine in combination with program flow control statements. It is used for flow control in error recovery processing and for recovery after the error is reset.

ERRMSG — ERRor MeSsaGe

(Command)

Action Defines the operation when an error occurs.

Format ERRMSG [*message*] [, [*buzzer*] [, *error signal control*]]

Description The ERRMSG command defines the message format, buzzer alarm, and error signal control method when an error occurs.

The *message* parameter determines whether the error message is displayed in English or Japanese.

0: Japanese
1: English (Default setting at OVL startup.)

The *buzzer* parameter controls buzzer operation.

0: No buzzer when an error occurs. (Default setting at OVL startup.)
1: Sound the buzzer when an error occurs.

The *error signal control* parameter controls the error signal.

0: No error signal when an error occurs. (Default setting at OVL startup.)
1: Turn ON the error signal when an error occurs.

The error signal is turned OFF the next time that a command is executed correctly.

The settings when OVL is started are as follows:

Error messages:	English
Buzzer control:	OFF
Error signal control:	OFF

ERROR ERROR

(Command)

Action Generates a pseudo-error.

Format ERROR *error code*

Description The ERROR command generates the error specified with the error code. When an error is generated which corresponds to a predefined system error code, program execution is interrupted and the appropriate error message is displayed.

Specify the error code as an integer between 0 and 255. This range of integers includes the system error codes.

Refer to *Section 8 Troubleshooting* for details on the error codes.

ERROUT ERRor OUT

(Command)

Action Controls the error output.

Format ERROUT *switch*

Description The ERROUT command controls the ON/OFF status of the power supply unit ERROR signal. Set the *switch* parameter as follows:

0: ERROR signal OFF
Other than 0: ERROR signal ON

EXIT DEF/DO/FOR/SUB EXIT DEF/DO/FOR/SUB

(Command)

Action Exits a control block.

Format Format 1: EXIT DE*F*
Format 2: EXIT DO
Format 3: EXIT FOR
Format 4: EXIT SUB

Description Format 1: Exits a user function.

Format 2: Exits a DO-LOOP loop.

Format 3: Exits a FOR-NEXT loop.

Format 4: Exits a structural subroutine.

EXP — EXPonential

(Function)

Action Determines the value of the natural number e raised to an exponential power.

Format EXP (*numeric expression*)

Description The EXP function returns the value of e raised to the specified power.

The numeric expression may be in the form of an integer, long integer, or a single or double-precision real number.

The LOG function is the opposite of the EXP function. The EXP function can be used to create other mathematical functions, such as the hyperbolic sine function (sinhX).

FCOPY — File COPY

(Command)

Action Copies files.

Format FCOPY *copy source*, *copy destination*

Description Use the FCOPY command to copy files.

Specify the name of the file to be copied with the *copy source* parameter. Be sure to specify the complete pathname (drive and directory path) if necessary. The wildcard character (* and ?) can be used to specify multiple files.

Specify the name of the new file with the *copy destination* (drive and directory path).

FIELD# — FIELD

(Command)

Action Allocates the variable areas to the file buffer of the random access file.

Format FIELD *file#*, *field length AS character variable* [, *field length AS character variable*...]

Description To use a file buffer as the I/O buffer for a random access file, the areas to store character data (variable areas) must be allocated in the file buffer.

Specify the *file#* as the number in which the file was opened with the OPEN statement.

Specify the length allocated for the *character variable* with the *field length* parameter as a positive integer between 1 and 255.

Multiple character variables can be allocated. However, the total of the field lengths specified for all variables must 255 bytes or less. The total of the field lengths specified with a FIELD# command must equal the length of one record for random access file I/O.

The FIELD# command is used only to define the character variable names and variable areas in the file buffer. The LSET and RSET commands are used to set data in the allocated variable areas. The GET# and PUT# statements are used for data random access file I/O. It is not possible to assign data to the character variables defined with the FIELD# command with any commands except LSET and RSET. Multiple FIELD# commands can be executed to allocate different character variables to a single file buffer.

FILES FILES

(Command)

Action Displays the file directory for the specified drive, showing the file names, sizes, and creation dates.

Format FILES ["*drive name*"]

Description The following drives can be specified:

"C:¥" : Memory Card
"D:¥" : ROM Disk
"E:¥" : RAM Disk

The default *drive name* is drive C.

FILTDATA FILTer DATA

(Command)

Action Specifies the line filter factors.

Format FILTDATA *data 0, data 1, ... , data 8* [, [*global factor*] [*, sobel function*]]

Description The FILTERDATA and FILTER commands control the F350's real-time image filtering functions. In order to enable the filter factors set with this command, the FILTER command's *function* parameter must be set to "2: sharpened image" or "5: line filter image."

Filtering is performed with the following equation when the *function* parameter has been set to "2: sharpened image" or "5: line filter image." (The term Mn is each pixel's density.)

$$\text{Center pixel's density} = \frac{(\sum \text{Data n} \times \text{Mn})}{\text{Global factor}}$$

The nine factors are specified with the parameters data 0 to data 8. The positional relationship of these pixels is shown in the diagram below.

Data 1	Data 2	Data 3
Data 4	Data 0	Data 5
Data 6	Data 7	Data 8

The filter factors from data 1 to data 8 can be set to the values 0, ±1, ±2, or ±4. The filter factor for data 0 can be set to 0, ±1, ±2, ±4, or ±8.

A filter result of 256 or higher is rounded to 255 while negative values are rounded to 0.

In cases where it is desirable to restrict filtering results to 255, the filtering results can be divided by the global factor.

The *global factor* parameter can be set to 0, 1, 2, 4, 8, or 16. The output becomes 0 if the global constant is set to 0. The default value is 1.

The *sobel function* parameter determines whether or not sobel integration processing is performed. Set this parameter to 0 to disable sobel processing, 1 to enable sobel processing. The default setting is 0.

When sobel processing is enabled, line filtering is performed on the original image and then the results of sobel processing are added to the resulting image. The sobel function is effective only when the *function* parameter has been set to "2: sharpened image" in the FILTER command.

FILTER FILTER

(Command)

Action Sets the image filtering function and LUT function.

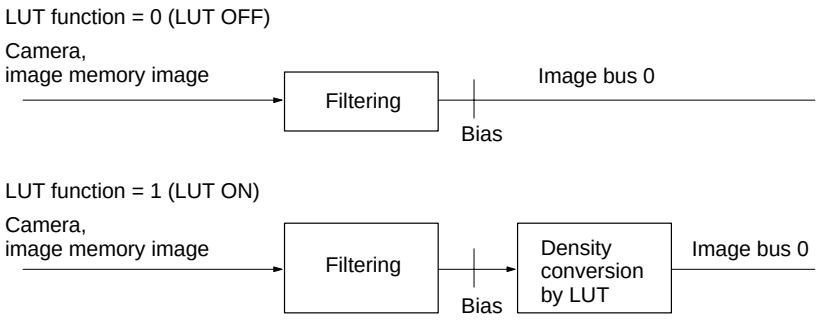
Format FILTER *function* [, [*LUT function,*] [*, [bias*] [*, peripheral pixels*]]]

Description The FILTER command controls the F350's real-time image filtering functions.

Set the *function* parameter to a number corresponding to one of the following filtering methods:

0: raw image
1: shading memory image
2: sharpened image
3: shading compensation image
4: sobel image
5: line filter image
6: (not used)
7: compound edge image

Set *LUT function* to 1 to enable the LUT function, which converts the pixel densities in filtering. The default setting is 0.



The *peripheral pixels* parameter defines the output density (as 0 or 255) of the 2-pixel unstable border which arises due to filtering.

Peripheral pixels setting	Output
0	Output with an output density of 0.
255	Output with an output density of 255.
Other than 0 or 255 (default setting)	Output unchanged.

The specified *bias* value is added to the image density after filtering. Specify a value between –128 and 127.

FILTERIN FILTER IN

(Command)

Action Selects the input path of the image to be displayed and measured.

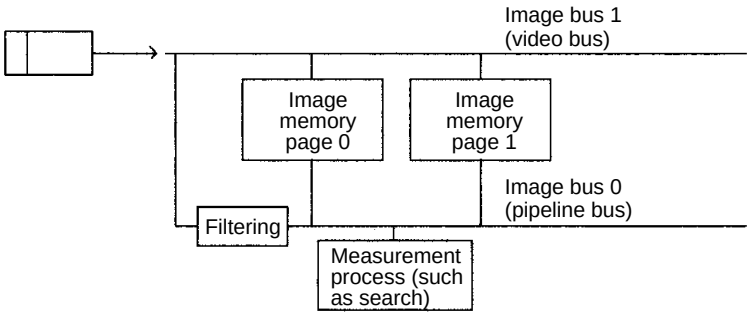
Format FILTERIN *path* [*, page#*]

Description The FILTERIN command selects the camera image or an image from image memory as the image to be displayed and measured. This image is subjected to the selected filtering function.

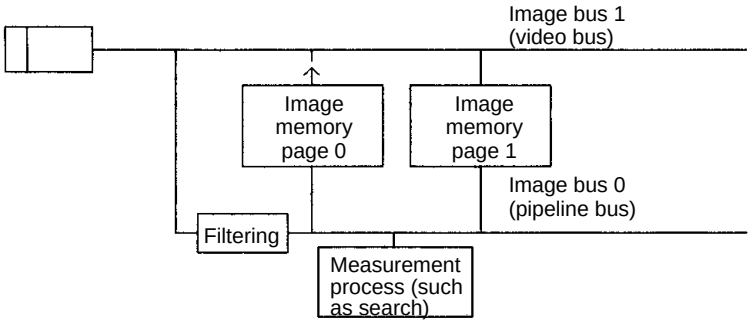
Set the path parameter to 0 to select the camera image or 1 to select the image from the image memory. The image selected by *the path* parameter is output to image bus#1 (video bus) and is also subjected to filtering before being output to image bus#0 (pipeline bus).

Either omit the *page#* (default setting 0) or set it to 1.

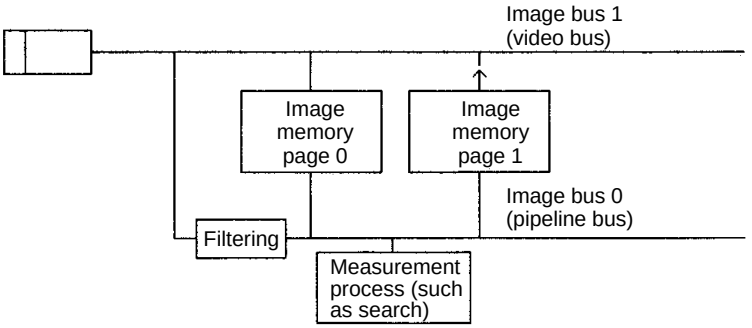
The following diagram shows the operation when the path=0. (The camera's input image flows to image buses 0 and 1.)



The following diagram shows the operation when the path=1 and the page#=0. (The image from page 0 in image memory flows to image buses 0 and 1.)



The following diagram shows the operation when the path=1 and the page#=1. (The image from page 1 in image memory flows to image buses 0 and 1.)



FILTSEL FILTer SELect

(Command)

Action Selects the type of image filtering.

Format FILTSEL *filter type* [*, image type*]

Description The FILTSEL command selects the type of filtering that will be performed. This allows the user to select frequently used filtering functions easily, without using the FILTER and FILTDATA commands.

Set the *filter type* parameter to one of the following filtering types:

0: OFF
1: Weak smoothing
2: Strong smoothing
3: Edge enhancement 1
4: Edge enhancement 2
5: Edge enhancement 3
6: Edge enhancement 4
7: Edge enhancement 5
8: Relief
9: Vertical edge
10: Horizontal edge
11: Edge detection

When the FILTSEL command is executed, the filtering settings set with the FILTER and FILTDATA commands are overwritten and invalid. The LUT settings will be changed if “8: Relief,” “9: Vertical edge,” or “10: Horizontal edge” is selected.

Specify whether the image is a gray or binary image with the *image type* parameter. The default setting is 0.

0: Gray image
Other than 0: Binary image

If a binary image is selected with the *image type* parameter, the binary level must be set with a command such as the LEVEL command.

FIND FIND

(Command)

Action Searches for a specified character string and displays the line containing the character string.

Format Format 1: FIND *search character string* [*, line# 1*] [– [*line# 2*]]] [*, A*]

Format 2: FIND

Description The FIND command searches for the specified *search character string* between *line#1* and *line#2* and displays the line containing the character string. If more than one line contains the specified *search character string*, the first line found is displayed.

Input the FIND command with no parameters to continue a previous find operation.

If the *A* option is specified, each line containing the specified *search character string* is displayed.

The FIND command does distinguish upper-case and lower-case characters.

Program contents can’t be searched with the FIND command if a password has been set with the PASSWD command.

FIX FIX

(Function)

Action Truncates the portion of a number past the decimal point and returns an integer.

Format FIX (*numeric expression*)

Description The FIX function truncates the non-integer portion of the specified *numeric expression* and returns an integer.

The INT and CINT functions are similar to the FIX function. Like the FIX function, the INT function also truncates the remainder of the specified value, but the INT function never returns a value larger than the value specified with the *numeric expression*. The CINT function rounds off the decimal places of the specified *numeric expression* (i.e., up or down to the nearest integer) and returns an integer.

FLASH FLASH

(Command)

Action Flashes the strobe and simultaneously reads the image to the Camera I/F Unit internal memory.

Format FLASH *mode* [*, interval*]

Description The FLASH command flashes the strobe and inputs the image to the Camera I/F Unit internal memory.

Set the mode parameter to one of the following values:

0: Stop strobe flashes
1: Strobe flash on FLASH command only
2: Synchronize the strobe flash with the to STEP input.
3: Strobe flash at specified intervals

The *interval* parameter setting is only valid when mode 3 (strobe flash at specified intervals) is selected. The interval is specified as a number of fields between 0 and 65535 fields. The default value is 2 fields (1 frame). Operation stops if a interval of 0 is specified.

To set a strobe flash every second, execute FLASH 3,60.

If a command that controls strobe operation has been executed, such as (MEASURE), SMGRUN, SMRUN, SQRUN, or VIDEOIN, the strobe flashes of that command have priority over the FLASH command.

FMODE Frame/field MODE

(Command)

Action Sets frame mode or field mode for image input or measurement.

Format FMODE *mode*

Description The FMODE command specifies whether the initial mode for image input or measurement is frame mode (512×512) or field mode (512×256).

Set the *mode* parameter to zero to select frame mode or an other-than-0 value to select field mode.

0: Frame mode (512×512)
Other than 0: Field mode (512×256)

Frame mode is the default setting when OVL is started. Once the mode has been set with FMODE, that mode setting is valid until it is changed with FMODE.

The mode set with FMODE will be used for the input mode in the (MEASURE), SMGRUN, SMRUN, and VIDEOIN commands if the mode setting is omitted when these commands are executed.

The following table shows the differences between frame mode and field mode.

Mode		Resolution	Image input time (excluding wait time)
0	Frame mode	512×512 pixels	33.3 ms
1	Field mode	512×256 pixels	16.7 ms

Frame mode is normally used, but field mode can be used to reduce the processing time.

FOR..TO..STEP–NEXT FOR..TO..STEP–NEXT

(Command)

Action Repeats the commands between the FOR and NEXT statements.

Format FOR *numeric variable = initial value* TO *final value* [STEP *increment*]
to
NEXT [*numeric variable* [, *numeric variable* ...]]

Description The FOR–NEXT loop is formed by a series of statements starting with the FOR statement and ending with the NEXT statement. The series of statements inside the FOR–NEXT loop is executed the specified number of times. Normally, each FOR statement has a corresponding NEXT statement.

The specified *numeric variable* counts the number of times the FOR–NEXT loop is executed. Consequently, the same *numeric variable* name must be specified for both the FOR and NEXT statements.

The units to calculate the loop execution can be specified with STEP and an *increment* parameter. Both STEP and the *increment* parameter can be omitted. The default value of the *increment* is +1. Each time the FOR–NEXT loop is executed, the increment is added to the current value of the *numeric variable* (starting from the *initial value* the first time the loop is executed) then assigned to the *numeric variable* again.

FORMAT FORMAT

(Command)

Action Formats the memory card.

Format FORMAT *drive name* [, *confirmation*]

Description The FORMAT command initializes (formats) an SRAM memory card or a RAM Disk (C41E only). Specify the drive name as follows:

"C:" : Memory card drive
"E:" : RAM disk drive

The F350 can handle memory cards with a capacity up to 2 Mbytes.

Memory cards can also be formatted from the Setup menu.

If a 0 is entered for the *confirmation* parameter, the confirmation prompt "Are you sure? (Y/N)," will be displayed when FORMAT is executed. Press Y to proceed with the formatting, N to cancel.

If a other-than-0 value is entered for the *confirmation* parameter, the initialization will be performed immediately.

FRE FREe memory

(Function)

Action Determines the amount of free space in each control area.

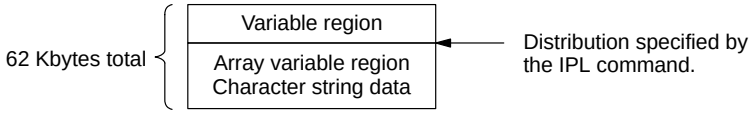
Format FRE (*function*)

Description The FRE function determines the free memory area in bytes of the control area specified with the function parameter.

Specify the function parameter as a value between 0 and 3, as follows:

0: Determines the free area in variable table region.
1: Determines the free area in array variable/character string data region.
2: Determines the free area in program region.
3: Determines the total free area for functions 0 to 2.

The F350 has 62 Kbytes in the variable region. The IPL command determines the distribution to the array variable/character string data region and other regions.



The F350 has 63 Kbytes in the OVL program region.

GATE GATE

(Command)

Action Controls the GATE signal.

Format GATE [*switch*]

Description The GATE command controls the ON/OFF status of the GATE signal in the Terminal Block Units and Parallel I/O Units. Set the *switch* parameter as follows:

0: GATE signal OFF
Other than 0: GATE signal ON

When multiple units are connected, the GATE signals are controlled simultaneously for all units.

GCOPY Gray COPY

(Command)

Action Copies the raw image in VRAM.

Format GCOPY *VRAM1*, [*page# 1*], *VRAM2*, [*page# 2*] [, [*X1*] [, [*Y1*] [, [*X2*] [, [*Y2*] [, [*X*] [, [*Y*]]]]]]]

Description The GCOPY command copies the raw image in VRAM.

Specify the copy source and destination VRAMs (*VRAM1* and *VRAM2*). A plane VRAM cannot be specified. Use the numbers as follows:

2: Window memory
3: Image memory
4: Shading memory (Cannot be used in the F350-C12E/C41E.)

Either omit *page# 1* and *page# 2* (default setting 0) or set them to 1.

Specify the top-left coordinates of the copy source rectangle in *(X1*, *Y1)* and the bottom-right coordinates in *(X2*, *Y2)*. The default values for *X1*, *Y1* are 0, 0 and the default values for *X2*, *Y2* are 511, 511. Specify the top-left coordinates of the copy destination rectangle in *(X*, *Y)*. The default values for *X*, *Y* are 0, 0.

The contents of planes write protected with the MASKBIT command remain unchanged.

GCOPY2 Gray COPY 2

(Command)

Action Makes enlarged and reduced copies of raw images in VRAM.

Format GCOPY2 *VRAM1,* [*page# 1*]*, VRAM2,* [*page# 2*] [, [*XS1*] [, *[YS1*] [, [*XS2*] [, [*YS2*] [, [*XD1*] [, [*YD1*] [, [*XD2*] [, [*YD2*]]]]]]]]

Description The GCOPY2 command makes enlarged or reduced copies of the raw image in VRAM.

Specify the copy source and destination VRAMs (*VRAM1* and *VRAM2*). A plane VRAM cannot be specified. Use the numbers as follows:

2: Window memory
3: Image memory
4: Shading memory (Cannot be used in the F350-C12E/C41E.)

Omit *page# 1* and *page# 2* or set them to 0.

Specify the top-left coordinates of the copy source rectangular region as *XS1*, *YS1* and the bottom-right coordinates as *XS2*, *YS2*. The default values for *XS1*, *YS1* are 0, 0 and the default values for *XS2*, *YS2* are 511, 511.

Specify the top-left coordinates of the copy destination rectangular region as *XD1*, *YD1* and the bottom-right coordinates as *XD2*, *YD2*. The default values for *XD1*, *YD1* are 0, 0 and the default values for *XD2*, *YD2* are 511, 511.

The contents of planes write protected with the MASKBIT command remain unchanged.

GDILA Gray DILAte

(Command)

Action Expands the gray image in the specified region.

Format GDILA [*page#*] [,[*X1*] [,[*Y1*] [,[*X2*] [,[*Y2*] [,*number of connections*]]]]]

Description GDILA performs a 3×3 gray expansion on the gray image in the rectangular region defined by points (*X1*, *Y1*) and (*X2*, *Y2*) in the specified page of image memory.

Specify the image memory page number (0 or 1) in the *page#* parameter. The default setting is 0.

Specify the upper-left and lower-right corners of the rectangular region in points *(X1*, *Y1)* and *(X2*, *Y2)*. The default settings are (1, 1) and (510, 510).

Specify the *number of connections* as follows. The default setting is 0.

0: 8
Other than 0: 4

GEROS Gray EROSion

(Command)

Action Compresses the gray image in the specified region.

Format GEROS [*page#*] [,[*X1*] [,[*Y1*] [,[*X2*] [,[*Y2*] [,*number of connections*]]]]]

Description GEROS performs a 3×3 gray compression on the gray image in the rectangular region defined by points (*X1*, *Y1*) and (*X2*, *Y2*) in the specified page of image memory.

Specify the image memory page number (0 or 1) in the *page#* parameter. The default setting is 0.

Specify the upper-left and lower-right corners of the rectangular region in points *(X1*, *Y1)* and *(X2*, *Y2)*. The default settings are (1, 1) and (510, 510).

Specify the *number of connections* as follows. The default setting is 0.

0: 8
Other than 0: 4

GET# GET

(Command)

Action Reads data from a random file to the file buffer.

Format GET *file#* [, *record#*]

Description The GET# command reads data from the random access file specified with the file# to the file buffer.

Specify the file# as the number in which the random access file was opened with the OPEN statement. After using a random access file, it must be closed with the CLOSE statement.

Specify the number of the record to be read with the *record#.* If omitted, the record after the record# used with the previous GET# or PUT# command is read.

The data in the file buffer is assigned to the character variables allocated to variable areas with the FIELD# command. The character variables are then transferred to the program.

GET@ GET @

(Command)

Action Reads image data from a VRAM to an array variable.

Format GET@ *X1*, *Y1*, *X2*, *Y2*, *array*, *VRAM* [, [*page#*] [, *plane#*]]

Description The GET@ command reads image data from the rectangular region defined by the corner coordinates (X1, Y1), (X2, Y2) to the array variable with the name indicated by the *array* parameter.

Specify the *VRAM* where the data is stored with a number, as follows:

0: Character memory
1: Graphic memory
2: Window memory
3: Image memory
4: Shading memory (Cannot be used in the F350-C12E/C41E.)

Either omit the *page#* (default setting 0) or set it to 1.

A frame type memory is specified as the VRAM along with a plane# to store the data in the plane as binary data. The data is stored as raw image data if the *plane#* is omitted.

If a plane memory is specified, the number of binary image pixels equals the number of bits in a single data array and the number of bytes in the array to store the data is given by the following equation:

$$\left\lceil \frac{\text{No. of horizontal pixels} + 7}{8} \right\rceil \times (\text{No. of vertical pixels} + 4)$$

If the array is D%, for example, the subscript is calculated as follows:

(No. of bytes required ÷ 2) + 1

This applies when the subscript minimum value is set to 0 with the OPTION BASE command.

If a frame memory is specified, the number of raw image pixels equals the number of bytes in a single data array and the number of bytes in the array to store the data is given by the following equation:

No. of horizontal pixels × No. of vertical pixels + 4

If the array is D%, for example, the number of dimensions is calculated as follows:

(No. of bytes required ÷ 2) + 1

This applies when the subscript minimum value is set to 0 with the OPTION BASE command.

The final 4 bytes in the equation above represents the 4 bytes at the start of array where the array width and height are stored, as follows:

D% (0) = No. of horizontal pixels (2 bytes)
D% (1) = No. of vertical pixels (2 bytes)
D% (2) = 1st pixels
D% (3) = 2nd pixels

GETBLUT GET Binary LUT

(Command)

Action Reads the contents of the present binary-coded LUT value as an array variable.

Format GETBLUT *binary image plane#, array name* [*,* [*subscript*] [*, size*]]

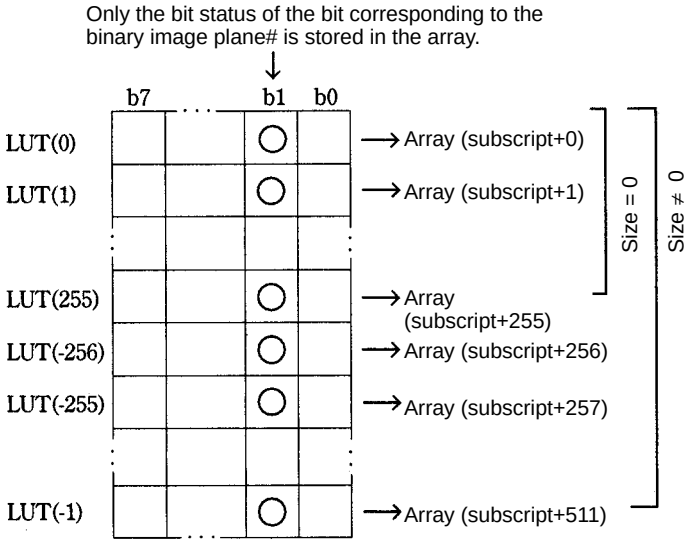
Description The GETBLUT command reads the contents of the binary LUT specified with the *binary image plane#* to the array specified with the *array name*.

The subscript specifies the first element in the data array to store the data. The default value is 0.

Specify the amount of data to be read in the *size* parameter. Set this parameter to 0 to specify 256 or 1 to specify 512. The default setting is 0.

Binary image plane# data that is 0 or a power of 2 (1, 2, 4, ... 128) can be stored in the array.

The number of bytes between the set subscript value and the maximum subscript value for the array must exceed the number of bytes set with the *size* parameter. When the binary LUT data is written to the array variable, either 256 or 512 elements are written from the start element specified by the subscript. Other parts of the array remain unchanged.



GETDLUT GET Display LUT

(Command)

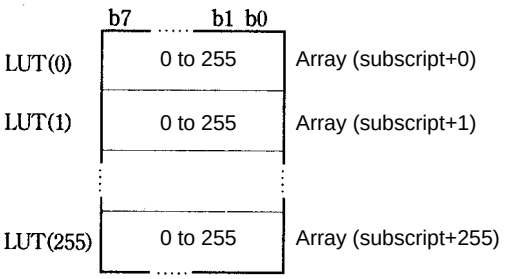
Action Reads the current display LUT data to an array variable.

Format GETDLUT *region, array name* [, [*subscript*]

Description The GETDLUT command reads the current display LUT data to the array specified with the *array name*.

Individual display LUTs are available for inside and outside the window. Specify which LUT is required with the *region* parameter. Set region to 0 to read the LUT data from outside the window or 1 to read the data from inside the window.

The *subscript* specifies the first element in the data array to store the LUT data. The default value is 0.



GETDLVL GET Display LeVeL

(Command)

Action Determines the display gradation (level) of the image.

Format GETDLVL (*image type*)

Description The GETDLVL command determines the display gradation of the image specified by the *image type* parameter, as follows:

0: Character memory
1: Graphic memory
2: Mask image
3: Binary image, white
4: Binary image, black
5: Window memory increments
6: Paint/pattern matching window memory increments

The display gradation is set with the SETDLVL command.

GETLUT GET LUT

(Command)

Action Reads the current filtering LUT data to an array variable.

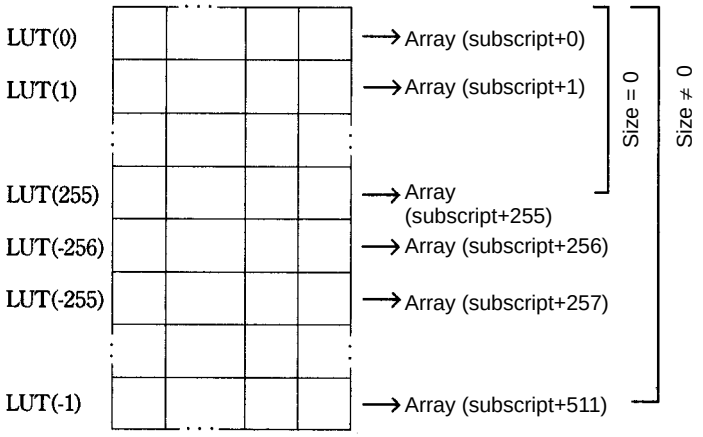
Format GETLUT *array name* [, [*subscript*] [, *size*]]

Description The GETLUT command reads the current filtering LUT data (binary LUT data) to the array specified with the *array name*.

The *subscript* specifies the first element in the data array to store the LUT data. The default value is 0.

The *size* parameter specifies the number of data bytes to be stored. Set the size to 0 to store 256 bytes or to any other value to store 512 bytes. The default value is 0.

The difference between the set *subscript* value and the maximum *subscript* value for the array must exceed the number of bytes set with the *size* parameter. When the filtering LUT data is written to the array variable, either 256 or 512 elements are written from the start element specified by the *subscript*. Other parts of the array remain unchanged.



(GETMDATA) (Not supported by the F350-C12E/C41E)
GET Measure DATA

(Command)

Action Reads the data acquired by the MDATA function all at once. This command cannot be used in the F350-C12E/C41E.

Format GETMDATA *data type, array* [*, subscript*]

Description Reads 8 windows of data acquired by the MDATA function.

Specify one of the following 12 *data types* with one of the values listed below.

0: Area (pixel units)
1: Area (after calibration)
2: Center of gravity X (pixel units)
3: Center of gravity X (after calibration)
4: Center of gravity Y (pixel units)
5: Center of gravity Y (after calibration)
6: Axis angle (pixel units)
7: Axis angle (after calibration)
8: Center of gravity X, center of gravity Y (pixel units)
9: Center of gravity X, center of gravity Y (after calibration)
10: All (pixel units)
11: All (after calibration)

Use the *array* parameter to specify the name of the array where the data will be stored. Be sure to declare the array name beforehand with the DIM command and be sure that the number of elements in the array is larger than the data being read. Neither a subscript nor parentheses are needed in the array name.

The *subscript* specifies the first element in the data array where data will be stored. The default value is 0, which indicates the beginning of the array.

Data is stored in the array as shown below when the data type is set to 0.

array(0) ← area of plane 0
array(1) ← area of plane 1
: :
: :
array(7) ← area of plane 7

Data is stored in the array as shown below when the data type is set to 8.

array(0) ← X center-of-gravity in plane 0
array(1) ← X center-of-gravity in plane 1
: :
: :
array(7) ← X center-of-gravity in plane 7
array(8) ← Y center-of-gravity in plane 0
: :
: :
array(15) ← Y center-of-gravity in plane 7

Data is stored in the array as shown below when the data type is set to 10.

array(0) ← area of plane 0
array(1) ← area of plane 1
: :
: :
array(7) ← area of plane 7
array(8) ← X center-of-gravity in plane 0
: :
: :
array(31) ← axis angle of plane 7

GETUINFO GET User INFOrmation

(Function)

Action Reads the information stored in the user information area.

Format GETUINFO ([*offset*], *type*)

Description Reads the user information that was set with the SETUINFO command.

Use the *offset* parameter to specify an offset from the beginning of the user information area where GETUINFO will start reading the data. The setting range is 0 to 1,022 bytes and the default value is 0.

Specify the type of data being read with the *type* parameter.

0: Integer (2 bytes)
1: Long integer (4 bytes)
2: Single-precision (4 bytes)
3: Double-precision (8 bytes)

In order to read data from the user information area properly, it is necessary to know the offset (in bytes) at which the desired type of data has been stored.

GETUINFO$ GET User INFOrmation $

(Function)

Action Reads character user information from the ROI model data area.

Format GETUINFO$ ([*offset*])

Description Reads the user information that was set with the SETUINFO command as character data.

Use the *offset* parameter to specify an offset from the beginning of the user information area where GETUINFO$ will start reading the data. The setting range is 0 to 1,022 bytes and the default value is 0.

GETVER GET VERsion information

(Function)

Action Reads version information or the unit code.

Format GETVER (*type,* [*, slot#*])

Description Reads information such as the system model, software version, or unit code.

Specify the desired type of information in the *type* parameter.

0: Model
1: Unit code
IMP Unit's code when the *slot#* is omitted.
Specified Unit's code when the *slot#* is specified.
2: OVL version
3: System version

Specify the slot number where the desired Unit is mounted with the *slot#* parameter. Slots are numbered 0, 1, 2, ... from the one nearest the IMP Unit. The F300-B32 has slots 0 to 3 and the F300-B52 has slots 0 to 5.

If the slot number is omitted, the IMP Unit's code will be returned.

F350-C12E: 10 is returned
F350-C41E: 40 is returned.

The following table shows the Unit codes (hexadecimal) and the corresponding model numbers. A value of 0 will be returned if the specified slot number doesn't exist or is empty.

Unit code		Model	Name
Hexadecimal	Decimal		
20	32	F300-A20	Normal Camera I/F Unit
21	33	F300-A20R	Shutter Camera I/F Unit
28	40	F300-A22S F300-A23RS	Normal Camera I/F Unit (Simultaneous) Frame Shutter Camera I/F Unit
29	41	F300-A22RS	Shutter Camera I/F Unit (Simultaneous)
30	48	F300-FS	Strobe I/F Unit
40	64	F300-D2	Terminal Block Unit
48	72	F300-DC2	Parallel I/O Unit
50	80	F300-E2	RS-232C I/F Unit
60	96	F300-L12E	F300 OVL Unit
61	97	F350-L12E	F350 OVL Unit

GMFUNC Gray Memory image FUNCtion

(Command)

Action Executes operations between gray images.

Format GMFUNC *operation*, *page# 1*, *page# 2* [, [*X1*] [, [*Y1*] [, [*X2*] [, *Y2*]]]]

Description GMFUNC performs an operation between the rectangular regions defined by points (*X1*, *Y1*) and (*X2*, *Y2*) in the two pages of image memory. The result of the operation is stored in page# 2 of image memory.

Specify one of the following 10 operations with the *operation* parameter.

0: AND
1: OR
2: XOR
3: Addition 1 (255 when 256 or greater.)
4: Addition 2 (remainder of division by 256 when 256 or greater)

5: Subtraction 1 (0 when less than zero.)
6: Subtraction 2 (remainder of division by 256 when less than zero)
7: Maximum value (greater of the two densities)
8: Maximum value (lesser of the two densities)
9: Absolute value (absolute value of the difference between the two)

Specify the image memory page numbers (0 or 1) in the *page# 1* and *page# 2* parameters. The default value for the page numbers is 0.

The same page number can be specified for both *page# 1* and *page# 2*.

Specify the top-left and bottom-right coordinates of the rectangular region with the *(X1*, *Y1)* and *(X2*, *Y2)* parameters. The default value of *(X1, Y1)* is (0, 0) and the default value of *(X2, Y2)* is (511, 511).

GOSUB GOSUB

(Command)

Action Branches to a specified subroutine.

Format GOSUB *line# **or** label*

Description The GOSUB command branches control to the subroutine starting with the specified *line number* or *label*. Control returns to the GOSUB statement position when the RETURN statement at the end of the subroutine is executed.

Specify the first line of the subroutine with the *line#* or *label name* as the GOSUB command parameter.

When subroutines are nested, GOSUB and RETURN statements must always be used in pairs.

GOTO GO TO

(Command)

Action Unconditionally jumps to a specified line.

Format FORMAT1: GO TO *line# **or** label*
FORMAT2: GOTO *line# **or** label*

Description The GOTO command unconditionally jumps control to the line with the specified line number or label.

The action of the GOTO and GO TO statements is identical.

HELP HELP

(Command)

Action Displays help messages.

Format HELP [*"command **or** function name"*]

Description The HELP command displays help messages for the specified OVL command or function.

The wildcard characters (?, ⁎) can be used in the *command name*. The ? wild-card represents a single character and the asterisk wildcard represents any number of characters. For example, the following command would display help messages for all commands beginning with the character W.

HELP "W⁎"

A list of command and function names is displayed if the *command name* parameter is omitted. No message is displayed if an unregistered command or function name is specified with the *command name*.

HELP ON/OFF/STOP HELP key ON/OFF/STOP

(Command)

Action Disables, enables, or stops interrupts from the HELP Key.

Format HELP ON *or* OFF *or* STOP

Description The HELP command controls branching to an interrupt processing routine when the HELP Key is pressed.

HELP ON: The HELP ON statement enables the interrupt processing routine when the HELP Key is pressed. When the HELP Key is pressed, operation branches to the interrupt processing routine at the line# or label defined with the ON HELP GOSUB statement.

HELP OFF: The HELP OFF statement disables the interrupt processing routine when the HELP Key is pressed. When the HELP Key is pressed, operation does not branch to an interrupt processing routine.

HELP STOP: The HELP STOP statement stops interrupt processing when the HELP Key is pressed. When the HELP Key is pressed, operation does not immediately branch to the interrupt processing routine. However, immediately branching is enabled by the HELP ON statement, operation branches to the interrupt processing routine at the line# or label defined with the ON HELP GOSUB statement.

HEX$ HEX $

(Function)

Action Converts a numeric expression to a hexadecimal character string.

Format HEX$ (*numeric expression*)

Description The HEX$ function converts a decimal value to a hexadecimal character string. It does not add the " &H" prefix to indicate a hexadecimal character string.

Specify the *numeric expression* as a decimal numeric constant or numeric variable between –2147483648 (-2^{31}) and 2147483647 ($2^{31}-1$). Any decimal places in the specified *numeric expression* are rounded off to create an integer which is converted to the hexadecimal character string.

To convert a hexadecimal character string back to numeric data, append the " &H" prefix to indicate a hexadecimal character string, then convert the character string with the VAL function.

HFILL Hole FILL

(Command)

Action Fills holes in a binary image.

Format HFILL [*page#*], *binary image plane#* [,[*X1*] [,[*Y1*] [,[*X2*] [,[*Y2*] [,*number of connections*]]]]]

Description HFILL fills holes in the binary image in the rectangular region defined by points (*X1*, *Y1*) and (*X2*, *Y2*) in image memory.

Specify the image memory page number (0 or 1) in the *page#* parameter. The default setting is 0.

The *binary image plane#* is used to specify the binary image plane number for the image memory that is to be processed.

Specify the upper-left and lower-right corners of the rectangular region in points *(X1*, *Y1)* and *(X2*, *Y2)*. The default settings are (1, 1) and (510, 510).

Specify the *number of connections* as follows. The default setting is 0.

0: 8
Other than 0: 4

A “Vision error” will occur if the number of elements in the graphic’s outline exceeds 255. A “Vision error” will also occur if the length of the graphic’s outline exceeds 4,096.

HISTGRAM HISToGRAM

(Command)

Action Reads the density histogram from the image memory.

Format HISTGRAM [*page#*], *X1, Y1, X2, Y2, array name* [, *subscript*]

Description The HISTGRAM command reads the density histogram from the image in the image memory.

Either omit the *page#* (default setting 0) or set it to 1.

The density histogram is read from the rectangular region with the specified corner coordinates *X1*, *Y1* and *X2*, *Y2*.

Specify the name of the array to store the density histogram with the *array name* parameter.

The *subscript* specifies the first element in the data array to store the density histogram. The default value is 0.

The array must have at least 256 elements between the specified *subscript* value and the maximum subscript value.

Array (subscript+0) ← frequency of use of density 0
Array (subscript+1) ← frequency of use of density 1
: :
: :
Array (subscript+255) ← frequency of use of density 255

IF..GOTO–ELSE IF..GOTO–ELSE

(Command)

Action Controls the program flow with a specified condition.

Format IF *conditional expression* GOTO *line #* ***or*** *label* [ELSE *statement* ***or*** *line #* ***or*** *label*]

Description The IF ... GOTO–ELSE command control the program flow with the specified *conditional expression*.

If the conditional expression is true (not 0), program operation jumps to the line# or label specified after GOTO. The statement, line# or label specified after ELSE is ignored.

If the *conditional expression* is false (0), program operation jumps to the line# or label specified after ELSE or executes the statement specified after ELSE. The line# or label specified after GOTO is ignored.

The ELSE statement can be omitted.

IF..THEN–ELSE IF..THEN–ELSE

(Command)

Action Controls the program flow with a specified condition.

Format IF *conditional expression* THEN *statement* ***or*** *line #* ***or*** *label* [ELSE *statement* ***or*** *line #* ***or*** *label*]

Description The IF ... THEN–ELSE command control the program flow with the specified *conditional expression*.

If the *conditional expression* is true (not 0), program operation jumps to the *line#* or *label* specified after THEN or executes the statement specified after THEN. The *statement*, *line#* or *label* specified after ELSE is ignored.

If the *conditional expression* is false (0), program operation jumps to the line# or label specified after ELSE or executes the statement specified after ELSE. The *statement*, *line#* or *label* specified after THEN is ignored.

The ELSE statement can be omitted.

IF..THEN–ELSEIF–ELSE–END IF

(Command)

Action Evaluates specified conditions.

Format

IF *conditional expression* THEN

Statement in THEN block

[ELSEIF *conditional expression* THEN

Statement in ELSE IF block

⋮]

[ELSE

Statement in ELSE block]

END IF

Description The IF ... THEN–ELSEIF–ELSE–END IF commands evaluate specified *conditional expressions*.

Executes the subsequent Statement in THEN block if the specified *conditional expression* is true (not 0). Jumps to the next ELSE IF, ELSE, or END statement if the *conditional expression* is false (0).

Multiple ELSE IF statements may be used or they may be omitted.

The ELSE statement may be omitted.

The END IF statement is required. It cannot be omitted.

The GOTO command must not be used to jump into or out of a IF ... THEN–ELSEIF–ELSE–END IF statement block.

IMGLOAD IMaGe data LOAD

(Command)

Action Loads image data to VRAM.

Format IMGLOAD *file name, X, Y, VRAM,* [, [*page#*] [, *plane#*]]]

Description The IMGLOAD command loads image data saved with the IMGSAVE command to VRAM.

Specify the image data with the *file name* parameter.

The data is saved to rectangular region. Specify the top-left coordinates of this rectangular region with the X, Y parameters.

Specify the VRAM where the data is loaded with a number, as follows:

0: Character memory
1: Graphic memory
2: Window memory
3: Image memory
4: Shading memory (Cannot be used in the F350-C12E/C41E.)

Either omit the *page#* (default setting 0) or set it to 1.

Specify the *plane#* when loading binary image data to a frame VRAM. An error occurs if an attempt is made to write raw image data to a plane VRAM. If image data is written to a frame VRAM, it cannot be loaded to planes write protected with the MASKBIT command.

IMGSAVE IMaGe data SAVE

(Command)

Action Saves image data from VRAM.

Format IMGSAVE *file name X1, Y1, X2, Y2, VRAM,* [, [*page#*] [, *plane#*]] [, *compression function*]]]

Description The IMGSAVE command saves image data from VRAM.

The contents of the image memory are saved to a file in the memory card with the specified *file name*. Only the image inside the rectangular region with corner coordinates (*X1*, *Y1*), (*X2*, *Y2*) is saved.

Specify the *VRAM* from which the data is saved with a number, as follows:

0: Character memory
1: Graphic memory
2: Window memory
3: Image memory
4: Shading memory (Cannot be used in the F350-C12E/C41E.)

Either omit the *page#* (default setting 0) or set it to 1.

Specify the *plane#* when saving binary image data or window data from a single plane of a frame VRAM. The data is treated as raw image data if the *plane#* is omitted.

Specify the *compression function*, as follows. The default setting is 0.

0: No compression
1: Compression for binary images
2: Compression for gray images

When the value 2 (compression for gray images) is specified, the rectangular region must be set to (0,0), (511,511).

If the “Out of memory” message is displayed when IMGSAVE is executed, either reduce the size of the rectangular region or enable compression by setting the *compression function* parameter to 1 or 2.

INKEY$ INput KEY $

(Function)

Action Determines the character input from the keyboard.

Format INKEY$

Description The INKEY$ function returns the key character of the key pressed when the function is executed. The function returns a null string (" ") if no key is pressed (that is, the keyboard buffer is empty).

Data input with the INKEY$ function is not displayed on the screen. Unlike other input commands, operation does not wait for a key to be pressed after the function is executed.

INPUT INPUT

(Command)

Action Assigns data input from the keyboard to a variable.

Format INPUT [*prompt character string* , ***or*** ;] *variable* [*, variable...*]

Description Operation waits for data input from the keyboard when the INPUT command is executed. Enter data delimited by commas (,) for the specified number of variables and press the Return Key to assign the data to the variables.

Specify a character string for the *prompt character string*. This character string is displayed to prompt for input of the required data.

The *prompt character string* must be separated from the subsequent *variable* by a comma (,) or semicolon (;). The specified *prompt character string* only is displayed if a comma is used. The specified *prompt character string* followed by a question mark (?) and a single space is displayed if a semicolon is used. If no specified *prompt character string* is specified, a question mark (?) and a single space are displayed.

Multiple variables can be specified delimited by commas (,). The variables can be either numeric or character variables.

If the Return Key is pressed without entering values, zero (0) is assigned to numeric variables and a null string (" ") is assigned to character variables.

INPUT# INPUT

(Command)

Action Reads data from a sequential access file and assigns it to variables.

Format INPUT *file#*, *variable* [, *variable*...]

Description Specify the *file#* as the number in which the random access file was opened with the OPEN statement.

The *variables* can be specified either as numeric or character variables. The specified variable format must match the corresponding data format.

The data is read from the file as numeric or character variables, complying with the rules governing these data types. Unlike the INPUT command, the INPUT# command displays no prompt message or question mark (?).

INPUT$ INPUT $

(Function)

Action Reads a specified length of data from a sequential access file, the RS-232C port, or the keyboard.

Format INPUT$ (*character string length* [, *file#*])

Description Specify the *file#* to read the data from with the *file#* parameter.

Data is read from the keyboard if the *file#* parameter is omitted. Unlike data input with the INPUT or LINE INPUT command, data input from the keyboard using the INPUT$ function is not displayed on the screen.

Specify the *character string length* as a positive integer value. The program operation waits for input of the specified number of characters.

If the length of the data input from the RS-232C port exceeds the number of characters specified with the *character string length*, the remaining data can be read by the next INPUT$ function.

INPUT WAIT INPUT WAIT

(Command)

Action Inputs data from the keyboard with a time limitation.

Format INPUT WAIT *wait time,* [*prompt character string , **or** ;*] *variable* [*, variable..*]

Description Operation waits for the specified *wait time* for data input from the keyboard when the INPUT WAIT command is executed. Enter data delimited by commas (,) for the specified number of variables and press the Return Key within the specified *wait time* to assign the data to the *variables*.

The INPUT WAIT command is identical to the INPUT command, except for the time limit. If the Return Key is pressed without entering values for the *variables*, zero (0) is assigned to numeric variables and a null string (" ") is assigned to character variables. If the time specified by *wait time* passes, the variables will not change.

Specify the *wait time* in units of 0.1 second.

INSTR IN STRing

(Function)

Action Determines the position of the specified characters in the character string.

Format INSTR ([*start position*,] *character string 1*, *character string 2*)

Description The INSTR function searches for *character string 2* within *character string 1*. If *character string 2* is found, INSTR returns the position of *character string 2* in bytes from the start of *character string 1*.

The function returns the value 0 if the character string 2 does not exist in character string 1. The INSTR function still returns the number of 1-byte characters if it is used to find a character string in a string of 2-byte characters. In this case, use the KINSTR function instead.

Specify a value with the *start position* parameter to set the the position to start searching *character string 1*. Specify the value between 1 and the number of characters in *character string 1*. If omitted, searching starts from the beginning of the character string. The function returns the value 0 if the specified *start position* is larger than the number of characters in *character string 1*.

Specify a character constant or a character variable for *character string 1*, the character string to search.

INT INTeger

(Function)

Action Rounds the specified value down to the next integer value.

Format INT (*numeric expression*)

Description The INT function rounds off the specified *numeric expression* and returns an integer not exceeding the value specified with the *numeric expression*.

The FIX and CINT functions are similar to the INT function. However, the FIX function simply cuts off (rounds down) the decimal places from the specified value. The CINT function rounds off the decimal places of the specified *numeric expression* (i.e., up or down to the nearest integer) and returns an integer.

INTR ON/OFF/STOP INTeRrupt ON/OFF/STOP

(Command)

Action Disables, enables, or stops interrupts with the STEP signal.

Format INTR ON ***or*** OFF ***or*** STOP

Description The INTR command control execution of the interrupt subroutine defined with the ON INTR GOSUB command when a STEP input is generated from a Terminal Block Unit or Parallel I/O Unit. The rising edge of the STEP input signal is detected to generate the interrupt.

The INTR ON statement enables the STEP interrupt processing. When the STEP signal is input, the interrupt routine is executed immediately.

The INTR OFF statement disables the STEP interrupt processing. The STEP input is ignored.

The INTR STOP statement stops STEP interrupt processing. An input STEP signal is stored in memory, and when STEP interrupt processing is enabled again with the INTR ON command, operation immediately branches to the STEP interrupt processing routine. Operation jumps to the STEP interrupt processing routine only once, regardless of the number of times the STEP signal was input.

IPL Initial Program Loading

(Command)

Action Sets the OVL boot-up mode.

Format IPL [{*0* ***or*** *1* ***or*** *file name*}] [, [*numeric value 1*] [, [*numeric value 2*] [, [*numeric value 3*] [, [*numeric value 4*] [, [*numeric value 5*] [, [*standard input*] [, [*standard output*] [, [*numeric value 6*] [, [*numeric value 7*] [, *numeric value 8*]]]]]]]]]

Description The IPL command sets the OVL boot-up mode.

The first parameter determines whether or not the OVL program is executed automatically.

0: The program in the IMP Unit isn't executed automatically.
1: The program in the IMP Unit is executed automatically.
file name: The specified program is loaded and executed automatically.

The *numeric value 1* parameter specifies the number of files which can be open simultaneously. Set a value between 1 and 11.

The *numeric value 2* parameter specifies the size of the array variable or character variable region in Kbytes. Set a value between 1 and 62 Kbyte.

The *numeric value 3* parameter specifies the size of the user stack region in Kbytes. Set a value between 1 and 2 Kbyte.

The *numeric value 4* parameter specifies the size of the compile region in Kbytes. Set a value between 2 and 512 Kbyte.

The *numeric value 5* parameter specifies the number of lines displayed on the screen. Set the value to 20 or 25.

The *standard input* parameter specifies the input device used when OVL is booted up.

0: Keyboard
1: RS-232C (channel 0)

The *standard output* parameter specifies the output device used when OVL is booted up.

0: Video monitor
1: RS-232C (channel 0)

The *numeric value 6* parameter specifies the size of the program region in Kbytes.

The *numeric value 7* parameter specifies the size of the user's region in Kbytes. This parameter is invalid with the C12E.

When the C12E is being used, the size of the user's region is determined automatically based on the size of the program region set with *numeric value 6*. The total of the user's region and program region is always 256 Kbytes.

When the C41E is being used, a total of 2,560 Kbytes are available for the user's region, program region, and RAM disk. Any memory that isn't allocated to the user's region and program region becomes RAM disk memory.

The *numeric value 8* parameter specifies the size of the local variable region used by the structured subroutine in Kbytes. Set a value between 1 and 64 Kbyte.

If all parameters are omitted, the current OVL boot-up mode status is displayed.

OVL must be rebooted to enable new parameter settings. The following table shows the factory settings.

Parameter	Factory setting
Numeric value 1 (number of open files)	11
Numeric value 2 (array variable or character variable region)	32
Numeric value 3 (user stack region)	2
Numeric value 4 (compile region)	64
Numeric value 5 (number of lines displayed)	25
Standard input device	Keyboard
Standard output device	Video monitor
Numeric value 6 (program region)	96
Numeric value 7 (user's region)	160
Numeric value 8 (local variable region)	10

JIS$ JIS code $

(Function)

Action Determines Shift JIS code for a 2-byte character.

Format JIS$ (*character string*)

Description The JIS$ function returns the hexadecimal character for the first 2-byte character in the specified *character string*.

If the specified character string contains 1-byte characters, the code is returned for the first and second bytes of the string. An error occurs if a null string (" ") or a string containing one byte is specified.

The KNJ$ function is the opposite of the JIS$ function. The KNJ$ function returns the 2-byte character corresponding to the specified 4-digit hexadecimal Shift JIS code. The F350 uses the Shift JIS codes.

KACNV$ Kanji Ank CoNVert $

(Function)

Action Converts 2-byte characters in a character string to 1-byte characters.

Format KACNV$ (*character string*)

Description The KACNV$ function converts an alphabetic, numeric, or kana 2-byte character to a 1-byte character of identical meaning. The KACNV$ function returns a character string containing only 1-byte characters.

Specify the character string with a character constant or character variable as a character string containing alphabetic, numeric, and kana characters defined with both 1-byte and 2-byte character codes.

An error occurs if the character string contains a Kanji character or a 2-byte character for which no 1-byte character is defined.

The AKCNV$ function is the opposite of the KACNV$ function. The AKCNV$ function converts 1-byte alphabetic, numeric, and kana characters in a character string to equivalent 2-byte characters.

KEXT$ Kanji EXTract $

(Function)

Action Extracts either 1-byte or 2-byte characters from the character string.

Format KEXT$ (*character string, function*)

Description The KEXT$ function extracts either the 1-byte or 2-byte characters from a character string containing by character types.

Specify the *character string* from which the characters are selected as a character constant or a character variable.

Specify the *function* parameter as either 0 or 1. The meanings of these settings are described in the table.

Function	Description
0	Extract the 1-byte characters
1	Extract the 2-byte characters

The KEXT$ function returns a null string (" ") if the type of character specified with the function parameter does not exist in the character string.

KEY KEY

(Command)

Action Assigns any character string to the function keys.

Format KEY [*key#, character expression*]

Description The KEY command sets the display for the function key (F1 to F10) with the specified key# using a character string or control code.

Specify the key# parameter as a value between 1 and 10. This number corresponds to one of the functions keys (F1 to F10).

Specify the character expression with up to 15 1-byte characters and control codes (CHR$ (1) to CHR$(31) and CHR$(127)).

The control codes cannot be input from the keyboard. Specify the codes using the CHR$ function with character codes linked by "+" signs. Each control code occupies one byte. The input guide displays the first five standard-sized characters.

If the key# and character expression parameters are omitted, the settings revert to the settings when the OVL was booted up.

KEY LIST KEY LIST

(Command)

Action Displays the function key settings on the screen.

Format KEY LIST

Description The KEY LIST command displays the present settings of the ten function keys.

KEY ON/OFF/STOP KEY ON/OFF/STOP

(Command)

Action Disables, enables, or stops interrupts from the function keys.

Format KEY [(*key#*)] ON ***or*** OFF ***or*** STOP

Description The KEY command controls branching to an interrupt processing routine when a function key is pressed.

Specify the key# parameter as a value between 1 and 10 corresponding to a function key number. The KEY ON/OFF/STOP command applies to all keys if the key# is not specified.

KEY ON: The KEY ON statement enables the interrupt processing routine when the function key is pressed. When the function key is pressed, operation branches to the interrupt processing routine at the line# or label defined with the ON KEY GOSUB statement.

KEY OFF: The KEY OFF statement disables the interrupt processing routine when the function key is pressed. When the function key is pressed, operation does not branch to an interrupt processing routine.

KEY STOP: The KEY STOP statement stops interrupt processing when the function key is pressed. When the function key is pressed, operation does not immediately branch to the interrupt processing routine but the pressed status is stored in memory. Immediately branching is enabled by the KEY ON statement, operation branches to the interrupt processing routine at the line# or label defined with the ON KEY GOSUB statement.

KEYIN KEYIN

(Function)

Action Reads the input status of the console keys.

Format KEYIN (*input mode*)

Description Returns the input status of the console keys.

If the *input mode* parameter is set to 0, the function returns the current key status at the time the function is executed. If the *input mode* parameter is set to 1, the function waits until a key is pressed.

The function returns a one byte (8-bit) value The 8 bits correspond to the following 8 keys as shown below. These bits are turned ON when the corresponding key is pressed.

Bit 0: ↑	Bit 4: ENT
Bit 1: ←	Bit 5: ESC
Bit 2: →	Bit 6: HELP
Bit 3: ↓	Bit 7: SHIFT

The status of the SHIFT Key is returned only when it is pressed simultaneously with another key. The SHIFT Key isn't recognized if it is pressed independently.

KILL KILL

(Command)

Action Deletes the file with the specified filename from the memory card.

Format KILL *filename*

Description Specify the name of a file existing in the memory card as a character string with the filename parameter. An error occurs if the specified filename does not exist in the memory card.

The filename parameter can specify any program or data file in the memory card. However, an error occurs if the KILL command is executed on a write-protected file.

An error occurs if the KILL command is executed on a file opened with the OPEN statement. Close the file with the CLOSE statement before deleting it with the KILL command.

A file cannot be deleted with the KILL command if the file attribute is set to write-protected with the SET command. Use the SET command to remove the write protection before deleting the file with the KILL command.

KINPUT Kanji INPUT

(Command)

Action Automatically sets the Japanese input mode and reads data from the keyboard.

Format KINPUT *character variable*

Description The Japanese input mode is selected automatically when the KINPUT command is executed and the system waits for input of a Japanese (2-byte) character from the keyboard. After the data is input, it is assigned to the *character variable* when the Return Key is pressed.

KINSTR Kanji IN STRing

(Function)

Action Determines the position of the specified characters in the character string

Format KINSTR ([*start position*,] *character string 1*, *character string 2*)

Description The KINSTR function finds a specified *character string 2* in a *character string 1* and returns the position of *character string 2* from the start of *character string 1*. 1-byte and 2-byte characters are each counted as a single character.

Specify a value with the *start position* parameter to set the the position to start searching *character string 1*. Specify the value between 1 and the number of characters in *character string 1*. If omitted, searching starts from the beginning of the character string. The function returns the value 0 if the specified *start position* is larger than the number of characters in *character string 1*.

Specify a character constant or a character variable for *character string 1*, the character string to search. The function returns the value 0 if the specified *character string 1* is a null string (" ").

Specify a character constant or a character variable for *character string 2*, the character string to be searched for. The function returns the specified *start position* value if *character string 2* is specified as a null string (" ").

The function returns the value 0 if the specified *character string 2* does not exist in *character string 1*.

KLEN Kanji LENgth

(Function)

Action Determines the number of characters in a character string including 2-byte characters.

Format KLEN (*character string* [, *function*])

Description The KLEN function returns the length of a specified character string. 1-byte and 2-byte characters are each counted as a single character.

Specify a character constant or a character variable for character string 1, the character string to search. The character string can contain both 1-byte and 2-byte characters.

Specify the function parameter as an integer. The default value is 0. The meaning of the function parameter is shown below.

Function	Description
0	Determines the total number of 1-byte and 2-byte characters in the character string.
1	Determines the total number of 1-byte characters only.
2	Determines the total number of 2-byte characters only.
3	Determines the total number of wide characters only.

KMID$ Kanji MIDdle $

(Function)

Action Extracts part of a character string containing 2-byte Japanese characters.

Format KMID$ (*character variable*, *start position* [, *number of characters*])

Description Extracts a specified length of character string from a character string containing 2-byte Japanese characters.

Specify a character constant or a character variable for *character variable* to search. Do not specify a null string (" ").

Specify the number of characters to be extracted with the *number of characters* parameter. The function returns a null string (" ") if the specified number of characters is negative or exceeds the actual number of characters between the *start position* and the right end of the character variable. Set the *start position* to 1 to select a string from the start of the specified *character variable*.

Specify the number of characters to be extracted with the *number of characters* parameter. All characters to the right of the *start position* are replaced if the *number of characters* is omitted or if the *number of characters* exceeds the actual number of characters between the *start position* and the right end of the *character variable*.

KNJ$ KaNJi $

(Function)

Action Determines the character corresponding to a Shift JIS character code.

Format KNJ$ (*character string*)

Description The KNJ$ function returns the 2-byte character corresponding to the specified 4-digit hexadecimal Shift JIS code.

Specify a Shift JIS code with the character string.

The JIS$ function has the opposite function to the KNJ$ function. The JIS$ function returns a 4-digit hexadecimal Shift JIS code corresponding to the first 2-byte character in a character string.

KPLOAD Kanji Pattern LOAD

(Command)

Action Registers user-defined character patterns in the F350.

Format KPLOAD *character code*, *integer array name*

Description The KPLOAD command registers user-defined character patterns in the F350.

Specify the code to be registered with the character code parameter. Use the codes between &HF8 and &HFF for standard characters and the codes between &HEB9F and &HEBFC or between &HEC40 and &HEC5D for wide characters. An error occurs if the character code is specified outside these ranges, although external characters for general-purpose structural subroutines are registered in codes &HEC5E to &HEC61.

Specify the array variable containing the character pattern with the integer array name. The array variable must previously be declared with the DIM command as a one-dimensional array with 16 elements.

Store the dot image of the character pattern in elements 1 to 16 of the array variable in the format shown below:

Array element 1 = &H1281
Array element 2 = &H2242
: :
: :
: :
Array element 16 = &H0

KPOS Kanji POSition

(Function)

Action Determines the number of bytes to the specified character position in the character string which includes 2-byte Japanese characters.

Format KPOS (*character string*, *character position*)

Description The specified *character string* may contain a mixture of 1-byte and 2-byte characters. The KPOS function returns the number of bytes up to the specified *character position* in the *character string*.

Specify the position of a character in the character string with the *character position* parameter.

The function returns 0 if the number of characters in the *character string* is less than the *character position*.

KTYPE Kanji TYPE

(Function)

Action Determines the type of character at a specified position in the character string.

Format KTYPE (*character string*, *character position*)

Description The KTYPE function determines the type of character at a specified *character position* in the *character-string* as a value between 0 and 2. The meanings of the returned numbers is shown below.

Returned value	Description
0	1-byte alphanumeric standard-sized character
1	2-byte double-sized character
2	2-byte standard-sized character

Specify the character string as a character constant or a character variable containing a mixture of 1-byte and 2-byte characters.

Specify the position of the required character from the start of the *character string* as an integer with the *character position* parameter. Specify the *character position* between 0 and the length of the *character string*. Count both 1-byte and 2-byte characters as one character.

The KLEN function can be used to determine the number of characters in a *character string*.

(LABEL)(Not supported by the F350-C12E/C41E) LABELing

(Command)

Action Carries out labelling based on the detailed runlength data measured with the MEASURE command.

Format LABEL [*link evaluation constant*]

Description The LABEL command carries out labelling based on the detailed runlength data measured with the MEASURE command. Labelling data is read with the LDATA function or LPOINT function.

The RMODE command and MEASURE command must be executed each time before the LABEL command is executed.

The specified *link evaluation constant* specifies the link evaluation method, as follows. (The default setting is 0.)

0: eight-neighbor evaluation
1: four-neighbor evaluation

The following commands and functions are related to the labelling carried out with the LABEL command:

LDATA
LNUM
LPUTIMG
LPOINT
LSORT

The LABEL command can't be used in the F350-C12E/C41E. When performing labelling in the F350-C12E/C41E, use the SLABEL command; the SLABEL command is similar to the LABEL command.

LBOUND — Lower BOUNDary

(Function)

Action Determines the lower boundary of an array dimension subscript.

Format LBOUND (*array name* [, *number of dimensions*])

Description The LBOUND function returns the lower boundary of an array dimension subscript.

Specify the name of the array for which the subscript is to be determined with the *array name* parameter.

Specify the number of dimensions of the array with the *number of dimensions* parameter. The default value is 1.

The returned value is either 0 or 1, as set when the OPTION BASE command was executed.

LCASE$ — Lower CASE $

(Function)

Action Converts uppercase letters in the character string to-lower case letters.

Format LCASE$ (*character string*)

Description The LCASE$ function converts uppercase letters in the *character string* to-lower case letters. Existing lowercase letters remain unchanged.

LDATA — Label DATA

(Function)

Action Measures data for the labelled image obtained with the (LABEL) or SLABEL command.

Format LDATA (*label#*, *item*)

Description The LDATA function measures data from an image labelled with the (LABEL) or SLABEL command.

Specify the number of the labelled image (1 to 255) in *label#*.

Specify the *item* with one of the following numbers:

0: Area
1: Center of gravity X
2: Center of gravity Y
3: Main axis angle
4: Peripheral length
5: Area after filling
6: Number of holes

7: X coordinate of top-left corner of external box
8: Y coordinate of top-left corner of external box
9: X coordinate of bottom-right corner of external box
10: Y coordinate of bottom-right corner of external box

The (LABEL) or SLABEL command must be executed before the LDATA function is used.

LEFT$ LEFT $

(Function)

Action Extracts a character string with the specified length from the left end of the specified character string.

Format LEFT$ (*character string*, *character string length*)

Description Extracts a character string of any length from the start of the specified *character string*.

The *character string* can be specified as a character constant or character variable. A null string (" ") cannot be specified.

Specify the length of the extracted character string in bytes with the *character string length* parameter as a value between 1 and the length of the *character string*. A null string is returned if 0 is specified for the *character string length*. The entire specified character string is returned if the *character string length* is greater than the length of the specified character string.

LEN LENgth

(Function)

Action Determines the number of bytes in a character string.

Format LEN (*character string*)

Description The LEN function returns the length of a specified *character string* in bytes.

Use the KLEN function to return the length of a string containing 2-byte characters.

LET LET

(Command)

Action Assigns the expression at the right to the variable at the left.

Format LET *variable = expression*

Description Assigns the expression at the right to the variable at the left. It is not possible to specify a variable at the right or an expression at the left.

The function name LET can be omitted. LET is normally omitted in a program.

The *variable* can be a numeric variable or a character variable. Similarly, the *expression* can be a numeric expression or a character expression. However, the types must match on the left and right. If a numeric variable is specified at the left, a numeric expression must be specified to the right. Similarly, if a character variable is specified at the left, a character expression must be specified to the right.

LEVEL LEVEL

(Command)

Action Sets the binary level for each binary image plane.

Format LEVEL *binary image plane#*, *lower limit*, *upper limit* [, *mode*]

Description The LEVEL command sets the binary level for each binary image plane.

Specify the binary image plane (0 to 7) for which the level is set with the *binary image plane#* parameter. Set the *binary image plane#* to –1 to set all binary image planes to the same binary level.

The gradations of the raw input image are converted to a binary image with all values between the specified *lower limit* and *upper limit* represented as 1.

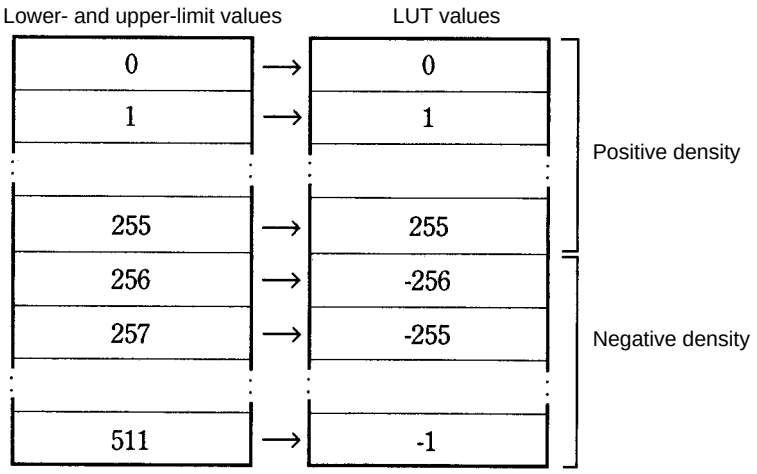
Specify the *lower limit* and *upper limit* between 0 and 511. The specified range between the lower limit and upper limit is set to 1 (normally white pixels).

Set the binary conversion mode with the mode parameter. The new binary level is calculated by a logical operation on the present binary level. The present setting is cancelled if the mode parameter is omitted. Before setting a value with the LEVEL command, cancel the setting that has been made.

OR: Set the specified range to 1.
NOT: Set the specified range to 0.
XOR: Reverse the values in the specified range.

The default setting for the *mode* parameter is OR.

The following diagram shows the relationship between the values set in the lower and upper limits and the LUT values.



LINE LINE

(Command)

Action Draws a straight line in VRAM.

Format LINE *X1*, *Y1*, *X2*, *Y2*, *VRAM* [, [*page#*] [,*drawing density* ***or*** *drawing mode*]]

Description The LINE command draws a straight line between the start and end points.

Specify the *VRAM* where the line is drawn with a number, as follows:

0: Character memory
1: Graphic memory
2: Window memory
3: Image memory
4: Shading memory (Cannot be used in the F350-C12E/C41E.)

Either omit the *page#* (default setting 0) or set it to 1.

The *drawing density* parameter specifies the density between 0 and 255 when drawing to the window, image, or shading memory. The default value is 255. The *drawing density* parameter has the following effect when set for the character or graphic memory:

0 : 0 written to memory
Other than 0: 1 written to memory

The *drawing mode* settings operate as follows. (The default setting is OR.)

OR: The current contents of VRAM ORed with 255 are written to memory.
NOT: 0 is written to memory
XOR: The current contents of VRAM are inverted.

When writing to a frame memory, the contents of planes write protected with the MASKBIT command remain unchanged.

LINE INPUT LINE INPUT

(Command)

Action Assigns a line of data input from the keyboard to a character variable.

Format LINE INPUT [*prompt character string;*] *character variable*

Description Operation waits for data input from the keyboard when the LINE INPUT command is executed. Enter the line of data and press the Return Key to assign the data to the variable.

Specify a character string for the *prompt character string*. This character string is displayed to prompt for input of the required data. The *prompt character string* must be separated from the subsequent character variable by a semicolon (;). The specified *prompt character string* only is displayed when the LINE INPUT command is executed.

Up to 255 1-byte characters, blanks, commas (,), double quotation marks (”), and numbers can be input into the *character variable*. All characters entered from the time the prompt is displayed until the Return Key is pressed are handled as a single character string.

If the Return Key is pressed without entering characters, a null string (”) is assigned to the *character variable*.

LINE INPUT WAIT LINE INPUT WAIT

(Command)

Action Inputs one line of data from the keyboard with a time limitation.

Format LINE INPUT WAIT *wait time*, [*prompt character string;*] *character variable*

Description Operation waits for the specified *wait time* for data input from the keyboard when the LINE INPUT WAIT command is executed. Enter the line of character data and press the Return Key within the specified *wait time* to assign the data to the *character variable*.

The LINE INPUT WAIT command is identical to the LINE INPUT command, except for the time limit. If the Return Key is pressed without entering characters, a null string (”) is assigned to *character variable*. The content of the variable will be uncertain if the specified wait time passes.

Specify the *wait time* in units of 0.1 second.

LINE INPUT# LINE INPUT

(Command)

Action Reads a line of data from a sequential access file and assigns it to a character variable.

Format LINE INPUT *file#*, *character variable*

Description Specify the *file#* as the number in which the random access file was opened with the OPEN statement.

All characters including blanks, commas (,), double quotation marks (”), and numbers between the first character read (including a blank space) and the carriage return (CHR$(13)) are read to the *character variable*.

The LINE INPUT# command is ideal for reading a line of data up to the carriage return (CHR$(13)) from a random access file containing both character and numeric variables delimited by blanks or commas (,).

Unlike the LINE INPUT command, the LINE INPUT# command can’t display a prompt message.

LIST LIST

(Command)

Action Displays all or part of the program contents.

Format LIST [*line# 1*] [–[*line# 2*]

Description Displays the program between *line# 1* and *line# 2* on the screen.

The entire program is displayed if the *line# 1* and *line# 2* parameters are omitted. Only *line# 1* is printed if *line# 2* is not specified. Lines from the start of the program to *line# 2* are printed if *line# 1* is not specified.

Use a period (.) instead of *line# 1* or *line# 2* to specify the current program line. The current line is indicated by the pointer. The current line is the last line input during program creation or the last line displayed with the LIST command.

LNUM Label NUMber

(Function)

Action Determines the number of labelled images.

Format LNUM

Description The LNUM function determines the number of images labelled with the (LABEL) or SLABEL command.

The (LABEL) or SLABEL command must be executed before the LNUM function is used.

LNUM will return –1 if there are over 255 labelled images.

LOAD LOAD

(Command)

Action Loads a program from disk to memory.

Format LOAD *file name* [, *R*]

Description The LOAD command loads the file specified with the *file name* parameter. The currently loaded program is deleted from memory.

If the *R* option is specified, the program is run with the currently open files immediately it is loaded.

Use the MERGE command to load a program without deleting the currently loaded program.

The backup variables are initialized when the LOAD command is executed.

LOC LOCation

(Function)

Action Determines the current I/O position within the specified file.

Format LOC (*file#*)

Description Specify the *file#* as the number in which the file was opened with the OPEN statement.

The position returned by the LOC function depends on the type of file, as defined in the table below.

File Type	Value returned by LOC
Random access file	The last record# read or written with the GET# or PUT# command.
Sequential access file	The number of records read or written since the file was opened.
RS-232C	The number of bytes remaining in the communication buffer.

LOCATE LOCATE

(Command)

Action Sets the cursor position on the text display and whether the cursor is displayed.

Format LOCATE [*column*] [, [*line*] [, *cursor display switch*]]

Description The LOCATE command moves the cursor on the text display and turns the cursor display on or off.

Specify the horizontal (X) coordinate with the column parameter between 0 and 63. The default value is 0.

Specify the vertical (Y) coordinate with the line parameter between 0 and 24. If this value is omitted, the cursor remains in the line position when the LOCATE command was executed.

Specify the cursor display switch parameter as 0 or 1. If this value is omitted, the cursor display status when the LOCATE command was executed is maintained.

0: The cursor isn't displayed on the screen.
1: The cursor is displayed on the screen.

LOF Length Of File

(Command)

Action Determines the size of a file.

Format LOF (*file#*)

Description Specify the *file#* as the number in which the file was opened with the OPEN statement.

The size returned by the LOF command depends on the type of file.

File Type	Value returned by LOF
Random access file	The file size as the maximum record number.
Sequential access file	The file size as a number of bytes.
RS-232C	The number of bytes available in the communication buffer.

LOG LOGarithm

(Function)

Action Determines the natural logarithm of a number.

Format LOG (*numeric expression*)

Description The LOG function returns the natural logarithm (to base e = 2.71828) of the *numeric expression* parameter. The *numeric expression* must be specified as a positive value.

The *numeric expression* may be in the form of an integer, long integer, or a single or double-precision real number.

LPOINT Label POINT

(Function)

Action Determines the label number of a labelled image at a specified position.

Format LPOINT (*X*, *Y*)

Description The LPOINT function finds the label number at a specified coordinate position from the label number data obtained with the (LABEL) or SLABEL command.

The function returns 0 if no labelled image or a hole in a labelled image exists at the specified coordinates.

The (LABEL) or SLABEL command must be executed before the LPOINT command is used.

LPUTIMG Label PUT IMaGe

(Command)

Action Draws a labelled image in VRAM.

Format LPUTIMG *label#*, *VRAM* [, [*page#*] , *plane#*]

Description The LPUTIMG command draws labeled image data obtained with the (LABEL) or SLABEL command to VRAM.

Specify the label number of the image to be drawn with the *label#* parameter.

Specify the *VRAM* where the image is drawn with a number, as follows:

0: Character memory
1: Graphic memory
2: Window memory
3: Image memory
4: Shading memory (Cannot be used in the F350-C12E/C41E.)

Either omit the *page#* (default setting 0) or set it to 1.

Specify the *plane#* when writing to a frame memory. Specify –1 to write the image to all planes. The image is not written to planes write protected with the MASKBIT command.

The (LABEL) or SLABEL command must be executed before the LPUTIMG command is used.

LSET Left SET

(Command)

Action Writes left-justified character data to a variable area defined with the FIELD command.

Format LSET *character variable = character string*

Description Specify a character variable name defined with the FIELD command with the *character variable* parameter.

Specify a character constant or character variable as the character string.

Excess characters are lost from the right of the character string if the length of the specified *character string* exceeds the length of the *character variable* defined with the FIELD command. Conversely, if the length of the specified *character string* is less than the length of the *character variable*, the remaining positions are filled with blanks.

LSORT Label SORT

(Command)

Action Reorders and renumbers labels.

Format LSORT *mode*

Description The LSORT command sorts the labeled image data obtained with the LABEL or SLABEL command according to the specified mode. Select one of the following *mode* parameter as the criteria for sorting the labelling results.

0: Area (descending order)
1: Area (ascending order)
2: Center of gravity X (descending order)
3: Center of gravity X (ascending order)
4: Center of gravity Y (descending order)
5: Center of gravity Y (ascending order)
6: Vertical direction from top-left
7: Vertical direction from bottom-right
8: Horizontal direction from top-left
9: Horizontal direction from bottom-right

The (LABEL) or SLABEL command must be executed before the LSORT command is used.

LSTYLE Line STYLE

(Command)

Action Sets the line style used in graphics functions.

Format LSTYLE *style*

Description The LSTYLE command sets the line style used in graphics functions. Select one of the following styles.

0: Solid line ■■■■■■■■■■■■■■■■■■■■ -----
1: Dashed line ■■□□■■□□■■□□■■□□■■□ -----
2: Broken line ■■■■■□□■■■■■□□■■■■■ -----

The line style selected with LSTYLE is effective for the following commands and remains effective until the line style is changed by executing LSTYLE again. The line style is set to solid line (style=0) when OVL is booted up.

ARC
BOX
CIRCLE
ELLIPSE
LINE
POLYGON
POLYLINE
SPCLOSE
SPLINE

LTRIM$ Left TRIM $

(Function)

Action Deletes spaces to the left of a character string.

Format LTRIM$ (*character string*)

Description Returns the *character string* (1- or 2-byte characters) with the spaces removed from the left.

MASK3 MASK 3×3

(Command)

Action Performs 3×3 filtering on gray-image data.

Format MASK3 [*page#*], *X1*, *Y1*, *X2*, *Y2*, *factor array* [, *global factor*]

Description Performs 3×3 filtering on the gray-image in the rectangular region defined by points (*X1*, *Y1*) and (*X2*, *Y2*) in image memory.

Specify the image memory page number (0 or 1) with the *page#* parameter. The default setting is 0.

Specify the upper-left and lower-right corners of the rectangular region with points *(X1*, *Y1)* and *(X2*, *Y2)*. The default settings are (1, 1) and (510, 510).

Specify the nine factors (values from –8 to +8) in the *factor array*. The following diagram shows the position of the pixels in this array.

Array(1)	Array(2)	Array(3)
Array(4)	Array(0)	Array(5)
Array(6)	Array(7)	Array(8)

The *global factor* can be set to 0, 1, 2, 4, 8, or 16. The default setting is 1.

MASKBIT MASKBIT

(Command)

Action Disables writing to specific planes in the frame memory.

Format MASKBIT *VRAM*, [*page#*], *bit data*

Description The MASKBIT command enables or disables writing to each plane of the frame memory.

Specify the *VRAM* with the *VRAM* parameter, as follows:

2: Window memory
3: Image memory
4: Shading memory (Cannot be used in the F350-C12E/C41E.)

Either omit the *page#* (default setting 0) or set it to 1.

Specify the plane to be write disabled with the *bit data* parameter. Set the bit corresponding to a plane to 1 to disable writing or to 0 to enable writing.

The image input to image memory will be input to all planes regardless of the setting made with the MASKBIT command.

(MDATA) (Not supported by the F350-C12E/C41E) Measure DATA

(Function)

The MDATA function can't be used with the F350-C12E/C41E.

Action Reads the results of binary raster measurements of area, center of gravity, and axis angle.

Format MDATA (*binary image plane#*, *measured data*)

Description The MDATA function reads the results of binary raster measurements of the area, center of gravity, and axis angle that were measured with the MEASURE command.

Specify the *binary image plane#* as the plane number (0 to 7) from which the data is read.

Specify the *measured data* with one of the values listed below to determine the the type of measured data to read.

0: Area (pixel units)
1: Area after calibration
2: Center of gravity X (pixel units)
3: Center of gravity X after calibration
4: Center of gravity Y (pixel units)
5: Center of gravity Y after calibration
6: Axis angle (pixel units)
7: Axis angle after calibration

The MMODE and MEASURE commands must be executed before the MDATA function is used.

(MEASURE) (Not supported by the F350-C12E/C41E)
MEASURE

(Command)

The MEASURE command can't be used in the F350-C12E/C41E.

Action Performs binary raster measurements with conditions set in advance.

Format MEASURE [*page#*] [, [*input path*] [, [*input mode*] [, *trigger mode*]]]

Description The MEASURE command performs binary raster measurements under the conditions set using MMODE and RMODE command.

It is possible to perform binary raster measurements and input the image to image memory at the same time.

The *page#* parameter specifies the page number of the image memory page where the image will be input. There is no image input when this parameter is omitted.

0: Input image memory 0 only.
1: Input image memory 1 only.
–1: Input to all of image memory.

The *input path* parameter specifies which bus will be used to input the image. The default setting is 0.

