DENSO ROBOT

SUPPLEMENT

Main System Software Version 2.2*

Preface

DENSO WAVE has updated main system software designed for DENSO robot series from Version 2.0* to Version 2.2*.

This book is a supplement to the DENSO robot manuals. It describes newly added and updated functions. Use this supplement together with other robot manuals.

For new optional devices, refer to the "Options Manual for RC7 Controller" separately issued.

Products covered by this manual

Robot series configured with RC7 robot controller Version 2.2* or later

Contents

1 USB Flash Memory Newly Supported

The robot controller RC7 newly supports a USB flash memory drive.

The following USB flash memory drives are available to the controller.

Manufacturer	Model
I-O DATA	USB-###ED2, EDP-###M, EDC-###M
Logitec	LMC-###UDA

Note: ### denotes the capacity.

1.1 Displaying the USB Memory Access Menu

Access: [F6 Set]—[F3 USB Memory]

Displays the USB memory access menu from which you can access a USB flash memory drive.

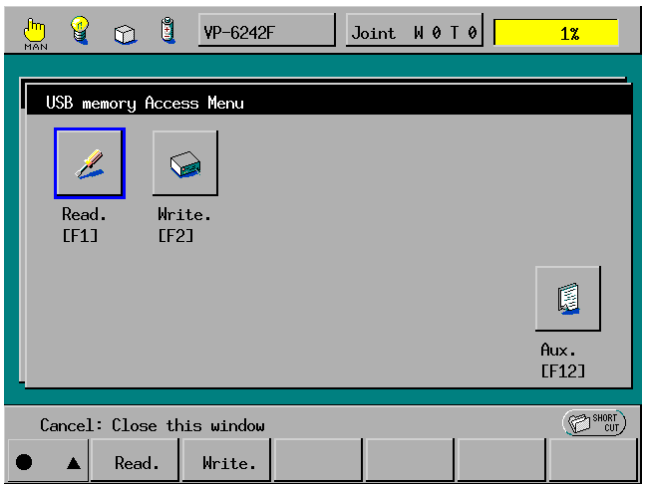
The following conditions should be satisfied for using a USB flash memory.

- The robot controller is in Manual mode.
- The motor power is off.
- No program is running.

NOTE: The frequency of the connection/disconnection of a USB flash memory drive is restricted to 10 times unless the robot controller restarts. If it exceeds the limit, you need to turn the controller power off and on.

NOTE: Never remove a USB flash memory drive or turn the controller power off when the USB flash memory drive is being accessed. (For details about the USB flash memory access status, refer to the memory's user's manual.)

(1) Press [F3 USB Memory] in the Setting (Main window), and the USB Memory Access Menu appears as shown below.



F1 F2

(2) Select the desired operation. The corresponding window appears as shown on pages 4 and 6.

Data that can be handled by USB flash memory drive

The table below lists data that can be handled by USB flash memory drives. Select the appropriate data to read or write as necessary.

Data Type	File or Data	Remarks
Source program data	Source program files (PAC, H, PNL) Executable files (NIC, MAP) Settings files (DAT)	Only files with their compile flags active ("Enable" in the Use column) can be written into USB flash memory.
Variables data	All global data Number of variables used	Reading variables data into the robot controller automatically changes the "number of variables used" in the controller.
I/O data	I/O settings Settings for expansion board	
Arm data	Arm parameters Tool/work/area coordinates definition	• Never read in arm data prepared for other robots. • Tool and work data modified by TOOL or WORK command will not be updated when written into the memory. To write updated data, first save the system parameters (see the SETTING-UP MANUAL, p. 5-179) and then write data into the USB flash memory.
Visual-related data	Visual equipment settings	Write (to USB flash memory) only.
Log data	Communications settings Version information Various log data	

Data exchange between robot controller and WINCAPSII

Data can be exchanged between the robot controller and WINCAPSII by means of a USB flash memory drive.

For the operating procedures in WINCAPSII, refer to WINCAPSII Guide, Chapter 4, Sections 4.3.4 and 4.3.5.

USB memory data modification not allowed

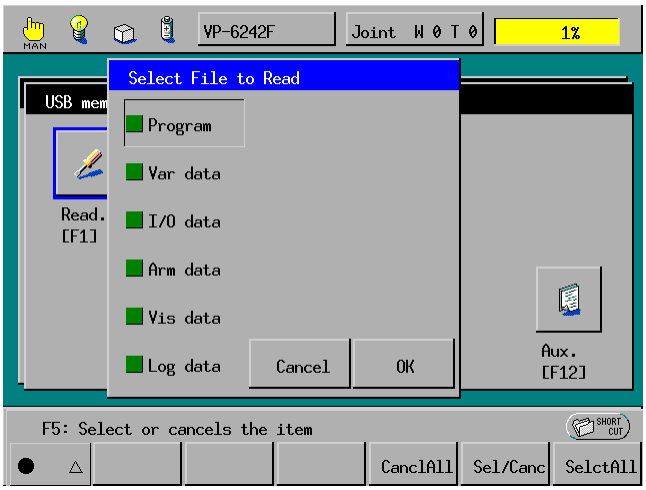
Never modify data stored in the USB flash memory drive from the robot controller. Any modification makes it impossible to access the memory because USB memory data contains check codes used for checking data corruption and guaranteeing accurate data read/write.

1.2 Reading USB Memory Data into Robot Controller

Access: [F6 Set]—[F3 USB Memory]—[F1 Read.]

Reads data stored in a USB flash memory into the robot controller.

(1) Press [F1 Read.] in the USB Memory Access Menu, and the Select File to Read window appears as shown below.



(2) Select data to read from the USB flash memory and then press the OK button.

⚠ **Caution:** Never read in arm data prepared for other robots. Doing so will cause the robot to malfunction. It is very DANGEROUS.

Data reading from the USB flash memory will start.

(3) Upon completion of reading, restart the robot controller.

To read program data, see "Note on reading program data from USB flash memory" on the next page.

⚠**Caution:** Without restarting the robot controller, the robot may malfunction.

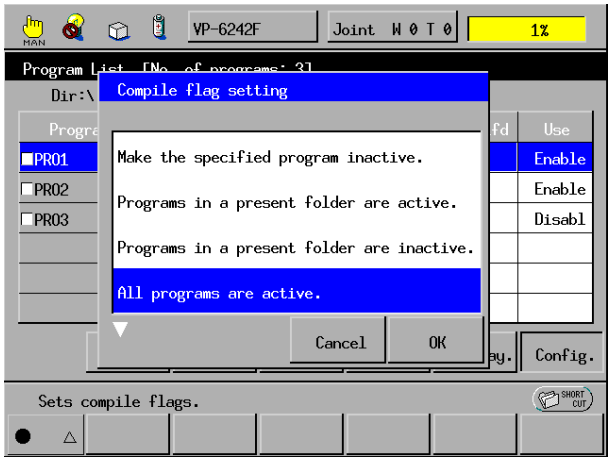
Notes on reading *program* data from USB flash memory

When reading *program* data from USB flash memory, the controller system first deletes all program data and executable data stored in itself and then starts reading *program* data from the USB flash memory.

Therefore, for selected data containing program data but no executable data, you need to first compile the program data read and then load the compiled ones using the steps below

1) On the top screen of the teach pendant, press [F1 Program] to display the Program List window.

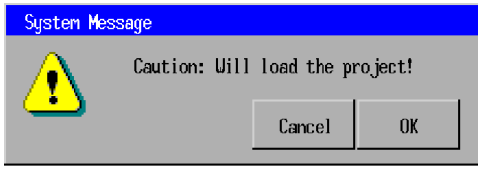
2) Press the Config. button in the bottom of the Program List window or press [F12 Config.] in the menu bar. The system message window appears as shown below. Select "All programs are active." to make all programs' compile flags active ("Enable" in the Use column). Press the OK button.



3) On the top screen, press [F1 Program]–[F6 Aux.]–[F12 Compile] to compile all the programs.

When the following system message appears, press the OK button to proceed.

Upon completion of compilation, the following system message appears. Press the OK button.



4) Restart the robot controller.

Notes on reading new *variables* data from USB flash memory

Reading new *variables* from the USB flash memory overwrites the current variables stored in the robot controller with the new ones.

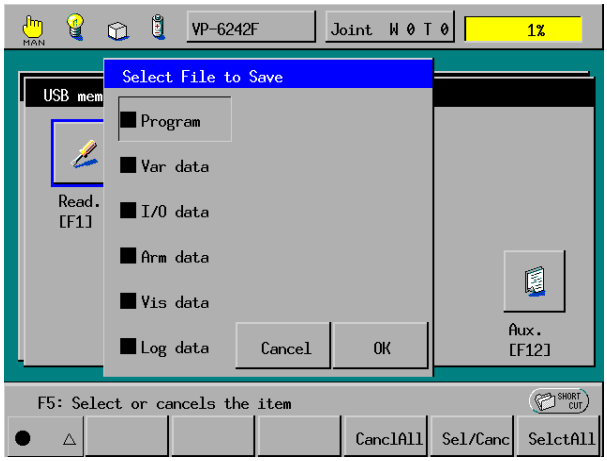
Note that, if the robot controller has 50 integer variables and the USB flash memory contains 30 integer variables, the 31st to 50th variables in the controller will be lost at the end of the read operation.

1.3 Writing Data Stored in Robot Controller into USB Flash Memory

Access: [F6 Set]—[F3 USB Memory]—[F2 Write.]

Writes (Saves) data stored in the robot controller into a USB flash memory.

(1) Press [F2 Write.] in the USB Memory Access Menu, and the Select File to Save window appears as shown below.



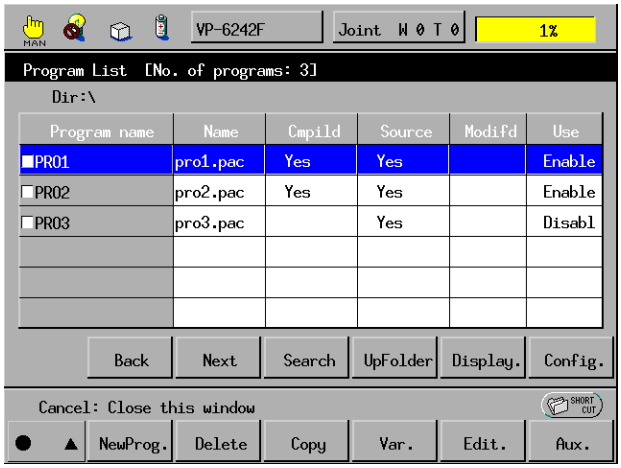
(2) Select the file to be written into the USB flash memory and press the OK button to start the write operation.

Listed below are probable causes of writing failures and actions to be taken

File type	Probable cause	Actions to be taken
Source	The loaded executable data does not match the PAC program.	If the executable data is more important, do not write a program into a USB flash memory. If you are maintaining the matching between the PAC program and the executable data, see "Relationship between executable data and program data" given two pages later.
Executable data	Executable data is missing.	Create executable data by compilation or send the executable data from WINCAPSII or USB flash memory to the controller.