0: Image bus 1 (video bus, gray images)
1: Image bus 0 (pipeline bus, mainly filtered images)

The *input mode* parameter specifies whether the measurement is performed in frame mode or field mode. The mode specified with the FMODE command is used when this parameter is omitted. The mode is set to frame mode when OVL is first started.

The following table shows the differences between frame mode and field mode.

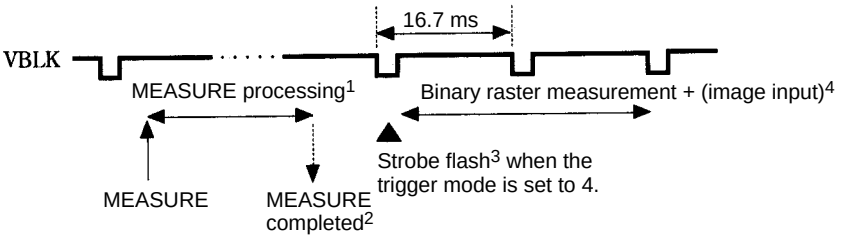
Mode		Resolution	Image input time (excluding wait time)
0	Frame mode	512×512 pixels	33.3 ms
1	Field mode	512×256 pixels	16.7 ms

The *trigger mode* parameter determines when the image is input, as shown in the following table.

Trigger mode		Image input timing
0	Stop	Cancels the STEP synchronization setting.
1	No strobe flash	A single image is input when MEASURE is executed.
2	STEP synchronization	A strobe flash is emitted in sync with the STEP signal and the image is input to image memory at that time.
3	Reserved	Not used.
4	Single strobe flash	A strobe flash is emitted and a single image is input when MEASURE is executed.

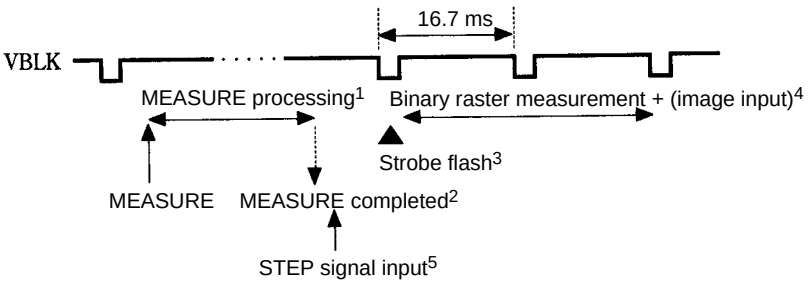
When the trigger mode is set to "2: STEP synchronization," a binary raster measurement + (image input) is performed when the STEP signal is input. Clear the STEP synchronization setting once the binary raster measurement has been performed with STEP synchronization.

The following diagram shows the timing of operation when the trigger mode is set to “1: No strobe flash” or “4: Single strobe flash.”



Note
1. Several ms are required for the MEASURE command’s processing.
2. The MEASURE command ends when it’s processing is completed.
3. When the MEASURE command is completed, a strobe flash is emitted in the next VBLK and then binary raster measurement + (image input) is started.
4. The time required for binary raster measurement + image input depends on the input mode.

The following diagram shows the timing of operation when the trigger mode is set to “2: STEP synchronization.”



Note
1. Several ms are required for the MEASURE command’s processing.
2. The MEASURE command ends when it’s processing is completed.
3. When the MEASURE command is completed, the binary raster measurement + (image input) is started in the next VBLK after the STEP signal is input.
4. The time required for binary raster measurement + image input depends on the input mode.
5. The STEP signal can be input after the MEASURE command has been completed.

MEDIAN MEDIAN

(Command)

Action Performs median filter processing on the gray image in the specified region.

Format MEDIAN [*page#*] [,[*X1*] [,[*Y1*] [,[*X2*] [,[*Y2*]]]]

Description MEDIAN performs a 3×3 median filter processing on the gray image in the rectangular region defined by points (*X1*, *Y1*) and (*X2*, *Y2*) in the specified page of image memory.

Specify the image memory page number (0 or 1) in the *page#* parameter. The default setting is 0.

Specify the upper-left and lower-right corners of the rectangular region in points *(X1*, *Y1)* and *(X2*, *Y2)*. The default settings are (1, 1) and (510, 510).

MERGE MERGE

(Command)

Action Merges a program from the memory card with the program in memory.

Format MERGE *filename* [*FOR APPEND*] [, *line# increment*]

Description Merges the program currently loaded in memory with the program specified by the filename parameter from the memory card. The program is reorganized into line number order after merging.

Existing lines in memory are deleted if line numbers are duplicated in the programs in memory and in the memory card.

The program cannot be executed automatically after merging. Use the CHAIN command to automatically execute programs.

If the *FOR APPEND* parameter is used, the program is merged from the end of the current program. The *line# increment* parameter is effective only when the *FOR APPEND* parameter is used.

If the *FOR APPEND* parameter is used, processing will stop if the text's line number exceeds 65,535.

If a program with a "GOTO line#" command is merged with the *FOR APPEND* specification, the line number in the GOTO command will not be updated.

MID$ MIDdle $

(Command, Function)

Action Extracts part of a character string. Changes part of a character string, if required.

Format Format 1: MID$(*character variable, start position* [*, number of characters*])

Format 2: MID$(*character variable, start position* [*, number of characters*]) = *character string*

Description Format 1

Extracts the specified *number of characters* from the *start position* of the *character variable*.

Specify the position of the first character in the *character variable* to be extracted as a value between 1 and the number of characters in the *character variable*.

Specify the number of characters to be extracted with the *number of characters* parameter. All characters to the right of the *start position* are extracted if the *number of characters* is omitted or if the *number of characters* exceeds the actual number of characters between the *start position* and the right end of the *character variable*.

Format 2

Replaces the specified *number of characters* from the start position of the *character variable* with the *character string*.

Specify the position of the first character in the *character variable* to be replaced as a value between 1 and the number of characters in the *character variable*.

Specify the number of characters to be replaced with the *number of characters* parameter. All characters to the right of the *start position* are replaced if the *number of characters* is omitted or if the number of characters exceeds the actual number of characters between the *start position* and the right end of the *character variable*.

Specify the *character string* as a character constant or a character variable.

MKD$ MaKe Double $

(Function)

Action Converts a double-precision value to internal character-type representation.

Format MKD$ (*double-precision value*)

Description The MKD$ function converts a *double-precision real number* to an 8-byte character string.

Because numeric data cannot be handled in a random access file, the double-precision numeric data must be converted to character-type numeric data with the MKD$ function before it can be written to a random access file. This character-type numeric data is then converted to a character string corresponding to the internal representation (binary representation) of the number type, which is used in the random access file.

The CVD function reverts a character-type *double-precision number* converted with the MKD$ function to a numeric value.

MKDIR MaKe DIRectory

(Command)

Action Creates a new directory for the memory card.

Format MKDIR *directory name*

Description Specify the name of the new directory name with the *directory name* parameter. An error occurs if a directory with the specified name already exists in the same path.

The new directory is created under the current directory if no new path is specified with the *directory name*.

If a new path is specified with the *directory name*, the new directory is created outside the current directory.

Use the ¥ symbol to delimit directories.

The path specifying a directory cannot exceed 63 characters (including the ¥ delimiters).

MKI$ MaKe Integer $

(Function)

Action Converts an integer value to internal character-type representation.

Format MKI$ (*integer*)

Description The MKI$ function converts an integer to a 2-byte character string.

Because numeric data cannot be handled in a random access file, the integer data must be converted to character-type numeric data with the MKI$ function before it can be written to a random access file. This character-type numeric data is then converted to a character string corresponding to the internal representation (binary representation) of the number type, which is used in the random access file.

The CVI function reverts a character-type integer number converted with the MKI$ function to a numeric value.

MKL$ MaKe Long $

(Function)

Action Converts a long integer value to internal character-type representation.

Format MKL$ (*long integer*)

Description The MKL$ function converts a *long integer* to a 4-byte character string.

Because numeric data cannot be handled in a random access file, the *long integer* data must be converted to character-type numeric data with the MKL$ function before it can be written to a random access file. This character-type numeric data is then converted to a character string corresponding to the internal representation (binary representation) of the number type, which is used in the random access file.

The CVL function reverts a character-type long integer number converted with the MKL$ function to a numeric value.

MKS$ MaKe Single $

(Function)

Action Converts a single-precision value to internal character-type representation.

Format MKS$ (*single-precision value*)

Description The MKS$ function converts a *single-precision real number* to a 4-byte character string.

Because numeric data cannot be handled in a random access file, the single-precision numeric data must be converted to character-type numeric data with the MKS$ function before it can be written to a random access file. This character-type numeric data is then converted to a character string corresponding to the internal representation (binary representation) of the number type, which is used in the random access file.

The CVS function reverts a character-type *single-precision number* converted with the MKS$ function to a numeric value.

(MMODE) (Not supported by the F350-C12E/C41E) Measure MODE

(Command)

Action Sets the binary raster measurement mode.

Format MMODE *binary image plane#, measured pixels* [, [*window function*] [, [*paint function*] [, *[fill function*] [, *run function*]]]]

Description The MMODE command sets the measurement conditions related to binary raster measurements. The MMODE command can't be used in the F350-C12E/C41E.

Binary raster measurement function takes real-time measurements of the area, center-of-gravity, and axis angle of the binary image in the specified region in window memory. The binary raster measurement function can be used only when an Expansion Board is installed.

The *binary image plane#* parameter specifies the binary image plane (0 to 7) for which the measurement conditions are set. Set the *binary image plane#* to –1 to set the conditions for all 8 binary image planes simultaneously.

Set the *measured pixels* parameter to set which pixels to measure. The default setting is "1: White pixels."

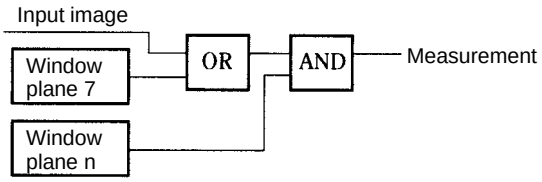
0: Black pixels
1: White pixels

The *window function* parameter specifies whether the region that has been drawn in window memory alone is to be measured. When the window function is OFF, that binary image plane's measurement results are set to 0.

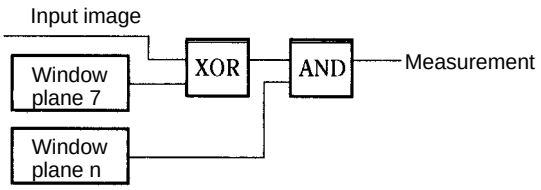
The *paint function* parameter specifies how to handle diagrams drawn in window plane 7. The default setting is 0.

0: Normal measurement (Treat it as a measurement region.)

1: Plane 7 is a paint window.
Diagrams drawn in window plane are measured as pixels.



2: Plane 7 is a patten matching window.
Diagrams drawn in window plane are XORed with measurement images and then measured.



Set the *fill function* parameter to set the contour measurement. The default setting is 0.

0: Contour measurement function OFF.

1: Contour measurement function ON.

The *run function* parameter specifies whether or not to measure the simple run length data. In the F350, the simple run length data measurement function is OFF regardless of the *run function* parameter setting.

NAME NAME

(Command)

Action Renames files stored in the memory card.

Format NAME *old filename AS new filename*

Description Specify the old filename as the name of an existing file in the memory card as a character string. An error occurs if the specified filename does not exist.

Specify the new name of the file with the new filename parameter. An error occurs if a file with the same name as the specified new filename already exists.

A filename cannot be renamed with the NAME command if the file attribute is set to write-protected with the SET command. Use the SET command to remove the write protection before renaming the file with the NAME command.

NEW NEW

(Command)

Action Deletes the program from memory.

Format NEW

Description The NEW command deletes the program from memory and resets numeric constants to 0 and character constants to a null string (" ").

Any open I/O files are closed.

The Direct mode is selected after the NEW command is executed.

Backup variables are initialized when the NEW command is executed.

NPIECE Number of PIECE

(Command)

Action Determines the number of elements in a character string divided by the specified delimiter.

Format NPIECE (*character string, delimiter*)

Description The NPIECE command determines the number of elements separated by the *delimiters* in the specified *character string*.

The function returns 1 if the *delimiter* character is not contained in the *character string*.

OCT$ OCTal $

(Function)

Action Converts a decimal number to an octal character string.

Format OCT$ (*numeric expression*)

Description The OCT$ function converts a decimal value to an octal character string. It does not add the " &O" prefix to indicate an octal character string.

Specify the *numeric expression* as a decimal numeric constant or numeric variable between –2147483648 (-2^{31}) and 2147483647 ($2^{31}-1$). Any decimal places in the specified *numeric expression* are rounded off to create an integer which is converted to the octal character string.

ON COM GOSUB ON COMmunication GO to SUBroutine

(Command)

Action Defines the jump destination of the interrupt subroutine when data is received at the RS-232C port.

Format ON COM (*RS-232C port#*) GOSUB *line#*

Description The ON COM GOSUB commands define the jump destination of the interrupt subroutine executed when data is received at the specified RS-232C port.

Specify the *RS-232C port#* as 1 or 2. The default value is 1. Interrupt operation starts when it is enabled by the COM ON command. Operation returns to the branch point when the RETURN command is executed.

ON CONKEY GOSUB — ON CONsole KEY GOSUB

(Command)

Action Defines the jump destination of the interrupt subroutine when a console key is pressed.

Format ON CONKEY GOSUB *line#* ***or*** *label*

Description The ON CONKEY GOSUB command defines the jump destination of the interrupt subroutine executed when a console key input is received. A label can be used instead of a line number.

ON ERROR GOTO — ON ERROR GOTO

(Command)

Action Defines the first line of the error processing routine executed when an error occurs.

Format ON ERROR GOTO *line#* ***or*** *label* ***or*** *0*

Description The ON ERROR ... GOTO command defines the jump destination of the error processing routine executed when an error occurs. When an error occurs, operation jumps to the specified jump destination and the error routine is executed to reset the error.

Specify the first line of the error processing routine with the *line#* or *label* parameter.

Specify *0* instead of the *line#* or *label* to cancel the error processing routine jump destination defined previously with the ON ERROR GOTO command. If an error subsequently occurs, an error message is displayed and program operation stops. When an error occurs, the error code and line number where the error occurred are assigned to the system variables ERR and ERL.

Operation returns to the original program when the RESUME command at the end of the error processing routine is executed.

ON GOSUB — ON GO to SUBroutine

(Command)

Action Defines the first line of the interrupt subroutine executed when the specified numeric expression is met.

Format ON *expression* GOSUB [*line#* ***or*** *label*] [, *line#* ***or*** *label*...]

Description The program branches to the line# or label indicated by the value of the expression. For example, when the expression=1, the program branches to the first *line number* or *label* listed. When the expression=n, the program branches to the n^{th} *line number* or *label* listed.

Program operation flows to the next line if the expression equals 0 or if the line# and label parameters are omitted.

ON GOTO — ON GO TO

(Command)

Action Branches program operation according to specified conditions.

Format ON *expression* GOTO [*line#* ***or*** *label*] [*, line#* ***or*** *label...*]

Description Program operation branches to the line specified by a *line#* or *label* according to the value of the expression. Operation jumps to the line defined with the first *line#* or *label* parameter when the expression equals 1, the second *line#* or *label* parameter when the expression equals 2, and so on.

Program operation flows to the next line if the expression equals 0 or if the line# and label parameters are omitted.

ON HELP GOSUB ON HELP key GO to SUBroutine

(Command)

Action Defines the first line of the interrupt subroutine executed when the HELP Key is pressed.

Format ON HELP GOSUB *line#* ***or*** *label*

Description The line# or label parameter defines the jump destination of the interrupt subroutine executed when the HELP Key is pressed.

The interrupt subroutine defined by the line# or label parameter is executed when the HELP Key is pressed during program execution if the interrupt operation is enabled with the HELP ON command.

The ON HELP GOSUB command is one type of key input interrupt routine. It is used as an aid to program operation by explaining how to use the program and input data.

Operation returns to the original program when the RETURN command at the end of the interrupt subroutine is executed. Operation can be transferred to a different position from the position where the interrupt occurred by specifying a line# or label parameter with the RETURN command.

ON INTR GOSUB ON INTeRrupt GO to SUBroutine

(Command)

Action Defines the first line of the interrupt subroutine executed when a STEP interrupt occurs.

Format ON INTR GOSUB *line#* ***or*** *label*

Description The ON INTR GOSUB command defines the line# or label of the first line of the interrupt subroutine executed when a STEP interrupt occurs. Any further STEP interrupt occurring during execution of the interrupt subroutine is ignored.

ON KEY GOSUB ON function KEY GO to SUBroutine

(Command)

Action Defines the first line of the interrupt subroutine executed when a function key is pressed.

Format ON KEY GOSUB *line#* ***or*** *label* [, *line#* ***or*** *label...*]

Description The *line#* or *label* parameter defines the jump destination of the interrupt subroutine executed when a function key is pressed.

Up to ten *line#* or *label* parameters can be specified delimited by commas (,). The order of the *line#* or *label* parameters corresponds to the function key numbers.

The interrupt subroutine defined by the *line#* or *label* parameter is executed only when the corresponding function key is pressed during program execution.

The ON KEY GOSUB command is one type of key input interrupt routine. Defining frequently executed routines or frequently input data in the interrupt subroutines makes the function keys a convenient aid to programming.

Operation returns to the original program when the RETURN command at the end of the interrupt subroutine is executed. Operation can be transferred to a different position from the position where the interrupt occurred by specifying a *line#* or *label* parameter with the RETURN command.

ON SQMEAS GOSUB ON SeQuential MEASure GOSUB

(Command)

Action Defines the first line of the sequential measure interrupt subroutine.

Format ON SQMEAS(*unit#*) GOSUB *line# **or** label*

Description The *line#* or *label* parameter defines the jump destination of the sequential measure interrupt subroutine for Unit specified with the *unit#* parameter.

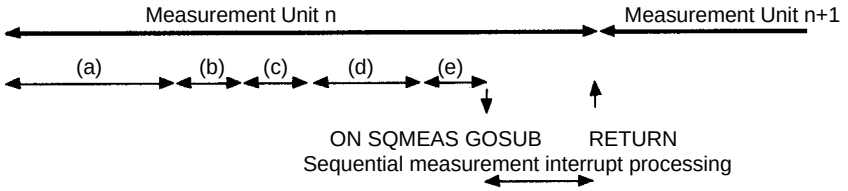
When a measurement Unit is completed, control branches to the interrupt subroutine if that Unit's sequential interrupt has been defined.

The *unit#* parameter specifies the unit number of the measurement Unit which will generate an interrupt when its raster processing is completed. The *line#* or *label* parameter defines the jump destination of the interrupt subroutine executed when the interrupt is received.

Operation returns to the original program when the RETURN command at the end of the interrupt subroutine is executed. The jump destination can be set separately for each measurement Unit.

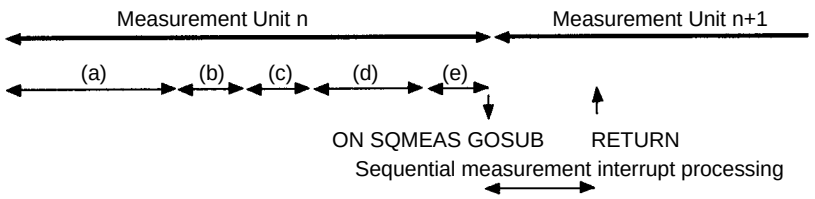
Commands related to sequential measurement settings can't be executed during the sequential measurement interrupt. The SQSET command's *@P* parameter determines when the measurement Unit's processing starts after the sequential measure interrupt, as shown in the following diagrams.

1, 2, 3... 1. @P0 specifies "non-pipeline process, brake enabled."



Note a) Search processing, binary raster processing
b) Position (displacement) compensation processing
c) Search processing completed machine language subroutine specified by SQSET.
d) ROI processing
e) ROI processing completed machine language subroutine specified by SQSET.

2. @P1 specifies "non-pipeline process, brake disabled."



Note a) Search processing, binary raster processing
b) Position (displacement) compensation processing
c) Search processing completed machine language subroutine specified by SQSET.
d) ROI processing
e) ROI processing completed machine language subroutine specified by SQSET.

ON STOP GOSUB ON STOP key GO to SUBroutine

(Command)

Action Defines the first line of the interrupt subroutine executed when the STOP Key is pressed.

Format ON STOP GOSUB *line#* ***or*** *label*

Description The *line#* or *label* parameter defines the jump destination of the interrupt subroutine executed when the STOP Key is pressed.

The interrupt subroutine defined by the *line#* or *label* parameter is executed when the STOP Key is pressed during program execution if the interrupt operation is enabled with the STOP ON command.

The ON STOP GOSUB command is one type of key input interrupt routine. Defining how to stop a specified operation in the interrupt subroutine at the specified jump destination allows the operation to be stopped by pressing the STEP Key.

Operation returns to the original program when the RETURN command at the end of the interrupt subroutine is executed. Operation can be transferred to a different position from the position where the interrupt occurred by specifying a *line#* or *label* parameter with the RETURN command.

ON TIME$ GOSUB ON TIME $ GO to SUBroutine

(Command)

Action Defines the time when the timer interrupt is generated and the first line of the interrupt subroutine.

Format ON TIME$ = *"HH:MM:SS"* GOSUB *line#* ***or*** *label*

Description The ON TIME$ GOSUB command sets the time when the timer interrupt is generated and defines the *line#* or *label* of the first line of the interrupt subroutine executed at the set time. If branching is enabled with the TIME$ ON command, the specified interrupt routine is executed when the set time is reached.

Set the time in the format: *HH:MM:SS*. Set the hour between 00 and 23 and set the minutes and seconds between 00 and 59.

The *line#* or *label* parameter defines the first line of the interrupt subroutine.

Operation returns to the original program when the RETURN command at the end of the interrupt subroutine is executed. Operation can be transferred to a different position from the position where the interrupt occurred by specifying a *line#* or *label* parameter with the RETURN command.

OPEN(1) OPEN

(Command)

Action Opens a file.

Format OPEN *filename* [*FOR OUTPUT* ***or*** *INPUT* ***or*** *APPEND*] *AS file#*

Description A sequential access file or random access file in the memory must be opened using the OPEN command with a specified file# before file I/O operations are possible. Use the CLOSE command to close the file after I/O operations are complete.

When opening a sequential access file, the I/O mode must be specified, as one of the following:

INPUT ... read data from the file
OUTPUT ... write data to the file
APPEND ... append data to an existing file

Do not specify the I/O mode when opening a random access file. If the I/O mode is specified, the random access file is treated as a sequential access file with the same name, so that an error occurs when I/O operations are attempted with the GET# and PUT# commands.

Specify the *file#* as a positive integer between 1 and 11. The same *file#* cannot be applied to more than one open file simultaneously. The *file#* assigned to the file can be used instead of the *filename* for all I/O operations until the file is closed with the CLOSE command.

Use the INPUT# and LINE INPUT# commands and the INPUT$ function to read data from a sequential access file and the PRINT#, PRINT# USING, and WRITE# command to write data to the sequential access file.

Define the file buffer with the FIELD command before I/O operations with a random access file. Use the LSET and RSET commands to set data in the file buffer and the PUT# and GET# commands to read and write data to and from the file.

OPEN(2) OPEN

(Command)

Action Opens the RS-232C port.

Format OPEN *"COM* [*RS-232C port#*]*:* [*baud rate* [*parity* [*data length* [*stop bits* [*XON switch* [*S parameter*]]]]]]*"* [*FOR OUTPUT* ***or*** *INPUT*] *AS file#*

Description The OPEN command opens the RS-232C port for data I/O. When data I/O is complete, close the RS-232C port with the CLOSE command.

Specify the RS-232C port# as 1 or 2, as follows. The default setting is 1.

1: channel 0
2: channel 1

Set the *baud rate*, *parity*, *data length*, *stop bits*, *XON switch* parameters, and *S parameter* to match the specifications of the device communicated with. Refer to the following table for details of these settings. The communications specifications set in the Setup menu will be used if these parameters are omitted.

Parameter	Setting	Description	Setup menu factory settings
Baud rate	1200	1,200 bps	9600
	2400	2,400 bps	
	4800	4,800 bps	
	9600	9,600 bps	
	19200	19,200 bps	
Parity	E	Even parity check	N
	O	Odd parity check	
	N	No parity check	
Data length	7	Each character 7 bits	8
	8	Each character 8 bits	
Stop bits	1	One stop bit	1
	2	Two stop bits	
XON switch	X	XON/XOFF control enabled	---
	---	XON/XOFF control disabled	
S parameter	S	ON	N
	N	OFF	

OPTION BASE OPTION BASE

(Command)

Action Declares minimum value of the array subscript.

Format OPTION BASE *0* ***or*** *1*

Description The OPTION BASE command declares the minimum value of the array subscript as 0 or 1.

The subscript minimum value is automatically set to 0 if an array variable is declared with the DIM command in a program containing no OPTION BASE command. The OPTION BASE command is used to set the subscript minimum value to 1. After setting the subscript minimum value with the OPTION BASE command, do not change the minimum value again in the same program.

OROUT OR OUT

(Command)

Action Controls the OR terminal.

Format OROUT *switch*

Description OROUT turns the Terminal Block Unit's OR terminal ON and OFF.

Specify the *switch* parameter as follows.

0: Turn OFF the OR terminal.
Other than 0: Turn ON the OR terminal.

When two or more Terminal Block Units are connected, the OROUT command controls the OR terminals on all of these Units.

PASSWD PASS WorD

(Command)

Action Sets or clears the OVL system password that prevents a program from being edited.

Format PASSWD *0* ***or*** *1*

Description Setting the password with the PASSWD command prevents the contents of the OVL program from being displayed or edited. **The password can't be cleared if it is forgotten, so be sure to keep a record of the password.**

Set the argument to 0 (the default setting) to set or change the password. Set the argument to 1 to clear the password.

The following commands are ineffective when the password has been set:

AUTO, DELETE, EDIT, FIND, LIST, RENUM, REPLACE, and SAVE

1, 2, 3... 1. Setting the password.
The user must enter the password twice when setting the password for the first time. Enter "PASSWD" to set the password.

2. Changing the password.
The user must enter the old password and then enter the new password twice when changing the password. Enter "PASSWD" to change the password.

3. Clearing the password.
The user must enter the password once to clear it. Enter "PASSWD 1" to clear the password.

PEEK PEEK

(Function)

Action Reads the contents of memory.

Format PEEK (*address* [*, data size*])

Description The PEEK function reads the contents of system memory.

The *address* parameter specifies the absolute address containing the data to be read. Be sure to use 32-bit data (long integer) to specify the address.

The *data size* parameter specifies the size of the data being read. The default setting is 0 (bytes).

0: Bytes (8 bits)
1: Words (16 bits)
2: Long words (32 bits)

PEEKDATA PEEK DATA

(Command)

Action Reads data directly from the specified address.

Format PEEKDATA *address, data elements, array*

Description The PEEKDATA command reads the specified number of data elements beginning at the specified address and stores this data in the specified array.

The *address* parameter specifies the beginning address in the memory region containing the data to be read.

The *data elements* parameter specifies the number of elements of array data.

The *array* parameter specifies the name of the one-dimensional array where the data is to be stored. Do not specify a character array variable for *array*.

The following table shows the number of bytes required for each type of array variable that can be specified.

Array variable type	Number of bytes/array element
Integer	2
Long integer	4
Single-precision	4
Double-precision	8

PIECE$ PIECE $

(Function)

Action Extracts partial character strings divided by the delimiter characters from the specified character string.

Format PIECE$ (*character string, delimiter character string* [*, start#* [*, end#*]])

Description The PIECE$ function returns the partial character strings between the specified *start#* and *end #* divided by the *delimiter characters* from the specified character string.

If the *end#* is omitted, the single partial string corresponding to the *start#* is returned. The first partial character string is returned if both the *start#* and *end#* are omitted. The entire character string is returned if it is not divided by the specified delimiter character.

PIN — Port IN

(Function)

Action Reads the bit status of a specified bit of the Terminal Block Unit or Parallel I/O Unit input port.

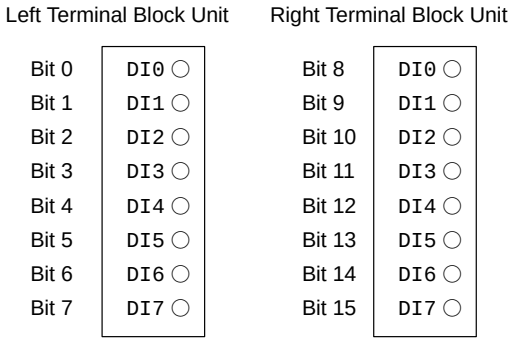
Format PIN (*bit address*)

Description The PIN function reads the bit status of the Terminal Block Unit or Parallel I/O Unit input port bit specified with the *bit address* parameter. The function returns a value, as follows:

0: bit status OFF
1: bit status ON

If both Terminal Block Units and Parallel I/O Units are connected, the PIN function does not differentiate between the two. The bit status can be specified up to the total number of input bits for all units.

Bit addresses are allocated in order from the left Unit, as shown below.



POINT — POINT

(Function)

Action Determines the density of the specified coordinates in VRAM.

Format POINT (*X, Y, VRAM* [, *page#*])

Description The POINT function reads the density at the specified coordinates *(X, Y)* in the specified VRAM.

Specify the *VRAM* with the *VRAM* parameter, as follows:

0: Character memory
1: Graphic memory
2: Window memory
3: Image memory
4: Shading memory (Cannot be used in the F350-C12E/C41E.)

The function returns either 0 or 1 if a plane VRAM is specified or a value between 0 and 255 if a frame VRAM is specified.

Either omit the *page#* (default setting 0) or set it to 1.

POKE — POKE

(Command)

Action Writes data to the specified address.

Format POKE (*address*, *expression* [, *data size*])

Description The POKE function writes data to the specified address. System operation may be jeopardized by writing data to memory areas used by the system.

The *address* parameter specifies the absolute address where the data is to be written. Be sure to use 32-bit data (long integer) to specify the address.

The *expression* parameter specifies the data to be written.

The *data size* parameter specifies the size of the data being written. The default setting is 0 (bytes).

0: Bytes (8 bits)
1: Words (16 bits)
2: Long words (32 bits)

POKEDATA — POKE DATA

(Command)

Action Writes data directly into the specified address.

Format POKEDATA *address, data elements, array*

Description The POKEDATA command writes the specified number of data elements from the array to memory beginning at the specified address.

The *address* parameter specifies the beginning address in the memory region where the data is to be written.

The *data elements* parameter specifies the number of elements of array data.

The *array* parameter specifies the name of the one-dimensional array where the desired data is stored. Do not specify a character array variable for *array*.

The following table shows the number of bytes required for each type of array variable that can be specified.

Array variable type	Number of bytes/array element
Integer	2
Long integer	4
Single-precision	4
Double-precision	8

POLYGON — POLYGON

(Command)

Action Draws a polygon in VRAM.

Format POLYGON *number of data, X array, Y array, VRAM* [*, [page#*] [*,* [*drawing density* ***or*** *drawing mode*] [*, lineart*]]]

Description The POLYGON command draws a polygon with points at the coordinates in the X array and Y array from the start of the array to the number of points specified with the number of data parameter.

Specify the number of data as a value up to 64.

Specify the *VRAM* where the polygon is drawn with a number, as follows:

0: Character memory
1: Graphic memory
2: Window memory
3: Image memory
4: Shading memory (Cannot be used in the F350-C12E/C41E.)

Either omit the *page#* (default setting 0) or set it to 1.

Specify the drawing method with the *drawing density* or the *drawing mode*. The default value is *drawing mode*, OR.

The *drawing density* parameter specifies the density between 0 and 255 when drawing to the window, image, or shading memory. The default value is 255. The

drawing density parameter has the following effect when set for the character or graphic memory:

0 : 0 written to memory
Other than 0: 1 written to memory

The *drawing mode* settings operate as follows. (OR is the default setting.)

OR: The current contents of VRAM ORed with 255 are written to memory.
NOT: 0 is written to memory
XOR: The current contents of VRAM are inverted.

Specify with the *lineart* parameter if the polygon is an outline only or filled. (The default setting is 0.)

0: Filled polygon
Other than 0: Polygon outline only

When writing to a frame memory, the contents of planes write protected with the MASKBIT command remain unchanged.

POLYLINE POLYLINE

(Command)

Action Draws a polyline in VRAM.

Format POLYLINE *number of data, X array, Y array, VRAM* [*, [page#*] [, [*drawing density* ***or*** *drawing mode*]]]

Description The POLYLINE command draws a polyline with points at the coordinates in the *X array* and *Y array* from the start of the array to the number of points specified with the *number of data* parameter.

Specify the *number of data* as a value up to 64.

Specify the *VRAM* where the polyline is drawn with a number, as follows:

0: Character memory
1: Graphic memory
2: Window memory
3: Image memory
4: Shading memory (Cannot be used in the F350-C12E/C41E.)

Either omit the *page#* (default setting 0) or set it to 1.

Specify the drawing method with the *drawing density* or the *drawing mode*. The default value is *drawing mode*, OR.

The *drawing density* parameter specifies the density between 0 and 255 when drawing to the window, image, or shading memory. The default value is 255. The *drawing density* parameter has the following effect when set for the character or graphic memory:

0 : 0 written to memory
Other than 0: 1 written to memory

The *drawing mode* settings operate as follows. (OR is the default setting.)

OR: The current contents of VRAM ORed with 255 are written to memory.
NOT: 0 is written to memory
XOR: The current contents of VRAM are inverted.

When writing to a frame memory, the contents of planes write protected with the MASKBIT command remain unchanged.

POS POSition

(Function)

Action Determines the current cursor column position.

Format POS (*numeric expression*)

Description The POS function determines the position of the character cursor in the X direction (along the line).

The parameter of the POS function is a dummy. It has no special significance, but cannot be omitted. Normally, set the dummy parameter to 0.

Use the CSRLIN function to determine the line where the cursor is positioned.

POUT Port OUT

(Command)

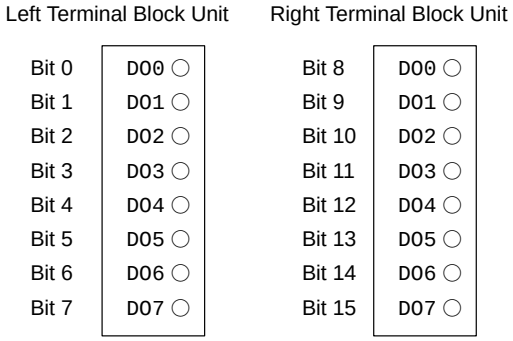
Action Controls the bit status of a specified bit of the Terminal Block Unit or Parallel I/O Unit output port.

Format POUT *bit address, 0* ***or*** *1*

Description The POUT function controls the bit status of the Terminal Block Unit or Parallel I/O Unit output port bit specified with the *bit address* parameter. Specify the bit address of the bit to control as a value between 0 and (maximum bit address – 1).

If both Terminal Block Units and Parallel I/O Units are connected, the POUT function does not differentiate between the two. The bit status can be specified up to the total number of input bits for all units.

Bit addresses are allocated in order from the left Unit, as shown below.



PRINT PRINT

(Command)

Action Displays data on the text display.

Format PRINT [*expression*] [*;* ***or*** *,* [*expression*]..] [*;* ***or*** *,*]

Description The *expression* parameter can be specified as a character expression or a numeric expression.

Delimit multiple *expressions* by commas (,) or semicolons (;). The display depends on the delimiter character used, as follows:

Comma (,)
14 characters are automatically allocated for each *expression*. An *expression* less than 14 characters in length is separated from the next expression by blanks.

Semicolon (;)
The *expressions* are displayed closed together. No fixed data length is allocated to the *expressions*. Numeric data is displayed preceded by a space for the sign and followed by a blank delimiter character.

Character returns are added automatically under the conditions described below.

Only a character return is displayed if all *expression* parameters are omitted. If no semicolon (;) is included after the final *expression* parameter, the character return is added after all the *expressions* are displayed. If a semicolon (;) is included after the final *expression* parameter, no character return is added after all the *expressions* are displayed and the first *expression* specified with a subsequent PRINT command is displayed consecutively.

A character return is added if the number of spaces remaining in the displayed line is less than the number of characters in the specified *expression*. Character data is displayed in the spaces remaining in the displayed line.

PRINT USING PRINT USING

(Command)

Action Displays formatted data on the text display.

Format PRINT USING *format control character string ; expression* [*;* ***or*** *, expression ...*]

Description Specify the *format control character string* in double quotations (”). The specified *format control character string* determines the number of characters and format in the display of the *expressions*.

Specify the character string defining character data or numeric string defining numeric data with the *expression* parameters.

Delimit multiple *expressions* by commas (,) or semicolons (;). The delimiter character used does not affect the display.

The format control character string may contain two types of characters: one type to control the display of character data and the other type to control the display of numeric data. The specified format control character string characters must match the type of data displayed. The control characters are described in the following tables.

Format Control Character Strings for Displaying Character Data

Character	Description
!	Display only the first character of the character string.
&(n blanks)&	The number of characters displayed is defined by the number of blanks (n) contained between the ampersands (&). A total of (n+2) characters is displayed from the start of the character string. The remaining characters are ignored if the character string contains more than (n+2) characters. The remaining character spaces are filled with blanks if the character string contains less than (n+2) characters.
@	Display the character string unchanged.
_	The character following the underscore character is not only a format control character and is printed.

Format Control Character Strings for Displaying Numeric Data

Character	Description
#	The number of # characters specifies the number of numeric data digits, including the +/– sign. If the specified number of characters is less than the number of numeric characters, the displayed data is right justified with blanks to the left.
.	Insert the decimal point (.) in combination with the # characters to specify the decimal position. Redundant decimal places are filled with zeros.
+	Specify the + sign as a prefix or suffix of the format control character string to print the + sign before or after the numeric data. If two or more + signs are specified consecutively, the second (and subsequent) + sign is considered to be outside the format control character string.
–	Specify the – sign as a suffix of the format control character string, the minus sign for negative numeric data is output to the right of the number (i.e., after the number) If two or more – signs are specified consecutively, the second (and subsequent) – sign is considered to be outside the format control character string.
**	Specify two or more asterisks (*) at the start of the format control character string to output asterisks instead of blanks to the left of the numeric data. The asterisks (*) fill all blanks in the specified number of characters.
¥¥	Specify two or more yen signs (¥) at the start of the format control character string to output a yen sign in the blank space to the left of the numeric data. The two yen signs (¥) occupy two character spaces and one of these is used to print the yen sign.
**¥	Specify two or more asterisks (*) plus a single yen sign (¥) at the start of the format control character string to output asterisks instead of blanks to the left of the numeric data and a yen sign in the blank space immediately to the left of the numeric data. The two asterisks (*) and the yen sign occupy three character spaces and one of these is used to print the yen sign.
,	Use a comma (,) in combination with the # number characters to delimit each 3 digits of the integer portion of the number. If the comma (,) is specified to the right of the decimal point (.), the comma (,) is output after the numeric data.
^ ^ ^ ^	Specify four carets (^) after the # number characters to output the number in exponential format.
_	The character following the underscore character is not only a format control character and is printed.

A non-control character in the format control character string is displayed before or after the character or numeric data.

A percent sign (%) is displayed in front of the numeric data if the digits of the numeric data format specified with the expression parameters exceed the size of the region specified by the format control characters.

PRINT# PRINT

(Command)

Action Writes data to a sequential access file.

Format PRINT *file#* [*, expression*[*; **or** , expression ...*]] [*; **or** ,*]

Description Specify the *file#* as the number in which the sequential access file was opened for output with the OPEN statement. After writing to the sequential access file is complete, the file must be closed with the CLOSE statement.

Except for the fact that the PRINT# command writes data to a file instead of displaying data on the screen, the command is identical to the PRINT command. Refer to the PRINT command for information on specifying the expressions, on the meaning of the delimiter characters, and on the rules covering the carriage return characters.

PRINT# USING PRINT # USING

(Command)

Action Writes formatted data to a sequential access file.

Format PRINT file#, USING *format control character string ; expression* [*; **or** , expression ...*]

Description Specify the *file#* as the number in which the sequential access file was opened for output with the OPEN statement. After writing to the sequential access file is complete, the file must be closed with the CLOSE statement.

Except for the fact that the PRINT# USING command writes data to a file instead of displaying data on the screen, the command is identical to the PRINT USING command. Refer to the PRINT USING command for information on specifying the *format control character string* and expressions, on the meaning of the delimiter characters, and on the rules covering the carriage return characters.

PSET Point SET

(Command)

Action Draws a point in VRAM.

Format PSET *X, Y, VRAM* [*, [page#*] [*, drawing density **or** drawing mode*]]

Description The PSET command draws a point at the specified coordinates.

Specify the *VRAM* where the point is drawn with a number, as follows:

0: Character memory
1: Graphic memory
2: Window memory
3: Image memory
4: Shading memory (Cannot be used in the F350-C12E/C41E.)

Either omit the *page#* (default setting 0) or set it to 1.

Specify the drawing method with the *drawing density* or the drawing mode. The default value is *drawing mode*, OR.

The *drawing density* parameter specifies the density between 0 and 255 when drawing to the window, image, or shading memory. The default value is 255. The *drawing density* parameter has the following effect when set for the character or graphic memory:

0 : 0 written to memory
Other than 0: 1 written to memory

The *drawing mode* settings operate as follows. (OR is the default setting.)

OR: The current contents of VRAM ORed with 255 are written to memory.
NOT: 0 is written to memory
XOR: The current contents of VRAM are inverted.

When writing to a frame memory, the contents of planes write protected with the MASKBIT command remain unchanged.

PUT# PUT

(Command)

Action Writes data from the file buffer to a random access file.

Format PUT *file#* [*, record#*]

Description The PUT# command writes the data stored in the file buffer to the random access file specified with the file#.

Specify the file# as the number in which the random access file was opened with the OPEN statement.

Specify the number of the record to be written to the file with the *record#*. If omitted, the record after the record# used with the previous GET# or PUT# command is read.

PUT@ PUT @

(Command)

Action Draws the array variable data stored in the image memory to VRAM.

Format PUT@ *X, Y, array name, VRAM* [, [*page#*] [, *plane#*]]

Description The PUT@ command draws (overwrites) data from the specified array to a rectangular area with the top-left coordinates *(X, Y)*.

Specify the *VRAM* where the data is drawn with a number, as follows:

0: Character memory
1: Graphic memory
2: Window memory
3: Image memory
4: Shading memory (Cannot be used in the F350-C12E/C41E.)

Either omit the *page#* (default setting 0) or set it to 1.

Specify the *plane number* (0 to 7) if a frame VRAM is specified with the VRAM parameter.

If a frame VRAM is specified, the contents of planes write protected with the MASKBIT command remain unchanged.

RANDOMIZE RANDOMIZE

(Command)

Action Initializes random number generation.

Format RANDOMIZE [*numeric expression*]

Description The RANDOMIZE command sets a new seed value to initialize random number generation. The RANDOMIZE command only initializes random number generation; use the RND function to generate random numbers.

Specify the seed value with the *numeric expression* between –32768 to 32767. If the *numeric expression* is omitted, a message is displayed and the system waits for a seed value between –32768 to 32767 to be input from the keyboard.

(RDATA) (Not supported by the F350–C12E/C41E) Runlength DATA

(Function)

The RDATA function can't be used in the F350-C12E/C41E.

Action Reads the detailed run length data.

Format RDATA (*XY coordinate, run#, measured data*)

Description The RDATA command reads measurement data based on the detailed run length data obtained with the MEASURE command.

Specify the coordinates where the run data (line length) is to be read. Set the Y coordinate if the RMODE measuring direction was set to the X direction, or set the X coordinate if the measuring direction was set to the Y direction.

Specify the number of the run data to be read at the specified XY coordinate position with the *run#* parameter. The function returns 0 if the specified run# exceeds the number of existing runs. The *run#* starts from 0.

Specify the type of data to be read from the specified run with the *measured data* parameter, as follows:

0: Number of runs 1: Run length
2: Edge coordinate (start point)
3: Edge coordinate (end point)

The *run#* is ignored if the *measure data* is set to 0.

READ READ

(Command)

Action Reads data defined with the DATA statement to a variable.

Format READ *variable* [*, variable...*]

Description The READ command reads the data defined with the DATA statement to sequentially specified variables.

Specify the variables separated by commas (,). The format of the variables must match the format specified with the DATA statement.

If the number of variables specified for the READ command is less than the number specified for the DATA statement, the remaining data is read by the next READ command. The remaining data is ignored if no further READ command is specified. The number of variables specified to be read must not exceed the number specified with the DATA statement. Take care when programming successive READ commands.

Execute the RESTORE command before the READ command to specify the line containing the DATA statement to read.

REM REMark

(Command)

Action Inserts remarks into the program.

Format REM ***or*** *'remarks*

Description The REM statement is used to insert remarks and explanations into the program text; it is non-executable and has no effect on program execution.

A single-quotation mark (') can be used instead of the REM statement.

All characters and symbols specified as the parameter are handled as a remark. Commands and functions included in the remark are not executed. A colon (:) is also considered as part of the remark and cannot be used to continue the line onto the next line.

RENUM RENUMber

(Command)

Action Renumbers the lines in all or part of the program.

Format RENUM [*new line#*] [, [*old line#*] , *increment*]

Description The RENUM command renumbers the *old line#* as the *new line#* and renumbers all subsequent lines to the end of the program by the specified increment. The default value for the *increment* is 10.

The program is renumbered from the first line if the *old line#* is omitted. The default value of the *new line#* is 10.

REPLACE REPLACE

(Command)

Action Searches for the specified old character string between line# 1 and line# 2 and replaces the it with the new character string.

Format REPLACE *old character string, new character string* [*,* [*line# 1* ***or*** *label 1*] [–[*line# 2* ***or*** *label 2*]]]

Description The REPLACE command replaces the *old character string* with the *new character string* between *line# 1* (*or label 1*) and *line# 2* (*or label 2*) and replaces the it with the new character string.

The search and replace operation starts from the first line of the program if the *line# 1* (or *label 1*) parameter is omitted. The search and replace operation continues to the last line of the program if the *line# 2* (or *label 2*) parameter is omitted.

RESTORE RESTORE

(Command)

Action Specifies the line with the DATA statement to be read by the READ command.

Format RESTORE [*line#* ***or*** *label*]

Description The RESTORE command specifies the DATA statement to be read by a subsequent READ command with a *line#* or *label*. If a *line#* is specified, the DATA statement in the specified line is read by the READ command. If a *label* is specified, the DATA statement declared after the label is read.

If the *line#* and *label* are omitted, subsequent READ commands read the DATA statement on the line with the smallest line number.

RESUME RESUME

(Command)

Action Ends an error processing routine and restarts the operation before the error occurred.

Format RESUME [*line#* ***or*** *label* ***or*** *0* ***or*** NEXT]

Description The RESUME command ends an error processing routine and returns operation to the position where the error occurred. The RESUME command must be executed within an error processing routine.

The *line#* or *label* specifies the jump destination after the error processing routine execution is complete. Normally, a *line#* or *label* would be specified to repeat the series of processes which led to the error. If the *line#* and *label* are omitted or if *0* is specified, operation returns to the statement where the error occurred.

If NEXT is specified, operation returns to the statement after the statement where the error occurred.

RETURN RETURN

(Command)

Action Ends a subroutine and returns operation to the position where the subroutine was called or to a specified line.

Format RETURN [*line#* ***or*** *label*]

Description The RETURN command ends a subroutine and returns operation to the position where the subroutine was called (the line after the GOSUB command) or to a specified line.

More than one RETURN command can be contained in a single subroutine.

The *line#* or *label* can be specified to force operation to jump to the specified position. The specified *line#* or *label* must be identical to the label called with the original GOSUB command. Correct operation cannot be guaranteed if the specified *line#* or *label* differs from the label specified with the GOSUB command, due to the loss of the correlation between the GOSUB and RETURN commands. Take care when executing a GOSUB command inside a FOR–NEXT loop if GOSUB commands are nested.

RIGHT$ RIGHT $

(Function)

Action Extracts a character string with the specified length from the right end of the specified character string.

Format RIGHT$ (*character string, character string length*)

Description Extracts a character string of any length from the end of the specified *character string*.

A null string (" ") cannot be specified in the *character string*.

Specify the length of the extracted character string in bytes with the *character string length* parameter. A null string is returned if 0 is specified for the *character string length*. The entire specified character string is returned if the *character string length* is greater than the length of the specified character string.

RMCLEAR ROI Model CLEAR

(Command)

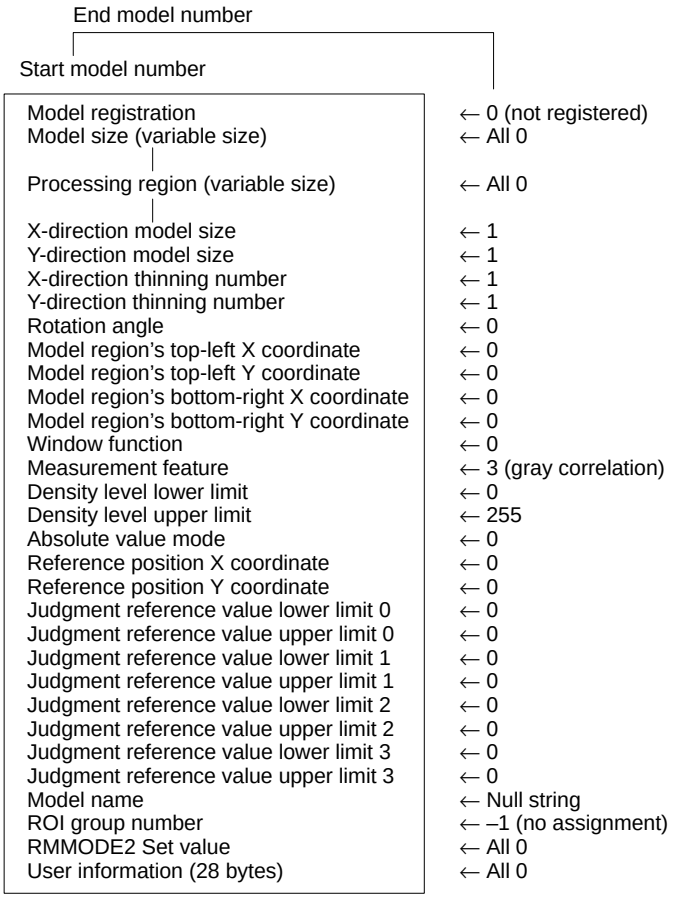
Action Clears and initializes the specified ROI model.

Format RMCLEAR *start model#* [, *end model#*]

Description The RMCLEAR command initializes the ROI model's information from the specified *start model#* to the specified *end model#*. The start and end model numbers can be set from 0 to 1023. If the end model number is omitted, just the start model number's ROI model will be initialized.

When the RMCLEAR command is executed, the ROI processing results (both single and sequential) for the specified ROI models are all cleared to 0. ROI models can be initialized even if they haven't been registered.

ROI model information is retained even when the power is interrupted or the OVL program is cleared. Use the RMCLEAR command to initialize the ROI models when an entirely new OVL program is loaded or the model information has been corrupted.



RMCOPY ROI Model COPY

(Command)

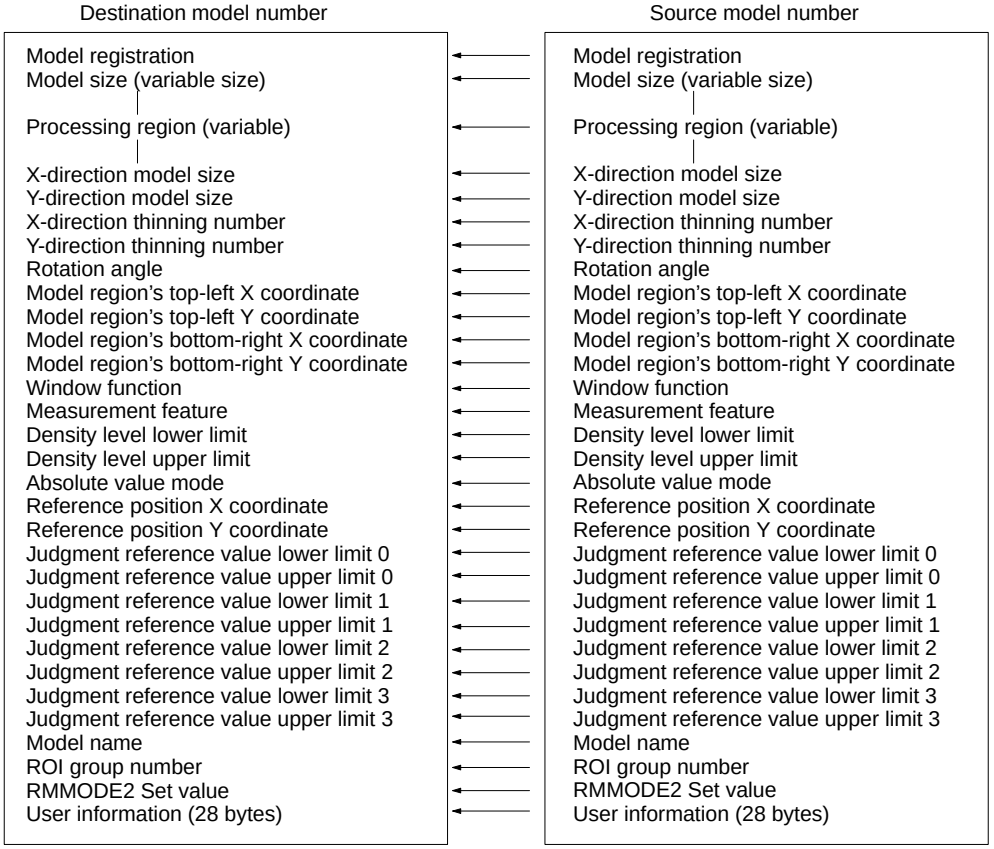
Action Copies all of the specified ROI model's information.

Format RMCOPY *source model#*, *destination model#*

Description The RMCOPY command copies all of the ROI model information from the specified *source model#* to the specified *destination model#*. The source and destination model numbers can be set from 0 to 1023.

This command is convenient when performing ROI processing on the same model image with different conditions.

An error will occur and copying won't be performed when there isn't free space for copying. This can occur when the model image of the source model is large.



RMDATA ROI Model DATA

(Command)

Action Reads ROI processing results at a different time from execution.

Format RMDATA *model#*, [*array 1*] [, [*array 2*] [, [*array 3*] [, [*array 4*] [, [*array 5*] [, [*array 6*] [, [*array 7*] [, [*array 8*] [, [*array 9*] [, [*array 10*] [, *array 11*]]]]]]]]]]

Description The RMDATA command reads the specified model's ROI processing results for single ROI processing (RMRUN or RMGRUN commands) or sequential ROI processing. Specify the model number of the desired model in *model#*. The model number can be set from 0 to 1023.

Specify the one-dimensional arrays that will contain the results in *array 1* through *array 4*. These arrays must be declared in advance with the DIM command.

The data stored in the arrays is determined by the measurement feature set with the RMMODE command and the processing mode set with the RMMODE2 command, as shown in the following tables.

Values specified by RMMODE's measurement features		Storage of measurement results			
		Array1	**Array2**	**Array3**	**Array4**
0	Binary features	Binary area	X center-of-gravity	Y center-of-gravity	---
1	Gray features	Average density	X center-of-gravity	Y center-of-gravity	---
2	Binary weighting correlation	Binary weighting correlation value	---	---	---
3	Gray correlation	Gray correlation value	Average density	Density deviation	---
4	Binary feature judgement	Binary area and binary area judgement	X center-of-gravity and X center-of-gravity judgement	Y center-of-gravity and Y center-of-gravity judgement	---
5	Gray feature judgement	Average density and average density judgement	X center-of-gravity and X center-of-gravity judgement	Y center-of-gravity and Y center-of-gravity judgement	---
6	Binary weighting correlation judgement	Binary weighting correlation value and Binary weighting correlation value judgement	---	---	---
7	Gray correlation judgement	Gray correlation value and gray correlation value judgement	Average density and average density judgement	Density deviation and density deviation judgement	---

Values specified by RMMODE's measurement features		Storage of measurement results			
		Array5	**Array6**	**Array7**	**Array8**
0	Binary features	X center-of-gravity before image scroll	Y center-of-gravity before image scroll	X execution position	Y execution position
1	Gray features	X center-of-gravity before image scroll	Y center-of-gravity before image scroll	X execution position	Y execution position
2	Binary weighting correlation	---	---	X execution position	Y execution position
3	Gray correlation	---	---	X execution position	Y execution position
4	Binary feature judgement	X center-of-gravity before image scroll	Y center-of-gravity before image scroll	X execution position	Y execution position
5	Gray feature judgement	X center-of-gravity before image scroll	Y center-of-gravity before image scroll	X execution position	Y execution position
6	Binary weighting correlation judgement	---	---	X execution position	Y execution position
7	Gray correlation judgement	---	---	---	---

Values specified by RMMODE2's processing mode		Storage of measurement results			
		Array1	**Array2**	**Array3**	**Array4**
0	Fixed position processing	Refer to the description of RMMODE.			
1	Precision search (no rotation)	Search position X coordinate and its judgement	Search position Y coordinate and its judgement	---	Search position correlation value and its judgement
2	Precision search (with rotation)	Search position X coordinate and its judgement	Search position Y coordinate and its judgement	Search angle and its judgement	Search position correlation value and its judgement
3	Processing in straight line region	X coordinate of defect position	Y coordinate of defect position	Estimated value of defect position	Judgement result
4	Processing in arc region	X coordinate of defect position	Y coordinate of defect position	Estimated value of defect position	Judgement result
5	Processing in circle region	X coordinate of defect position	Y coordinate of defect position	Estimated value of defect position	Judgement result
6	Processing in arbitrary region	X coordinate of defect position	Y coordinate of defect position	Estimated value of defect position	Judgement result
7	Number of straight line defects	X coordinate of defect position	Y coordinate of defect position	Estimated value of defect position	Judgement result
8	Number of arc defects	X coordinate of defect position	Y coordinate of defect position	Estimated value of defect position	Judgement result
9	Number of circle defects	X coordinate of defect position	Y coordinate of defect position	Estimated value of defect position	Judgement result
10	Number of arbitrary region defects	X coordinate of defect position	Y coordinate of defect position	Estimated value of defect position	Judgement result
11	Density level follow-up	(Same as the fixed position processing.)			

Values specified by RMMODE2's processing mode		Storage of measurement results			
		Array5	**Array6**	**Array7**	**Array8**
0	Fixed position processing	Refer to the description of RMMODE.			
1	Precision search (no rotation)	Search X before image scroll	Search Y before image scroll	---	---
2	Precision search (with rotation)	Search X before image scroll	Search Y before image scroll	Search angle before image scroll	---
3	Processing in straight line region	Defect X position before image scroll	Defect Y position before image scroll	---	---
4	Processing in arc region	Defect X position before image scroll	Defect Y position before image scroll	---	---
5	Processing in circle region	Defect X position before image scroll	Defect Y position before image scroll	---	---
6	Processing in arbitrary region	Defect X position before image scroll	Defect Y position before image scroll	---	---
7	Number of straight line defects	Defect X position before image scroll	Defect Y position before image scroll	Number of defects	Judgement result of the number of defects
8	Number of arc defects	Defect X position before image scroll	Defect Y position before image scroll	Number of defects	Judgement result of the number of defects
9	Number of circle defects	Defect X position before image scroll	Defect Y position before image scroll	Number of defects	Judgement result of the number of defects
10	Number of arbitrary region defects	Defect X position before image scroll	Defect Y position before image scroll	Number of defects	Judgement result of the number of defects
11	Density level follow-up	(Same as the fixed position processing.)			

Values specified by RMMODE2's processing mode		Storage of measurement results		
		Array9	**Array10**	**Array11**
0	Fixed position processing	Refer to the description of RMMODE.		
1	Precision search (no rotation)	---	---	---
2	Precision search (with rotation)	---	---	---
3	Processing in straight line region	---	---	---
4	Processing in arc region	---	---	---
5	Processing in circle region	---	---	---
6	Processing in arbitrary region	---	---	---
7	Number of straight line defects	---	---	---
8	Number of arc defects	---	---	---
9	Number of circle defects	---	---	---
10	Number of arbitrary region defects	---	---	---
11	Density level follow-up	Density lower limit after follow-up	Density upper limit after follow-up	Absolute value mode after follow-up

The following list provides details on the data stored in the arrays.

1, 2, 3... 1. Execution of fixed position processing (with no judgement)

RMMODE's measurement feature = 0 (binary features):
- Array1 (0) = Binary area (0 to 247808)
- Array2 (0) = Binary X center-of-gravity (0 to 511)
- Array3 (0) = Binary Y center-of-gravity (0 to 483)
- Array4 (0) = (Reserved, unpredictable value)
- Array5 (0) = Binary X center-of-gravity conversion coord. before image scroll
- Array6 (0) = Binary Y center-of-gravity conversion coord. before image scroll
- Array7 (0) = X execution position (0 to 511)
- Array8 (0) = Y execution position (0 to 483)

RMMODE's measurement feature = 1 (gray features):
- Array1 (0) = Average density (0 to 255)
- Array2 (0) = Gray-scale X center-of-gravity (0 to 511)
- Array3 (0) = Gray-scale Y center-of-gravity (0 to 483)
- Array4 (0) = (Reserved, unpredictable value)
- Array5 (0) = Gray-scale X center-of-gravity conversion coord. before image scroll
- Array6 (0) = Gray-scale Y center-of-gravity conversion coord. before image scroll
- Array7 (0) = X execution position (0 to 511)
- Array8 (0) = Y execution position (0 to 483)

RMMODE's measurement feature = 2 (binary weighting correlation):
- Array1 (0) = Binary weighting correlation value
- Array2 (0) = (Reserved, contents not updated)
- Array3 (0) = (Reserved, contents not updated)
- Array4 (0) = (Reserved, contents not updated)
- Array5 (0) = (Reserved, contents not updated)
- Array6 (0) = (Reserved, contents not updated)

Array7 (0) = X execution position (0 to 511)
Array8 (0) = Y execution position (0 to 483)

RMMODE's measurement feature = 3 (gray correlation):
Array1 (0) = Gray correlation value (0 to 100)
Array2 (0) = Average density (0 to 255)
Array3 (0) = Density deviation (0 to 127.5)
Array4 (0) = (Reserved, contents not updated)
Array5 (0) = (Reserved, contents not updated)
Array6 (0) = (Reserved, contents not updated)
Array7 (0) = X execution position (0 to 511)
Array8 (0) = Y execution position (0 to 483)

Unpredictable values will be stored for the coordinates before image scroll when the RMPOSCNV command is set for no coordinate conversion.

When ROI processing of all candidate points has been selected in the SQMODE command's processing mode, the elements in the array correspond to the candidate points. The type of data stored in each array is the same. For example, when the RMMODE's measurement feature = 0:
Array1 (0) = 1st candidate point's binary area
Array1 (1) = 2nd candidate point's binary area
Array1 (n–1) = nth candidate point's binary area

2. Execution of fixed position processing (with judgement)
A judgement result of 0 indicates "OK," –1 indicates "no good."

RMMODE's measurement feature = 4:
Binary features judgement
Array1 (0) = Binary area (0 to 247808)
Array1 (1) = Binary area judgement (0, –1, –2)
Array2 (0) = Binary X center-of-gravity (0 to 511)
Array2 (1) = Binary X center-of-gravity judgement (0, –1, –2)
Array3 (0) = Binary Y center-of-gravity (0 to 483)
Array3 (1) = Binary Y center-of-gravity judgement (0, –1, –2)
Array4 (0) = (Reserved, unpredictable value)
Array5 (0) = Binary X center-of-gravity conversion coord. before image scroll
Array6 (0) = Binary Y center-of-gravity conversion coord. before image scroll
Array7 (0) = X execution position (0 to 511)
Array8 (0) = Y execution position (0 to 483)

RMMODE's measurement feature = 5:
Gray features judgement
Array1 (0) = Average density (0 to 255)
Array1 (1) = Average density judgement (0, –1, –2)
Array2 (0) = Gray-scale X center-of-gravity (0 to 511)
Array2 (1) = Gray-scale X center-of-gravity judgement (0, –1, –2)
Array3 (0) = Gray-scale Y center-of-gravity (0 to 483)
Array3 (1) = Gray-scale Y center-of-gravity judgement (0, –1, –2)
Array4 (0) = (Reserved, unpredictable value)
Array5 (0) = Gray-scale X center-of-gravity conversion coord. before image scroll
Array6 (0) = Gray-scale Y center-of-gravity conversion coord. before image scroll
Array7 (0) = X execution position (0 to 511)
Array8 (0) = Y execution position (0 to 483)

RMMODE's measurement feature = 6:
Binary weighting correlations judgement
Array1 (0) = Binary weighting correlation value
Array1 (1) = Binary weighting corr. value judgement (0, –1, –2)

Array2 (0) = (Reserved, contents not updated)
Array3 (0) = (Reserved, contents not updated)
Array4 (0) = (Reserved, contents not updated)
Array5 (0) = (Reserved, contents not updated)
Array6 (0) = (Reserved, contents not updated)
Array7 (0) = X execution position (0 to 511)
Array8 (0) = Y execution position (0 to 483)

RMMODE's measurement feature = 7:
Gray correlation judgement
Array1 (0) = Gray correlation value (0 to 100)
Array1 (1) = Gray correlation value judgement (0, –1, –2)
Array2 (0) = Average density (0 to 255)
Array2 (1) = Average density judgement (0, –1, –2)
Array3 (0) = Density deviation (0 to 127.5)
Array3 (1) = Density deviation judgement (0, –1, –2)
Array4 (0) = (Reserved, contents not updated)
Array5 (0) = (Reserved, contents not updated)
Array6 (0) = (Reserved, contents not updated)
Array7 (0) = X execution position (0 to 511)
Array8 (0) = Y execution position (0 to 483)

Unpredictable values will be stored for the coordinates before image scroll when the RMPOSCNV command is set for no coordinate conversion.

3. Execution of precision search processing
A judgement result of 0 indicates "OK," –1 indicates "no good."

RMMODE2's processing mode = 1: Precision search (no rotation)
Array1 (0) = Search position X coord. (0.000 to 511.000)
Array1 (1) = Search position X coord. judgement (0, –1, –2)
Array2 (0) = Search position Y coord. (0.000 to 483.000)
Array2 (1) = Search position Y coord. judgement (0, –1, –2)
Array4 (0) = Search position correlation value (0 to 100)
Array4 (1) = Search position correlation value judgement (0, –1, –2)
Array5 (0) = Search X conversion coord. before image scroll
Array6 (0) = Search Y conversion coord. before image scroll

RMMODE2's processing mode = 2: Precision search (with rotation)
Array1 (0) = Search position X coord. (0.000 to 511.000)
Array1 (1) = Search position X coord. judgement (0, –1, –2)
Array2 (0) = Search position Y coord. (0.000 to 483.000)
Array2 (1) = Search position Y coord. judgement (0, –1, –2)
Array3 (0) = Search angle (–180° to 180°)
Array3 (1) = Search angle judgement (0, –1, –2)
Array4 (0) = Search position correlation value (0 to 100)
Array4 (1) = Search position correlation value judgement (0, –1, –2)
Array5 (0) = Search X conversion coord. before image scroll
Array6 (0) = Search Y conversion coord. before image scroll
Array7 (0) = Search angle conversion value before image scroll

Unpredictable values will be stored for the coordinates before image scroll when the RMPOSCNV command is set for no coordinate conversion.

4. Execution of defect inspection
A judgement result of 0 indicates "OK," –1 indicates "no good," and –2 indicates that that item wasn't inspected.

RMMODE2's processing mode = 3: Straight line region inspection
= 4: Arc region inspection
= 5: Circle region inspection
= 6: Arbitrary region inspection
Array1 (0) = Most probable large defect's X coord. (0 to 511)

Array1 (1) = Most probable small defect's X coord. (0 to 511)
Array1 (2) = Most probable min. density X coord. (0 to 511)
Array1 (3) = Most probable max. density X coord. (0 to 511)
Array2 (0) = Most probable large defect's Y coord. (0 to 483)
Array2 (1) = Most probable small defect's Y coord. (0 to 483)
Array2 (2) = Most probable min. density Y coord. (0 to 483)
Array2 (3) = Most probable max. density Y coord. (0 to 483)
Array3 (0) = Most probable large defect's estimate (0 to 255)
Array3 (1) = Most probable small defect's estimate (0 to 255)
Array3 (2) = Most probable min. density estimate (0 to 255)
Array3 (3) = Most probable max. density estimate (0 to 255)
Array4 (0) = Large defect judgement (0, –1, –2)
Array4 (1) = Small defect judgement (0, –1, –2)
Array4 (2) = Inspection min. density judgement (0, –1, –2)
Array4 (3) = Inspection max. density judgement (0, –1, –2)
Array5 (0) = Most probable large defect's X conversion coord. before image scroll
Array5 (1) = Most probable small defect's X conversion coord. before image scroll
Array5 (2) = Most probable min. density X conversion coord. before image scroll
Array5 (3) = Most probable max. density X conversion coord. before image scroll
Array6 (0) = Most probable large defect's Y conversion coord. before image scroll
Array6 (1) = Most probable small defect's Y conversion coord. before image scroll
Array6 (2) = Most probable min. density Y conversion coord. before image scroll
Array6 (3) = Most probable max. density Y conversion coord. before image scroll
Array7 (0) = Number of large defects (0 to 32,767)
Array7 (1) = Number of small defects (0 to 32,767)
Array7 (2) = Number of density defects (0 to 32,767)
Array7 (3) = (Reserved, unpredictable value)
Array8 (0) = Number of large defects judgement (0, –1, –2)
Array8 (1) = Number of small defects judgement (0, –1, –2)
Array8 (2) = Number of density defects judgement (0, –1, –2)
Array8 (3) = (Reserved, unpredictable value)

Unpredictable values will be stored for the coordinates before image scroll when the RMPOSCNV command is set for no coordinate conversion.

5. Execution of density follow-up measurement (with no judgement)

RMMODE's measurement feature = 0 (binary features):
Array1 (0) = Binary area (0 to 247808)
Array2 (0) = Binary X center-of-gravity (0 to 511)
Array3 (0) = Binary Y center-of-gravity (0 to 483)
Array4 (0) = (Reserved, unpredictable value)
Array5 (0) = Binary X center-of-gravity conversion coord. before image scroll
Array6 (0) = Binary Y center-of-gravity conversion coord. before image scroll
Array7 (0) = X execution position (0 to 511)
Array8 (0) = Y execution position (0 to 483)
Array9 (0) = Density lower limit after follow-up (0 to 255)
Array10 (0) = Density upper limit after follow-up (0 to 255)
Array11 (0) = Absolute value mode after follow-up (0, 1)

RMMODE's measurement feature = 1 (gray features):
- Array1 (0) = Average density (0 to 255)
- Array2 (0) = Gray-scale X center-of-gravity (0 to 511)
- Array3 (0) = Gray-scale Y center-of-gravity (0 to 483)
- Array4 (0) = (Reserved, unpredictable value)
- Array5 (0) = Gray-scale X center-of-gravity conversion coord. before image scroll
- Array6 (0) = Gray-scale Y center-of-gravity conversion coord. before image scroll
- Array7 (0) = X execution position (0 to 511)
- Array8 (0) = Y execution position (0 to 483)
- Array9 (0) = Density lower limit after follow-up (0 to 255)
- Array10 (0) = Density upper limit after follow-up (0 to 255)
- Array11 (0) = Absolute value mode after follow-up (0, 1)

RMMODE's measurement feature = 2 (binary weighting correlation):
- Array1 (0) = Binary weighting correlation value
- Array2 (0) = (Reserved, contents not updated)
- Array3 (0) = (Reserved, contents not updated)
- Array4 (0) = (Reserved, contents not updated)
- Array5 (0) = (Reserved, contents not updated)
- Array6 (0) = (Reserved, contents not updated)
- Array7 (0) = X execution position (0 to 511)
- Array8 (0) = Y execution position (0 to 483)
- Array9 (0) = Density lower limit after follow-up (0 to 255)
- Array10 (0) = Density upper limit after follow-up (0 to 255)
- Array11 (0) = Absolute value mode after follow-up (0, 1)

RMMODE's measurement feature = 3 (gray correlation):
- Array1 (0) = Gray correlation value (0 to 100)
- Array2 (0) = Average density (0 to 255)
- Array3 (0) = Density deviation (0 to 127.5)
- Array4 (0) = (Reserved, contents not updated)
- Array5 (0) = (Reserved, contents not updated)
- Array6 (0) = (Reserved, contents not updated)
- Array7 (0) = X execution position (0 to 511)
- Array8 (0) = Y execution position (0 to 483)
- Array9 (0) = Density lower limit after follow-up (0 to 255)
- Array10 (0) = Density upper limit after follow-up (0 to 255)
- Array11 (0) = Absolute value mode after follow-up (0, 1)

Unpredictable values will be stored for the coordinates before image scroll when the RMPOSCNV command is set for no coordinate conversion.

6. Execution of density follow-up measurement (with judgement)
A judgement result of 0 indicates "OK," –1 indicates "no good."

RMMODE's measurement feature = 4:
Binary features judgement
- Array1 (0) = Binary area (0 to 247808)
- Array1 (1) = Binary area judgement (0, –1, –2)
- Array2 (0) = Binary X center-of-gravity (0 to 511)
- Array2 (1) = Binary X center-of-gravity judgement (0, –1, –2)
- Array3 (0) = Binary Y center-of-gravity (0 to 483)
- Array3 (1) = Binary Y center-of-gravity judgement (0, –1, –2)
- Array4 (0) = (Reserved, unpredictable value)
- Array5 (0) = Binary X center-of-gravity conversion coord. before image scroll
- Array6 (0) = Binary Y center-of-gravity conversion coord. before image scroll
- Array7 (0) = X execution position (0 to 511)

Array8 (0) = Y execution position (0 to 483)
Array9 (0) = Density lower limit after follow-up (0 to 255)
Array10 (0) = Density upper limit after follow-up (0 to 255)
Array11 (0) = Absolute value mode after follow-up (0, 1)

RMMODE's measurement feature = 5:
Gray features judgement
Array1 (0) = Average density (0 to 255)
Array1 (1) = Average density judgement (0, –1, –2)
Array2 (0) = Gray-scale X center-of-gravity (0 to 511)
Array2 (1) = Gray-scale X center-of-gravity judgement (0, –1, –2)
Array3 (0) = Gray-scale Y center-of-gravity (0 to 483)
Array3 (1) = Gray-scale Y center-of-gravity judgement (0, –1, –2)
Array4 (0) = (Reserved, unpredictable value)
Array5 (0) = Gray-scale X center-of-gravity conversion coord. before image scroll
Array6 (0) = Gray-scale Y center-of-gravity conversion coord. before image scroll
Array7 (0) = X execution position (0 to 511)
Array8 (0) = Y execution position (0 to 483)
Array9 (0) = Density lower limit after follow-up (0 to 255)
Array10 (0) = Density upper limit after follow-up (0 to 255)
Array11 (0) = Absolute value mode after follow-up (0, 1)

RMMODE's measurement feature = 6:
Binary weighting correlations judgement
Array1 (0) = Binary weighting correlation value
Array1 (1) = Binary weighting corr. value judgement (0, –1, –2)
Array2 (0) = (Reserved, contents not updated)
Array3 (0) = (Reserved, contents not updated)
Array4 (0) = (Reserved, contents not updated)
Array5 (0) = (Reserved, contents not updated)
Array6 (0) = (Reserved, contents not updated)
Array7 (0) = X execution position (0 to 511)
Array8 (0) = Y execution position (0 to 483)
Array9 (0) = Density lower limit after follow-up (0 to 255)
Array10 (0) = Density upper limit after follow-up (0 to 255)
Array11 (0) = Absolute value mode after follow-up (0, 1)

RMMODE's measurement feature = 7:
Gray correlation judgement
Array1 (0) = Gray correlation value (0 to 100)
Array1 (1) = Gray correlation value judgement (0, –1, –2)
Array2 (0) = Average density (0 to 255)
Array2 (1) = Average density judgement (0, –1, –2)
Array3 (0) = Density deviation (0 to 127.5)
Array3 (1) = Density deviation judgement (0, –1, –2)
Array4 (0) = (Reserved, contents not updated)
Array5 (0) = (Reserved, contents not updated)
Array6 (0) = (Reserved, contents not updated)
Array7 (0) = X execution position (0 to 511)
Array8 (0) = Y execution position (0 to 483)
Array9 (0) = Density lower limit after follow-up (0 to 255)
Array10 (0) = Density upper limit after follow-up (0 to 255)
Array11 (0) = Absolute value mode after follow-up (0, 1)

Unpredictable values will be stored for the coordinates before image scroll when the RMPOSCNV command is set for no coordinate conversion.

Data won't be stored in arrays that weren't specified.
A value of 0 will be stored when the array has been specified but data doesn't exist for the item that should be stored in the array element.

A value of –2 will be stored when a measurement feature that doesn't perform judgement processing has been specified, but a judgement result has been read.

RMDIR ReMove DIRectory

(Command)

Action Deletes a directory from the memory card.

Format RMDIR *directory name*

Description Specify the *directory name* as the name of a directory in the memory card with a character string. An error occurs if the specified directory does not exist.

The directory specified with the *directory name* cannot be deleted if it contains files. The files in the directory must be deleted with the KILL command before using the RMDIR command.

Do not specify the current directory as the *directory name* parameter for the RMDIR command. To delete the current directory, first use the CHDIR command to change the current directory to another directory (normally one directory up the hierarchy) before using the RMDIR command.

RMDRAW ROI Model DRAW

(Command)

Action Draws an ROI model in image memory.

Format RMDRAW *model#*, [*page#*], *X1*, *Y1*, *X2*, *Y2* [, [*window function*] [, *window plane#*]]

Description Draws the ROI model specified by *model#* in image memory. This command is convenient for checking a registered model image.

Specify the image memory page number (0 or 1) where the ROI model is to be drawn with the *page#* parameter. The default setting is 0.

Specify the upper-left and lower-right corners of the rectangular region with points *(X1*, *Y1)* and *(X2*, *Y2)*. The enlargement/reduction ratio is determined automatically from the size of the registered ROI model and the size of the rectangular region defined by *(X1*, *Y1)* and *(X2*, *Y2)*.

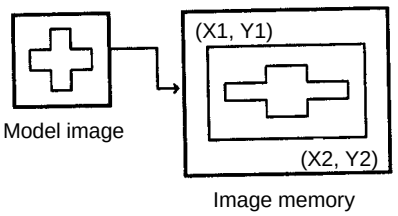
The *window function* parameter specifies which image(s) are to be drawn in window memory. The default setting is 0.

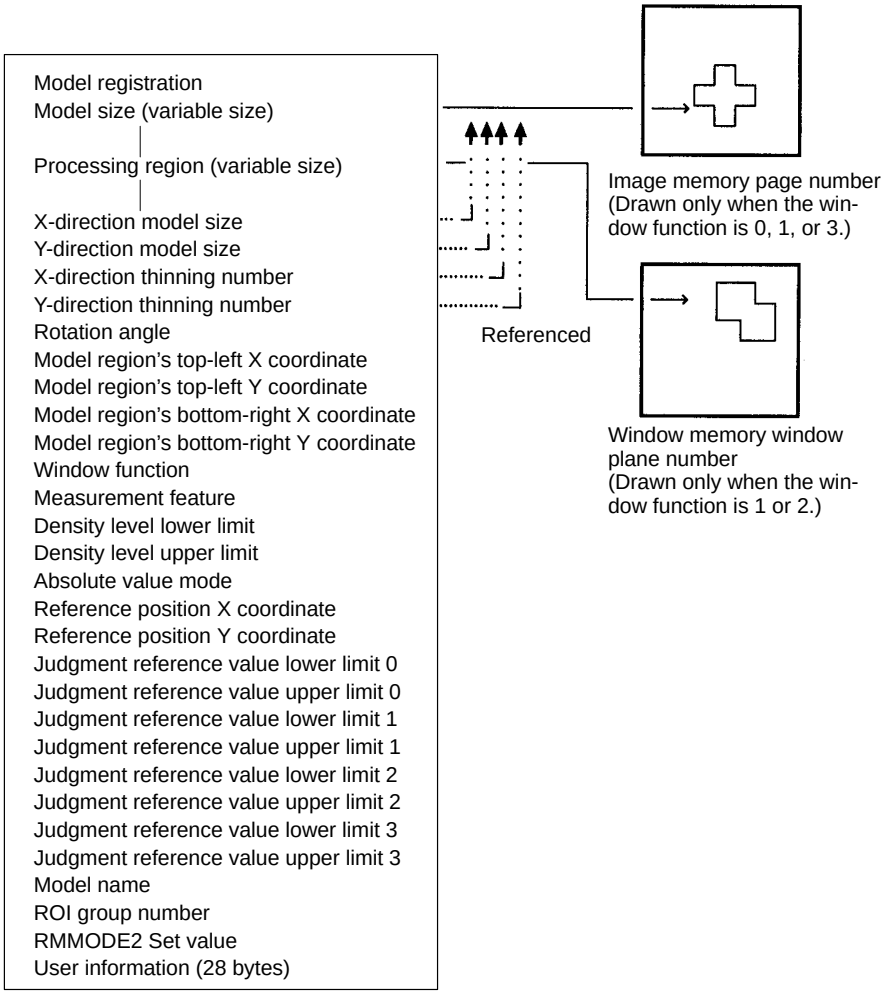
Setting	Function
0	Draw the model image only.
1	Draw the model image and the processing region.
2	Draw the processing region only.
3	Draw the model image and the processing region without updating the image data outside the region.
4	Draw the processing region only without updating the image data outside the region.

The *window plane#* parameter specifies the plane (0 to 7) in which the processing region is to be drawn. The default setting is 7.

The processing region will be drawn in the window memory specified by the *window plane#*.

When thinning registration is performed during model registration, a mosaic pattern image is drawn. When the ROI model image and the rectangular region don't have the same X/Y ratio, the enlargement/reduction X/Y ratio is calculated automatically so the image fills the rectangular region.





RMDRAW2 ROI Model DRAW 2

(Command)

Action Draws an enlarged or reduced ROI model in image memory.

Format RMDRAW2 *model#* [, [*page#*] [, [*X*], [*Y*] [, [*ratio*] [, [*window function*] [, *window plane#*]]]]]

Description Draws the ROI model specified by *model#* in image memory.

Specify the image memory page number (0 or 1) where the ROI model is to be drawn with the *page#* parameter. The default setting is 0.

Specify the upper-left corner of the region where the image is to be drawn with point *(X, Y)*. If these coordinates are omitted, the coordinates of the upper-left corner of the registered ROI model will be used.

Specify the enlargement/reduction ratio (0.001 to 512) with the *ratio* parameter. A decimal point can be specified. The default ratio is 1.0.

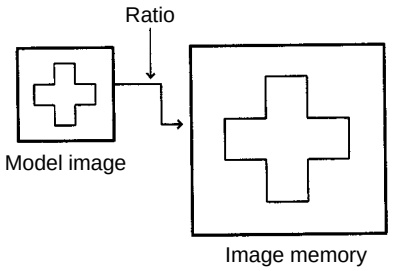
The *window function* parameter specifies which image(s) are to be drawn in window memory. The default setting is 0.

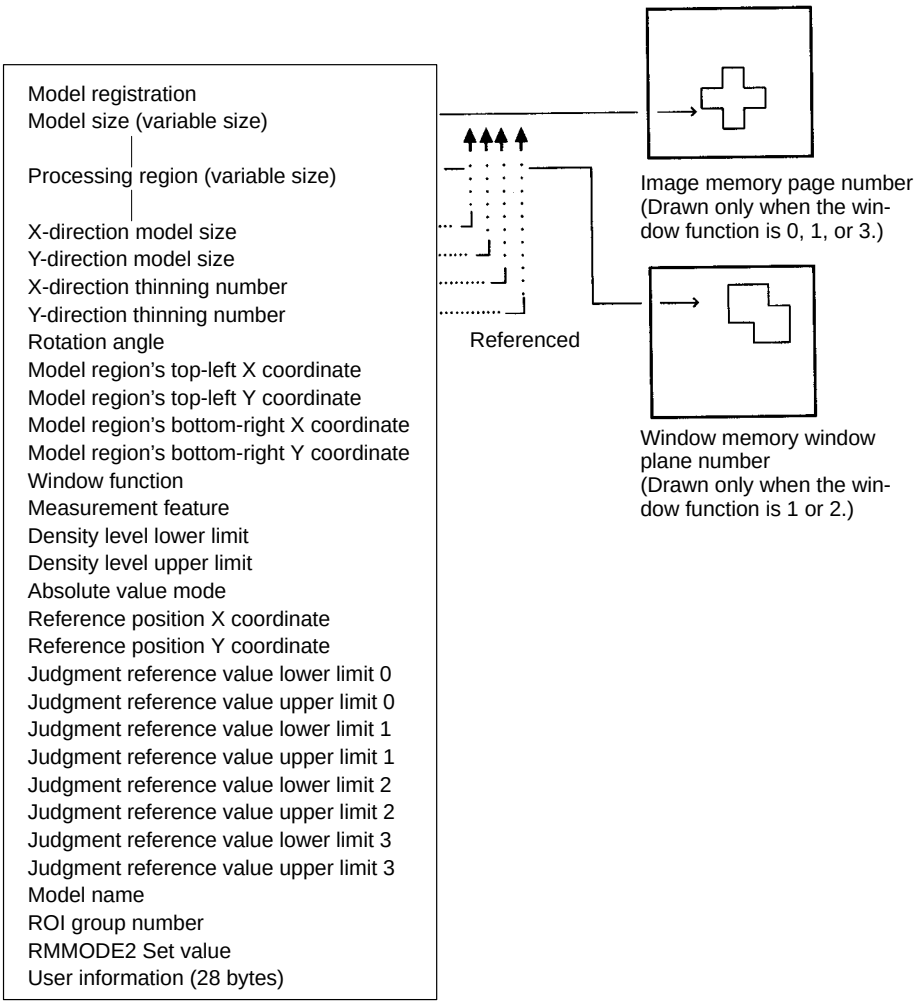
Setting	Function
0	Draw the model image only.
1	Draw the model image and the processing region.
2	Draw the processing region only.
3	Draw the model image and the processing region without updating the image data outside the region.
4	Draw the processing region only without updating the image data outside the region.

The *window plane#* parameter specifies the plane (0 to 7) in which the processing region is to be drawn. The default setting is 7.

The processing region will be drawn in the window memory specified by the *window plane#*.

When thinning registration is performed during model registration, a mosaic pattern image is drawn.





RMEXTGET ROI Model EXTra data GETting

(Function)

Action Reads user information from the ROI model data region.

Format RMEXTGET (*model#*, [*displacement*], *data format*)

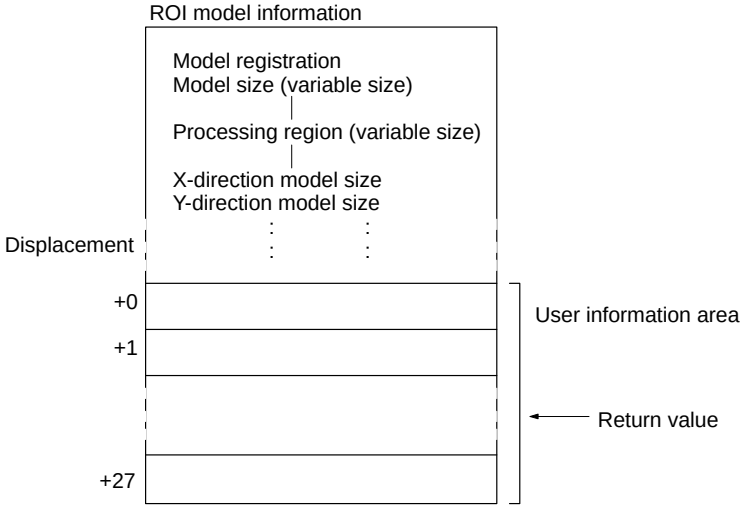
Description Reads the user information that was set with the RMEXTSET command. Specify the model number of the desired ROI model in *model#*. The model number can be set from 0 to 1023.

Specify the number of bytes to *displace* from the beginning of the user information area. The *displacement* can be set from 0 to 26.

Specify the type of data to be read in the *data format* parameter.

0: Integer (2 bytes)
1: Long integer (4 bytes)
2: Single precision number (4 bytes)
3: Double precision number (8 bytes)

In order to read the data correctly, the user must know what type of data is stored at each displacement position in the user information area



RMEXTGET$ ROI Model EXTra data GETting $

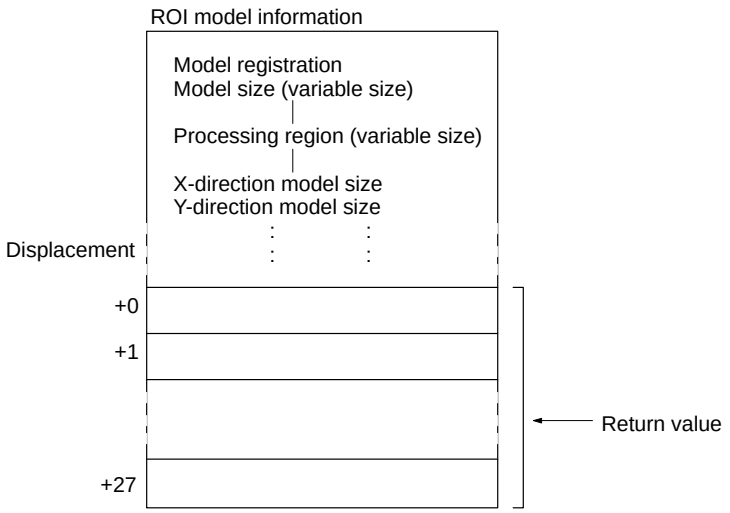
(Function)

Action Reads user information (character string) from the ROI model data region.

Format RMEXTGET$ (*model#*, [*offset*])

Description Reads user information that was set with the RMEXTSET command as character string data. Specify the model number of the desired ROI model in *model#*. The model number can be set from 0 to 1023.

Specify the number of bytes to *offset* from the beginning of the user information area. The *offset* can be set from 0 to 26.



RMEXTSET ROI Model EXTra data SETting

(Command)

Action Sets user information in the ROI model data region.

Format RMEXTSET *model#*, [*offset*], *data* [, *data*...]

Description Sets arbitrary user information into the user information area of the specified ROI model. For example, this area can be used to store what kind of filtering was performed when the ROI model was registered or the time that it was registered.

There are 28 bytes in the user information area and the user can store any type of data in any of these addresses. The user information has no effect whatsoever on ROI processing.

The user information is managed separately in each ROI model and is controlled by the RMCOPY, RMCLEAR, RMSAVE, and RMGSAVE commands; it is retained during power interruptions.

Specify the model number of the desired ROI model in *model#*. The model number can be set from 0 to 1023.

Specify the number of bytes to *offset* from the beginning of the user information area. The *offset* can be set from 0 to 26.

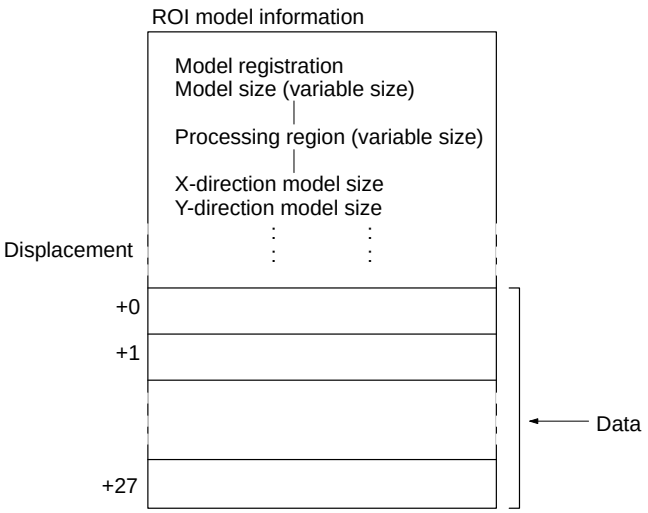
Specify the desired data in the *data* parameter(s). When a constant is specified, it is treated as an integer. When a variable is specified, the required number of bytes is determined by the variable type. The following table shows the required number of bytes for each variable.

Variable type	Required number of bytes
Simple character variable	Number of characters + 1
Simple integer variable	2
Simple long integer variable	4
Simple single precision variable	4
Simple double precision variable	8

An array variable can't be specified in the *data* parameter.

When two or more *data* parameters are specified, the data will be stored in order from the specified *displacement* location. An error will occur if the data can't be stored in the user information area.

Use the RMEXTGET command to read the data from the user information area. The user must manage the structure of the user information area. Keep track of each data element's displacement and data type.



RMGCLEAR ROI Model Group CLEAR

(Command)

Action Clears and initializes the specified ROI groups.

Format RMGCLEAR *start group#* [, *end group#*]

Description The RMGCLEAR command initializes the information of ROI models belonging to the groups from the specified *start group#* to the specified *end group#*. Specify the desired start and end groups with the *start group#* and *end group#*. These numbers can be set from 0 to 511. If the end group number is omitted, just the information for ROI models belonging to the start group will be initialized.

ROI model information is retained even when the power is interrupted or the OVL program is cleared. Use the RMGCLEAR command to initialize the ROI models when an entirely new OVL program is loaded or the model information has been corrupted.

ROI models between the start group# and end group#

Model registration	← 0 (not registered)
Model size (variable size)	← All 0
Processing region (variable size)	← All 0
X-direction model size	← 1
Y-direction model size	← 1
X-direction thinning number	← 1
Y-direction thinning number	← 1
Rotation angle	← 0
Model region's top-left X coordinate	← 0
Model region's top-left Y coordinate	← 0
Model region's bottom-right X coordinate	← 0
Model region's bottom-right Y coordinate	← 0
Window function	← 0
Measurement feature	← 3 (gray correlation)
Density level lower limit	← 0
Density level upper limit	← 255
Absolute value mode	← 0
Reference position X coordinate	← 0
Reference position Y coordinate	← 0
Judgment reference value lower limit 0	← 0
Judgment reference value upper limit 0	← 0
Judgment reference value lower limit 1	← 0
Judgment reference value upper limit 1	← 0
Judgment reference value lower limit 2	← 0
Judgment reference value upper limit 2	← 0
Judgment reference value lower limit 3	← 0
Judgment reference value upper limit 3	← 0
Model name	← Null string
ROI group number	← –1 (no assignment)
RMMODE2 Set value	← All 0
User information (28 bytes)	← All 0

RMGDEL ROI Model Group DELete

(Command)

Action Deletes an ROI model from an ROI group.

Format RMGDEL *group#*, *model#*

Description The RMGDEL command deletes the specified ROI model from the specified ROI group. Specify the desired ROI group with the *group#* (0 to 511) and the desired ROI model with *model#* (0 to 1023). The command can be executed even if the specified ROI model isn't registered in the specified ROI group.

RMGINFO ROI Model Group INFOrmation

(Function)

Action Reads ROI group information for the specified group.

Format RMGINFO (*group#*, *data type* [, *displacement position*])

Description Reads the information for the ROI group specified in *group#*. Specify the desired ROI group with the *group#* parameter (0 to 511).

Specify the type of data to be read in the *data type* parameter.

0: number of models (0 to 1024) belonging to the group
1: the model number of the model in the specified *displacement position* (0 to 1023)

If the *data type* parameter is set to 1, RMGINFO will return the model number of the model in the specified *displacement position*. The first ROI model in the group is at displacement position 0, the second is at displacement position 1, and so on. The default setting for the displacement position is 0.

A value of –1 will be returned if the specified displacement position is greater than or equal to the number of ROI models in the group.

Information for group m

Number of ROI models in group# m (n)	→ RMGINFO(m,0)
ROI model number in position 0	→ RMGINFO(m,1,0), RMGINFO(m,1)
ROI model number in position 1	→ RMGINFO(m,1,)
: :	
: :	
: :	
ROI model number in position n–1	→ RMGINFO(m,1,n–1) –1 → RMGINFO(m,1,n)

RMGJUDGE ROI Model Group JUDGEment

(Command)

Action Sets the judgement criteria for the ROI group.

Format RMGJUDGE *group#*, *data type*, *lower limit, upper limit*

Description The RMGJUDGE command sets the lower and upper limits used in judgement processing for the ROI models belonging to the specified ROI group.

The judgement criteria set here are used in the ROI processing executed by the RMRUN, RMGRUN, and SQRUN commands; the judgement results are read by commands such as RMDATA, RMMDATA, and SQJDATA. The judgement criteria are used when processing ROI models in which the measurement features have been set for judgement.

Specify the desired ROI group with the *group#* parameter (0 to 511). The judgement conditions will be set in the specified ROI group.

The meaning of the *data type* parameter is different for the RMMODE command and RMMODE2 command, as described below.

Measurement feature set in RMMODE or RMMODE2	**Data type**			
	0	**1**	**2**	**3**
Binary features	Lower and upper limits are meaningless because the judgement results can't be read.			
Gray features				
Binary weighting correlation				
Gray correlation				
Binary feature judgement	Binary area	X center-of-gravity (binary)	Y center-of-gravity (binary)	---

Measurement feature set in RMMODE or RMMODE2	Data type			
	0	1	2	3
Gray feature judgement	Average density	X center-of-gravity (gray)	Y center-of-gravity (gray)	---
Binary weighting correlation judgement	Binary weighting correlation	Lower and upper limits are meaningless.		
Gray correlation judgement	Gray correlation	Average density	Density deviation	---
Precision search	Gray correlation	Precision X center-of-gravity	Precision X center-of-gravity	---
Region processing	Large defect	Small defect	Max. density	Min. density

The judgements are made as follows:

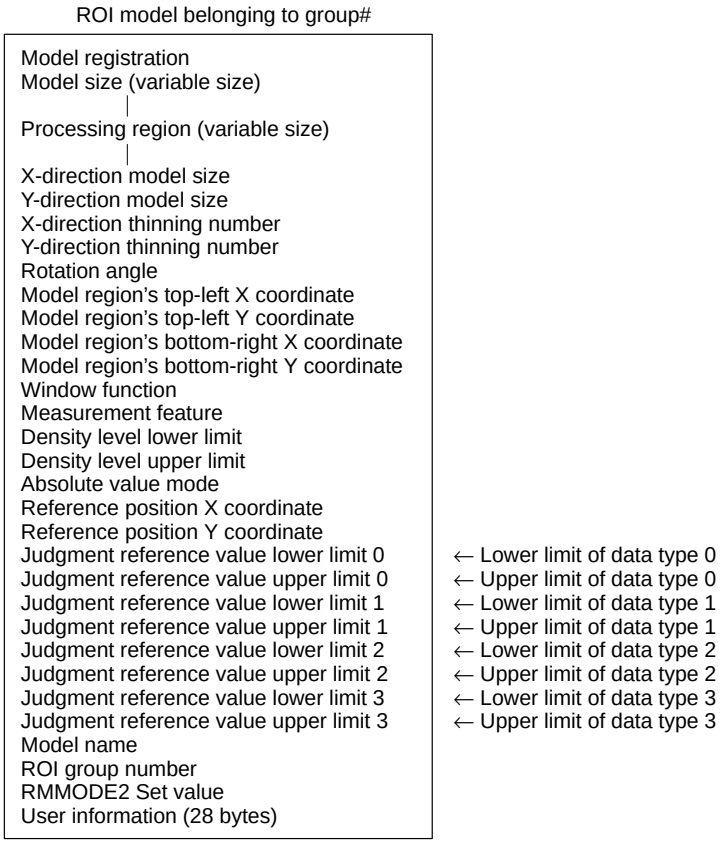
A value of 0 (OK) is returned if the measurement result is between the lower and upper limits (lower limit ≤ result ≤ upper limit).

A value of –1 (no good) is returned if the measurement result isn't between the lower and upper limits (result < lower limit or upper limit < result).

No matter what lower limit value is set for large and small defects in region processing, a value of 0 will be used.

The upper limit value must conform to the limits for the measurement feature set with the RMMODE command.

Binary area:	0 to 247808
Binary X center-of-gravity:	0 to 511
Binary Y center-of-gravity:	0 to 483
Average density:	0 to 255
Gray X center-of-gravity:	0 to 511
Gray Y center-of-gravity:	0 to 483
Binary weighting correlation:	0 to 100
Gray correlation:	0 to 100
Density deviation:	0 to 127.5
Precision X center-of-gravity:	0 to 511
Precision Y center-of-gravity:	0 to 483
Large defect:	0 to 255
Small defect:	0 to 255
Maximum density:	0 to 100
Minimum density:	0 to 100



Even if the measurement features are changed with RMMODE or RMGMODE, the lower and upper limits recorded in each ROI model won't be changed. For example, if an upper limit of 1000 were set for the binary area and then that model's measurement feature were changed to "gray feature judgement," the meaningless upper limit of 1000 would be used as the upper limit for the average density.

RMGLOAD ROI Model Group LOAD

(Command)

Action Retrieves the ROI model data from the memory card and registers the data in the system.

Format RMGLOAD *file name* [, *group#*] [, *processing mode*]

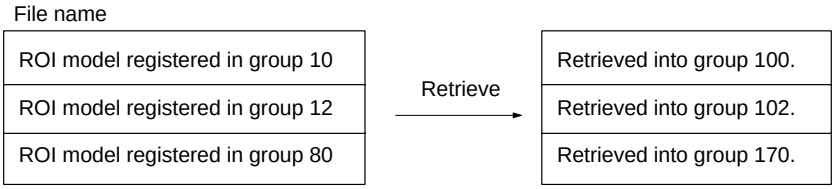
Description Retrieves the ROI model data specified by the *file name* parameter and registers the data in the system's ROI model memory. The data can't be retrieved unless the file was saved with the RMGSAVE or RMSAVE command.

Specify the destination ROI group (0 to 511), where the ROI model is loaded with the *group#* parameter. Since the ROI model can be registered in only one group, if loaded to another group, the ROI model will be deleted from the group that was saved. If the *group#* is omitted, the data will be retrieved into the same group number as the one that was saved. Normally, the group number is omitted and the same group number is used for saving and loading.

The following equation is used to determine the group in which the model data will belong.

Group = (group number when loaded)+(*group#*–group number that the first model belonged to when the file was created.)

A group# of 100 was specified in the following example.



Specify the model number of the load destination for the *processing mode* parameter.

0: Registers into the unregistered model numbers. Be sure that there is enough space available for the ROI models being loaded.

1: Overwrites the model numbers when saved.

The default setting is 0.

If the group into which the ROI model is to be retrieved hasn't been set, the group still won't be set after the ROI model is retrieved.

Even if an error occurs while the ROI model is being retrieved, the ROI model data retrieved up to that point will be valid.

RMGMODE ROI Model Group MODE

(Command)

Action Determines the ROI group's processing mode.

Format RMGMODE *group#*, *measurement feature* [, *density lower limit* [, *density upper limit* [, *absolute value mode*]]]

Description Sets the conditions used when ROI processing is executed for all of the ROI models belonging to the specified group. Specify the desired ROI group with the *group#* parameter (0 to 511).

The *measurement feature* parameter specifies which feature is measured for the specified ROI models.

Values specified by the measurement feature		Storage of measurement results			
		Feature 1	**Feature 2**	**Feature 3**	**Feature 4**
0	Binary features	Binary area	X center-of-gravity	Y center-of-gravity	---
1	Gray features	Average density	X center-of-gravity	Y center-of-gravity	---
2	Binary weighting correlation	Binary weighting correlation value	---	---	---
3	Gray correlation	Gray correlation value	Average density	Density deviation	---
4	Binary feature judgement	Binary area judgement	X center-of-gravity judgement	Y center-of-gravity judgement	---
5	Gray feature judgement	Average density judgement	X center-of-gravity judgement	Y center-of-gravity judgement	---
6	Binary weighting correlation judgement	Binary weighting correlation value judgement	---	---	---
7	Gray correlation judgement	Gray correlation value judgement	Average density judgement	Density deviation judgement	---

For example, if the *measurement feature* parameter is set to 3, the gray correlation value, average density, and density deviation can be read simultaneously when ROI processing is performed with the RMRUN command.

A value of –2 will be returned for the judgement result when a measurement feature that doesn't include judgement processing is selected.

Measurement feature 4 in the table above hasn't been completed yet.

A processing result of 0 and judgement result of –2 will be returned for blank items (---) in the table above.

The density lower limit (0 to 255) and density upper limit (0 to 255) set the density levels used in image clipping. The default setting for the density lower limit is 0 and the default setting for the density upper limit is 255. The previous settings will remain valid if these parameters are omitted.

The absolute value mode is used to reverse the above density level. The default setting is 0.

0: Do not reverse.
Other than 0: Reverse.

The gray correlation value for ROI processing is 0 to 100; Negative correlation values are set to 0.

RMGROUP ROI Model GROUP

(Command)

Action Registers a single ROI model as an ROI group.

Format RMGROUP *group#*, *model#*

Description Registers the specified ROI model in an ROI group. Specify the desired ROI group with the *group#* (0 to 511) and the desired ROI model with *model#* (0 to 1023).

A single ROI model can't be registered in more than one group. If the specified model number is already registered in another group, the model will be deleted from that group and registered again in the specified group.

RMGRUN ROI Model Group RUN

(Command)

Action Executes ROI processing for the models in the group.

Format RMGRUN *group#*, [*X*], [*Y*]

Description Executes ROI processing for all of the ROI models in the specified group. Specify the desired ROI group with the *group#* parameter (0 to 511).

The *X* and *Y* parameters specify the reference coordinates used in ROI process; The point (*X*, *Y*) must be within the range (0, 0) through (511, 483). Each ROI model's processing position is added to the displacement set with the RMORIGIN command from this reference point and this position is treated as the upper-left corner of the model region.

When the *X* and *Y* parameters are omitted, the X and Y coordinates when the ROI model was registered are used as the reference position.

Read the ROI processing results with the RMDATA or RMMDATA command.

RMGSAVE ROI Model Group SAVE

(Command)

Action Saves the ROI model data belonging to the specified ROI group to the memory card.

Format RMGSAVE *file name*, *group#* [*FOR APPEND*]

Description Saves the data of the ROI models belonging to the specified ROI group in the memory card. Specify the file in which the data will be saved with the *file name* parameter.

Specify the desired ROI group (0 to 511) with the *group#* parameter.

Data can't be saved to the RS-232C port.

When the "FOR APPEND" argument is specified, data will be appended to the specified file. Use this argument when saving several ROI groups in a single file.

Data can be appended to files saved with the RMGSAVE and RMSAVE commands, but not to files saved with the SMGSAVE and SMSAVE commands.

RMGSAVE *file name*, *group#* FOR APPEND

OK	File saved with RMSAVE.
OK	File saved with RMGSAVE.
Not allowed.	File saved with SMGSAVE.
Not allowed.	File saved with SMSAVE.

The following diagram shows the structure of files saved with RMGSAVE.

	File header (48 bytes)
Model number of number 0 in the group →	Model number when saved (2 bytes)
Model info. of number 0 in the group →	Model data (variable length)
Model number of number 1 in the group →	Model number when saved (2 bytes)
Model info. of number 1 in the group →	Model data (variable length)
	: :
	: :
	Model number when saved (2 bytes)
	Model data (variable length)

When appending data, model numbers and model data will be appended to the existing file.

RMINFO ROI Model INFOrmation

(Function)

Action Reads ROI model information.

Format RMINFO (*model#*, *data type* [, *data#*])

Description Reads the information for the specified ROI model. Specify the desired ROI model with the *model#* parameter (0 to 1023).

Specify the type of data to be read in the *data type* parameter.

When the *data type* parameter is set to 21, the lowest unregistered model number after the number specified in *model#* will be returned.

Data type		Returned value
0	Model registration	0: Not registered 1: Registered.
1	X-direction model size	0 to 512
2	Y-direction model size	0 to 484
3	X-direction thinning number	1 to 511
4	Y-direction thinning number	1 to 483
5	Rotation angle	–180° to 180°
6	Model region's top-left X coordinate	0 to 511
7	Model region's top-left Y coordinate	0 to 483
8	Model region's bottom-right X coordinate	0 to 511
9	Model region's bottom-right Y coordinate	0 to 483
10	Window function	0: OFF 1: ON

Data type		Returned value
11	Measurement feature	0: Binary feature 1: Gray feature 2: Binary weighting correlation 3: Gray correlation 4: Binary feature judgement 5: Gray feature judgement 6: Binary weighting correlation judgement 7: Gray correlation judgement
12	Density level lower limit	0 to 255
13	Density level upper limit	0 to 255
14	Absolute value mode	0: Normal 1: Reverse mode
15	Processing position offset, X coordinate	–512 to 511
16	Processing position offset, Y coordinate	–512 to 511
17	Judgment reference value lower limit	See the RMGJUDGE (page 177) and RMJUDGE (page 185) commands.
18	Judgment reference value upper limit	See the RMGJUDGE (page 177) and RMJUDGE (page 185) commands.
19	ROI group number	0 to 511
20	Number of ROI models registered	0 to 1024
21	Open ROI model number (Lowest unregistered ROI model number)	0 to 1023
22	Bytes of unused ROI model data	0 to ...
23	RMMODE2 processing mode	0: Fixed position processing (default setting) 1: Precision search (no rotation) 2: Precision search (with rotation) 3: Straight line region inspection 4: Arc region inspection 5: Circle region inspection 6: Arbitrary region inspection 7: Number of defects on line acquisition 8: Number of defects on arc acquisition 9: Number of defects on circle acquisition 10: Number of defects in arbitrary region 11: Gray level follow-up measurement
24	RMMODE2 set value	See the following explanation.
100	Error information	–2: An error occurred but initialization executed. –1: No error 0 to 1023: First model number where an error was detected.

The *data#* parameter specifies the desired information; it is valid when the *data type* parameter is set to 17, 18, or 24.

When the *data type* parameter is set to 24, the data# setting (0 to 8) specifies the element number of the parameter specified in the RMMODE2 command's array.

Data#	Fpp*	Precision search (no rotation)	Precision search (w/rotation)	Linear region inspection	Arc region inspection	Circle region inspection	Arbitrary region inspection	Density level follow-up
0	---	Search region X size	Search region X size	Black/white flag	Black/white flag	Black/white flag	Black/white flag	Density level follow-up mode
1	---	Search region Y size	Search region Y size	Inspection items	Inspection items	Inspection items	Inspection items	Model no. for level follow-up
2	---	Frame/Field mode	Start rotation angle	Mask interval	Mask interval	Mask interval	Mask interval	Model image's average density (× 1,000)
3	---	Estimated value judgement value (× 1,000)	End rotation angle × 100	Scan X vector (× 1,024)	Scan start angle (× 1,024)	Scan start angle (× 1,024)	Scan X pitch (× 1,024)	Model image's density deviation (× 1,000)
4	---	---	Step angle × 100	Scan Y vector (× 1,024)	Scan step angle (× 1,024)	Scan step angle (× 1,024)	Scan Y pitch (× 1,024)	Correlation judgement value (× 1,000)
5	---	---	Frame/Field mode	Number of scans	Number of scans	Number of scans	Number of X scans	---
6	---	---	---	---	Arc center X coord.	Center X coord.	Number of Y scans	---
7	---	---	Estimated value judgement value (× 1,000)	---	Arc center Y coord.	Center Y coord.	Mask X size	---
8	---	---	---	---	Circle radius	Circle radius	Mask Y size	---

Note *Fpp stands for Fixed position processing.

RMINFO$ ROI Model INFOrmation $

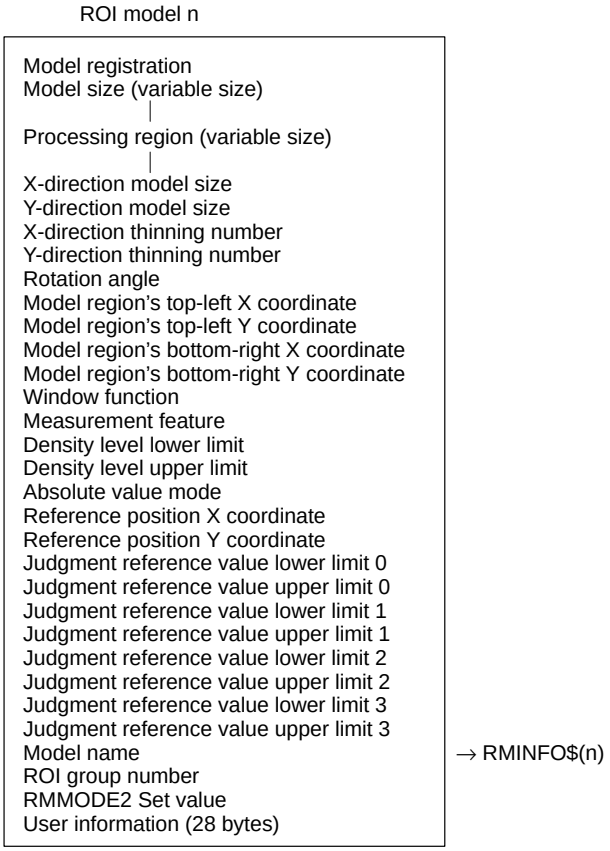
(Function)

Action Reads the specified ROI model's model name.

Format RMINFO$ (*model#* [, *data type*])

Description Reads the specified ROI model's model name. Specify the desired ROI model with the *model#* parameter (0 to 1023).

Specify 0 for the *data type* parameter or omit it.



RMJUDGE ROI Model JUDGEment

(Command)

Action Sets the judgement criteria for an ROI model.

Format RMJUDGE *model#*, *data type*, *lower limit, upper limit*

Description The RMJUDGE command sets the judgement criteria (lower and upper limits) used in judgement processing for the specified ROI model.

The judgement criteria set here are used in the ROI processing executed by the RMRUN, RMGRUN, and SQRUN commands; the judgement results are read by commands such as RMDATA, RMMDATA, and SQJDATA.

Specify the desired ROI model with the *model#* parameter (0 to 1023).

The meaning of the *data type* parameter is different for the RMMODE command and RMMODE2 command, as described below.

Measurement feature set in RMMODE or RMMODE2	Data type			
	0	1	2	3
Binary features	Lower and upper limits are meaningless because the judgement results can't be read.			
Gray features				
Binary weighting correlation				
Gray correlation				
Binary feature judgement	Binary area	X center-of-gravity (binary)	Y center-of-gravity (binary)	---
Gray feature judgement	Average density	X center-of-gravity (gray)	Y center-of-gravity (gray)	---
Binary weighting correlation judgement	Binary weighting correlation	Lower and upper limits are meaningless.		
Gray correlation judgement	Gray correlation	Average density	Density deviation	---

Measurement feature set in RMMODE or RMMODE2	Data type			
	0	1	2	3
Precision search	Gray correlation	Precision X center-of-gravity	Precision Y center-of-gravity	Precision angle
Region processing	Large defect	Small defect	Max. density	Min. density
Number of defects inspection	Large defect	Small defect	Max. density	Min. density

Measurement feature set in RMMODE or RMMODE2	Data type			
	4	5	6	7
Number of defects inspection	Number of large defects	Number of small defects	Number of density defects	---

The *data type* can be between 0 and 6 for the number of defects inspection and between 0 and 3 for other measurement features. An *illegal function call* error will be generated if an invalid value is used.

The judgements are made as follows:

A value of 0 (OK) is returned if the measurement result is between the lower and upper limits (lower limit ≤ result ≤ upper limit).

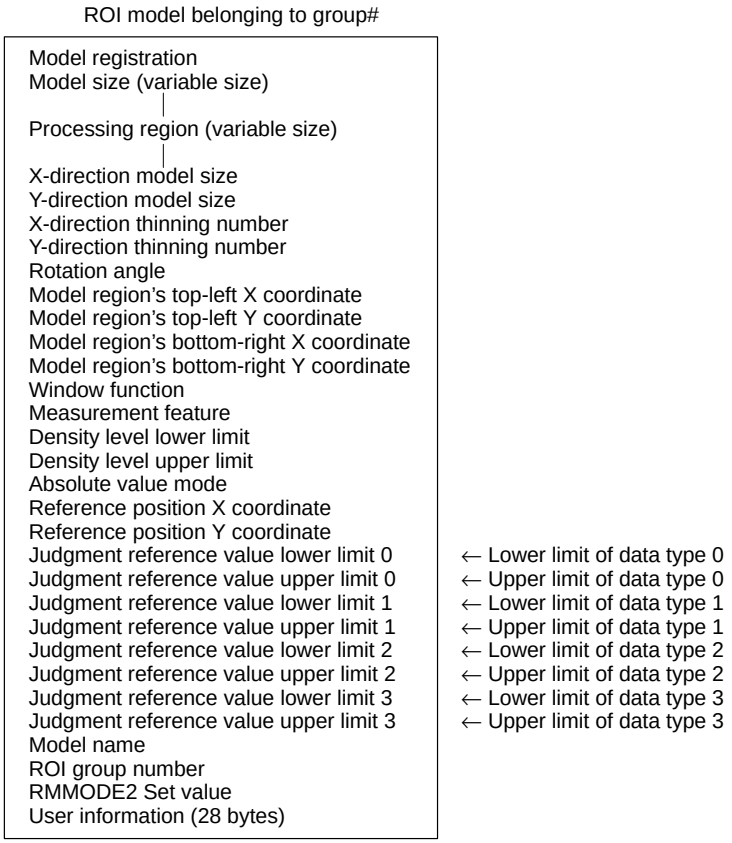
A value of –1 (no good) is returned if the measurement result isn't between the lower and upper limits (result < lower limit or upper limit < result).

No matter what lower limit value is set for large and small defects in region processing, a value of 0 will be used.

The lower and upper limit values must conform to the limits for the measurement feature set with the RMMODE command.

Binary area:	0 to 247808
Binary X center-of-gravity:	0 to 511
Binary Y center-of-gravity:	0 to 483
Average density:	0 to 255
Gray X center-of-gravity:	0 to 511
Gray Y center-of-gravity:	0 to 483
Binary weighting correlation:	0 to 100
Gray correlation:	0 to 100
Density deviation:	0 to 127.5
Precision X center-of-gravity:	0 to 511
Precision Y center-of-gravity:	0 to 483
Large defect:	0 to 255
Small defect:	0 to 255
Maximum density:	0 to 100
Minimum density:	0 to 100
Number of large defects:	0 to 32,767
Number of small defects:	0 to 32,767
Number of density defects:	0 to 32,767

Even if the measurement features are changed with RMMODE or RMGMODE, the lower and upper limits recorded in each ROI model won't be changed. For example, if an upper limit of 1000 were set for the binary area and then that model's measurement feature were changed to "gray feature judgement," the meaningless upper limit of 1000 would be used as the upper limit for the average density.



RMLOAD ROI Model LOAD

(Command)

Action Retrieves the ROI model data from the memory card.

Format RMLOAD *file name*

Description Retrieves the ROI model data specified by the *file name* parameter and stores the data in memory. The data can't be retrieved unless the file was saved with the RMGSAVE or RMSAVE command.

The data will be retrieved into the same model number from which it was saved.

RMMDATA ROI Model Multi DATA

(Command)

Action Reads a batch of ROI processing results at a different time from execution.

Format RMMDATA *number of models*, *model# array*, *data type*, [*array 1*] [, [*array 2*] [, [*array 3*] [, [*array 4*] [, [*array 5*] [, [*array 6*] [, [*array 7*] [, [*array 8*] [, [*array 9*] [, [*array 10*] [, *array 11*]]]]]]]]]]

Description The RMMDATA command reads the ROI processing results from single ROI processing (RMRUN or RMGRUN commands) or sequential ROI processing for several ROI models simultaneously.

The data is read in the order that the ROI model numbers appear in the *model# array*. The *number of models* parameter specifies how many model numbers are stored in the model number array.

The *model# array* parameter specifies the one-dimensional model number array that contains the ROI model numbers. This array must be declared in advance with the DIM command. The model numbers can be set from 0 to 1023.

The *data type* parameter specifies what type of data is to be read, as shown below.

0: Information on fixed position processing
1: Information on judgement results from fixed position processing
2: Information on precision search
3: Information on judgement results from precision search
4: Information on the most probable large defect
5: Information on the most probable small defect
6: Information on the most probable minimum density
7: Information on the most probable maximum density
8: Information on density level follow-up measurement processing
9: Information on density level follow-up measurement judgement results

A value of 0 will be stored in arrays 1 through 4 when the data type differs from the one specified in the ROI model's measurement feature or processing mode. If an item in the model number array has a value of –1, results won't be stored in arrays 1 through 4.

Specify the one-dimensional arrays that will contain the results in *array 1* through *array 4*. The data stored in the arrays is determined by the measurement feature set with the RMMODE command and the processing mode set with the RMMODE2 command, as shown in the following tables.

The following list provides details on the data stored in the arrays.

1, 2, 3... 1. Data type = 0 (fixed position processing information)

RMMODE's measurement feature = 0 (binary features):
- Array1 (n) = Binary area (0 to 247808)
- Array2 (n) = Binary X center-of-gravity (0 to 511)
- Array3 (n) = Binary Y center-of-gravity (0 to 483)
- Array4 (n) = (Reserved, unpredictable value)
- Array5 (n) = Binary X center-of-gravity conversion coord. before image scroll
- Array6 (n) = Binary Y center-of-gravity conversion coord. before image scroll
- Array7 (n) = X execution position (0 to 511)
- Array8 (n) = Y execution position (0 to 483)

RMMODE's measurement feature = 1 (gray features):
- Array1 (n) = Average density (0 to 255)
- Array2 (n) = Gray X center-of-gravity (0 to 511)
- Array3 (n) = Gray Y center-of-gravity (0 to 483)
- Array4 (n) = (Reserved, unpredictable value)
- Array5 (n) = Gray-scale X center-of-gravity conversion coord. before image scroll
- Array6 (n) = Gray-scale Y center-of-gravity conversion coord. before image scroll
- Array7 (n) = X execution position (0 to 511)
- Array8 (n) = Y execution position (0 to 483)

RMMODE's measurement feature = 2 (binary weighting correlation):
- Array1 (n) = Binary weighting correlation value
- Array2 (n) = (Reserved, contents not updated)
- Array3 (n) = (Reserved, contents not updated)
- Array4 (n) = (Reserved, contents not updated)
- Array5 (n) = (Reserved, contents not updated)
- Array6 (n) = (Reserved, contents not updated)
- Array7 (n) = X execution position (0 to 511)
- Array8 (n) = Y execution position (0 to 483)

RMMODE's measurement feature = 3 (gray correlation):
- Array1 (n) = Gray correlation value (0 to 100)

Array2 (n) = Average density (0 to 255)
Array3 (n) = Density deviation (0 to 127.5)
Array4 (n) = (Reserved, contents not updated)
Array5 (n) = (Reserved, contents not updated)
Array6 (n) = (Reserved, contents not updated)
Array7 (n) = X execution position (0 to 511)
Array8 (n) = Y execution position (0 to 483)

2. Data type = 1 (judgement results from fixed position processing)
A judgement result of 0 indicates “OK,” –1 indicates “no good.”

RMMODE’s measurement feature = 4
(binary feature judgement):
Array1 (n) = Binary area judgement (0, –1, –2)
Array2 (n) = Binary X center-of-gravity judgement (0, –1, –2)
Array3 (n) = Binary Y center-of-gravity judgement (0, –1, –2)

RMMODE’s measurement feature = 5
(gray feature judgement):
Array1 (n) = Average density judgement (0, –1, –2)
Array2 (n) = Gray X center-of-gravity judgement (0, –1, –2)
Array3 (n) = Gray Y center-of-gravity judgement (0, –1, –2)

RMMODE’s measurement feature = 6
(binary weighting correlation judgement):
Array1 (n) = Binary weighting corr. value judgement (0, –1, –2)

RMMODE’s measurement feature = 7 (gray correlation judgement):
Array1 (n) = Gray correlation value (0, –1, –2)
Array2 (n) = Average density judgement (0, –1, –2)
Array3 (n) = Density deviation judgement (0, –1, –2)

3. Data type = 2 (precision search information)

RMMODE2’s processing mode = 1 (precision search without rotation):
Array1 (n) = Search position X coord. (0.000 to 511.000)
Array2 (n) = Search position Y coord. (0.000 to 483.000)
Array4 (n) = Search position correlation value (0 to 100)
Array5 (n) = Search X conversion coord. before image scroll
Array6 (n) = Search Y conversion coord. before image scroll

RMMODE2’s processing mode = 2 (precision search with rotation):
Array1 (n) = Search position X coord. (0.000 to 511.000)
Array2 (n) = Search position Y coord. (0.000 to 483.000)
Array3 (n) = Search angle (–180° to 180°)
Array4 (n) = Search position correlation value (0 to 100)
Array5 (n) = Search X conversion coord. before image scroll
Array6 (n) = Search Y conversion coord. before image scroll
Array7 (n) = Search angle conversion value before image scroll

4. Data type = 3 (judgement results from precision search)
A judgement result of 0 indicates “OK,” –1 indicates “no good.”

RMMODE2’s processing mode = 1 (precision search without rotation):
Array1 (n) = Search position X coord. judgement (0, –1, –2)
Array2 (n) = Search position Y coord. judgement (0, –1, –2)
Array4 (n) = Search position correlation value judgement (0, –1, –2)

RMMODE2’s processing mode = 2 (precision search with rotation):
Array1 (n) = Search position X coord. judgement (0, –1, –2)
Array2 (n) = Search position Y coord. judgement (0, –1, –2)
Array3 (n) = Search angle judgement (0, –1, –2)
Array4 (n) = Search position correlation value judgement (0, –1, –2)

5. Data type = 4 (most probable large defect information)
   A judgement result of 0 indicates “OK,” –1 indicates “no good,” and –2 indicates that that item wasn’t processed.

   Array1 (n) = Most probable large defect’s X coord. (0 to 511)
   Array2 (n) = Most probable large defect’s Y coord. (0 to 483)
   Array3 (n) = Most probable large defect’s estimate (0 to 255)
   Array4 (n) = Large defect judgement (0, –1, –2)
   Array5 (n) = Most probable large defect’s X conversion coord. before image scroll
   Array6 (n) = Most probable large defect’s Y conversion coord. before image scroll
   Array7 (n) = Number of large defects
   Array8 (n) = Number of large defects judgement (0, –1, –2)

6. Data type = 5 (most probable small defect information)
   A judgement result of 0 indicates “OK,” –1 indicates “no good,” and –2 indicates that that item wasn’t processed.

   Array1 (n) = Most probable small defect’s X coord. (0 to 511)
   Array2 (n) = Most probable small defect’s Y coord. (0 to 483)
   Array3 (n) = Most probable small defect’s estimate (0 to 255)
   Array4 (n) = Small defect judgement (0, –1, –2)
   Array5 (n) = Most probable small defect’s X conversion coord. before image scroll
   Array6 (n) = Most probable small defect’s Y conversion coord. before image scroll
   Array7 (n) = Number of small defects
   Array8 (n) = Number of small defects judgement (0, –1, –2)

7. Data type = 6 (density minimum value information)
   A judgement result of 0 indicates “OK” and –1 indicates “no good.”

   Array1 (n) = Most probable min. density X coord. (0 to 511)
   Array2 (n) = Most probable min. density Y coord. (0 to 483)
   Array3 (n) = Most probable min. density estimate (0 to 255)
   Array4 (n) = Min. density judgement (0, –1, –2)
   Array5 (n) = Most probable min. density X conversion coord. before image scroll
   Array6 (n) = Most probable min. density Y conversion coord. before image scroll
   Array7 (n) = Number of density defects
   Array8 (n) = Number of density defects judgement (0, –1, –2)

   Data type 6 and data type 7 have the same values for the “number of density defects” and “number of density defects judgement”.

8. Data type = 7 (density maximum value information)
   A judgement result of 0 indicates “OK” and –1 indicates “no good.”

   Array1 (n) = Most probable max. density X coord. (0 to 511)
   Array2 (n) = Most probable max. density Y coord. (0 to 483)
   Array3 (n) = Most probable max. density estimate (0 to 255)
   Array4 (n) = Max. density judgement (0, –1, –2)
   Array5 (n) = Most probable max. density X conversion coord. before image scroll
   Array6 (n) = Most probable max. density Y conversion coord. before image scroll
   Array7 (n) = Number of density defects
   Array8 (n) = Number of density defects judgement (0, –1, –2)

   Data type 6 and data type 7 have the same values for the “number of density defects” and “number of density defects judgement”.

9. Data type = 8 (density level follow-up measurement information)

   RMMODE's measurement feature = 0 (binary features):
   - Array1 (n) = Binary area (0 to 247808)
   - Array2 (n) = Binary X center-of-gravity (0 to 511)
   - Array3 (n) = Binary Y center-of-gravity (0 to 483)
   - Array4 (n) = (Reserved, contents not updated)
   - Array5 (n) = Binary X center-of-gravity conversion coord. before image scroll
   - Array6 (n) = Binary Y center-of-gravity conversion coord. before image scroll
   - Array7 (n) = X execution position (0 to 511)
   - Array8 (n) = Y execution position (0 to 483)

   RMMODE's measurement feature = 1 (gray features):
   - Array1 (n) = Average density (0 to 255)
   - Array2 (n) = Gray-scale X center-of-gravity (0 to 511)
   - Array3 (n) = Gray-scale Y center-of-gravity (0 to 483)
   - Array4 (n) = (Reserved, contents not updated)
   - Array5 (n) = Gray-scale X center-of-gravity conversion coord. before image scroll
   - Array6 (n) = Gray-scale Y center-of-gravity conversion coord. before image scroll
   - Array7 (n) = X execution position (0 to 511)
   - Array8 (n) = Y execution position (0 to 483)

   RMMODE's measurement feature = 2 (binary weighting correlation):
   - Array1 (n) = Binary weighting correlation value
   - Array2 (n) = (Reserved, contents not updated)
   - Array3 (n) = (Reserved, contents not updated)
   - Array4 (n) = (Reserved, contents not updated)
   - Array5 (n) = (Reserved, contents not updated)
   - Array6 (n) = (Reserved, contents not updated)
   - Array7 (n) = X execution position (0 to 511)
   - Array8 (n) = Y execution position (0 to 483)

   RMMODE's measurement feature = 3 (gray correlation):
   - Array1 (n) = Gray correlation value (0 to 100)
   - Array2 (n) = Average density (0 to 255)
   - Array3 (n) = Density deviation (0 to 127.5)
   - Array4 (n) = (Reserved, contents not updated)
   - Array5 (n) = (Reserved, contents not updated)
   - Array6 (n) = (Reserved, contents not updated)
   - Array7 (n) = X execution position (0 to 511)
   - Array8 (n) = Y execution position (0 to 483)

10. Data type = 9 (density level follow-up measurement judgement information)

    A judgement result of 0 indicates "OK," –1 indicates "no good."

    RMMODE's measurement feature = 4:
    Binary features judgement
    - Array1 (n) = Binary area judgement (0, –1, –2)
    - Array2 (n) = Binary X center-of-gravity judgement (0, –1, –2)
    - Array3 (n) = Binary Y center-of-gravity judgement (0, –1, –2)

    RMMODE's measurement feature = 5:
    Gray features judgement
    - Array1 (n) = Average density judgement (0, –1, –2)
    - Array2 (n) = Gray-scale X center-of-gravity judgement (0, –1, –2)
    - Array3 (n) = Gray-scale Y center-of-gravity judgement (0, –1, –2)

RMMODE's measurement feature = 6:
Binary weighting correlations judgement
Array1 (n) = Binary weighting corr. value judgement (0, –1, –2)
RMMODE's measurement feature = 7:
Gray correlation judgement
Array1 (n) = Gray correlation value judgement (0, –1, –2)
Array2 (n) = Average density judgement (0, –1, –2)
Array3 (n) = Density deviation judgement (0, –1, –2)

Data won't be stored in arrays that weren't specified. A value of 0 will be stored when the array has been specified but data doesn't exist for the item that should be stored in the array element.

A value of –2 will be stored when a measurement feature that doesn't perform judgement processing has been specified, but a judgement result has been read.

RMMODE ROI Model MODE

(Command)

Action Determines the ROI model's processing mode.

Format RMMODE *model#*, *measurement feature* [, *density lower limit* [, *density upper limit* [, *absolute value mode*]]]

Description Sets the conditions used when ROI processing is executed for the specified ROI model. Specify the desired ROI model with the *model#* parameter (0 to 1023).

The *measurement feature* parameter specifies which feature is measured for the specified ROI model.

Measurement feature parameter setting		Measurement features read in ROI processing			
		Feature 1	Feature 2	Feature 3	Feature 4
0	Binary features	Binary area	X center-of-gravity	Y center-of-gravity	---
1	Gray features	Average density	X center-of-gravity	Y center-of-gravity	---
2	Binary weighting correlation	Binary weighting correlation	---	---	---
3	Gray correlation	Gray correlation value	Average density	Density deviation	---
4	Binary feature judgement	Binary area and Binary area judgement	X center-of-gravity and X center-of-gravity judgement	Y center-of-gravity and Y center-of-gravity judgement	---
5	Gray feature judgement	Average density and Average density judgement	X center-of-gravity and X center-of-gravity judgement	Y center-of-gravity and Y center-of-gravity judgement	---
6	Binary weighting correlation judgement	Binary weighting correlation and Binary weighting correlation judgement	---	---	---
7	Gray correlation judgement	Gray correlation and Gray correlation judgement	Average density and Average density judgement	Density deviation and Density deviation judgement	---

For example, if the *measurement feature* parameter is set to 3, the gray correlation value, average density, and density deviation can be read simultaneously when ROI processing is performed with the RMRUN command.

A value of –2 will be returned for the judgement result when a measurement feature that doesn't include judgement processing is selected.

Measurement feature 4 in the table above hasn't been completed yet.

A processing result of 0 and judgement result of –2 will be returned for blank items (---) in the table above.

The density lower limit (0 to 255) and density upper limit (0 to 255) set the density levels used in image clipping. The default setting for the density lower limit is 0 and the default setting for the density upper limit is 255. The previous settings will remain valid if these parameters are omitted.

The absolute value mode is used to reverse the above density level. The default setting is 0.

0: Do not reverse.
Other than 0: Reverse.

The gray correlation value for ROI processing is 0 to 100; Negative correlation values are set to 0.

RMMODE2 ROI Model MODE 2

(Command)

Action Determines the ROI model's extended processing mode.

Format RMMODE2 *model#*, *processing mode*, *array1* [, *array2*]

Description Specifies the extended operating mode when processing with the RMRUN or RMGRUN commands. The operating modes set with the RMMODE command are for ROI processing performed just one time at a fixed position. The RMMODE2 command specifies the operating mode for ROI processing that accompanies movement.

Specify the desired ROI model with the *model#* parameter (0 to 1023).

Specify the desired type of processing with the *processing mode* parameter, as shown in the following table.

Processing mode parameter setting		Storage of measurement results			
		Array1	Array2	Array3	Array4
0	Fixed position processing	Refer to the description of RMMODE.			
1	Precision search (no rotation)	Search position X coordinate and its judgement	Search position Y coordinate and its judgement	---	Search position correlation value and its judgement
2	Precision search (with rotation)	Search position X coordinate and its judgement	Search position Y coordinate and its judgement	Search angle and its judgement	Search position correlation value and its judgement
3	Processing in straight line region	X coordinate of defect position	Y coordinate of defect position	Estimated value of defect position	Judgement result
4	Processing in arc region	X coordinate of defect position	Y coordinate of defect position	Estimated value of defect position	Judgement result
5	Processing in circle region	X coordinate of defect position	Y coordinate of defect position	Estimated value of defect position	Judgement result
6	Processing in arbitrary region	X coordinate of defect position	Y coordinate of defect position	Estimated value of defect position	Judgement result
7	Number of straight line defects	X coordinate of defect position	Y coordinate of defect position	Estimated value of defect position	Judgement result
8	Number of arc defects	X coordinate of defect position	Y coordinate of defect position	Estimated value of defect position	Judgement result
9	Number of circle defects	X coordinate of defect position	Y coordinate of defect position	Estimated value of defect position	Judgement result
10	Number of arbitrary region defects	X coordinate of defect position	Y coordinate of defect position	Estimated value of defect position	Judgement result
11	Density level follow-up	(Same as the fixed position processing.)			

Processing mode parameter setting		Storage of measurement results			
		Array5	**Array6**	**Array7**	**Array8**
0	Fixed position processing	Refer to the description of RMMODE.			
1	Precision search (no rotation)	Search X before image scroll	Search Y before image scroll	---	---
2	Precision search (with rotation)	Search X before image scroll	Search Y before image scroll	Search angle before image scroll	---
3	Processing in straight line region	Defect X position before image scroll	Defect Y position before image scroll	---	---
4	Processing in arc region	Defect X position before image scroll	Defect Y position before image scroll	---	---
5	Processing in circle region	Defect X position before image scroll	Defect Y position before image scroll	---	---
6	Processing in arbitrary region	Defect X position before image scroll	Defect Y position before image scroll	---	---
7	Number of straight line defects	Defect X position before image scroll	Defect Y position before image scroll	Number of defects	Judgement result of the number of defects
8	Number of arc defects	Defect X position before image scroll	Defect Y position before image scroll	Number of defects	Judgement result of the number of defects
9	Number of circle defects	Defect X position before image scroll	Defect Y position before image scroll	Number of defects	Judgement result of the number of defects
10	Number of arbitrary region defects	Defect X position before image scroll	Defect Y position before image scroll	Number of defects	Judgement result of the number of defects
11	Density level follow-up	(Same as the fixed position processing.)			

Processing mode parameter setting		Storage of measurement results		
		Array9	**Array10**	**Array11**
0	Fixed position processing	Refer to the description of RMMODE.		
1	Precision search (no rotation)	---	---	---
2	Precision search (with rotation)	---	---	---
3	Processing in straight line region	---	---	---
4	Processing in arc region	---	---	---
5	Processing in circle region	---	---	---
6	Processing in arbitrary region	---	---	---
7	Number of straight line defects	---	---	---
8	Number of arc defects	---	---	---
9	Number of circle defects	---	---	---
10	Number of arbitrary region defects	---	---	---
11	Density level follow-up	Density lower limit after follow-up	Density upper limit after follow-up	Absolute value mode after follow-up

The following list provides details on the data stored in the arrays.

1, 2, 3... 1. Processing mode = 0 (fixed position processing)

The processing specified in RMMODE or RMGMODE is performed, so the data set in *array* is meaningless.

2. Processing mode = 1 (precision search, no rotation)

   A precise search in sub-pixel units is performed on the region around the reference point specified in the RMRUN or RMGRUN command. A rotational search is not performed.

   The sub-pixel unit's search position and correlation can be determined.

   The following parameters are specified in *array1*.

   Array1 (0) = Search region's X size
   Array1 (1) = Search region's Y size
   Array1 (2) = Frame/field mode specification
   (0=frame, other than 0=field)

   Array2 (0) = Correlation value criteria
   When –1 is specified, the judgement isn't carried out with an estimate value. (The default setting is –1.)

   The precision search finds the position that is most similar to the rectangular region in the specified ROI model centered on the reference point specified in RMRUN or RMGRUN with a height and width specified by the Y size and X size. If both the Y size and X size are 0, the precision search's reference point will be returned as the most similar point.

   Be sure to specify gray correlation as the measurement feature with the RMMODE command.

   When a "correlation value criteria" has been specified, processing will be stopped if the correlation value from the first ROI processing operation is less than the specified value. The initial coordinates will be returned as the search position if processing is stopped. This function is used to avoid the long processing times that occur when the search object isn't within the specified search region.

3. Processing mode = 2 (precision search, with rotation)

   A precise search in sub-pixel units is performed on the region around the reference point specified in the RMRUN or RMGRUN command. A sub-angle rotational search is also performed.

   The sub-pixel unit's search position/angle and correlation can be determined.

   The following parameters are specified in *array1*.

   Array1 (0) = Search region's X size
   Array1 (1) = Search region's Y size
   Array1 (2) = Start rotation angle (degrees)
   Array1 (3) = End rotation angle (degrees)
   Array1 (4) = Step angle (degrees)
   Array1 (5) = Frame/field mode specification
   (0=frame, other than 0=field)

   Array2 (0) = Correlation value criteria
   When –1 is specified, the judgement isn't carried out with an estimate value. (The default setting is –1.)

   The precision search finds the position that is most similar to the rectangular region in the specified ROI model centered on the reference point specified in RMRUN or RMGRUN with a height and width specified by the Y size and X size. If both the Y size and X size are 0, the precision search's reference point will be returned as the most similar point.

   Be sure to specify gray correlation as the measurement feature with the RMMODE command.

When a “correlation value criteria” has been specified, processing will be stopped if the correlation value from the first ROI processing operation is less than the specified value. The initial coordinates will be returned as the search position if processing is stopped. This function is used to avoid the long processing times that occur when the search object isn't within the specified search region.

4. Processing mode = 3 (straight line region inspection)
Performs a defect inspection within a linear region. Can also inspect for the presence of objects using the minimum and maximum densities in the area. The position of the most probable defect and the defect level can be determined.

   The following parameters are specified in *array1*. The data set in *array2* is meaningless.

   Array1 (0) = White/black flag
   (0=white, 1=black, 2=black and white)
   Array1 (1) = Inspection item
   0: Large defect
   1: Small defect
   2: Presence
   3: Large + small defect
   4: Large defect + presence
   5: Small defect + presence
   6: Large/small defect + presence
   Array1 (2) = Mask interval

   Light defects are detected on a dark background when the white/black flag is set to white. Conversely, dark defects are detected on a light background when the white/black flag is set to black. Both light and dark defects are detected when the white/black flag is set to “black and white.”

   The inspection item specifies the subject of the inspection. Processing may require some time if several items are selected.

   Be sure to specify gray features as the measurement feature with the RMMODE command.

5. Processing mode = 4 (arc region inspection)
Performs a defect inspection within an arc region. Can also inspect for the presence of objects using the minimum and maximum densities in the area. The position of the most probable defect and the defect level can be determined.

   The following parameters are specified in *array1*. The data set in *array2* is meaningless.

   Array1 (0) = White/black flag
   (0=white, 1=black, 2=black and white)
   Array1 (1) = Inspection item
   0: Large defect
   1: Small defect
   2: Presence
   3: Large + small defect
   4: Large defect + presence
   5: Small defect + presence
   6: Large/small defect + presence
   Array1 (2) = Mask interval

   Light defects are detected on a dark background when the white/black flag is set to white. Conversely, dark defects are detected on a light background when the white/black flag is set to black. Both light and dark defects are detected when the white/black flag is set to “black and white.”

   The inspection item specifies the subject of the inspection. Processing may require some time if several items are selected.

   Be sure to specify gray features as the measurement feature with the RMMODE command.

6. Processing mode = 5 (circle region inspection)
   Performs a defect inspection within a circle region. Can also inspect for the presence of objects using the minimum and maximum densities in the area. The position of the most probable defect and the defect level can be determined.

   The following parameters are specified in *array1*. The data set in *array2* is meaningless.

   Array1 (0) = White/black flag
   (0=white, 1=black, 2=black and white)
   Array1 (1) = Inspection item
   0: Large defect
   1: Small defect
   2: Presence
   3: Large + small defect
   4: Large defect + presence
   5: Small defect + presence
   6: Large/small defect + presence
   Array1 (2) = Mask interval

   Light defects are detected on a dark background when the white/black flag is set to white. Conversely, dark defects are detected on a light background when the white/black flag is set to black. Both light and dark defects are detected when the white/black flag is set to “black and white.”

   The inspection item specifies the subject of the inspection. Processing may require some time if several items are selected.

   Be sure to specify gray features as the measurement feature with the RMMODE command.

7. Processing mode = 6 (arbitrary region inspection)
   Performs a defect inspection within an arbitrary region. Can also inspect for the presence of objects using the minimum and maximum densities in the area. The position of the most probable defect and the defect level can be determined.

   The following parameters are specified in *array1*. The data set in *array2* is meaningless.

   Array1 (0) = White/black flag
   (0=white, 1=black, 2=black and white)
   Array1 (1) = Inspection item
   0: Large defect
   1: Small defect
   2: Presence
   3: Large + small defect
   4: Large defect + presence
   5: Small defect + presence
   6: Large/small defect + presence

   Light defects are detected on a dark background when the white/black flag is set to white. Conversely, dark defects are detected on a light background when the white/black flag is set to black. Both light and dark defects are detected when the white/black flag is set to “black and white.”

   The inspection item specifies the subject of the inspection. Processing may require some time if several items are selected.

   Be sure to specify gray features as the measurement feature with the RMMODE command.

8. Processing mode = 11 (density level follow-up measurement)
   Performs a measurement by automatically picking up the density level (clipping level). The measurement operation after density level follow-up is equivalent to the “0: fixed position processing” operation.

   The following parameters are specified in *array1*. The data set in *array2* is meaningless.

   Array1 (0) = Density level follow-up mode
   (0 = no follow-up, 1 = mode 1, 2 = mode 2)
   Array1 (1) = ROI model number (0 to 1,023) for density level follow-up

Array1 (2) = Average density of the model image

When –1 is specified, the average density of the image registered as the ROI model for follow-up will be set as the average density of the model image.

When –2 is specified, the previous setting is retained.

Array1 (3) = Density deviation of the model image

When –1 is specified, the density deviation of the image registered as the ROI model for follow-up will be set as the density deviation of the model image.

When –2 is specified, the previous setting is retained.

Array1 (4) = Correlation value criteria

When –1 is specified, the correlation value judgement is not performed.

The three modes for density level follow-up are described in the following table. The mode is set with array element Array1 (0).

Mode	Function
No follow-up	ROI processing is executed with the upper and lower limit density levels set with the RMMODE command – without performing a density level follow-up. The correlation value judgement described below is valid even when density level follow-up isn't performed.
Mode 1	The density level is determined from the average density and standard deviation. This mode is effective when a binary measurement is made in a region of a gray image, such as character inspection with the binary weighting correlation. The level after follow-up is determined with the following equation: $Th = S/Sr \times Thr + (Im - (S/Sr) \times Imr)$ Th: Density level after follow-up S: Standard deviation of the input image Im: Average density of the input image Sr: Standard deviation of the model image Imr: Average density of the model image Thr: Density level that was set.
Mode 2	Varies the density level according to changes in the average density. This mode is effective when a binary measurement is made in a region in which the density is fairly uniform, such as in inspections of dirty surfaces. The level after follow-up is determined with the following equation: $Th = Thr + (Im - Imr)$ Th: Density level after follow-up Im: Average density of the input image Imr: Average density of the model image Thr: Density level that was set.

The RMMODE and RMMODE2 settings for the ROI model for density level follow-up are meaningless during the density level follow-up processing. The processes required for the specified density level follow-up mode will be executed.

When the density level follow-up mode is set to 1, the specification for the model image's density deviation is meaningless.

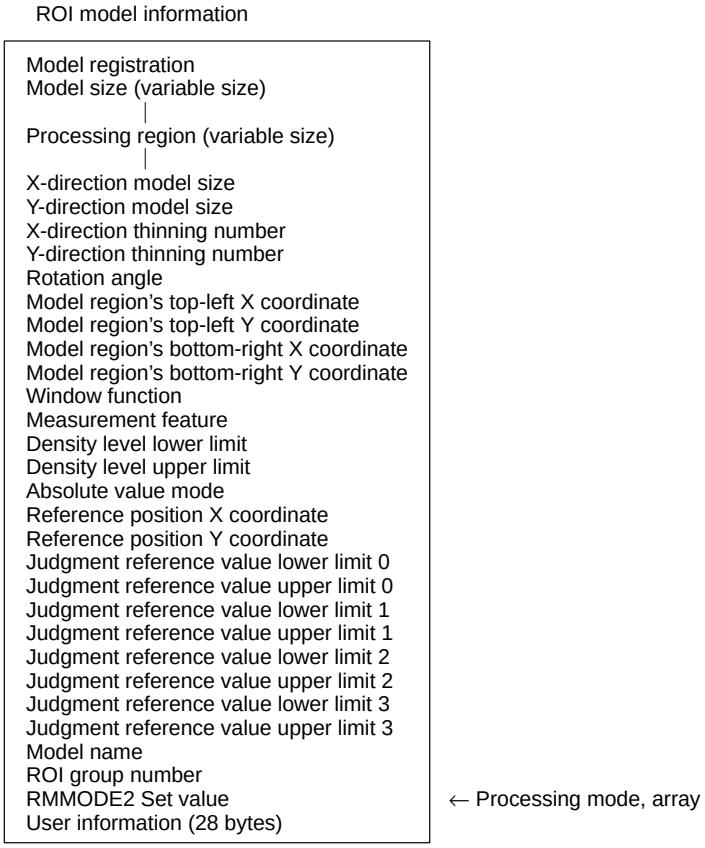
A meaningless value may be set if –2 is specified for the model image's average density array1 (3) or density deviation array1 (3). If –2 is specified, the previous settings will be preserved, making them meaningless.

If a "correlation value criteria" is specified, ROI processing after the follow-up won't be executed when the correlation value of the ROI model for follow-up is less than the judgement limit. In this case, the results for ROI processing after follow-up will all be "0" and the judgements will all be "NG".

The ROI processing position is determined by the various parameters as well as the position specified by the RMPUT2 command + the displacement position specified by the RMORIGIN command.

The processing results will be uncertain when ROI processing is performed at the specified position and the ROI model projects out from the display.

The measurement feature, density upper/lower limits, and absolute value mode set with the RMMODE command become invalid for ROI processing. Set the measurement feature to 1 (gray features) with the RMMODE command when performing defect inspection.



RMNAME ROI Model NAME

(Command)

Action Sets the ROI model's name.

Format RMNAME *model#*, *character string*

Description Sets the name specified in *character string* as the name of the specified ROI model. The name can be used to distinguish the ROI model from other models.

Specify the desired ROI model with the *model#* parameter (0 to 1023).

Specify the desired name with the *character string* parameter. The name can be up to 11 bytes long and can contain character codes 01 to FF (hexadecimal).

The name has absolutely no effect on ROI execution or ROI results. The name will be retained unchanged, even if the ROI model is registered.

(RMODE) (Not supported by the F350-C12E/C41E) Runlength MODE

(Command)

The RMODE command can't be used in the F350-C12E/C41E.

Action Sets the measurement mode for detailed run length data.

Format RMODE *binary image plane#, measured pixels* [, [*window function*] [, [*direction*] [, [*noise*] [, *image cutout*]]]]

Description The RMODE command sets the conditions for measurement of the detailed runlength with the MEASURE command. Specify the binary image plane for which the measurement conditions are set with the *binary image plane#* parameter.

Set the *measured pixels* parameter to set which pixels to measure, as follows.

0: Black pixels
1: White pixels

Set the *window function* to specify measurement in the window, as follows. The default setting is 1.

0: Measure the whole screen.
1: Measure in window only

Set the *direction* parameter to specify the measurement direction of the detailed run length data, as follows. The default setting is 0.

0: X direction
1: Y direction

Measurement must be made in the X direction before measuring in the Y direction.

Set the *noise* parameter to specify a number of pixels between 0 and 3 to ignore as noise. This number or less of consecutive pixels is not considered as run length data.

Set the *image cutout* to 1 to recognize all parts outside the window as part of the run.

The RMODE command must be executed before the MEASURE command.

RMORIGIN ROI Model ORIGINal point

(Command)

Action Sets the ROI model's processing position displacement.

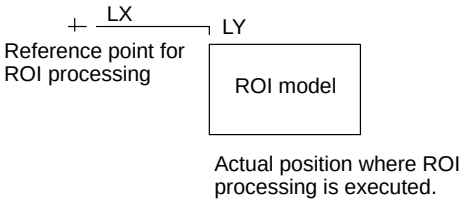
Format RMORIGIN *model#* [,*LX*,*LY*]

Description Sets the reference point during ROI registration for the specified ROI model. The displacement value specified with this command is added to the reference point specified in the RMRUN command, RMGRUN command, or sequential processing, and ROI processing is performed at the resulting position.

Specify the desired ROI model with the *model#* parameter (0 to 1023). The displacement value will be set for this ROI model.

Specify the desired displacement values in *LX* and *LY*. The relative coordinate from the reference point to the model's upper-left corner must be from –512 to 511. The default setting is 0.

A positive displacement is to the right for *LX*, up for *LY*. A negative displacement is to the left for *LX*, down for *LY*.



ROI model information

Model registration	
Model size (variable size)	
Processing region (variable size)	
X-direction model size	
Y-direction model size	
X-direction thinning number	
Y-direction thinning number	
Rotation angle	
Model region's top-left X coordinate	
Model region's top-left Y coordinate	
Model region's bottom-right X coordinate	
Model region's bottom-right Y coordinate	
Window function	
Measurement feature	
Density level lower limit	
Density level upper limit	
Absolute value mode	
Reference position X coordinate	← LX
Reference position Y coordinate	← LY
Judgment reference value lower limit 0	
Judgment reference value upper limit 0	
Judgment reference value lower limit 1	
Judgment reference value upper limit 1	
Judgment reference value lower limit 2	
Judgment reference value upper limit 2	
Judgment reference value lower limit 3	
Judgment reference value upper limit 3	
Model name	
ROI group number	
RMMODE2 Set value	
User information (28 bytes)	

RMPOSCNV ROI Model POSition CoNVert

(Command)

Action Specifies the conversion mode for the coordinates obtained during ROI processing.

Format RMPOSCNV *model#*, *conversion mode* [, *reference image memory*]

Description Specifies how the coordinates obtained through ROI processing will be converted and output.

Specify the desired ROI model number with the *model#* parameter (0 to 1023).

Specify the coordinate conversion mode (0 or 1) with the *conversion mode* parameter.

0: Do not perform coordinate conversion before image memory scroll.
1: Perform coordinate conversion before image memory scroll.

Specify the reference image memory page number (0 or 1) used in coordinate conversion with the *reference image memory* parameter. The default setting is 1.

The coordinates obtained during ROI processing depend on the processing mode specified in RMMODE2, as shown in the following table. These coordinate values can be read with commands such as RMDATA.

Value specified by RMMODE2's processing mode		Coordinates
0	Fixed position processing	Binary/gray center-of-gravity coordinates
1	Precision search (no rotation)	Search coordinates
2	Precision search (with rotation)	Search coordinates and angle
3	Processing in straight line region	Defect position coordinates
4	Processing in arc region	
5	Processing in circle region	
6	Processing in arbitrary region	
7	Density level follow-up processing	Binary/gray center-of-gravity coordinates

RMPUT ROI Model PUT

(Command)

Action Registers ROI model data.

Format RMPUT *model#*, [*page#*], *X1*, *Y1*, *X2*, *Y2* [, [*STPX*] [, [*STPY*] [, [*window function*] [, *window plane#*]]]

Description Registers the specified rectangular region in image memory as an ROI model image. Once the ROI model data has been registered, it is retained in the system even if the power is turned off.

The RMPUT command just registers the image data for ROI processing; the conditions for ROI processing are set with the RMMODE and RMMODE2 commands.

The ROI model data will be completely overwritten.

Specify the desired ROI model with the *model#* parameter (0 to 1023).

Specify the image memory page number (0 or 1) with the *page#* parameter. The default setting is 0.

The ROI model data must be input or drawn into the specified image memory page in advance.

Specify the upper-left and lower-right corners of the rectangular region with points *(X1*, *Y1)* and *(X2*, *Y2)*. The size of the rectangular region must be at least 1×1 pixels but not more than 512×484 pixels.

The *STPX* and *STPY* parameters specify the thinning numbers (1 to 15). When the thinning number is set to 2 or more, RMPUT will skip over that amount of image data and register the data. The processing time decreases as the thinning number is increased, but accuracy is reduced as well. The default setting is 1.

The *window function* is used to register an arbitrary region other than a rectangle; first draw the arbitrary region's shape in window memory and then register the region. The following settings can be made for the *window function* parameter. The default setting is 0.

0: Window function OFF
The entire rectangular region is used as the ROI region.

1: Window function ON (Registers the model image and the window.)
The window memory contents are registered as the arbitrary region at the same time that the image data is registered.

2: Window function ON (Registers the window only.)
Just the region data drawn in window memory is registered. Previously registered image data is retained.

3: Window function maintained (Registers the model image only.)
Retains the previously registered image data as is, and registers only the image data.

The *window plane#* parameter specifies the plane (0 to 7) in which the arbitrary region's shape has been drawn. The default setting is 7.

The display angle of the specified image memory page is automatically registered as the model's reference angle.

RMPUT2 ROI Model PUT 2

(Command)

Action Registers the ROI model for extended mode.

Format RMPUT2 *model#*, *mode*, [, [*array 1*] [, *plane#*]]

Description Registers the ROI model for ROI extended mode. Specify the desired ROI model with the *model#* parameter (0 to 1023).

The *mode* parameter specifies what kind of processing is to be performed in extended ROI processing. The mode specified here must agree with the processing mode set in the RMMODE2 command. The other parameter settings depend on the *mode* parameter setting, as described in the following list.

1, 2, 3... 1. Mode = 0

Registers the ROI model used for straight-line region defect inspection. Can also inspect for the presence of objects using the minimum and maximum densities in the area. The RMMODE2 command's processing mode should be set to 3 (straight line region inspection).

Array 1 (0) = The line's start-point X coordinate
Array 1 (1) = The line's start-point Y coordinate
Array 1 (2) = The line's end-point X coordinate
Array 1 (3) = The line's end-point Y coordinate
Array 1 (4) = Horizontal (parallel to the line) model width
Array 1 (5) = Vertical (perpendicular to the line) model width

2. Mode = 1

Registers the ROI model used for arc region defect inspection. Can also inspect for the presence of objects using the minimum and maximum densities in the area. The RMMODE2 command's processing mode should be set to 4 (arc region inspection).

Array 1 (0) = The circle's center-point X coordinate
Array 1 (1) = The circle's center-point Y coordinate
Array 1 (2) = The circles radius
Array 1 (3) = The arc's starting angle (degrees)
Array 1 (4) = The arc's ending angle (degrees)
Array 1 (5) = Horizontal (parallel to the arc) model width
Array 1 (6) = Vertical (perpendicular to the arc) model width

3. Mode = 2

Registers the ROI model used for circle region defect inspection. Can also inspect for the presence of objects using the minimum and maximum densities in the area. The RMMODE2 command's processing mode should be set to 5 (circle region inspection).

Array 1 (0) = The circle's center-point X coordinate
Array 1 (1) = The circle's center-point Y coordinate
Array 1 (2) = The circles radius
Array 1 (3) = Horizontal (parallel to the circle) model width
Array 1 (4) = Vertical (perpendicular to the circle) model width

4. Mode = 3

   Registers the ROI model used for arbitrary region defect inspection. Can also inspect for the presence of objects using the minimum and maximum densities in the area. The RMMODE2 command's processing mode should be set to 6 (arbitrary region inspection).

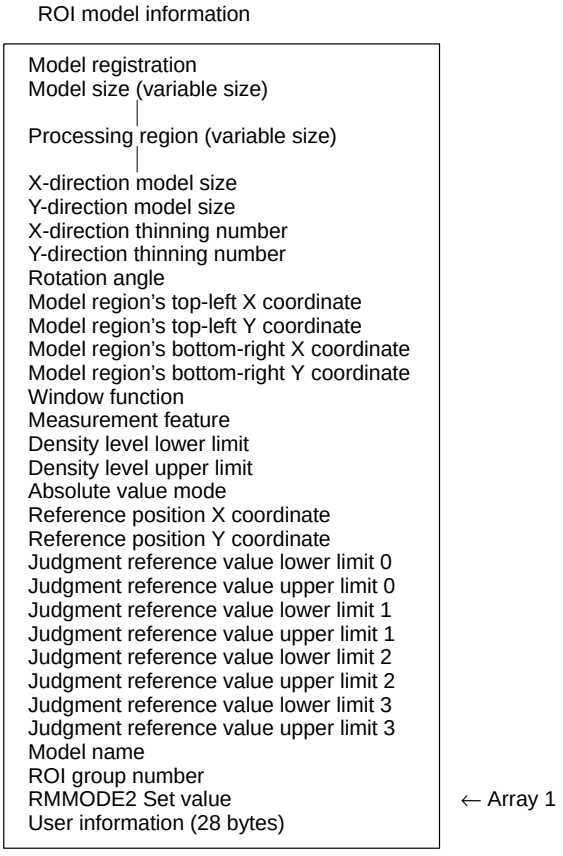
   Before executing RMPUT2, draw the arbitrary shape within the rectangular region specified by array elements array(0) through array(3) in the plane specified by the *plane#* parameter.

   Array 1 (0) = The region's start-point X coordinate
   Array 1 (1) = The region's start-point Y coordinate
   Array 1 (2) = The region's end-point X coordinate
   Array 1 (3) = The region's end-point Y coordinate
   Array 1 (4) = Horizontal (X-direction) model width
   Array 1 (5) = Vertical (Y-direction) model width
   Array 1 (6) = Mask interval

When the array elements are omitted, the existing settings aren't changed. Omit the array settings when you want to just register the model or change the reference position.

The *plane#* parameter specifies the window plane (0 to 7) used in internal processing when the ROI model is registered. The default setting is 7.

The specified window plane is used as a working plane and any patterns drawn in the plane before RMPUT2 is executed will be cleared.



RMPUT3 ROI Model PUT 3

(Command)

Action Creates a weighted image for measurement of the binary weight correlation and registers it as the ROI model.

Format RMPUT3 *model#*, *window plane #1*, *page#*, *X1*, *Y1*, *X2*, *Y2*, *pixel color*, *weighting saturation value*, *weighting saturation initial distance*, *disregard pixel number* [, [*number of connections*] [, [*STPX*], [*STPY*] [, *window function* [, *window plane #2*] [, *image memory mode*]]]]

Description Specify the desired ROI model number with the *model#* parameter (0 to 1023).

The *window plane #1* parameter specifies the plane (0 to 7) in which the binary image has been drawn. The binary image in the rectangular region of window plane #1 is used to create the weighted image for measurement of the binary weight correlation and registered as the ROI model.

Specify the image memory page number (0 or 1) that will create the weighted image to be registered as the model with the *page#* parameter. The weighted image will be drawn in the page specified with the *page#* parameter, so the specified page must be cleared in advance.

Specify the upper-left and lower-right corners of the rectangular region with points *(X1*, *Y1)* and *(X2*, *Y2)* between (0, 0) and (511, 483). The size of the rectangular region must be at least 1×1 pixels but not more than 512×484 pixels.

Specify the color of the pixels that are used in weighting the distance from the edge with the *pixel color* parameter.