Notes on writing *program* data into USB flash memory

When writing *program* data stored in the robot controller into the USB flash memory, the controller system writes only program source files whose compile flags are active ("Enable" in the Use column).



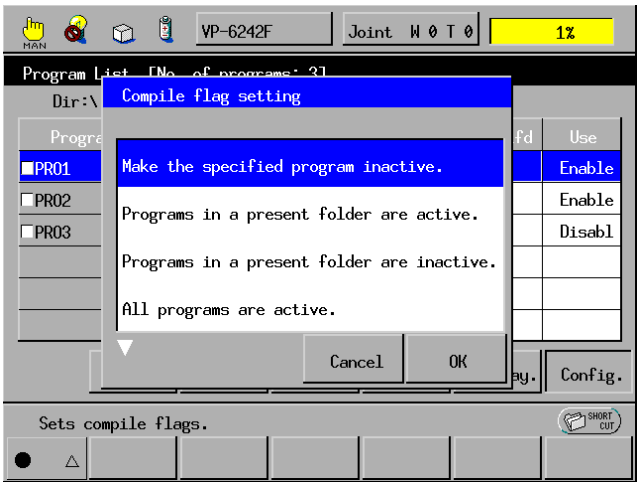
The compile flags of program source files become active ("Enable" in the Use column) in either of the following cases.

1) The project has been loaded in the robot controller. (Loading the project automatically makes the compile flags of all program source files in the project active.)
2) In the Program List window of the teach pendant, the user has made selected programs' compile flags active.

To write all program source files of the newly loaded project into a USB flash memory, do not make any file's compile flag inactive ("Disable" in the Use column) or modify any file after loading the project. For details, see "Relationship between executable data and program data," cases 4) through 7) given on the next page.

To write arbitrary program source files into a USB flash memory, make the compile flags of target files only active ("Enable") using the steps below.

1) On the top screen of the teach pendant, press [F1 Program] to display the Program List window.
2) Press the Config. button in the bottom of the Program List window or press [F12 Config.] in the menu bar. In the system message window shown below, confirm the settings and change them if necessary. Press the OK button.



Relationship between executable data and program data

If there is any discrepancy between the executable data (compiled objects) and the program data (program source data), the robot will not function as programmed.

Discrepancies may result from any of the following cases:

1) Although the PC has sent executable data (executable files and cross-reference table) to the robot controller, it has not sent the source program data (except the interpreter table and program table).
2) Although the PC has sent source program data to the robot controller, it has not sent executable data.
3) Although executable data has been read from the USB flash memory into the robot controller, the corresponding source program has not.
4) The source program has been edited after compilation.
5) A new project has been created.
6) A new program having the same name as one of the already loaded program files has been added.
7) Any program has been deleted.
8) Any file's compile flag has been changed from active to inactive or vice versa.

In the above cases except 1) and 3), you can resolve the discrepancy between the executable data and the source program data by compiling and loading.

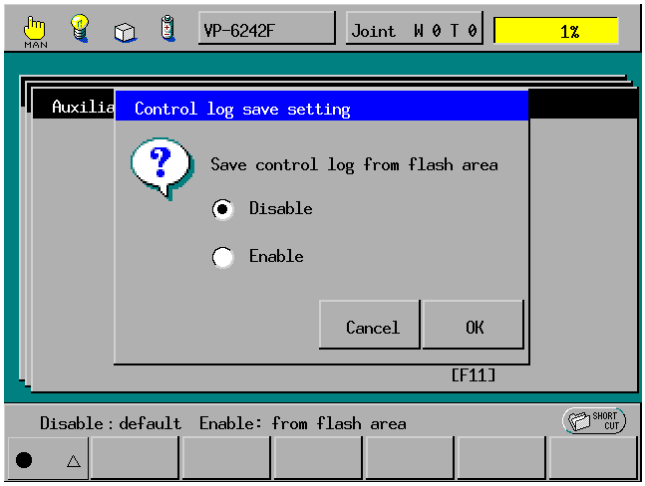
1.4 Saving Control Log into USB Flash Memory

Access: [F6 Set]—[F3 USB Memory]—[F12 Aux.]—[F11 CtrlLog.]

Determines whether or not to save the control log into a USB flash memory together with data stored in the robot controller.

The control log takes up a large storage space, while it is not required as backup of the facility data. Therefore, enable this feature only when you need to save the control log.

(1) In the USB Memory Access Menu, press [F12 Aux.]–[F11 CtrlLog.], and the Control Log Save Setting window appears as shown below.



(2) Select "Enable" or "Disable" and then press the OK button.

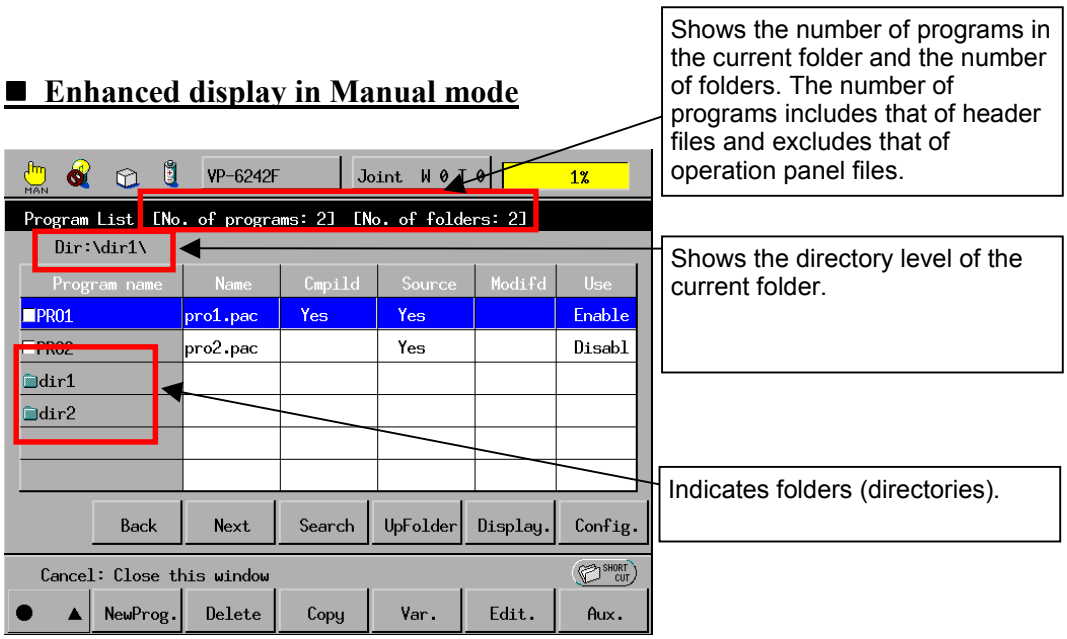
2 Folder Feature Newly Added

The folder feature newly added allows you to hierarchically manage user programs using a set of folders.

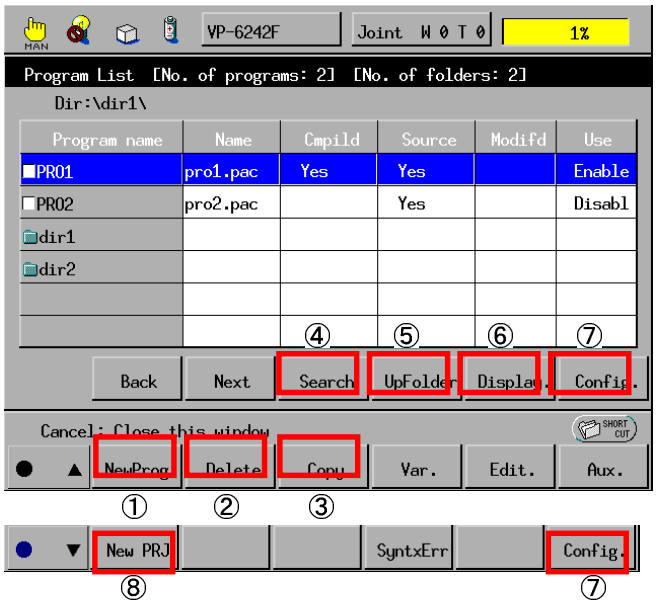
2.1 Enhancement to the Teach Pendant

To support the folder feature, the teach pendant is enhanced as described below.

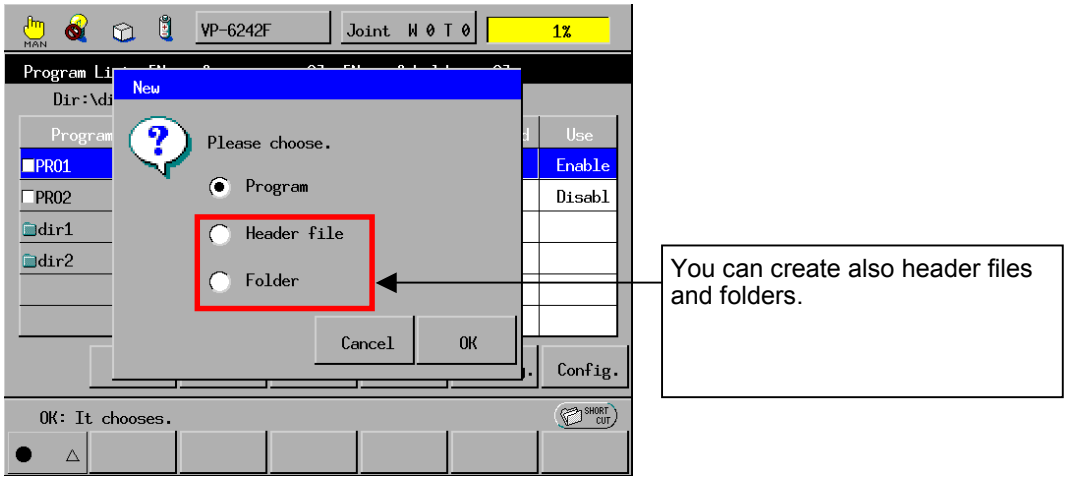
■ Enhanced display in Manual mode



■ Enhanced function keys in Manual mode



① [NewProg.]



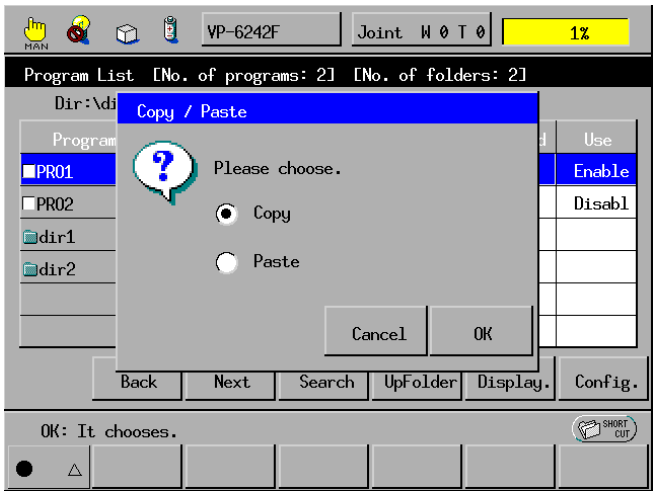
- Program: One controller can hold up to 256 programs created.
- Header file: The file name can be up to 32 characters long. One controller can hold up to 256 "header files plus operation panel files."
- Folder: The folder name can be up to 16 characters long. One controller can hold up to 256 folders created. Folders can be organized in a maximum of four levels excluding the root folder.