0: Black pixels
1: White pixels
2: Black pixels and white pixels

Specify the *weighting saturation value* for the distance from the edge.

Specify the distance from the edge that causes saturation of the weighting with the *weighting saturation initial distance* parameter.

Specify the number of pixels from the edge where weighting is not performed (weighting = 0) with the *disregard pixel number* parameter. Be sure that the *weighting saturation initial distance* parameter is set to a higher value than the *disregard pixel number* parameter.

Specify the *number of connections* used when creating the distance conversion image as follows. (The default setting is 0.)

0: 8
Other than 0: 4

The *STPX* and *STPY* parameters specify the thinning numbers (0 to 15). When the thinning number is set to 2 or more, RMPUT will skip over that amount of image data and register the data. The processing time decreases as the thinning number is increased, but accuracy is reduced as well. The default setting is 1.

The *window function* is used to register an arbitrary region other than a rectangle; first draw the arbitrary region's shape in window memory and then register the region. The following settings can be made for the *window function* parameter. The default setting is 0.

0: Window function OFF (The entire rectangular region is used.)
1: Window function ON (Registers the model image and the window.)

The *window plane #2* parameter specifies the plane (0 to 7) in which the arbitrary region's shape has been drawn. The default setting is 7.

The *image memory mode* parameter specifies the mode.

0: Frame mode (512×512)
Other than 0: Field mode (512×256)

RMRUN ROI Model RUN

(Command)

Action Executes ROI processing for the specified ROI model and reads the results.

Format RMRUN *model#*, [*X*], [*Y*] [, [*array 1*] [*,* [*array 2*] [*,* [*array 3*] [*,* [*array 4*] [*,* [*array 5*] [*,* [*array 6*] [*,* [*array 7*] [*,* [*array 8*] [*,* [*array 9*] [*,* [*array 10*] [*, array 11*]]]]]]]]]]

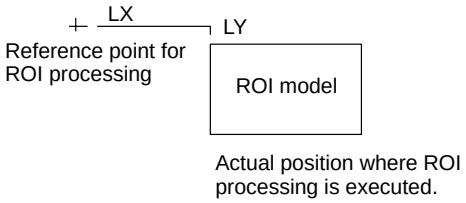
Description Executes ROI processing for the specified ROI model. Specify the desired ROI model with the *model#* parameter (0 to 1023).

The *X* and *Y* parameters specify the reference coordinates used in ROI processing; The point (*X*, *Y*) must be within the range (0, 0) through (511, 483).

When the *X* and *Y* parameters are omitted, the X and Y coordinates when the ROI model was registered are used as the reference position.

Specify the one-dimensional array variable names that will contain the results in *array 1* through *array 11*. Refer to the RMDATA command for details on the information stored in these arrays. Data won't be stored in arrays that aren't specified.

When specifying an displacement with the RMORIGIN command, the actual point where ROI processing is performed is determined by adding the displacement values to the reference coordinates.



RMSAVE ROI Model SAVE

(Command)

Action Saves the ROI model data to the memory card.

Format RMSAVE *file name*, *start model#* [[*, end model#*], *skip*]

Description Saves the data of the specified ROI models in the memory card. Specify the file in which the data will be saved with the *file name* parameter.

Specify the first desired ROI model with the *start model#* parameter (0 to 1023). Specify the last desired ROI model with the *end model#* parameter (0 to 1023). When the *end model#* parameter is omitted, just the ROI model specified by *start model#* will be saved.

The following values can be set for the *skip* parameter. The default setting is 0.

0: Save all of the specified ROI models.
Other than 0: Save only those specified models that have been registered.

Data is compressed before being saved in the file.

Data can't be saved to the RS-232C port.

RMTOSM ROI Model TO Search Model

(Command)

Action Generates a search model from an ROI model.

Format RMTOSM *ROI model#*, *search model#*, *X1*, *Y1*, *X2*, *Y2* [, *SX*, *SY*]

Description Registers a search model from an ROI model image. The search model can be registered without drawing an ROI model in image memory or inputting a new image in image memory.

Specify the desired ROI model with the *ROI model#* parameter (0 to 1023).

Specify the desired search model with the *search model#* parameter (0 to 435).

Specify the upper-left and lower-right corners of the search model region with points *(X1*, *Y1)* and *(X2*, *Y2)*; specify these points as displacements from the upper-left corner of the ROI model.

Parameters *X1*, *Y1*, *X2*, *Y2* must satisfy the following equations. Coordinates that satisfy these conditions can be obtained with the SMSELECT2 command.

X2 – *X1* + 1 = 22, 29, 36, 43, 50, 57, or 64
Y2 – *Y1* + 1 = 18, 26, 32, 40, 46, 54, or 60

The *SX* and *SY* parameters specify the coordinates to return as the search position as a displacement from the upper-left corner of model. The default setting is (0, 0).

The X size of the ROI model must be specified as 511 or smaller.

Search model model#	
Model registration	← 1 (registered)
Model image	
Upper-left X coordinate at registration	← Upper-left X coordinate of ROI model
Upper-left Y coordinate at registration	← Upper-left Y coordinate of ROI model
X-direction model size	← X size of ROI model
Y-direction model size	← Y size of ROI model
Model reference point X coordinate	← SX
Model reference point Y coordinate	← SY
Model reference angle	← Current image memory rotation angle
Y-direction thinning number	
Rotation angle	
Search region X1	
Search region X2	
Search region Y1	
Search region Y2	
Estimate feature	
Number of search items	
Estimate level	
Noise level	
Absolute value mode	
Search group number	
System model number	
Feature extraction window X width	
Feature extraction window Y height	
Feature extraction window X change	
Feature extraction window Y change	
Dispersion level	
Σ M	
Maximum storage start number	

RND RaNDom

(Function)

Action Generates a random number between 0 and 1.

Format RND [(*numeric expression*)]

Description Returns a random number between 0 and 1. The random number is generated with 1 as the random number seed until the seed is changed with the RANDOMIZE command.

The returned random number depends on the value of the specified *numeric expression*. The meaning of the specified *numeric expression* is described in the table below.

Value	Description
Negative value	Initializes the random number sequence.
0	Returns the previous random number in the current random number sequence.
Positive value or omitted	Returns the next random number.

RSET Right SET

(Command)

Action Writes right-justified character data to a variable area defined with the FIELD command.

Format RSET *character variable = character string*

Description Specify a character variable name defined with the FIELD command with the *character variable* parameter.

Specify a character constant or character variable as the *character string*.

Excess characters are lost from the left of the character string if the length of the specified *character string* exceeds the length of the *character variable* defined with the FIELD command. Conversely, if the length of the specified *character string* is less than the length of the *character variable*, the remaining positions are filled with blanks.

RTRIM$ Right TRIM $

(Function)

Action Deletes spaces to the right of a character string.

Format RTRIM$ (*character string*)

Description Returns the character string (1- or 2-byte characters) with the spaces removed from the right.

RUN RUN

(Command)

Action Runs a program.

Format Format 1: RUN [*line#*]
Format 2: *filename* [*, R*]

Description The command in Format 1 runs the program from the specified line#. The program runs from the first line if the line# is omitted.

The command in Format 2 loads the program with the specified filename from the memory card and runs if from the first line. If the R option is specified, files previously opened for I/O are kept open.

The RUN signal turns ON when the RUN command is executed.

RUNOUT RUN OUT

(Command)

Action Controls the RUN signal.

Format RUNOUT *switch*

Description RUNOUT turns the Power Supply Unit's RUN signal ON and OFF.

Specify the *switch* parameter as follows.

0: Turn OFF the RUN signal.
Other than 0: Turn ON the RUN signal.

SAVE SAVE

(Command)

Action Saves an OVL program from memory to the memory card or RS-232C port.

Format SAVE *filename* [*, P*]

Description The SAVE command saves a program in memory in ASCII format to the file specified with the filename in the memory card or RS-232C port. If the name of an existing file is specified as the *filename*, the old contents of the file are overwritten.

The file is coded if the P option is specified.

The contents of a file saved using the P option can't be viewed with the LOAD, LIST, or EDIT commands. An error will occur if the OPEN command is executed on a file saved using the P option.

A program can't be saved with the SAVE command when a password has been set with the PASSWD command.

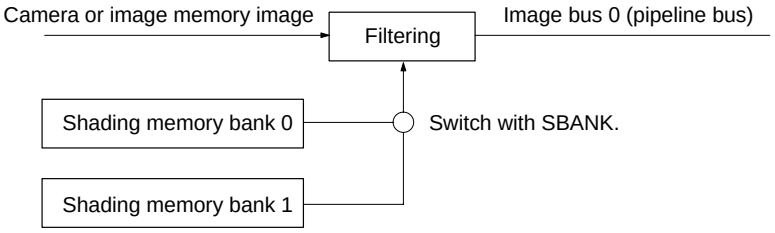
SBANK Shading memory BANK

(Command)

Action Selects the shading memory bank number.

Format SBANK (*bank#*)

Description The SBANK command switches to the shading memory bank 0 or 1 specified with the *bank#*. The shading memory can't be used in the F350-C12E/C41E.



SEARCH SEARCH

(Function)

Action Searches for an integer in an integer array and determines the number of the array element where the integer is found.

Format SEARCH (*array, value to find* [*,* [*start element#*] [*, increment*]])

Description The SEARCH function finds the specified value in the integer array and returns the number of the array element where the value is found as an integer. –1 is returned if the specified value is not found in the array.

Specify the *array* as the name of an array variable defined as a one-dimensional array with the DIM command.

Specify the integer to be found with the *value to find* parameter.

Specify the number of the array element to start the search with the *start element#* parameter. Specify any value between the minimum and maximum subscript value. The search starts from the start of the array variable if this parameter is omitted.

Specify the *increment* as a positive integer. The *increment* sets the counter between the searched array elements. The *increment* is added to the number of a searched element to determine the number of the next element to search. Intermediate elements are not searched. The default value is 1 if the parameter is omitted and all elements after the *start element#* are searched.

SEARCH2 SEARCH 2

(Function)

Action Searches for an element in an integer array that meets the given conditions and returns the number of that array element.

Format SEARCH2 (*array, lower limit, upper limit, condition*)

Description The SEARCH2 function returns the number of the array element in a one-dimensional array which has a value meeting the specified condition. –1 is returned if the a value meeting the specified condition is not found in the array.

Specify the name of an array variable with the *array* parameter. The array can contain integer, long integer, single-precision, or double-precision values.

Specify one of the following values for the *condition* parameter.

0: First value in the array within the lower and upper limits
1: Last value in the array within the lower and upper limits
2: Maximum value in the array within the lower and upper limits
3: Minimum value in the array within the lower and upper limits
4: Maximum positive difference
(The value of n where array(n) – array(n–1) is highest.)
5: Maximum negative difference
(The value of n where array(n) – array(n–1) is lowest.)

The search is performed from the smallest array number. The setting in the OPTION BASE command determines whether the search starts from element number 0 or element number 1.

SELECT...CASE–CASE ELSE–END SELECT
SELECT...CASE–CASE ELSE–END SELECT

(Command)

Action Provides multiple branching depending on the result of a conditional expression.

Format

SELECT *expression*
[CASE *item* [*, item...*]
Statement in CASE block
CASE ELSE
Statement in CASE ELSE block]
END SELECT

Description The SELECT command branches program execution depending on the value of the specified expression.

The *expression* can be defined as a numeric or character expression. The program branches as defined when the value of the expression matches the specified CASE value.

Multiple CASE statements may be defined. The CASE and CASE ELSE statement may be omitted.

The END SELECT statement is required; it must not be omitted.

If more than one CASE statement matches the result of the expression, the first of the CASE statements is executed.

Do not use the GOTO command to jump into or out of the SELECT block.

SET SET

(Command)

Action Sets the write-protect attribute for a file.

Format SET *filename* **or** *file#, "attribute character"*

Description Sets the write-protect attribute (write protect or write enable) with the *attribute character* for the file specified with the *filename* or *file#*. The attribute is applied to the specified file only. Other files in the memory card remain unchanged.

Specify the *filename* parameter as the name of an existing file with a character string. After the write-protect attribute is set for a file specified with a filename it remains unchanged until cancelled with the SET command.

For *file#*, specify the number with which the file was opened with the OPEN command. This attribute is maintained only while the file is open.

Specify the *attribute character* as either P or a null string or single space. Other characters cause an error.

"P": Set attribute.
"": Clear attribute.
" ": Clear attribute.

SETBLUT SET Binary LUT

(Command)

Action Sets array data as binary LUT data.

Format SETBLUT *binary image plane#, array name* [, [*subscript*] [, *size*]]

Description The SETBLUT command sets the array specified by the *array name* as the binary LUT data for the specified *binary image plane#*.

The *subscript* specifies the first array element to be set as the binary LUT data. The default value is 0.

The *size* parameter specifies the number of array elements, as follows. The default setting is 0.

0: 256
Other than 0: 512

The array data corresponds to the binary LUT data, as follows:

0: 0 set to the LUT
Other than 0: 1 set to the LUT

SETDLUT SET Display LUT

(Command)

Action Sets an array variable in the display LUT.

Format SETDLUT *region, array name,* [*, subscript*]

Description The SETDLUT command sets the values in the array variable specified by the *array name* as the display LUT data.

Separate display LUTs are provided for inside and outside the window. Specify the display LUT for inside or outside the window with the *region*, parameter. Enter 0 to set the display LUT for outside the window or 1 for inside the window.

The *subscript* specifies the first array element to be set as the display LUT. The default value is 0.

SETDLVL SET Display LeVeL

(Command)

Action Sets the type of display image outside of the window and the gradation outside of the image region.

Format SETDLVL *image type* [*,* [*gradation*] [*, page#*]]

Description The SETDLVL command sets the display image outside of the window and the display gradation (gray level) for the region outside of image memory.

Specify one of the following values for the *image type* parameter.

0: Character memory
1: Graphic memory
2: Mask image
3: Binary image, white
4: Binary image, black
5: Window memory increment when displaying a binary image
6: Paint/pattern matching window memory increment
7: Image memory

The *gradation* parameter sets the gradation (0 to 255) for the image specified with the *image type* parameter. If the *gradation* parameter is omitted, the initial setting will be used. The following table shows the initial settings for each image.

Image type	Initial gradation setting
Character memory	255
Graphic memory	192
Mask image	0
Binary image, white	12
Binary image, black	0
Window memory increment when displaying a binary image	48
Paint/pattern matching window memory increment	32
Image memory	0

When the *image type* is set to 5 (window memory increment) or 6 (paint/pattern matching window memory increment), the display gradation will be the sum of the background gradation and the gray level set in *gradation*.

Unless image memory is specified in the *image type*, SETDLVL sets only the display image's gradation, so there is absolutely no effect on measurement.

When the *image type* is set to 7 (image memory), SETDLVL sets the gradation outside of the image memory region when image memory is scrolled. When an image is scrolled and a measurement is made on image memory, that gradation is measured as the subject image.

SETLUT SET LUT

(Command)

Action Sets array data as the filter LUT data.

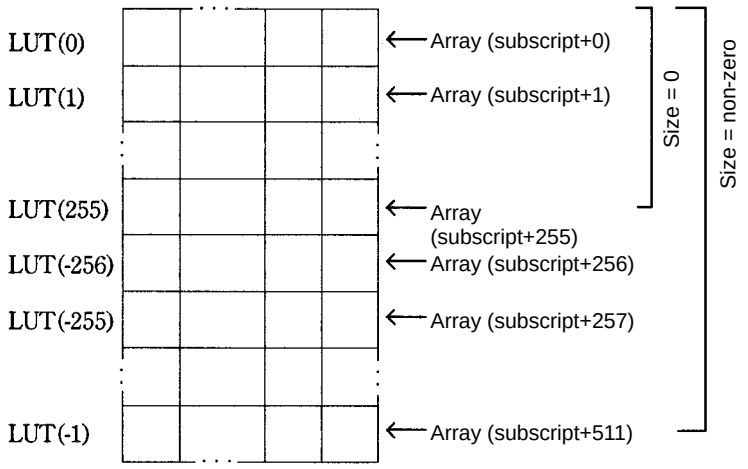
Format SETLUT *array name,* [*,* [*subscript*] [*, size*]]

Description The SETLUT command sets the array data variable specified by the *array name* as the filter LUT data.

The *subscript* specifies the first array element to be set as the filter LUT data. The default value is 0.

The *size* parameter specifies the number of array elements, as follows. The default setting is 0.

0: 256
Other than 0: 512



SETUINFO SET User INFOrmation

(Command)

Action Stores individual user information.

Format SETUINFO [*offset*]*, data* [, *data ...*]

Description Stores the specified information in the user information area allocated in backup memory.

The size of the entire user information area is 1 Kbyte and any kind of data can be stored in any address within this area. Data stored in the user information area has no effect whatsoever on OVL system processing.

Data stored in the user information area is retained when the power is interrupted and isn't affected by restarting the system or OVL, executing the NEW command, or loading an OVL program.

Use the *offset* parameter to specify an offset from the beginning of the user information area where SETUINFO will start writing the data. The setting range is 0 to 1,022 bytes and the default value is 0.

Input the data that you wish to store in the user information area with the *data* parameter(s). When a variable is specified for the *data* parameter, the number of bytes occupied by the data depends on the type of variable, as shown in the following table.

Variable type	Required number of bytes
Character type simple variable	Character length + 1
Integer type simple variable	2
Long integer simple variable	4
Single-precision simple variable	4
Double-precision simple variable	8

An array variable cannot be specified for a *data* parameter.

When two or more *data* parameters are specified, the data elements are stored in order beginning from the address indicated by the offset. An error will occur if the data exceeds the last address of the user information area.

Use the GETINFO command to read stored data. The structure of the data stored with SETINFO must be managed by the user. The data structure consists of offsets and data types.

SFTBLUT ShiFT Binary LUT

(Command)

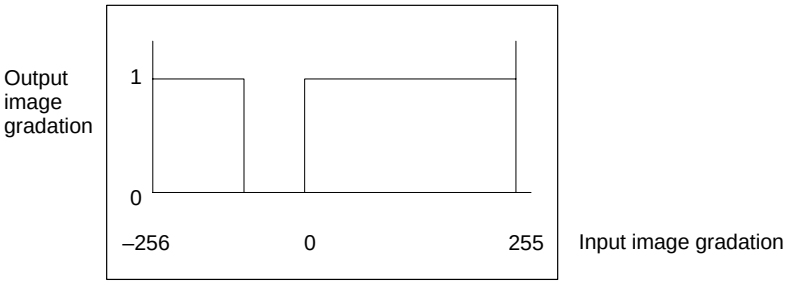
Action Shifts the binary LUT contents for each binary image plane.

Format SFTBLUT *binary image plane#, shift*

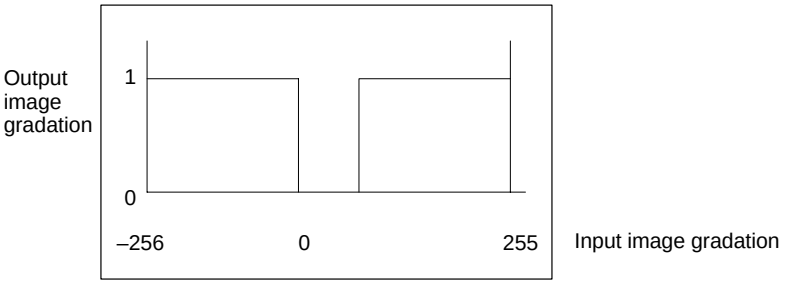
Description The SFTBLUT command shifts the contents of the binary LUT for the specified binary image plane# by the amount specified with the shift parameter.

The shift operation with positive and negative *shift* parameters is shown below.

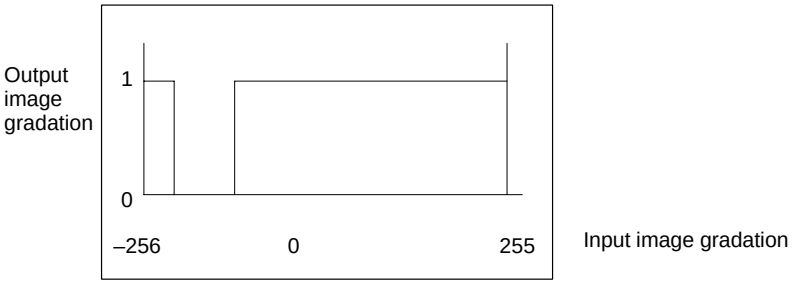
Before shifting:



Positive shift:



Negative shift:



SFTLUT ShiFT LUT

(Command)

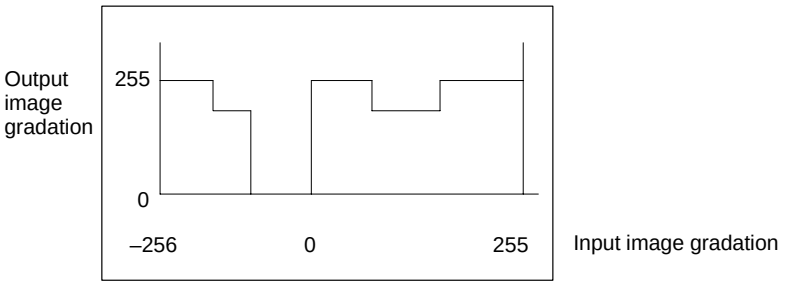
Action Shifts the filter LUT.

Format SFTLUT *shift*

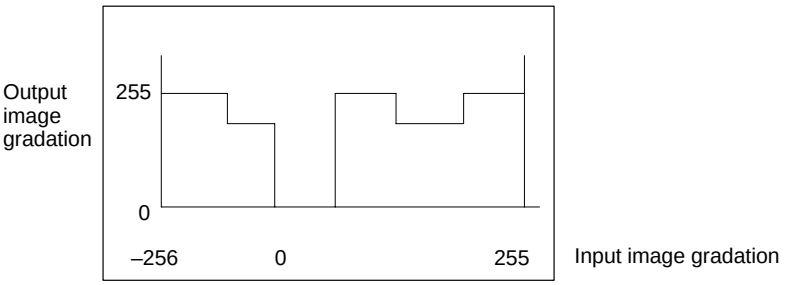
Description The SFTLUT command shifts the contents of the filter LUT by the amount specified with the shift parameter.

The shift operation with positive and negative *shift* parameters is shown below.

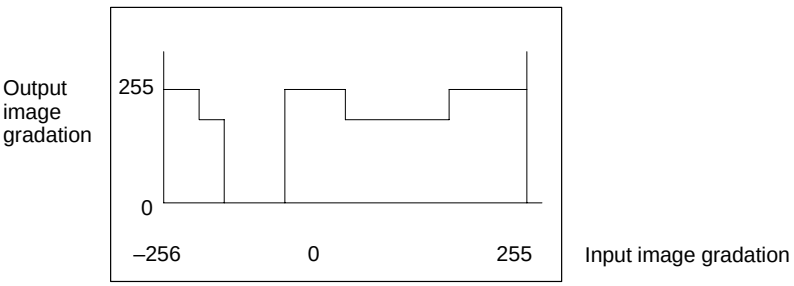
Before shifting:



Positive shift:



Negative shift:



SGN SiGN

(Function)

Action Determines the positive or negative sign of a numeric expression.

Format SGN (*numeric expression*)

Description The SGN function returns a value (–1, 0, 1) to indicate the sign of the numeric expression. The relationship between the sign of the *numeric expression* and returned value is shown below.

Numeric expression	Returned value
Positive	1
0	0
Negative	–1

SGTIME Sensor Gate TIME

(Command)

Action Sets the strobe flash interval (SG timing).

Format SGTIME *start timing* [, *end timing*]

Description The SGTIME command sets the SG timing, in which the strobe flash is disabled. SG timing is the interval when the CCD camera's image is transferred; a normal still image can't be taken if the strobe flashes or shutter operates in this interval.

The SGTIME command sets this interval and prevents a strobe flash or shutter operation during this interval.

The SG timing varies with different camera types and must be set correctly for each camera based on the camera's specifications.

The *start timing* and *end timing* parameters set the number of horizontal retrace lines from the vertical interval start position. The *start timing* parameter can be set from –20 to 10 and the *end timing* parameter can be set from –17 to 18. The default setting for the *start timing* parameter is +2.

The SGTIME command sets the same values for all cameras, so it isn't possible to make separate SG timing settings for different types of cameras that are connected.

When the F350 is started, the SG timing will be set according to the Camera Unit mounted in the leftmost slot (slot number 0), as shown in the following table.

Unit in slot 0	SG timing
F300-A20	F300-S
F300-A22S	F300-S
F300-A20R	PJLG-55-Z
F300-A22RS	PJLG-55-Z

The following table shows the SG timing for each camera type.

Camera type	Start timing	End timing	Notes
F300-S	6	8	Default for A20 and A22S
F200-S	10	17	---
PJLG-50-Z	–11	–9	---
PJLG-55-Z	–11	–9	Default for A20R and A22RS

For the F300-S2R/S3DR/S4R, set the SG timing according to the shutter speed.

The following table shows the strobe disable start and end times for the F300-S2R/S3DR/S4R.

Shutter speed	F300-S2R		F300-S3DR		F300-S4R	
	Start	End	Start	End	Start	End
1/1,000	–8	–6	–10	–8	–16	–14
1/1,500	---	---	---	---	–11	–9
1/2,000 (default)	–2	0	–2	0	–8	–6
1/3,000	---	---	---	---	–6	–4
1/4,000	–2	0	3	5	–5	–3
1/6,000	---	---	---	---	–3	–1
1/8,000	---	---	---	---	–3	–1
1/10,000	–2	0	5	7	–2	0
1/30,000	---	---	---	---	–2	0
1/50,000	---	---	---	---	–2	0

SIN SINe

(Function)

Action Determines the sine of a numeric expression.

Format SIN (*numeric expression*)

Description The SIN function returns a value between –1 and +1. The numeric expression must be set in radians. Convert an angle in degrees to radians by multiplying by π/180.

The *numeric expression* may be in the form of an integer, long integer, or a single or double-precision real number.

SLABEL Software LABELing

(Command)

Action Labels with software.

Format SLABEL *VRAM* [, [*page#*] [, [*binary image plane#*] [, [*measured pixels*] [, [*window function*] [, [*link evaluation constant*] [, *X1*, *Y1*, *X2*, *Y2*]]]]]]

Description Labels binary images or binary patterns drawn in image memory or window memory.

Specify the *VRAM* with the *VRAM* parameter, as follows:

2: Window memory
3: Image memory

Specify the image/window memory page number in the *page#* parameter. The *page#* can be set to 0 or 1 if *VRAM*=3 (image memory); it must be set to 0 if *VRAM*=2 (window memory). The default setting is 0.

The *binary image plane#* parameter specifies the binary image plane (0 to 7) that contains the binary image. The default setting is 7.

The *measured pixels* parameter specifies whether the binary image's white pixels or black pixels are to be labelled. The white part of the image is the part with bits ON, the black part is the part with bits OFF. The default setting is 1 (white pixels).

0: Black pixels
Other than 0: White pixels

The *window function* parameter specifies whether the entire screen is to be labelled, or just a particular region. When the window function is ON, the bits that are ON in the plane specified by the window memory's binary image plane number indicates the region that will be labelled; the image outside of this region won't be labelled. The default setting is 0 (OFF).

0: Window function OFF
Other than 0: Window function ON

Specify the *link evaluation constant* as follows. The default setting is 0.

0: 8-neighboring connections
1: 4-neighboring connections

Specify the upper-left and lower-right corners of the region with points *(X1*, *Y1)* and *(X2*, *Y2)*. The default settings are (0,0) and (511,483).

The following commands and functions are related to the labelling carried out with the SLABEL command:

LDATA, LNUM, LPOINT, LPUTIMG, LSORT, and RDATA

The LABEL command can't be used in the F350-C12E/C41E. The SLABEL command is provided to perform labelling in the F350-C12E/C41E.

The subject image must be input into image memory or window memory in advance in order to perform labelling with the SLABEL command.

SMAREA Search Model AREA

(Command)

Action Specifies the region to be searched for each search model.

Format SMAREA *model#* [, *X1*, *Y1*, *X2*, *Y2*]

Description Sets a region within the specified search model where the search is to be performed.

Specify the desired search model with the *model#* parameter (0 to 435).

Specify the upper-left and lower-right corners of the search region with points *(X1*, *Y1)* and *(X2*, *Y2)*. The default settings are (0,0) and (511,483).

When the specified search region projects from the display, it will be adjusted to fit within the display.

The search results will be unreliable if the search model is larger than the search region. The size of the search region doesn't affect the processing time.

The search region isn't overwritten even when the search model is registered again. Once a search region has been set it is valid until it is reset or initialized.

When setting the same search region for several search models, it may be easier to register the search models in a group and then use the SMGAREA command on the entire group.

Search model model#

Model registration	
Model image	
Upper-left X coordinate at registration	
Upper-left Y coordinate at registration	
X-direction model size	
Y-direction model size	
Model reference point X coordinate	
Model reference point Y coordinate	
Model reference angle	
Search region X1	← *X1*
Search region Y1	← *Y1*
Search region X2	← *X2*
Search region Y2	← *Y2*
Evaluation feature	
Number of search items	
Evaluation level	
Noise level	
Absolute value mode	
Model name	
Search group number	
System model number	
Feature extraction window X width	
Feature extraction window Y height	
Feature extraction window X change	
Feature extraction window Y change	
Dispersion level	
Σ M	
Maximum point storage start number	

SMCLEAR Search Model CLEAR

(Command)

Action Clears and initializes the specified search model.

Format SMCLEAR *start model#* [, *end model#*]

Description The SMCLEAR command initializes the search model's information from the specified *start model#* to the specified *end model#*. The start and end model numbers can be set from 0 to 435. If the end model number is omitted, just the start model number's search model will be initialized.

The search results aren't cleared when the SMCLEAR command is executed.

Search model information is retained even when the power is interrupted or the OVL program is cleared. Use the SMCLEAR command to initialize the search models when an entirely new OVL program is loaded or the model information has been corrupted.

Search model model#

Model registration	← 0 (not registered)
Model image	← 0
Upper-left X coordinate at registration	← 0
Upper-left Y coordinate at registration	← 0
X-direction model size	← 64
Y-direction model size	← 64
Model reference point X coordinate	← 0
Model reference point Y coordinate	← 0
Model reference angle	← 0
Search region X1	← 0
Search region Y1	← 0
Search region X2	← 511
Search region Y2	← 483
Evaluation feature	← 0 (gray correlation)
Number of search items	← 1
Evaluation level	← 70
Noise level	← 0
Absolute value mode	← 0
Model name	← Null
Search group number	← –1 (no assignment)
System model number	← model#
Feature extraction window X width	← 8
Feature extraction window Y height	← 8
Feature extraction window X change	← 0
Feature extraction window Y change	← 0
Dispersion level	← 0
Σ M	← 0
Maximum point storage start number	← 0

SMCOPY Search Model COPY

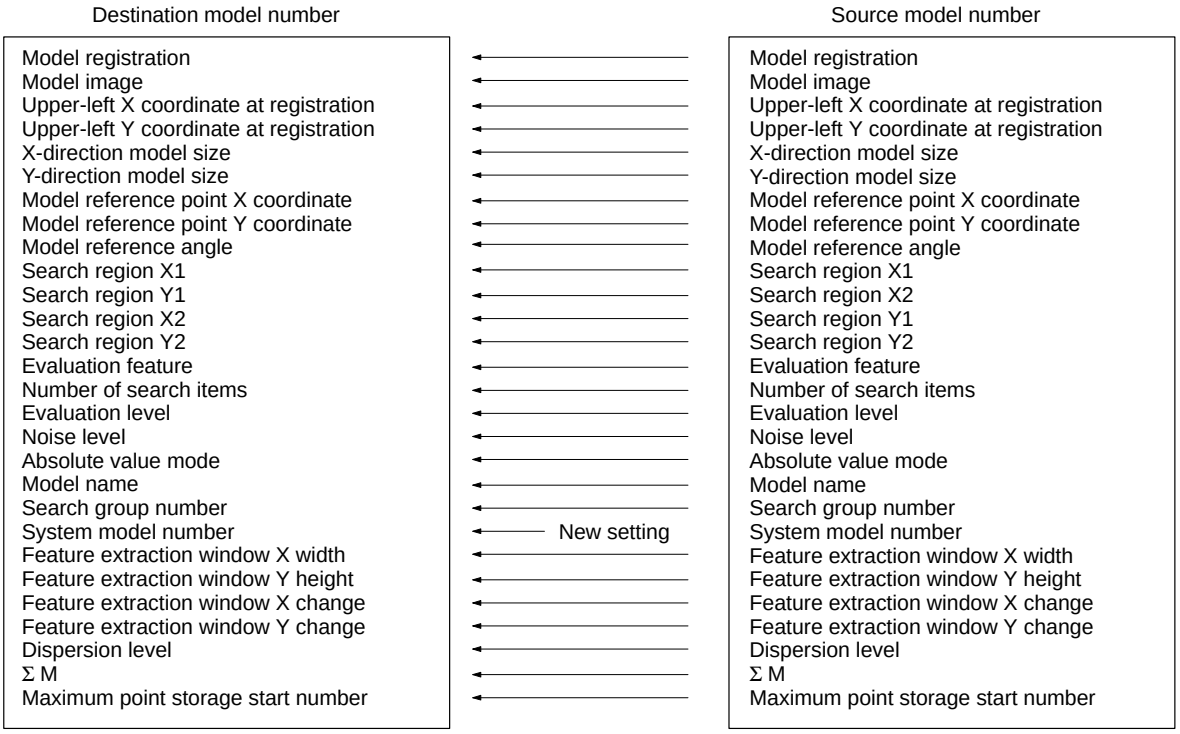
(Command)

Action Copies a search model's information.

Format SMCOPY *source model#*, *destination model#*

Description The SMCOPY command copies all of the search model information from the specified *source model#* to the specified *destination model#*. The source and destination model numbers can be set from 0 to 435.

This command is convenient when performing search processing on the same model image with different conditions.



SMDATA Search Model get DATA

(Function)

Action Reads the search results of a single search model.

Format SMDATA (*model#*, *data type* [*, data#*])

Description The SMDATA command reads the specified model's search results from execution of an SMRUN, SMGRUN, or SQRUN command. Specify the model number of the desired model in *model#*. The model number can be set from 0 to 435.

Specify the type of data to be read in the *data type* parameter.

0: Number of candidate points stored
1: Total number of candidate points
2: Candidate point X coordinate
4: Candidate point Y coordinate
6: Candidate point evaluation value
8: X coordinate of position with max. evaluation value
10: Y coordinate of position with max. evaluation value
12: Evaluation value of position with max. evaluation value
14: X coordinate of position with min. evaluation value
16: Y coordinate of position with min. evaluation value
18: Evaluation value of position with min. evaluation value
20: Error information

The *data#* parameter is used to specify the candidate point's number (0 to 127) when reading the results for a candidate point. The default setting is 0. When reading results unrelated to candidate points, any value can be specified or the *data#* parameter can be omitted altogether.

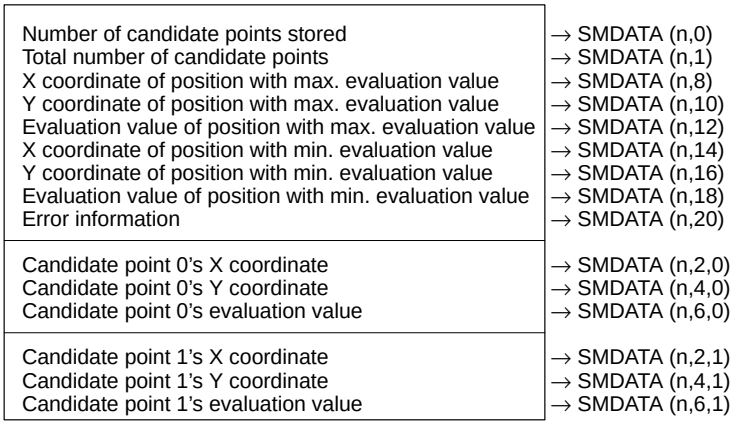
A value of –1 will be returned if the candidate point number specified in the *data#* parameter is greater than or equal to the "number of candidate points stored."

The candidate point numbers are determined according to the sort conditions in the SMRUN, SMGRUN, or SQRUN command.

The following table shows the relationship between the *data type*, *data#*, and returned value.

Data type		Data#	Returned value
0	Number of candidate points stored	Can be omitted.	0 to 127
1	Total number of candidate points		0 to 127
2	Candidate point X coordinate	Candidate point number (0 to 127)	0 to 511 (–1: no such candidate point)
4	Candidate point Y coordinate		0 to 483 (–1: no such candidate point)
6	Candidate point evaluation value		–100 to 100 for gray correlation 0 to 255 for average density level 0 to 127.5 for density dispersion
8	X coordinate of position with max. evaluation value	Can be omitted.	0 to 511
10	Y coordinate of position with max. evaluation value		0 to 483
12	Evaluation value of position with max. evaluation value		–100 to 100 for gray correlation 0 to 255 for average density level 0 to 127.5 for density dispersion
14	X coordinate of position with min. evaluation value		0 to 511
16	Y coordinate of position with min. evaluation value		0 to 483
18	Evaluation value of position with min. evaluation value		–100 to 100 for gray correlation 0 to 255 for average density level 0 to 127.5 for density dispersion
20	Error information		0: No error –1: Too many similar locations

The following diagram shows how to read the search results for search model n.



SMGAREA Search Model Group AREA

(Command)

Action Specifies the region to be searched in the search model group.

Format SMGAREA *group#* [, *X1*, *Y1*, *X2*, *Y2*]

Description Sets the same search region for all of the search models belonging to the specified search group.

Specify the desired search group with the *group#* parameter (0 to 511).

Specify the upper-left and lower-right corners of the search region with points *(X1*, *Y1)* and *(X2*, *Y2)*. The default settings are (0,0) and (511,483).

When the specified search region projects from the display, it will be adjusted to fit within the display.

The search results will be unreliable if the search model is larger than the search region. The size of the search region doesn't affect the processing time.

The search region isn't overwritten even when the search model is registered again. Once a search region has been set it is valid until it is reset or initialized.

Use the SMAREA command to set the search area for a single search models.

If no search models belong to the specified search group, the command will be completed normally without any action.

Search models belonging to the specified search group.

Model registration	
Model image	
Upper-left X coordinate at registration	
Upper-left Y coordinate at registration	
X-direction model size	
Y-direction model size	
Model reference point X coordinate	
Model reference point Y coordinate	
Model reference angle	
Search region X1	← *X1*
Search region Y1	← *Y1*
Search region X2	← *X2*
Search region Y2	← *Y2*
Evaluation feature	
Number of search items	
Evaluation level	
Noise level	
Absolute value mode	
Model name	
Search group number	
System model number	
Feature extraction window X width	
Feature extraction window Y height	
Feature extraction window X change	
Feature extraction window Y change	
Dispersion level	
Σ M	
Maximum point storage start number	

SMGCLEAR Search Model Group CLEAR

(Command)

Action Clears and initializes the specified search groups.

Format SMGCLEAR *start group#* [, *end group#*]

Description The SMGCLEAR command initializes the information of search models belonging to the groups from the specified *start group#* to the specified *end group#*. Specify the desired start and end groups with the *start group#* and *end group#*. These numbers can be set from 0 to 511. If the end group number is omitted, just the information for search models belonging to the start group will be initialized.

Search model information is retained even when the power is interrupted or the OVL program is cleared. Use the SMGCLEAR command to initialize the search models when an entirely new OVL program is loaded or the model information has been corrupted.

Search models between the start group# and end group#

Model registration	← 0 (not registered)
Model image	← 0
Upper-left X coordinate at registration	← 0
Upper-left Y coordinate at registration	← 0
X-direction model size	← 64
Y-direction model size	← 64
Model reference point X coordinate	← 0
Model reference point Y coordinate	← 0
Model reference angle	← 0
Search region X1	← 0
Search region Y1	← 0
Search region X2	← 511
Search region Y2	← 483
Evaluation feature	← 0 (gray correlation)
Number of search items	← 1
Evaluation level	← 70
Noise level	← 0
Absolute value mode	← 0
Model name	← Null
Search group number	← –1 (no assignment)
System model number	← model#
Feature extraction window X width	← 8
Feature extraction window Y height	← 8
Feature extraction window X change	← 0
Feature extraction window Y change	← 0
Dispersion level	← 0
Σ M	← 0
Maximum point storage start number	← 0

SMGDATA Search Model get Group DATA

(Command)

Action Reads the search results of the search models registered in a search group.

Format SMGDATA *group#*, [*data type*], [*array 1*] [, [*array 2*] [, [*array 3*] [, [*array 4*]]]

Description Reads the search results from execution of an SMGRUN or SQRUN command for all of the search models belonging to the specified search group. The data read by the SMGDATA command is the sorted search data for all search models in the group, so SMGDATA can't be used to read the results for a particular search model.

Specify the desired search group (0 to 511) with the *group#* parameter.

Specify the type of data to be read in the *data type* parameter. The default setting is 0.

0: Data on the position of candidate points
2: Data on the minimum and maximum evaluation values

Specify the one-dimensional arrays that will contain the results in *array 1* through *array 4*. These arrays must be defined in advance with the DIM command.

The data stored in the arrays is determined by the *data type* parameter, as shown in the following table.

Data type		Array1	Array2	Array3	Array4
0	Candidate point data	Candidate point's X coordinate	Candidate point's Y coordinate	Candidate point's evaluation value	Candidate point's model number
2	Evaluation value data	Positions with the min. and max. evaluation values	---	---	---

Data is stored as shown below when the *data type* parameter is set to 0. The "n" is the candidate point number determined by the search group's sort conditions.

Array1(n) = X coordinate of the n^{th} candidate point
Array2(n) = Y coordinate of the n^{th} candidate point
Array3(n) = Evaluation value of the n^{th} candidate point
Array4(n) = Search model number of the n^{th} candidate point

The number of elements in each array must be sufficient to store data for all of the candidate points in all of the search models belonging to the search group.

Data is stored in array1 as shown below when the *data type* parameter is set to 2. No data is stored in arrays 2, 3, and 4. The content of these arrays won't be changed if they are specified.

Array1(0) = Total of all candidate points stored in all models in the group
Array1(1) = X coordinate of position with max. evaluation value in the group
Array1(2) = Y coordinate of position with max. evaluation value in the group
Array1(3) = Max. evaluation value in the group
Array1(4) = Model number with max. evaluation value in the group
Array1(5) = X coordinate of position with min. evaluation value in the group
Array1(6) = Y coordinate of position with min. evaluation value in the group
Array1(7) = Min. evaluation value in the group
Array1(8) = Model number with min. evaluation value in the group

If an array contains more elements than needed, the contents of the extra array elements won't be changed.

Array1, array2, and array3 must have the same format.

SMGDEL Search Model Group DELete

(Command)

Action Deletes a search model from its search group.

Format SMGDEL *model#*

Description The SMGDEL command deletes the specified search model from the search group in which it is registered. Specify the desired search model with *model#* (0 to 435). The command can be executed even if the specified search model isn't registered in a search group.

SMGINFO Search Model Group INFOrmation

(Function)

Action Reads search group information for the specified search group.

Format SMGINFO (*group#*, *data type* [, *position*])

Description Reads the information for the search group specified in *group#*. Specify the desired search group with the *group#* parameter (0 to 511).

Specify the type of data to be read in the *data type* parameter.

0: number of models (0 to 436) registered in the group
1: the model number of the model in the specified *position* (0 to 435)

If the *data type* parameter is set to 1, The *position* indicates the location of the search model among the models in the group. The search models are listed in ascending order (0 to 435). The first search model in the group is at position 0, the second is at position 1, and so on. The default setting for the *position* is 0.

A value of –1 will be returned if the specified position is greater than or equal to the number of search models in the group.

Information for group m

Number of models in group# m (n)	→ SMGINFO(m,0)
Search model number in position 0	→ SMGINFO(m,1,0), SMGINFO(m,1)
Search model number in position 1	→ SMGINFO(m,1,)
: :	
: :	
: :	
Search model number in position n–1	→ SMGINFO(m,1,n–1)
	–1 → SMGINFO(m,1,n)

SMGLOAD Search Model Group LOAD

(Command)

Action Retrieves the search model data from the memory card and registers the data in the system.

Format SMGLOAD *file name* [, *group#*] [, *operating mode*]

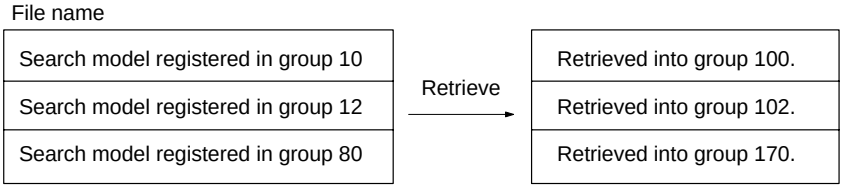
Description Retrieves the search model data specified by the *file name* parameter and registers the data in the system's search model memory. The data can't be retrieved unless the file was saved with the SMGSAVE or SMSAVE command.

Specify the destination search group (0 to 511), where the search model is loaded with the *group#* parameter. Since the search model can be registered in only one group, if loaded to another group, the search model will be deleted from the group that was saved. If the *group#* is omitted, the data will be retrieved into the same group number as the one that was saved. Normally, the group number is omitted and the same group number is used for saving and loading.

The following equation is used to determine the group in which the model data will belong.

Group = (group number when saved)+(*group#*–group number that the first model belonged to when the file was created.)

A group# of 100 was specified in the following example.



Specify the model number of the load destination for the *processing mode* parameter.

0: Registers into the unregistered model numbers. Be sure that there is enough space available for the search models being loaded.

1: Overwrites the model numbers when saved.

The default setting is 0.

If the group into which the search model is to be retrieved hasn't been set, the group still won't be set after the search model is retrieved.

Even if an error occurs while the search model is being retrieved, the search model data retrieved up to that point will be valid.

SMGMODE Search Model Group MODE

(Command)

Action Sets the search mode and conditions for all of the search models in the group.

Format SMGMODE *group#*, [*evaluation feature*] [, [*search items*] [, [*evaluation level*] [, *absolute value mode*]]]

Description Sets the search mode and the conditions (used when search processing is executed) for all of the search models belonging to the specified group. The SMMODE command makes the same settings for a single search model.

Specify the desired search group with the *group#* parameter (0 to 511).

The *evaluation feature* parameter specifies what kind of search processing is to be performed. If this parameter is omitted, the previous setting will remain effective.

0: Gray correlation
1: Average density level
2: Density deviation level

The *search items* parameter specifies a target for the maximum number of positions (candidate points) to be searched. The *search items* parameter can be set from 0 to 128. When 0 is specified, only information on the evaluation feature's maximum position and minimum position can be read.

The *evaluation level* parameter specifies the evaluation level for the specified feature, as shown in the following table.

Evaluation feature		Evaluation level range	Default setting
0	Gray correlation	0 to 100	70.0
1	Average density level	0 to 255	128.0
2	Density deviation level	0 to 127.5	64.0

Only positions with an evaluation value equal to or greater than the *evaluation level* setting will be stored as search results, so the detection standards become stricter as the *evaluation level* is increased.

When the *evaluation level* parameter is omitted, the default setting shown in the table above will be used. When both the *evaluation level* and *evaluation level* parameters are omitted, the previous settings for these parameters will remain in effect.

The *absolute value mode* specifies whether or not to to reverse the image gray level before performing the search. The default setting is 0.

0: Do not reverse.
Other than 0: Reverse.

The *absolute value mode* is valid only when the evaluation feature is set to 0 (gray correlation).

The previous setting will be used if the other parameters are omitted.

These settings won't be changed when the search model is registered or the search region is changed.

Search models belonging to group#

Model registration	
Model image	
Upper-left X coordinate at registration	
Upper-left Y coordinate at registration	
X-direction model size	
Y-direction model size	
Model reference point X coordinate	
Model reference point Y coordinate	
Model reference angle	
Search region X1	
Search region Y1	
Search region X2	
Search region Y2	
Evaluation feature	← *evaluation feature*
Number of search items	← *search items*
Evaluation level	← *evaluation level*
Noise level	
Absolute value mode	← *absolute value mode*
Model name	
Search group number	
System model number	
Feature extraction window X width	
Feature extraction window Y height	
Feature extraction window X change	
Feature extraction window Y change	
Dispersion level	
Σ M	
Maximum point storage start number	

SMGROUP Search Model GROUP set

(Command)

Action Registers a single search model in a search group.

Format SMGROUP *group#*, *model#*

Description Registers the specified search model in a search group. Specify the desired search group with the *group#* (0 to 511) and the desired search model with *model#* (0 to 435).

There is no limit on the number of search models that can be registered in a single search group, but a single search model can't be registered in more than one group. If the specified model number is already registered in another group, the model will be deleted from that group and registered again in the specified group.

SMGRUN Search Model Group RUN

(Command)

Action Executes search processing for the specified search group.

Format SMGRUN *start group#* [, [*end group#*] [, [*processing mode*] [, [*page#*] [, [*input path*] [, [*input mode*] [, *trigger mode*]]]]]]

Description Executes search processing for all of the search models in the specified groups. Specify the desired start and end groups with the *start group#* and *end group#*. These numbers can be set from 0 to 511. If the end group number is omitted, search processing will be performed for search models belonging to the start group only.

The *processing mode* parameter specifies how the search data will be sorted after the search is completed, as shown below. The default setting is 1.

Mode	Sorting process
0	No sorting (order of appearance)
1	Sort each model by evaluation value.
2	Sort each model by X search position (descending order).
3	Sort each model by X search position (ascending order).
4	Sort each model by Y search position (descending order).
5	Sort each model by Y search position (ascending order).
6	Sort all candidate points in the group by evaluation value.
7	Sort all candidate points in the group by X search position (desc. order).
8	Sort all candidate points in the group by X search position (asc. order).
9	Sort all candidate points in the group by Y search position (desc. order).
10	Sort all candidate points in the group by Y search position (asc. order).
11	Sort all candidate points in the group vertically from the upper-left.
12	Sort all candidate points in the group vertically from the bottom-right.
13	Sort all candidate points in the group horizontally from the upper-left.
14	Sort all candidate points in the group horizontally from the bottom-right.

Read the search results with the SMGDATA command.

The conditions for each search model must be set in advance with the SMMODE or SMGMODE command.

Search processing is performed on the gray image in image bus 0 (pipeline bus), so the settings must be made so that binary conversion isn't performed with LUT.

When the *page#* and later parameters are specified, search processing is performed as the image is input into image memory.

The *page#* parameter specifies the page number of the image memory page where the image will be input. There is no image input when this parameter is omitted.

0: Input image memory 0 only.
1: Input image memory 1 only.
–1: Input to all of image memory.

The *input path* parameter specifies which bus will be used to input the image. The default setting is 0.

0: Image bus 1 (video bus, gray images)
1: Image bus 0 (pipeline bus, mainly filtered images)

The *input mode* parameter specifies whether the measurement is performed in frame mode or field mode. The mode specified with the FMODE command is used when this parameter is omitted. The mode is set to frame mode when OVL is first started.

The following table shows the differences between frame mode and field mode.

Mode		Resolution	Image input time (excluding wait time)
0	Frame mode	512×512 pixels	33.3 ms
1	Field mode	512×256 pixels	16.7 ms

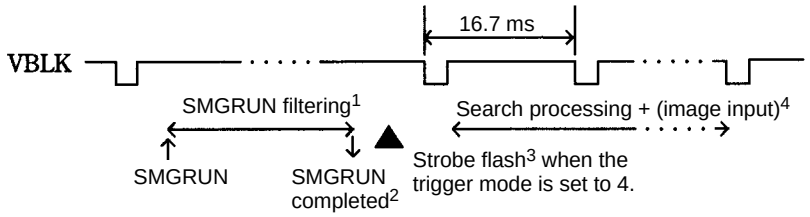
The *trigger mode* parameter determines when the image is input, as shown in the following table.

Trigger mode		Image input timing
0	Stop	Cancels the STEP synchronization setting.
1	No strobe flash	A single image is input when MEASURE is executed.

Trigger mode		Image input timing
2	STEP synchronization	A strobe flash is emitted in sync with the STEP signal and the image is input to image memory at that time.
3	Reserved	Not used.
4	Single strobe flash	A strobe flash is emitted and a single image is input when MEASURE is executed.

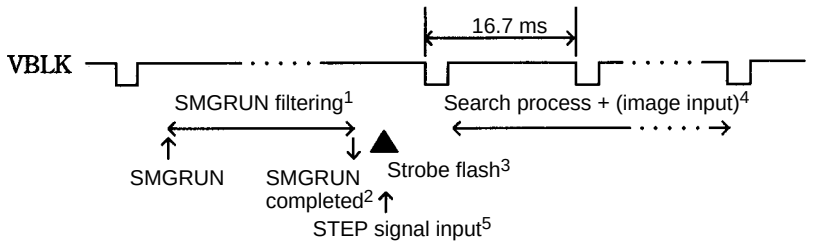
When the trigger mode is set to “2: STEP synchronization,” search processing + (image input) is performed automatically when the STEP signal is input. The STEP synchronization setting is cleared once the search has been performed with STEP synchronization.

The following diagram shows the timing of operation when the trigger mode is set to “1: No strobe flash” or “4: Single strobe flash.”



Note
1. Several ms are required for the SMGRUN command’s filtering.
2. The SMGRUN command ends when it’s filtering is completed.
3. When the SMGRUN command is completed, a strobe flash is emitted and the search process + (image input) is started.
4. The time required for search process + image input depends on the input mode and the number of models being searched.

The following diagram shows the timing of operation when the trigger mode is set to “2: STEP synchronization.”



Note
1. Several ms are required for the SMGRUN command’s filtering.
2. The SMGRUN command ends when it’s filtering is completed.
3. When the STEP signal is input, a strobe flash is emitted and the search process + (image input) is started.
4. The time required for search process + image input depends on the input mode and the number of models being searched.
5. The STEP signal can be input after the SMGRUN command has been completed.

Before executing the SMGRUN command, the search models must be sorted in group number order by the SMSORT command.

SMGSAVE Search Model Group SAVE

(Command)

Action Saves the search model data belonging to the specified search group to the memory card.

Format SMGSAVE *file name*, *group#* [*FOR APPEND*]

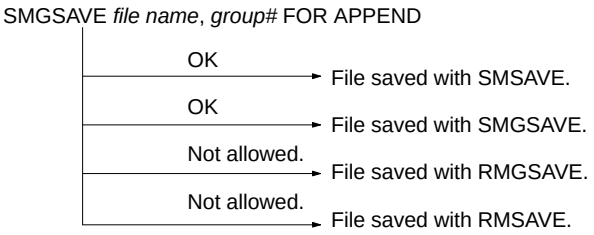
Description Saves the data of the search models belonging to the specified search group in the memory card. Specify the file in which the data will be saved with the *file name* parameter.

Specify the desired search group (0 to 511) with the *group#* parameter.

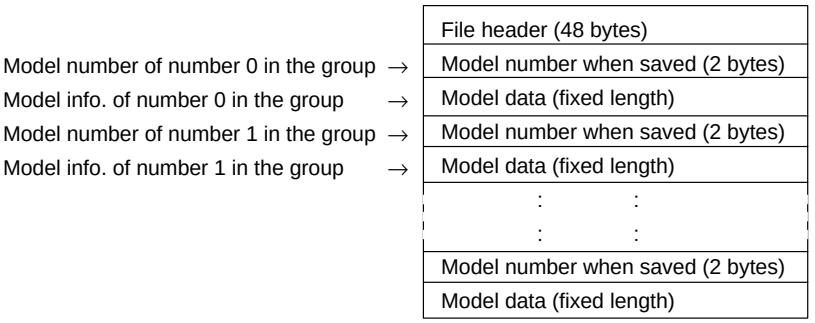
Data can't be saved to the RS-232C port.

When the "FOR APPEND" argument is specified, data will be appended to the specified file. Use this argument when saving several search groups in a single file.

Data can be appended to files saved with the SMSAVE and SMGSAVE commands, but not to files saved with the RMGSAVE and RMSAVE commands.



The following diagram shows the structure of files saved with SMGSAVE.



When appending data, model numbers and model data will be appended to the existing file.

SMINFO Search Model INFOrmation

(Function)

Action Reads search model information.

Format SMINFO (*model#*, *data type*)

Description Specify the desired search model with the *model#* parameter (0 to 435).

Specify the type of data to be read in the *data type* parameter.

Data type		Returned value
0	Model registration	0: Not registered 1: Registered.
1	Upper-left X coordinate at registration	0 to 489
2	Upper-left Y coordinate at registration	0 to 465
3	X-direction model size	23 to 71

Data type		Returned value
4	Y-direction model size	19 to 67
5	Model reference point X coordinate	0 to 511
6	Model reference point Y coordinate	0 to 483
7	Model reference angle	–180° to 180°
8	Model region's upper-left X coordinate	0 to 511
9	Model region's upper-left Y coordinate	0 to 483
10	Model region's lower-right X coordinate	0 to 511
11	Model region's lower-right Y coordinate	0 to 483
12	Evaluation feature	0: Gray correlation 1: Average density level 2: Density deviation level
13	Number of search items	0 to 128
14	Evaluation feature	0 to 100 (gray correlation) 0 to 255 (average density level) 0 to 127.5 (density level dev.)
15	Noise level[1]	---
16	Absolute value mode	0: Normal –1: Reverse mode
17	Search group number	0 to 511 (–1: Unregistered)
18	System model number	0 to 435
19	Feature extraction window X width[1]	---
20	Feature extraction window Y height[1]	---
21	Feature extraction window X change[1]	---
22	Feature extraction window Y change[1]	---
23	Dispersion level[1]	---
24	Σ M[1]	---
25	Maximum point storage start number	0
26	Number of registered search models	0 to 436
27	Lowest unregistered search model number	0 to 435
28	Number of search models that can be registered.	0 to 436
100	Error information	See note 2.

Note 1. Calculated internally when the model is registered.

2. Error information is as follows:
   –2: An error occurred but initialization was performed.
   –1: No error
   0 to 435: First model number where an error was detected.

The following diagram shows how to use the SMINFO command to read search model information.

Search model n (model#=n)

Model registration	← SMINFO (n,0)
Model image	
Upper-left X coordinate at registration	← SMINFO (n,1)
Upper-left Y coordinate at registration	← SMINFO (n,2)
X-direction model size	← SMINFO (n,3)
Y-direction model size	← SMINFO (n,4)
Model reference point X coordinate	← SMINFO (n,5)
Model reference point Y coordinate	← SMINFO (n,6)
Model reference angle	← SMINFO (n,7)
Search region X1	← SMINFO (n,8)
Search region Y1	← SMINFO (n,9)
Search region X2	← SMINFO (n,10)
Search region Y2	← SMINFO (n,11)
Evaluation feature	← SMINFO (n,12)
Number of search items	← SMINFO (n,13)
Evaluation level	← SMINFO (n,14)
Noise level	← SMINFO (n,15)
Absolute value mode	← SMINFO (n,16)
Model name	
Search group number	← SMINFO (n,17)
System model number	← SMINFO (n,18)
Feature extraction window X width	← SMINFO (n,19)
Feature extraction window Y height	← SMINFO (n,20)
Feature extraction window X change	← SMINFO (n,21)
Feature extraction window Y change	← SMINFO (n,22)
Dispersion level	← SMINFO (n,23)
Σ M	← SMINFO (n,24)
Maximum point storage start number	← SMINFO (n,25)

SMINFO$ Search Model INFOrmation $

(Function)

Action Reads the specified search model's model name.

Format SMINFO$ (*model#* [, *data type*])

Description Reads the specified search model's model name. Specify the desired search model with the *model#* parameter (0 to 435).

Specify 0 for the *data type* parameter or omit it.

Search model n

Model registration	
Model image	
Upper-left X coordinate at registration	
Upper-left Y coordinate at registration	
X-direction model size	
Y-direction model size	
Model reference point X coordinate	
Model reference point Y coordinate	
Model reference angle	
Search region X1	
Search region Y1	
Search region X2	
Search region Y2	
Evaluation feature	
Number of search items	
Evaluation level	
Noise level	
Absolute value mode	
Model name	← SMINFO$ (n)
Search group number	
System model number	
Feature extraction window X width	
Feature extraction window Y height	
Feature extraction window X change	
Feature extraction window Y change	
Dispersion level	
Σ M	
Maximum point storage start number	

SMLOAD Search Model LOAD

(Command)

Action Retrieves the search model data from the memory card.

Format SMLOAD *file name*

Description Retrieves the search model data specified by the *file name* parameter and stores the data in memory. The data will be retrieved into the same model number that was used when it was saved.

SMMDATA Search Model get Multi DATA

(Command)

Action Reads a batch of a search model's search results.

Format SMMDATA *model#*, [*data type*], [*array 1*] [*,* [*array 2*] [*, array 3*]]

Description The SMMDATA command reads the search result executed with the SMRUN, SMGRUN, or SQRUN command. Specify the desired search model with the *model#* parameter (0 to 435).

Set the *data type* parameter to 0 or omit it. The default setting is 0.

Specify the one-dimensional arrays that will contain the results in *array 1* through *array 3*. These arrays must be defined in advance with the DIM command. The three arrays contain the following data; the number of elements in the arrays equals the number of candidate points (n).

Array1 (n): Contains the n^{th} candidate point's X-coordinate value.
Array2 (n): Contains the n^{th} candidate point's Y-coordinate value.
Array3 (n): Contains the n^{th} candidate point's evaluation value.

The value of n is the candidate point number assigned when the data was sorted according to the search group's sorting conditions.

The order of the data stored in the arrays is determined by the sorting condition of the SMRUN, SMGRUN, or SQRUN command.

Be sure that the arrays are larger than the maximum number of candidate points that will be stored in the arrays.

SMMDATA2 Search Model get Multi DATA 2

(Command)

Action Reads a batch of search models' search results for multiple search models.

Format SMMDATA2 *model# array*, *number*, *data type*, [*array 1*] [*,* [*array 2*] [*,* [*array 3*]]

Description The search results for the search models specified in the *model# array* parameter are read in order and stored in array 1, array 2, and array 3.

Use the *number* parameter to specify the number of search model that are contained in the model# array.

Specify the one-dimensional array that contains the desired search model (0 to 435) with the *model# array* parameter. This array must be defined in advance with the DIM command.

Set the *data type* parameter to 0 or 1.

0: Evaluation value maximum point information
1: Evaluation value minimum point information

The results won't be stored for elements in the *model# array* that have data of –1.

Specify the one-dimensional arrays that will contain the results in *array 1* through *array 3*. If one of these array names is omitted, the search results for that array won't be stored.

SMMODE Search Model MODE

(Command)

Action Specifies the search model's search mode.

Format SMMODE *model#*, [*evaluation feature*] [, [*search items*] [, [*evaluation level*] [, *absolute value mode*]]]

Description Sets the search mode and the conditions (used when search processing is executed) for the specified search model.

Specify the desired search model with the *model#* parameter (0 to 435).

The *evaluation feature* parameter specifies what kind of search processing is to be performed. If this parameter is omitted, the previous setting will remain effective.

0: Gray correlation
1: Average density level
2: Density deviation level

The *search items* parameter specifies a target for the maximum number of positions (candidate points) to be searched. The *search items* parameter can be set from 0 to 128. When 0 is specified, only information on the evaluation feature's maximum position and minimum position can be read.

The *evaluation level* parameter specifies the evaluation level for the specified feature, as shown in the following table.

Evaluation feature		Evaluation level range	Default setting
0	Gray correlation	0 to 100	70.0
1	Average density level	0 to 255	128.0
2	Density deviation level	0 to 127.5	64.0

Only positions with an evaluation value equal to or greater than the *evaluation level* setting will be stored as search results, so the detection standards become stricter as the *evaluation level* is increased.

When the *evaluation level* parameter is omitted, the default setting shown in the table above will be used. When both the *evaluation level* and *evaluation level* parameters are omitted, the previous settings for these parameters will remain in effect.

The *absolute value mode* specifies whether or not to to reverse the image gray level before performing the search. The default setting is 0.

0: Do not reverse.
Other than 0: Reverse.

The *absolute value mode* is valid only when the evaluation feature is set to 0 (gray correlation).

The previous setting will be used if the other parameters are omitted.

These settings won't be changed when the search model is registered or the search region is changed.

Search model's model#

Model registration	
Model image	
Upper-left X coordinate at registration	
Upper-left Y coordinate at registration	
X-direction model size	
Y-direction model size	
Model reference point X coordinate	
Model reference point Y coordinate	
Model reference angle	
Search region X1	
Search region Y1	
Search region X2	
Search region Y2	
Evaluation feature	← *evaluation feature*
Number of search items	← *search items*
Evaluation level	← *evaluation level*
Noise level	
Absolute value mode	← *absolute value mode*
Model name	
Search group number	
System model number	
Feature extraction window X width	
Feature extraction window Y height	
Feature extraction window X change	
Feature extraction window Y change	
Dispersion level	
Σ M	
Maximum point storage start number	

SMNAME Search Model NAME

(Command)

Action Sets the search model's name.

Format SMNAME *model#*, *character string*

Description Sets the name specified in *character string* as the name of the specified search model. The name can be used to distinguish the search model from other models.

Specify the desired name with the *character string* parameter. The name can be up to 11 bytes long and can contain character codes 01 to FF (hexadecimal).

The name has absolutely no effect on search execution or search results. The name will be retained unchanged even if the search model is registered.

SMPUT Search Model PUT

(Command)

Action Registers the image within the specified region of image memory as a search model image.

Format SMPUT *model#*, [*page#*], *X1*, *Y1*, *X2*, *Y2* [, *SX*, *SY*]

Description Registers the specified rectangular region in image memory as a search model image. Once the search model data has been registered, it is retained in the system even if the power is turned off.

The SMPUT command just registers the image data for search processing; the conditions for search processing are set with commands such as SMMODE and SMGMODE.

Specify the desired search model with the *model#* parameter (0 to 435).

Specify the image memory page number (0 or 1) that contains the desired image with the *page#* parameter. The default setting is 0.

The search model image must be input or drawn into the specified image memory page in advance.

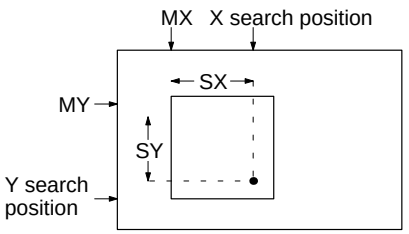
Specify the upper-left and lower-right corners of the rectangular region with points *(X1*, *Y1)* and *(X2*, *Y2)*. These points must meet the following conditions:

$0 \leq X1 < X2 \leq 511$
$0 \leq Y1 < Y2 \leq 483$
$23 \leq X2-X1+1 \leq 71$
$19 \leq Y2-Y1+1 \leq 67$

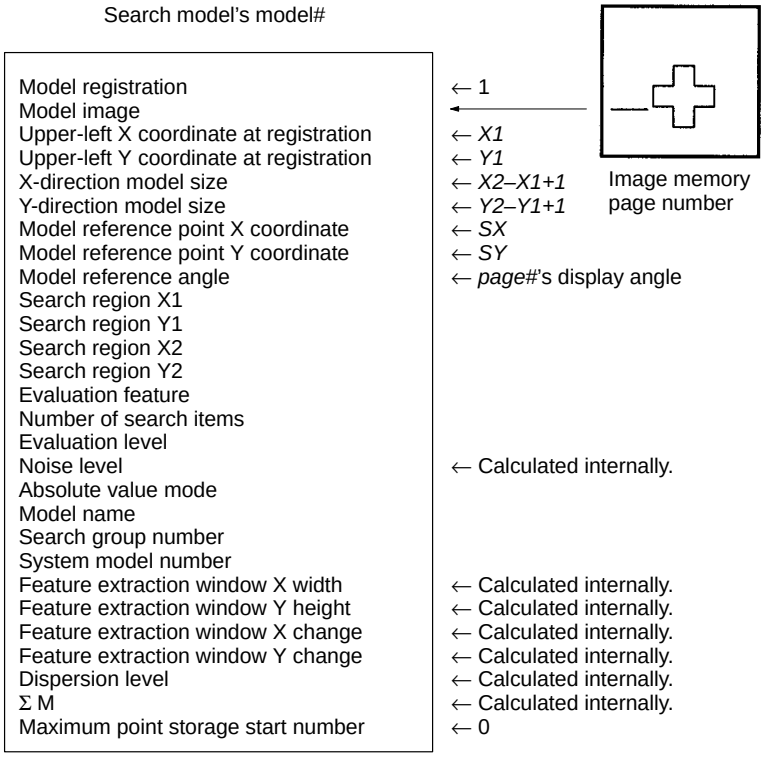
The size of the rectangular region must be from 23×19 to 71×67.

Parameters *SX* and *SY* specify the position of the search model data's reference point relative to *X1*, *Y1*. The search results (search position) is stored in SX and SY. The default setting for (SX, SY) is (0, 0). The following diagram shows the search position relative to the absolute coordinates of the upper-left corner of the search model (MX, MY).

X search position = MX + SX
Y search position = MY + SY



The specified image memory page's display angle is automatically registered as the model's reference angle.



SMPUT2 Search Model PUT 2

(Command)

Action Manually registers the image within the specified region of image memory as a search model image.

Format SMPUT2 *model#*, [*page#*], *X1*, *Y1*, *X2*, *Y2*, *AX1*, *AY1*, *AX2*, *AY2*, [, *SX*, *SY*]

Description Registers the specified rectangular region in image memory as a search model image. The SMPUT2 command can be used to make more precise settings than the SMPUT command. Normally, data obtained with the SMSELECT command are used as parameters in the SMPUT2 command.

The specified search model will be registered when the SMPUT2 command is executed. When a search model image hasn't been registered, that model's search processing can't be executed.

Once the search model data has been registered, it is retained in the system even if the power is turned off.

The SMPUT2 command just registers the image data for search processing; the conditions for search processing are set with commands such as SMMODE and SMGMODE.

Specify the desired search model with the *model#* parameter (0 to 435).

Specify the image memory page number (0 or 1) that contains the desired image with the *page#* parameter. The default setting is 0.

The search model image must be input or drawn into the specified image memory page in advance.

Specify the upper-left and lower-right corners of the search model image's rectangular region with points *(X1*, *Y1)* and *(X2*, *Y2)*. These points must meet the following conditions:

$0 \leq X1 \leq 489$
$0 \leq Y1 \leq 465$

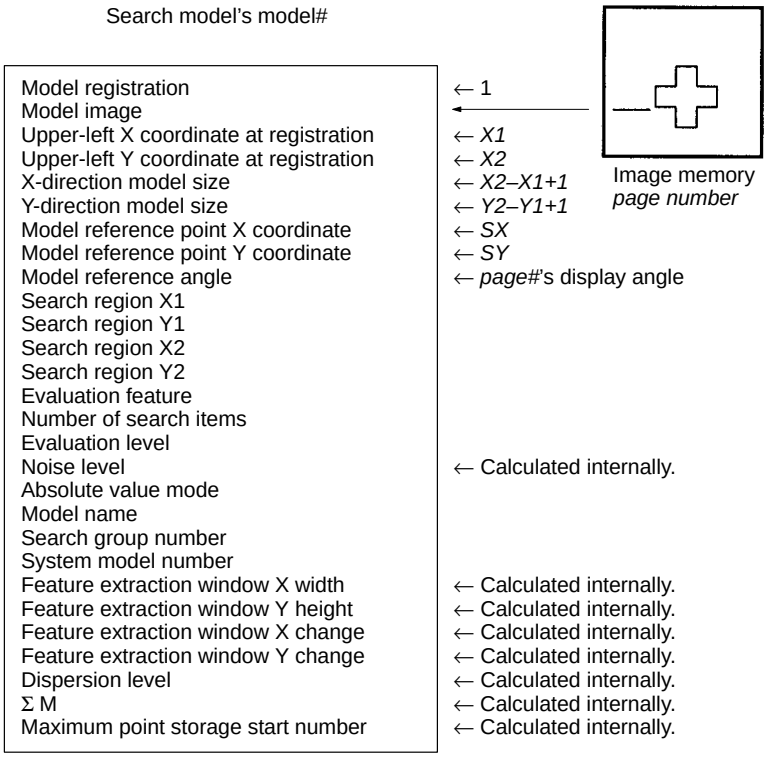
22 ≤ X2 ≤ 510
18 ≤ Y2 ≤ 484

Specify the upper-left and lower-right corners of the focus region with points *(AX1*, *AY1)* and *(AX2*, *AY2)*. Normally, this region is determined in advance with the SMSELECT command. These points must meet the following conditions:

X1 ≤ AX1 < AX2 ≤ X2
Y1 ≤ AY1 < AY2 ≤ Y2
23 ≤ AX2–AX1+1 ≤ 71
19 ≤ AY2–AY1+1 ≤ 67

Parameters *SX* and *SY* specify the position of the search model data's reference point relative to *X1*, *Y1*. The search results (search position) is stored in SX and SY. If SX and SY are omitted or are outside the search region, SX and SY will be set to the upper-left corner of the rectangular region.

The specified image memory page's display angle is automatically registered as the model's reference angle.



SMROTATE Search Model ROTATE

(Command)

Action Continuously registers the search model image as it is rotated.

Format SMROTATE *model#*, *group#*, [*page#*], [*CX*], [*CY*], *start angle*, *end angle* [, *step angle*]

Description Automatically creates a rotated search model for angle detection in search processing. When the search models rotated with the SMROTATE command are registered in a search group and search processing is performed on the group, the angle of rotation can be determined from the model number of the rotated model with the highest evaluation level.

The specified search model will be registered when the SMROTATE command is executed.

The model region and rotation center set for the specified model are referenced, so the model data must be registered in advance with the SMPUT or SMPUT2 command.

Image memory is consecutively rotated around (X1+CX, Y1+CY) and the region (X1, Y1) to (X2, Y2) is registered as model data. The point X1, Y1 is the upper-left corner of the specified model region registered with the SMPUT command.

Specify the desired search model with the *model#* parameter (0 to 435). Specify the search group (0 to 511) in which the rotated model will be registered with the *group#* parameter.

Specify the image memory page number (0 or 1) that contains the desired image with the *page#* parameter. The default setting is 0.

Specify the rotational center coordinates relative to the upper-left corner of the model region with *CX* and *CY*. If these coordinates are omitted, the search model's reference position (SX, SY) will be used instead of (CX, CY). The search position will be (SX, SY).

Specify the starting and ending angles for the rotation models in degrees with the *start angle* and *end angle* parameters. Specify how many degrees of separation to rotate between registrations with the *step angle* parameter. The default step angle is 5°.

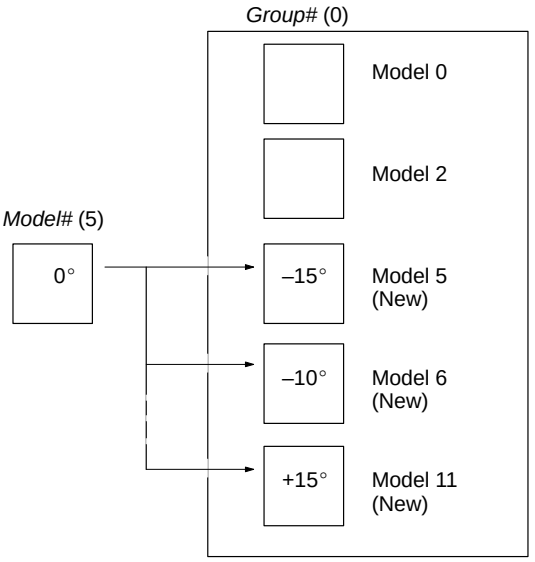
Unregistered model numbers following the specified model number will be used as rotation models, so a sufficient number of unregistered models must be set aside following the specified model number.

The specified search model will be registered in the specified group, too.

When search models are registered in the search group, the models are allocated the lowest available model numbers. Search models previously registered in that search group won't be cleared.

The display image might rotate as models are registered.

The following diagram shows an example of rotation model registration. Two models (model 0 and model 2) are already registered in group 0. Search model 5 is rotated between –15° and +15° in 5° steps and these rotation models are registered in model numbers 5 through 11. (Models 0, 2, and 5 were the only 3 models registered before execution of SMROTATE.)



Any conditions set for the specified model, such as the evaluation feature registered with the SMMODE command, will be copied to the newly created models.

SMRUN Search Model RUN

(Command)

Action Executes search processing for the specified search model or group.

Format SMRUN *start model#* [, [*end model#*] [, [*processing mode*] [, [*page#*] [, [*input path*] [, [*input mode*] [, *trigger mode*]]]]]]

Description Executes search processing for the specified search models. Specify the desired start and end models with the *start model#* and *end model#*. These numbers can be set from 0 to 435. If the end model number is omitted, search processing will be performed for the start search model only.

The models specified by the *start model#* and *end model#* parameters must be registered beforehand by a command such as SMPUT. A correct measurement can't be made if the model hasn't been registered.

The *processing mode* parameter specifies how the search data will be sorted after the search is completed, as shown below. The default setting is 1.

Mode	Sorting process
0	No sorting (order of appearance)
1	Sort each model by evaluation value.
2	Sort each model by X search position (descending order).
3	Sort each model by X search position (ascending order).
4	Sort each model by Y search position (descending order).
5	Sort each model by Y search position (ascending order).

Read the search results with the SMDATA function, SMMDATA or SMGDATA command.

The conditions for each search model must be set in advance with the SMMODE command.

Search processing is performed on the gray image in image bus 0 (pipeline bus), so the settings must be made so that binary conversion isn't performed with LUT.

When the *page#* and later parameters are specified, search processing is performed as the image is input into image memory.

The *page#* parameter specifies the page number of the image memory page where the image will be input. There is no image input when this parameter is omitted.

0: Input image memory 0 only.
1: Input image memory 1 only.
–1: Input to all of image memory.

The *input path* parameter specifies which bus will be used to input the image. The default setting is 0.

0: Image bus 1 (video bus, gray images)
1: Image bus 0 (pipeline bus, mainly filtered images)

The *input mode* parameter specifies whether the measurement is performed in frame mode or field mode. The mode specified with the FMODE command is used when this parameter is omitted. The mode is set to frame mode when OVL is first started.

The following table shows the differences between frame mode and field mode.

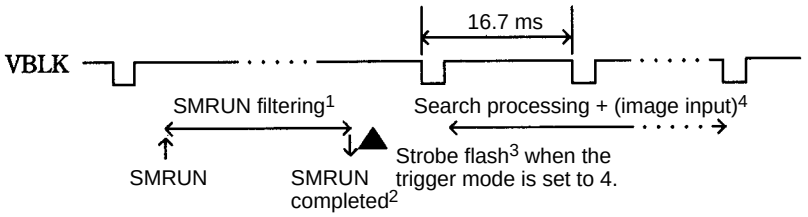
Mode		Resolution	Image input time (excluding wait time)
0	Frame mode	512×512 pixels	33.3 ms
1	Field mode	512×256 pixels	16.7 ms

The *trigger mode* parameter determines when the image is input, as shown in the following table. The default setting is 1.

Trigger mode		Image input timing
0	Stop	Cancels the STEP synchronization setting.
1	No strobe flash	A single image is input when SMRUN is executed.
2	STEP synchronization	A strobe flash is emitted in sync with the STEP signal and the image is input to image memory at that time.
3	Reserved	Not used.
4	Single strobe flash	A strobe flash is emitted and a single image is input when SMRUN is executed.

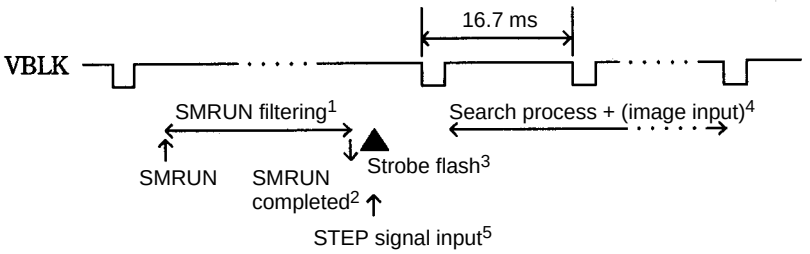
When the trigger mode is set to "2: STEP synchronization," search processing + (image input) is performed automatically when the STEP signal is input. The STEP synchronization setting is cleared once the search has been performed with STEP synchronization.

The following diagram shows the timing of operation when the trigger mode is set to "1: No strobe flash" or "4: Single strobe flash."



Note
1. Several ms are required for the SMRUN command's filtering.
2. The SMRUN command ends when it's filtering is completed.
3. When the SMRUN command is completed, a strobe flash is emitted and the search process + (image input) is started.
4. The time required for search process + image input depends on the input mode and the number of models being searched.

The following diagram shows the timing of operation when the trigger mode is set to "2: STEP synchronization."



Note
1. Several ms are required for the SMRUN command's filtering.
2. The SMRUN command ends when it's filtering is completed.
3. When the STEP signal is input, a strobe flash is emitted and the search process + (image input) is started.
4. The time required for search process + image input depends on the input mode and the number of models being searched.
5. The STEP signal can be input after the SMRUN command has been completed.

SMSAVE Search Model SAVE

(Command)

Action Saves the search model data to the memory card.

Format SMSAVE *file name*, *start model#* [[*, end model#*], *skip*]

Description Saves the data of the specified search models in the memory card. Specify the file in which the data will be saved with the *file name* parameter.