② [Delete]

This command can delete PAC programs, header files, folders, and operation panel files.

Deleting a folder erases all the objects inside the deleted folder.

③ [Copy]



- Copy: Selecting this option copies the selected PAC program, header file, or folder. The name of the object copied last appears in parenthesis following the Paste option button, meaning that the object is the one that will be pasted next.

  Copying a folder copies also all objects contained in that folder.

  No operation panel file can be copied except when it is contained in a folder. Copying such a folder can copy also operation panel files contained.

- Paste: Selecting this option pastes the object copied last into the current folder designated by "Dir:."

  If the current folder already contains an object having the same name as the object to copy, a "Paste under a new name or Overwrite" dialog appears, prompting the user's choice.

  Pasting PAC program(s) under a new name automatically makes its compile flag inactive ("Disable" in the Use column); overwriting it with a new one retains the compile flag status of the old one.

  Overwriting the destination folder with a source one

  In overwriting a folder, if the destination folder contains objects not contained in the source folder, those objects will remain intact.

④ [Search]

This command searches for a PAC program having the specified name in the current folder designated by "Dir:." It cannot search for header files, operation panel files, or folders.

The following sample bar code and scanner settings output data as listed below depending upon the substitution conditions.

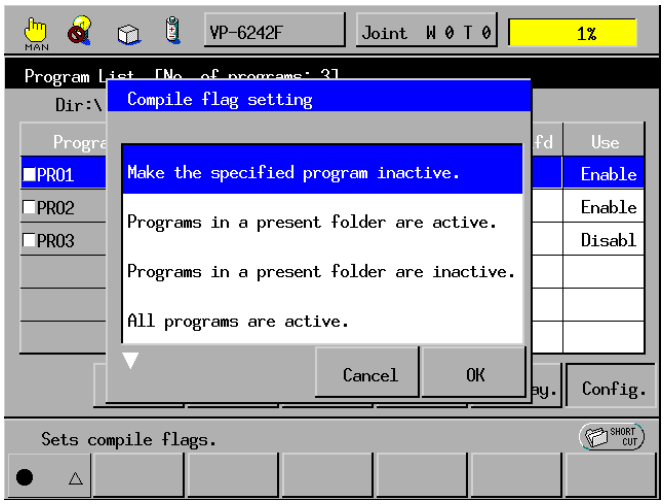
⑤ [UpFolder]

This command shifts to the upper folder from the current folder. If the current folder is already the root folder, nothing happens.

⑥ [Display.]

Pressing this command button with a PAC program or header file being selected displays its contents; pressing it with a folder being selected shifts the screen to the inside of that folder.

⑦ [Config.]



Pressing this command button toggles the compile flag on (active) and off (inactive). This applies to PAC programs only. Programs whose compile flags are active are subject to compilation.

The compile flag provides the following five choices:

- "Make the specified program active." or "Make the specified program inactive."
This appears only when the cursor points to a particular PAC program. Taking this choice reverses the current status of the selected PAC program.

- "Make all programs in the current folder active."
Selecting this makes all the PAC programs displayed in the current folder (folder designated by "Dir:") active.

- "Make all programs in the current folder inactive."
Selecting this makes all PAC programs displayed in the current folder designated by "Dir:" inactive.

- "Make all programs active."
Selecting this makes all PAC programs in the controller active.

- "Make all programs inactive."
Selecting this makes all PAC programs in the controller inactive.

⑧ [New PRJ]

This command erases all PAC programs, header files, folders, and operation panel files stemming from the root folder.

NOTE: Measuring run time of a program in a folder

Selecting [F1 Program]–[F6 Aux.]–[F1 Set PRJ.] displays the configuration list. The list contains the parameter "13: Delete cycle time calculation code" that provides the following three choices.

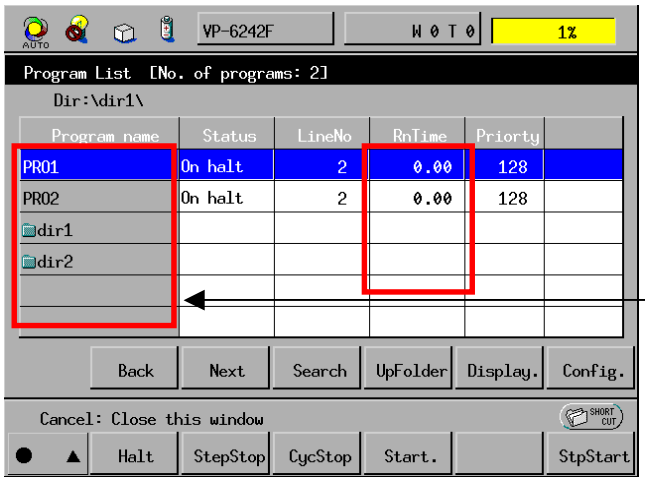
0: Measure the run time of all programs.

1: Measure the run time of programs that can be called through IO (that is, their roots are named "PRO*").

2: Do not measure the run time of any programs.

To display the run time of programs in the folder, set this parameter to "0."

■ Enhanced display in modes other than Manual mode



In modes other than Manual mode, only executable PAC programs and the folders containing such programs are displayed.

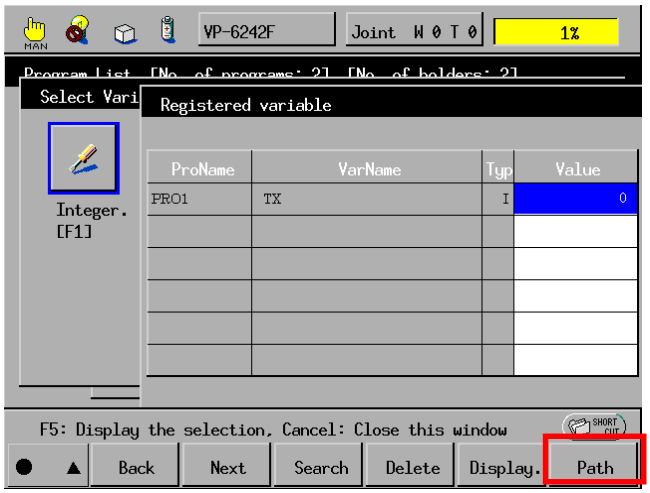
- The functions of the ④ [Search], ⑤ [UpFolder], ⑥ [Display.] are the same as in Manual mode (page 12). You can move objects from one folder to another in the same way as in Manual mode.

■ Handling local variables

The folder feature newly added enables programs or variables having the same name to coexist in a robot controller, as long as those programs or variables are located individually in different folders.

If located in different folders, variables having the same variable name and used in programs with the same name will be treated as different variables.

Register variable window

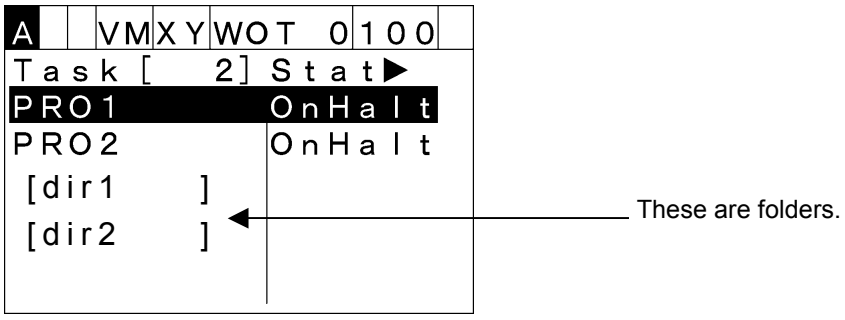


To locate a particular program, click the Path button.

2.2 Enhancement to the Mini-Pendant

The mini-pendant displays folders as shown below.

Key: [PRO]



Pressing [OK] calls up the "Functions" screen that lists the following.

SearchPRO: Search for a specified program name in the current folder.

Display: Show the property of the selected program. Pressing this with a folder being selected shifts the screen to the inside of that folder.

UpFolder: Shift to the upper folder from the current folder. If the current folder is already the root folder, nothing happens.

NowFolder: Show the current folder.

2.3 Staring Programs from I/O

An I/O can start only PAC programs located in the root. It cannot start programs in a folder even they have the name "PRO*."

2.4 Supervisory Tasks

The folder feature newly added enables supervisory tasks having the same name to coexist in a robot controller, as long as those tasks are located in individual folders.

The function and operation of supervisory tasks are the same as those in the previous system version, except that the number of supervisory tasks allowed is changed from 10 to 32 (TSR0 to TSR31).

In Version 2.2* or later, up to 32 supervisory tasks can run simultaneously independent of normal task programs. If more than 32 supervisory tasks are made active simultaneously, the error "7799: The maximum number of TSR was exceeded." occurs, terminating all programs including supervisory tasks.

2.5 Enhanced WINCAPSII

2.5.1 Functional Overview

The folder feature enables hierarchically managing of user programs with a directory structure.

You can organize your programs into a set of hierarchically structured folders by function or by type. This makes the configuration of programs easy to understand and allows you to handle programs on a folder basis, facilitating the use of existing programs for development of other programs.

This section outlines the enhancements to WINCAPSII with regard to the folder feature.

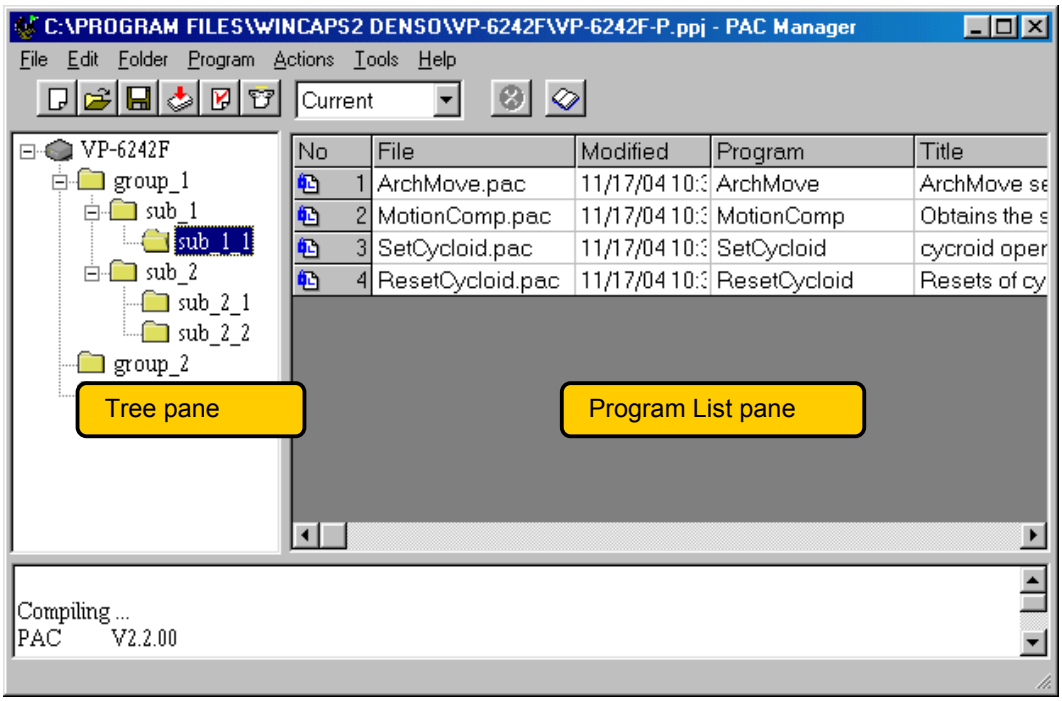
2.5.2 PAC Manager

2.5.2.1 Main window

A tree pane showing how the folders are organized is newly added in the left part of the PAC Manager screen.

The root of the tree bears the project name.

The Program List pane at the right lists programs contained in the selected folder.



[PAC Manager window]

2.5.2.2 Menus

This section outlines the menu items newly added or modified in conjunction with the introduction of the folder feature.

(1) File menu

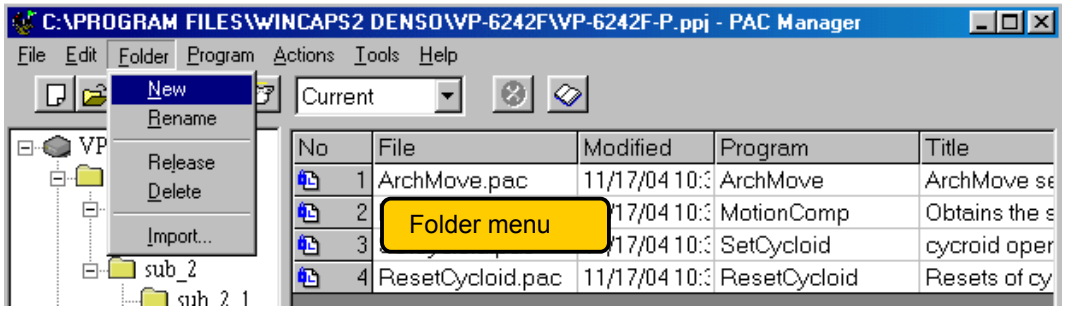
1) Creation of an executable program

If the project contains an operation panel file, you need to perform "Compilation of operation panel file" after completing the conventional "Creation of executable program." The compilation result appears in the message pane as in earlier versions.

(2) Folder menu (New feature)

The Folder menu is newly added as a special menu for handling the organization of folders.

Right-clicking in the tree pane also displays the same Folder menu.



Folder menu

1) New

This command creates a new folder under the selected folder.

Restrictions

There are the following restrictions on the creation of a new folder.

1. Folder name: This must start with an alphanumeric character. The length is a maximum of 16 characters.
2. Depth of levels: Up to 4, counting from the root of the project
3. Number of folders: Up to 256

2) Rename

This command changes the name of the selected folder.

3) Release

This command erases the selected folder from the program list together with its subordinate objects.

> **Note:** This command erases only data from the project registry. Files remain intact.

4) Delete

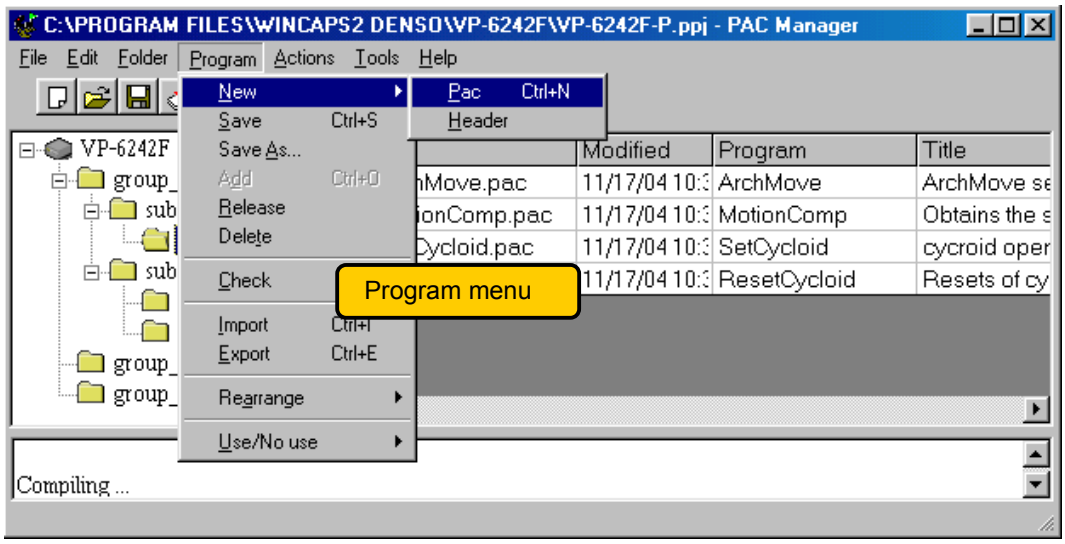
This command deletes the selected folder from the program list together with its subordinate objects. It also deletes files.

5) Import

This command brings (imports) a folder under the selected one together with its subordinate objects to add all of them into the program list.

> **Note:** If importing a folder causes the number of levels to exceed four counting from the project's root, the excess will be trimmed.

(3) Program menu



[Program menu]

1) New

This command creates not only a PAC program as in earlier versions but also a header file.

2) Add

This command takes effect only when the root folder is selected.

3) Delete (Newly added menu)

This command deletes the selected file from the program list and also deletes the file itself.

4) Import/Export

These commands can specify a header file and operation panel file.

5) Sort

This command sorts files by type.

The program list displays first PAC programs and then header files.

2.5.3 Variables Manager

2.5.3.1 FOLDER variables

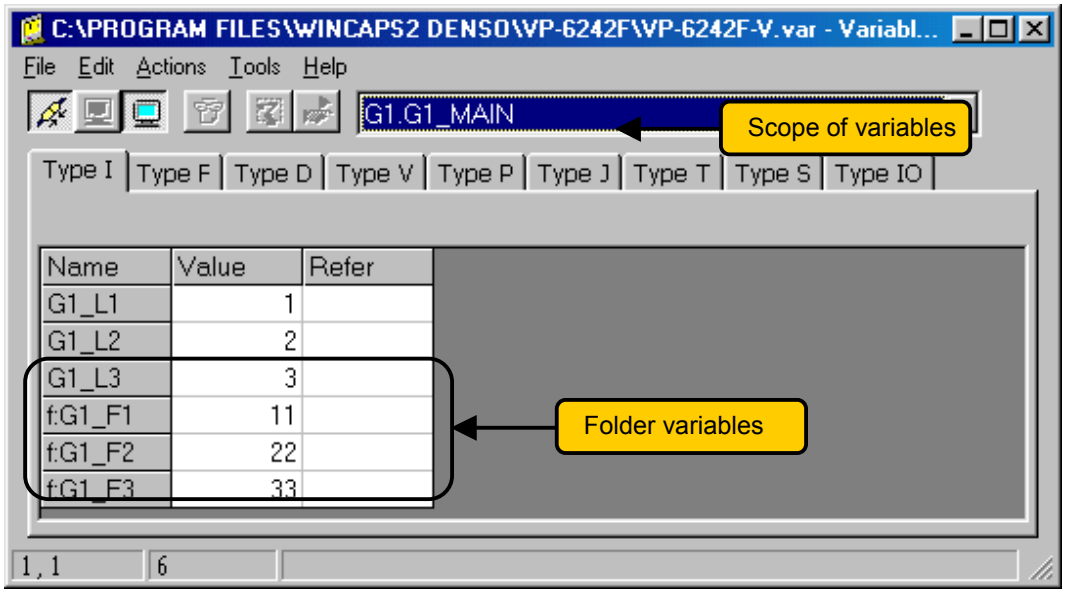
The Variables Manager monitors FOLDER variables in the same way as local variables.

Operating procedure

(1) Connect WINCAPSII with the robot controller.

(2) Select an arbitrary program using the scope of variables.

Note: The name of a FOLDER variable is preceded by [f:].



[Variables Manager window]

2.5.4 Other New Features

2.5.4.1 Display of program path

In conjunction with the introduction of the folder feature, program names are displayed in full path on the "Program Monitor," "Program Transfer" and other screens.

The program name thus takes the format: `foldername.programname`.

Note: The PAC Manager's program list window displays only program file names (without the path of folder names) as in earlier versions since it always displays only files in the selected folder.

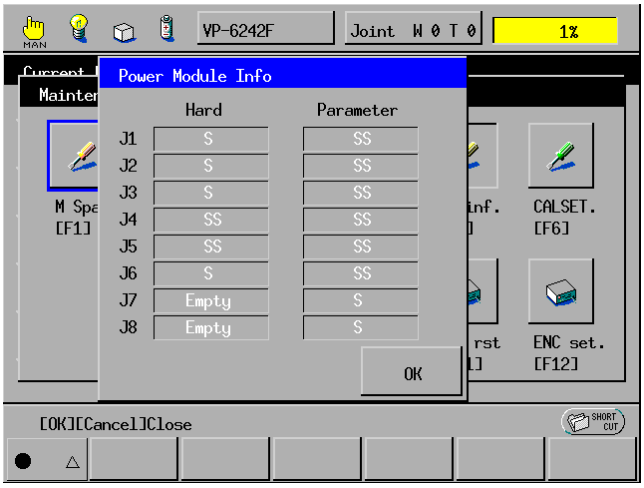
3 Power Module Information Display Newly Added

The display of the power module information is newly added.

Access: [F2 Arm]—[F12 Maint.]—[F5 PM Info.]

Displays the power module information.

(1) Press [F5 PM Info.] in the Maintenance Functions (Arm) window, and the Power Module Info window appears as shown below.



This window displays the capabilities of the power modules.

[Hardware] Shows any of the "SS," "S," "M," "L," "None," and "Unknown"

SS, S, M, or L: Represents the capacity of the power module.

None: Means that there is no power module.

Unknown: Means that the status of the power module cannot be recognized.

[Parameter] Shows any of the "SS," "S," "M," and "L"

SS, S, M, or L: Indicates the capacity setting of the power module in the controller.

The background of unavailable joints are dimmed.

Available joints whose hardware settings do not match the parameter settings will be *shown in red*. Any of such joints will cause a Jx power module capacity error (6149 for J1 through 6150 for J8).