Specify the first desired search model (0 to 435) with the *start model#* parameter. Specify the last desired search model (0 to 435) with the *end model#* parameter. When the *end model#* parameter is omitted, just the search model specified by *start model#* will be saved.

The following values can be set for the *skip* parameter. The default setting is 0.

0: Save all of the specified search models. (default)
Other than 0: Save only those specified models that have been registered.

Data is compressed before being saved in the file.

Data can't be saved to the RS-232C port.

SMSELECT Search Model SELECT

(Command)

Action Determines the best search model positions.

Format SMSELECT [*page#*], *X1*, *Y1*, *X2*, *Y2*, *positions*, *AX1*, *AY1*, *AX2*, *AY2* [, *window function* [, *window plane#*]]

Description Finds the specified number of regions (specified with the *positions* parameter) that are suitable as model positions and stores these positions in arrays AX1, AY1, AX2, and AY2.

The SMSELECT command is normally used to determine positions when registering search models with the SMPUT command.

Specify the image memory page number (0 or 1) that contains the desired image with the *page#* parameter. The default setting is 0. An image must be input into the specified image memory page in advance.

Specify the upper-left and lower-right corners of the region to be searched for candidate search models with points *(X1*, *Y1)* and *(X2*, *Y2)*. The size of the rectangular region must be from 23×19 to 512×484 pixels.

Parameters AX1, AY1, AX2, and AY2 specify the 4 one-dimensional arrays that will contain the best positions, as follows.

AX1: Contains the X coordinates of the upper-left corners.
AY1: Contains the Y coordinates of the upper-left corners.
AX2: Contains the X coordinates of the lower-right corners.
AY2: Contains the Y coordinates of the lower-right corners.

These arrays must be defined in advance with the DIM command. The absolute coordinates of the positions are stored in the order of their suitability as search models. In other words, the most suitable position is first and the least suitable is last.

Specify one of the following settings for the *window function* parameter. The default setting is 0.

Setting		Processing
0	Window function OFF	Processing is performed in the entire region.
1	Window function ON	Processing is performed only in the window region that was drawn.

The *window plane#* parameter specifies the plane (0 to 7) in which the window region's shape has been drawn. The default setting is 7.

SMSELECT2 Search Model SELECT 2

(Command)

Action Determines the best search model position.

Format SMSELECT2 [*page#*], *X1*, *Y1*, *X2*, *Y2*, *BX1*, *BY1*, *BX2*, *BY2*

Description Finds the region of image memory that is most suitable as a model position and stores this position in arrays BX1, BY1, BX2, and BY2.

The SMSELECT2 command is normally used to determine positions when registering search models with the RMTOSM command.

Specify the image memory page number (0 or 1) that contains the desired image with the *page#* parameter. The default setting is 0. An image must be input into the specified image memory page in advance.

Specify the upper-left and lower-right corners of the region to be searched for candidate search models with points *(X1*, *Y1)* and *(X2*, *Y2)*. The size of the rectangular region must be from 23×19 to 71×67 pixels.

Parameters BX1, BY1, BX2, and BY2 specify the 4 one-dimensional arrays that will contain the best positions, as follows.

BX1: Contains the X coordinates of the upper-left corners.
BY1: Contains the Y coordinates of the upper-left corners.
BX2: Contains the X coordinates of the lower-right corners.
BY2: Contains the Y coordinates of the lower-right corners.

These arrays must be defined in advance with the DIM command.

SMSORT Search Model SORT

(Command)

Action Sorts the search model numbers in accordance with the conditions and allows faster searches.

Format SMSORT *sort type*

Description When executing a search of several models, those models must be consecutive in the system's internal model region. The SMSORT command sorts the models according to the specified conditions so that they are consecutive.

Before executing a search with the SMGRUN, SMRUN, or SQRUN command, the models must be sorted into the required order with the SMSORT command.

The model numbers won't change when the models are sorted. For example, the model data registered with model number 0 is still handled with model number 0 after the models have been sorted.

A long time might be required for search processing if the models aren't sorted.

Specify the type of sorting to be performed with the *sort type* parameter.

0: Specify 0 when searching by model number (from the lowest number).
1: Specify 1 when searching by group number (from the lowest number).

The sorting can take some time, so SMSORT must be executed prior to the search.

The order of the search models won't change unless search models are registered or loaded.

SOBEL SOBEL

(Command)

Action Performs sobel processing.

Format SOBEL [[*page#*] [, [*X1*] [, [*Y1*] [, [*X2*] [, [*Y2*]]]]]]

Description Performs sobel processing on the specified rectangular region in image memory.

Specify the image memory page number (0 or 1) that contains the desired image with the *page#* parameter. The default setting is 0.

Specify the upper-left and lower-right corners of the rectangular region with points *(X1*, *Y1)* and *(X2*, *Y2)*. The default points are (1, 1) and (510, 510).

SPACE$ SPACE $

(Function)

Action Creates a character string comprised of the number of spaces specified by the numeric expression.

Format SPACE$ (*numeric expression*)

Description The SPACE$ function returns a character string containing the number of space characters (CHR$(32)) specified by the *numeric expression*.

Specify the *numeric expression* as a value between 0 and 255.

The STRING$ function is similar to the SPACE$ function.

SPC SPaCe

(Function)

Action Outputs the specified number of spaces.

Format SPC (*numeric expression*)

Description The SPC function creates a character string containing the number of spaces specified by the *numeric expression*. The SPC function cannot be used alone. Use it together with the PRINT command.

A negative numeric expression (-2^{15} to –1) is treated as 0. If the designated characters do not fit in one line, the remainder obtained by dividing the number of the designated characters by the number of characters fitting in one line shall be used as the set value.

SPCLOSE SPline CLOSE

(Command)

Action Draws a region bounded by a spline curve in VRAM.

Format SPCLOSE *number of data, X array, Y array, VRAM,* [, [*page#*] [, [*density* ***or*** *drawing mode*] [, *lineart*]]]

Description The SPCLOSE command draws a region bounded by a spline curve. The spline curve is defined by the coordinate points in *X array* and *Y array* between the first array element and the element defined by the number of data parameter.

The *number of data* must not exceed 64.

Specify the *VRAM* where the region is drawn with a number, as follows:

0: Character memory
1: Graphic memory
2: Window memory
3: Image memory
4: Shading memory (Can't be used in the F350-C12E/C41E.)

Omit the *page#* (default setting 0) or set it to 1.

Specify the *drawing method* with the *drawing density* or the *drawing mode*. The default value is *drawing mode*, OR.

Specify with the *lineart* parameter if the spline curve is an outline only or filled. The default setting is 0.

0: Filled spline curve
Other than 0: Spline curve outline only

When writing to a frame memory, the contents of planes write protected with the MASKBIT command remain unchanged.

SPLINE SPLINE

(Command)

Action Draws a spline curve in VRAM.

Format SPLINE *number of data, X array, Y array, VRAM,* [, [*page#*] [*, density* ***or*** *drawing mode*]]

Description The SPLINE command draws a spline curve defined by the coordinate points in *X array* and *Y array* between the first array element and the element defined by the *number of data* parameter.

The *number of data* must not exceed 64.

Specify the *VRAM* where the spline curve is drawn with a number, as follows:

0: Character memory
1: Graphic memory
2: Window memory
3: Image memory
4: Shading memory (Can't be used in the F350-C12E/C41E.)

Omit the *page#* (default setting 0) or set it to 1.

Specify the *drawing method* with the *drawing density* or the *drawing mode*. The default value is *drawing mode*, OR.

When writing to a frame memory, the contents of planes write protected with the MASKBIT command remain unchanged.

SQINFO SeQuential INFOrmation

(Function)

Action Reads information set by sequential processing.

Format SQINFO (*unit#*, *data type* [, *data#*])

Description Reads various types of information set by sequential processing.

Specify the unit number (0 to 15) of the Measurement Unit that contains the desired data with the *unit#* parameter.

Set the *data type* and *data#* parameters as shown in the following table to read the desired data. Refer to the description of the SQSET command for more details on the returned values.

Data type		Data#	Returned value
0	Strobe flash mode	Can be omitted.	0:No flash 1:Flash
1	Pipeline specification		0: Non-pipeline with break 1: Pipeline without break 2: Non-pipeline
2	Measurement type		0: No measurement 1: Binary raster 2: Search
3	Measurement units		0: Frame 1: Field
4	Starting search group number		0 to 511
5	Ending search group number		0 to 511
6	Search results sorting condition		0 to 14 (See SQSET.)
7	Image input page		0: Page 0 only 1: Page 1 only –1: All pages –2: Image input only
8	Image input path		0 to 3 (See SQSET.)
9	Image input mode		0: Frame 1: Field
10	Image scroll status		0: Process at original position 1: Process scroll status
11	Unit mask specification		0: No mask 1: Mask 2: Master for raster processing only
12	Shading bank		0: Bank 0 1: Bank 1

Data type		Data#	Returned value
13	Video bus input path	Can be omitted.	0: Camera 1: Image memory
14	Video bus input page number		0: Image memory 0 1: Image memory 1 –1: Input from camera
15	Input camera number		0 to 7
16	Filter selection		0 to 11 (See FILTSEL.)
17	Display image		0 to 31 (See DISPLAY.)
18	Display path		0 to 3 (See DISPLAY.)
19	Window display image type		0 to 255 (See WDISP.)
20	Window display ON/OFF		0 to 255 (See WDISP.)
21	Window binary reverse		0 to 255 (See WDISP.)
22	Level outside the image memory region		0 to 255
23	Search completed machine language subroutine number		0 to 9 (–1: not registered)
24	ROI processing completed machine language subroutine number		0 to 9 (–1: not registered)
25	Measurement pixels for binary raster measurement	Plane number (0 to 7)	0: Black pixels 1: White pixels
26	Window function for binary raster measurement		0: OFF 1: ON
27	Paint function for binary raster measurement		0: Normal measurement 1: Paint 2: Matching
28	Fill/outline function for binary raster measurement		0: OFF 1: ON
29	Detailed run length binary plane number	Can be omitted.	0 to 7
30	Detailed run length measurement pixels		0: Black pixels 1: White pixels
31	Detailed run length window function		0: OFF 1: ON
32	Detailed run length measurement direction		0: X direction 1: Y direction –1: Stop measurement
33	Detailed run length noise reduction number		0 to 3
34	Detailed run length image slicing function		0: OFF 1: ON
35	LUT setting classification		0: LUT ON (negative regions = 0) 1: LUT ON (absolute value output) 2: LUT ON (Adjust LUT contents when SQSET is executed.) 3: LUT ON (Used with binary level set for SQSET's condition 3.) 4: LUT OFF
36	Binary level lower limit	Plane number (0 to 7)	0 to 255
37	Binary level upper limit		0 to 255
38	Binary level setting mode		0: Positive direction only 1: Target system
39	ROI Y-direction processing position adjustment width	Can be omitted.	0: Specified position only 1: Specified position ±1 pixel in Y direction 2: Specified position ±2 pixels in Y direction
40	ROI X-direction processing position adjustment width		0: Specified position only 1: Specified position ±1 pixel in X direction 2: Specified position ±2 pixels in X direction

SQINIT SeQuential INITialize

(Command)

Action Initializes the sequential processing conditions.

Format SQINIT *unit#*

Description Initializes sequential processing in the specified Measurement Unit.

The *unit#* parameter specifies the unit number (0 to 15) of the desired Measurement Unit. Specify –1 to initialize all Measurement Units.

Use the SQINIT command to return the conditions to their default settings the first time they are used or after they have been set with the SQSET command.

The SQINIT command can't be executed while sequential processing is being performed.

Initialization by the SQINIT command is the same as executing the SQSET command with the following parameters:

SQSET *unit#*

,"F0 @P0 M0 S0 @S0 V Z0 @M0"
,"@S0 B0 C0 F0 D31,2,0,255,0 L0 @O0 <–1 >–1"
,"M–1,1,1,0,0 R0,1,1,0,0,0"

Condition 1 parameters:

F0: No strobe flash when the Unit starts up
@P0: Non-pipeline processing, with break
M0,0: No measurement processing (neither binary raster nor search)
S0,0: Search group number = 0
@S0: Search data sort condition = no sorting
V: No image input
Z0: No image scroll
@M0: Don't mask the Unit

Condition 2 parameters:

@S0: Shading memory bank = 0
B0,0: Image read path = camera image
C0: Camera number = 0
F0: Filtering OFF
D31,2,0,255,0: All images gray display, window display ON
L0: Gray LUT OFF
@O0: Level outside of image memory region = 0
<–1: No "Search processing completed machine language subroutine"
>–1: No "ROI processing completed machine language subroutine"

Condition 3 parameters:

M–1,1,1,0,0: All planes, white pixels, window ON, normal measurement, no outline
R0,1,1,0,0,0: Plane 0, white pixels, window ON, X direction, noise 0, image slicing OFF

SQJDATA SeQuential Judgement DATA

(Function)

Action Reads sequential processing judgement results.

Format SQJDATA (*unit#*, *data type*)

Description Reads the overall judgement results from ROI processing in tact units or Measurement Units.

Specify the unit number (0 to 15) of the desired Measurement Unit with the *unit#* parameter. The overall judgement results of all Units can be read by specifying –1.

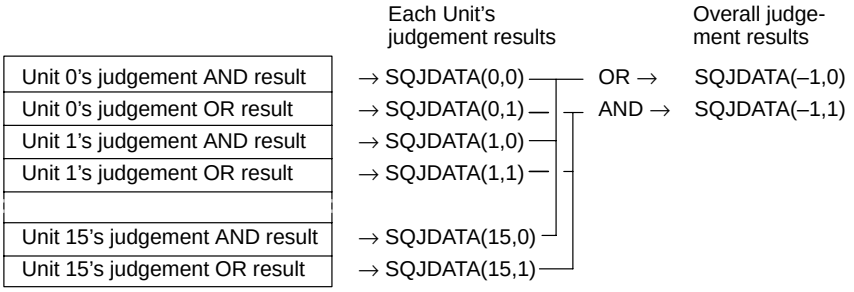
The overall judgement results can have one of the following 3 values:

–2: All of the ROI processing results are “no judgement ”.
–1: No good
0: OK

The *data type* parameter specifies how to read the overall judgement, as shown below.

0: AND (–1 if all of the ROI processing results are –1; otherwise 0.)
1: OR (–1 if even one of the ROI processing results is –1; otherwise 0.)

The SQJDATA command handles only those ROI processing results specified by the SQMODE command. The SQJDATA command doesn’t handle the results of ROI processing from the measurement sequence interrupt subroutine defined by the ON SQMEAS GOSUB command.



SQMDATA SeQuential Measure DATA

(Function)

Action Reads binary the raster measurement results from sequential processing.

Format SQMDATA (*unit#*, *binary image plane#*, *data type*)

Description Reads the results of ROI processing of binary raster measurements. The binary raster measurement function can’t be used in the F350-C12E/C41E.

Specify the unit number (0 to 15) of the desired Measurement Unit with the *unit#* parameter. Specify the desired binary image plane (0 to 7) with the *binary image plane#* parameter.

The *data type* parameter can specify one of the following 4 values:

0: Area
2: X center-of-gravity
4: Y center-of-gravity
6: Main axis angle (degrees)

There is a storage area for the binary raster measurement results in each Measurement Unit, so the measurement results form different Measurement Units can be read after sequential processing has been completed.

Binary raster measurement results

Unit 0, plane 0 area	→ SQMDATA(0,0,0)
Unit 0, plane 0 X center-of-gravity	→ SQMDATA(0,0,2)
Unit 0, plane 0 Y center-of-gravity	→ SQMDATA(0,0,4)
Unit 0, plane 0 main axis angle	→ SQMDATA(0,0,6)
Unit 0, plane 1 area	→ SQMDATA(0,1,0)
Unit 0, plane 1 X center-of-gravity	→ SQMDATA(0,1,2)
Unit 0, plane 1 Y center-of-gravity	→ SQMDATA(0,1,4)
Unit 0, plane 1 main axis angle	→ SQMDATA(0,1,6)
Unit 0, plane 7 area	→ SQMDATA(0,7,0)
Unit 0, plane 7 X center-of-gravity	→ SQMDATA(0,7,2)
Unit 0, plane 7 Y center-of-gravity	→ SQMDATA(0,7,4)
Unit 0, plane 7 main axis angle	→ SQMDATA(0,7,6)
Unit 1, plane 0 area	→ SQMDATA(1,0,0)
Unit 1, plane 0 X center-of-gravity	→ SQMDATA(1,0,2)
Unit 1, plane 0 Y center-of-gravity	→ SQMDATA(1,0,4)
Unit 1, plane 0 main axis angle	→ SQMDATA(1,0,6)
Unit 15, plane 7 area	→ SQMDATA(15,7,0)
Unit 15, plane 7 X center-of-gravity	→ SQMDATA(15,7,2)
Unit 15, plane 7 Y center-of-gravity	→ SQMDATA(15,7,4)
Unit 15, plane 7 main axis angle	→ SQMDATA(15,7,6)

SQMEAS ON/OFF/STOP SeQuential MEASure ON/OFF/STOP

(Command)

Action Enables, disables, or stops measurement sequence interrupts.

Format SQMEAS [(*unit#*)] ON ***or*** OFF ***or*** STOP

Description Controls measurement sequence interrupts. Specify the unit number (0 to 15) of the desired Measurement Unit with the *unit#* parameter. If the unit number is omitted, the command will act on all Measurement Units.

SQMEAS ON: The SQMEAS ON statement enables measurement sequence interrupts.

SQMEAS OFF: The SQMEAS OFF statement disables measurement sequence interrupts. After the SQMEAS OFF statement is executed, an interrupt won't be generated even if the interrupt condition is met.

SQMEAS STOP: The SQMEAS STOP statement disables measurement sequence interrupts. After the SQMEAS STOP statement is executed, an interrupt won't be generated even if the interrupt condition is met. However, the interrupt is recorded so that operation branches to the interrupt processing routine the next time that a SQMEAS ON statement is executed.

SQMODE SeQuence MODE

(Command)

Action Specifies the processing mode for sequential processing.

Format SQMODE *unit#*, *processing mode*, [*elements*], *array1* [, *array2*]

Description Sets what kind of ROI processing is to be performed for sequential processing in individual Measurement Unit.

The *unit#* parameter specifies the unit number (0 to 15) of the desired Measurement Unit.

The *processing mode* parameter specifies settings such as the position compensation mode, ROI processing contents, and relationship between search processing and ROI processing.

The *elements* parameter specifies the number of array elements for processing modes in which the number of elements varies in array1 and array2. This parameter can be omitted for processing modes in which the array's number of elements is fixed.

The following table shows the relationship between the *processing mode* and the *elements*, *array1*, and *array2* parameters. Parameters that have "Omit" in the column are usually omitted. Array1 and Array2 indicate variable names containing the indicated number of elements.

Processing mode		Elements	Array1	Array2
0	No position compensation	Omit	Elements: 1	Omit
1	1-model positioning (no rotation)		Elements: 3	
2	1-model positioning (with rotation)		Elements: 3	
3	2-model positioning (no rotation)		Elements: 5	
4	2-model positioning (with rotation)		Elements: 5	
5	2-model positioning 2 (no rotation)		Elements: 3	
6	2-model positioning 2 (with rotation)		Elements: 3	
7	Circular positioning (no rotation)		Elements: 5	
8	Circular positioning (with rotation)		Elements: 6	
9	Search model evaluation value max. point → ROI model	Number of search models	Same as *elements*.	Same as *elements*.
10	Search model all candidate points → ROI model			
11	Search group evaluation value max. point → ROI model	Number of search groups		
12	Search group all candidate points → ROI model			
13	Search model evaluation value max. point → ROI group	Number of search models		
14	Search model all candidate points → ROI group			
15	Search group evaluation value max. point → ROI group	Number of search groups		
16	Search group all candidate points → ROI group			
17	Search model evaluation value min. point → ROI model	Number of search models		
18	Search group evaluation value min. point → ROI model			
19	Search model evaluation value min. point → ROI group			
20	Search group evaluation value min. point → ROI group			
50	1-model positioning (with rotation and angle interpolation)	Omit	Elements: 5	Omit
51	2-model positioning (with rotation and angle correspondence)		Elements: 5	
52	Circular positioning (with rotation and ROI verification)		Elements: 9	
53	Circular positioning (with rotation, angle detection from one model, and ROI verification)		Elements: 11	
54	Circular positioning (with rotation, angle detection from the center point and an arc point, and ROI verification)		Elements: 11	
55	Circular positioning (with rotation, defect rotation, and ROI verification)		Elements: 11	

0, 1, 2, 3... 1. Processing mode = 0: No position compensation

Uses the position at which the ROI model was registered regardless of search processing and performs ROI processing on the ROI models in the specified ROI group.

Array1(0) = ROI group number (0 to 511)

2. Processing mode = 1: 1-model positioning (no rotation)

Sets a single search model's search position as the reference point and performs ROI processing on the ROI models in the specified ROI group.

When an ROI verification model has been specified, the correlation of the detected candidate points is measured in the ROI verification model and the position with the highest correlation value is set as the ROI reference point. Be sure to use the RMMODE command to set the ROI verification model so that the correlation value is measured.

A precision search is performed to detect the position when an ROI model set for precision search has been specified as the ROI verification model.

Array1(0) = Search model (0 to 435) used for reference point detection
Array1(1) = ROI group number (0 to 511) or –1 to disable ROI
Array1(2) = ROI verification model (0 to 1023) or –1 to disable verification

3. Processing mode = 2: 1-model positioning (with rotation)

Determines the position and angle of search models registered in a single search group, compensates the images, and performs ROI processing on all of the ROI models in the ROI group.

When an ROI verification model has been specified, the correlation of the detected candidate points is measured in the ROI verification model and the position with the highest correlation value is set as the ROI reference point.

A precision search is performed to detect the position when an ROI model set for precision search has been specified as the ROI verification model.

Normally, the search models registered in the group are rotation models rotated and registered with the SMROTATE command.

Array1(0) = Search group (0 to 511) used for reference point/angle detection
Array1(1) = ROI group number (0 to 511) or –1 to disable ROI
Array1(2) = ROI verification model (0 to 1023) or –1 to disable verification

4. Processing mode = 3: 2-model positioning (no rotation)

Sets the midpoint of two search models' search positions as the reference point and performs ROI processing on the ROI models in the specified ROI group.

When an ROI verification model has been specified, the correlation of the detected candidate points is measured in the ROI verification model and the position with the highest correlation value is set as the ROI reference point.

A precision search is performed to detect the position when an ROI model set for precision search has been specified as the ROI verification model.

Array1(0) = Search model (0 to 435) for the first point's position detection
Array1(1) = Search model (0 to 435) for the second point's position detection
Array1(2) = ROI group number (0 to 511) or –1 to disable ROI
Array1(3) = 1st point's ROI verification model (0 to 1023)
or –1 to disable verification
Array1(4) = 2nd point's ROI verification model (0 to 1023)
or –1 to disable verification

5. Processing mode = 4: 2-model positioning (with rotation)

   Determines the position of each of the two search groups, sets the midpoint and vector of the two search positions as the ROI processing reference point and inclination, and performs ROI processing on all of the ROI models in the ROI group.

   When an ROI verification model has been specified, the correlation of the detected candidate points is measured in the ROI verification model and the position with the highest correlation value is set as the ROI reference point.

   A precision search is performed to detect the position when an ROI model set for precision search has been specified as the ROI verification model.

   Normally, the search models registered in the group are rotation models rotated and registered with the SMROTATE command.

   Array1(0) = Search group (0 to 511) for the first point's position detection
   Array1(1) = Search group (0 to 511) for the second point's position detection
   Array1(2) = ROI group number (0 to 511) or –1 to disable ROI
   Array1(3) = 1st point's ROI verification model (0 to 1023)
   or –1 to disable verification
   Array1(4) = 2nd point's ROI verification model (0 to 1023)
   or –1 to disable verification

6. Processing mode = 5: 2-model positioning 2 (no rotation)

   Sets the midpoint between the first candidate points of the two search models as the reference point. The first and last candidate points are determined by the sorting condition in search processing.

   An example application for this mode is preforming position compensation on the first and last characters in a character string that contains several identical characters.

   Array1(0) = Search model (0 to 435) for the first point's position detection
   Array1(1) = Search model (0 to 435) for the second point's position detection
   Array1(2) = ROI group number (0 to 511) or –1 to disable ROI

7. Processing mode = 6: 2-model positioning 2 (with rotation)

   Sets the midpoint between the first candidate points of the two search models as the reference point. The first and last candidate points are determined by the sorting condition in search processing.

   An example application for this mode is preforming position compensation on the first and last characters in a character string that contains several identical characters.

   Array1(0) = Search group (0 to 511) for the first point's position detection
   Array1(1) = Search group (0 to 511) for the second point's position detection
   Array1(2) = ROI group number (0 to 511) or –1 to disable ROI

8. Processing mode = 7: Circular positioning (no rotation)

   Finds the center of a circular object using 4 search models that detect 4 points on the circle (left, bottom, right, and top), sets that center-point as the ROI reference point, and performs ROI processing on all of the ROI models registered in the ROI group.

   Array1(0) = Search model (0 to 435) for left circular edge detection
   Array1(1) = Search model (0 to 435) for bottom circular edge detection
   Array1(2) = Search model (0 to 435) for right circular edge detection
   Array1(3) = Search model (0 to 435) for top circular edge detection
   Array1(4) = ROI group number (0 to 511) or –1 to disable ROI

9. Processing mode = 8: Circular positioning (with rotation)

   Finds the center of a circular object using 4 search models that detect 4 points on the circle (left, bottom, right, and top) and finds the angle of the circular object using another search group. Sets the center-point as the ROI reference point and performs ROI processing on all of the ROI models registered in the ROI group.

   Array1(0) = Search model (0 to 435) for left circular edge detection
   Array1(1) = Search model (0 to 435) for bottom circular edge detection
   Array1(2) = Search model (0 to 435) for right circular edge detection
   Array1(3) = Search model (0 to 435) for top circular edge detection
   Array1(4) = Search group (0 to 511) for angle detection
   Array1(5) = ROI group number (0 to 511) or –1 to disable ROI

10. Processing mode = 9:
    Search model evaluation value max. point → ROI model processing

    Performs processing of a single ROI model at the search position where the search model's evaluation value is at its maximum.

    The order of the elements in array1 and array2 determines the correspondence between search models and ROI models.

    The number of data elements in array1 and array2 equals n.

    Array1(0) = 1st search model (0 to 435)
    Array1(1) = 2nd search model (0 to 435)
    : : :
    Array1(n–1) = nth search model (0 to 435)
    Array2(0) = 1st ROI model (0 to 1023) or –1 to disable ROI
    Array2(1) = 2nd ROI model (0 to 1023) or –1 to disable ROI
    : : :
    Array2(n–1) = nth ROI model (0 to 1023) or –1 to disable ROI

11. Processing mode = 10:
    Search model all candidate points → ROI model processing

    Performs ROI model processing consecutively at each candidate point of a single search model.

    The order of the elements in array1 and array2 determines the correspondence between search models and ROI models.

    The number of data elements in array1 and array2 equals n.

    Array1(0) = 1st search model (0 to 435)
    Array1(1) = 2nd search model (0 to 435)
    : : :
    Array1(n–1) = nth search model (0 to 435)
    Array2(0) = 1st ROI model (0 to 1023) or –1 to disable ROI
    Array2(1) = 2nd ROI model (0 to 1023) or –1 to disable ROI
    : : :
    Array2(n–1) = nth ROI model (0 to 1023) or –1 to disable ROI

12. Processing mode = 11:
    Search group evaluation value max. point → ROI model processing

    Performs processing of a single ROI model at the search position where the search group's evaluation value is at its maximum.

    The order of the elements in array1 and array2 determines the correspondence between search groups and ROI models.

    The number of data elements in array1 and array2 equals n.

    Array1(0) = 1st search group (0 to 511)
    Array1(1) = 2nd search group (0 to 511)
    : : :
    Array1(n–1) = nth search group (0 to 511)
    Array2(0) = 1st ROI model (0 to 1023) or –1 to disable ROI
    Array2(1) = 2nd ROI model (0 to 1023) or –1 to disable ROI
    : : :
    Array2(n–1) = nth ROI model (0 to 1023) or –1 to disable ROI

13. Processing mode = 12:
    Search group all candidate points → ROI model processing

    Performs ROI model processing consecutively at each candidate point of a single search group.

    The order of the elements in array1 and array2 determines the correspondence between search groups and ROI models.

    The number of data elements in array1 and array2 equals n.

    Array1(0) = 1st search group (0 to 511)
    Array1(1) = 2nd search group (0 to 511)
    : : :
    Array1(n–1) = nth search group (0 to 511)
    Array2(0) = 1st ROI model (0 to 1023) or –1 to disable ROI
    Array2(1) = 2nd ROI model (0 to 1023) or –1 to disable ROI
    : : :
    Array2(n–1) = nth ROI model (0 to 1023) or –1 to disable ROI

14. Processing mode = 13:
    Search model evaluation value max. point → ROI group processing

    Performs processing of the ROI models in the group at the search position where the search model's evaluation value is at its maximum.

    The order of the elements in array1 and array2 determines the correspondence between search models and ROI groups.

    The number of data elements in array1 and array2 equals n.

    Array1(0) = 1st search model (0 to 435)
    Array1(1) = 2nd search model (0 to 435)
    : : :
    Array1(n–1) = nth search model (0 to 435)
    Array2(0) = 1st ROI group (0 to 511) or –1 to disable ROI
    Array2(1) = 2nd ROI group (0 to 511) or –1 to disable ROI
    : : :
    Array2(n–1) = nth ROI group (0 to 511) or –1 to disable ROI

15. Processing mode = 14:
    Search model all candidate points → ROI group processing

    Performs processing consecutively on the ROI models belonging to a single ROI group at each candidate point of a single search model.

    The order of the elements in array1 and array2 determines the correspondence between search models and ROI groups.

    The number of data elements in array1 and array2 equals n.

    Array1(0) = 1st search model (0 to 435)
    Array1(1) = 2nd search model (0 to 435)
    : : :
    Array1(n–1) = nth search model (0 to 435)
    Array2(0) = 1st ROI group (0 to 511) or –1 to disable ROI
    Array2(1) = 2nd ROI group (0 to 511) or –1 to disable ROI
    : : :
    Array2(n–1) = nth ROI group (0 to 511) or –1 to disable ROI

16. Processing mode = 15:
    Search model evaluation value max. point → ROI group processing

    Performs processing of the ROI models in a group at the search position where the search group's evaluation value is at its maximum.

    The order of the elements in array1 and array2 determines the correspondence between search groups and ROI groups.

    The number of data elements in array1 and array2 equals n.

    Array1(0) = 1st search group (0 to 511)
    Array1(1) = 2nd search group (0 to 511)
    : : :
    Array1(n–1) = nth search group (0 to 511)
    Array2(0) = 1st ROI group (0 to 511) or –1 to disable ROI
    Array2(1) = 2nd ROI group (0 to 511) or –1 to disable ROI
    : : :
    Array2(n–1) = nth ROI group (0 to 511) or –1 to disable ROI

17. Processing mode = 16:
    Search group all candidate points → ROI group processing

    Performs processing consecutively on the ROI models belonging to a single ROI group at each candidate point of a search group.

    The order of the elements in array1 and array2 determines the correspondence between search groups and ROI groups.

    The number of data elements in array1 and array2 equals n.

    Array1(0) = 1st search group (0 to 511)
    Array1(1) = 2nd search group (0 to 511)
    : : :
    Array1(n–1) = nth search group (0 to 511)
    Array2(0) = 1st ROI group (0 to 511) or –1 to disable ROI
    Array2(1) = 2nd ROI group (0 to 511) or –1 to disable ROI
    : : :
    Array2(n–1) = nth ROI group (0 to 511) or –1 to disable ROI

18. Processing mode = 17:
    Search model evaluation value min. point → ROI model processing

    Performs processing of a single ROI model at the search position where the search model's evaluation value is at its minimum.

    The order of the elements in array1 and array2 determines the correspondence between search models and ROI models.

    The number of data elements in array1 and array2 equals n.

    Array1(0) = 1st search model (0 to 435)
    Array1(1) = 2nd search model (0 to 435)
    : : :
    Array1(n–1) = nth search model (0 to 435)
    Array2(0) = 1st ROI model (0 to 1023)
    Array2(1) = 2nd ROI model (0 to 1023)
    : : :
    Array2(n–1) = nth ROI model (0 to 1023)

19. Processing mode = 18:
    Search group evaluation value min. point → ROI model processing

    Performs processing of a single ROI model at the search position where the search group's evaluation value is at its minimum.

    The order of the elements in array1 and array2 determines the correspondence between search groups and ROI models.

    The number of data elements in array1 and array2 equals n.

    Array1(0) = 1st search group (0 to 511)
    Array1(1) = 2nd search group (0 to 511)
    : : :
    Array1(n–1) = nth search group (0 to 511)
    Array2(0) = 1st ROI model (0 to 1023)
    Array2(1) = 2nd ROI model (0 to 1023)
    : : :
    Array2(n–1) = nth ROI model (0 to 1023)

20. Processing mode = 19:
    Search model evaluation value min. point → ROI group processing

    Performs processing of the ROI models in the group at the search position where the search model's evaluation value is at its minimum.

    The order of the elements in array1 and array2 determines the correspondence between search models and ROI groups.

    The number of data elements in array1 and array2 equals n.

    Array1(0) = 1st search model (0 to 435)
    Array1(1) = 2nd search model (0 to 435)
    : : :
    Array1(n–1) = nth search model (0 to 435)
    Array2(0) = 1st ROI group (0 to 511)
    Array2(1) = 2nd ROI group (0 to 511)
    : : :
    Array2(n–1) = nth ROI group (0 to 511)

21. Processing mode = 20:
Search group evaluation value min. point → ROI group processing

Performs processing of the ROI models in the group at the search position where the search group's evaluation value is at its minimum.

The order of the elements in array1 and array2 determines the correspondence between search groups and ROI groups.

The number of data elements in array1 and array2 equals n.

Array1(0) = 1st search group (0 to 511)
Array1(1) = 2nd search group (0 to 511)
: : :
Array1(n–1) = nth search group (0 to 511)
Array2(0) = 1st ROI group (0 to 511)
Array2(1) = 2nd ROI group (0 to 511)
: : :
Array2(n–1) = nth ROI group (0 to 511)

22. Processing mode = 50: 1-model positioning
(with rotation and angle interpolation)

Determines the position and angle of search models registered in a single search group, compensates the images, and performs ROI processing on all of the ROI models in the ROI group.

Interpolation processing is performed on the angle during rotation angle detection to provide angle detection that is more precise than the registered rotation model's step angle.

When an ROI verification model has been specified, the correlation of the detected candidate points is measured in the ROI verification model and the position with the highest correlation value is set as the ROI reference point.

A precision search is performed to detect the position when an ROI model set for precision search has been specified as the ROI verification model.

Normally, the search models registered in the group are rotation models rotated and registered with the SMROTATE command.

Array1(0) = Search group (0 to 511) used for reference point/angle detection
Array1(1) = ROI group number (0 to 511) or –1 to disable ROI
Array1(2) = ROI verification model (0 to 1023) or –1 to disable verification
Array1(3) = Rotation step angle used during search model registration
Array1(4) = Rotation range
0: All angles (360°) 1: Not all angles

23. Processing mode = 51: 2-model positioning
(with rotation and angle correspondence)

Determines the position of each of the two search groups, sets the midpoint and vector of the two search positions as the ROI processing reference point and inclination, and performs ROI processing on all of the ROI models in the ROI group.

When an ROI verification model has been specified, the correlation of the detected candidate points is measured in the ROI verification model and the position with the highest correlation value is set as the ROI reference point.

A precision search is performed to detect the position when an ROI model set for precision search has been specified as the ROI verification model.

Normally, the search models registered in the group are rotation models rotated and registered with the SMROTATE command.

Array1(0) = Search group (0 to 511) for the first point's position detection
Array1(1) = Search group (0 to 511) for the second point's position detection
Array1(2) = ROI group number (0 to 511)
Array1(3) = 1st point's ROI verification model (0 to 1023)
or –1 to disable verification
Array1(4) = 2nd point's ROI verification model (0 to 1023)
or –1 to disable verification

24. Processing mode = 52: Circular positioning
(without rotation, with ROI verification)

Finds the center of a circular object using 4 search models that detect 4 points on the circle (left, bottom, right, and top), sets that center-point as the ROI reference point, and performs ROI processing on all of the ROI models registered in the ROI group.

When an ROI verification model has been specified, the correlation of the detected candidate points is measured in the ROI verification model and the position with the highest correlation value is set as the ROI reference point.

A precision search is performed to detect the position when an ROI model set for precision search has been specified as the ROI verification model.

Array1(0) = Search model (0 to 435) for left circular edge detection
Array1(1) = Search model (0 to 435) for bottom circular edge detection
Array1(2) = Search model (0 to 435) for right circular edge detection
Array1(3) = Search model (0 to 435) for top circular edge detection
Array1(4) = ROI group number (0 to 511)
Array1(5) = ROI verification model (0 to 1023) for left circular edge
or –1 to disable verification
Array1(6) = ROI verification model (0 to 1023) for bottom circular edge
or –1 to disable verification
Array1(7) = ROI verification model (0 to 1023) for right circular edge
or –1 to disable verification
Array1(8) = ROI verification model (0 to 1023) for top circular edge
or –1 to disable verification

25. Processing mode = 53: Circular positioning
(with rotation, angle detection from 1 model, and ROI verification)

Finds the center of a circular object using 4 search models that detect 4 points on the circle (left, bottom, right, and top) and finds the angle of the circular object using another search group. Sets the center-point as the ROI reference point and performs ROI processing on all of the ROI models registered in the ROI group.

Interpolation processing is performed on the angle during rotation angle detection to provide angle detection that is more precise than the registered rotation model's step angle.

When an ROI verification model has been specified, the correlation of the detected candidate points is measured in the ROI verification model and the position with the highest correlation value is set as the ROI reference point.

A precision search is performed to detect the position when an ROI model set for precision search has been specified as the ROI verification model.

Array1(0) = Search model (0 to 435) for left circular edge detection
Array1(1) = Search model (0 to 435) for bottom circular edge detection
Array1(2) = Search model (0 to 435) for right circular edge detection
Array1(3) = Search model (0 to 435) for top circular edge detection
Array1(4) = Search group (0 to 511) for angle detection
Array1(5) = ROI group number (0 to 511)
Array1(6) = ROI verification model (0 to 1023) for left circular edge
or –1 to disable verification
Array1(7) = ROI verification model (0 to 1023) for bottom circular edge
or –1 to disable verification
Array1(8) = ROI verification model (0 to 1023) for right circular edge
or –1 to disable verification
Array1(9) = ROI verification model (0 to 1023) for top circular edge
or –1 to disable verification
Array1(10) = ROI verification model (0 to 1023) for angle
or –1 to disable verification

26. Processing mode = 54: Circular positioning
(with rotation, angle detection from the center and a point on the circle, and ROI verification)

Finds the center of a circular object using 4 search models that detect 4 points on the circle (left, bottom, right, and top). Position compensation is executed by detecting the angle between the line joining the circle's center coordinates and a point on the circumference of the circle and the line joining a point determined for one search group and the circle's center coordinate. After an image memory scroll for position compensation, ROI processing is performed on all of the ROI models registered in the specified ROI group.

When an ROI verification model has been specified, the correlation of the detected candidate points is measured in the ROI verification model and the position with the highest correlation value is set as the ROI reference point.

A precision search is performed to detect the position when an ROI model set for precision search has been specified as the ROI verification model.

Array1(0) = Search model (0 to 435) for left circular edge detection
Array1(1) = Search model (0 to 435) for bottom circular edge detection
Array1(2) = Search model (0 to 435) for right circular edge detection
Array1(3) = Search model (0 to 435) for top circular edge detection
Array1(4) = Search group (0 to 511) for angle detection
Array1(5) = ROI group number (0 to 511)
Array1(6) = ROI verification model (0 to 1023) for left circular edge
or –1 to disable verification
Array1(7) = ROI verification model (0 to 1023) for bottom circular edge
or –1 to disable verification
Array1(8) = ROI verification model (0 to 1023) for right circular edge
or –1 to disable verification
Array1(9) = ROI verification model (0 to 1023) for top circular edge
or –1 to disable verification
Array1(10) = ROI verification model (0 to 1023) for angle
or –1 to disable verification

27. Processing mode = 55: Circular positioning
(with defect rotation and ROI verification)

Finds the center of a circular object using 4 search models that detect 4 points on the circle (left, bottom, right, and top). Detects the angle for position compensation from that center point and the position of a defect detected with an ROI model for angle detection. After an image memory scroll for position compensation, ROI processing is performed on all of the ROI models registered in the specified ROI group.

It is assumed that the RMMODE2 command has been used to make the circle/arc defect inspection settings on the ROI model for angle detection. Improper operation may occur when the settings are made with another mode.

When an ROI verification model has been specified, the correlation of the detected candidate points is measured in the ROI verification model and the position with the highest correlation value is set as the ROI reference point.

A precision search is performed to detect the position when an ROI model set for precision search has been specified as the ROI verification model.

Array1(0) = Search model (0 to 435) for left circular edge detection
Array1(1) = Search model (0 to 435) for bottom circular edge detection
Array1(2) = Search model (0 to 435) for right circular edge detection
Array1(3) = Search model (0 to 435) for top circular edge detection
Array1(4) = ROI model (0 to 1023) for angle detection
Array1(5) = ROI group number (0 to 511)
Array1(6) = ROI verification model (0 to 1023) for left circular edge
or –1 to disable verification
Array1(7) = ROI verification model (0 to 1023) for bottom circular edge
or –1 to disable verification
Array1(8) = ROI verification model (0 to 1023) for right circular edge
or –1 to disable verification
Array1(9) = ROI verification model (0 to 1023) for top circular edge
or –1 to disable verification
Array1(10) = Reference angle (–180° to 180°)

SQPDATA SeQuential Position DATA

(Function)

Action Reads sequential processing of position compensation data.

Format SQPDATA (*unit#*, *data type*)

Description Reads the ROI processing reference point position information registered when the processing mode was specified with the SQMODE command.

Specify the unit number (0 to 15) of the desired Measurement Unit with the *unit#* parameter.

The *data type* parameter determines what type of data is read, as shown in the following table.

Data type		Description	Returned value
0	Search position X coordinate	Search result (X) of the search model used to find the reference position	0 to 511
1	Search position Y coordinate	Search result (Y) of the search model used to find the reference position	0 to 483
2	Search position angle	Search result (angle) of the search model used to find the reference position	–180° to 180°
3	X displacement	Difference between the X coordinate of the reference position registered with SQMODE and the search position X coordinate	–512 to 511

Data type		Description	Returned value
4	Y displacement	Difference between the Y coordinate of the reference position registered with SQMODE and the search position Y coordinate	–484 to 483
5	Angle displacement	Difference between the reference angle registered with SQMODE and the search angle	–180° to 180°
6	Reference position X coordinate	X coordinate of the reference position registered with SQMODE	0 to 511
7	Reference position Y coordinate	Y coordinate of the reference position registered with SQMODE	0 to 483
8	Reference position angle	Reference angle registered with SQMODE	–180° to 180°
9	Position compensation judgement result	Judgement result for position compensation	0: OK –1: NG

The method used to calculate the search position depends on the processing mode set with the SQMODE command, as shown in the following list.

1, 2, 3... 1. Processing mode = 0: No position compensation

Search position X coordinate = 0
Search position Y coordinate = 0
Search position angle = 0

2. Processing mode = 1: 1-model positioning (no rotation)

Search position X coordinate = search model's search position X coordinate
Search position Y coordinate = search model's search position Y coordinate
Search position angle = 0

3. Processing mode = 2: 1-model positioning (with rotation)

Search position X coordinate = search model's search position X coordinate
Search position Y coordinate = search model's search position Y coordinate
Search position angle = search model's search angle

4. Processing mode = 3: 2-model positioning (no rotation)

Search position X coordinate = The midpoint between the 2 search models' search position X coordinates
Search position Y coordinate = The midpoint between the 2 search models' search position Y coordinates
Search position angle = 0

5. Processing mode = 4: 2-model positioning (with rotation)

Search position X coordinate = The midpoint between the 2 search models' search position X coordinates
Search position Y coordinate = The midpoint between the 2 search models' search position Y coordinates
Search position angle = The inclination of the two search models' search positions

6. Processing mode = 5: 2-model positioning 2 (no rotation)

Search position X coordinate = The midpoint between the 2 search models' search position X coordinates
Search position Y coordinate = The midpoint between the 2 search models' search position Y coordinates
Search position angle = 0

7. Processing mode = 6: 2-model positioning 2 (with rotation)

Search position X coordinate = The midpoint between the 2 search models' search position X coordinates
Search position Y coordinate = The midpoint between the 2 search models' search position Y coordinates
Search position angle = The inclination of the two search models' search positions

8. Processing mode = 7: Circular positioning (no rotation)
   Search position X coordinate = The midpoint between the search position X coordinates of the left and right search models
   Search position Y coordinate = The midpoint between the search position Y coordinates of the top and bottom search models
   Search position angle = 0
9. Processing mode = 8: Circular positioning (with rotation)
   Search position X coordinate = The midpoint between the search position X coordinates of the left and right search models
   Search position Y coordinate = The midpoint between the search position Y coordinates of the top and bottom search models
   Search position angle = Search angle of the search group

SQR SQuare Root

(Function)

Action Determines the square-root of a numeric expression.

Format SQR (*numeric expression*)

Description The SQR function determines the square-root of the specified *numeric expression*. The *numeric expression* must be 0 or a positive value.

The *numeric expression* may be in the form of an integer, long integer, or a single or double-precision real number. The SQR function always returns a double-precision real number.

SQRUN SeQuential RUN

(Command)

Action Executes sequential processing.

Format SQRUN *trigger mode* [, *start unit#* [, [*end unit#*] [, *busy control*]]]

Description Executes sequential processing with the conditions set with the SQSET and SQMODE commands.

The *trigger mode* parameter determines when sequential processing is started, as shown in the following table.

Trigger mode		Processing
0	Stop sequential processing	Stops continuous measurement and prevents the start of sequential processing when the STEP signal is input.
1	Single measurement (with strobe flash)	The start Unit through the end Unit make just one measurement.
2	STEP synchronization	Sequential measurement is performed each time the step signal is input, not synchronized to program execution.
3	Continuous measurement	Sequential measurements are repeated continuously until SQRUN is executed with the trigger mode set to 0.
4	Single measurement (with strobe flash)	The start Unit through the end Unit make just one measurement.

Specify the first Measurement Unit (0 to 15) with the *start unit#* parameter. The default setting is 0 if this parameter is omitted.

Specify the last Measurement Unit (0 to 15) with the *end unit#* parameter. If the end unit number is omitted, SQRUN will act on the start Unit only.

The *busy control* parameter specifies whether to turn ON the BUSY signal while sequential processing is being performed.

0: Don't control the BUSY signal.
Other than 0: Turn BUSY signal ON while STEP signal reception is disabled.

When busy control is ON, an error won't occur even if a Terminal Block Unit or Parallel I/O Unit isn't connected.

Once sequential processing is started, its processing is performed internally within the system, so other OVL commands can be executed.

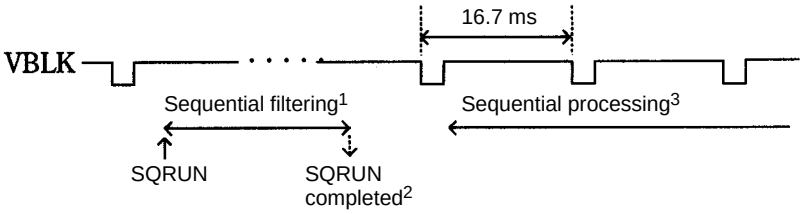
Measurements won't be taken by Measurement Units that have been masked with the SQSET command, even if they are between the start Unit and end Unit.

When strobe flash has been specified with the SQSET command, the strobe flash mode specified with the FLASH command is invalid.

It isn't necessary to stop measurement when changing the trigger mode with the SQRUN command, but all 4 fields are required in the switching process when switching to a different trigger mode.

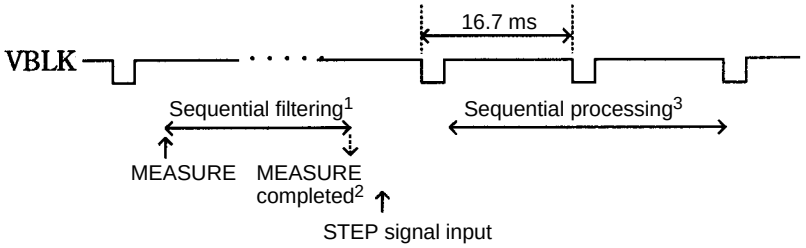
Strobe flash conditions become valid when executed after the FLASH command and SQRUN commands.

The following diagram shows the timing of operation when the trigger mode is set to "1: Single measurement".



Note
1. Several ms are required for the SQRUN command's filtering.
2. The SQRUN command ends when its filtering is completed.
3. Sequential processing is started from the next VBLK after the SQRUN command is completed. The time required for sequential processing depends on the conditions set with the SQSET command.

The following diagram shows the timing of operation when the trigger mode is set to "2: STEP synchronization".



Note
1. Several ms are required for the SQRUN command's filtering.
2. The SQRUN command ends when its filtering is completed.
3. Sequential processing is started from the next VBLK after the SQRUN command is completed and the STEP signal is input. The time required for sequential processing depends on the conditions set with the SQSET command.

The ON SQMEAS GOSUB, SQSET, SQINIT, SQMODE, and SQTIME commands can't be used while sequential processing is being executed. These commands can be executed after stopping sequential processing by executing "SQRUN 0".

SQSET SeQuential SET

(Command)

Action Sets the conditions for sequential processing.

Format SQSET *unit#*[, [*condition 1*] [, [*condition 2*] [, *condition 3*]]]

Description Registers the measurement conditions for the specified Measurement Unit.

Specify the unit number (0 to 15) of the desired Measurement Unit with the *unit#* parameter.

The settings for the *condition 1*, *condition 2*, and *condition 3* parameters are described in detail below.

The sequential processing conditions are set to their defaults when OVL is started, so specify only those conditions that you want to change from their defaults. The default settings will be used in sequential processing unless the settings are changed with SQSET.

Only the settings specified with the *condition 1*, *condition 2*, and *condition 3* parameters will overwrite the system's internal table, so previously changed settings will remain unchanged unless they are specified again. Use the SQINIT command to initialize the contents of the entire table.

When there is an error in the conditions, the location where the error was found will be indicated by the "^" character and operation will stop.

The *condition 1*, *condition 2*, and *condition 3* parameters control the following categories:

Condition 1: Search conditions and measurement timing conditions
Condition 2: Display conditions
Condition 3: Binary raster measurement conditions

Condition 1: The *condition 1* parameters have the following functions. The default setting is indicated by an asterisk.

Parameter		Setting	Function
@Cx	ROI processing adjustment width X (See note.)	x=0*	ROI processing at the fixed position only
		x> 0	Set the ROI processing result to the result for the position with the maximum measurement feature within ±x pixels in the X direction. Together, @Cx and @Ry specify a two dimensional range (±x, ±y).
Fx	Strobe flash mode	x=0*	Strobe doesn't flash when the Unit is started.
		x≠0	For Camera Units without a simultaneous imaging function that flashes the strobe when the Unit is started.
@Px	Pipeline specification	x=0*	Non-pipeline processing, with break The next Measurement Unit's processing won't be performed until this Unit's processing is all completed. The break will be applied if an ON SQMEAS GOSUB subroutine is registered.
		x=1	Non-pipeline processing, without break The next Measurement Unit's processing won't be performed until this Unit's processing (except the ON SQMEAS GOSUB subroutine) is completed. The next Measurement Unit's processing will be started without waiting for completion of the ON SQMEAS GOSUB subroutine's processing.
		x=2	Pipeline processing The next Measurement Unit's raster processing is performed while this Unit's ROI processing is being performed.

Parameter		Setting	Function
Mx [, y]	Measurement mode x: type of processing y: frame/field mode	x=0*	No measurement
		x=1	Binary raster measurement
		x=2	Search
		y=0*	Frame mode (512×512)
		y≠0	Field mode (512×256)
Sx [, y]	Search group number x: start group# y: end group#	0* to 511	The default settings for both x and y are 0. If the end group# is omitted, processing is performed on the start group only. The search models must be sorted by group in advance using the SMSORT command.
@Sx	Search data sorting condition	x=0	No sorting (order of appearance)
		x=1	Sort each model by evaluation value.
		x=2	Sort each model by X search position (descending order).
		x=3	Sort each model by X search position (ascending order).
		x=4	Sort each model by Y search position (descending order).
		x=5	Sort each model by Y search position (ascending order).
		x=6	Sort all candidate points in the group by evaluation value.
		x=7	Sort all candidate points in the group by X search position (desc. order).
		x=8	Sort all candidate points in the group by X search position (asc. order).
		x=9	Sort all candidate points in the group by Y search position (desc. order).
		x=10	Sort all candidate points in the group by Y search position (asc. order).
		x=11	Sort all candidate points in the group vertically from the upper-left.
		x=12	Sort all candidate points in the group vertically from the lower-right.
		x=13	Sort all candidate points in the group horizontally from the upper-left.
		x=14	Sort all candidate points in the group horizontally from the lower-right.
V[x[,y[,z]]]	Image input specification x: image input page y: image input path z: frame/field mode	x=0	Input page 0 only. (An image isn't input if x is omitted.)
		x=1	Input page 1 only. (An image isn't input if x is omitted.)
		x=–1	Input all pages. (An image isn't input if x is omitted.)
		y=0*	Page 1: image bus 1 Page 0: image bus 1
		y=1	Page 1: image bus 1 Page 0: image bus 0
		y=2	Page 1: image bus 0 Page 0: image bus 1
		y=3	Page 1: image bus 0 Page 0: image bus 0
		z=0*	Frame mode (512×512)
		z≠0	Field mode (512×256)
Zx [, y]	Image scrolling x:image scroll mode y:image memory page (y is used when x=0.)	x=0*	Start processing with the image returned to its original position.
		x≠0	Start processing in the position as it is, without changing the scrolled status of image memory.
		y=0	Reverts only image memory page 0 to original.
		y=1	Reverts only image memory page 1 to original.
		y=–1*	Reverts all image memory pages to original.
@Mx [, y]	Unit mask specification x:Mask specification y:Execution mode when unmasked (y is used when x=0.)	x=0*	Don't mask the Unit.
		x≠0	Mask the Unit. (Use this setting when the conditions have been set but you don't want to take a measurement.)
		y=0*	Execute ROI processing and measurement sequence interrupt processing after raster processing is performed.
		y≠0	ROI processing and measurement sequence interrupt processing are performed without raster processing only when the processing is being performed earlier in another Unit that has raster processing. ROI processing is performed according to the SQMODE settings and measurement sequence interrupt processing is performed according to the ON SQMEAS GOSUB settings.

Parameter		Setting	Function
@Ry	ROI processing adjustment width Y (See note.)	y=0*	ROI processing at the fixed position only
		y > 0	Set the ROI processing result to the result for the position with the maximum measurement feature within ±y pixels in the Y direction. Together, @Cx and @Ry specify a two dimensional range (±x, ±y).

Note The @Cx and @Ry parameters are valid only when the measurement feature is set to gray correlation or gray correlation judgement.

Condition 2: The *condition 2* parameters have the following functions. The default setting is indicated by an asterisk.

Parameter		Setting	Function
@Sx	Shading memory bank	x=0*	Bank 0
		x=1	Bank 1
Bx [, y]	Image read path x:image input bus y:image memory page# (y is used when x=1.)	x=0*	Camera
		x=1	Image memory (The image input from the camera has priority when the image input specification in condition 1 is from the camera.)
		y=0*	Image memory page 0
		y≠0	Image memory page 1
Cx	Camera switch	x=0* to 7	x specifies the camera number. The default setting is 0.
Fx	Filter selection	x=0*	OFF
		x=1	Weak smoothing
		x=2	Strong smoothing
		x=3	Edge enhancement level 1
		x=4	Edge enhancement level 2
		x=5	Edge enhancement level 3
		x=6	Edge enhancement level 4
		x=7	Edge enhancement level 5
		x=8	Relief
		x=9	Vertical edges (Condition 2 parameter L must be set to L1.)
		x=10	Horizontal edges (Condition 2 parameter L must be set to L1.)
		x=11	Edge extraction
D[,v[,w[,x[,y[,z[,a[,b[,c]]]]]]]] Display mode v: Display image w: Display path x: Image type in window y: Window display ON/OFF z: Binary reverse in window a: Display mode outside window b: Binary plane outside window c: Binary reverse outside window		v=0 to 31*	Bit 4: Character memory image Bit 3: Graphic memory image Bit 2: Window image Bit 1: Paint/pattern matching window image Bit 0: Camera image
		w=0	Gray image: image bus 0 Binary image: image bus 0
		w=1	Gray image: image bus 0 Binary image: image bus 1
		w=2*	Gray image: image bus 1 Binary image: image bus 0
		w=3	Gray image: image bus 1 Binary image: image bus 1
		x bit=0*	Same as image outside of window (See note 1.)
		x bit=1	Binary image (See note 1.)
		y bit=0	Display OFF (See note 2.)
		y bit=1*	Display ON (See note 2.)
		z bit=0*	Normal display (See note 3.)
		z bit=1	Reverse display (See note 3.)
		a=0	Gray image
		a=1	Binary image
		a=2	Mask image
		b	Binary image plane outside of the window
		c=0	Normal display
		c=1	Reverse display of binary image

Parameter		Setting	Function
Lx	Gray LUT (See note 4.)	x=0*	LUT ON (0 to 255 output as is, –256 to –1 output as 0.)
		x=1	LUT ON (0 to 255 output as is, –256 to –1 reversed.)
		x=2	LUT ON (Uses the current value set for the LUT.)
		x=3	LUT OFF
@Ox	Level outside the image memory region	x=0* to 255	Specifies the level outside of the image memory region. The default setting is 0.
<x	Search completed subroutine number	x=0 to 9	Specifies the search completed machine language subroutine number registered with “DEF USRn”.
		x=–1	Deletes the registered machine language subroutine.
>x	ROI completed subroutine number	x=0 to 9	Specifies the ROI processing completed machine language subroutine number registered with “DEF USRn”.

Note
1. Bit specification (in decimal) of the image type.
2. Bit specification (in decimal) of the window planes to be displayed. The default setting is 255 (display of all planes).
3. Bit specification (in decimal) of the window planes to be reversed. The default setting is 0 (normal display of all planes).
4. The condition 2 setting has priority when it is set at the same time as condition 3’s “Binary level for binary raster processing”.

Condition 3: The *condition 3* parameters have the following functions. The default setting is indicated by an asterisk.

Parameter	Setting	Function
M[,v[,w[,x[,y[,z]]]]] Binary raster measurement mode v: Binary image plane number w: Measured pixels x: Window function y: Paint function z: Line art function	v=–1*	All planes
	v=0 to 7	Binary image plane 0 to 7 only
	w=0	Black pixels
	w=1*	White pixels
	x=0	Window OFF
	x=1*	Window ON
	y=0*	Normal measurement
	y=1	Plane 7 functions as the paint window
	y=2	Plane 7 functions as the pattern matching window
	z=0*	Normal measurement
	z=1	Line art (outline) measurement
Ru[,v[,w[,x[,y[,z]]]]] Detailed run length measurement mode u: Binary image plane number v: Measured pixels w: Window function x: Direction y: Noise z: Image slicing	u=0* to 7	Binary image plane 0 to 7 only. Default setting is plane 0 only.
	v=0	Black pixels
	v=1*	White pixels
	w=0	Window OFF
	w=1*	Window ON
	x=0	X direction
	x=1	Y direction
	x=–1*	Stop
	y=0*	No noise reduction number
	y=1	Reduce noise of 1 pixel in the horizontal direction.
	y=2	Reduce noise of 2 pixels in the horizontal direction.
	y=3	Reduce noise of 3 pixels in the horizontal direction.
	z=0*	Image slicing OFF
	z=1	Image slicing ON

Parameter	Setting	Function
Lw,x [,y[,z]] Binary level for binary raster processing w: Plane number x: Binary level lower limit y: Binary level upper limit z: Mode	w=–1*	All planes
	w=0 to 7	Plane 0 to 7 only
	x=0 to 255	Binary level lower limit (The default setting is 128.)
	y=0 to 255*	Binary level upper limit (The default setting is 255.)
	z=0*	Positive direction only
	z=1	Pattern toward positive direction and negative direction

Long characters (2-byte characters) can't be included in the *condition 1*, *condition 2*, and *condition 3* parameters. The maximum length of the character string used to specify *condition 1*, *condition 2*, and *condition 3* at one time is 253 bytes. If 253 bytes isn't sufficient, execute SQSET several times with different settings. The previous settings aren't changed unless they are specified in SQSET.

When image memory is scrolled during position compensation, both image memory 0 and image memory 1 are scrolled.

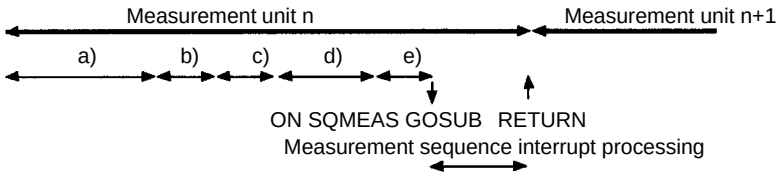
Perform searches on the image in page 0 when performing searches on images in image memory. Reliable search results can't be obtained from searches performed on the image in page 1 while ROI processing is being performed.

The functions related to binary raster processing can't be used in the F350-C12E/C41E.

The following diagrams show the operation timing for the three different pipeline specifications.

1, 2, 3... 1. @P0 (Non-pipeline, with break)

The next Measurement Unit's processing won't be started until the measurement sequence interrupt processing is completed (with RETURN).



a) Search processing, binary raster processing (integer multiple of 16.7 ms)

b) Position compensation (according to the conditions set with SQMODE)

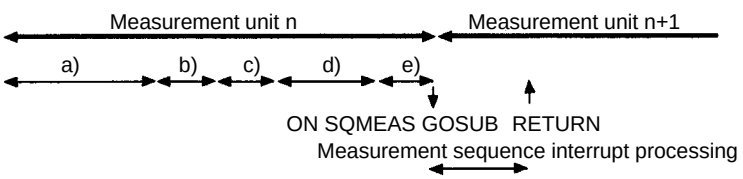
c) The "search processing completed machine language subroutine" specified with SQSET

d) ROI processing (according to the conditions set with SQMODE)

e) The "ROI processing completed machine language subroutine" specified with SQSET

2. @P1 (Non-pipeline, without break)

The next Measurement Unit's processing is started without waiting for completion of the measurement sequence interrupt. Proper timing is necessary when commands or functions related to raster measurement are executed during measurement sequence interrupt processing. The time required for measurement sequence interrupt processing must be shorter than the next Unit's processing time.



a) Search processing, binary raster processing
(integer multiple of 16.7 ms)

b) Position compensation (according to the conditions set with SQMODE)

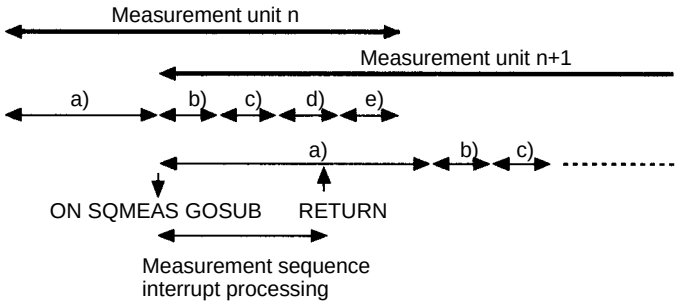
c) The “search processing completed machine language subroutine” specified with SQSET

d) ROI processing (according to the conditions set with SQMODE)

e) The “ROI processing completed machine language subroutine” specified with SQSET

3. @P2 (Pipeline processing)

When raster processing is completed, the next Measurement Unit’s raster processing is started while the first Unit’s ROI processing is being performed. The ROI processing must end by the time that the next Unit’s raster processing is completed.



a) Search processing, binary raster processing
(integer multiple of 16.7 ms)

b) Position compensation (according to the conditions set with SQMODE)

c) The “search processing completed machine language subroutine” specified with SQSET

d) ROI processing (according to the conditions set with SQMODE)

e) The “ROI processing completed machine language subroutine” specified with SQSET

SQSETLUT SeQuence SET LUT

(Command)

Action Sets the LUT used for sequential processing.

Format SQSETLUT *unit#*, *array*

Description Specifies the data that will be set in LUT when the specified Measurement Unit’s raster processing is started.

Specify the unit number (0 to 15) of the desired Measurement Unit with the *unit#* parameter.

Specify the name of the one-dimensional array containing the data that will be set in LUT with the *array* parameter. The array must have at least 512 elements.

The data set with this command is the same as the data set with the L parameter in SQSET’s *condition 2* and *condition 3*. The most recent data will be used whether it is written by SQSETLUT or SQSET.

SQSTAT SeQuence STATus

(Function)

Action Reads the status of sequential processing, i.e., whether sequential processing is being executed or not.

Format SQSTAT (*mode*)

Description Indicates whether sequential processing is being executed or not.

Specify 0 or 1 for the *mode#* parameter. This parameter indicates how to handle ON SQMEAS GOSUB processing that is in progress.

0: Treat ON SQMEAS GOSUB processing as “Not being executed.”
1: Treat ON SQMEAS GOSUB processing as “Being executed.”

A return value of 0 indicates that sequential processing is not being executed and a return value of –1 indicates that sequential processing is being executed.

0: Not being executed
–1: Being executed

The STEP standby status that occurs during STEP synchronization measurements is treated as execution of sequential processing.

The execution of ON SQMEAS GOSUB processing that occurs after a false interrupt is treated as execution of sequential processing.

SQTIME SeQuential TIME

(Function)

Action Reads the time taken for sequential processing.

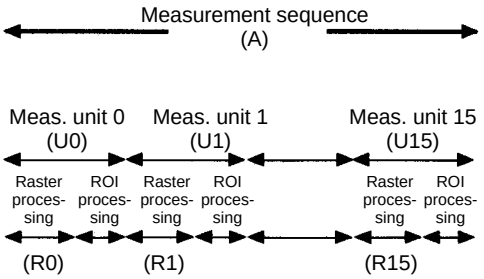
Format SQTIME (*unit#*, *data type* [, *mode*])

Description Reads the amount of time that was taken for sequential processing. Use this command to estimate the measurement’s processing time.

Specify the unit number (0 to 15) of the desired Measurement Unit with the *unit#* parameter. Any value can be set for the *unit#* parameter when the *data type* is set to 0, 1, or 2.

Set the *data type* parameter to one of the following values.

0: Total execution processing time
1: Sequential processing time (most recent)
2: Sequential processing time (max.)
3: Raster processing time (most recent)
4: Raster processing time (max.)
5: Measurement Unit processing time (most recent)
6: Measurement Unit processing time (max.)



Sequential processing time = A = U0 + U1 + ... + U15
Raster processing time = Rn
Measurement Unit processing time = Un

Data type		Meaning
0	Total execution processing time	Total time from the start to the end of sequential processing
1	Sequential processing time (most recent)	Processing time for the last executed tact
2	Sequential processing time (max.)	Longest processing time for a tact from the start to the end of sequential processing
3	Raster processing time (most recent)	Last raster processing time in the specified Measurement Unit
4	Raster processing time (max.)	Longest raster processing time in the specified Measurement Unit from the start to the end of sequential processing
5	Meas. Unit processing time (most recent)	Last processing time in the specified Measurement Unit
6	Meas. Unit processing time (max.)	Longest processing time in the specified Measurement Unit from the start to the end of sequential processing

The phrase “from the start to the end of sequential processing” has the following meanings:

1, 2, 3...

1. 1 tact when the SQRUN command was executed by a one-shot trigger.
2. From execution of the SQRUN command with STEP synchronization until processing is stopped by execution of a “SQRUN 0” command.
3. From execution of the SQRUN command with continuous measurement until processing is stopped by execution of a “SQRUN 0” command.

The *mode* parameter indicates whether or not the measurement sequence interrupt processing time is included. The default setting is 0.

0: Don’t include the measurement sequence interrupt processing time.
1: Include the measurement sequence interrupt processing time.

The various processing times do not include the time from the execution of the SQRUN command until the start of the first Measurement Unit’s raster processing.

The processing time is returned in ms units.

SQUNITNO SeQuence UNIT NO

(Function)

Action Reads the unit number of the Measurement Unit for which measurement sequence interrupt processing is being executed.

Format SQUNITNO

Description The SQUNITNO function returns the unit number (0 to 15) of the Measurement Unit for which measurement sequence interrupt processing is being executed.

A value of –1 will be returned when measurement sequence interrupt processing is not being executed.

SSCROLL Shading memory SCROLL

(Command)

Action Scrolls the shading master memory.

Format SSCROLL *X, Y* [*, center of rotation X, center of rotation Y, angle of rotation*]

Description The SSCROLL command scrolls the shading master memory in the X, Y, and θ directions.

Specify the amount of scroll in the X direction with the *X* parameter and the amount of scroll in the Y direction with the *Y* parameter. Specify the scroll amounts as a number of pixels between –1024 and +1023.

Specify the amount of rotational scroll in degrees with the *angle of rotation* parameter and the coordinates of the center of rotation with *center of rotation X* and *center of rotation Y*. No rotation occurs if these parameters are omitted. Specify the *center of rotation X* and *center of rotation Y* as a number of pixels between –1024 and +1023.

Scrolling is completed in the order: rotation, X, Y.

Shading memory isn't installed in the F350-C12E/C41E.

STOP STOP

(Command)

Action Stops program execution.

Format STOP

Description The STOP command stops program execution. It can be used anywhere inside the program.

STOP ON/OFF/STOP STOP key ON/OFF/STOP

(Command)

Action Disables, enables, or stops interrupts from the STOP Key.

Format STOP ON *or* OFF *or* STOP

Description The STOP command controls branching to an interrupt processing routine when the STOP Key is pressed.

STOP ON: The STOP ON statement enables the interrupt processing routine when the STOP Key is pressed. When the STOP Key is pressed, operation branches to the interrupt processing routine at the line# or label defined with the ON STOP GOSUB statement.

STOP OFF: The STOP OFF statement disables the interrupt processing routine when the STOP Key is pressed. When the STOP Key is pressed, program operation stops but does not branch to an interrupt processing routine.

STOP STOP: The STOP STOP statement stops interrupt processing when the STOP Key is pressed. When the STOP Key is pressed, operation does not immediately branch to the interrupt processing routine. However, immediately branching is enabled by the STOP ON statement, operation branches to the interrupt processing routine at the line# or label defined with the ON STOP GOSUB statement.

Note that if the STOP OFF/STOP command is used the STOP Key does not function as normal to stop program execution.

The interrupt subroutine must be defined with the ON STOP GOSUB statement before a STOP ON/OFF/STOP command is executed.

STR$ STRing $

(Function)

Action Converts a number to character string representation.

Format STR$ (*numeric expression*)

Description The STR$ function converts the numeric expression to a character string. A *numeric expression* cannot be directly assigned to a character variable. First convert the *numeric expression* with the STR$ function before assigning it to a character variable.

The first character in the character string is a space if the *numeric expression* is positive. The space is replaced by a minus sign if the numeric expression is negative.

The VAL function has the opposite action to the STR$ function.

STRCHK STRobe CHecK

(Function)

Action Checks for incorrect strobe flashing.

Format STRCHK

Description The STRCHK function checks for incorrect strobe flashing. The function returns a number as follows:

0: Normal strobe flash
Other than 0: No strobe flash or disconnected strobe cable.

The bit position corresponds to the strobe number. For example, a returned value of 2 indicates that an incorrect strobe flash occurred in strobe number 2; a returned value of 128 indicates that an incorrect strobe flash occurred in strobe number 7.

STRING$ STRING $

(Function)

Action Creates a character string containing the specified number of the specified character.

Format STRING$ (*number of characters, character string* ***or*** *numeric expression*)

Description The STRING$ function returns a character string containing the number of specified 1-byte characters specified by the numeric expression.

Specify the *number of characters* as a value between 1 and 255.

Specify the character to fill the character string with the *character string* or *numeric expression* parameter.

The SPACE$ function is similar to the STRING$ function.

Specifying a *character string*:

Specify the character string as a single 1-byte character.

Only the leading character is valid if more than one character is specified. The STRING$ function returns a character string filled with the leading character.

Specifying a *numeric expression*:

Specify the numeric expression as the decimal character code for a 1-byte character as an integer between 0 and 255.

STRMODE STRobe MODE

(Command)

Action Enables and disables strobe flashing on the FLASH command.

Format STRMODE *strobe#, switch*

Description The STRMODE command enables and disables strobe flashing when a command or function that controls strobe flashes (such as the FLASH command) is executed.

Specify the number of the strobe as a value between 0 and 7 with the *strobe#* parameter. Set *strobe#* to –1 to specify all the strobes.

The *switch* parameter determines whether or not the specified strobe will flash.

Switch setting	Strobe operation
0	The specified strobe won't flash when a strobe-controlling command is executed. (The specified strobe is masked.)
1	The specified strobe will flash when a strobe-controlling command is executed.

All strobes enabled with the STRMODE command will flash simultaneously.

SUB–END SUB SUB–END SUB

(Command)

Action Defines a structural subroutine called by the CALL command.

Format SUB *label* (*argument* [, *argument...*])

Statements in SUB *block*

END SUB

Description The block between SUB–END SUB defines the structural subroutine called by the CALL command.

All variables used in a subroutine are treated as local variables. The variables specified by the *arguments* are assigned the values of the local variables after execution of the subroutine.

The *label* defines the name of the subroutine. This name is used to call the subroutine with the CALL command.

The *arguments* can be specified as any type of variables, except array variables. No logical limitation is placed on the number of arguments. However, the command line can physically accommodate up to 255 characters only.

No further SUB–END SUB command may be nested inside a subroutine block.

Statements in subprogram blocks must be located after the main program.

SWAP SWAP

(Command)

Action Switches two variables.

Format SWAP *variable 1, variable 2*

Description The SWAP command switches the contents of *variable 1* and *variable 2*. Both variables must be of the same type.

SYSINIT SYStem INITialize

(Command)

Action Initializes the OVL system.

Format SYSINIT *mode*

Description Initializes the OVL system, returning it to the same status that it was in when the OVL system was started.

Executing SYSINIT with the *mode* parameter is set to 1 is the same as executing the following OVL program:

```
CAMERA 0
CAMMODE 0
FILTERIN 0
FILTDATA 1,0,0,0,0,0,0,0,1,0
FILTER 0,1,0,0,0
FOR I=0 TO 255
   LUT0(I)=I
   LUT0(511–I)=I
NEXT
SETLUT LUT0,0,512
DISPLAY 16,2
```

```
FOR I=0 TO 255
   LUT0(I)=(I*160)/256
   LUT1(I)=LUT0(I)+32
NEXT
SETDLUT 0,LUT0
SETDLUT 1,LUT1
FOR I=0 TO 6
   SETDLVL I
NEXT
SETDLVL 7,0,0
SETDLVL 7,0,1
BACKDISP 0,0,0
WDISP –1,0,1,0
STRMODE –1,0
FLASH 0
FMODE 0
WZOOM 1,0,0
SZOOM 1,0,0
VZOOM 1,0,0,–1
SBANK 0
MMODE –1,1,1,0,0,0
RMODE 0,1,0,0,0,0
LSTYLE 0
```

Executing SYSINIT with the *mode* parameter is set to 2 is the same as executing the following OVL program:

```
CLS 0,0
CLS 0,1
CLS 1,0
CLS 1,1
CLS 2
CLS 3,0
CLS 3,1
SBANK 1:CLS 4
SBANK 0:CLS 4
MASKBIT 2,0,0
MASKBIT 3,0,0
MASKBIT 3,1,0
MASKBIT 4,0,0
```

Executing SYSINIT with the *mode* parameter is set to 3 initializes the search model, ROI model, search processing results, ROI processing results, and sequence measurement conditions.

Executing SYSINIT with the *mode* parameter is set to 4 initializes the definitions related to option functions and commands.

Executing SYSINIT with the *mode* parameter is set to 0 performs all of the processes of modes 1, 2, and 3.

SZOOM — Shading memory ZOOM

(Command)

Action Zooms (enlarges or reduces) the shading memory.

Format SZOOM [*zoom factor* [, *X reference coordinate*, *Y reference coordinate*]]

Description The SZOOM command zooms (enlarges or reduces) the shading memory.

The *zoom factor* parameter specifies the factor by which the shading memory will be magnified (0.25 to 512.00). The default zoom factor is 1.

The X and Y reference coordinates define the point of reference for zooming. The default point is (0,0).

Shading memory isn't installed in the F350-C12E/C41E.

TAB TABulate

(Function)

Action Specifies the position to display characters.

Format TAB (*numeric expression*)

Description The TAB function moves the cursor along a line by the specified number of characters from the left-hand position. If the character position specified by the *numeric expression* is less than the current cursor position, the cursor moves to the specified character position on the next line.

The TAB function cannot be used alone. Use it together with the PRINT command.

A negative *numeric expression* is treated as 0. If a positive *numeric expression* is specified which exceeds the number of character in a line, the number of characters in a line is subtracted from the specified *numeric expression* to determine the cursor position.

TAN TANgent

(Function)

Action Determines the tangent of a numeric expression.

Format TAN (*numeric expression*)

Description The TAN function determines the tangent.

The *numeric expression* must be set in radians. The *numeric expression* may be in the form of an integer, long integer, or a single or double-precision real number.

THIN THIN

(Command)

Action Reduces line thickness for a binary image in image memory.

Format THIN [*page#*], *binary image plane#* [, [*X1*] [, [*Y1*] [, [*X2*] [, [*Y2*] [, *number of connections*]]]]]

Description The THIN command reduces line thickness for binary images in image memory within the rectangular region defined by (X1, Y1) and (X2, Y2).

The *page#* specifies the page number (0 or 1) in image memory. The default page number is 0.

The *binary image plane#* specifies the binary image plane in which the binary image will be processed.

The start point (*X1*, *Y1*) and end point (*X2*, *Y2*) can be input to specify a rectangular region that will be processed. The coordinates can be set in the range 0 to 511. The default points are (0, 0) and (510, 510).

The *link evaluation constant* specifies the number of linkages that will be used in processing. There will be 8 linkages when the *link evaluation constant*=0, and 4 linkages for any setting besides 0. The default value is 0.

TIME$ TIME $

(Function, Command)

Action Displays and sets the time in the internal clock.

Format Format 1: TIME$

Format 2: TIME$*="HH:MM:SS"*

Description Format 1

Reads the time from the F350's clock.

A character string in the format *HH:MM:SS* is returned when the TIME$ function is executed.

Format 2

Sets the F350's clock:

Set the hours between 00 and 23 with 2 characters.

Set the minutes between 00 and 59 with 2 characters.

Set the seconds between 00 and 59 with 2 characters.

The time must be in the format HH:MM:SS. No part can be omitted.

TIME$ ON/OFF/STOP TIME $ ON/OFF/STOP

(Command)

Action Disables, enables, or stops timer interrupts.

Format TIME$ ON *or* OFF *or* STOP

Description The TIME$ command controls branching to an interrupt processing routine due to the timer.

TIME$ ON: The TIME$ ON statement enables the interrupt processing routine on timer operation. When timer set with the ON TIME$ GOSUB command is reached, operation branches to the interrupt processing routine at the line# or label defined with the ON TIME$ GOSUB statement.

TIME$ OFF: The TIME$ OFF statement disables the interrupt processing routine on timer operation. All timer interrupts are ignored.

TIME$ STOP: The TIME$ STOP statement stops interrupt processing on timer operation. When the set time is reached, operation does not immediately branch to the interrupt processing routine. However, immediately branching is enabled by the TIME$ ON statement, operation branches to the interrupt processing routine at the line# or label defined with the ON TIME$ GOSUB statement.

TIMER TIMER

(Function, Command)

Action Reads and sets the 10 ms timer.

Format Format 1: TIMER

Format 2: TIMER = *numeric expression*

Description Format 1

Reads the 10 ms timer PV. The timer PV is 0 when the power is turned on and increases by 1 every 10 ms.

Format 2

Sets the 10 ms timer to the value between 0 and 2147483647. specified with the *numeric expression*.

TROFF TRace OFF

(Command)

Action Exits the Trace mode.

Format TROFF

Description The TROFF command cancels the Trace mode.

The Trace mode is cancelled by the TROFF command only. It is not cancelled when a program is executed, completed, interrupted, or re-executed.

TRON TRace ON

(Command)

Action Enters the Trace mode.

Format TRON

Description The TRON command enters the Trace mode.

When a program is executed in the Trace mode, the line numbers are displayed continuously on the screen as an aid to debugging. The Trace mode is cancelled by the TROFF command only.

UBOUND Upper BOUND

(Function)

Action Determines the upper boundary of an array dimension subscript.

Format UBOUND (*array name* [*, number of dimensions*])

Description The UBOUND function returns the upper boundary of an array dimension subscript.

Specify the name of the array for which the subscript is to be determined with the *array name* parameter.

Specify the number of dimensions of the array with the *number of dimensions* parameter. The default value is 1.

UCASE$ Upper CASE $

(Function)

Action Converts lowercase letters in the character string to uppercase letters.

Format UCASE$ (*character string*)

Description The UCASE$ function converts lowercase letters in the *character string* to uppercase letters. Existing uppercase letters remain unchanged.

VAL VALue

(Function)

Action Converts a number represented as a character string to a number.

Format VAL (*character string*)

Description The VAL function converts the character string to a real number.

The *character string* must be specified as a character variable or character constant starting with +, –, & or a digit between 0 and 9. The VAL function returns 0 if the *character string* does not conform to this format.

If the *character string* contains a character which cannot be converted to a number, the characters before the unconvertable character are converted.

Spaces in the *character string* are ignored and the positions of the spaces are closed in the returned number.

The STR$ function has the opposite action to the VAL function.

VARBACK VARiable BACK up

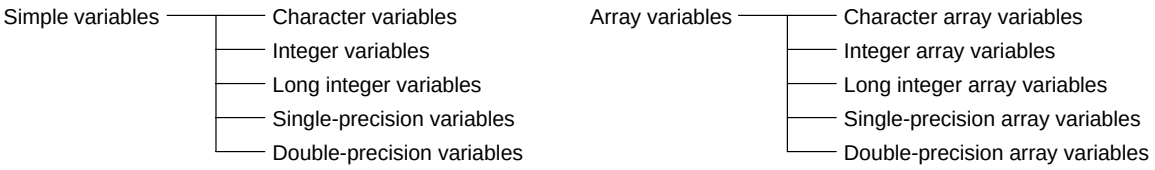
(Command)

Action Specifies the variables to be backed up.

Format VARBACK [*character* [–*character*] [, *character* [–*character*]]]

Description Variables that have been specified for backup will retain their last value even if power is interrupted. A file such as a data file is used to store the set data, but the VARBACK command provides an easy way to back up variables.

Specify the first character of the variables that you want to back up with the *character* parameter. All variables beginning with the specified character will be backed up.



If the *character* parameter is omitted, the backup specifications for all variables will be cleared.

All variables are initialized by the following processes, even if they have been specified for back up.

1, 2, 3...

1. Retrieval of an OVL program
2. Execution of the NEW command
3. Execution of the CLEAR command
4. Execution of the ERASE command (only the specified variables are cleared)
5. Execution of the CHAIN command (although the specified variables aren't cleared when the ALL specification or COMMON specification is used)

An error will occur when an array variable is declared with the DIM command if it has been specified for backup. In this case, execute the CLEAR command or ERASE command beforehand.

VARLOAD VARiable LOAD

(Command)

Action Retrieves variables from the Memory Card.

Format VARLOAD *filename*

Description Retrieves the variable data specified by *filename* and stores the data in the variable area. All variables are stored in the global variable area and variables with the same name will be overwritten.

Array variables won't be retrieved if the array variable has been declared with the DIM command, but the array variable to be retrieved and the maximum array subscript are different.

Only files saved with the VARSAVE command can be retrieved.

The data retrieval will end when the variable area to be retrieved runs out.

The data can't be retrieved if the OPTION BASE value isn't the same as the value used when the data was saved.

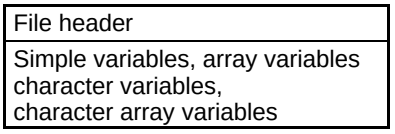
The variable type used when the data was saved is valid over type declarations such as DEFINT.

The VARBACK setting when the command is executed is the valid setting.

When characters are being retrieved, control codes within the character string are treated as characters.

The data retrieval will be stopped if an error is detected in a variable name, variable type, or entity.

The following diagram shows the format of the command:



1, 2, 3...

1. Header
   " VARSAVE"
   Data of option base
2. Simple variables
   `Example:` `ABC%=100`

Item	Value
Number of characters	3
Variable name	ABC
Variable type	Integer
"0"	---
Variable value	100
Delimiter	CR + LF

3. Array variables
   `Example:`
```
DIM DATA! (2)
FOR I=0 TO 2:DATA ! (I)=I/2:NEXT
```

Item	Value
Number of characters	4
Variable name	DATA
Array variable type	Single-precision real type array
"0"	---
Number of dimensions	1
Maximum subscript	2
Entity	0
Entity	0.5
Entity	1
Delimiter	CR + LF

4. Character variables
   `Example:` `CHAR$="ABCDE"`

Item	Value
Number of characters	4
Variable name	CHAR
Variable type	Character
Character string length	5
Character string	ABCDE
Delimiter	CR + LF

5. Character array variables

```
Example:    DIM CHAR$ (2)
            CHAR$ (0) =”Defect, scratch”
            CHAR$ (1) =”Pattern”
            CHAR$ (2) =”Position”
```

Item	Value
Number of characters	4
Variable name	CHAR
Array variable type	Character array
Character string length	---
Number of dimensions	1
Maximum subscript	2
Character string length	14
Character string	Defect, scratch
Character string length	7
Character string	Pattern
Character string length	8
Character string	Position
Delimiter	CR + LF

VARPTR VARiable PoinTeR

(Function)

Action Determines the absolute address where the variable’s value is stored.

Format VARPTR (*variable name*)

Description When the variable is a character variable, VARPTR returns the address that contains the pointer to the region where the character string is stored.

When the variable is an array variable, include the element number in the specification.

VARSAVE VARiable SAVE

(Command)

Action Saves variables to the Memory Card.

Format VARSAVE *filename*, *characters* [–*characters*] [, *characters* [–*characters*] ...]

Description Saves the variables beginning with *character* to the Memory Card. Data can’t be saved to the RS-232C port.

Specify the name of the file in which the variable data is to be stored with the *filename* parameter. Variable names, variable types, and variable values will all be saved. The data is saved in the file in text format.

A single character can be specified or a number of characters can be specified by separating the characters with a –. The characters must be specified in alphabetical order and a character can’t be specified more than once. If there are no variables beginning with a specified characters, just the file header will be saved.

The OPTION BASE status will be saved.

Type declarations by commands such as DEFINT can’t be saved and the VARBACK setting can’t be saved.

Local variables won’t be saved even if VARSAVE is executed from a structured subroutine.

VDWAIT VD WAIT

(Command)

Action Delays the VD interrupt the specified number of times.

Format VDWAIT *number of times*

Description The VDWAIT command counts the number of VD interrupts and delays VD interrupt processing the specified number of VD interrupts.

Use VDWAIT when waiting for the completion of an image input or measurement. For example, when inputting an image in frame mode, at least 2 fields are required, so execute “VDWAIT 3”.

Each VD interrupt is equivalent to 1/60 second (approx. 16.7 ms).

Program operation cannot be interrupted with the STOP Key or CNTRL+C Keys during VDWAIT operation.

VIDEOIN VIDEO IN

(Command)

Action Inputs an image to the image memory.

Format VIDEOIN [*page#*] [, [*input path*] [, [*input mode*] [, *trigger mode*]]]

Description Inputs an image into the IMP Unit’s internal image memory.

The *page#* parameter specifies the page number of the image memory page where the image will be input. The default setting is –1.

0: Input image memory 0 only.
1: Input image memory 1 only.
–1: Input to all of image memory.

The *input path* parameter specifies which bus will be used to input the image. The default setting is 0.

0: Image bus 1 (video bus: gray images)
1: Image bus 0 (pipeline bus: mainly filtered images)

The *input mode* parameter specifies whether the measurement is performed in frame mode or field mode. The mode specified with the FMODE command is used when this parameter is omitted. The mode is set to frame mode when OVL is first started.

The following table shows the differences between frame mode and field mode.

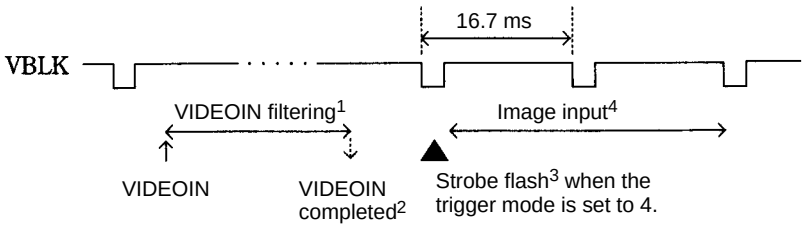
Mode		Resolution	Image input time (excluding wait time)
0	Frame mode	512×512 pixels	33.3 ms
1	Field mode	512×256 pixels	16.7 ms

The *trigger mode* parameter controls the image input, as shown in the following table. The default setting is 1 (No strobe flash).

Trigger mode		Image input timing
0	Stop	Cancels the STEP synchronization setting.
1	No strobe flash	A single image is input when VIDEOIN is executed.
2	STEP synchronization	A strobe flash is emitted in sync with the STEP signal and the image is input to image memory at that time.
3	Reserved	Not used.
4	Single strobe flash	A strobe flash is emitted and a single image is input when VIDEOIN is executed.

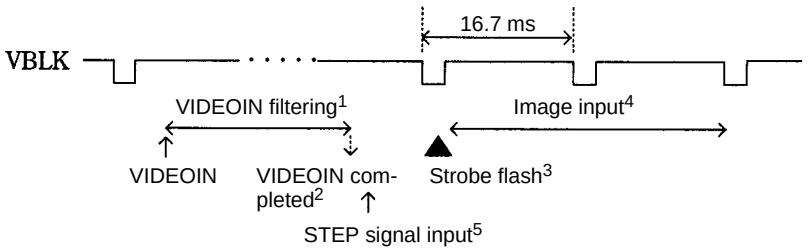
When the trigger mode is set to “2: STEP synchronization”, the image is input automatically when the STEP signal is input. The STEP synchronization setting is cleared once the image has been input with STEP synchronization.

The following diagram shows the timing of operation when the trigger mode is set to “1: No strobe flash” or “4: Single strobe flash”.



Note
1. Several ms are required for the VIDEOIN command’s filtering.
2. The VIDEOIN command ends when it’s filtering is completed.
3. A strobe flash is emitted and the image input is started in the next VBLK after the end of the VIDEOIN command.
4. The time required for the image input depends on the input mode.

The following diagram shows the timing of operation when the trigger mode is set to “2: STEP synchronization”.



Note
1. Several ms are required for the VIDEOIN command’s filtering.
2. The VIDEOIN command ends when it’s filtering is completed.
3. The image input is started in the next VBLK after the VIDEOIN command has ended and the STEP signal has been input.
4. The time required for binary raster measurement + image input depends on the input mode.
5. The STEP signal can be input after the VIDEOIN command has been completed.

VSCROLL Vram SCROLL

(Command)

Action Scrolls the image memory.

Format VSCROLL *X, Y* [, [*center of rotation X*], [*center of rotation Y*], [*angle of rotation*] [, *page#*]]

Description The VSCROLL command scrolls the image memory. Scrolling is performed in the following order: rotation, X, Y.

Specify the amount of scroll in the X direction with the *X* parameter and the amount of scroll in the Y direction with the *Y* parameter. Specify the scroll amounts between –1023 and +1023. The positive *X* direction is right, negative is left. The positive *Y* direction is down, negative is up.

Specify the coordinates of the center of rotation with *center of rotation X* and *center of rotation Y*. The setting range for these parameters is –1023 to +1023. The default center of rotation is (0, 0).

Specify the amount of rotational scroll in degrees with the *angle of rotation* parameter. The positive direction is clockwise, negative is counter-clockwise. The default setting is 0°.

The *page#* parameter specifies the page number of the image memory page where the image will be scrolled. The default setting is –1.

0: Scroll page 0 only.
1: Scroll page 1 only.
–1: Scroll all pages.

When an image is scrolled, the drawing function also operates in the scrolled image memory. (When window memory is scrolled with the WSCROLL command, patterns are drawn in unscrolled window memory.)

Even if image memory is scrolled, images are input without scrolling and image memory returns to its original scrolled status after the image is input.

Scrolling performed with the WSCROLL and SSCROLL commands is executed during the vertical retrace, but scrolling performed with the VSCROLL command is reflected immediately when the command is executed.

The gray level outside of the image region can be specified with the SETDLVL command when image memory has been scrolled.

VZOOM — Vram ZOOM

(Command)

Action Zooms (enlarges or reduces) image memory.

Format VZOOM [*zoom factor* [, [*X reference coordinate*], [*Y reference coordinate*] [, *page#*]]]

Description The VZOOM command zooms (enlarges or reduces) image memory.

The *zoom factor* parameter specifies the factor by which the image memory will be magnified (0.25 to 512.00). The default zoom factor is 1.

The X and Y reference coordinates specify the upper-left corner of the display. The default point is (0, 0). The reference coordinates correspond to the coordinate space after enlargement/reduction, so specify the reference coordinates for a coordinate space of (0, 0) to (1023, 1023) when the image is enlarged by a factor of 2.

The *page#* parameter specifies the page number of the image memory page where the image will be zoomed. The default setting is –1.

0: Zoom page 0 only.
1: Zoom page 1 only.
–1: Zoom all pages.

Enlargement/reduction returns to the origin after execution of VZOOM (the same status resulting from execution of “VSCROLL 0,0”), so be sure to execute the VZOOM command first when using VZOOM and VSCROLL.

When an image is zoomed, the drawing function also operates in the zoomed image memory. (When window memory is zoomed with the WZOOM command, patterns are drawn in unzoomed window memory.)

Even if image memory is zoomed, images are input without zooming and image memory returns to its original zoom status after the image is input.

Zooming performed with the WZOOM and SZOOM commands is executed during the vertical retrace, but zooming performed with the VZOOM command is reflected immediately when the command is executed.

The gray level outside of the image region can be specified with the SETDLVL command when image memory has been zoomed.

WDILA Window DILAte

(Command)

Action Enlarges the binary image in window memory.

Format WDILA *binary image plane#* [, [*X1*] [, [*Y1*] [, [*X2*] [, [*Y2*] [,[*number of connections*] [, *enlargements*]]]]]]

Description The WDILA command enlarges the binary image in window memory within the region defined by (X1, Y1) and (X2, Y2).

The *binary image plane#* specifies the binary image plane in which the binary image will be reduced.

The start point (*X1*, *Y1*) and end point (*X2*, *Y2*) can be input to specify a rectangular region that will be enlarged. The default points are (1, 1) and (510, 510). The coordinates can be set in the range 1 to 510.

Specify the number of connections as shown below. (The default setting is 0.)

0: 8
Other than 0: 4

The *enlargements* setting specifies the number of times that the image will be enlarged. The default value is 1.

When a pixel within the rectangular region has a value of 0 and the adjacent pixel has a value of 1, enlargement processing will replace it with a value of 1.

The specified rectangular region cannot be exceeded in enlargement. Enlargement processing will end before the enlargements setting is reached if it is not possible to enlarge the image further.

WDISP Window DISPlay

(Command)

Action Sets the type of image display in a window.

Format WDISP *window plane#, image type* [*,* [*display or not*] [*, binary reverse*]]

Description The WDISP command sets the type of image display inside the window plane specified by the *window plane#*. Set *window plane#* to –1 to specify all window planes.

Set the *image type* to 0 to display the same image type inside the window as outside the window. Specify 1 to display a binary image in the window.

Specify whether the window itself is displayed with the *display or not* item. Set to 1 to display the window or to 0 to hide the window. The default value is 0. If window display is selected, the window gradations are increased, or made lighter, to display the window.

Set the *binary reverse* parameter to 1 to reverse the displayed image or to 0 for a normal displayed image. The default value is 0. The *binary reverse* setting is valid only when the image type is set to 1 (binary image).

WEROS Window EROSion

(Command)

Action Reduces the binary image in window memory.

Format WEROS *binary image plane#* [, [*X1*] [, [*Y1*] [, [*X2*] [, [*Y2*] [, [*number of connections*] [, *reductions*]]]]]]

Description Reduces the binary image in window memory within the region defined by (X1, Y1) and (X2, Y2).

The *binary image plane#* specifies the binary image plane in which the binary image will be reduced.

The start point (*X1*, *Y1*) and end point (*X2*, *Y2*) can be input to specify a rectangular region that will be reduced. The coordinates can be set in the range 1 to 510. The default points are (1, 1) and (510, 510).

Specify the number of connections as shown below. (The default setting is 0.)

0: 8
Other than 0: 4

The *reductions* setting specifies the number of times that the image will be reduced. The default value is 1.

When a pixel within the rectangular region has a value of 1 and the adjacent pixel has a value of 0, reduction processing will replace it with a value of 0.

The specified rectangular region cannot be exceeded in reduction. Reduction processing will end before the reductions setting is reached if it is not possible to reduce the image further.

WHILE–WEND WHILE–While END

(Command)

Action The commands between the WHILE and WEND statements are executed repeatedly while a condition remains fulfilled.

Format

WHILE *logical expression*
⋮
WEND

Description The commands between the WHILE and WEND statements (WHILE–WEND loop) are executed repeatedly while the logical expression remains true (non-0).

The *logical expression* declared with the WHILE statement provides the condition for the commands to be executed. The WHILE–WEND loop is executed repeatedly while the logical expression remains true (non-0). Control is transferred to the line after the WEND statement when the logical expression becomes false (0).

The WHILE–WEND loop is not executed if the logical expression is initially false (0). Control jumps immediately to the line after the WEND statement. Other WHILE–WEND loops may be nested inside a WHILE–WEND loop. The nested WHILE–WEND loops must have a one-to-one correspondence between the WHILE and WHEN statements from the inside of the nested loops toward the outside. WEND statements must not be omitted.

WRITE WRITE

(Command)

Action Displays data on the display.

Format WRITE *expression* [, ***or*** *; expression*]

Description Displays the numeric data and character data specified with the *expression list* on the text display.

List the numeric and character expressions in the *expression list* delimited by commas (,) or semicolons (;). Commas (,) or semicolons (;) are identical in function. If the expressions in the *expression list* are delimited by commas (,), the data items are displayed on the screen separated by commas (,). A character string is displayed enclosed in double-quotations (“ ”).

If a numeric expression is specified, spaces in front of the data are deleted on the display.

A carriage return character is inserted automatically after all the expressions are displayed.

WRITE# WRITE

(Command)

Action Writes data to a sequential file.

Format WRITE *file#, expression* [, ***or*** *; expression*]

Description Specify the *file#* as the number in which the output file was opened with the OPEN command. Specify the same *file#* with the CLOSE command to close the file after output is complete.

List the numeric and character expressions in the *expression list* delimited by commas (,) or semicolons (;). Commas (,) or semicolons (;) are identical in function. If the expressions in the *expression list* are delimited by commas (,), the data items are written to the file separated by commas (,).

A character string is automatically enclosed in double-quotations (") when written to a file.

If a numeric expression is specified, spaces in front of the data are deleted when the data is written to the file. Consequently, the file area used is less than with the PRINT# command which writes leading spaces to the file.

The WRITE# command automatically inserts a linefeed character (CHR$(10)) after writing the last expression to the file.

WSCROLL Window SCROLL

(Command)

Action Scrolls the window memory.

Format WSCROLL *X, Y* [*, center of rotation X, center of rotation Y, angle of rotation*]

Description The WSCROLL command scrolls the window memory in the *X*, *Y*, and θ directions.

Specify the amount of scroll in the X direction with the *X* parameter and the amount of scroll in the Y direction with the *Y* parameter. Specify the scroll amounts as a number of pixels between –1023 and +1023.

Specify the amount of rotational scroll in degrees with the *angle of rotation* parameter and the coordinates of the *center of rotation* with *center of rotation X* and *center of rotation Y*. No rotation occurs if these parameters are omitted. Specify the *center of rotation X* and *center of rotation Y* as a number of pixels between –1023 and +1023.

Scrolling is completed in the order: rotation, X, Y.

WZOOM Window ZOOM

(Command)

Action Zooms (enlarges or reduces) the window memory.

Format WZOOM [*zoom factor* [, *X reference coordinate*, *Y reference coordinate*]]

Description The WZOOM command zooms (enlarges or reduces) the window memory.

The *zoom factor* parameter specifies the factor by which the window memory will be magnified (0.25 to 512.00). The default zoom factor is 1.

The X and Y reference coordinates define the point of reference for zooming. The default point is (0,0).

When the WZOOM command is executed, WSCROLL 0,0 is executed internally. After the WZOOM command has been executed, the scroll quantities and rotation center specified with the WSCROLL command are specified for the coordinate space after zooming.

Measurements can be made on the zoomed window image (the image displayed on the screen).

SECTION 6
General-purpose Structural Subroutines

This section provides the general-purpose structural subroutines used to simplify programming and reduce processing time.

6-1 Introduction

6-1-1 Differences with Structural Subroutines

OVL uses both structural subroutines and general-purpose structural subroutines. A structural subroutine is a specific function that the user converts to a subroutine and uses in the program. The subroutine is defined with a SUB–END SUB command and local variables can be used in the subroutine.

General-purpose structural subroutines are frequently used functions that OMRON has converted to subroutines and incorporated into the OVL System. The following list indicates some of the ways that general-purpose structural subroutines can improve programming:

1, 2, 3...

1. The subroutines can reduce program development time.
2. Concise programs simplify debugging and maintenance.
3. Processing time can be reduced.
4. Program size can be reduced.

The following table shows the differences between structural subroutines and general-purpose structural subroutines.

Item	Structural subroutine	General-purpose structural subroutine
Provider	User (programmed in OVL)	OMRON (incorporated in the OVL System)
Processing speed	Low-speed	High-speed
Omission of parameters	Not possible	Possible
Array management	Not possible to use arrays in parameters	Arrays can be transferred as parameters
Process contents	Standard OVL functions and commands	Various processing is possible.
Change of process contents	Can be done by changing the OVL program.	Not possible
Break	Breaks are possible anywhere.	Breaks are not possible while the subroutine is being executed.

6-1-2 Basic Operations

Labels

Labels registered as general-purpose structural subroutines cannot be handled as reserved words, so labels defined in a general-purpose structural subroutine can be used as labels or variables within the program.

When there is a label within the program with the same label that is registered for a general-purpose structural subroutine, the label within the program is effective.

Parameters

Except for parts of general-purpose structural subroutines, all types of variables, constants, and expressions can be used as arguments (referred to as parameters here).

When a constant or expression is specified in a parameter, a return value cannot be returned to that parameter. It is possible to omit parameters, but intermediate parameters cannot be omitted; for example, the following diagram shows two acceptable formats and one unacceptable format for the parameters in the command CALL *SUB1(A,B[,C[,D]]).

OK:	CALL *SUB1(A,B)
OK:	CALL *SUB1(A,B,C)
Not allowed:	CALL *SUB1(A,B,,D)

Error Checks

Even if an error occurs somewhere in the general-purpose structural subroutine, the line where the error occurs becomes the call line of the CALL command. It is possible that there is an error in the parameter setting when an “Illegal function call” error occurs when a general-purpose structural subroutine is called.

Refer to *Section 8 Troubleshooting* for more details on errors.

Breaks

Checks for inputs of the STOP Key or Control+C Keys are not performed in general-purpose structural subroutines. When the program is stopped during execution of a general-purpose structural subroutine, be sure to end the subroutine after pressing the STOP Key.

When an external interrupt occurs during processing of a general-purpose structural subroutine, the interrupt will be recorded but operation will not branch to the interrupt subroutine until the general-purpose structural subroutine is completed.

6-1-3 Types of General-purpose Structural Subroutines

General-purpose structural subroutines are broadly divided into 6 libraries according to their functions.

Menu Library

These subroutines provide menu operations equivalent to those used in application software such as F350-U□□□E. These subroutines are useful when creating programs that operate with menus.

Subroutine	Function	Page no.
*MENUINIT	Menu library initial settings processing	297
*MBARDISP	Menu bar display processing	297
*MBARSELECT	Menu bar selection processing	297
*MENUDISP	Selection menu display processing	298
*MENUSELECT	Selection menu selection processing	298
*MBOXDS	Menu box display/selection processing	299
*MBOXDISP	Menu box display	300
*MBOXCLR	Menu box clear processing	301
*DBOXDS	Dialog box display/selection/clear processing	301
*BTNBOXDS	Button box display/selection/clear processing	304
*MSGBOXDISP	Message box display processing	304
*MSGBOXCLR	Message box clear processing	305

Region Setting Library

These subroutines provide operations that use the console or keyboard to set regions such as rectangles or circles. These subroutines are useful when setting regions within images.

Subroutine	Function	Page no
*SETBOX	Rectangle settings	305
*SETCIRCLE	Circle settings	307
*SETCCIRCLE	Concentric circle settings	308
*SETLINE	Line settings	309
*SETCLINE	Continuous line settings	310
*SETPOINT	Single point settings	312
*SETPOLYGON	Polygon settings	313
*SETSPLINE	Independent curve settings	314

Character Display Library

Some new functions have been added to the character display functions. These subroutines are useful for displaying measurement results quickly, making displays easier to read, etc.

Subroutine	Function	Page no.
*BTMMSG	Bottom message display	316
*MAKETABLE	Displays a framework of lines.	316
*FASTPRINT	Displays data quickly.	317
*FASTPRINT2	Displays “OK/NG” quickly.	318
*FASTPRINT3	Displays character strings quickly.	318
*SYMBOL	Displays enlarged characters.	319

Image Control Library

These subroutines combine image processing functions and make them easier to use.

Subroutine	Function	Page no.
*MAKEBRIGHT	Displays the line brightness.	320
*MAKEHIST	Displays a histogram.	321
*DISPMODE	Displays the selected image.	322

File Operations Library

This subroutine manages files such as those in a memory card. It is useful for operations such as inputting or editing file names.

Subroutine	Function	Page no.
*SELECTFILE	File directory display/selection	322

Graphic Display Library

This subroutine provides additional functions for managing graphic displays. It is useful for operations such as displaying rectangles or plus cursors at high speed.

Subroutine	Function	Page no.
*FBOX	Displays rectangles quickly.	325
*FCURSOR	Displays the cursor quickly	326

Other Library

This library contains other frequently used functions.

Subroutine	Function	Page no.
*DATASORT	Data sorting	326
*CALCDATE	Date data conversion	327
*SUBTHETA	Angle compensation	327
*GETTHETA	Line slope calculation	327

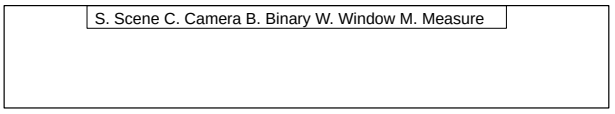
6-2 Menu Library

6-2-1 Types of Menus

Six types of menu operations can be performed with the subroutines in the menu library.

Menu Bar

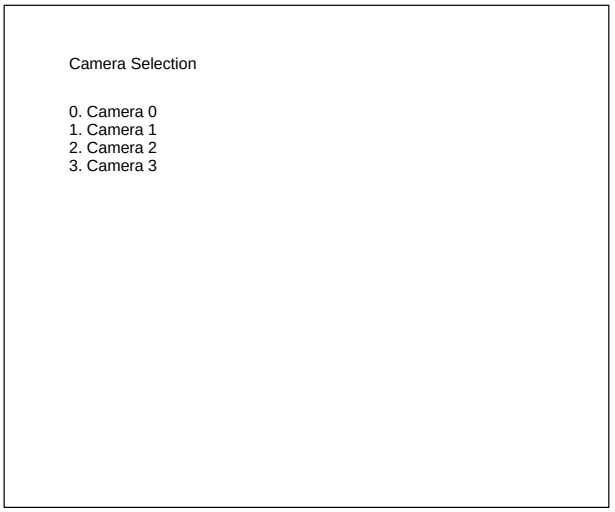
The menu bar is the menu displayed at the top of the screen. The menu bar is used to select the highest-level menus.



Selection Menus

A selection menu bar is a menu displayed at the upper-left corner of the screen. Program control changes when different menu items are selected in the menu. The menu title is optional.

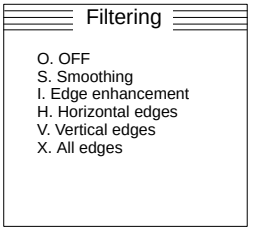
In this case, changing the menu item selects a different camera.



Menu Box

A menu box is a pop-up menu that can be displayed anywhere on the screen. A menu box is used when one item must be selected from several choices. The menu title is optional.

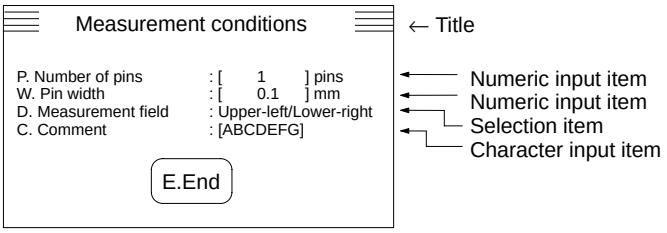
The menu position can be set arbitrarily. The box is normally displayed.



Dialog Box

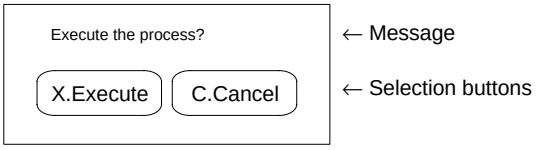
A dialog box is a pop-up menu in which several items can be set. The input item can be selected either by the title or by incrementing/decrementing. The title is optional.

Dialog boxes always appear in the center of the screen.



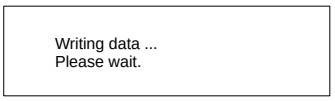
Selection Button Box

A selection button box is a selection menu like the one shown in the following diagram. The message is optional. A selection button box can have from 1 to 12 buttons.



Message Box

A message box is a box that just displays a message. Message boxes always appear in the center of the screen.



6-2-2 Key Operations

Menu operations can be controlled from the console or keyboard. The menu library automatically judges whether a console key or keyboard key has been pressed. Operations cannot be controlled from the RS-232C port.

The following table shows the keys that can be used to control operation.

Console	Keyboard	Function
▲	↑ or Control+E	Move up
▼	↓ or Control+X	Move down
◀	← or Control+D	Move left
▶	→ or Control+S	Move right
ENT	ENT	Select item
ESC	ESC or Control+[	Cancel
HELP	HELP	Not used.
SHIFT	SHIFT ($20) to ($7E)	Used to input an operation's code letter, numerical values, or character strings. (The characters $ and / cannot be input.)

6-2-3 Character and Graphic Memory

Menu displays use character memory and graphic memory. The menu library functions are performed with the procedures described the following table.

Function	Procedure
Displaying characters	Active page display through the OVL System
Character gradation	Not changed in the library, so the SETDVL command's setting is used.
Displaying menu boxes	Drawn using the graphic function in page 0 of character memory.
Reversing characters (highlighting)	Reversed by directly accessing page 0 of character memory.
Clearing the menu region	Cleared using the graphic function in page 0 of character memory.
Displaying the menu background	Drawn using the graphic function in page 0 of character memory.
Menu background gradation	The library overwrites the display gradation in graphic memory.

The desired menu might not be displayed if the active page is different from the display page. The display will not be correct when page 1 is the active page.

The character string is stored in the OVL's code buffer when a menu has been displayed even though the characters are cleared from the screen, so the program might be changed if the Enter Key is pressed in the screen editor. Always clear the code buffer by pressing HOME_CLR.

Depending on the character memory's display gradation settings, characters might not be cleared from the screen.

When a menu has been displayed, it may not be possible to clear the pattern drawn in graphic memory.

6-2-4 Menu Library Subroutines

This section describes the general-purpose structural subroutines in the menu library.

*MENUINIT MENU INITialize

(General-purpose Structural Subroutine)

Action Initializes the menu library.

Format CALL *MENUINIT (*dummy parameter*)

Description Registers the external characters used in the menu library. This subroutine must be executed before using the menu library.

Any value can be specified in the *dummy parameter*, but the parameter cannot be omitted.

The following external character codes are registered:

&HEC5E:	Button display's upper-left roundness
&HEC5F:	Button display's lower-left roundness
&HEC60:	Button display's upper-right roundness
&HEC61:	Button display's lower-right roundness

External characters are registered in character codes &HEC5E through &HEC61, so do not register other codes in this region.

*MBARDISP Menu BAR DISPlay

(General-purpose Structural Subroutine)

Action Displays the menu bar at the top of the screen.

Format CALL *MBARDISP (*message*, *menu#*)

Description Displays the menu bar at the top of the screen.

Specify the menu items to be displayed in the menu bar in the *message* parameter. Separate the menu items by "/" characters. The message can be up to 62 bytes long. The first character in each item specifies the operation's code character, as in the following example.

MSG$="S. Scene/C. Camera/B. Binary/W. Window/M. Measure"

Any code from $20 to $7F (except "$" and "/") can be used for the operation's code character.

A single space will be inserted between the menu items when they are displayed.

Specify the first menu item to be highlighted with the *menu#* parameter. The *menu#* can be set from 0 to n–1, where n is the number of items in the menu.

The menu bar's background is filled in using graphic memory. The gradation used for the background is set with the SETDLVL command.

The *MBARDISP subroutine just displays the menu bar. Use the *MBARSELECT subroutine to select items from the menu bar.

The menu bar is displayed to the right from character coordinate (1,0).

*MBARSELECT Menu BAR SELECT

(General-purpose Structural Subroutine)

Action Selects an item from the menu bar displayed with the *MBARDISP subroutine.

Format CALL *MBARSELECT (*message*, *menu#*)

Description Uses the console or keyboard to select an item from the menu bar displayed with the *MBARDISP subroutine. The following keys are used to make a selection:

Key	Function
→ Key	Moves one menu item to the right.
← Key	Moves one menu item to the left.
Enter Key	Selects the highlighted menu item.
Operation code character	Selects the menu item with the corresponding code character.

Specify the menu items displayed in the menu bar in the *message* parameter. Separate the menu items by "/" characters. The *message* parameter must be the same as the one specified with the *MBARDISP subroutine. The first character in each item specifies the operation's code character, as in the following example.

MSG$="S. Scene/C. Camera/B. Binary/W. Window/M. Measure"

Any code from $20 to $7F (except "$" and "/") can be used for the operation's code character.

A single space will be inserted between the menu items when they are displayed.

Specify the first menu item to be highlighted with the *menu#* parameter. The *menu#* can be set from 0 to n–1, where n is the number of items in the menu. The selected menu item's menu number will be stored in *menu#*.

The menu bar will not be cleared when this subroutine ends.

*MENUDISP MENU DISPlay

(General-purpose Structural Subroutine)

Action Clears the display and then displays a selection menu.

Format CALL *MENUDISP (*title*, *message*, *menu#*)

Description The *MENUDISP subroutine is used together with the *MENUSELECT subroutine. Use this subroutine when some kind of processing is needed along with the selection of the menu item. For example, when the camera is switched along with selection of a menu item in the camera selection menu.

The selection menu is displayed in the upper-left corner of the screen after the entire screen is cleared. When the selection menu is displayed, neither the box nor the graphics are displayed.

Specify the title to be displayed at the top of the menu with the *title* parameter. No title will appear when the null string is specified.

Specify the menu items to be displayed in the menu in the *message* parameter. Separate the menu items by "/" characters. The message can be up to 64 bytes long and can contain up to 23 menu items.

The first character in each menu item specifies the operation's code character. Any code from $20 to $7F (except "$" and "/") can be used for the operation's code character.

The character screen will be cleared. The menu's position depends on the title.

Item	Menu with title	Menu without title
Title	Row 0	---
Menu	From row 2	Row 0

*MENUSELECT MENU SELECT

(General-purpose Structural Subroutine)

Action Selects an item from the menu displayed with the *MENUDISP subroutine.

Format CALL *MENUSELECT (*title*, *message*, *menu#*, *key code*)

Description Uses the console or keyboard to select an item from the menu displayed with the *MENUDISP subroutine. The following keys are used to make a selection:

Key	Function
↑ Key	Moves up one menu item.
↓ Key	Moves down one menu item.
Escape Key	Exits the selection menu without making a selection.

Key	Function
Enter Key	Selects the highlighted menu item.
Operation code character	Selects the menu item with the corresponding code character.

Use this subroutine when some kind of processing is needed along with the selection of the menu item.

Specify the same *message* parameter that was specified in the *MENUDISP subroutine.

Specify the currently highlighted menu item with the *menu#* parameter. The *menu#* can be set from 0 to n–1, where n is the number of items in the menu.

Processing will end if any key is pressed. The selected key code will be returned in the *key code* parameter. The return values are listed in the following table.

Key	Return value
Escape Key	0 (&H00)
Enter Key or operation code character	1 (&H01)
Shift+Escape Keys	128 (&H80)
Shift+Enter Keys	129 (&H81)
Other Key	–1

The selection menu must be displayed in advance with the *MENUDISP subroutine.

The menu will not be cleared even if the Enter Key or Escape Key is pressed.

*MBOXDS Menu BOX Display & Select

(General-purpose Structural Subroutine)

Action Displays a menu box and selects an item from the displayed menu.

Format CALL *MBOXDS (*X*, *Y*, *title*, *message*, *menu#*, *key code*)

Description Performs menu box display and selection processes. The following keys are used to make a selection:

Key	Function
↑ Key	Moves up one menu item.
↓ Key	Moves down one menu item.
Escape Key	Exits the selection menu without making a selection.
Enter Key	Selects the highlighted menu item.
Operation code character	Selects the menu item with the corresponding code character.

Specify the character coordinates of the upper-left corner of the menu box with the *X* and *Y* parameters. The possible setting ranges are as follows:

Coord.	Menu with title	Menu without title
X	2 ≤ X ≤ 62 – Max. item length	2 ≤ X ≤ 62 – Max. item length
Y	0 ≤ Y ≤ 23 – Number of menu items	1 ≤ Y ≤ 24 – Number of menu items

Specify the title to be displayed at the top of the menu with the *title* parameter. The title must be shorter than the menu items. No title will appear when the null string is specified.

Specify the menu items to be displayed in the menu in the *message* parameter. Separate the menu items by "/" characters. The first character in each item specifies the operation's code character, as in the following example.

MSG$="O. OFF/S. Smoothing/1. Edge enhancement/E. All edges"

Any code from $20 to $7F (except "$" and "/") can be used for the operation's code character. If the first character is a "$" character, that menu item will be treated as a comment line.

The cursor cannot be moved to comment lines. The cursor cannot be moved at all if all of the menu items are comments. In this case, the *menu#* will be returned unchanged.

Specify the currently highlighted menu item with the *menu#* parameter. The *menu#* can be set from 0 to n–1, where n is the number of items in the menu. The selected menu item's menu number will be stored in *menu#* when the subroutine ends. The *menu#* will be returned unchanged if the subroutine is terminated by pressing the Escape Key.

The selected key code will be returned in the *key code* parameter. The return values are listed in the following table.

Key	Return value
Escape Key	0 (&H00)
Enter Key or operation code character	1 (&H01)
Shift+Escape Keys	128 (&H80)
Shift+Enter Keys	129 (&H81)

The menu's background is filled in using graphic memory. The graphic display gradation will be set to 0 temporarily, but will be returned to its original setting when this subroutine ends.

Use the *MBOXCLR subroutine to clear the menu after this subroutine ends.

Use the *MBARDISP subroutine if just the display portion of this subroutine is needed.

*MBOXDISP Menu BOX DISPlay

(General-purpose Structural Subroutine)

Action Displays a menu box.

Format CALL *MBOXDISP (*X*, *Y*, *title*, *message*, *menu#*)

Description Displays a menu box.

Specify the character coordinates of the upper-left corner of the menu box with the *X* and *Y* parameters. The possible setting ranges are as follows:

Coord.	Menu with title	Menu without title
X	2 ≤ X ≤ 62 – Max. item length	2 ≤ X ≤ 62 – Max. item length
Y	0 ≤ Y ≤ 23 – Number of menu items	1 ≤ Y ≤ 24 – Number of menu items

Specify the title to be displayed at the top of the menu with the *title* parameter. The title must be shorter than the menu items. No title will appear when the null string is specified.

Specify the menu items to be displayed in the menu in the *message* parameter. Separate the menu items by "/" characters. The first character in each item specifies the operation's code character, as in the following example.

MSG$="O. OFF/S. Smoothing/1. Edge enhancement/E. All edges"

Any code from $20 to $7F (except "$" and "/") can be used for the operation's code character. If the first character is a "$" character, that menu item will be treated as a comment line.

The cursor cannot be moved to comment lines. The cursor cannot be moved at all if all of the menu items are comments. In this case, the *menu#* will be returned unchanged.

Specify the menu item to be highlighted with the *menu#* parameter. The *menu#* can be set from 0 to n–1, where n is the number of items in the menu.

The graphic memory's gradation is not set in this subroutine. Set the gradation from the portion of the program that calls this subroutine.

*MBOXCLR Menu BOX CLeaR

(General-purpose Structural Subroutine)

Action Clears the region where the menu box is displayed.

Format CALL *MBOXCLR (*X*, *Y*, *title*, *message*)

Description Clears a menu box displayed with the *MBOXDS or *MBOXDISP subroutine.

Specify the same parameters specified in the *MBOXDS or *MBOXDISP subroutine. The menu box will not be cleared correctly if the parameters are not the same.

The character memory and graphic memory will be cleared in the region where the menu box is displayed.

*DBOXDS Dialog BOX Display & Select

(General-purpose Structural Subroutine)

Action Controls dialog boxes, including display, data input, selection, and clearing.

Format CALL *DBOXDS (*title*, *menu items*, *menu info*, *data*, *key code*, [, *exit message*])

Description The *DBOXDS subroutine is used to display a dialog box in the center of the screen, input data or select items, and clear the dialog box from the screen.

The keyboard or console can be used to select items displayed by *DBOXDS. The following keys are used to make a selection:

Key	Function
↑ Key	Moves up one menu item.
↓ Key	Moves down one menu item.
Escape Key	Exits the menu without making a selection.
Enter Key	Selects a menu item. (numeric and character inputs only)
Operation code character	Selects the menu item with the corresponding code character.

The following keys are used when the menu item requires a numeric input. After the Enter Key is pressed, the cursor moves to the leftmost number and data can be entered.

Selecting an item:

Key	Function
→ Key	Moves one menu item to the right.
← Key	Moves one menu item to the left.

Numeric input key operations before the Enter Key is pressed:

Key	Function
→ Key	Increments by the amount specified in the increment information. Won't increment past the max. value.
← Key	Decrements by the amount specified in the increment information. Won't decrement past the min. value.

Numeric input key operations after the Enter Key is pressed:

Key	Function
↑ Key	Displays the next higher value. The minus sign and then the decimal point are displayed after 9.
↓ Key	Displays the next lower value. The decimal point and then the minus sign are displayed after 0.
0 to 9, –, .	Inputs the number directly.
→ Key	Moves one digit to the right.
← Key	Moves one digit to the left.
Escape Key	Returns to the data displayed before the Enter Key was pressed and ends input status.
Enter Key	Enters the data and ends input status. (The maximum value will be entered if the data exceeds the maximum value.)

Character string input key operations after the Enter Key is pressed:

Key	Function
↑ Key	Increments the specified range's character code.
↓ Key	Decrements the specified range's character code.
Character code	Inputs the number directly.
→ Key	Moves the cursor one digit to the right.
← Key	Moves the cursor one digit to the left.
Escape Key	Returns to the data displayed before the Enter Key was pressed and ends input status.
Enter Key	Enters the data and ends input status.

Specify the title to be displayed at the top of the dialog box with the *title* parameter. No title will appear when the null string is specified. The title name must be shorter than the menu selection item name.

Specify the menu items to be displayed with the *menu items* parameter. Separate the menu items by "/" characters. The first character in each item specifies the operation's code character, as in the following example.

MSG$="P. Number of pins:/W. Pin width:/D. Measurement field:"

Adding a colon after each menu item specifies that the length of the character strings will be made equal.

Any code from $20 to $7F (except "$" and "/") can be used for the operation's code character.

If the first character is a "$" character, that menu item will be treated as a comment line. The cursor cannot be moved to a comment line.

Specify the information for each menu item with the *menu info* parameter. Separate the menu information for each item with "/" characters. There are three kinds of menu items: selection items, numeric input items, and character input items. The following list describes how to set each kind of menu item.

1, 2, 3...

1. The following line shows the *menu info* format for selection items. Separate each parameter by "&" characters.

   S&*item name 0*&*item name 1* ... &*item name n*

Parameter(s)	Function
S	Indicates the menu item is a selection item.
Item names 0 to n	Specify the selection item names.

2. The following line shows the *menu info* format for numeric input items. Separate each parameter by "&" characters.

   N&*units&decimal point&increment&digits&min. value&max. value*

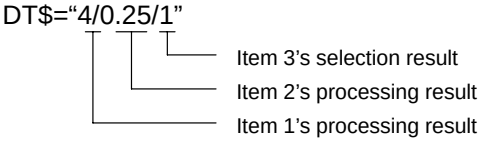
Parameter(s)	Function
N	Indicates the menu item is a numeric input item.
units	Specify the value's units as a character string.
decimal point	Specifies whether the value has a decimal point. Set this parameter to 1 to specify a decimal point, 0 to specify no decimal point.
increment	Specifies the amount to increment (decrement) when the → (←) Key is pressed.
digits	Specifies the number of digits in the input area.
min. value	Specifies the minimum value for the input data.
max. value	Specifies the maximum value for the input data.

3. The following line shows the *menu info* format for character input items. Separate each parameter by "&" characters. The *first character* and *last character* parameters cannot be set to "&" or "/."

   C&*units&characters&first character&last character*

Parameter(s)	Function
C	Indicates the menu item is a character input item.
units	Arbitrary character string
characters	Specifies the number of characters in the input area.
first character	Specifies the first character in the allowed input range.
last character	Specifies the last character in the allowed input range.

Specify the data values to be passed to the subroutine with the *data* parameter. The processed data can be read from this parameter when the *DBOXDS subroutine ends. When selection items have been specified, the data corresponds to the selection results for selection items 0 to n, in that order. Separate the data values with "/" characters, as shown in the following diagram.



When the menu item is a character input item, a character string with the specified number of *characters* will be returned, regardless of the position of the cursor when the Enter Key is pressed.

The selected key code will be returned in the *key code* parameter. The return values are listed in the following table.

Key	Return value
Escape Key	0 (&H00)
Enter Key	1 (&H01)
Shift+Escape Keys	128 (&H80)
Shift+Enter Keys	129 (&H81)

The input data is set when the Enter Key is pressed at the last item or the Shift+Escape or Shift+Enter Keys are pressed during data input.

An exit message can be specified in the *exit message* parameter if desired. The string "E. Exit" will be used if this parameter is omitted or the null string is specified.

Do not include the "/" character in character inputs. The set data cannot be separated correctly if the "/" character is input.

Operation will not be reliable if the character strings for the *menu items* and *menu info* parameters are set incorrectly.

The dialog box's background is filled in using graphic memory. The graphic display gradation will be set to 0 temporarily, but will be returned to its original setting when this subroutine ends.

After this subroutine ends, the contents of the character memory and graphic memory are returned to the contents before the dialog box was displayed.

*BTNBOXDS BuTtoN BOX Display & Select

(General-purpose Structural Subroutine)

Action Controls the display of button boxes and selection of items in button boxes.

Format CALL *BTNBOXDS (*title*, *message*, *button#*, *key code*)

Description The *BTNBOXDS subroutine is used to display a button box in the center of the screen, select items, and clear the button box from the screen.

Specify the title to be displayed at the top of the button box with the *title* parameter. Line returns can be inserted in the title with "/" characters. Lines in the title can be up to 60 bytes long and there can be up to 21 lines. An example title is shown below.

TITL$="Execute the process?/Please select."

Specify the button items to be displayed in the box with the *message* parameter. Separate the button items with "/" characters. The first character in each item specifies the operation's code character, as in the following example.

MSG$="X. Execute/C. Cancel"

Specify the currently highlighted button item with the *button#* parameter. The *button#* can be set from 0 to n–1, where n is the number of items in the box. The selected item's button number will be stored in *button#* when the subroutine ends. The *button#* will be returned unchanged if the subroutine is terminated by pressing the Escape Key.

The selected key code will be returned in the *key code* parameter. The return values are listed in the following table.

Key	Return value
Escape Key	0 (&H00)
Enter Key	1 (&H01)
Shift+Escape Keys	128 (&H80)
Shift+Enter Keys	129 (&H81)

The button box's background is filled in using graphic memory. The graphic display gradation will be set to 0 temporarily, but will be returned to its original setting when this subroutine ends.

After this subroutine ends, the contents of the character memory and graphic memory are returned to the contents before the button box was displayed.

*MSGBOXDISP MeSsaGe BOX DISPlay

(General-purpose Structural Subroutine)

Action Displays a message box.

Format CALL *MSGBOXDISP (*message*)

Description The *MSGBOXDISP subroutine displays the specified message in a box in the center of the screen.

Specify the message to be displayed at with the *message* parameter. Line returns can be inserted in the title with "/" characters. Lines in the message can be up to 60 bytes long and there can be up to 22 lines. An example message is shown below.

MSG$="Writing data.../Please wait."

Use the *MSGBOXCLR subroutine to clear the message box from the screen. The display will be returned to its status before the *MSGBOXDISP subroutine was executed. Always use the *MSGBOXDISP and *MSGBOXCLR subroutines together.

*MSGBOXCLR MeSsaGe BOX CLeaR

(General-purpose Structural Subroutine)

Action Clears a message box.

Format CALL *MSGBOXCLR (*message*)

Description Clears the message box displayed with the *MSGBOXDISP subroutine.

Specify the same *message* that was specified in the *MSGBOXDISP subroutine. Always use the *MSGBOXDISP and *MSGBOXCLR subroutines together.

6-3 Region Setting Library

6-3-1 Introduction

The subroutines in the region setting library can be used to make the following eight settings.

Rectangle settings
Circle settings
Concentric circle settings
Line settings
Continuous line settings
Single-point settings
Polygon settings
Independent curve settings

The region setting library is intended to simplify operations that set the coordinates of patterns being drawn, so these subroutines do not draw the patterns, they just provide the coordinates.

Subroutines in the region setting library use the graphic memory to perform operations such as drawing boxes. The graphic display gradation is temporarily set to 255 while the graphic memory is being used, but is returned to its original setting when this subroutine ends.

Drawing in the graphic memory is achieved by XOR operations, so the patterns that have been drawn in graphic memory are retained.

6-3-2 Region Setting Library Subroutines

This section describes the general-purpose structural subroutines in the region setting library.

*SETBOX SET BOX

(General-purpose Structural Subroutine)

Action Sets a rectangular region.

Format CALL *SETBOX (*X1*, *Y1*, *X2*, *Y2*, *key code* [, *display* [, *XMAX* [, *XMIN* [, *YMAX* [, *YMIN*]]]]])

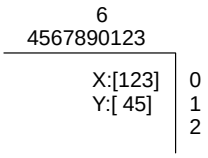
Description Sets a rectangular region with the console or keyboard and returns the rectangle's coordinates.

Coordinates *X1*, *Y1*, *X2*, and *Y2* specify the initial settings for the rectangle and the set coordinates are returned in these parameters. The setting range for X1 and X2 is 0 ≤ X ≤ 511 and the setting range for Y1 and Y2 is 0 ≤ Y ≤ 483.

The *key code* parameter indicates whether the subroutine was ended by pressing the Escape Key or Enter Key.

0: The Escape Key was pressed.
1: The Enter Key was pressed.

When the *display* parameter is other than 0, the cursor coordinates are displayed in the upper-right corner of the screen and coordinates can be input directly from the keyboard by inputting X,x and Y,y. The default setting for *display* is 0.

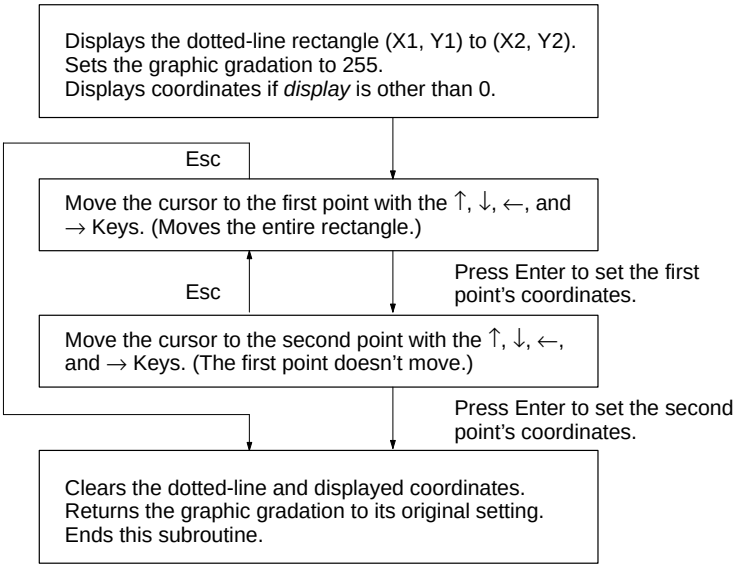


The *XMAX*, *XMIN*, *YMAX*, and *YMIN* parameters limit the size of the rectangle that can be set. It is not possible to set rectangles smaller or larger than these limits. These parameters must meet the following conditions.

1 ≤ XMIN ≤ XMAX ≤ 512
1 ≤ YMIN ≤ YMAX ≤ 484

The default settings are as follows: XMIN=1, XMAX=512, YMIN=1, YMAX=484.

The overall flow of this subroutine is shown in the following flowchart:



If the Shift+Arrow Keys are pressed together, the cursor will be moved by 10 pixels instead of 1. The cursor can be moved diagonally by pressing two Arrow Keys simultaneously. The cursor can be moved within the range (0,0) to (511,483).

The contents displayed in the character memory's coordinate display area will be deleted when there is a display.

No value will be returned for an *X1*, *Y1*, *X2*, *Y2*, or *key code* parameter if a constant was specified.

The dotted-line box is drawn using graphic memory.

*SETCIRCLE SET CIRCLE

(General-purpose Structural Subroutine)

Action Sets a circular region.

Format `CALL *SETCIRCLE` (*X*, *Y*, *R*, *key code* [, *display* [, *RMAX* [, *RMIN*]]])

Description Sets a circular region with the console or keyboard and returns the circle's coordinates.

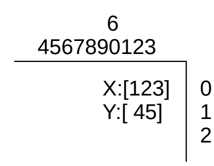
Parameters *X* and *Y* specify the circle's center coordinates and *R* specifies the circle's radius. The setting ranges for these parameters is as follows:

$0 \leq X \leq 511$
$0 \leq Y \leq 483$
$0 \leq R \leq 511$

The *key code* parameter indicates whether the subroutine was ended by pressing the Escape Key or Enter Key.

0: The Escape Key was pressed.
1: The Enter Key was pressed.

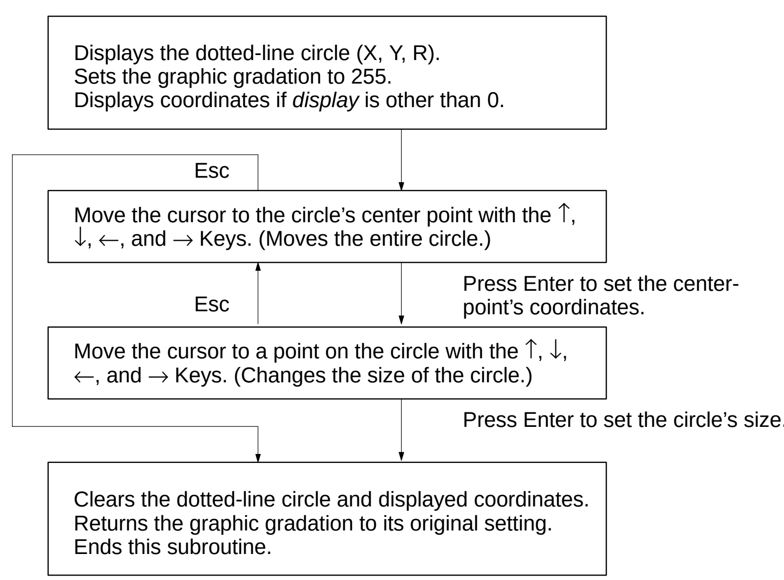
When the *display* parameter is other than 0, the cursor coordinates are displayed in the upper-right corner of the screen and coordinates can be input directly from the keyboard by inputting X,x and Y,y. The default setting for *display* is 0.



The *RMAX* and *RMIN* parameters limit the size of the circle that can be set. It is not possible to set circles smaller or larger than these limits. These parameters must meet the following condition: $0 \leq RMIN \leq RMAX \leq 511$.

The default settings are as follows: RMIN=0, RMAX=511.

The overall flow of this subroutine is shown in the following flowchart:



If the Shift+Arrow Keys are pressed together, the cursor will be moved by 10 pixels instead of 1. The cursor can be moved diagonally by pressing two Arrow Keys simultaneously. The cursor can be moved within the range (0,0) to (511,483).

Circles that go beyond the display can be set when selecting a point on the circle.

The contents displayed in the character memory's coordinate display area will be deleted when there is a display.

No value will be returned for an *X*, *Y*, *R*, or *key code* parameter if a constant was specified.

The dotted-line circle is drawn using graphic memory.

*SETCCIRCLE SET Concentric CIRCLE

(General-purpose Structural Subroutine)

Action Sets two concentric circles.

Format CALL *SETCCIRCLE (*X*, *Y*, *R1*, *R2*, *key code* [, *display* [, *R1MAX* [, *R2MIN* [, *WMIN*]]]])

Description Sets concentric circles with the console or keyboard and returns the circles' coordinates.

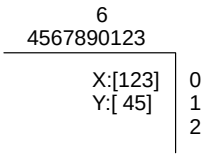
Parameters *X* and *Y* specify the circles' center coordinates. *R1* specifies the outer circle's radius and *R2* specifies the inner circle's radius. The setting ranges for these parameters is as follows:

0 ≤ X ≤ 511
0 ≤ Y ≤ 483
0 ≤ R2 ≤ R1 ≤ 511

The *key code* parameter indicates whether the subroutine was ended by pressing the Escape Key or Enter Key.

0: The Escape Key was pressed.
1: The Enter Key was pressed.

When the *display* parameter is other than 0, the cursor coordinates are displayed in the upper-right corner of the screen and coordinates can be input directly from the keyboard by inputting X,x and Y,y. The default setting for *display* is 0.

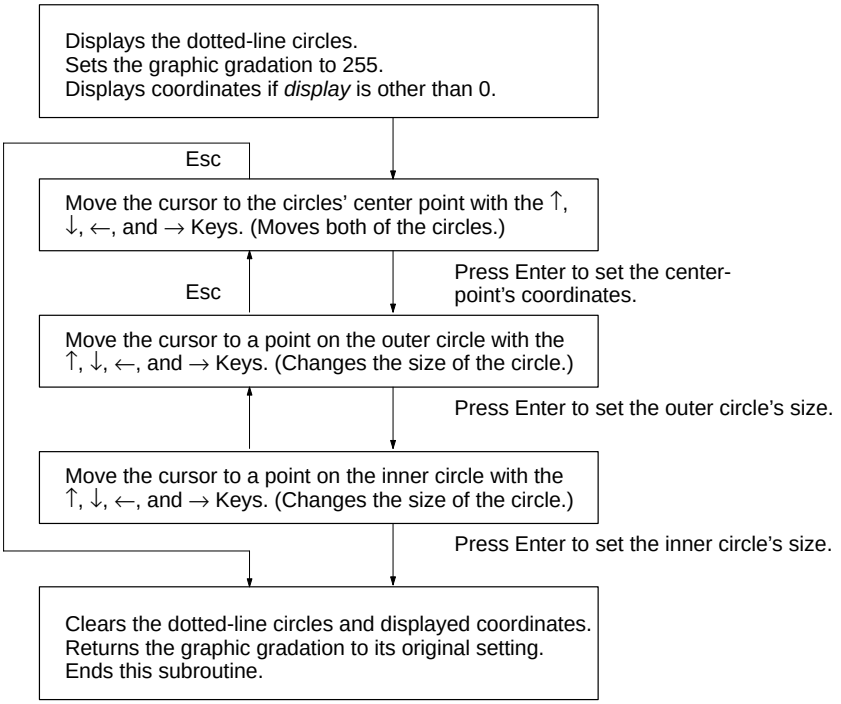


The value in *R1MAX* specifies the maximum radius for the outer circle, *R2MIN* specifies the minimum radius for the inner circle, and *WMIN* specifies the minimum difference between the two radii. These parameters must meet the following conditions:

0 ≤ R2MIN ≤ R1MAX ≤ 511
0 ≤ WMIN ≤ R1MAX – R2MIN

The default settings are as follows: R1MAX=511, R2MIN=0, WMIN=0.

The overall flow of this subroutine is shown in the following flowchart:



If the Shift+Arrow Keys are pressed together, the cursor will be moved by 10 pixels instead of 1. The cursor can be moved diagonally by pressing two Arrow Keys simultaneously. The cursor can be moved within the range (0,0) to (511,483).

Circles that go beyond the display can be set when selecting a point on the circle.

The contents displayed in the character memory's coordinate display area will be deleted when there is a display.

No value will be returned for an *X*, *Y*, *R1*, *R2*, or *key code* parameter if a constant was specified.

*SETLINE SET LINE

(General-purpose Structural Subroutine)

Action Sets a straight line.

Format CALL *SETLINE (*X1*, *Y1*, *X2*, *Y2*, *key code* [, *display*])

Description Sets a line with the console or keyboard and returns the line's coordinates.

Coordinates *X1*, *Y1*, *X2*, and *Y2* specify the initial settings for the endpoints of the line and the set coordinates are returned in these parameters. The setting range for X1 and X2 is 0 ≤ X ≤ 511 and the setting range for Y1 and Y2 is 0 ≤ Y ≤ 483.

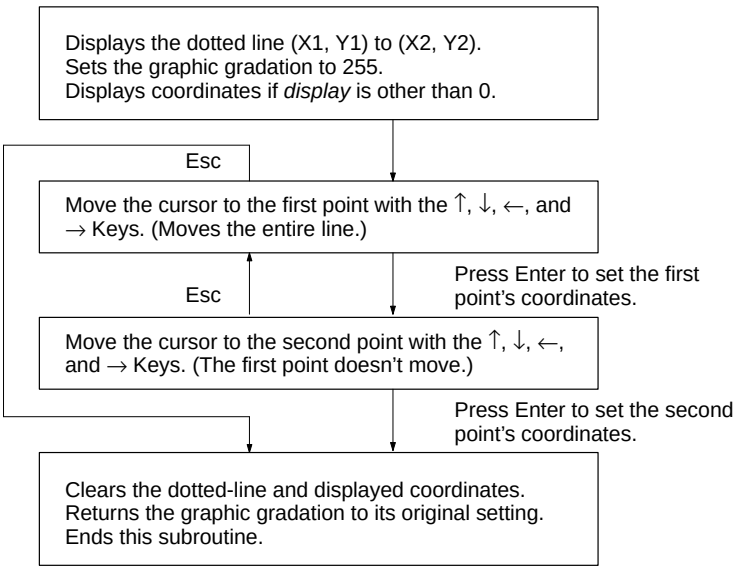
The *key code* parameter indicates whether the subroutine was ended by pressing the Escape Key or Enter Key.

0: The Escape Key was pressed.
1: The Enter Key was pressed.

When the *display* parameter is other than 0, the cursor coordinates are displayed in the upper-right corner of the screen and coordinates can be input directly from the keyboard by inputting X,x and Y,y. The default setting for *display* is 0.

```
      6
4567890123
          |
X:[123]   | 0
Y:[ 45]   | 1
          | 2
```

The overall flow of this subroutine is shown in the following flowchart:



If the Shift+Arrow Keys are pressed together, the cursor will be moved by 10 pixels instead of 1. The cursor can be moved diagonally by pressing two Arrow Keys simultaneously. The cursor can be moved within the range (0,0) to (511,483).

The contents displayed in the character memory's coordinate display area will be deleted when there is a display.

No value will be returned for an *X1*, *Y1*, *X2*, *Y2*, or *key code* parameter if a constant was specified.

The dotted line is drawn using graphic memory.

*SETCLINE SET Continuous LINE

(General-purpose Structural Subroutine)

Action Sets a continuous set of line segments.

Format CALL *SETCLINE (*elements*, *X array*, *Y array*, *key code* [, *display*])

Description Sets a continuous set of line segments with the console or keyboard and returns the line's coordinates.

Specify the number of elements in the X and Y arrays with the *elements* parameter. The continuous line cannot have more points than specified in the *elements* parameter.

The *X array* and *Y array* specify names of one-dimensional arrays containing the coordinates of the line segments. The setting range for the data elements in the *X array* is 0 ≤ X ≤ 511 and the setting range for the data elements in the *Y array* is 0 ≤ Y ≤ 483.

If the subroutine is ended by pressing the Enter Key, the coordinate data that was set will be stored in the X and Y arrays. The remaining array elements will not be changed if fewer points are set than were specified in the *elements* parameter.

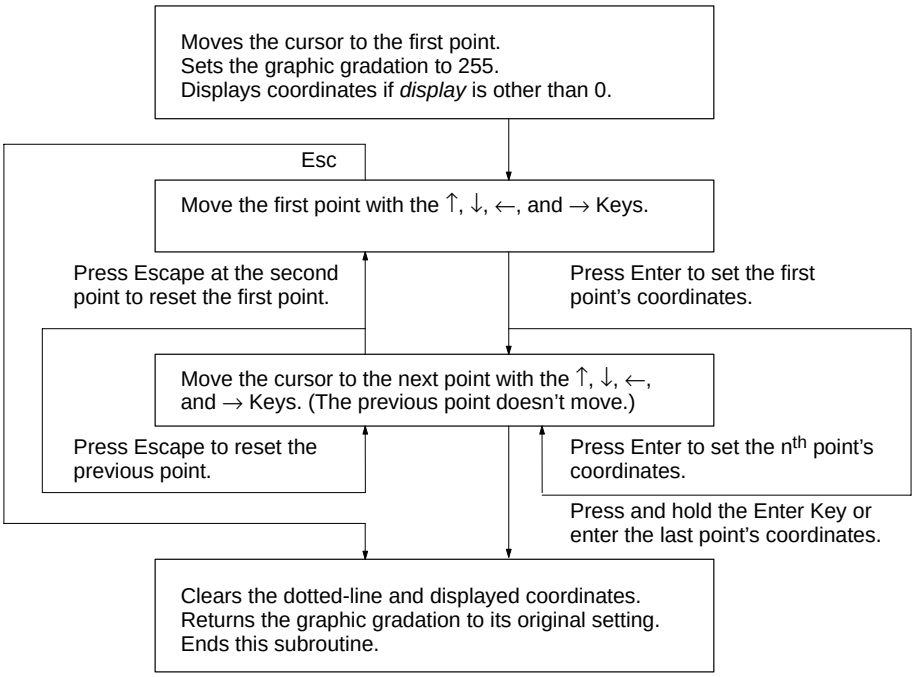
The *key code* parameter indicates whether the subroutine was ended by pressing the Escape Key or Enter Key.

0: The Escape Key was pressed.
1: The Enter Key was pressed.

When the *display* parameter is other than 0, the cursor coordinates are displayed in the upper-right corner of the screen and coordinates can be input directly from the keyboard by inputting X,x and Y,y. The default setting for *display* is 0.

```
      6
4567890123
          X:[123]  0
          Y:[ 45]  1
                   2
```

The overall flow of this subroutine is shown in the following flowchart:



If the Shift+Arrow Keys are pressed together, the cursor will be moved by 10 pixels instead of 1. The cursor can be moved diagonally by pressing two Arrow Keys simultaneously. The cursor can be moved within the range (0,0) to (511,483).

The contents displayed in the character memory's coordinate display area will be deleted when there is a display.

No value will be returned for an *X*, *Y*, or *key code* parameter if a constant was specified.

The dotted line is drawn using graphic memory.

*SETPOINT SET POINT

(General-purpose Structural Subroutine)

Action Sets a single point on the coordinate system.

Format CALL *SETPOINT (*X*, *Y*, *key code* [, *display* [*X1, Y1, X2, Y2*]])

Description Sets a point with the console or keyboard and returns the point's coordinates.

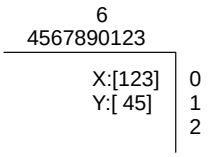
Coordinates *X* and *Y* specify the initial cursor position when the coordinates are specified, and the set coordinates are returned in these parameters. The setting range for *X* is 0 ≤ X ≤ 511 and the setting range for *Y* is 0 ≤ Y ≤ 483.

The *key code* parameter indicates whether the subroutine was ended by pressing the Escape Key or Enter Key.

0: The Escape Key was pressed.
1: The Enter Key was pressed.

When the *display* parameter is zero, the cursor coordinates are not displayed in the upper right corner of the screen.

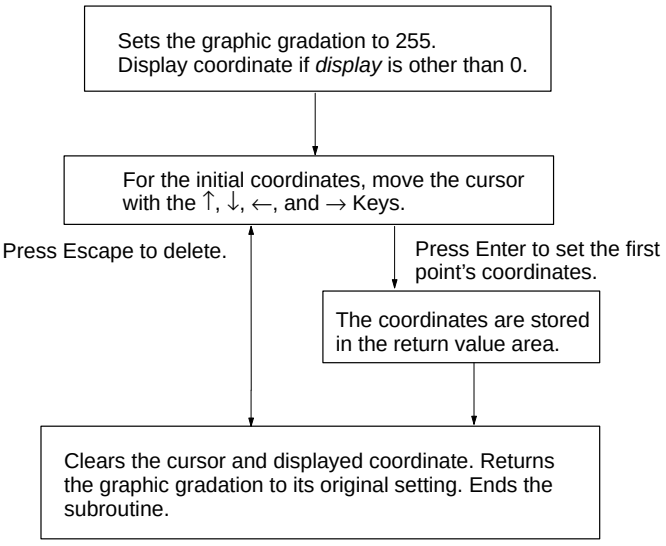
When the *display* parameter is other than 0, the cursor coordinates are displayed in the upper-right corner of the screen and coordinates can be input directly from the keyboard by inputting X, (x) and Y, (y). The default setting for *display* is 0.



The cursor's range of movement when setting the coordinates is specified by *X1*, *Y1*, *X2*, and *Y2*. The cursor cannot move outside of the region set here. The setting range for *X1* and *X2* is 0 ≤ X ≤ 511, and the setting range for *Y1* and *Y2* is 0 ≤ Y ≤ 483. If this setting is omitted, the following values will be regarded as being set:

X1=0, X2=511
Y1=1, Y2=483

The overall flow of this subroutine is shown in the following flowchart:



If the Shift+Arrow Keys are pressed together, the cursor will be moved by 10 pixels instead of 1.

The contents displayed in the character memory's coordinate display area will be deleted when there is a display.

No value will be returned for an *X*, *Y*, and *key code* parameters if a constant was specified.

The graphic cursor is drawn using the graphic memory.

*SETPOLYGON SET POLYGON

(General-purpose Structural Subroutine)

Action Sets a polygon.

Format CALL *SETPOLYGON (*elements*, *X array*, *Y array*, *key code* [, *display*])

Description Sets a polygon with the console or keyboard and returns the polygon's coordinates.

Specify the number of elements in the X and Y arrays with the *elements* parameter. The polygon cannot have more points than specified in the *elements* parameter.

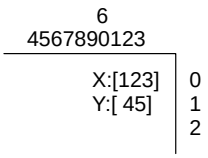
The *X array* and *Y array* specify names of one-dimensional arrays containing the coordinates of the points in the polygon. The setting range for the data elements in the *X array* is 0 ≤ X ≤ 511 and the setting range for the data elements in the *Y array* is 0 ≤ Y ≤ 483.

If the subroutine is ended by pressing the Enter Key, the coordinate data that was set will be stored in the X and Y arrays. The remaining array elements will not be changed if fewer points are set than were specified in the *elements* parameter.

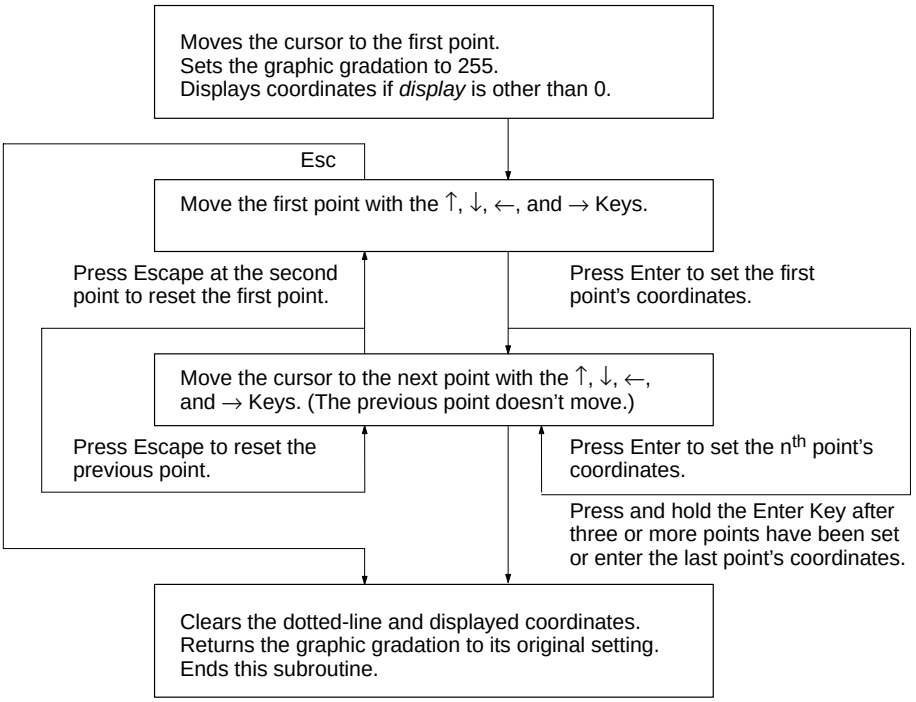
The *key code* parameter indicates whether the subroutine was ended by pressing the Escape Key or Enter Key.

0: The Escape Key was pressed.
1: The Enter Key was pressed.

When the *display* parameter is other than 0, the cursor coordinates are displayed in the upper-right corner of the screen and coordinates can be input directly from the keyboard by inputting X,x and Y,y. The default setting for *display* is 0.



The overall flow of this subroutine is shown in the following flowchart:



If the Shift+Arrow Keys are pressed together, the cursor will be moved by 10 pixels instead of 1. The cursor can be moved diagonally by pressing two Arrow Keys simultaneously. The cursor can be moved within the range (0,0) to (511,483).

The contents displayed in the character memory's coordinate display area will be deleted when there is a display.

The dotted line is drawn using graphic memory.

If fewer than three points have been set the input mode will not end even if the Return Key is held down.

*SETSPLINE SET SPLINE

(General-purpose Structural Subroutine)

Action Sets an independent curve.

Format CALL *SETSPLINE (*elements*, *X array*, *Y array*, *key code* [, *display*])

Description Sets an independent curve with the console or keyboard and returns the curve's coordinates.

Specify the number of elements in the X and Y arrays with the *elements* parameter. The curve cannot have more points than specified in the *elements* parameter.

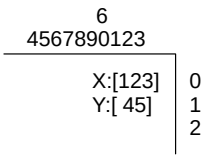
The *X array* and *Y array* specify names of one-dimensional arrays containing the coordinates of the points in the curve. The setting range for the data elements in the *X array* is 0 ≤ X ≤ 511 and the setting range for the data elements in the *Y array* is 0 ≤ Y ≤ 483.

If the subroutine is ended by pressing the Enter Key, the coordinate data that was set will be stored in the X and Y arrays. The remaining array elements will not be changed if fewer points are set than were specified in the *elements* parameter.

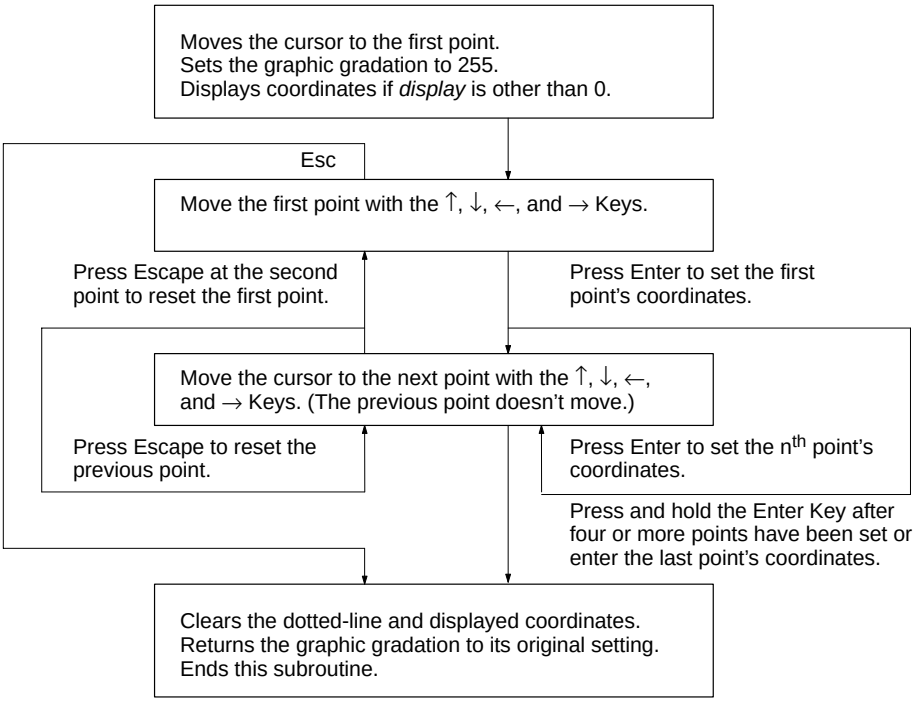
The *key code* parameter indicates whether the subroutine was ended by pressing the Escape Key or Enter Key.

0: The Escape Key was pressed.
1: The Enter Key was pressed.

When the *display* parameter is other than 0, the cursor coordinates are displayed in the upper-right corner of the screen and coordinates can be input directly from the keyboard by inputting X,x and Y,y. The default setting for *display* is 0.



The overall flow of this subroutine is shown in the following flowchart:



If the Shift+Arrow Keys are pressed together, the cursor will be moved by 10 pixels instead of 1. The cursor can be moved diagonally by pressing two Arrow Keys simultaneously. The cursor can be moved within the range (0,0) to (511,483).

The contents displayed in the character memory's coordinate display area will be deleted when there is a display.

The dotted line is drawn using graphic memory.

If fewer than four points have been set the input mode will not end even if the Return Key is held down.

6-4 Character Display Library

6-4-1 Introduction

The six subroutines in the character display library perform the following functions:

Subroutine	Function
*BTMMSG	Button message display
*MAKETABLE	Displays a framework of lines.
*FASTPRINT	Displays data quickly.
*FASTPRINT2	Displays “OK/NG” quickly.

Subroutine	Function
*FASTPRINT3	Displays character string data quickly.
*SYMBOL	Enlarges and displays characters.

The character display library provides functions that display measurement results and help messages.

Drawing in the graphic memory is achieved by XOR operations, so the patterns that have been drawn in graphic memory are retained.

6-4-2 Character Display Library Subroutines

This section describes the general-purpose structural subroutines in the character display library.

The subroutines in the character display library can be used to make the following five displays.

Framework displays
HIgh-speed data displays
High-speed OK/NG displays
High-speed character string displays
Expanded character displays

*BTMMSG BoTtoM MeSsaGe

(General-purpose Structural Subroutine)

Action Displays a message on the bottom line of the screen.

Format CALL *BTMMSG (*message*)

Description Clears the second to last line (above the function key line) and displays the *message*. If the character string in *message* is longer than one line, the extra characters will not be displayed.

*MAKETABLE MAKE TABLE

(General-purpose Structural Subroutine)

Action Creates a table framework in graphic memory.

Format CALL *MAKETABLE (*X*, *Y*, *vertical line format*, *horizontal line format*)

Description Displays a framework of lines with the specified format in graphic memory.

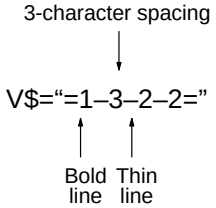
Parameters *X* and *Y* specify the character coordinates of the upper-left corner of the table. The setting ranges for these parameters are as follows:

0 ≤ X ≤ 63 – table's width
0 ≤ Y ≤ 24 – table's height

The *vertical line format* parameter is a character string that specifies the spacing and thickness of the vertical lines.

The *horizontal line format* parameter is a character string that specifies the spacing and thickness of the horizontal lines.

The line spacing is specified by integer values and the line thickness is specified by "=" or "–" characters. A "=" character indicates a bold line and a "–" character indicates a thin line. A bold line is twice as thick as a thin line.



The lines are drawn in graphic memory. The graphic memory's gradation is not set in this subroutine. Set the gradation from the portion of the program that calls this subroutine.

*FASTPRINT FAST PRINT

(General-purpose Structural Subroutine)

Action Displays data at high speed.

Format CALL *FASTPRINT (*elements*, *X array*, *Y array*, *data array*, *format* [[, *display array*] [, *reverse array*]])

Description Displays the data stored in an array all at once at high speed.

Specify the number of data elements (0 to 256) to be displayed with the *elements* parameter. If this parameter is set to 0, the subroutine will end normally without performing any processing.

The *X array* and *Y array* parameters specify names of one-dimensional arrays containing the character coordinates where each data element will be displayed. The setting ranges for these coordinates are as follows. (The corresponding data element will not be displayed if its coordinate is set to –1.)

–1 ≤ X ≤ 63
–1 ≤ Y ≤ 24

The *data array* parameter specifies the name of a one-dimensional array containing the data to be displayed.

The *format* parameter specifies the number and position of the digits with "#" characters and the position of the decimal point with a decimal point, as shown below.