(2) Press the OK or Cancel button to go back to the Maintenance Functions (Arm) window.

4 Supervisory Task Start Mode Newly Added

In system software versions earlier than 2.2, any of the following events activates *all* supervisory tasks in the robot controller.

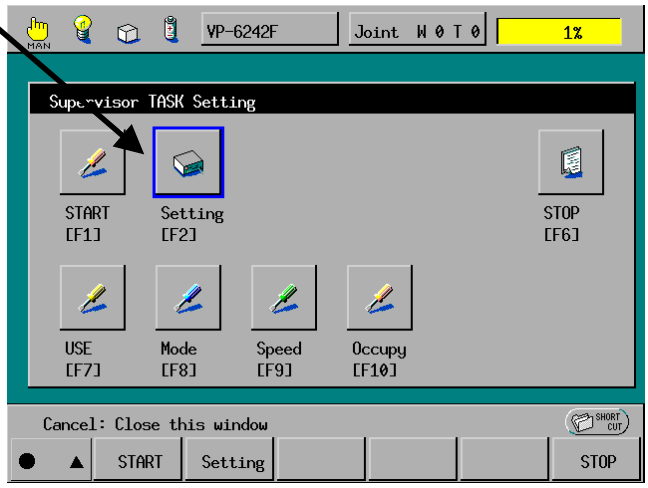
(1) Turning the controller power on

(2) Switching the operation mode from Manual to Auto mode

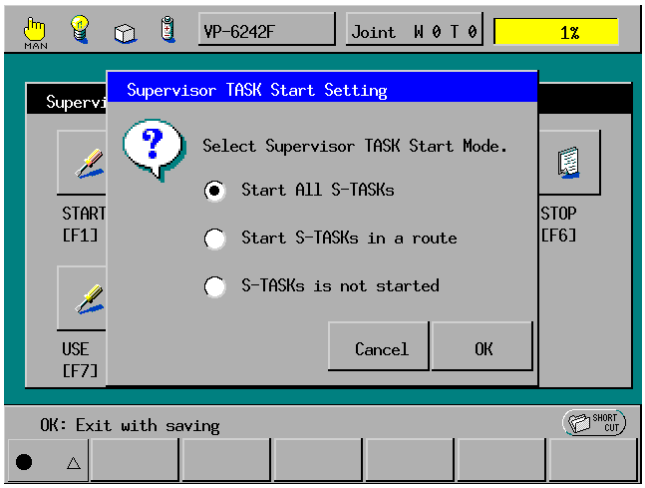
(3) Pressing [F8 S-TASK]–[F1 START]

The supervisory task start mode newly added in Ver. 2.2 or later provides the following three choices: Start all supervisory tasks, Start supervisory tasks in a root, and Do not start supervisory tasks.

Access: [F8 S-TASK]—[F2 Setting]



The Supervisory Task Start Mode window appears.



- Start All S-TASKs: Selecting this starts all supervisory tasks in the robot controller when any of the above three events occurs.
- Start S-TASKs in the root: Selecting this starts only supervisory tasks in the current route when any of the above three events occurs.
- Do not start S-TASKs: Selecting this starts no supervisory tasks when any of the above three events occurs.

Note: Independent of the current setting of the supervisory task start mode, supervisory tasks can start by selecting them on the Program List or by using a RUN command.

5 Commands Newly Added or Modified

This section outlines the commands that are newly added or modified in Ver. 2.2 or later.

5.1 Commands Associated with Folder Feature

5.1.1 Declaration of FOLDER Variable

FOLDER (Statement)

Function Declares local variables that are accessible from external programs.

Format

```
FOLDER DEFINT <variablename>[=<Constant>][,<variablename>[=<constant>],...]
FOLDER DEFSNG <variablename>[=<Constant>][,<variablename>[=<constant>],...]
FOLDER DEFDBL <variablename>[=<Constant>][,<variablename>[=<constant>],...]
FOLDER DEFSTR <variablename>[=<Constant>][,<variablename>[=<constant>],...]
FOLDER DEFVEC <variablename>[=<vector constant>][,<variablename>[=<vector constant>],...]
FOLDER DEFPOS <variablename>[=<position constant>][,<variablename>[=<position constant>],...]
FOLDER DEFJNT <variablename>[=<position constant>][,<variablename>[=<position constant>],...]
FOLDER DEFTRN <variablename>[=<trans constant>][,<variablename> [=<trans constant>],...]
FOLDER DIM <variablename><suffix>[(<element count>[,<element count>[,<element count>]])]
[AS<variabletype>][,<variablename>,...]
```

Description Declaring a local variable with this statement can access the variable from an external program using an EXTERN statement.

(1) Two or more local variables having the same name cannot be defined in a program.

List 1 causes a double declaration error since three declaration statements define AA in the same program.

```
PRO1
 FOLDER DEFINT AA
 FOLDER DEFSNG AA
 DEFINT AA
END
```
List 1

(2) FOLDER variables having the same name cannot be defined in a folder.

```
PRO1
 FOLDER DEFINT AA
END

PRO2
 FOLDER DEFSNG AA
END
```
List 2

```
PRO1
 FOLDER DEFINT AA
END
```

```
PRO2
 FOLDER DEFINT AA
END
```
List 3

List 2 causes an error since FOLDER variables AA are defined in two programs located in the same folder.

List 3 is correct in defining FOLDER variables AA. These variables have the same name but are defined in different folders so that they are treated as different ones.

(3) The same variable name can be declared for both a FOLDER variable and local variable if those variable are located in different programs.

In List 4, AA in PRO1 and AA in PRO2 are treated as different variables.

```
PRO1
 FOLDER DEFINT AA
END

PRO2
 DEFINT AA
END
```

List 4

Editing a program from the teach pendant or sending <Map/executable program> to the robot controller in WINCAPSII clears the FOLDER variables.

Related item

EXTERN

Coding example

(1) When three programs (PRO1 through PRO3) exist in the same folder:

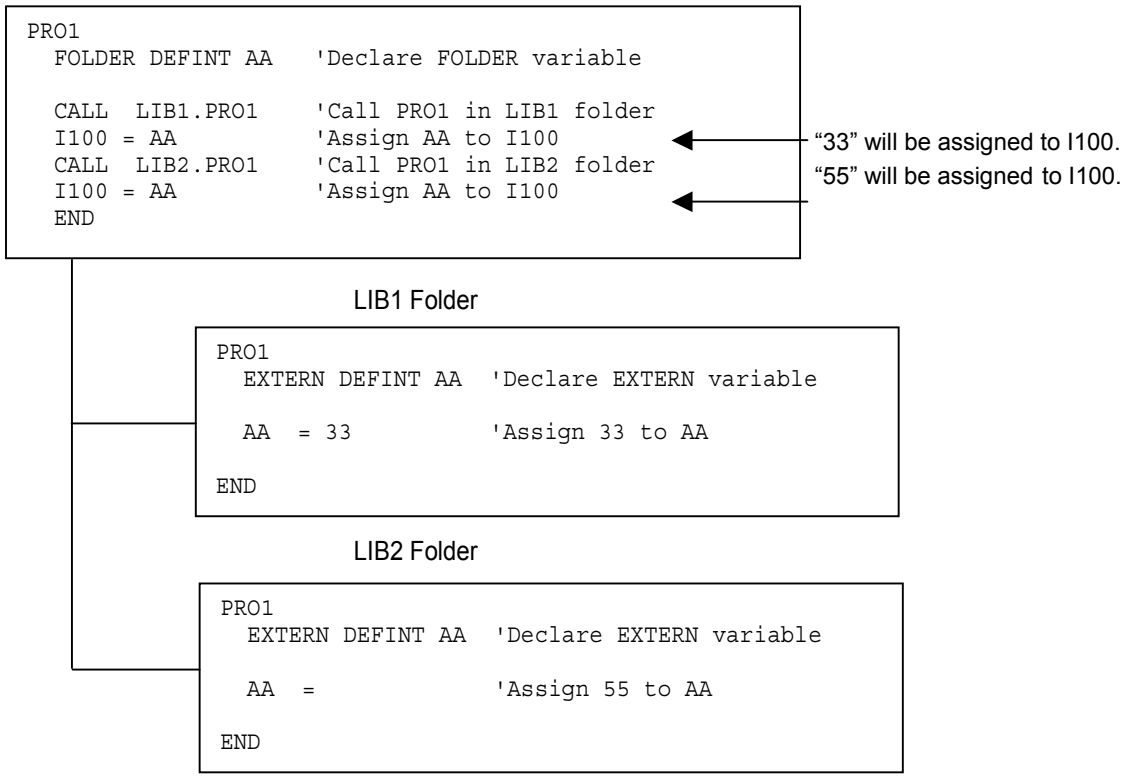
Execution of PRO3 after that of PRO1 terminates the program (STOP).

Execution of PRO3 after that of PRO2 temporarily halts the program (HOLD "STOP").

Variable AA declared in the three programs will have the same value.

```
PRO1
 FOLDER DEFINT AA        'Declare FOLDER variable

 AA =1                   'Assign 1 to variable AA
END

PRO2
 EXTERN DEFINT  AA       'Declare EXTERN variable

 AA=2                    'Assign 2 to variable AA
END

PRO3
 EXTERN DEFINT  AA       'Declare EXTERN variable

 SELECT  CASE  AA        'If value of AA matches the CASE number, the
                         'corresponding command will be executed.
 CASE  1                 'If AA is 1, the program execution will
     STOP                'terminate.
 CASE  2                 'If AA is 2, the program execution will be held
     HOLD   "STOP"       'temporarily.

 END SELECT              'Declare the end of the CASE statement
END
```

(2) When the programs exist in the folder structure shown below:



(3) When an initial value is declared by a FOLDER statement:

1) Execution of PRO2 after that of PRO1 assigns 3 to both I1 and I2.

2) Execution of PRO1 after that of PRO2 assigns 3 to I1 and 0 to I2.

Note: Without execution of PRO1, the initial value will not be assigned to any variable.

```
PRO1
  FOLDER DEFINT AA =3
  I1 = AA
END

PRO2
  EXTERN DEFINT AA
  I2 = AA
END
```

5.1.2 Declaration of EXTERN Variable

EXTERN (Statement)

Function

Declares access to a FOLDER variable defined in another program.

Format

```
EXTERN DEFINT <variablename>[,<variablename>,...]
EXTERN DEFSNG <variablename>[,<variablename>,...]
EXTERN DEFDBL <variablename>[,<variablename>,...]
EXTERN DEFSTR <variablename>[,<variablename>,...]
EXTERN DEFVEC <variablename>[,<variablename>,...]
EXTERN DEFPOS <variablename>[,<variablename>,...]
EXTERN DEFJNT <variablename>[,<variablename>,...]
EXTERN DEFTRN <variablename>[,<variablename>,...]
EXTERN DIM <variablename><suffix>[(<element count>[,<element count>[,<element count>]])]
[AS<variabletype>][, <variablename>,...]
```

Description

Searching for a FOLDER variable

(1) This statement first searches for a FOLDER variable in a folder where the PAC program containing EXTERN is located.