"####.##"

A maximum of nine integer digits (before the decimal point) and six non-integer digits (following the decimal point) can be displayed. If the data exceeds these maximum values, just the maximum number of digits will be displayed.

The *display array* parameter specifies the name of a one-dimensional array that indicates whether or not the corresponding data elements will be displayed. A data element will not be displayed if the corresponding element in the *display array* is –1.

–1: The data element is not displayed.
Not –1: The data element is displayed.

All of the data elements will be displayed if the *display array* parameter is omitted.

The *reverse array* parameter specifies the name of a one-dimensional array that indicates whether or not the corresponding data elements will be displayed in reverse video. A data element will be displayed normally if the corresponding element in the *reverse array* is 0.

0: Normal display
Other than 0: Reverse display

All of the data elements will be displayed normally if the *reverse array* parameter is omitted.

Only integer arrays can be used for the *X array*, *Y array*, *display array*, and *reverse array*.

*FASTPRINT2 FAST PRINT 2

(General-purpose Structural Subroutine)

Action Displays OK/NG results at high speed.

Format CALL *FASTPRINT2 (*elements*, *X array*, *Y array*, *results array* [, [*display array*] [, [*reverse mode*] [, *side multiple mode*]]])

Description Displays the results stored in an array all at once at high speed.

Specify the number of data elements (0 to 256) to be displayed with the *elements* parameter. If this parameter is set to 0, the subroutine will end normally without performing any processing.

The *X array* and *Y array* parameters specify names of one-dimensional arrays containing the character coordinates where each data element will be displayed. The setting ranges for these coordinates are as follows. (The corresponding data element will not be displayed if its coordinate is set to –1.)

–1 ≤ X ≤ 63
–1 ≤ Y ≤ 24

The *results array* parameter specifies the name of a one-dimensional array containing the results to be displayed. A value of 0 indicates OK and a other-than-0 value indicates NG (no good).

0: Displays OK.
Other than 0: Displays NG.

The *display array* parameter specifies the name of a one-dimensional array that indicates whether or not the corresponding result will be displayed. A result will not be displayed if the corresponding element in the *display array* is –1.

–1: The result is not displayed.
Not –1: The result is displayed.

All of the results will be displayed if the *display array* parameter is omitted.

The *reverse mode* parameter specifies whether NG results will be displayed in reverse video. The default setting is normal display.

–1: Reverse display
Not –1: Normal display

The *side multiple mode* parameter specifies whether OK/NG results will be displayed in side multiple mode. The default setting is 0.

0: Normal mode
Other than 0: Side multiple mode

Only integer arrays can be used for the *X array*, *Y array*, *results array*, and *display array*.

*FASTPRINT3 FAST PRINT 3

(General-purpose Structural Subroutine)

Action Displays multiple character strings at high speed.

Format CALL *FASTPRINT3 (*elements*, *X array*, *Y array*, *data array*, [[, *display array*] [, *reverse array*]])

Description Displays the data stored in an array all at once at high speed.

Specify the number of data elements (0 to 256) to be displayed with the *elements* parameter. If this parameter is set to 0, the subroutine will end normally without performing any processing.

The *X array* and *Y array* parameters specify names of arrays containing the character coordinates where data elements are to be displayed. The setting ranges for these coordinates are as follows. (The corresponding data element will not be displayed if its coordinate is set to –1.)

–1 ≤ X ≤ 63
–1 ≤ Y ≤ 24

The *data array* parameter specifies the name of a one-dimensional array containing the data to be displayed.

The *display array* parameter specifies the name of a one-dimensional array that indicates whether or not the corresponding data elements will be displayed. A data element will not be displayed if the corresponding element in the *display array* is –1.

–1: The data element is not displayed.
Not –1: The data element is displayed.

All of the data elements will be displayed if the *display array* parameter is omitted.

The *reverse array* parameter specifies the name of a one-dimensional array that indicates whether or not the corresponding data elements will be displayed in reverse video. A data element will be displayed normally if the corresponding element in the *reverse array* is 0.

0: Normal display
Other than 0: Reverse display

All of the data elements will be displayed normally if the *reverse array* parameter is omitted.

Only integer arrays can be used for the *X array*, *Y array*, *display array*, and *reverse array*.

*SYMBOL SYMBOL

(General-purpose Structural Subroutine)

Action Enlarges the specified character string and draws it in character memory.

Format CALL *SYMBOL (*X*, *Y*, *ratio*, *character string* [, *format*])

Description Enlarges and displays the specified character string.

Parameters *X* and *Y* specify the character coordinates where the character string is to be displayed. The upper-left point of the enlarged character string will be displayed at character coordinates (X,Y). The setting ranges for these parameters are as follows:

0 ≤ X ≤ 63
0 ≤ Y ≤ 24

The *ratio* parameter specifies the zooming ratio (1 to 16). The ratio must be a power of 2 (1, 2, 4, 8, or 16); if another value between 1 and 16 is specified, the power of 2 lower than the specified value will be used.

The *character string* parameter specifies the character string to be enlarged.

The *format* parameter indicates whether or not the enlarged character string will be displayed in reverse video. The default setting is 0.

0: Normal display
Other than 0: Reverse display

Characters that extend off the screen will not be displayed.

Smoothing is not performed after the character display is enlarged.

6-5 Image Control Library

6-5-1 Introduction

The three subroutines in the image control library perform the following functions:

Subroutine	Function
*MAKEBRIGHT	Displays the line brightness.
*MAKEHIST	Displays a histogram.
*DISPMODE	Displays the selected image.

The subroutines in the image control library operate on the contents of image memory, so the desired image data must be input into image memory before these subroutines are called.

6-5-2 Image Control Library Subroutines

This section describes the general-purpose structural subroutines in the image control library.

*MAKEBRIGHT MAKE line BRIGHT

(General-purpose Structural Subroutine)

Action Displays the specified image memory's line brightness.

Format CALL *MAKEBRIGHT (*page#*, *X*, *Y*, *key code*)

Description Draws the line brightness on the specified image memory cross-line in graphic memory.

Specify the image memory page number (0 or 1) from which the line brightness will be taken with the *page#* parameter.

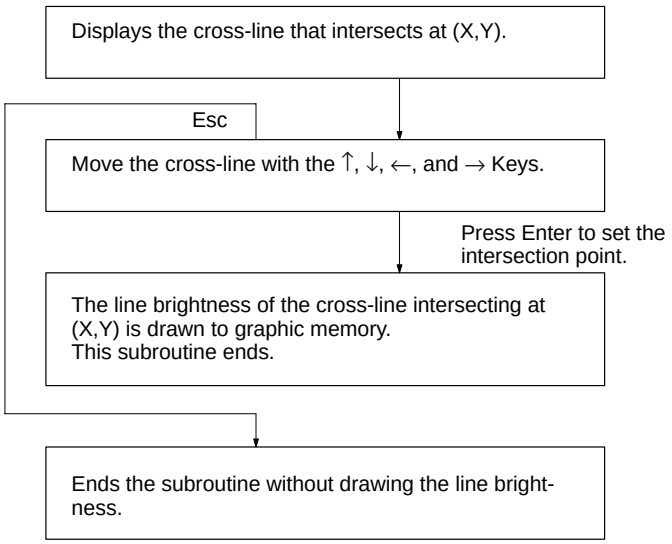
Parameters *X* and *Y* specify the intersection coordinates from which the line brightness will be taken. The setting ranges for these parameters is as follows:

0 ≤ X ≤ 511
0 ≤ Y ≤ 483

The *key code* parameter indicates whether the subroutine was ended by pressing the Escape Key or Enter Key.

0: The Escape Key was pressed.
1: The Enter Key was pressed.

The overall flow of this subroutine is shown in the following flowchart:



The image must be input in image memory in advance.

The line brightness is drawn in graphic memory.

*MAKEHIST MAKE HISTogram

(General-purpose Structural Subroutine)

Action Displays a histogram of the specified image memory.

Format CALL *MAKEHIST (*page#*, *X1*, *Y1*, *X2*, *Y2*, *key code*)

Description Draws a histogram in graphic memory based on the image in the specified rectangular region in image memory.

Specify the image memory page number (0 or 1) that contains the desired image with the *page#* parameter.

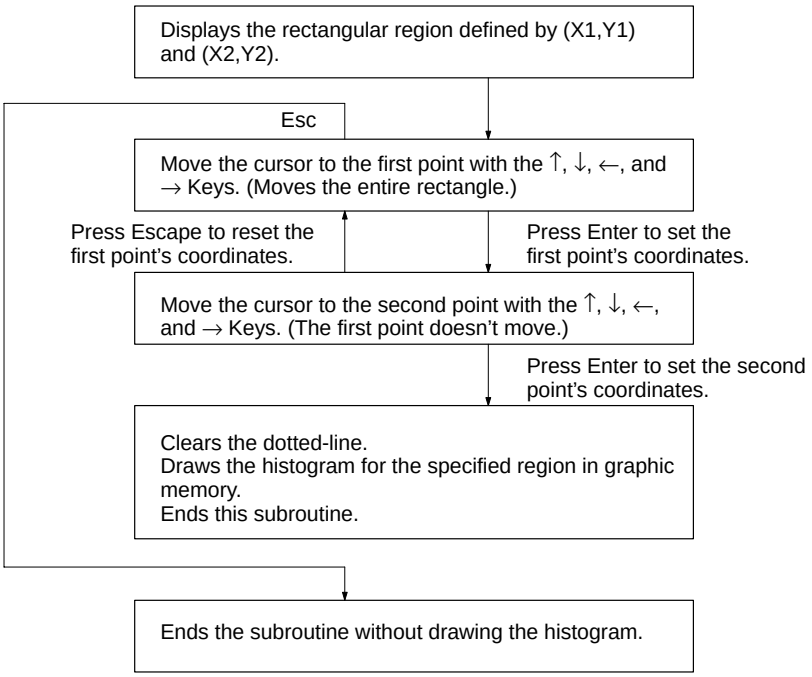
Specify the upper-left and lower-right corners of the rectangular region with points *(X1*, *Y1)* and *(X2*, *Y2)*. The setting ranges for these parameters is as follows:

0 ≤ X ≤ 511
0 ≤ Y ≤ 483

The *key code* parameter indicates whether the subroutine was ended by pressing the Escape Key or Enter Key.

0: The Escape Key was pressed.
1: The Enter Key was pressed.

The overall flow of this subroutine is shown in the following flowchart:



The image must be input in image memory in advance.

The histogram is drawn in graphic memory and it is displayed in the center of the screen.

*DISPMODE DISPlay MODE

(General-purpose Structural Subroutine)

Action Displays the specified image.

Format CALL *DISPMODE (*display image*, *input path*, [, *page#* [, *window*]])

Description This subroutine is mainly used to control the status of the image output to the video monitor.

Specify whether the image is to be filtered with the *display image* parameter.

0: Original image
1: Filtered image

Specify the type of image that is passed through the video bus (image bus 1) and pipeline bus (image bus 0) with the *input path* parameter.

0: Camera through image
1: Image memory
2: Camera freeze image

If the image path is set to 1: image memory, specify the image memory page number (0 or 1) that contains the desired image with the *page#* parameter. The default setting is 0.

The ON/OFF status of the bits in the *window* parameter specify whether or not the corresponding window plane (0 to 7) will be displayed. The current setting will be used if the *window* parameter is omitted.

6-6 File Operations Library

There is just one subroutine in the file operations library: *SELECTFILE. This subroutine can be used to display a file directory and select a file from the directory. The following disk drives can be used. The current drive will be used if the drive name is omitted.

A: (OVL Unit's internal ROM)
C: (Memory Card)
D: (System ROM)

*SELECTFILE SELECT FILE

(General-purpose Structural Subroutine)

Action Displays a file directory and returns the selected file name.

Format CALL *SELECTFILE (*X*, *Y*, *H*, *file name*, *format*)

Description Displays the file directory for the path specified in *file name* and returns the selected file name.

Specify the character coordinates where the upper-left corner of the file directory menu will be displayed with parameters *X* and *Y*. The setting ranges for these parameters depends on the setting of the *format* parameter, as shown in the following table.

Format	X setting range	Y setting range
0	X=–1 or 2 ≤ X ≤ 47	Y=–1 or 1 ≤ Y ≤ 20
1	2 ≤ X ≤ 39	
2	2 ≤ X ≤ 30	

Specify the number of lines in the file directory (including the title) with parameter *H*. The setting range for *H* is: H=–1 or 2 ≤ H ≤ 21, with *Y*+*H* ≤ 22.

The position of the file directory will be adjusted automatically if *X*, *Y*, and *H* are set to –1.

The *file name* parameter specifies the path of the directory to be displayed. Either relative path names or absolute path names can be specified. Wild card characters can also be used. The only file attributes that can be displayed are normal files, read-only files, and directory files.

The selected file name will be returned in the *file name* parameter. If the file is in the current directory, just the file name will be returned. If the directory has been changed, the absolute path name (including the drive name) will be returned. The null string will be returned in *file name* if the Escape Key is pressed.

The following disk drives can also be specified.

A: (OVL Unit's internal ROM)
C: (Memory Card)
D: (System ROM)

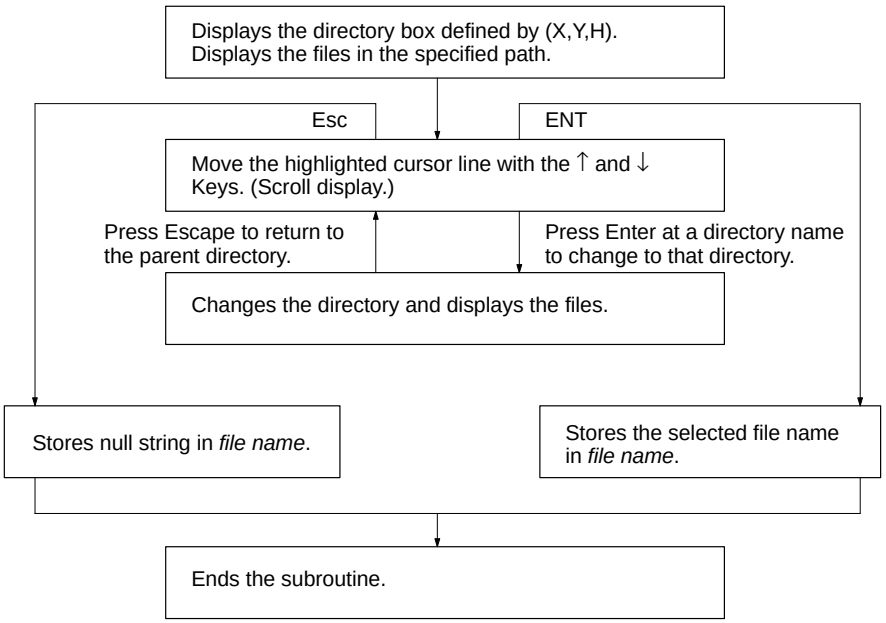
The *format* parameter specifies the directory format, as shown below.

0: File name
1: File name + size
2: File name + size + date

It is possible to change directories.

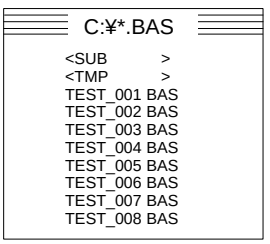
The display will scroll if there are too many files to display in one directory list.

The overall flow of this operation is shown in the following flowchart:



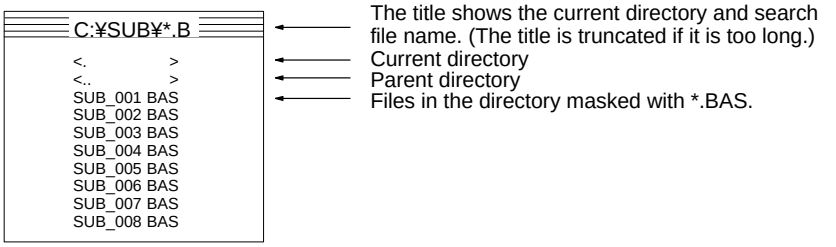
The following displays show example file directories.

Format=0, H=10 lines
In this example, *format*=0 (file name) and *H*=10 lines.¥



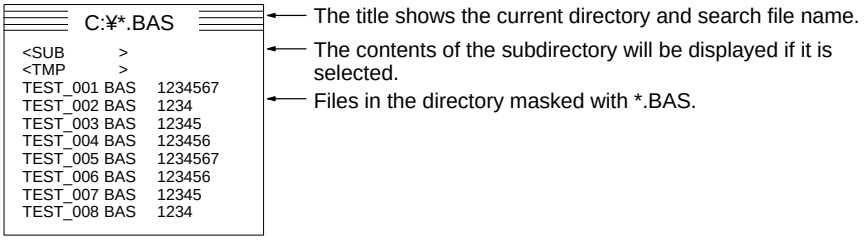
← The title shows the current directory and search file name.
← The contents of the subdirectory will be displayed if it is selected.
← Files in the directory masked with *.BAS.

The masked contents of subdirectory SUB will be displayed if it is selected.

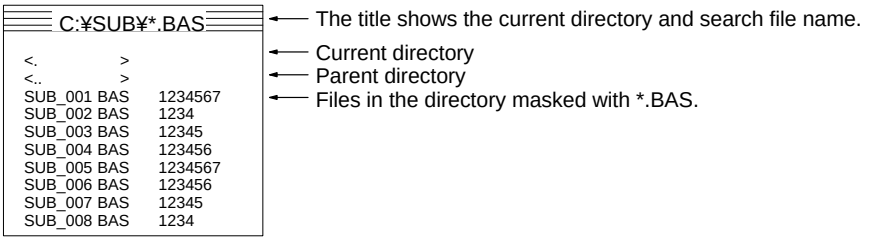


Format=1, H=10 lines

In this example, *format*=1 (file name+size) and *H*=10 lines.

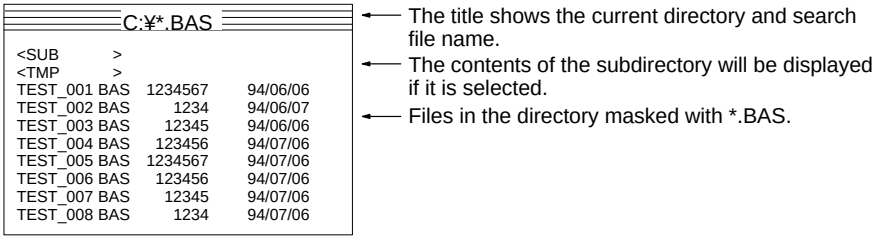


The masked contents of subdirectory SUB will be displayed if it is selected.

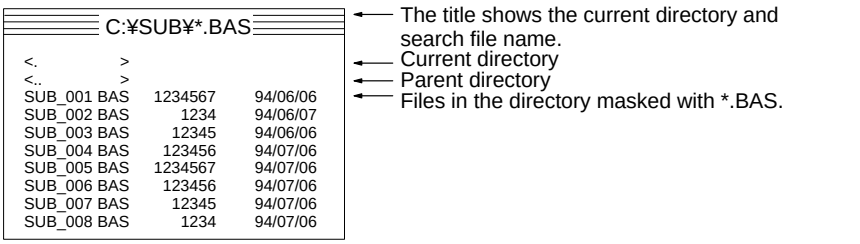


Format=2, H=10 lines

In this example, *format*=2 (file name+size+date) and *H*=10 lines.



The masked contents of subdirectory SUB will be displayed if it is selected.



6-7 Graphic Display Library

6-7-1 Introduction

The two subroutines in the graphic display library perform the following functions:

Subroutine	Function
*FBOX	Displays a rectangle quickly.
*FCURSOR	Displays the cursor quickly.

6-7-2 Graphic Display Library Subroutines

This section describes the general-purpose structural subroutines in the graphic display library.

*FBOX Fast BOX

(General-purpose Structural Subroutine)

Action Quickly draws a rectangle at the coordinate position specified by the array.

Format CALL *FBOX (*elements*, *start-point X array*, *start-point Y array*, *end-point X array*, *end-point Y array*, *VRAM* [,[*page#*] [,*drawing method*] [,*drawing mode* **or** *drawing density*]])

Description This subroutine draws a rectangle at each position specified by the *start-point X array*, *start-point Y array*, *end-point X array* and *end-point Y array* parameters. Specify the number of elements to be drawn (0 to 256) with the *elements* parameter.

Specify the VRAM in which to draw the rectangle with the *VRAM* parameter.

0: Character memory
1: Graphic memory
2: Window memory
3: Image memory
4: Shading memory (Cannot be used with F350-C12E/C41E.)

With the *page#* parameter, specify 0 or 1. (The default setting is 0.)

With the *drawing method* parameter, specify one of the following:
The default setting is 0.

0: Drawing mode
1: Drawing density

Specify OR, NOT, or XOR for the *drawing mode* parameter, as shown below. This parameter is only enabled when the *drawing method* parameter is set to 0 (drawing mode). The default setting is OR.

OR: Draws a value taking an OR of the current VRAM contents and 255.
NOT: Draws 0.
XOR: Reverses the current VRAM contents.

When a frame-type VRAM is specified for the *VRAM* parameter, the contents of planes write-protected with the MASKBIT command remain unchanged.

With the *drawing density* parameter, specify the density (0 to 255) for drawing to the window memory, image memory, and shading memory. The default setting is 255. This parameter is only enabled when the *drawing method* parameter is set to 1 (drawing density).

When either character memory or graphic memory is specified with the *VRAM* parameter, the following values are drawn for the drawing density specification.

0: 0 is drawn.
Other than 0: 1 is drawn.

Solid lines are fixed for drawing.

*FCURSOR Fast CURSOR

(General-purpose Structural Subroutine)

Action Quickly draws a cursor at the coordinate position specified by the array.

Format CALL *FCURSOR (*elements*, *X-coordinate array*, *Y-coordinate array*, *angle array*, *VRAM* [,[*page#*] [,*drawing method*] [,*drawing mode* **or** *drawing density*]])

Description This subroutine draws a cursor at each position specified with the *X-coordinate array*, *Y-coordinate array*, and *angle array* parameters. Specify the number of elements to be drawn (0 to 512) with the *elements* parameter.

With the *X-coordinate array*, *Y-coordinate array*, and *angle array* parameters, specify the names of the one-dimensional variable arrays containing the rectangle drawing start-point and end-point coordinates.

With the *VRAM* parameter, specify the VRAM in which to draw the rectangle.

0: Character memory
1: Graphic memory
2: Window memory
3: Image memory
4: Shading memory (Cannot be used with F350-C12E/C41E.)

With the *page#* parameter, specify 0 or 1. (The default setting is 0.)

With the *drawing method* parameter, specify one of the following:
The default setting is 0.

0: Drawing mode
1: Drawing density

Specify OR, NOT, or XOR for the *drawing mode* parameter, as shown below. This parameter is only enabled when the *drawing method* parameter is set to 0 (drawing mode). The default setting is OR.

OR: Draws a value taking an OR of the current VRAM contents and 255.
NOT: Draws 0.
XOR: Reverses the current VRAM contents.

When a frame-type VRAM is specified for the *VRAM* parameter, the contents of planes write-protected with the MASKBIT command remain unchanged.

With the *drawing density* parameter, specify the density (0 to 255) for drawing to the window memory, image memory, and shading memory. The default setting is 255. This parameter is only enabled when the *drawing method* parameter is set to 1 (drawing density).

When either character memory or graphic memory is specified with the *VRAM* parameter, the following values are drawn for the drawing density specification.

0: 0 is drawn.
Other than 0: 1 is drawn.

6-8 “Other” Library

The “other” library contains additional subroutines that perform common functions.

*DATASORT DATA SORT

(General-purpose Structural Subroutine)

Action Sorts one-dimensional array data according to the specified condition.

Format CALL *DATASORT (*array*, *sort condition* [, *first element* [, *last element*]])

Description Sorts the elements of a one-dimensional array in ascending or descending order.

Specify the name of the one-dimensional array in the *array* parameter.

Specify the *sort condition* parameter as follows:

0: Ascending order
1: Descending order

Specify the range of elements to be sorted with the *first element* and *last element* parameters. The element numbers begin with the initial subscript value (0 or 1) set with the OPTION BASE command.

The default setting for the *first element* parameter is the first element in the array and the default setting for the *last element* parameter is the last element in the array.

Array elements outside of the specified range will not be sorted.

*CALCDATE CALCulate DATE

(General-purpose Structural Subroutine)

Action Adds the specified displacement to the specified date and returns the result.

Format CALL *CALCDATE (*date*, *YY*, *MM*, *DD*)

Description Adds the specified displacement values (*YY*, *MM*, *DD*) to the year, month, and day specified in *date* and returns the result in *date*.

If the displacement values are negative, the date will be decremented.

The setting range for the *date* is Jan. 1, 1980 (80/01/01) to Dec. 31, 2079 (79/12/31). If the result of the addition exceeds these limits, the limit date will be returned.

The last day in a short month will be used when a month displacement (MM) changes the month from a long month to a short month that doesn't include the day specified in date. For example, when one month is added to 92/01/01, the result is 92/02/29 not 92/02/31.

Specified values: date="92/01/31", YY=0, MM=1, and DD=0
Returned value: date="92/02/29"

*SUBTHETA SUB THETA

(General-purpose Structural Subroutine)

Action Performs angle compensation.

Format CALL *SUBTHETA (*T1*, *T2*, *T3*)

Description Determines the difference (–180° to +180°) between two angles.

Specify the two angles in degrees in parameters *T1* and *T2*. The setting range is the range of numbers that OVL can handle.

The result of the calculation is stored in *T3* in degrees.

*GETTHETA GET THETA

(General-purpose Structural Subroutine)

Action Determines the slope of the line specified with two points.

Format CALL *GETTHETA (*X1*, *Y1*, *X2*, *Y2*, *T*)

Description Determines the slope of a line.

Specify two points on the line with coordinates (*X1*,*Y1*) and (*X2*,*Y2*). The result of the calculation is stored in *T* in degrees. A value of 0 will be returned if the same point is specified for both (*X1*,*Y1*) and (*X2*,*Y2*).

SECTION 7
Sample Programs

This section provides sample programs prepared mainly for measurement processing. Refer to these sample programs and become familiar with the specifications of each command and function to create user's programs for actual application. Brief comments are added as reference to important commands.

Introduction

The latest version of the sample programs are stored in the ROM disk in the IMP Unit. They can be easily executed by inputting as follows.

Example) 7.1 Searching for 1 Model (Single Processing)

Input this number when executing this program. Input “01” to “09” for 1 to 9.

RUN“D:S01.BAS”

These sample programs may not run properly depending on the registration state of the model. If the program doesn’t run properly, execute SYSINIT3 to initialize the model and then execute the sample program.

7-1 Searching for 1 Model: Single Processing

The following program registers a model image in the search model 0, searches the model based on the input image, and indicates the searched position in real-time by the cross cursor. Press the ENT Key on the Console during searching to change the registered search model.

```
10000 ’ ====================================================
10010 ’ Data definition
10020 ’ ====================================================
10030 X1=256:Y1=240           ’Initial rectangle region value
10040 X2=X1+63:Y2=Y1+63       ’Rectangle region size
10050 X=511 :Y=511            ’Initial cursor position
10060 ’ ====================================================
10070 ’ Main program
10080 ’ ====================================================
10090 ’ ----------------------------------------------------
10100 ’ Initial settings
10110 ’ ----------------------------------------------------
10120 SYSINIT 1:SYSINIT 2     ’System initialization
10130 LOCATE ,,0              ’Character cursor OFF
10140 DISPLAY 31              ’Density image display
10150 SMCLEAR 0,0             ’Search model 0 initialized
10160 GOSUB *SETMODEL         ’Model registration
10170 LOCATE 0,0:PRINT ” Search position:”
10180 LOCATE 0,1:PRINT ” Correlation    :”
10190 ’ ----------------------------------------------------
10200 ’ Main routine
10210 ’ ----------------------------------------------------
10220 DO                      ’Unlimited loop
10230   SMRUN 0,0,0           ’Search model 0 only, no sort
10240   X=SMDATA(0,8)         ’X coordinate for max. evaluation value
10250   Y=SMDATA(0,10)        ’Y coordinate for max. evaluation value
10260   R=SMDATA(0,12)        ’Max. evaluation value
10270   CLS 1                 ’Cursor deleted
10280   CURSOR X,Y,0,1        ’Cursor displayed
10290   LOCATE 18,0:PRINT USING ”(### ###)”;X,Y
10300   LOCATE 18,1:PRINT USING ”###.###”;R
10310   K=KEYIN(0)
10320   IF K=&H10 THEN        ’Press ENT to register a new model
10330     GOSUB *SETMODEL
10340   END IF
```

```
10350 LOOP WHILE (K<>&H20)      'Press ESC to end
10360 LOCATE ,,1              'Character cursor ON
10370 END
10380 ' ------------------------------------------------------
10390 ' Model registration
10400 ' ------------------------------------------------------
10410 *SETMODEL
10420   CALL *BTMMSG ("Set the model region")
10430   DO
10440     CALL *SETBOX(X1,Y1,X2,Y2,KY,0,71,23,67,19)
10450   LOOP WHILE (KY<>1)            'Waiting until ENT is pressed
10460   K=KEYIN(0)                    'Dummy input
10470   CALL *BTMMSG("ENT:Model registration  ESC:End")
10480   VIDEOIN 0,0                   'Video bus to image memory 0
10490   VDWAIT 3                      'Waiting for image input to be completed
10500   SX=(X2-X1)/2                  'Search pos in the center of model
10510   SY=(Y2-Y1)/2
10520   SMPUT  0,0,X1,Y1,X2,Y2,SX,SY  'SM 0 registration
10530   SMMODE 0,0,0,70,0             'Search model 0
10540                                 'Density correlation, min. and max. only
10550   SMAREA 0,0,0,511,483          'Search entire screen
10560  RETURN
```

7-2 Searching Multiple Images for 1 Model: Single Processing

The following program registers a model image in the search model 0, searches multiple positions of images whose correlation values with the model image exceed 70, and indicates searched positions in real-time. Press the ENT Key on the Console during searching to change the registered search model.

```
10000 ' ======================================================
10010 ' Data definition
10020 ' ======================================================
10030 X1=256:Y1=240           'Initial rectangle region value
10040 X2=X1+63:Y2=Y1+63       'Rectangle region size
10050 N=128                   'Max. no. of candidate points
10060 DIM X(N),Y(N)           'Array for candidate pt coordinates
10070 ' ======================================================
10080 ' Main program
10090 ' ======================================================
10100 ' ------------------------------------------------------
10110 ' Initial settings
10120 ' ------------------------------------------------------
10130 SYSINIT 1:SYSINIT 2     'System initialization
10140 LOCATE ,,0              'Character cursor OFF
10150 DISPLAY 31              'Density image display
10160 SMCLEAR 0,0             'Search model 0 initialized
10170 GOSUB *SETMODEL         'Model registration
10180 ' ------------------------------------------------------
10190 ' Main routine
10200 ' ------------------------------------------------------
10210 DO                      'Unlimited loop
10220   SMRUN 0,0,0           'Search model 0 only, no sort
10230   SMMDATA 0,0,X,Y       'Candidate pts for model 0
```

```
10240   N=SMDATA(0,0)         'No. of the canditate points obtained
10250   LOCATE 0,0            'No. of searches
10260   PRINT USING "### ";N
10270   CLS 1                 'Cursor deleted
10280   FOR I=0 TO N-1
10290     CURSOR X(I),Y(I),0,1'Cursor displayed
10300   NEXT
10310   K=KEYIN(0)
10320   IF K=&H10 THEN        'Press ENT to register a new model
10330     GOSUB *SETMODEL
10340   END IF
10350 LOOP WHILE (K<>&H20)    'Press ESC to end
10360 LOCATE ,,1              'Character cursor ON
10370 END
10380 ' ------------------------------------------------------
10390 ' Model registration
10400 ' ------------------------------------------------------
10410 *SETMODEL
10420   CALL *BTMMSG("Set the model region")
10430   DO
10440     CALL *SETBOX(X1,Y1,X2,Y2,KY,0,71,23,67,19)
10450   LOOP WHILE (KY<>1)            'Waiting until ENT is pressed
10460   K=KEYIN(0)                    'Dummy input
10470   CALL *BTMMSG("ENT:Model registration  ESC:End")
10480   VIDEOIN 0,0                   'Video bus to image memory 0
10490   VDWAIT 3                      'Waiting for image input to be completed
10500   SX=(X2-X1)/2                  'Search pos in the center of model
10510   SY=(Y2-Y1)/2
10520   SMPUT  0,0,X1,Y1,X2,Y2,SX,SY  'SM 0 registration
10530   SMMODE 0,0,128,70,0           'Search model 0
10540                                 'Density correlation, the no. of searches
is 128
10550   SMAREA 0,0,0,511,483          'Search entire screen
10560  RETURN
```

7-3 Searching Rotated Images for 1 Model: Single Processing

The following program registers a model image in the search model 0, searches the model based on the input image, and indicates the searched position and rotated angle in real-time by the cross cursor.

Press the ENT Key on the Console during searching to change the registered search model.

```
10000 ' ======================================================
10010 ' Data definition
10020 ' ======================================================
10030 X1=256:Y1=240           'Initial rectangle region value
10040 X2=X1+63:Y2=Y1+63       'Rectangle region size
10050 X=511 :Y=511 :T=0       'Initial cursor position
10060 DT=5                    'Search angle step
10070 DIM SMD(8)              'Array for search results
10080 ' ======================================================
10090 ' Main program
10100 ' ======================================================
```

```
10110 ' ------------------------------------------------------
10120 ' Initial settings
10130 ' ------------------------------------------------------
10140 SYSINIT 1:SYSINIT 2     'System initialization
10150 LOCATE ,,0              'Character cursor OFF
10160 DISPLAY 31              'Density image display
10170 SMCLEAR 0,435           'All search models initialized
10180 GOSUB *SETMODEL         'Model registration
10190 LOCATE 0,0:PRINT "Search position:"
10200 LOCATE 0,1:PRINT "Search angle   :"
10210 LOCATE 0,2:PRINT "Correlation    :"
10220 ' ------------------------------------------------------
10230 ' Main routine
10240 ' ------------------------------------------------------
10250 DO                      'Unlimited loop
10260   SMGRUN 0,0,0          'Search model 0 only, no sort
10270   SMGDATA 0,2,SMD       'Search results
10280   X=SMD(1)              'X coordinate for max. evaluation value
10290   Y=SMD(2)              'Y coordinate for max. evaluation value
10300   R=SMD(3)              'Maximum evaluation value
10310   T=SMD(4)*DT           'Search angle
10320   IF T>180 THEN T=T-360 'Adjusted to -1805 through +1805
10330   CLS 1                 'Cursor deleted
10340   CURSOR X,Y,T,1        'Cursor displayed
10350   LOCATE 18,0:PRINT USING "(### ###)";X,Y
10360   LOCATE 18,1:PRINT USING "####";T
10370   LOCATE 18,2:PRINT R
10380   K=KEYIN(0)
10390   IF K=&H10 THEN        'Press ENT to register a new model
10400     GOSUB *SETMODEL
10410   END IF
10420 LOOP WHILE (K<>&H20)    'Press ESC to end
10430 LOCATE ,,1              'Character cursor OFF
10440 END
10450 ' ------------------------------------------------------
10460 ' Model registration
10470 ' ------------------------------------------------------
10480 *SETMODEL
10490   CALL *BTMMSG("Set the model region")
10500   DO
10510     CALL *SETBOX(X1,Y1,X2,Y2,KY,0,71,23,67,19)
10520   LOOP WHILE (KY<>1)            'Waiting until ENT is pressed
10530   K=KEYIN(0)                    'Dummy input
10540   CALL *BTMMSG("ENT:Model registration  ESC:End")
10550   VIDEOIN 0,0                   'Video bus to image memory 0
10560   VDWAIT 3                      'Waiting for image input to be completed
10570   SX=(X2-X1)/2                  'Search pos in the center of model
10580   SY=(Y2-Y1)/2
10590   SMGCLEAR 0                    'Models in Group 0 initialized
10600   SMPUT 0,0,X1,Y1,X2,Y2,SX,SY   'SM 0 registration
10610   SMROTATE 0,0,0,,,0,360-DT,DT  'Rotation models registered
10620   SMGROUP 0,0                   'Group registration
```

```
10630   SMGMODE 0,0,0,70,0            'Search model group 0
10640                                 'Density correlation, min. and max. only
10650   SMGAREA 0,0,0,511,483         'Search entire screen
10660  RETURN
```

7-4 Registering a Search Model

The following program registers a search model.

```
10000 ' ======================================================
10010 ' Data definition
10020 ' ======================================================
10030 SMN=0                    'Initial search model no.
10040 X1=256:Y1=240            'Initial model region
10050 X2=X1+63:Y2=Y1+63        'Initial model size
10060 MSG1$="UP:Model no.+1   DOWN:Model no.-1"
10070 MSG2$="ENT:Model reg  SHIFT+ENT:Model del  ESC:End"
10080 ' ======================================================
10090 ' Main program
10100 ' ======================================================
10110 ' ------------------------------------------------------
10120 ' Initial settings
10130 ' ------------------------------------------------------
10140 SYSINIT 1:SYSINIT 2     'System initialization
10150 LOCATE ,,0              'Character cursor OFF
10160 DISPLAY 31              'Density image display
10170 LOCATE 0,20:PRINT "Model" +RIGHT$(STR$(1000+SMN),3)
10180 LOCATE 0,21:PRINT MSG1$;
10190 LOCATE 0,22:PRINT MSG2$;
10200 ' ------------------------------------------------------
10210 ' Main routine
10220 ' ------------------------------------------------------
10230 DO
10240   K=KEYIN(1)            'Waiting for key input
10250   SELECT K
10260   CASE 1                'UP: Model no. +1
10270     SMN=SMN+1:GOSUB *SETSMN
10280   CASE 8                'DOWN: Model no. -1
10290     SMN=SMN-1:GOSUB *SETSMN
10300   CASE &H90             'SHIFT+ENT: Delete
10310     SMCLEAR SMN
10320   CASE &H10             'ENT: Register
10330     GOSUB *SETMODEL
10340   END SELECT
10350 LOOP WHILE (K<>&H20)    'Press ESC to end
10360 CLS
10370 LOCATE ,,1              'Character cursor OFF
10380 END
10390 ' ------------------------------------------------------
10400 ' Search model no. change
10410 ' ------------------------------------------------------
10420 *SETSMN
10430   IF SMN>435 THEN SMN=0
10440   IF SMN<0   THEN SMN=435
```

```
10450   LOCATE 0,20:PRINT ”Model”+RIGHT$(STR$(1000+SMN),3)
10460  RETURN
10470 ’ ------------------------------------------------------
10480 ’ Search model registration
10490 ’ ------------------------------------------------------
10500 *SETMODEL
10510   CALL *BTMMSG(”Set the model region”)
10520   DO
10530     CALL *SETBOX(X1,Y1,X2,Y2,KY,0,71,23,67,19)
10540   LOOP WHILE (KY<>1)            ’Waiting until ENT is pressed
10550   CALL *BTMMSG(””)              ’Delete message
10560   VIDEOIN 0,0                   ’Video bus to image memory 0
10570   VDWAIT 3                      ’Waiting for image input to be completed
10580   SX=(X2–X1)/2                  ’Reference position in the center of model
10590   SY=(Y2–Y1)/2
10600   SMPUT   SMN,0,X1,Y1,X2,Y2,SX,SY ’Search model registered
10610   SMMODE  SMN,0,0                 ’Density correlation, min. and max. only
10620   SMAREA  SMN,0,0,511,483         ’Search entire screen
10630  RETURN
```

7-5 Registering an ROI Model

The following program registers an ROI model.

```
10000 ’ ======================================================
10010 ’ Data definition
10020 ’ ======================================================
10030 RMN=0                   ’Initial ROI model number
10040 X1=256:Y1=240           ’Initial model region
10050 X2=X1+63:Y2=Y1+63       ’Initial model size
10060 MSG1$=”UP:Model no.+1  DOWN:Model no.–1”
10070 MSG2$=”ENT:Model reg  SHIFT+ENT:Model del  ESC:End”
10080 ’ ======================================================
10090 ’ Main program
10100 ’ ======================================================
10110 ’ ------------------------------------------------------
10120 ’ IROI initialization
10130 ’ ------------------------------------------------------
10140 SYSINIT 1:SYSINIT 2     ’System initialization
10150 LOCATE ,,0              ’Character cursor OFF
10160 DISPLAY 31              ’Density image display
10170 LOCATE 0,20:PRINT ”Model”+RIGHT$(STR$(10000+RMN),4)
10180 LOCATE 0,21:PRINT MSG1$;
10190 LOCATE 0,22:PRINT MSG2$;
10200 ’ ------------------------------------------------------
10210 ’ Main routine
10220 ’ ------------------------------------------------------
10230 DO
10240   K=KEYIN(1)            ’Waiting for key input
10250   SELECT K
10260   CASE 1                ’UP: Model no. +1
10270     RMN=RMN+1:GOSUB *SETRMN
10280   CASE 8                ’DOWN: Model no. –1
10290     RMN=RMN–1:GOSUB *SETRMN
```

```
10300   CASE &H90             'SHIFT+ENT:SHIFT+ENT: Delete
10310     RMCLEAR RMN
10320   CASE &H10             'ENT:ENT: Register
10330     GOSUB *SETMODEL
10340   END SELECT
10350 LOOP WHILE (K<>&H20)    'Press ESC to end
10360 CLS
10370 LOCATE ,,1              'Character cursor OFF
10380 END
10390 ' ------------------------------------------------------
10400 ' ROI model no. change
10410 ' ------------------------------------------------------
10420 *SETRMN
10430   IF RMN>1023 THEN RMN=0
10440   IF RMN<0    THEN RMN=1023
10450   LOCATE 0,20:PRINT "Model"+RIGHT$(STR$(10000+RMN),4)
10460  RETURN
10470 ' ------------------------------------------------------
10480 ' ROI model registration
10490 ' ------------------------------------------------------
10500 *SETMODEL
10510   CALL *BTMMSG("Set the model region")
10520   DO
10530     CALL *SETBOX(X1,Y1,X2,Y2,KY)    'Rectangle region setting
10540   LOOP WHILE (KY<>1)                'Waiting until ENT is pressed
10550   CALL *BTMMSG("")                  'Delete message
10560   VIDEOIN 0,0                       'Video bus to image memory 0
10570   VDWAIT 3                          'Waiting for image input to be com-
pleted
10580   RMPUT  RMN,0,X1,Y1,X2,Y2          'ROI model registered
10590   RMMODE RMN,3                      'Density correlation
10600  RETURN
```

7-6 Registering a Search Model and an ROI Model

The following program registers a search model and an ROI model simultaneously. The program can be used to register a model for high-precision searching.

```
10000 ' ======================================================
10010 ' Data definition
10020 ' ======================================================
10030 RMN=0                   'Initial ROI model number
10040 X1=256:Y1=240           'Initial model region
10050 X2=X1+63:Y2=Y1+63       'Initial model size
10060 MSG1$="UP:Model no.+1  DOWN:Model no.-1"
10070 MSG2$="ENT:Model reg  SHIFT+ENT:Model del  ESC: End"
10080 DIM AX1(1),AY1(1),AX2(1),AY2(1)
10090 DIM BX1(0),BY1(0),BX2(0),BY2(0)
10100 ' ======================================================
10110 ' Main program
10120 ' ======================================================
10130 ' ------------------------------------------------------
10140 ' IROI initialization
```

```
10150 ' ------------------------------------------------------
10160 SYSINIT 1:SYSINIT 2     'System initialization
10170 LOCATE ,,0              'Character cursor OFF
10180 DISPLAY 31              'Density image display
10190 LOCATE 0,20:PRINT "Model"+RIGHT$(STR$(1000+RMN),3)
10200 LOCATE 0,21:PRINT MSG1$;
10210 LOCATE 0,22:PRINT MSG2$;
10220 ' ------------------------------------------------------
10230 ' Main routine
10240 ' ------------------------------------------------------
10250 DO
10260   K=KEYIN(1)
10270   SELECT K
10280   CASE 1                'UP: Model no. +1
10290     RMN=RMN+1:GOSUB *SETRMN
10300   CASE 8                'DOWN: Model no. -1
10310     RMN=RMN-1:GOSUB *SETRMN
10320   CASE &H90             'SHIFT+ENT:SHIFT+ENT: Delete
10330     RMCLEAR RMN
10340   CASE &H10             'ENT:ENT: Register
10350     GOSUB *SETMODEL
10360   END SELECT
10370 LOOP WHILE (K<>&H20)    'Press ESC to end
10380 CLS
10390 LOCATE ,,1              'Character cursor ON
10400 END
10410 ' ------------------------------------------------------
10420 ' ROI model no. change
10430 ' ------------------------------------------------------
10440 *SETRMN
10450   IF RMN>435  THEN RMN=0
10460   IF RMN<0    THEN RMN=435
10470   LOCATE 0,20:PRINT "Model"+RIGHT$(STR$(1000+RMN),3)
10480  RETURN
10490 ' ------------------------------------------------------
10500 ' ROI model registration
10510 ' ------------------------------------------------------
10520 *SETMODEL
10530   CALL *BTMMSG("Set the model region")
10540   DO
10550     CALL *SETBOX(X1,Y1,X2,Y2,KY)
10560   LOOP WHILE (KY<>1)            'Waiting until ENT is pressed
10570   CALL *BTMMSG("")              'Delete message
10580   VIDEOIN 0,0                   'Video bus to image memory 0
10590   VDWAIT 3                      'Waiting for image input to be completed
10600   RMPUT   RMN,0,X1,Y1,X2,Y2     'ROI model registered
10610   RMMODE  RMN,3                 'Density correlation
10620   RMORIGIN RMN,FIX((X1-X2)/2),FIX((Y1-Y2)/2)
10630   SMSELECT 0,X1,Y1,X2,Y2,1,AX1,AY1,AX2,AY2
10640   SMSELECT2 0,AX1(0),AY1(0),AX2(0),AY2(0),BX1,BY1,BX2,BY2
10650   RX1=BX1(0)-X1:RY1=BY1(0)-Y1
10660   RX2=BX2(0)-X1:RY2=BY2(0)-Y1
```

```
10670   RMTOSM RMN,RMN,RX1,RY1,RX2,RY2,-RMINFO(RMN,15),-RMINFO(RMN,16)
10680   SMMODE RMN,0,0,0              'Density correlation, min. and max. only
10690  RETURN
```

7-7 Searching Multiple Models: Single Processing

The following program searches the search models 0 to 11 (12 search models) in real-time and indicates their positions.

```
10000 ' ======================================================
10010 ' Data definition
10020 ' ======================================================
10030 DIM X(12),Y(12)         'Array for Search position
10040 X1=256:Y1=240           'Initial rectangle region value
10050 X2=X1+63:Y2=Y1+63       'Rectangle region size
10060 ' ======================================================
10070 ' Main program
10080 ' ======================================================
10090 ' ------------------------------------------------------
10100 ' Initial settings
10110 ' ------------------------------------------------------
10120 SYSINIT 1:SYSINIT 2     'System initialization
10130 LOCATE ,,0              'Character cursor OFF
10140 DISPLAY 31              'Density image display
10150 ' ------------------------------------------------------
10160 ' Search group 0 not registered
10170 ' ------------------------------------------------------
10180 N=SMGINFO(0,0)          'No. of registrations for search grp 0
10190 IF N<>0 THEN            'Registration deleted if number is not 0
10200   FOR I=0 TO N-1
10210     SMGDEL SMGINFO(0,1,0)
10220   NEXT
10230 ELSE
10240   GOSUB *SETMODEL       'Model registered if number is 0
10250 END IF
10260 ' ------------------------------------------------------
10270 ' Search model 0 to 11 registered in search group 0
10280 ' ------------------------------------------------------
10290 FOR SM=0 TO 11          'Registered models registered in group
10300   IF SMINFO(SM,0)=1 THEN SMGROUP 0,SM
10310 NEXT
10320 SMGMODE 0,0,1           'Density correlation, 1 search
10330 SMSORT 1                'Search models sorted in group
10340 ' ------------------------------------------------------
10350 ' Main routine
10360 ' ------------------------------------------------------
10370 CALL *BTMMSG("ENT:Model registration  ESC:End")
10380 DO
10390   SMGRUN 0,0,0          'Group 0 searched without sorting
10400   CLS 1                 'Graphic memory cleared
10410   SMGDATA 0,0,X,Y       'Candidate point data in group
10420   FOR SM=0 TO 11
10430     IF SMINFO(SM,0)=1 THEN
10440       CURSOR X(SM),Y(SM),0,1    'Cursor displayed
```

```
10450     END IF
10460   NEXT
10470   K=KEYIN(0)
10480   IF K=&H10 THEN        'Press ENT to register a new model
10490     GOSUB *SETMODEL
10500     CALL *BTMMSG("ENT:Model registration  ESC:End")
10510   END IF
10520 LOOP WHILE (K<>&H20)    'Press ESC to end
10530 LOCATE ,,1              'Character cursor ON
10540 END
10550 ' ------------------------------------------------------
10560 ' Search model registration
10570 ' ------------------------------------------------------
10580 *SETMODEL
10590   CLS 1
10600   FOR SMN=0 TO 11
10610     M$="Set the model region of model " +STR$(SMN)
10620     M$=M$+"  ESC:End registration"
10630     CALL *BTMMSG(M$)
10640     IF SMINFO(SMN,0)=1 THEN     'If model registration completed
10650       X1=SMINFO(SMN,1)          'obtain coordinates of rectangle
10660       Y1=SMINFO(SMN,2)          'region when model was registered
10670       X2=X1+SMINFO(SMN,3)-1
10680       Y2=Y1+SMINFO(SMN,4)-1
10690     END IF
10700     CALL *SETBOX(X1,Y1,X2,Y2,KY,0,71,23,67,19)
10710     IF KY=1 THEN                'Register if ENT is pressed
10720       VIDEOIN 0,0               'Video bus to image memory 0
10730       VDWAIT 3                  'Waiting for image input to be completed
10740       SX=(X2-X1)/2              'Reference position in the center of model
10750       SY=(Y2-Y1)/2
10760       SMPUT   SMN,0,X1,Y1,X2,Y2,SX,SY 'Search model registered
10770       SMMODE  SMN,0,1           'Density correlation, one search
10780       SMAREA  SMN,0,0,511,483   'Search entire screen
10790       SMGROUP 0,SMN            'Register Model SMN in group
10800     ELSE
10810       EXIT FOR
10820     END IF
10830   NEXT
10840   WHILE (KEYIN(0)):WEND         'End registration if ESC pressed.
10850   SMSORT 1                      'Search models sorted in group
10860  RETURN
```

7-8 Searching Multiple Models: Sequential Processing

The following program searches the search models 0 to 11 (12 search models) in real-time and indicates their positions. Shown below is the sample program No.07 processed in the sequential processing.

```
10000 ' ======================================================
10010 ' Data definition
10020 ' ======================================================
10030 DIM X(12),Y(12)         'Array for Search position
10040 X1=256:Y1=240           'Initial rectangle region value
```

```
10050 X2=X1+63:Y2=Y1+63       'Rectangle region size
10060 ' ============================================================
10070 ' Main program
10080 ' ============================================================
10090 ' ------------------------------------------------------------
10100 ' Initial settings
10110 ' ------------------------------------------------------------
10120 SYSINIT 1:SYSINIT 2     'System initialization
10130 LOCATE ,,0              'Character cursor OFF
10140 DISPLAY 31              'Density image display
10150 ' ------------------------------------------------------------
10160 ' Search group 0 not registered
10170 ' ------------------------------------------------------------
10180 N=SMGINFO(0,0)          'No. of registrations
10190 IF N<>0 THEN            'Registration deleted if number is not 0
10200   FOR I=0 TO N-1
10210     SMGDEL SMGINFO(0,1,0)
10220   NEXT
10230 ELSE
10240   GOSUB *SETMODEL       'Model registered if number is 0
10250 END IF
10260 ' ------------------------------------------------------------
10270 ' Search model 0 to 11 registered in search group 0
10280 ' ------------------------------------------------------------
10290 FOR SM=0 TO 11          'Registered models registered in group
10300   IF SMINFO(SM,0)=1 THEN SMGROUP 0,SM
10310 NEXT
10320 SMGMODE 0,0,1           'Density correlation, 1 search
10330 SMSORT 1                'Search models sorted in group
10340 ' ------------------------------------------------------------
10350 ' Main routine
10360 ' ------------------------------------------------------------
10370 ON SQMEAS GOSUB *SQINT  'Define sequence interrupt
10380 SQMEAS ON               'Enable sequence interrupt
10390 SQSET 0,"M2 S0"         'Search group 0
10400 SQRUN 3,0,0             'Repeatedly measure unit 0
10410 CALL *BTMMSG("ENT:Model registration  ESC:End")
10420 DO
10430   K=KEYIN(0)
10440   IF K=&H10 THEN        'Register model if ENT is pressed
10450     SQRUN 0             'Stop sequence to register model
10460     GOSUB *SETMODEL
10470     CALL *BTMMSG("ENT:Model registration  ESC:End")
10480     SQRUN 3,0,0         'Restart sequence-type processing
10490   END IF
10500 LOOP WHILE (K<>&H20)    'Press ESC to end
10510 SQRUN 0                 'Stop sequence-type processing
10520 LOCATE ,,1              'Character cursor ON
10530 END
10540 ' ------------------------------------------------------------
10550 ' Sequence interrupt
10560 ' ------------------------------------------------------------
```

```
10570 *SQINT
10580   CLS 1                 'Graphic memory cleared
10590   SMGDATA 0,0,X,Y       'Candidate point data in group
10600   FOR SM=0 TO 11
10610     IF SMINFO(SM,0)=1 THEN
10620       CURSOR X(SM),Y(SM),0,1    'Cursor displayed
10630     END IF
10640   NEXT
10650  RETURN
10660 ' -----------------------------------------------------
10670 ' Search model registration
10680 ' -----------------------------------------------------
10690 *SETMODEL
10700   CLS 1                         'Cursor deleted
10710   FOR SMN=0 TO 11
10720     M$="Set the model region of model "+STR$(SMN)
10730     M$=M$+"  ESC:End registration"
10740     CALL *BTMMSG(M$)
10750     IF SMINFO(SMN,0)=1 THEN     'If model registration completed
10760       X1=SMINFO(SMN,1)          'obtain coordinates of rectangle
10770       Y1=SMINFO(SMN,2)          'region when model was registered
10780       X2=X1+SMINFO(SMN,3)-1
10790       Y2=Y1+SMINFO(SMN,4)-1
10800     END IF
10810     CALL *SETBOX(X1,Y1,X2,Y2,KY,0,71,23,67,19)
10820     IF KY=1 THEN                'Register if ENT is pressed
10830       VIDEOIN 0,0               'Video bus to image memory 0
10840       VDWAIT 3                  'Waiting for image input to be completed
10850       SX=(X2-X1)/2              'Reference position in the center of model
10860       SY=(Y2-Y1)/2
10870       SMPUT   SMN,0,X1,Y1,X2,Y2,SX,SY 'Search model registered
10880       SMMODE  SMN,0,1           'Density correlation, min. and max. only
10890       SMAREA  SMN,0,0,511,483   'Search entire screen
10900       SMGROUP 0,SMN             'Register Model SMN in group
10910     ELSE                        'End registration if ESC pressed.
10920       EXIT FOR
10930     END IF
10940   NEXT
10950   WHILE (KEYIN(0)):WEND         'Wait for key to be released
10960   SMSORT 1                      'Search models sorted in group
10970  RETURN
```

7-9 Searching Using Multiple Cameras: Sequential Processing

The following program searches the search model 0 using camera 0 and the search model 1 using camera 1, and also indicates the searched positions. Use the Console key for executing the search.

```
10000 ' =====================================================
10010 ' Data definition
10020 ' =====================================================
10030 X1=256  :Y1=240         'Initial rectangle region value
```

```
10040 X2=X1+63:Y2=Y1+63       'Rectangle region size
10050 ' ========================================================
10060 ' Main program
10070 ' ========================================================
10080 ' --------------------------------------------------------
10090 ' Initial settings
10100 ' --------------------------------------------------------
10110 SYSINIT 1:SYSINIT 2     'System initialization
10120 LOCATE ,,0              'Character cursor OFF
10130 DISPLAY 31              'Density image display
10140 ' --------------------------------------------------------
10150 ' Search group 0,1 not registered.
10160 ' --------------------------------------------------------
10170 FOR SG=0 TO 1
10180   N=SMGINFO(SG,0)       'No. of registrations
10190   IF N<>0 THEN          'Registration deleted if number is not 0
10200     FOR I=0 TO N-1
10210       SMGDEL SMGINFO(SG,1,0)
10220     NEXT
10230   END IF
10240 NEXT
10250 ' --------------------------------------------------------
10260 ' Search model 0,1 will be registered.
10270 ' --------------------------------------------------------
10280 GOSUB *SETMODEL         'Model registration
10290 SMGROUP 0,0             'Search model 0 into search group 0
10300 SMGROUP 1,1             'Search model 1 into search group 1
10310 ' --------------------------------------------------------
10320 ' Main routine
10330 ' --------------------------------------------------------
10340 SQSET 0,"M2 S0","C0"      'Unit 0: search with group 0
10350 SQSET 1,"M2 S1","C1"      'Unit 1: search with group 1
10360 ON SQMEAS(1) GOSUB *SQINT 'Interrupt after Unit 1
10370 SQMEAS(1) ON
10380 CALL *BTMMSG ("ENT:Measure  ESC:End")
10390 DO
10400   K=KEYIN(1)
10410   IF K=&H10 THEN SQRUN 1,0,1
10420 LOOP WHILE (K<>&H20)      'Press ESC to end
10430 LOCATE ,,1                'Character cursor ON
10440 END
10450 ' --------------------------------------------------------
10460 ' Sequence interrupt
10470 ' --------------------------------------------------------
10480 *SQINT
10490    LOCATE 0,0
10500    PRINT USING "CAMERA 0 (X Y)=(###, ###)";SMDATA(0,8),SMDATA(0,10)
10510    LOCATE 0,1
10520    PRINT USING "CAMERA 1 (X Y)=(###, ###)";SMDATA(1,8),SMDATA(1,10)
10530  RETURN
10540 ' --------------------------------------------------------
10550 ' Search model registration
```

```
10560 ' ------------------------------------------------
10570 *SETMODEL
10580   FOR SMN=0 TO 1
10590     CAMERA SMN                  'Switch cameras according to model
10600     M$="Set the model region of model "+STR$(SMN)
10610     CALL *BTMMSG(M$)
10620     CALL *SETBOX(X1,Y1,X2,Y2,KY,0,71,23,67,19)
10630     IF KY=1 THEN                'Register if ENT is pressed
10640       VIDEOIN 0,0               'Video bus to image memory 0
10650       VDWAIT 3                  'Waiting for image input to be completed
10660       SX=(X2-X1)/2              'Reference position in the center of model
10670       SY=(Y2-Y1)/2
10680       SMPUT   SMN,0,X1,Y1,X2,Y2,SX,SY 'Search model registered
10690       SMMODE  SMN,0,0           'Density correlation, min. and max. only
10700       SMAREA  SMN,0,0,511,483   'Search entire screen
10710       SMGROUP 0,SMN             'Register Model SMN in group
10720       CNT=CNT+1                 'No. of models registered
10730     ELSE                        'End registration if ESC pressed.
10740       EXIT FOR
10750     END IF
10760   NEXT
10770  RETURN
```

7-10 High-precision Searching: Sequential Processing

The following program provides highly precise positions using the search model 0 and ROI model 0. Use the Console key for executing the search.

A search model and an ROI model must be registered in advance using the sample program No. 6.

Refer to *7-6 Registering a Search Model and an ROI Model*.

```
10000 ' ====================================================
10010 ' Data definition
10020 ' ====================================================
10030 DIM RM(2)               'Array for setting high-precision search
10040 DIM Q1(1),Q2(1)         'Array for setting SQMODE
10050 DIM RD1(3),RD2(3)       'Array for ROI results
10060 DIM RD4(3)              'Array for ROI results
10070 DIM AX1(1),AY1(1),AX2(1),AY2(1)
10080 DIM BX1(1),BY1(1),BX2(1),BY2(1)
10090 X1=256:Y1=240           'Initial model region
10100 X2=X1+63:Y2=Y1+63       'Initial model size
10110 ' ====================================================
10120 ' Main program
10130 ' ====================================================
10140 ' ------------------------------------------------------
10150 ' Initial settings
10160 ' ------------------------------------------------------
10170 SYSINIT 1:SYSINIT 2     'System initialization
10180 LOCATE ,,0              'Character cursor OFF
10190 DISPLAY 31              'Density image display
10200 IF (SMINFO(0,0) AND RMINFO(0,0))=0 THEN
10210   GOSUB *SETMODEL       'Register model if not registered
```

```
10220 END IF
10230 ' ------------------------------------------------------
10240 ' Measurement condition settings
10250 ' ------------------------------------------------------
10260 SMGROUP 0,0             'Register model 0 in group 0
10270 RM(0)=5                 'X size of high-precision search region
10280 RM(1)=5                 'Y size of high-precision search region
10290 RM(2)=0                 'High-precision search (frame mode)
10300 RMMODE2 0,1,RM          'High-precision search for model 0, no rotation
10310 Q1(0)=0                 'Search model number
10320 Q2(0)=0                 'ROI model number
10330 SQMODE 0,9,1,Q1,Q2      'Search eval value max. for ROI model
10340 SQSET 0,"M2 S0 V-1"     'Unit 0: search with group 0
10350 ON SQMEAS GOSUB *SQINT
10360 SQMEAS ON
10370 ' ------------------------------------------------------
10380 ' Main routine
10390 ' ------------------------------------------------------
10400 CALL *BTMMSG("ENT:Model registration  ESC:End")
10410 SQRUN 3,0,0              'Continuous sequence-type processing
10420 DO
10430   K=KEYIN(0)
10440   IF K=&H10 THEN
10450     SQRUN 0             'Stop sequence to register model
10460     GOSUB *SETMODEL
10470     CALL *BTMMSG("ENT:Model registration  ESC:End")
10480     SQRUN 3,0,0         'Restart sequence-type processing
10490   END IF
10500 LOOP WHILE (K<>&H20)    'Press ESC to end
10510 SQRUN 0                 'Stop sequence-type processing
10520 LOCATE ,,1              'Character cursor ON
10530 END
10540 ' ------------------------------------------------------
10550 ' Sequence interrupt
10560 ' ------------------------------------------------------
10570 *SQINT
10580   CLS 1                 'Delete frame
10590   RMDATA 0,RD1,RD2,,RD4 'High-precision search results
10600   LOCATE 0,0            'Display results
10610   PRINT USING "Search position:###.### ###.###";RD1(0),RD2(0)
10620   PRINT USING "Correlation    :###.###";RD4(0)
10630   BOX
RD1(0)-RMINFO(0,1)/2,RD2(0)-RMINFO(0,2)/2,RD1(0)+RMINFO(0,1)/2,RD2(0)+RMINFO(0,2
)/2,1,,,1
10640  RETURN
10650 ' ------------------------------------------------------
10660 ' ROI model registration
10670 ' ------------------------------------------------------
10680 *SETMODEL
10690   CLS 1                       'Delete frame
10700   CALL *BTMMSG("Set the model region")
10710   IF RMINFO(0,0)=1 THEN       'If ROI model is registered
```

```
10720     X1=RMINFO(0,6)              'obtain coordinates of rectangle
10730     Y1=RMINFO(0,7)              'region when model was registered
10740     X2=RMINFO(0,8)
10750     Y2=RMINFO(0,9)
10760   END IF
10770   DO
10780     CALL *SETBOX(X1,Y1,X2,Y2,KY)
10790   LOOP WHILE (KY<>1)            'Waiting until ENT is pressed
10800   CALL *BTMMSG("")              'Delete message
10810   VIDEOIN 0,0                   'Video bus to image memory 0
10820   VDWAIT 3                      'Waiting for image input to be completed
10830   RMPUT 0,0,X1,Y1,X2,Y2         'ROI model registered
10840   RMMODE 0,3                    'Density correlation
10850   RMORIGIN 0,FIX((X1-X2)/2),FIX((Y1-Y2)/2)
10860   SMSELECT 0,X1,Y1,X2,Y2,1,AX1,AY1,AX2,AY2
10870   SMSELECT2 0,AX1(0),AY1(0),AX2(0),AY2(0),BX1,BY1,BX2,BY2
10880   RX1=BX1(0)-X1:RY1=BY1(0)-Y1
10890   RX2=BX2(0)-X1:RY2=BY2(0)-Y1
10900   RMTOSM 0,0,RX1,RY1,RX2,RY2,-RMINFO(0,15),-RMINFO(0,16)
10910   SMMODE 0,0,0,0                'Density correlation, min. and max. only
10920   SMGROUP 0,0                   'Register search model 0 in group
10930   WHILE (KEYIN(0)):WEND         'Wait for key to be released
10940  RETURN
```

7-11 Inspecting Multiple Positions after Positioning: Sequential Processing

The following program provides correlation values for the multiple positions (0 to 9) using the ROI model, after they are positioned using the search model.

```
10000 ' ======================================================
10010 ' Data definition
10020 ' ======================================================
10030 RC=10                     'The total number of ROI models
10040 DIM Q1(2)                 'Array for setting SQMODE
10050 DIM X%(RC),Y%(RC),RN%(RC) 'Array for results
10060 DIM R1(RC)                'Array for ROI results
10070 X1=256:Y1=240:X2=X1+60:Y2=Y1+60
10080 ' ======================================================
10090 ' Main program
10100 ' ======================================================
10110 ' ------------------------------------------------------
10120 ' Initial settings
10130 ' ------------------------------------------------------
10140 SYSINIT 1:SYSINIT 2     'System initialization
10150 LOCATE ,,0              'Character cursor OFF
10160 DISPLAY 31              'Density image display
10170 ' ------------------------------------------------------
10180 ' Search group 0 not registered
10190 ' ------------------------------------------------------
10200 N=SMGINFO(0,0)          'No. of registrations
10210 IF N<>0 THEN            'Registration deleted if number is not 0
10220   FOR I=0 TO N-1
```

```
10230     SMGDEL SMGINFO(0,1,0)
10240   NEXT
10250 ELSE
10260   GOSUB *SETSMODEL      'Register search model if 0 registrations
10270 END IF
10280 ' -----------------------------------------------------
10290 ' Search model 0 will be registered in search group 0.
10300 ' -----------------------------------------------------
10310 SMGROUP 0,0
10320 SMSORT 1
10330 ' -----------------------------------------------------
10340 ' ROI group 0 not registered.
10350 ' -----------------------------------------------------
10360 N=RMGINFO(0,0)          'No. of registrations
10370 IF N<>0 THEN            'Registration deleted if number is not 0
10380   FOR I=0 TO N-1
10390     RMGDEL 0,RMGINFO(0,1,0)
10400   NEXT
10410 ELSE
10420   GOSUB *SETRMODEL      'Register ROI model if 0 registrations
10430 END IF
10440 ' -----------------------------------------------------
10450 ' ROI model 0 to 9 will be registered in ROI group 0.
10460 ' -----------------------------------------------------
10470 FOR RMN=0 TO RC-1
10480   RMGROUP 0,RMN
10490 NEXT
10500 ' -----------------------------------------------------
10510 ' ROI result display position setting (FASTPRINT)
10520 ' -----------------------------------------------------
10530 FOR I=0 TO RC-1
10540   X%(I)=10:Y%(I)=I:RN%(I)=I
10550   LOCATE 0,I:PRINT "Model";I;"="
10560 NEXT
10570 GOSUB *DRAWWIN          'Draw ROI region
10580 GOSUB *SETSQMODE        'Set measurement mode
10590 ON SQMEAS GOSUB *SQINT
10600 SQMEAS ON
10610 SQRUN 3,0,0             'Continuous sequence measurement
10620 CALL *BTMMSG("ENT:Model registration  ESC:End")
10630 ' -----------------------------------------------------
10640 ' Main routine
10650 ' -----------------------------------------------------
10660 DO
10670   K=KEYIN(0)
10680   IF K=&H10 THEN
10690     SQRUN 0             'Stop sequence to register model
10700     GOSUB *SETSMODEL    'Search model registration
10710     GOSUB *SETRMODEL    'ROI model registration
10720     GOSUB *DRAWWIN      'Draw ROI region
10730     GOSUB *SETSQMODE    'Set measurement mode
10740     CALL *BTMMSG("ENT:Model registration  ESC:End")
```

```
10750     SQRUN 3,0,0          'Restart sequence measurement
10760    END IF
10770 LOOP WHILE (K<>&H20)    'Press ESC to end
10780 SQRUN 0
10790 LOCATE ,,1              'Character cursor ON
10800 CLS
10810 END
10820 ' ------------------------------------------------------
10830 ' Sequence interrupt
10840 ' ------------------------------------------------------
10850 *SQINT
10860   CLS 1                 'Cursor deleted
10870   RMMDATA RC,RN%,0,R1   'Obtain ROI correlation
10880   CALL *FASTPRINT(RC,X%,Y%,R1,"###.##")
10890   CURSOR SMDATA(0,8),SMDATA(0,10),0,1
10900   DX=SMDATA(0,8)-(SMINFO(0,1)+SMINFO(0,5))
10910   DY=SMDATA(0,10)-(SMINFO(0,2)+SMINFO(0,6))
10920   WSCROLL DX,DY
10930  RETURN
10940 ' ------------------------------------------------------
10950 ' Search model registration
10960 ' ------------------------------------------------------
10970 *SETSMODEL
10980   VSCROLL 0,0                  'Return to original drawing position
10990   CLS 1                        'Cursor deleted
11000   M$="Set the search model region."
11010   CALL *BTMMSG(M$)
11020   IF SMINFO(SMN,0)=1 THEN      'If model registration completed
11030     X1=SMINFO(SMN,1)           'obtain coordinates of rectangle
11040     Y1=SMINFO(SMN,2)           'region when model was registered
11050     X2=X1+SMINFO(SMN,3)-1
11060     Y2=Y1+SMINFO(SMN,4)-1
11070   END IF
11080   CALL *SETBOX(X1,Y1,X2,Y2,KY,0,71,23,67,19)
11090   IF KY=1 THEN                 'Register if ENT is pressed
11100     VIDEOIN 0,0                'Video bus to image memory 0
11110     VDWAIT 3                   'Waiting for image input to be completed
11120     SX=(X2-X1)/2               'Reference position in the center of model
11130     SY=(Y2-Y1)/2
11140     SMPUT   SMN,0,X1,Y1,X2,Y2,SX,SY 'Search model registered
11150     SMMODE  SMN,0,1            'Density correlation, min. and max. only
11160     SMAREA  SMN,0,0,511,483    'Search entire screen
11170     SMGROUP 0,SMN              'Register Model SMN in group
11180   ELSE                         'End registration if ESC pressed.
11190     EXIT FOR
11200   END IF
11210   WHILE (KEYIN(0)):WEND        'Wait for key to be released
11220  RETURN
11230 ' ------------------------------------------------------
11240 ' ROI model registration
11250 ' ------------------------------------------------------
11260 *SETRMODEL
```

```
11270   VSCROLL 0,0                   'Return to original drawing position
11280   CLS 1                         'Cursor deleted
11290   FOR RMN=0 TO RC-1
11300     M$="Set the model region of ROI model "+STR$(RMN)
11310     M$=M$+"ESC:End registration"
11320     CALL *BTMMSG(M$)
11330     IF RMINFO(RMN,0)=1 THEN     'If model registration completed
11340       X1=RMINFO(RMN,6)          'obtain coordinates of rectangle
11350       Y1=RMINFO(RMN,7)          'region when model was registered
11360       X2=RMINFO(RMN,8)
11370       Y2=RMINFO(RMN,9)
11380     END IF
11390     CALL *SETBOX(X1,Y1,X2,Y2,KY)'Rectangle region setting
11400     IF KY=1 THEN
11410       VIDEOIN 0,0               'Video bus to image memory 0
11420       VDWAIT 3                  'Waiting for image input to be completed
11430       RMPUT   RMN,0,X1,Y1,X2,Y2 'ROI model registered
11440       RMMODE  RMN,3             'Density correlation
11450       RMORIGIN RMN,0,0
11460       RMGROUP 0,RMN             'Group registration
11470     ELSE
11480       EXIT FOR
11490     END IF
11500   NEXT
11510   WHILE (KEYIN(0)):WEND         'Wait for key to be released
11520  RETURN
11530 ' -----------------------------------------------------
11540 ' Setting ROI region in window memory
11550 ' -----------------------------------------------------
11560 *DRAWWIN
11570   CLS 2
11580   FOR RMN=0 TO RC-1
11590     X1=RMINFO(RMN,6)
11600     Y1=RMINFO(RMN,7)
11610     X2=RMINFO(RMN,8)
11620     Y2=RMINFO(RMN,9)
11630     BOX X1,Y1,X2,Y2,2
11640   NEXT
11650  RETURN
11660 ' -----------------------------------------------------
11670 ' Measurement condition settings
11680 ' -----------------------------------------------------
11690 *SETSQMODE
11700   SQSET 0,"M2 S0 V1"    'Group 0 search + input image on page 1
11710   Q1(0)=0               'Search model no. for positioning
11720   Q1(1)=0               'ROI group number
11730   Q1(2)=-1              'No positioning
11740   SQMODE 0,1,,Q1        '1-model positioning (no rotation)
11750 RETURN
```

7-12 Acknowledging Characters: Sequential Processing

The following program registers characters 0 to 9 in the search models 0 to 9 and acknowledges the characters.

```
10000 ' ======================================================
10010 ' Data definition
10020 ' ======================================================
10030 X1=256:Y1=240           'Initial rectangle region value
10040 X2=X1+63:Y2=Y1+63       'Rectangle region size
10050 CN=20                   'Max. No. of the same characters
10060 N=40                    'Max. No. of characters
10070 DIM SD(N),SDX(N),SDY(N) 'Array for search results
10080 ' ------------------------------------------------------
10090 ' Search group 0 not registered
10100 ' ------------------------------------------------------
10110 N=SMGINFO(0,0)          'No. of registrations for search grp 0
10120 IF N<>0 THEN            'Registration deleted if number is not 0
10130   FOR I=0 TO N-1
10140     SMGDEL SMGINFO(0,1,0)
10150   NEXT
10160 ELSE
10170   GOSUB *SETMODEL       'Model registered if number is 0
10180 END IF
10190 ' ------------------------------------------------------
10200 ' Search model 0 to 9 will be registered in search group 0.
10210 ' ------------------------------------------------------
10220 FOR SM=0 TO 9
10230   IF SMINFO(SM,0)=1 THEN SMGROUP 0,SM
10240 NEXT
10250 SMGMODE 0,0,CN,85 'Density correlation, No. of searches: CN
10260 SMSORT 1                'Search models sorted in group
10270 ' ------------------------------------------------------
10280 ' Initial settings
10290 ' ------------------------------------------------------
10300 SYSINIT 1:SYSINIT 2     'System initialization
10310 LOCATE ,,0              'Character cursor OFF
10320 DISPLAY 31              'Density image display
10330 ' ------------------------------------------------------
10340 ' Main routine
10350 ' ------------------------------------------------------
10360 SQSET 0,"M2 S0 @S13"    'Unit 0: search with group 0
10370 ON SQMEAS GOSUB *SQINT
10380 SQMEAS ON
10390 CALL *BTMMSG("ENT:Model registration  ESC:End")
10400 SQRUN 3,0,0             'Continuous sequence-type processing
10410 DO
10420   K=KEYIN(0)
10430   IF K=&H10 THEN
10440     SQRUN 0             'Stop sequence to register model
10450     GOSUB *SETMODEL
10460     CALL *BTMMSG("ENT:Model registration  ESC:End")
10470     SQRUN 3,0,0         'Restart sequence-type processing
10480   END IF
```

```
10490 LOOP WHILE (K<>&H20)
10500 SQRUN 0                 'Stop sequence-type processing
10510 LOCATE ,,1              'Character cursor ON
10520 END
10530 ' ------------------------------------------------------
10540 ' Sequence interrupt
10550 ' ------------------------------------------------------
10560 *SQINT
10570    CLS 1                        'Delete frame
10580    SMGDATA 0,2,SD
10590    N=SD(0)                      'Total number of candidate points
10600    SMGDATA 0,0,SDX,SDY,,SD      'Candidate point data
10610    S$=""
10620    FOR I=0 TO N-1
10630      S$=S$+RIGHT$(STR$(SD(I)),1)
10640      S=SD(I)                    'Associated search model number
10650      BOX
SDX(I)-SMINFO(S,3)/2,SDY(I)-SMINFO(S,4)/2,SDX(I)+SMINFO(S,3)/2,SDY(I)+SMINFO(S,4
)/2,1,,,1
10660    NEXT
10670    LOCATE 0,1:PRINT SPC(63);
10680    LOCATE 0,0:PRINT "Character recognition result";AKCNV$(S$)
10690   RETURN
10700 ' ------------------------------------------------------
10710 ' Search model registration
10720 ' ------------------------------------------------------
10730 *SETMODEL
10740   CLS 1                        'Cursor deleted
10750   FOR SMN=0 TO 9
10760     M$="Register"+AKCNV$(STR$(SMN))
10770     CALL *BTMMSG(M$)
10780     IF SMINFO(SMN,0)=1 THEN    'If model registration completed
10790       X1=SMINFO(SMN,1)         'obtain coordinates of rectangle
10800       Y1=SMINFO(SMN,2)         'region when model was registered
10810       X2=X1+SMINFO(SMN,3)-1
10820       Y2=Y1+SMINFO(SMN,4)-1
10830     END IF
10840     CALL *SETBOX(X1,Y1,X2,Y2,KY,0,71,23,67,19)
10850     IF KY=1 THEN               'Register if ENT is pressed
10860       VIDEOIN 0,0              'Video bus to image memory 0
10870       VDWAIT 3                 'Waiting for image input to be completed
10880       SX=(X2-X1)/2             'Reference position in the center of mode
10890       SY=(Y2-Y1)/2
10900       SMPUT   SMN,0,X1,Y1,X2,Y2,SX,SY 'Search model registered
10910       SMMODE  SMN,0,CN,85      'Density corr, No. of searches: CN
10920       SMAREA  SMN,0,0,511,483  'Search entire screen
10930       SMGROUP 0,SMN            'Register Model SMN in group
10940     ELSE                       'End registration if ESC pressed.
10950       EXIT FOR
10960     END IF
10970   NEXT
10980   WHILE (KEYIN(0)):WEND        'Wait for key to be released
```

```
10990   SMSORT 1                        'Search models sorted in group
11000  RETURN
```

7-13 Inspecting Defects in Linear Region: Sequential Processing

The following program inspects defects such as breaks, burrs, cracks, stains, etc. within the specified linear region and indicates their positions.

```
10000 ' ======================================================
10010 ' Data definition
10020 ' ======================================================
10030 X1=100:Y1=240           'Initial start point coordinates of line.
10040 X2=400:Y2=240
10050 MD=1                    'Measurement feature volume (small defect)
10060 DIM AR(5),R1(2),RS1(3),RS2(3),RS3(3),RS4(3)
10070 ' ======================================================
10080 ' Main program
10090 ' ======================================================
10100 ' ------------------------------------------------------
10110 ' IROI initialization
10120 ' ------------------------------------------------------
10130 SYSINIT 1:SYSINIT 2     'System initialization
10140 SYSINIT 3               'Initialize models
10150 LOCATE ,,0              'Character cursor OFF
10160 DISPLAY 31              'Density image display
10170 GOSUB *SETMODEL         'ROI model registration
10180 ' ------------------------------------------------------
10190 ' Main routine
10200 ' ------------------------------------------------------
10210 CALL *BTMMSG("ENT:Model registration  ESC:End")
10220 DO
10230   K=KEYIN(0)            'Waiting for key input
10240   IF K=&H10 THEN
10250     GOSUB *SETMODEL     'ROI model registration
10260     CALL *BTMMSG("ENT:Model registration  ESC:End")
10270   END IF
10280   GOSUB *MEAS
10290 LOOP WHILE (K<>&H20)    'Press ESC to end
10300 CLS
10310 LOCATE ,,1              'Character cursor ON
10320 END
10330 ' ------------------------------------------------------
10340 ' Inspection on line
10350 ' ------------------------------------------------------
10360 *MEAS
10370   VIDEOIN 1,0           'Input image on image memory page 1
10380   VDWAIT 3              'Waiting for image input to be completed
10390   RMRUN 0,,,RS1,RS2,RS3,RS4
10400   IF RS4(MD)=0 THEN JUDGE$="OK" ELSE JUDGE$="NG"
10410   LOCATE 0,0:PRINT USING "Results           : @"; JUDGE$
10420   LOCATE 0,1:PRINT USING "Degree of defect  : ###";RS3(MD)
10430   LOCATE 0,2:PRINT USING "Position of defect: (### , ###)";RS1(MD);RS2(MD)
```

```
10440   CLS 1                  ’
10450   CURSOR RS1(MD),RS2(MD),0,1
10460  RETURN
10470 ’ ------------------------------------------------------
10480 ’ ROI model registration
10490 ’ ------------------------------------------------------
10500 *SETMODEL
10510   CLS 1:CLS 2
10520   CALL *BTMMSG(”Set the model region”)
10530   DO
10540     CALL *SETLINE(X1,Y1,X2,Y2,KY,1)
10550   LOOP WHILE (KY<>1)
10560   CALL *BTMMSG(””)
10570   VIDEOIN –1,0:VDWAIT 3
10580   RMMODE 0,1           ’Measure the density feature volume
10590   AR(0)=X1:AR(1)=Y1    ’Start point coord of line
10600   AR(2)=X2:AR(3)=Y2    ’End point coord of line
10610   AR(4)=10:AR(5)=20  ’Model width horizontal and vertical
10620   RMPUT2 0,0,AR        ’Inspection on line with model 0
10630   RMMODE 0,1            ’Measure the density feature volume
10640   R1(0)=2              ’Defects both white and black
10650   R1(1)=MD             ’Small defect
10660   R1(2)=3              ’Mask pitch
10670   RMMODE2 0,3,R1       ’Set conditions for model 0
10680   RMJUDGE 0,MD,0,15    ’OK if degree of defect is 0 to 15
10690   CLS 2
10700   LINE X1,Y1,X2,Y2,2   ’Display region.
10710   WHILE (KEYIN(0)):WEND ’Wait for key to be released
10720  RETURN
```