If the FOLDER variable is found, the search terminates.

If it is not found, the search proceeds to step (2).

(2) The search for a FOLDER variable continues with the folder just above the previous level.

If the FOLDER variable is found, the search terminates.

If it is not found, step (2) is repeated.

If it cannot be found even after reaching the top folder, then a compilation error will result.

```
PRO1
  EXTERN DEFINT AA
  DEFINT AA
END
```
List 1

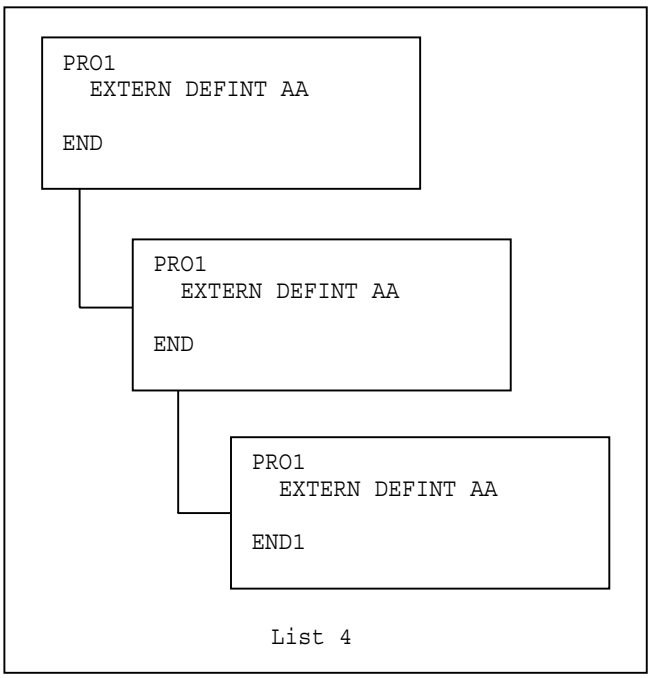
List 1 causes a compilation error since the same variables AA are declared in a PAC program.

```
PRO1
  EXTERN DEFINT AA
  EXTERN DEFINT AA
END
```
List 2

List 2 causes a compilation error since the same EXTERN variables are declared in a PAC program.

```
PRO1
  EXTERN DEFINT AA
END

PRO2
  EXTERN DEFINT AA
END
```
List 3

List 3 is correct in declaring EXTERN variables AA. These variables have the same name but are declared in different folders so that they are treated as different ones.



List 4

List 4 is correct in declaring EXTERN variables AA. These variables have the same name but are declared in different folders.

Editing a program from the teach pendant or sending <Map/executable program> to the robot controller in WINCAPSII clears the EXTERN variables.

Related item

FOLDER

Coding example

(1) When three programs (PRO1 through PRO3) exist in the same folder:

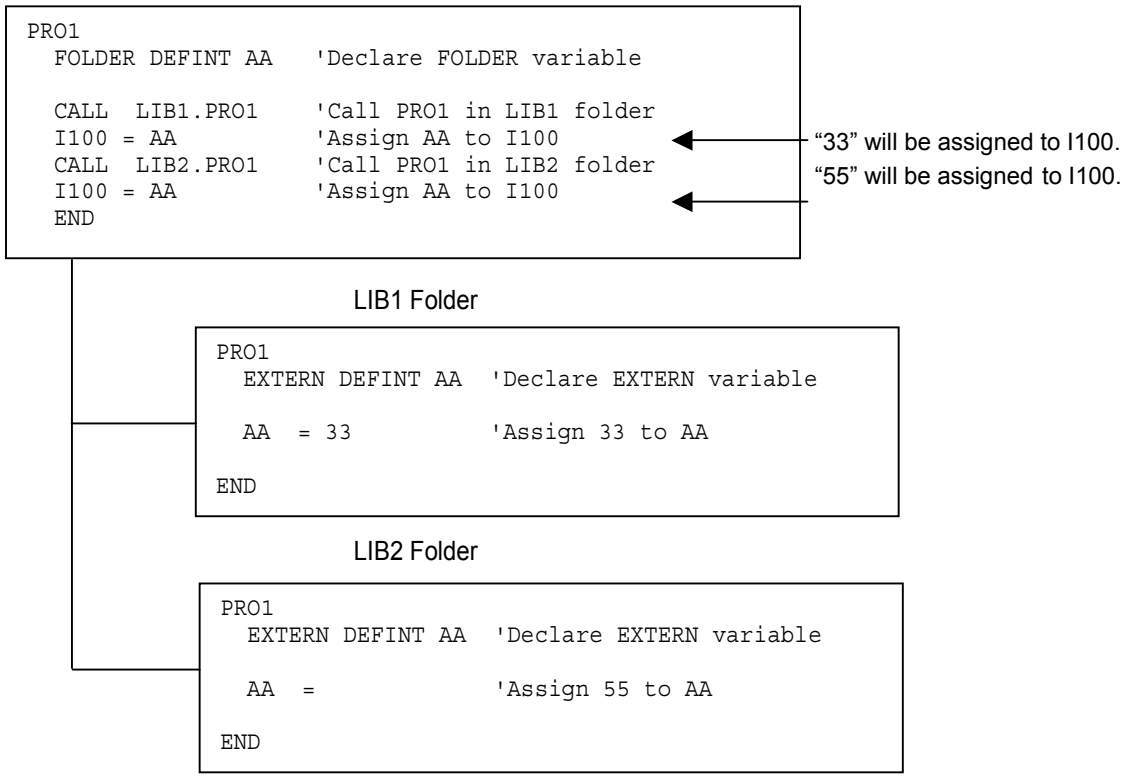
Execution of PRO3 after that of PRO1 terminates the program (STOP).

Execution of PRO3 after that of PRO2 temporarily halts the program (HOLD "STOP").

Variable AA declared in the three programs will have the same value.

```
PRO1
 FOLDER DEFINT AA        'Declare FOLDER variable

 AA =1                   'Assign 1 to variable AA
END

PRO2
 EXTERN DEFINT  AA       'Declare EXTERN variable

 AA=2                    'Assign 2 to variable AA
END

PRO3
 EXTERN DEFINT  AA       'Declare EXTERN variable

 SELECT  CASE  AA        'If value of AA matches the CASE number, the
                         'corresponding command will be executed.
 CASE  1                 'If AA is 1, the program execution will
     STOP                'terminate.
 CASE  2                 'If AA is 2, the program execution will be held
     HOLD   "STOP"       'temporarily.

 END SELECT              'Declare the end of the CASE statement
END
```

(2) When the programs exist in the folder structure shown below:



(3) When an initial value is declared by a FOLDER statement:

1) Execution of PRO2 after that of PRO1 assigns 3 to both I1 and I2.

2) Execution of PRO1 after that of PRO2 assigns 3 to I1 and 0 to I2.

Note: Without execution of PRO1, the initial value will not be assigned to any variable.

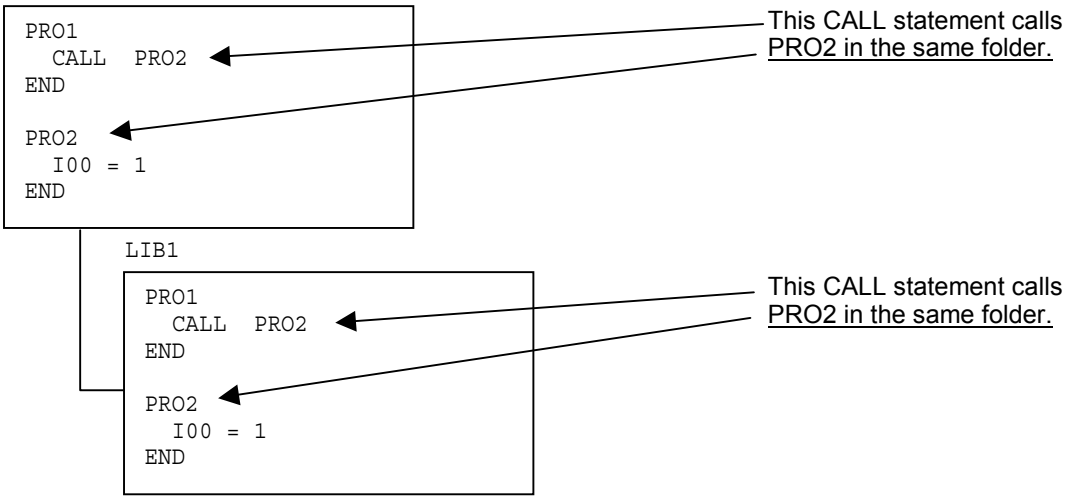
```
PRO1
  FOLDER DEFINT AA =3
  I1 = AA
END

PRO2
  EXTERN DEFINT AA
  I2 = AA
END
```

5.1.3 Calling a Program through Folder Structure

[1] CALL and RUN statements

(1) Specifying a target program name only

Calling a program using a target program name only calls the specified program in the folder where the current calling program is located.



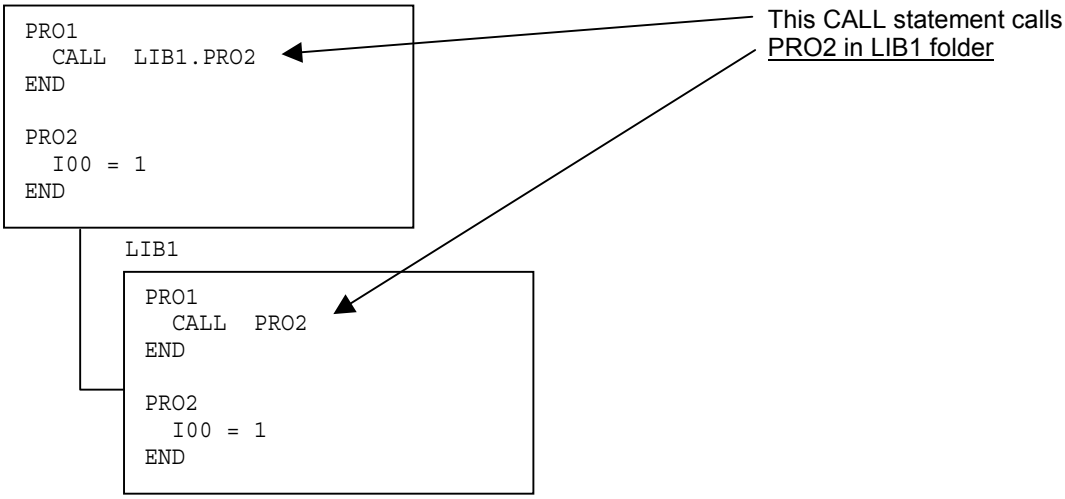
(2) Specifying a target folder name and program name

Calling a program using a target folder name and program name calls the specified program in the specified folder.

The syntax is "CALL (folder name).(program name)."

The target folder can be just one level below the current folder.

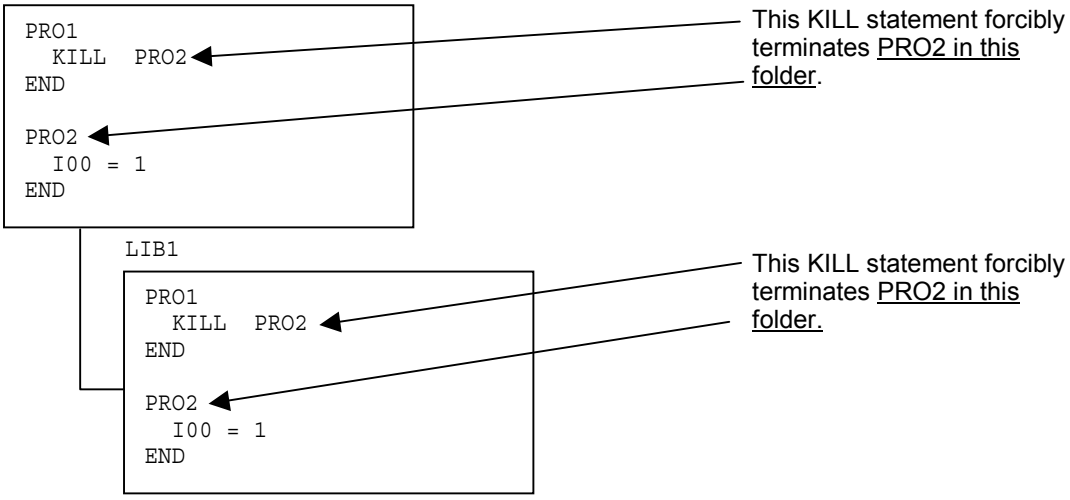
The "CALL (folder name).(folder name).(program name)" will result in a compilation error.



[2] KILL, SUSPEND, and STATUS statements

(1) Specifying a target program name only

The KILL, SUSPEND, or STATUS statement processes the specified program in the folder where the current program is located.

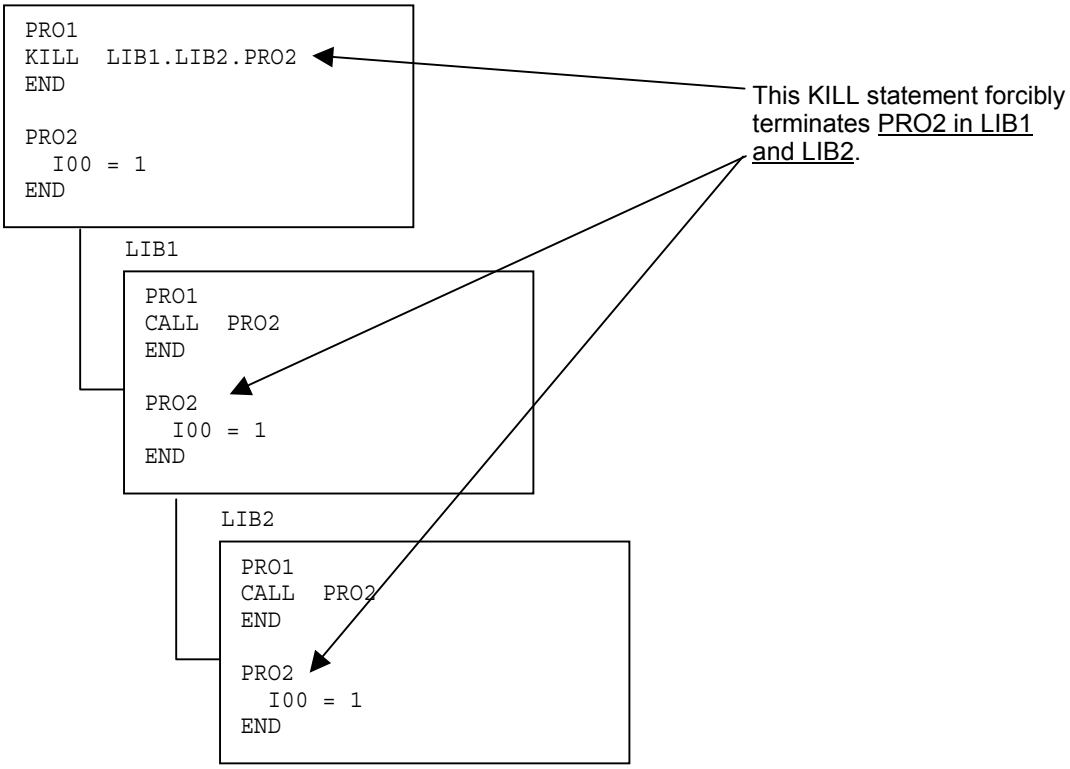


(2) Specifying a target folder name and program name

The KILL, SUSPEND, or STATUS statement processes the specified program in the specified folder.

The syntax is "KILL (folder name).(program name)."

Two or more folders can be specified as shown below.
Example: KILL (folder name).(folder name).(program name)



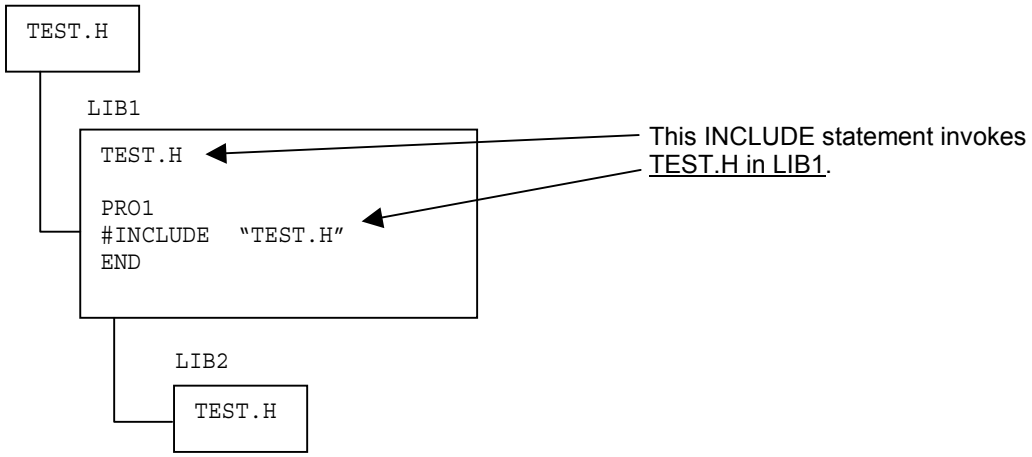
5.1.4 INCLUDE Preprocessor Statement

(1) #INCLUDE <filename>

Searches for the specified file in the system folder.

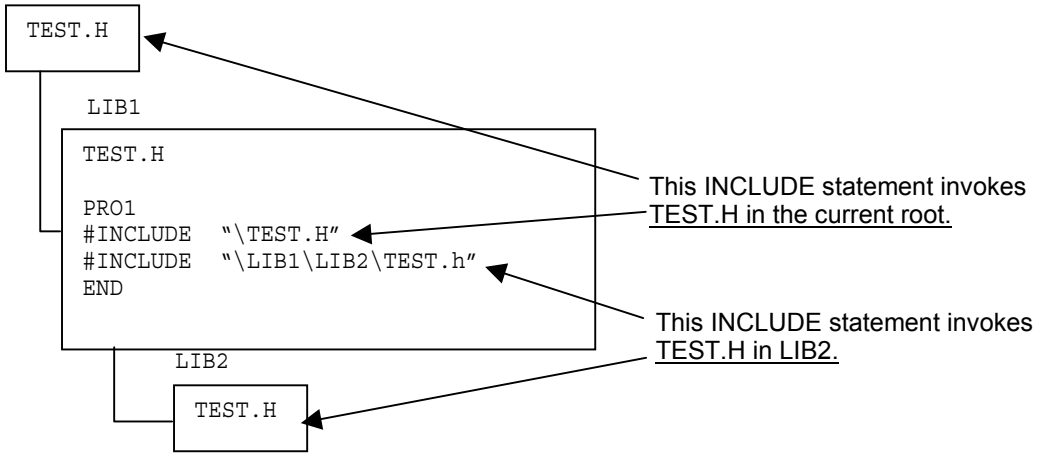
(2) #INCLUDE "filename"

Searches for the specified file in the folder where the current program is located.



(3) #INCLUDE "\(folderpath)\filename"

Searches for the specified file in the folder located at the end of the folder path starting from the root.

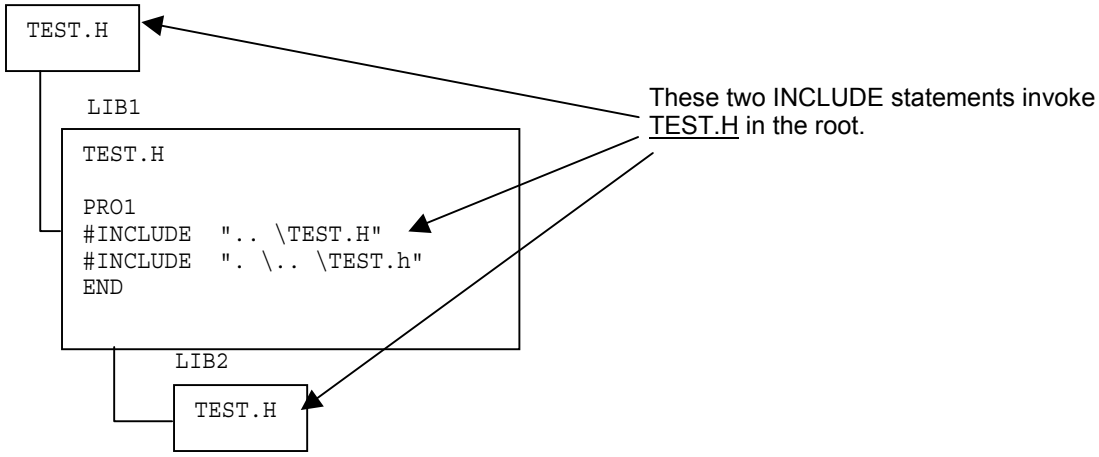


(4) #INCLUDE "..\filename" or #INCLUDE ".\filename."

.. \: Searches for the specified file in the folder just above the current level.

. \: Searches for the specified file in the folder where the current program is located.

Example) #INCLUDE ". \.. \TEST.H"
Searches for TEST.H in the folder just above the current level.



5.2 Other Commands Added

GETLANGUAGE (Function)

Function Gets the current language setting.

Syntax `GETLANGUAGE`

Description Gets the current language setting. 0: Japanese 1: English

Coding example

```
defint 1i
1i=GETLANGUAGE
```

PrintWarning (Statement)

Function Displays a message in the alarm message area on the teach pendant.