7-14 Menu Library

The following program determines the image filtering selection and date setting using the menu library.

```
10000 ’ ======================================================
10010 ’ Data definition
10020 ’ ======================================================
10030 M0$=”F.Filtering/D.Date/E.End”
10040 MN0=0
10050 T1$=”Filtering”
10060 M1$=”0.OFF/W.Weak smoothing/S.Strong smoothing/1.Edge enhancement 1/2.Edge
enhancement 2/3.Edge enhancement 3/4.Edge enhancement 4/5.Edge enhancement
5/R.Relief/V.Vertical edges/H.Horizontal edges/X.All edges”
10070 MN1=0
10080 T2$=”Calendar”
10090 M2$=”Y.Year :/M.Month:/D.Date :”
10100 I2$=”N&&0&1&4&1980&2079/N&&0&1&2&1&12/N&&0&1&2&1&31”
10110 ’ ======================================================
10120 ’ Main program
10130 ’ ======================================================
10140 ’------------------------------------------------------
10150 ’ Initial menu setting
10160 ’------------------------------------------------------
```

```
10170 SYSINIT 1:SYSINIT 2     'System initialization
10180 LOCATE ,,0              'Character cursor OFF
10190 DISPLAY 31,0            'Display filtered image
10200 SETDLVL 1,0             'Graphic gradation 0
10210 CALL *MENUINIT(0)       'Initialize menu
10220 MN0=0                   'Initial values on menu bar
10230 MN1=0                   'Initial values on filter menu
10240 '-----------------------------------------------------
10250 ' Main menu
10260 '-----------------------------------------------------
10270 WHILE (1)
10280   CALL *MBARDISP(M0$,MN0)
10290   CALL *MBARSELECT(M0$,MN0)
10300   SELECT MN0
10310     CASE 0:GOSUB *SETFILT
10320     CASE 1:GOSUB *SETDATE
10330     CASE 2:LOCATE ,,1:END
10340   END SELECT
10350 WEND
10360 END
10370 '-----------------------------------------------------
10380 ' Filtering setting
10390 '-----------------------------------------------------
10400 *SETFILT
10410   CALL *MENUDISP(T1$,M1$,MN1)   'Display menu
10420   DO
10430     CALL *MENUSELECT(T1$,M1$,MN1,KY1)
10440     FILTSEL MN1
10450   LOOP WHILE (KY1<>0 AND KY1<>1)
10460   CLS
10470  RETURN
10480 '-----------------------------------------------------
10490 ' Date setting
10500 '-----------------------------------------------------
10510 *SETDATE
10520   DT$=DATE$
10530   IF VAL(PIECE$(DT$,"/",1))>79 THEN
10540     DT$="19"+DT$
10550   ELSE
10560     DT$="20"+DT$
10570   END IF
10580   CALL *DBOXDS(T2$,M2$,I2$,DT$,KY2)
10590   IF KY2=1 THEN
10600     D$=RIGHT$(PIECE$(DT$,"/",1),2)
10610     D$=D$+"/"+RIGHT$("0"+PIECE$(DT$,"/",2),2)
10620     D$=D$+"/"+RIGHT$("0"+PIECE$(DT$,"/",3),2)
10630     DATE$=D$
10640   END IF
10650  RETURN
```

7-15 Changing Search Area Using the Search Model

The following program sets the search area of the search model 0.

```
10000 ' ------------------------------------------------------
10010 ' Initial settings
10020 ' ------------------------------------------------------
10030 SYSINIT 1:SYSINIT 2     'System initialization
10040 LOCATE ,,0              'Character cursor OFF
10050 DISPLAY 31              'Density image display
10060 ' ------------------------------------------------------
10070 ' Obtain present search region of search model 0
10080 ' ------------------------------------------------------
10090 IF SMINFO(0,0) THEN     'If model is registered
10100   X1=SMINFO(0,8)        ' Upper left X coordinate of region
10110   Y1=SMINFO(0,9)        ' Upper left Y coordinate of region
10120   X2=SMINFO(0,10)       ' Lower right X coordinate of region
10130   Y2=SMINFO(0,11)       ' Lower right Y coordinate of region
10140 ELSE                    'If model not registered, entire screen
10150   X1=0  :Y1=0
10160   X2=511:Y2=483
10170 END IF
10180 ' ------------------------------------------------------
10190 ' Set rectangle as initial value for present region
10200 ' ------------------------------------------------------
10210 CALL *SETBOX(X1,Y1,X2,Y2,KY) 'Set region
10220 IF KY=1 THEN            'Register region if ENT is pressed
10230   SMAREA 0,X1,Y1,X2,Y2
10240   PRINT "Search region for search model 0 was changed"
10250 END IF
10260 LOCATE ,,1              'Character cursor ON
10270 END
```

7-16 Indication of Search Model Setting Conditions

The following program indicates the setting conditions of the search model.

```
10000 ' ------------------------------------------------------
10010 ' Data setting
10020 ' ------------------------------------------------------
10030 DIM DNAME$(100)
10040 I=0
10050 FOR I=0  TO 17:READ DNAME$(I):NEXT
10060 FOR I=25 TO 28:READ DNAME$(I):NEXT
10070 READ DNAME$(100)
10080 ' ------------------------------------------------------
10090 ' Display data for search model 0
10100 ' ------------------------------------------------------
10110 IF SMINFO(0,0)=0 THEN
10120   PRINT USING "@ : @";DNAME$(0);"Not registered"
10130 ELSE
10140   PRINT USING "@ : @";DNAME$(0);"Registered"
10150 END IF
10160 FOR I=1 TO 17
10170   PRINT USING "@ : ####.## ";DNAME$(I);SMINFO(0,I)
```

```
10180 NEXT
10190 FOR I=25 TO 28
10200   PRINT USING "@ : ####.## ";DNAME$(I);SMINFO(0,I)
10210 NEXT
10220 SELECT SMINFO(0,100)
10230   CASE –2: PRINT USING "@ : @";DNAME$(100);"Initialize"
10240   CASE –1: PRINT USING "@ : @";DNAME$(100);"No error"  '
10250   CASE ELSE:PRINT USING "@ : ###";DNAME$(100);SMINFO(0,100)
10260 END SELECT
10270 END
10280 '
10290 ' ------------------------------------------------------
10300 ' Data
10310 ' ------------------------------------------------------
10320 DATA "Model registered Y/N"
10330 DATA "X coordinate when model was registered"
10340 DATA "Y coordinate when model was registered"
10350 DATA "Model size in the X direction"
10360 DATA "Model size in the Y direction"
10370 DATA "X coordinate of the model reference point"
10380 DATA "Y coordinate of the model reference point"
10390 DATA "Model reference angle"
10400 DATA "X coordinate of the upper left of search region"
10410 DATA "Y coordinate of the upper left of search region"
10420 DATA "X coordinate of the lower right of search region"
10430 DATA "Y coordinate of the lower right of search region"
10440 DATA "Type of evaluation feature volume"
10450 DATA "No. of searches"
10460 DATA "Evaluation level"
10470 DATA "Noise level"
10480 DATA "Absolute value mode"
10490 DATA "Search group no."
10500 DATA "Maximum point of stored start nos."
10510 DATA "The total number of registered search models"
10520 DATA "The number of vacant search models"
10530 DATA "The total no. of search models that can be registered"
10540 DATA "Error data"
```

7-17 Saving the Search Model

The following program saves the search model specified by its number in the file format into the memory card.

```
10000 ' ======================================================
10010 ' Data definition
10020 ' ======================================================
10030 T$="Saving Search Model"
10040 M$="S.Start model no.:/E.End model no.  :/F.File name      :"
10050 I$="N&&0&1&3&0&435/N&&0&1&3&0&435/C&&8&A&Z"
10060 D$="0/0/        "
10070 ' ======================================================
10080 ' Main program
10090 ' ======================================================
10100 ' ------------------------------------------------------
```

```
10110 ' IROI initialization
10120 ' ------------------------------------------------------
10130 CALL *MENUINIT(0)               'Initialize menu
10140 ' ------------------------------------------------------
10150 ' Main routine
10160 ' ------------------------------------------------------
10170 CALL *DBOXDS(T$,M$,I$,D$,KY)    'Display dialog box
10180 IF KY=1 THEN                    'If ENT is pressed
10190   F$=PIECE$(D$,"/",3,3)         'File name
10200   SMS=VAL(PIECE$(D$,"/",1,1))   'Start model number
10210   SME=VAL(PIECE$(D$,"/",2,2))   'End model number
10220   SMSAVE F$,SMS,SME             'Save search model
10230 END IF
10240 END
```

7-18 Loading the Search Model

The following program loads the search model from the specified file name.

```
10000 ' ------------------------------------------------------
10010 ' Initial settings
10020 ' ------------------------------------------------------
10030 SYSINIT 1:SYSINIT 2:    'System initialization
10040 ' ------------------------------------------------------
10050 ' Memory file directory display and selection
10060 ' ------------------------------------------------------
10070 F$="*.*"
10080 CALL *SELECTFILE(-1,-1,-1,F$,2)
10090 ' ------------------------------------------------------
10100 ' Selected files loaded as search models.
10110 ' ------------------------------------------------------
10120 IF F$<>"" THEN            'If ESC is not pressed
10130   LOCATE 0
10140   PRINT "Search model file =";F$
10150   SMLOAD F$             'Load search model
10160 END IF
```

7-19 Selecting the Search Model

The following program finds an optimum region for the search model out of a broad area.

```
10000 ' ======================================================
10010 ' Data definition
10020 ' ======================================================
10030 N=10                    'Maximum number of candidates
10040 DIM AX1(N),AY1(N),AX2(N),AY2(N)
10050 ' ======================================================
10060 ' Main program
10070 ' ======================================================
10080 ' ------------------------------------------------------
10090 ' Initial settings
10100 ' ------------------------------------------------------
10110 SYSINIT 1:SYSINIT 2     'System initialization
```

```
10120 LOCATE ,,0              'Character cursor OFF
10130 DISPLAY 31,0            'Density image display
10140 CALL *BTMMSG("ENT: Input image")
10150 K=KYEIN(1)
10160 VIDEOIN                 'ENT: Input image
10170 VDWAIT 3                'Waiting for image input to be completed
10180 ' ------------------------------------------------------
10190 ' Candidate points extracted from entire screen.
10200 ' ------------------------------------------------------
10210 SMSELECT 0,0,0,511,483,N,AX1,AY1,AX2,AY2
10220 LSTYLE 1                'Display dotted line
10230 FOR I=0 TO N-1
10240   BOX AX1(I),AY1(I),AX2(I),AY2(I),1,,,1
10250 NEXT
10260 LSTYLE 0                'Display solid line
10270 ' ------------------------------------------------------
10280 ' Select the frame
10290 ' ------------------------------------------------------
10300 I=0
10310 CALL *BTMMSG("ESC:Quit")
10320 DO
10330   BOX AX1(I),AY1(I),AX2(I),AY2(I),0,,OR,1
10340   K=KEYIN(1)
10350   BOX AX1(I),AY1(I),AX2(I),AY2(I),0,,NOT,1
10360   SELECT K
10370   CASE 1                'UP key
10380     I=I+1:IF I>=N THEN I=0
10390   CASE 8                'DOWN key
10400     I=I-1:IF I<0  THEN I=N-1
10410   END SELECT
10420 LOOP WHILE(K<>&H20)     'Press ESC to end
10430 LOCATE ,,0              'Character cursor ON
10440 END
```

7-20 Setting the ROI Model Data

The following program registers the present date and the type of image filtering before the ROI model is registered.

```
10000 ' ======================================================
10010 ' Data definition
10020 ' ======================================================
10030 X1=256  :Y1=256         'Initial rectangle region value
10040 X2=X1+63:Y2=Y1+63       'Rectangle region size
10050 ' ======================================================
10060 ' Main program
10070 ' ======================================================
10080 ' ------------------------------------------------------
10090 ' Initial settings
10100 ' ------------------------------------------------------
10110 SYSINIT 1:SYSINIT 2     'System initialization
10120 DISPLAY 31              'Density image display
10130 ' ------------------------------------------------------
10140 ' Main routine
```

```
10150 ' ------------------------------------------------------
10160 VIDEOIN                 'ENT: Input image
10170 VDWAIT 3                'Waiting for image input to be completed
10180 CALL *SETBOX(X1,Y1,X2,Y2,KY)
10190 IF KY=1 THEN            'If ENT is pressed
10200   RMPUT 0,0,X1,Y1,X2,Y2 ' ROI model registration

10210   RMEXTSET 0,0,DATE$    'Set date
10220   RMEXTSET 0,9,TIME$    'Set time
10230   PRINT "Following data registered in ROI model 0."
10240   PRINT "Date of registration:";RMEXTGET$(0,0)
10250   PRINT "Time of registration:";RMEXTGET$(0,9)
10260 END IF
```

7-21 Setting the ROI Model Conditions

The following program sets the ROI model conditions.

```
10010 '-------------------------------------------------------
10020 ' Model no. setting
10030 '-------------------------------------------------------
10040 CLS:LOCATE 0,0
10050 INPUT "Designate the model no. to set data. ";RMNO
10060 '-------------------------------------------------------
10070 ' Set measurement feature volume
10080 '-------------------------------------------------------
10090 CLS:LOCATE 0,0
10110 PRINT
10120 PRINT " 0 : Binary feature volume"
10130 PRINT " 1 : Density feature volume"
10140 PRINT " 2 : Binary weighted correlation"
10150 PRINT " 3 : Density correlation"
10160 PRINT " 4 : Binary feature volume criteria"
10170 PRINT " 5 : Density feature volume criteria"
10180 PRINT " 6 : Binary weighted correlation criteria"
10190 PRINT " 7 : Density correlation criteria"
10200 INPUT "Specify the measurement feature volume";MSMODE
10220 '-------------------------------------------------------
10230 ' Setting density for clipping
10240 '-------------------------------------------------------
10250 CLS:LOCATE 0,0
10260 INPUT "Input density lower limit (0 to 255)";GRYLOW
10270 PRINT "Input density upper limit (";GRYLOW;" to 255)";:INPUT GRYHIGH
10280 '-------------------------------------------------------
10290 ' Set absolute value mode
10300 '-------------------------------------------------------
10310 CLS:LOCATE 0,0
10320 PRINT " ** Absolute value mode setting **"
10330 PRINT
10340 PRINT "     0: No inversion"
10350 PRINT " Not 0: Inversion"
10360 INPUT "Set the absolute value mode";AMODE
10370 '-------------------------------------------------------
10380 ' Set ROI conditions
```

```
10390 ’-----------------------------------------------------
10400 RMMODE RMNO,MSMODE,GRYLOW,GRYHIGH,AMODE
10410 DIM MODE$(7)
10420 FOR I=0 TO 7:READ MODE$(I):NEXT
10430 CLS:LOCATE 0,0
10440 PRINT ”The setting is as follows:”
10450 PRINT ” Model no. :”;RMNO
10460 PRINT USING ” MFV :@”;MODE$(RMINFO(RMNO,11))
10470 PRINT ” Density lower limit :”;RMINFO(RMNO,12)
10480 PRINT ” Density upper limit :”;RMINFO(RMNO,13)
10490 IF RMINFO(RMNO,14)=0 THEN
10500 PRINT ” Absolute value mode: No inversion”
10510 ELSE
10520 PRINT ” Absolute value mode: Inversion”
10530 END IF
10540 END
10550 DATA ”Binary feature volume”
10560 DATA ”Density feature volume”
10570 DATA ”Binary weighted correlation”
10580 DATA ”Density correlation”
10590 DATA ”Binary feature volume criteria”
10600 DATA ”Density feature volume criteria”
10610 DATA ”Model no. setting 0”
10620 DATA ”Density correlation”
```

7-22 Setting the ROI Model Judgment Criteria

The following program sets the judgment criteria in the ROI model.

```
10000 ’ ======================================================
10010 ’ Data definition
10020 ’ ======================================================
10030 DIM R1(1)              ’Array for ROI results
10040 RJ=70                  ’Evaluation level of ROI density correlation
10050 ’ ======================================================
10060 ’ Main program
10070 ’ ======================================================
10080 ’ ------------------------------------------------------
10090 ’ Initial settings
10100 ’ ------------------------------------------------------
10110 SYSINIT 1:SYSINIT 2    ’System initialization
10120 SYSINIT 3              ’Initialize models
10130 LOCATE ,,0             ’Character cursor OFF
10140 DISPLAY 31             ’Density image display
10150 ’ ------------------------------------------------------
10160 ’ ROI model registration
10170 ’ ------------------------------------------------------
10180 CALL *BTMMSG(”Designate the ROI region”)
10190 VIDEOIN:VDWAIT 3       ’ENT: Input image
10200 CALL *SETBOX(X1,Y1,X2,Y2,KY)
10210 RMPUT 0,0,X1,Y1,X2,Y2  ’ROI model registration
10220 CALL *BTMMSG(””)
10230 RMMODE 0,7             ’Set the measurement mode (density correlation
eval)
```

```
10240 RMJUDGE 0,0,RJ,100      'Set criteria (Corr val RJ to 100)
10250 BOX RMINFO(0,6),RMINFO(0,7),RMINFO(0,8),RMINFO(0,9),1,,,1
10260 ' ------------------------------------------------------
10270 ' Main routine
10280 ' ------------------------------------------------------
10290 GOSUB *SETJUDGE         'Display criteria
10300 CALL *BTMMSG("ESC:Quit ")
10310 DO
10320   GOSUB *MEAS
10330   K=KEYIN(0)            'Waiting for key input
10340   IF K=&H1  THEN RJ=RJ+1 GOSUB *SETJUDGE
10350   IF K=&H8  THEN RJ=RJ-1 GOSUB *SETJUDGE
10360 LOOP WHILE (K<>&H20)    'Press ESC to end
10370 LOCATE ,,1              'Character cursor ON
10380 END
10390 '------------------------------------------------------
10400 ' ROI processing and evaluation results display
10410 '------------------------------------------------------
10420 *MEAS
10430   VIDEOIN 1,0:VDWAIT 3 'Input image
10440   RMRUN 0,,,R1         'Execute ROI
10450   LOCATE 0,1
10460   PRINT USING "Corr  : ###";R1(0) 'Display den corr val
10470   IF R1(1)=0 THEN                'Display density correlation value
10480     PRINT "Result: OK"
10490   ELSE
10500     PRINT "Result: NG"
10510   END IF
10520  RETURN
10530 ' ------------------------------------------------------
10540 ' Set criteria
10550 ' ------------------------------------------------------
10560 *SETJUDGE
10570    IF RJ>100 THEN RJ=100
10580    IF RJ<0   THEN RJ=0
10590    LOCATE 0,0
10600    PRINT USING "Evaluation value: ###";RJ
10610  RETURN
```

7-23 Registering the ROI Model for Inspecting Defects in Circumferential Areas

The following program registers the model for inspecting defective rate in the circumferential areas.

```
10000 ' ======================================================
10010 ' Data definition
10020 ' ======================================================
10030 DIM AR(4),R1(2)
10040 X1=256:Y1=240           'Coordinates of the center of circle
10050 R1=100                  'The radius of circle (outside)
10060 R2=R1-10                'The radius of circle (inside)
10070 ' ======================================================
```

```
10080 ’ Main program
10090 ’ ========================================================
10100 ’ --------------------------------------------------------
10110 ’ IROI initialization
10120 ’ --------------------------------------------------------
10130 SYSINIT 1:SYSINIT 2     ’System initialization
10140 LOCATE ,,0              ’Character cursor OFF
10150 DISPLAY 31              ’Density image display
10160 ’ --------------------------------------------------------
10170 ’ Circle settings
10180 ’ --------------------------------------------------------
10190 CALL *BTMMSG(”Set the model region”)
10200 DO
10210   CALL *SETCCIRCLE(X1,Y1,R1,R2,KY,1,511,0,5)
10220 LOOP WHILE (KY<>1)
10230 ’ --------------------------------------------------------
10240 ’ Model registration
10250 ’ --------------------------------------------------------
10260 VIDEOIN 0,0             ’ENT: Input image
10270 VDWAIT 3                ’Waiting for image input to be completed
10280 AR(0)=X1                ’Coordinates of the center of circle
10290 AR(1)=Y1                ’Coordinates of the center of circle
10300 AR(2)=(R1+R2)/2         ’The radius of circle
10310 AR(3)=10                ’The width of model (direction of circle)
10320 AR(4)=R1–R2             ’The width of model (perpendicular to circle)
10330 RMPUT2 0,2,AR           ’Model registration
10340 RMMODE 0,1              ’Density feature volume
10350 R1(0)=2                 ’Black and white
10360 R1(1)=1                 ’Inspect for small defect
10370 R1(2)=4                 ’Mask width
10380 RMMODE2 0,5,R1          ’Inspect on circumference
10390 RMJUDGE 0,0,0,70
10400 CALL *BTMMSG(”ROI model registered for circle inspect.”)
10410 END
```

7-24 Registering the ROI Model for Inspecting Defects in Optional Areas

The following program registers the model for inspecting defective rate in the optional areas.

```
10000 ’ ========================================================
10010 ’ Data definition
10020 ’ ========================================================
10030 DIM AR(7),R1(2)
10040 ’ ========================================================
10050 ’ Main program
10060 ’ ========================================================
10070 ’ --------------------------------------------------------
10080 ’ IROI initialization
10090 ’ --------------------------------------------------------
10100 SYSINIT 1:SYSINIT 2     ’System initialization
10110 LOCATE ,,0              ’Character cursor OFF
```

```
10120 DISPLAY 31              'Density image display
10130 ' ------------------------------------------------------
10140 ' Main routine
10150 ' ------------------------------------------------------
10160 ' ------------------------------------------------------
10170 ' ROI model registration
10180 ' ------------------------------------------------------
10190 MN=0
10200 M$="OR/NOT/XOR/END"
10210 DO
10220   CALL *MBOXDS(2,1,"",M$,MN,KY) 'Select menu display
10230   CALL *MBOXCLR(2,1,"",M$)
10240   IF MN<>3 THEN GOSUB *MKBOX    'Set region
10250   SELECT MN
10260     CASE 0:BOX X1,Y1,X2,Y2,2,,OR
10270     CASE 1:BOX X1,Y1,X2,Y2,2,,NOT
10280     CASE 2:BOX X1,Y1,X2,Y2,2,,XOR
10290   END SELECT
10300 LOOP WHILE (MN<>3)
10310 CALL *BTMMSG("Model registration")
10320 VIDEOIN 0,0
10330 VDWAIT 3
10340 ' ------------------------------------------------------
10350 ' Determine coordinates of external rectangle of the region
10360 ' ------------------------------------------------------
10370 SLABEL 2,0,7,1
10380 X1=511:Y1=511
10390 X2=0  :Y2=0
10400 FOR I=1 TO LNUM
10410   IF LDATA(I,7)<X1  THEN X1=LDATA(I,7)
10420   IF LDATA(I,8)<Y1  THEN Y1=LDATA(I,8)
10430   IF LDATA(I,9)>X2  THEN X2=LDATA(I,9)
10440   IF LDATA(I,10)>Y2 THEN Y2=LDATA(I,10)
10450 NEXT
10460 BOX X1,Y1,X2,Y2,1,,,1
10470 ' ------------------------------------------------------
10480 ' Model registration
10490 ' ------------------------------------------------------
10500 AR(0)=X1:AR(1)=Y1       'Coordinates of start point
10510 AR(2)=X2:AR(3)=Y2       'Coordinates of end point
10520 AR(4)=10                'Horizontal width of model
10530 AR(5)=10                'Vertical width of model
10540 AR(6)=4                 'Mask pitch
10550 RMPUT2 0,3,AR,7
10560 RMMODE 0,1              'Density feature volume
10570 R1(0)=2                 'Black and white
10580 R1(1)=0                 'Large defect
10590 RMMODE2 0,6,R1
10600 CALL *BTMMSG("Model registration completed")
10610 LOCATE ,,1              'Character cursor ON
10620 END
10630 ' ------------------------------------------------------
```

```
10640 ’ Set rectangle
10650 ’ ------------------------------------------------------
10660 *MKBOX
10670   CALL *BTMMSG(”Set the model region”)
10680   DO
10690     CALL *SETBOX(X1,Y1,X2,Y2,KY)
10700   LOOP WHILE (KY<>1)
10710   CALL *BTMMSG(””)
10720  RETURN
```

7-25 Repeatedly Executing Image Filtering: Sequential Processing

The following program sets the sequence conditions for searching against the image smoothed after extracting edges.

```
10000 ’ ======================================================
10010 ’ Data definition
10020 ’ ======================================================
10030 X1=256:Y1=240           ’Initial model region
10040 X2=X1+63:Y2=Y1+63       ’Initial model size
10050 ’ ------------------------------------------------------
10060 ’ Initial settings
10070 ’ ------------------------------------------------------
10080 SYSINIT 1:SYSINIT 2     ’System initialization
10090 LOCATE ,,0              ’Character cursor OFF
10100 DISPLAY 31,0            ’Density image display
10110 GOSUB *SETMODEL         ’Model registration
10120 SMGROUP 0,0 ’Register model 0 in group 0
10130 ’ ------------------------------------------------------
10140 ’ Initialize sequence conditions
10150 ’ ------------------------------------------------------
10160 SQINIT –1
10170 ’ ------------------------------------------------------
10180 ’ Unit 0: Input image after edge extraction, no measurement
10190 ’ ------------------------------------------------------
10200 SQSET 0,”M0 V0,3”,”B0 F11 D31,0”
10210 ’ ------------------------------------------------------
10220 ’ Unit 1: Smoothing image in image memory and group 0 search
10230 ’ ------------------------------------------------------
10240 SQSET 1,”M2 S0”,”B1,0 F2 D31,0”
10250 ’ ------------------------------------------------------
10260 ’ Single measurement for units 0,1
10270 ’ ------------------------------------------------------
10280 SQRUN 3,0,1             ’Continuous sequence–type processing
10290 CALL *BTMMSG(”ESC:Stop”)
10300 DO
10310   K=KEYIN(0)
10320   CLS 1
10330   CURSOR SMDATA(0,8),SMDATA(0,10),0,1
10340 LOOP WHILE (K<>&H20)    ’Press ESC to end
10350 SQRUN 0                 ’Stop sequence–type processing
10360 CLS
```

```
10370 LOCATE ,,1              'Character cursor ON
10380 END
10390 ' ------------------------------------------------------
10400 ' Search model registration
10410 ' ------------------------------------------------------
10420 *SETMODEL
10430   FILTSEL 11                      'Edge extraction
10440   VIDEOIN 1,1                     'Input image on page 1
10450   VDWAIT 3
10460   FILTSEL 2                       'Smoothing
10470   FILTERIN 1,1                    'Display page 1
10480   VIDEOIN 0,1                     'Input image on page 0
10490   VDWAIT 3
10500   CALL *BTMMSG("Set the model region")
10510   DO
10520     CALL *SETBOX(X1,Y1,X2,Y2,KY,0,71,23,67,19)
10530   LOOP WHILE (KY<>1)              'Waiting until ENT is pressed
10540   CALL *BTMMSG("")                'Delete message
10550   SX=(X2-X1)/2                    'Reference position in the center of
model
10560   SY=(Y2-Y1)/2
10570   SMPUT   0,0,X1,Y1,X2,Y2,SX,SY   'Search model registered
10580   SMMODE  0,0,0                   'Density correlation, min. and max. only
10590   SMAREA  0,0,0,511,483           'Search entire screen
10600   SMGROUP 0,0
10610  RETURN
```

7-26 Obtaining Sequential Processing Time

The following program provides the sequential processing time.

```
10000 ' ======================================================
10010 ' Data definition
10020 ' ======================================================
10030 DIM X(12),Y(12)         'Array for Search position
10040 X1=256:Y1=240           'Initial rectangle region value
10050 X2=X1+63:Y2=Y1+63       'Rectangle region size
10060 ' ======================================================
10070 ' Main program
10080 ' ======================================================
10090 ' ------------------------------------------------------
10100 ' Initial settings
10110 ' ------------------------------------------------------
10120 SYSINIT 1:SYSINIT 2     'System initialization
10130 LOCATE ,,0              'Character cursor OFF
10140 DISPLAY 31              'Density image display
10150 ' ------------------------------------------------------
10160 ' Search group 0 not registered
10170 ' ------------------------------------------------------
10180 N=SMGINFO(0,0)          'No. of registrations
10190 IF N<>0 THEN            'Registration deleted if number is not 0
10200   FOR I=0 TO N-1
10210     SMGDEL SMGINFO(0,1,0)
10220   NEXT
```

```
10230 ELSE
10240   GOSUB *SETMODEL       'Model registered if number is 0
10250 END IF
10260 ' -----------------------------------------------------
10270 ' Search model 0 to 11 registered in search group 0
10280 ' -----------------------------------------------------
10290 FOR SM=0 TO 11          'Registered models registered in group
10300   IF SMINFO(SM,0)=1 THEN SMGROUP 0,SM
10310 NEXT
10320 SMGMODE 0,0,1           'Density correlation, 1 search
10330 SMSORT 1                'Search models sorted in group
10340 ' -----------------------------------------------------
10350 ' Main routine
10360 ' -----------------------------------------------------
10370 SQINIT -1               'Initialize sequence processing
10380 ON SQMEAS GOSUB *SQINT  'Define sequence interrupt
10390 SQMEAS ON               'Enable sequence interrupt
10400 SQSET 0,"M2 S0"         'Search group 0
10410 SQRUN 3,0,0             'Repeatedly measure unit 0
10420 CALL *BTMMSG("ENT:Model registration  ESC:End")
10430 DO
10440   K=KEYIN(0)
10450   IF K=&H10 THEN        'Register model if ENT is pressed
10460     SQRUN 0             'Stop sequence to register model
10470     GOSUB *SETMODEL
10480     CALL *BTMMSG("ENT:Model registration  ESC: End")
10490     SQRUN 3,0,0         'Restart sequence-type processing
10500   END IF
10510 LOOP WHILE (K<>&H20)    'Press ESC to end
10520 SQRUN 0                 'Stop sequence-type processing
10530 LOCATE ,,1              'Character cursor ON
10540 END
10550 ' -----------------------------------------------------
10560 ' Sequence interrupt
10570 ' -----------------------------------------------------
10580 *SQINT
10590   CLS 1                 'Graphic memory cleared
10600   SMGDATA 0,0,X,Y       'Candidate point data in group
10610   FOR SM=0 TO 11
10620     IF SMINFO(SM,0)=1 THEN
10630       CURSOR X(SM),Y(SM),0,1    'Cursor displayed
10640     END IF
10650   NEXT
10660   LOCATE 0,0:PRINT USING "Total  Time=####ms";SQTIME(0,1)
10670   LOCATE 0,1:PRINT USING "Raster Time=####ms";SQTIME(0,3)
10680  RETURN
10690 ' -----------------------------------------------------
10700 ' Search model registration
10710 ' -----------------------------------------------------
10720 *SETMODEL
10730   CLS 1                       'Cursor deleted
10740   FOR SMN=0 TO 11
```

```
10750     M$="Set the model region of model "+STR$(SMN)
10760     M$=M$+"  ESC: End registration"
10770     CALL *BTMMSG(M$)
10780     IF SMINFO(SMN,0)=1 THEN     'If model registration completed
10790       X1=SMINFO(SMN,1)          'obtain coordinates of rectangle
10800       Y1=SMINFO(SMN,2)          'region when model was registered
10810       X2=X1+SMINFO(SMN,3)-1
10820       Y2=Y1+SMINFO(SMN,4)-1
10830     END IF
10840     CALL *SETBOX(X1,Y1,X2,Y2,KY,0,71,23,67,19)
10850     IF KY=1 THEN                'Register if ENT is pressed
10860       VIDEOIN 0,0               'Video bus to image memory 0
10870       VDWAIT 3                  'Waiting for image input to be completed
10880       SX=(X2-X1)/2              'Reference position in the center of model
10890       SY=(Y2-Y1)/2
10900       SMPUT   SMN,0,X1,Y1,X2,Y2,SX,SY 'Search model registered
10910       SMMODE  SMN,0,1           'Density correlation, min. and max. only
10920       SMAREA  SMN,0,0,511,483   'Search entire screen
10930       SMGROUP 0,SMN             'Register Model SMN in group
10940     ELSE                        'End registration if ESC pressed.
10950       EXIT FOR
10960     END IF
10970   NEXT
10980   WHILE (KEYIN(0)):WEND         'Wait for key to be released
10990   SMSORT 1                      'Search models sorted in group
11000  RETURN
```

7-27 Cutting Off Characters

The following program cuts off characters using the labelling function.

```
10000 ' ======================================================
10010 ' Main program
10020 ' ======================================================
10030 ' ------------------------------------------------------
10040 ' IROI initialization
10050 ' ------------------------------------------------------
10060 SYSINIT 1:SYSINIT 2     'System initialization
10070 LOCATE ,,0              'Character cursor OFF
10080 DISPLAY 31,0            'Display binary image
10090 FILTER 0,1              'LUT ON
10100 LVL=128:GOSUB *SETLVL   'Set initial value for binary level
10110 LOCATE 0,21:PRINT "LEFT/RIGHT: Bin level; SHIFT+ENT: Inverse"
10120 LOCATE 0,22:PRINT "ENT: Execute labeling"
10130 ' ------------------------------------------------------
10140 ' Main routine
10150 ' ------------------------------------------------------
10160 DO
10170   K=KEYIN(1)
10180   SELECT K
10190   CASE &H1,&H4          'UP: Increase binary level
10200     LVL=LVL+1:GOSUB *SETLVL
10210   CASE &H2,&H8          'DOWN: Decrease binary level
10220     LVL=LVL-1:GOSUB *SETLVL
```

```
10230   CASE &H90             'SHIFT+ENT: Invert binary level
10240     LEVEL -1,0,511,XOR
10250   END SELECT
10260 LOOP WHILE (K<>&H10)      'End if ENT is pressed
10270 VIDEOIN 0,1             'Input binary image via pipeline bus
10280 VDWAIT 3                'Waiting for image input to be completed
10290 SLABEL 3,0,7,1          'Label white pixels
10300 IF LNUM<0 THEN
10310   PRINT " Too many labels":END
10320 END IF
10330 LSORT 8                 'Sort horizontally from upper left of screen
10340 FOR I=1 TO LNUM         'All labels
10350   X1=LDATA(I,7)
10360   Y1=LDATA(I,8)
10370   X2=LDATA(I,9)
10380   Y2=LDATA(I,10)
10390   BOX X1,Y1,X2,Y2,1,,,1 'Display external rectangle
10400 NEXT
10410 LOCATE ,,1              'Character cursor ON
10420 END
10430 ' ------------------------------------------------------
10440 ' Binary level change routine
10450 ' ------------------------------------------------------
10460 *SETLVL
10470    IF LVL<0   THEN LVL=0
10480    IF LVL>255 THEN LVL=255
10490    LEVEL -1,LVL,255
10500  RETURN
```

7-28 Parallel Port Output of Measured Result

The following program outputs the search result for search model 0 via the parallel port when the handshake is set to ON.

```
10000 ' ======================================================
10010 ' Data definition
10020 ' ======================================================
10030 DIM SD(3)               'Array for search results
10040 ' ======================================================
10050 ' Main program
10060 ' ======================================================
10070 ' ------------------------------------------------------
10080 ' IROI initialization
10090 ' ------------------------------------------------------
10100 SYSINIT 1:SYSINIT 2     'System initialization
10110 LOCATE ,,0              'Character cursor OFF
10120 DISPLAY 31
10130 CALL *BTMMSG("ENT: Measurement + parallel output")
10140 ON ERROR GOTO *ERRINT
10150 ' ------------------------------------------------------
10160 ' Main routine
10170 ' ------------------------------------------------------
10180 WHILE (1)
10190   IF KEYIN(1)=&H10 THEN 'Measure and output data for ENT
```

```
10200     SMRUN 0,0,0          'Search with search model 0
10210     SD(0)=SMDATA(0,8)    'X coordinates searched
10220     SD(1)=SMDATA(0,10)   'Y coordinates searched
10230     SD(2)=SMDATA(0,12)   'Correlation
10240     DBOUT 3,SD,16,0      'Output data to Parallel I/O unit
10250   END IF                 'without handshake
10260 WEND
10270 ' ------------------------------------------------------
10280 ' Routine  when an error occurs
10290 ' ------------------------------------------------------
10300 *ERRINT
10310  IF ERR=33 THEN
10320    PRINT "Parallel I/O Unit is not connected"
10330  END IF
10340  END
```

7-29 Scrolling the Image Memory

The following program allows the scrolling, enlarging, or reducing of the image memory using the Console key.

```
10000 ' ======================================================
10010 ' Data definition
10020 ' ======================================================
10030 X=0:Y=0:Z=1:R=0 'Initial values of scroll amount and magnification
10040 ' ======================================================
10050 ' Main program
10060 ' ======================================================
10070 ' ------------------------------------------------------
10080 ' IROI initialization
10090 ' ------------------------------------------------------
10100 SYSINIT 1:SYSINIT 2     'System initialization
10110 LOCATE ,,0              'Character cursor OFF
10120 DISPLAY 31              'Density image display
10130 VIDEOIN 0               'ENT: Input image
10140 VDWAIT 3                'Waiting for image input to be completed
10150 FILTERIN 1              'Display image from image memory
10160 LOCATE 0,20:PRINT "ENT:Input image"
10170 LOCATE 0,21:PRINT"Cursor:Scroll  SHIFT+UP:Rotate  SHIFT+DOWN:Rotate";
10180 LOCATE 0,22:PRINT"SHIFT+ENT:Enlarge  SHIFT+ESC:Reduce";
10190 ' ------------------------------------------------------
10200 ' Main routine
10210 ' ------------------------------------------------------
10220 WHILE (1)
10230   SELECT KEYIN(1)
10240   CASE &H10:VIDEOIN:VDWAIT 3
10250   CASE &H1:Y=Y-5        'UP
10260   CASE &H2:X=X-5        'LEFT
10270   CASE &H4:X=X+5        'RIGHT
10280   CASE &H8:Y=Y+5        'DOWN
10290   CASE &H81:R=R-5       'SHIFT+UP
10300   CASE &H88:R=R+5       'SHIFT+DOWN
10310   CASE &H90:Z=Z+.1      'SHIFT+ENT
10320   CASE &HA0:Z=Z-.1      'SHIFT+ESC
```

```
10330   END SELECT
10340   GOSUB *SCROLL
10350 WEND
10360 END
10370 ' ------------------------------------------------------
10380 ' Subroutine for scroll, enlargement, reduction
10390 ' ------------------------------------------------------
10400 *SCROLL
10410   IF X> 1023 THEN X=1023
10420   IF X<-1023 THEN X=-1023
10430   IF Y> 1023 THEN Y=1023
10440   IF Y<-1023 THEN Y=-1023
10450   IF Z<.25  THEN Z=.3
10460   IF Z>512  THEN Z=512
10470   R=R MOD 360
10480   VZOOM Z
10490   VSCROLL X,Y,,,R
10500  RETURN
```

7-30 Displaying the Unit Status

The following program displays the status of connected unit.

```
10000 ' ======================================================
10010 ' Data definition
10020 ' ======================================================
10030 DIM UCODE%(10),UNAME$(10) 'Unit code and name
10040 ' ======================================================
10050 ' Main program
10060 ' ======================================================
10070 ' ------------------------------------------------------
10080 ' Unit code and name stored as variables
10090 ' ------------------------------------------------------
10100 I=0
10110 DO
10120   READ UCODE%(I),UNAME$(I)
10130   I=I+1
10140 LOOP WHILE (UCODE%(I-1)<>0)
10150 ' ------------------------------------------------------
10160 ' Display model type and version
10170 ' ------------------------------------------------------
10180 PRINT USING "Machine = F###@C##";GETVER(0),"-",GETVER(1)
10190 PRINT USING "System Ver #.##";GETVER(3)
10200 PRINT USING "OVL    Ver #.##";GETVER(2)
10210 PRINT
10220 ' ------------------------------------------------------
10230 ' Unit code and name display for slot 0 to 5
10240 ' ------------------------------------------------------
10250 FOR U=0 TO 5
10260   UC=GETVER(1,U)
10270   UN$=UNAME$(SEARCH(UCODE%,UC))
10280   PRINT USING " Slot(#) = @ @";U,RIGHT$("0"+HEX$(UC),2),UN$
10290 NEXT
10300 DATA 72,"F300-DC   :Parallel I/O Unit"
```

```
10310 ’ ======================================================
10320 ’ Data definition
10330 ’ ======================================================
10340 DATA 32,”F300–A20  :Normal Camera I/F Unit”
10350 DATA 33,”F300–A20R :Shutter Camera I/F Unit”
10360 DATA 40,”F300–A20S :Simultaneously Normal Camera I/F Unit”
10370 DATA 41,”F300–A20RS:Simultaneously Shutter Camera I/F Unit”
10380 DATA 48,”F300–FS   :Strobe I/F Unit”
10390 DATA 64,”F300–D    :Terminal Block Unit”
10410 DATA 80,”F300–E    :RS–232C I/F Unit”
10420 DATA 96,”F300–L100 :OVL Unit (model F300)”
10430 DATA 97,”F350–L100 :OVL Unit (model F350)”
10440 DATA 0,”Not connected”
```

SECTION 8
Troubleshooting

This section provides an alphabetical list of error messages and their causes and remedies.

8-1 Error Messages in Alphabetical Order

The following table shows error messages in alphabetical order. Refer to the table for the cause and remedy. (For error messages in the order of error code, refer to *8-2 Error Messages in Code Order*.)

	Error code	Error message	Cause and remedy
A	85	`Access denied`	An attempt is made to access a write-protected file. Clear the write protection using the SET command.
	102	`Application error (accum)`	Calculation formula is too complicated. Simplify the formula.
	101	`Application error (polish)`	Calculation formula is too complicated. Simplify the formula.
B	70	`Bad drive specification`	Drive designation is wrong. Designate a correct drive name. The drive designation for the memory card is “C:”.
	56	`Bad file name`	Either the file name or the path designation is wrong. Designate a correct file name or path name.
	52	`Bad file number`	The number of files designated is larger than the number of files that can be opened simultaneously. Designate the number of files in a range between 1 to 11.
C	124	`CASE without END SELECT`	CASE is used without END SELECT. Use CASE together with END SELECT.
	121	`CASE without SELECT`	CASE is used without SELECT. Use CASE together with SELECT.
	17	`Can’t continue`	An attempt is made to resume the program using the CONT command after stopping the program using STOP or CTRL+C command, or after rewriting the program. After rewriting the program, start it using the RUN comand.
	201	`Channel number error`	Designation of the Unit number is wrong. Designate a correct word (channel) and bit number.
	117	`Checksum error`	The program data backed up in the RAM is destroyed. Load the back-up programs in the memory card, etc.
	119	`Compile timing error`	The definition of user-defined functions FN is too long or too complicated. Either reduce the user-defined functions or simplify them.
	202	`Contact number error`	Designation of the bit number (contact number) for the Terminal Block and Parallel I/O Unit is wrong. Designate a correct bit number.
D	134	`DEF FN wihtout END DEF`	The block-type DEF FN is used without END DEF. Use DEF FN together with END DEF.
	135	`DO without LOOP`	DO is used without LOOP. Use DO together with LOOP.
	57	`Direct statement in file`	An attempt is made to load a program without any line number. Check if the program contents to be loaded are correct. Only files saved in ASCII can be loaded on the F350.
	64	`Disk I/O error`	Access the disk (Memory Card) is not possible. Either the disk has not been initialized or it is broken. Initialize the disk again using the setup menu or the FORMAT command.

	Error code	Error message	Cause and remedy
	68	`Disk full`	Available space in the Memory Card is insufficient. Either use another Memory Card with sufficient available space or delete unnecessary files using the KILL command.
	62	`Disk offline`	Either the Memory Card is not mounted or the cover of the MMI Unit is open. Check if the Memory Card is mounted properly. If it is mounted properly, the green indicator of the MMI Unit is lit.
	11	`Division by zero`	Division is made by 0. Modify the program so that division is not made by 0.
	10	`Duplicate definition`	An attempt is made to define the array statement which has been already declared. Either use a different variable name or delete the previous command using the ERASE command and then define again using the same variable name. The array variable that has been backed up using the VARBACK command should not be declared using the DIM command when the subsequent program is executed.
	31	`Duplicate label`	There are more than one label name in a program. Change the label name so that there is only one label in the program.
E	130	`ELSE without END IF`	ELSE is used without END IF. Use ELSE together with END IF.
	126	`ELSE without IF`	ELSE is used without IF. Use ELSE together with IF.
	129	`ELSEIF without END IF`	ELSE IF is used without END IF. Use ELSE IF together with END IF.
	125	`ELSEIF without IF`	ELSE IF is used without IF. Use ELSE IF together with IF.
	120	`END DEF without DEF FN`	END DEF is used without DEF FN. Use END DEF together with DEF FN.
	127	`END IF without IF`	END IF is used without IF. Use END IF together with IF.
	122	`END SELECT without SELECT`	END SELECT is used without SELECT. Use END SELECT together with SELECT.
	133	`END SUB without SUB`	END SUB is used without SUB. Use END SUB together with SUB.
	138	`EXIT without DEF`	EXIT is used without DEF. Use EXIT together with DEF.
	141	`EXIT without DO`	EXIT is used without DO. Use EXIT together with DO.
	140	`EXIT without FOR`	EXIT is used without FOR. Use EXIT together with FOR.
	139	`EXIT without SUB`	EXIT is used without SUB. Use EXIT together with SUB.

	Error code	Error message	Cause and remedy
F	50	`FIELD overflow`	Random access file record length is designated to be larger than 256 bytes in the FIELD command. Set the total of the field width to less than 255 bytes.
	26	`FOR without NEXT`	FOR is used without NEXT. Use FOR together with NEXT.
	33	`Feature not available`	Either the Unit to be used is not mounted or the function selected is not available with the current system. Mount the Unit properly. Don't use any functions not supported.
	65	`File already exist`	The new file name to be created using the NAME command already exists. Change either the new file name or the one that already exists.
	54	`File already open`	An attempt is made to execute the OPEN, KILL, or NAME command on a file that has been opened. Close the file using the CLOSE command.
	53	`File not found`	Designated file is not found. Check if the drive name and file name are correct, and also check if the designated file exists.
	60	`File not open`	An attempt is made to refer to an unopened file using PRINT# or INPUT#. Before referring to the file, open the file using the OPEN command.
	61	`File write protected`	An attempt was made to write on a file contained in the flash-type Memory Card or in the write-protected Memory Card. Clear the write protection or clear the write protection using the SET command.
	208	`Format error`	There is an error in the format of the file to be loaded using the RMLOAD, RMGLOAD, SMLOAD ro AMGLOAD command. Load a file that has been saved correctly.
I	128	`IF without END IF`	IF is used without END IF. Use IF together with END IF.
	12	`Illegal direct`	An attempt is made to execute the command that cannot be used in the direct mode such as CALL or SUB. Use the command in program mode.
	214	`Illegal figure`	The ROI model could not be registered using the RMPUT2 command because the region width was narrower than the model width. Enlarge the width of the region.
	5	`Illegal function call`	The designated argument of the command or function exceeds the allowable range due to one of the following reasons: • Use of an array not declared by DIM. • The array dimension is different. • The character string is a null string. • Size restrictions are inconsistent with designated coordinates in "*SETBOX()". • Tried to execute a combination of functions that cannot be used with IMGSAVE command, etc. Refer to *Section 5 Reference* for proper use of commands and functions. When the error occurs, display the arguments to check if correct arguments are used.

	Error code	Error message	Cause and remedy
	205	`Illegal group number`	The value designated in the search group number or ROI group number is outside the allowable range. Designate a correct group number.
	137	`Illegal jump`	Jumping out of the structured subroutine or to an illegal destination. Check to be sure that no conflict occurs in the jumping destination.
	204	`Illegal model number`	The value designated in the search model number or ROI model number is outside the allowable range. Designate a correct model number.
	207	`Illegal model size`	The search model size or ROI model size is either too small or too large. Designate a model size within the allowable range.
	74	`Illegal operation`	Insertion cannot be made properly in the insert mode. Clear the insert mode and then perform key operations.
	206	`Illegal unit number`	The number designated in the measurement unit number for sequence processing measurement is not within the allowable range. Designate a correct measurement unit number.
	55	`Input past end`	After reading all the data in a file using the INPUT# or GET# command, an attempt is made to input new data. Match the number of data in the file with the number of data to be input or detect the end of the file using the EOF or LOF function.
	103	`Internal error`	Either the communications buffer via RS-232C is full or there is an error in the transmission data when transmitting via SO/SI cord. Check the communications programs and the data to be transmitted.
L	136	`LOOP without DO`	LOOP is used without DO. Use LOOP together with DO.
	23	`Line buffer overflow`	An attempt is made to input more characters than the allowable range (255 bytes) per line. This error often occurs when reading data received via the RS-232C. Use the INPUT$ function to input only the allowed bytes one by one.
	131	`Local variable overflow`	The number of internal variables used in the structured subroutine is too large. Either reduce the internal variables or use the DEFGLOBAL command to treat them as a global variable.
M	22	`Missing operand`	Required arguments are not designated in the command or function. Check the format and designate required arguments.
N	1	`NEXT without FOR`	NEXT is used without FOR. Use NEXT together with FOR.
	19	`No RESUME`	Program cannot be executed because no RESUME command is provided at the end of error processing routine. End the program using the RESUME, END, or ON ERROR GOTO 0 command.

	Error code	Error message	Cause and remedy
O	100	`OV`	Input data exceeds the allowable range. Check to be sure that the variable range will not cause an overflow.
	4	`Out of DATA`	The data to be read by the READ command is not defined. Check if the number of data to be read by the READ command is the same as that defined in DATA. Also, check if the position designated by the RESORTRE command is correct.
	110	`Out of compile work space`	The work area for pseudo-compiling, which is executed internally by the system prior to a RUN command, is insufficient. This occurs when the number of labelling, loop processing, and structuring statements is too large. Enlarge the compiling work area using the IPL command or reduce the number of labelling, loop processing, and structuring statements.
	7	`Out of memory`	The internal memory capacity of the system is insufficient due to following possible causes: • The program is too large. • Too many arrays or character-type variables are used. • The nesting in loop processing is too deep. • The image is too large for the IMGLOAD or IMGSAVE command. Check the remaining size of the program area and variables area using the FRE function. Take the following countermeasures depending on the cause. • Use multiple statements and shorten variable names to reduce the program size. • Execute the CLEAR command or restart the OVL Unit to initialize the variables area. • Modify the program structure so that the nesting in the loop processing doesn't become too deep. • Reduce the size of images to be saved or loaded or use the compression function.
	211	`Out of model memory`	ROI model cannot be registered because the remaining area available for registering the ROI model is insufficient. Delete unnecessary ROI models using the RMCLEAR command or reduce the size of ROI models to be registered.
	212	`Out of model number`	An attempt is made to allocate the search model number and ROI model number, which are not within the specified range, by using the RMGLOAD or SMGLOAD command. Load the models after securing unregistered models for the number of models contained in the file to be loaded.
	14	`Out of string space`	The total number of character strings exceeds the memory capacity. Reduce the number of character strings and array character strings.
	6	`Overflow`	Computed results or input values are exceeding the allowable ranges. They are not designated in long integer type for address setting. Check if the data type fits the value range.
P	76	`Path not found`	Wrong path number is designated. Check if the designated path number exists.
	75	`Path/File access error`	An attempt is made to execute file operation for a directory. Designate a correct path name.

	Error code	Error message	Cause and remedy
R	20	`RESUME without error`	RESUME command is used in a routine not for error processing. Use the RESUME command in a routine that is designated by the ON ERROR GOTO command. Provide END at the end of the main routine.
	3	`RETURN without GOSUB`	RETURN is used without GOSUB. Use GOSUB together with RETURN.
	73	`Rename across disks`	An attempt is made to use the NAME command for a file in a different drive. Execute the NAME command for a file in the same drive.
S	123	`SELECT without END SELECT`	SELECT is used without END SELECT. Use SELECT together with END SELECT.
	132	`SUB without END SUB`	SUB is used without END SUB. Use SUB together with END SUB.
	210	`Sequence is running`	An attempt is made to change the measuring conditions while the sequencial measurement processing is being executed. Stop the sequencial measurement using SQRUN 0 and then execute the command.
	209	`Sequence syntax error`	The format for setting measuring conditions for sequencial measurement designated by the SQSET command is wrong. The error found first will be displayed. Refer to *Section 5 Reference* and correct the error.
	59	`Sequential I/O only`	An attempt is made to use an I/O command other than the sequential I/O command. Use the sequential I/O command.
	15	`String too long`	The number of characters to be substituted for the character variable exceeds 255 bytes. Divide them into two variables.
	9	`Subscript out of range`	The subscript of an array variable is beyond the range specified by the DIM and OPTION BASE command. The value of the subscript of an array variable with DIM declaration omitted exceeds 10. The number of array elements is insufficient. Check the subscript range defined in the array variable and also the array variable range to be used.
	2	`Syntax error`	Commands or functions are not used as specified in the format. Commands or functions not specified in the keywords are used. Use them as specified in the format.
T	144	`Table file format error`	The format for the user's command and function file is wrong. Create a correct user's command and function file.
	213	`Time out error`	When the DBOUT command is used with the handshake set to ON, DSA signal cannot be input even after the set time. Input the DSA signal correctly or extend the time-out time.
	13	`Type mismatch`	The variable type is inconsistent on the left-side and right-side of a formula, and in the argument of a function. Designate a correct variable type.

	Error code	Error message	Cause and remedy
U	32	`Undefined label`	An attempt is made to refer to an undefined label. Define the label correctly.
	8	`Undefined line number`	An attempt is made to jump to a line number that doesn't exist. Check if the line number is correct.
	18	`Undefined user function`	An attempt is made to use an undefined user function or machine language. Define them using the DEF FN or DEF USR command.
	99	`Unprintable error`	Any message that is not defined for an error. When created by using the ERROR command, the error (error code no.99) can be used for user's own error processing.
	142	`User program load error`	Program cannot be loaded properly because either the machine language program for the user's commands and functions is too large or the address for loading is incorrect. Check if the address for loading is correct and the program size is appropriate.
	143	`User table over flow`	The newly defined user's commands and functions are too large. Reduce the number of user's commands and functions.
V	200	`Vision error`	Possible causes are as follows: • When the number of contour lines of a graphic pattern designated by the EDGRJECT command exceeds 255 or the length of the contour line exceeds 4096 • When the profile of a hole designated by the HFILL command is complicated, the number of holes exceeds 255, or the length of the contour line exceeds 4096 • When the number of labels designated by the labelling command is 0 or more than 256 • When an attempt is made to execute a labelling-related command before executing labelling • When an attempt is made to execute a measurement-related command while one measurement command is being executed Take the following countermeasures: • As for the EDGRJECT or HFILL command, simplify objects to be processed or divide the processing areas. • As for the labelling commands, eliminate noise or reduce the processing areas to minimize the number of labels. If the number of labels is too large, "–1" is inverted by the LNUM function. • Use the VDWAIT command to wait until the measurement related command is completed.
W	30	`WEND without WHILE`	WEND is used without WHILE. Use WEND together with WHILE.
	29	`WHILE without WEND`	WHILE is used without WEND. Use WHILE together with WEND.

8-2 Error Messages in Code Order

Error code	Error message	Cause and remedy
1	`NEXT without FOR`	NEXT is used without FOR. Use NEXT together with FOR.

Error code	Error message	Cause and remedy
2	`Syntax error`	Commands or functions are not used as specified in the format. Commands or functions not specified in the keywords are used. Use them as specified in the format.
3	`RETURN without GOSUB`	RETURN is used without GOSUB. Use GOSUB together with RETURN.
4	`Out of DATA`	The data to be read by the READ command is not defined. Check if the number of data to be read by the READ command is the same as that defined in DATA. Also, check if the position designated by the RESORTRE command is correct.
5	`Illegal function call`	The designated argument of the command or function exceeds the allowable range due to one of the following reasons: • Use of an array not declared by DIM. • The array dimension is different. • The character string is a null string. • Size restrictions are inconsistent with designated coordinates in "*SETBOX()". • Tried to execute a combination of functions that cannot be used with IMGSAVE command, etc. Refer to *Section 5 Reference* for proper use of commands and functions. When the error occurs, display the arguments to check if correct arguments are used.
6	`Overflow`	Computed results or input values are exceeding the allowable ranges. They are not designated in long integer type for address setting. Check if the data type fits the value range.
7	`Out of memory`	The internal memory capacity of the system is insufficient due to following possible causes: • The program is too large. • Too many arrays or character-type variables are used. • The nesting in loop processing is too deep. • The image is too large for the IMGLOAD or IMGSAVE command. Check the remaining size of the program area and variables area using the FRE function. Take the following countermeasures depending on the cause. • Use multiple statements and shorten variable names to reduce the program size. • Execute the CLEAR command or restart the OVL Unit to initialize the variables area. • Modify the program structure so that the nesting in the loop processing doesn't become too deep. • Reduce the size of images to be saved or loaded or use the compression function.
8	`Undefined line number`	An attempt is made to jump to a line number that doesn't exist. Check if the line number is correct.
9	`Subscript out of range`	The subscript of an array variable is beyond the range specified by the DIM and OPTION BASE command. The value of the subscript of an array variable with DIM declaration omitted exceeds 10. The number of array elements is insufficient. Check the subscript range defined in the array variable and also the array variable range to be used.

Error code	Error message	Cause and remedy
10	`Duplicate definition`	An attempt is made to define the array statement which has been already declared. Either use a different variable name or delete the previous command using the ERASE command and then define again using the same variable name. The array variable that has been backed up using the VARBACK command should not be declared using the DIM command when the subsequent program is executed.
11	`Division by zero`	Division is made by 0. Modify the program so that division is not made by 0.
12	`Illegal direct`	An attempt is made to execute the command that cannot be used in the direct mode such as CALL or SUB. Use the command in program mode.
13	`Type mismatch`	The variable type is inconsistent on the left-side and right-side of a formula, and in the argument of a function. Designate a correct variable type.
14	`Out of string space`	The total number of character strings exceeds the memory capacity. Reduce the number of character strings and array character strings.
15	`String too long`	The number of characters to be substituted for the character variable exceeds 255 bytes. Divide them into two variables.
17	`Can't continue`	An attempt is made to resume the program using the CONT command after stopping the program using STOP or CTRL+C command, or after rewriting the program. After rewriting the program, start it using the RUN comand.
18	`Undefined user function`	An attempt is made to use an undefined user function or machine language. Define them using the DEF FN or DEF USR command.
19	`No RESUME`	Program cannot be executed because no RESUME command is provided at the end of error processing routine. End the program using the RESUME, END, or ON ERROR GOTO 0 command.
20	`RESUME without error`	RESUME command is used in a routine not for error processing. Use the RESUME command in a routine that is designated by the ON ERROR GOTO command. Provide END at the end of the main routine.
22	`Missing operand`	Required arguments are not designated in the command or function. Check the format and designate required arguments.
23	`Line buffer overflow`	An attempt is made to input more characters than the allowable range (255 bytes) per line. This error often occurs when reading data received via the RS-232C. Use the INPUT$ function to input only the allowed bytes one by one.
26	`FOR without NEXT`	FOR is used without NEXT. Use FOR together with NEXT.
29	`WHILE without WEND`	WHILE is used without WEND. Use WHILE together with WEND.
30	`WEND without WHILE`	WEND is used without WHILE. Use WEND together with WHILE.

Error code	Error message	Cause and remedy
31	`Duplicate label`	There are more than one label name in a program. Change the label name so that there is only one label in the program.
32	`Undefined label`	An attempt is made to refer to an undefined label. Define the label correctly.
33	`Feature not available`	Either the Unit to be used is not mounted or the function selected is not available with the current system. Mount the Unit properly. Don't use any functions not supported.
50	`FIELD overflow`	Random access file record length is designated to be larger than 256 bytes in the FIELD command. Set the total of the field width to less than 255 bytes.
52	`Bad file number`	The number of files designated is larger than the number of files that can be opened simultaneously. Designate the number of files in a range between 1 to 11.
53	`File not found`	Designated file is not found. Check if the drive name and file name are correct, and also check if the designated file exists.
54	`File already open`	An attempt is made to execute the OPEN, KILL, or NAME command on a file that has been opened. Close the file using the CLOSE command.
55	`Input past end`	After reading all the data in a file using the INPUT# or GET# command, an attempt is made to input new data. Match the number of data in the file with the number of data to be input or detect the end of the file using the EOF or LOF function.
56	`Bad file name`	Either the file name or the path designation is wrong. Designate a correct file name or path name.
57	`Direct statement in file`	An attempt is made to load a program without any line number. Check if the program contents to be loaded are correct. Only files saved in ASCII can be loaded on the F350.
59	`Sequential I/O only`	An attempt is made to use an I/O command other than the sequential I/O command. Use the sequential I/O command.
60	`File not open`	An attempt is made to refer to an unopened file using PRINT# or INPUT#. Before referring to the file, open the file using the OPEN command.
61	`File write protected`	An attempt was made to write on a file contained in the flash-type Memory Card or in the write-protected Memory Card. Clear the write protection or clear the write protection using the SET command.
62	`Disk offline`	Either the Memory Card is not mounted or the cover of the MMI Unit is open. Check if the Memory Card is mounted properly. If it is mounted properly, the green indicator of the MMI Unit is lit.
64	`Disk I/O error`	Access the disk (Memory Card) is not possible. Either the disk has not been initialized or it is broken. Initialize the disk again using the setup menu or the FORMAT command.
65	`File already exist`	The new file name to be created using the NAME command already exists. Change either the new file name or the one that already exists.

Error code	Error message	Cause and remedy
68	`Disk full`	Available space in the Memory Card is insufficient. Either use another Memory Card with sufficient available space or delete unnecessary files using the KILL command.
70	`Bad drive specification`	Drive designation is wrong. Designate a correct drive name. The drive designation for the memory card is “C:”.
73	`Rename across disks`	An attempt is made to use the NAME command for a file in a different drive. Execute the NAME command for a file in the same drive.
74	`Illegal operation`	Insertion cannot be made properly in the insert mode. Clear the insert mode and then perform key operations.
75	`Path/File access error`	An attempt is made to execute file operation for a directory. Designate a correct path name.
76	`Path not found`	Wrong path number is designated. Check if the designated path number exists.
85	`Access denied`	An attempt is made to access a write-protected file. Clear the write protection using the SET command.
99	`Unprintable error`	Any message that is not defined for an error. When created by using the ERROR command, the error (error code no.99) can be used for user’s own error processing.
100	`OV`	Input data exceeds the allowable range. Check to be sure that the variable range will not cause an overflow.
101	`Application error (polish)`	Calculation formula is too complicated. Simplify the formula.
102	`Application error (accum)`	Calculation formula is too complicated. Simplify the formula.
110	`Out of compile work space`	The work area for pseudo-compiling, which is executed internally by the system prior to a RUN command, is insufficient. This occurs when the number of labelling, loop processing, and structuring statements is too large. Enlarge the compiling work area using the IPL command or reduce the number of labelling, loop processing, and structuring statements.
117	`Checksum error`	The program data backed up in the RAM is destroyed. Load the back-up programs in the memory card, etc.
119	`Compile timing error`	The definition of user-defined functions FN is too long or too complicated. Either reduce the user-defined functions or simplify them.
120	`END DEF without DEF FN`	END DEF is used without DEF FN. Use END DEF together with DEF FN.
121	`CASE without SELECT`	CASE is used without SELECT. Use CASE together with SELECT.
122	`END SELECT without SELECT`	END SELECT is used without SELECT. Use END SELECT together with SELECT.
123	`SELECT without END SELECT`	SELECT is used without END SELECT. Use SELECT together with END SELECT.
124	`CASE without END SELECT`	CASE is used without END SELECT. Use CASE together with END SELECT.

Error code	Error message	Cause and remedy
125	ELSEIF without IF	ELSE IF is used without IF. Use ELSE IF together with IF.
126	ELSE without IF	ELSE is used without IF. Use ELSE together with IF.
127	END IF without IF	END IF is used without IF. Use END IF together with IF.
128	IF without END IF	IF is used without END IF. Use IF together with END IF.
129	ELSEIF without END IF	ELSE IF is used without END IF. Use ELSE IF together with END IF.
130	ELSE without END IF	ELSE is used without END IF. Use ELSE together with END IF.
131	Local variable overflow	The number of internal variables used in the structured subroutine is too large. Either reduce the internal variables or use the DEFGLOBAL command to treat them as a global variable.
132	SUB without END SUB	SUB is used without END SUB. Use SUB together with END SUB.
133	END SUB without SUB	END SUB is used without SUB. Use END SUB together with SUB.
134	DEF FN wihtout END DEF	The block-type DEF FN is used without END DEF. Use DEF FN together with END DEF.
135	DO without LOOP	DO is used without LOOP. Use DO together with LOOP.
136	LOOP without DO	LOOP is used without DO. Use LOOP together with DO.
137	Illegal jump	Jumping out of the structured subroutine or to an illegal destination. Check to be sure that no conflict occurs in the jumping destination.
138	EXIT without DEF	EXIT is used without DEF. Use EXIT together with DEF.
139	EXIT without SUB	EXIT is used without SUB. Use EXIT together with SUB.
140	EXIT without FOR	EXIT is used without FOR. Use EXIT together with FOR.
141	EXIT without DO	EXIT is used without DO. Use EXIT together with DO.
142	User program load error	Program cannot be loaded properly because either the machine language program for the user's commands and functions is too large or the address for loading is incorrect. Check if the address for loading is correct and the program size is appropriate.
143	User table over flow	The newly defined user's commands and functions are too large. Reduce the number of user's commands and functions.
144	Table file format error	The format for the user's command and function file is wrong. Create a correct user's command and function file.

Error code	Error message	Cause and remedy
200	Vision error	Possible causes are as follows: • When the number of contour lines of a graphic pattern designated by the EDGRJECT command exceeds 255 or the length of the contour line exceeds 4096 • When the profile of a hole designated by the HFILL command is complicated, the number of holes exceds 255, or the length of the contour line exceeds 4096 • When the number of labels designated by the labelling command is 0 or more than 256 • When an attempt is made to execute a labelling-related command before executing labelling • When an attempt is made to execute a measurement-related command while one measurement command is being executed Take the following countermeasures: • As for the EDGRJECT or HFILL command, simplify objects to be processed or divide the processing areas. • As for the labelling commands, eliminate noise or reduce the processing areas to minimize the number of labels. If the number of labels is too large, “–1” is inverted by the LNUM function. • Use the VDWAIT command to wait until the measurement related command is completed.
201	Channel number error	Designation of the Unit number is wrong. Designate a correct word (channel) and bit number.
202	Contact number error	Designation of the bit number (contact number) for the Terminal Block and Parallel I/O Unit is wrong. Designate a correct bit number.
204	Illegal model number	The value designated in the search model number or ROI model number is outside the allowable range. Designate a correct model number.
205	Illegal group number	The value designated in the search group number or ROI group number is outside the allowable range. Designate a correct group number.
206	Illegal unit number	The number designated in the measurement unit number for sequence processing measurement is not within the allowable range. Designate a correct measurement unit number.
207	Illegal model size	The search model size or ROI model size is either too small or too large. Designate a model size within the allowable range.
208	Format error	There is an error in the format of the file to be loaded using the RMLOAD, RMGLOAD, SMLOAD ro AMGLOAD command. Load a file that has been saved correctly.
209	Sequence syntax error	The format for setting measuring conditions for sequencial measurement designated by the SQSET command is wrong. The error found first will be displayed. Refer to *Section 5 Reference* and correct the error.
210	Sequence is running	An attempt is made to change the measuring conditions while the sequencial measurement processing is being executed. Stop the sequencial measurement using SQRUN 0 and then execute the command.
211	Out of model memory	ROI model cannot be registered because the remaining area available for registering the ROI model is insufficient. Delete unnecessary ROI models using the RMCLEAR command or reduce the size of ROI models to be registered.

Error code	Error message	Cause and remedy
212	`Out of model number`	An attempt is made to allocate the search model number and ROI model number, which are not within the specified range, by using the RMGLOAD or SMGLOAD command. Load the models after securing unregistered models for the number of models contained in the file to be loaded.
213	`Time out error`	When the DBOUT command is used with the handshake set to ON, DSA signal cannot be input even after the set time. Input the DSA signal correctly or extend the time-out time.
214	`Illegal figure`	The ROI model could not be registered using the RMPUT2 command because the region width was narrower than the model width. Enlarge the width of the region.

Appendix A
Reserved Words

Reserved words						
A	ABS	AKCNV$	ALL	ANDOUT	APPEND	ARC
	ARRYFUNC	AS	ASC	ATN	ATTR$	AUTO
	AUTOLVL					
B	BACKDISP	BASE	BATCHK	BCDTOBIN	BCOPY	BCOPY2
	BEDGE	BEEP	BINFO	BINTOBCD	BLOAD	BMFUNC
	BOX	BSAVE	BUSY			
C	CALL	CAMCHK	CAMERA	CAMMODE	CAMSYNC	CANCEL
	CASE	CDBL	CGPAGE	CHAIN	CHDIR	CHR$
	CHRCUT	CHRCUTM	CINT	CIRCLE	CLEAR	CLNG
	CLOSE	CLS	CLRUINFO	COLOR	COLOR@	COM
	COMMON	CONKEY	CONSOLE	CONT	COPY	COS
	CSNG	CSRLIN	CURSOR	CVD	CVI	CVL
	CVS					
D	DATA	DATE$	DBOUT	DCASE	DEF	DEFDBL
	DEFGLOBAL	DEFINT	DEFLNG	DEFSNG	DEFSTR	DELETE
	DENY	DEVICE	DGOTO	DILA	DIM	DIN
	DISPLAY	DNEXT	DO	DOUT	DSA	DSKF
E	EDGRJECT	EDIT	ELIM	ELLIPSE	ELSE	ELSEIF
	END	ENHANCE	EOF	ERASE	ERL	EROS
	ERR	ERRMSG	ERROR	ERROUT	EXIT	EXP
F	FCOPY	FIELD	FILES	FILTDATA	FILTER	FILTERIN
	FILTSEL	FIND	FIX	FLASH	FMODE	FN
	FOR	FORMAT	FRE			
G	GATE	GCOPY	GCOPY2	GDILA	GEROS	GET
	GET@	GETBLUT	GETDLUT	GETDLVL	GETLUT	GETMDATA
	GETVER	GETUINFO	GETUINFO$	GMFUNC	GOSUB	GO
	GOTO					
H	HELP	HEX$	HFILL	HISTGRAM		
I	IF	IMGLOAD	IMGSAVE	INKEY$	INPUT	INPUT$
	INSTR	INT	INTR	IPL		
J	JIS$					
K	KACNV$	KEXT$	KEY	KEYIN	KILL	KINPUT
	KINSTR	KLEN	KMID$	KNJ$	KPLOAD	KPOS
	KTYPE					
L	LABLE	LBOUND	LCASE$	LDATA	LEFT$	LEN
	LET	LEVEL	LFILES	LHELP	LINE	LIST
	LLIST	LNUM	LOAD	LOC	LOCATE	LOF
	LOG	LOOP	LPOINT	LPOS	LPRINT	LPUTIMG
	LSET	LSORT	LSTYLE	LTRIM$		
M	MASK3	MASKBIT	MDATA	MEASURE	MEDIAN	MENU
	MERGE	MID$	MKDIR	MKD$	MKI$	MKL$
	MKS$	MMODE	MON			
N	NAME	NEW	NPIECE			
O	OCT$	ON	OFF	OROUT	OPEN	OPTION
	OUTPUT					
P	PASSWD	PEEK	PEEKDATA	PIECE$	PIN	POINT
	POKE	POLYGON	POLYLINE	POKEDATA	POS	POUT
	PRINT	PSET	PUT	PUT@		

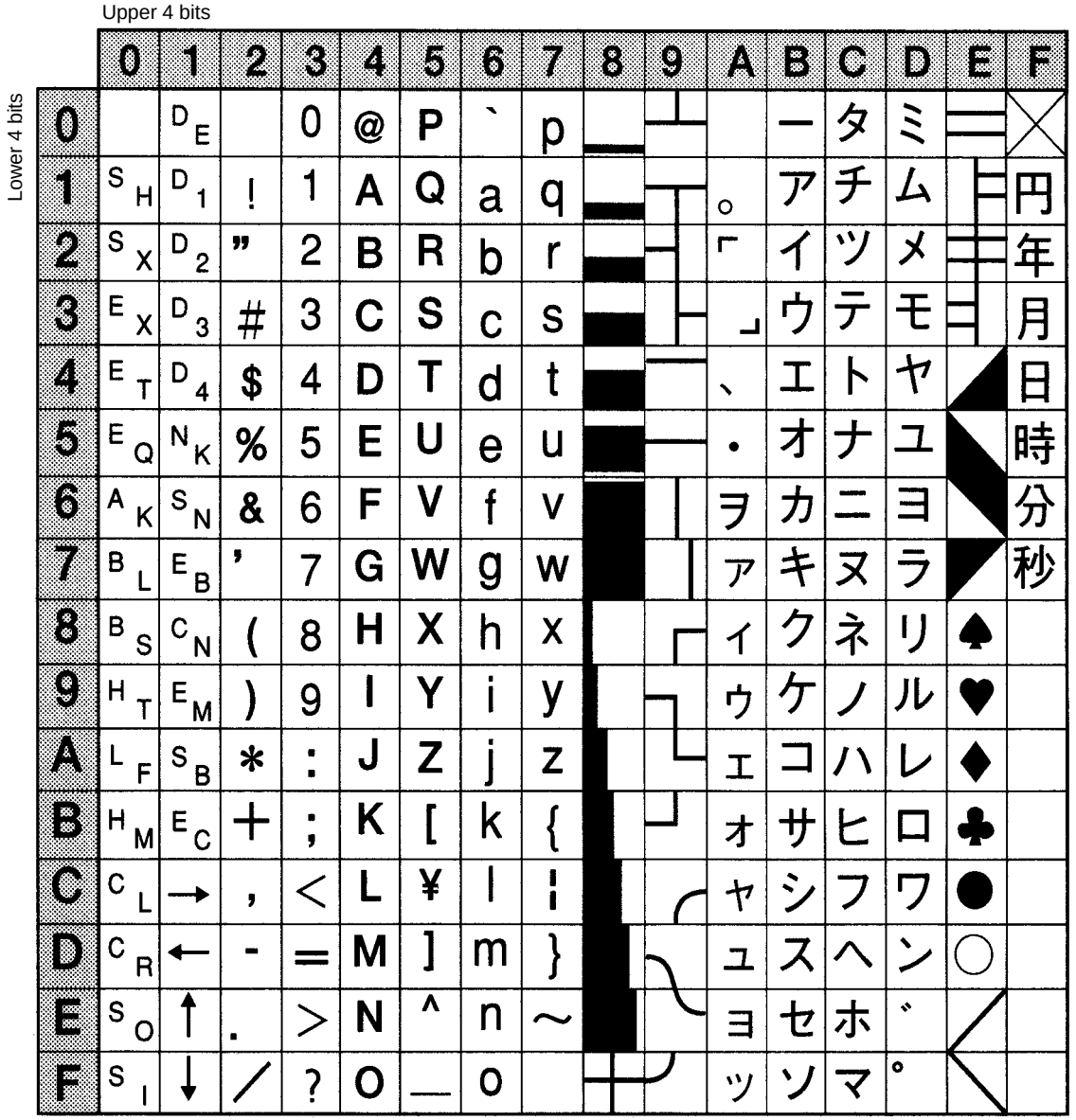
Reserved words						
R	RANDOMIZE	RDATA	READ	REM	RENUM	REPEAT
	REPLACE	RESTORE	RESUME	RETURN	RIGHT$	RMCLEAR
	RMCOPY	RMDATA	RMDIR	RMDRAW	RMDRAW2	RMEXTSET
	RMEXTGET	RMEXTGET$	RMGCLEAR	GRMGDEL	RMGINFO	RMGJUDGE
	RMGLOAD	RMGMODE	RMGROUP	RMGRUN	RMGSAVE	RMINFO
	RMINFO$	RMJUDGE	RMLOAD	RMMDATA	RMMODE	RMMODE2
	RMNAME	RMODE	RMORIGIN	RMPOSCNV	RMPUT	RMPUT2
	RMPUT3	RMRUN	RMSAVE	RMTOSM	RND	RNEXT
	RSET	RTRIM$	RUN	RUNOUT		
S	SAVE	SBACK	SEARCH	SEARCH2	SEGPTR	SELECT
	SET	SETBLUT	SETDLUT	SETDLVL	SETLUT	SETUINFO
	SFTBLUT	SFTLUT	SGN	SGTIME	SIN	SLABEL
	SMAREA	SMCLEAR	SMCOPY	SMDATA	SMGAREA	SMGCLEAR
	SMGDATA	SMGDEL	SMGINFO	SMGLOAD	SMGMODE	SMGROUP
	SMGRUN	SMGSAVE	SMINFO	SMINFO$	SMLOAD	SMMDATA
	SMMDATA2	SMMODE	SMNAME	SMPUT	SMPUT2	SMROTATE
	SMRUN	SMSAVE	SMSELECT	SMSELECT2	SMSORT	SOBEL
	SPACE$	SPC	SPCLOSE	SPLINE	SQINFO	SQINIT
	SQJDATA	SQMDATA	SQMEAS	SQMODE	SQPDATA	SQR
	SQRUN	SQSET	SQSETLUT	SQSTAT	SQTIME	SQUNITNO
	SSCROLL	STEP	STOP	STR$	STRCHK	STRING$
	STRMODE	SUB	SWAP	SYSINIT	SYSTEM	SZOOM
T	TAB	TAN	THEN	THIN	TIME$	TIMER
	TO	TROFF	TRON			
U	UBOUND	UCASE$	UNTIL	USING	USR	USR0
	USR1	USR2	USR3	USR4	USR5	USR6
	USR7	USR8	USR9			
V	VAL	VARBACK	VARLOAD	VARPTR	VARSAVE	VDWAIT
	VENUS	VIDEOIN	VSCROLL	VZOOM		
W	WAIT	WDILA	WDISP	WEND	WEROS	WHILE
	WIDTH	WPBIT	WRITE	WSCROLL	WZOOM	

Appendix B
Induction Functions

(P# = π = 3.14159265358979)

Function	Derivation formula using system-defined functions
logaX	LOG (X)/LOG (a)
secX	1/COS (X)
cosecX	1/SN (X)
sotX	1/TAN (X)
$\sin^{-1}$X	ATN (X/SQR (–X*X+1))
$\cos^{-1}$X	–ATN (X/SQR (–X*X+1))+P#/2
$\sec^{-1}$X	ATN (SQR (X*X–1)+(SGN (X)–1)*P#/2
cosec^{-1}X	ATN (1/SQR (X*X–1)+(SGN (X)–1)*P#/2
$\cot^{-1}$X	–ATN (X)+P#/2
sinhX	(EXP (X)–EXP (–X))/2
coshX	(EXP (X)+EXP (–X))/2
tanhX	–EXP (–X)/(EXP (X)+EXP (–X))*2+1
sechX	2/(EXP (X)+EXP (–X))
cosechX	2/(EXP (X)–EXP (–X))
cothX	EXP (–X)/(EXP (X)–EXP (–X))*2+1
$\sinh^{-1}$X	LOG (X+SQR (X*X+1))
$\cosh^{-1}$X	LOG (X+SQR (X*X–1))
$\tanh^{-1}$X	LOG ((1+X)/(1–X))/2
sech^{-1}X	LOG (SQR (–X*X+1)+1)+1/X
cosech^{-1}X	LOG ((SGN (X)*SQR (X*X+1)+1)/X)
$\coth^{-1}$X	LOG ((X+1)/(X–1))/2

Appendix C
List of Characters and Codes



Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

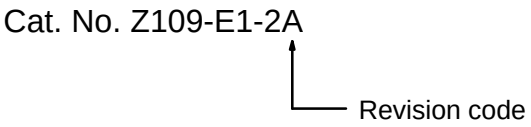
U

V

W

Revision History

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



The following table outlines the changes made to the manual during each revision. Page numbers refer to the previous version.

Revision code	Date	Revised content
1	August 1995	Original production
2	March 1997	Manual revised for version 2 of the product.
2A	January 2001	Page 202: Thinning number range corrected.