Syntax `PRINTWARNING <message text>[,<message color>]`

Description Displays a message in the alarm message area on the teach pendant.

This statement does not record the message in the error log.

`<message text>` Up to 55 characters can be specified. Specifying more than 55 characters discards the excess.

`<message color>` 0: black 1: blue 2: green 3: red Others: black
Specifying nothing to the message color uses black by default.

This feature is available only for the teach pendant.

Coding example

```
printWarning "Hello World!",1
```

5.3 Commands Modified

This section outlines the commands that are modified in Ver. 2.2.

Refer to the PROGRAMMER'S MANUAL I, Chapter 13, "Input/Output Control Statements."

INPUT (Statement) [SLIM compliant]

Function Gets data from an RS-232C or Ethernet port.

Syntax

```
INPUT [#<line number>,]<variablename>[,WTIME=<timeout>
[,RVAL=<restore variable>]][<variablename>[,WTIME=<timeout>
[,RVAL=<restore variable>]]...]
```

Description Stores the data that has been received at the RS232C or Ethernet port into the variable specified by <variablename>.

In <line number>, specify the line number of the RS232C or Ethernet port being used. If you omit the <line number> assignment, the default value of "ch2" will be assumed. You cannot specify "ch1" since it is reserved for a teach pendant. (See 2.4.1 Line number.)

To store more than one piece of data into the same number of variables, insert a comma (,) between each adjacent pair of the data.

All the information preceding a delimiter (CR or CR + LF) will be stored into the specified variable. The delimiter itself will not be stored into the specified variable.

The baud rate needs to be specified using a system parameter.

<timeout> Ver. 2.2 or later
You can specify a timeout period for each variable. If no input data is found for any variable, the system will wait for the timeout period specified for this variable, and after that, it will pass control to the next command. The timeout period should be specified in ms (milliseconds). In practice, however, the actual wait time is incremented every 1/60 of a second.

<restore variable> Ver. 2.2 or later
This parameter is used in conjunction with <timeout> . If the processing of a variable is normally completed by data received, then TRUE (1) will be stored into the specified restore variable; if it terminates as a result of a timeout, FALSE (0) will be stored.

Notes: (1) Be sure to issue a FLUSH command before receiving data in order to clear the input buffer for incoming data.

(2) If the received data exceeds the maximum value that can be expressed by the variable type of the target variable, then this maximum value will be stored into the variable.

Related item FLUSH, PRINT and WRITE

Coding example

```
DIM 1i1 As Integer
defint 1i
DEFSTR 1s1, 1s2, 1s3, 1s4
INPUT #1, 1s1                  'Write data from ch2 into 1s1.
INPUT #1, 1i1, 1s2, 1s3        'Write data from ch2 into '1i1, 1s2, and 1s3.
INPUT 1s4                      'Write data from ch2 into 1s4.
INPUT 1s4, WTIME=100, RVAL=1i  'If there is no input data, transfer control
                               'to the next command after the timeout of
                               '100 ms. Store "0" into 1i, indicating that
                               'the processing has terminated because of
                               'timeout.
```

LINEINPUT (Statement)

Function Reads data received at an RS232C or Ethernet port preceding a delimiter, and stores (assigns) it to a character string type variable.

Syntax

```
LINEINPUT [#<line number>,]<string variablename>[,
WTIME=<timeout>[,RVAL=<restore variable>]]
```

Description Stores all the characters preceding a delimiter (CR or CR + LF) in the data which has been received at the RS232C or Ethernet port into the variable specified by <character string type variable name>. The delimiter itself will not be stored into the specified variable.

In < line number>, specify the line number of the RS232C or Ethernet port being used. If you omit the < line number> assignment, the default value of "ch2" will be assumed. You cannot specify "ch1" since it is reserved for a teach pendant. (see 2.4.1 Line number.)

In <string variablename>, specify the name of a character string type variable.

<timeout> Ver. 2.2 or later
If no delimiter is received within the specified period of time, the processing of the LINEINPUT statement will terminate and control will be transferred to the next command. The timeout period should be specified in ms (milliseconds). In practice, however, the actual wait time is incremented every 1/60 of a second.

<restore variable> Ver. 2.2 or later
This parameter is used in conjunction with <timeout>. If the processing of a variable is normally completed by data received, then TRUE (1) will be stored into the specified restore variable; if it terminates as a result of a timeout, FALSE (0) will be stored.

Coding example

```
DEFSTR 1s1,1s2,1s3,1s4
DEFINT 1i
LINEINPUT #0, 1s1      'Receive character string from port 1 and assign it to 1s1.
LINEINPUT #1, 1s2      'Receive character string from port 2 and assign it to 1S2.
LINEINPUT 1s3          'Receive character string from port 2 and assign it to 1S3.
LINEINPUT#1, 1s4,WTIME=100,RVAL=1i
                       'If no data exists preceding a delimiter, transfer control
                       'to the next command after the timeout of 100 ms.
                       'Set "0" into 1i, indicating that the processing
                       'has terminated because of timeout.
```

inputb

Function

Inputs one byte of data through an RS232C or Ethernet port.

Syntax

```
inputb[#<portnumber>,]<integervarnumber>[,WTIME=<timeout>
[,RVAL=<restore variable>]
```

<portnumber> input port number
(1: controller's RS232C port, -1: μVision board's RS232C port,
4-7: Ethernet server ports, 8-15: Ethernet client ports)
Default: 1 (controller's RS232C port)

<integervarnumber>
Variable number of the integer variable into which the input data is to be stored.

<timeout> Ver. 2.2 or later
If no input data is found for any variable, the system will wait for the timeout period specified for this variable, and after that, it will terminate the processing of this inputb statement and will pass control to the next command. The timeout period should be specified in ms (milliseconds). In practice, however, the actual wait time is incremented every 1/60 of a second.

<restore variable> Ver. 2.2 or later
This parameter is used in conjunction with <timeout>. If the processing of a variable is normally completed by data received, then TRUE (1) will be stored into the specified restore variable; if it terminates as a result of a timeout, FALSE (0) will be stored.

Description

Stores the data input through a specified port into the integer type variable specified by <integervarnumber>.

Note: If you have not specified a timeout length for this command, the absence of data at the specified port would result in an endless wait. To check whether data actually exists, issue a com_state command.

Coding example

```
'!TITLE "<title>"
PROGRAM sample
        defint 1i
        .
        .
        .
        inputb #1,I10       'Store data received through RS232C port
                            'in I10.
        inputb #1,I10,WTIME=100,RVAL=1i
                            'If there is no input data, transfer control to
                            'the next command after timeout of 100 ms.
                            'Store "0" into 1i, indicating that the
                            'processing has terminated because of timeout.
        .
        .
        .
end
```

linputb [Ver. 1.5 and later]

Function

Inputs more than one byte of data through an RS232C or Ethernet port.

Syntax

```
linputb[#<portnumber>,]<arrayheadelement>,<inputbytes>[,
WTIME=<timeout>[,RVAL=<restore variable>]]
```

<portnumber> input port number
(1: controller's RS232C port, -1: μVision board's RS232C port
4-7: Ethernet server ports, 8-15: Ethernet client ports)
Default: 1 (controller's RS232C port)

<arrayheadelement>
First element of the array into which the input data is to be stored.

<inputbytes>
Number of bytes in the input data

<timeout> Ver. 2.2 or later
If the actual input data for the variable is shorter than the specified number of bytes, the system will wait for the specified timeout period, and after that, it will terminate the processing of this linputb statement and will pass control to the next command. The timeout period should be specified in ms (milliseconds). In practice, however, the actual wait time is incremented every 1/60 of a second.

<restore variable> Ver. 2.2 or later
This parameter is used in conjunction with <timeout>. If the processing of a variable is normally completed by data received, then TRUE (1) will be stored into the specified restore variable; if it terminates as a result of a timeout, FALSE (0) will be stored.

Description

Inputs, through the specified port, the specified number of bytes into the destination array starting at the specified array element.

Note: If you have not specified a timeout length for this command, a shortage of data at the specified port (the actual data length is shorter than the specified number of input bytes) would result in an endless wait. To check whether data actually exists, issue a com_state command.

Coding example

```
'!TITLE "<title>"
PROGRAM sample
         defint 1i
         .
         .
         linputb #1,I64,30  'Input data into I64 to I93 from RS232C port.
linput #1, I64, 30,WTIME=100,RVAL=1i
                            'If the actual length of data is shorter than the
                            'specified number of input bytes, wait for
                            '100 ms, after that, terminate processing
                            'of this linputb and pass control to the
                            'next command. Store "0" into l1, indicating
                            'that the processing has terminated because of
                            'timeout.
         .
         .
end
```

6 Error Code Tables Modified

Refer to the ERROR CODE TABLES.

6.1 Error Codes Modified

The table below lists the error codes that are modified in Ver. 2.2 or later.

Code	Message	Level	Description	Remedy
1201	Prepare for communication (unconnected)	4	The DeviceNet module is operating normally, but I/O connection is not yet established, although the explicit connection with the master device is established.	Establish a connection from the master device. If the network connection can be established even after this error that occurred during a power-on sequence, then increase the wait time for network error detection.
1203	Prepare for communication (idle status)	4	The DeviceNet module is operating normally, but only empty data has been received from the master device within the specified time period.	Check the contents of the I/O data sent from the master device. If the network connection can be established even after this error that occurred during a power-on sequence, then increase the wait time for network error detection.
1204	Prepare for communication (I/O timeout).	4	The DeviceNet module is operating normally, but no data has been received from the master device within the specified time period.	Check and ensure that (a) there is no broken wire in the network cable, (b) the connectors are properly inserted, (c) the cable length is within the limit, and (d) the termination resistor is properly installed at the right position. If the network connection can be established even after this error that occurred during a power-on sequence, then increase the wait time for network error detection.
1205	Robot access failure in DPRAM	4	The robot cannot access the DPRAM on the DeviceNet or CC-Link board.	Power the controller off and on for restart.
1246	MACID overlaps	4	The node address of this node is used by another node.	Change the node address of either node.
1249	CAN transmission timeout	4	Transmission to the CAN chip failed on the DeviceNet master.	Locate and solve the problem occurring on the network. This error can happen when there is no other node in the network and the network is powered on.
124A	DeviceNet RAM failure	4	The DeviceNet's communications software detected a hardware error of RAM.	Power the controller off and on for restart.

Code	Message	Level	Description	Remedy
124D	DeviceNet board access failure in DPRAM	4	The DeviceNet's communications software cannot access DPRAM.	Power the controller off and on for restart.
21BA	Interference check execution error	4	An error was detected during the execution of an interference check.	Check the interference check setting. Ensure also that the IO port number of the destination is set to general output or internal IO.
27AA	Program reset signal ON.	2	The program reset signal is on.	Turn off the program reset signal and then start again. A program reset signal being on during Continue Start triggers this error. Before Continue Start, turn off the program reset signal.
6149	J1 power module capacity failure	4	The J1 power module does not match the motor parameter values.	1. Check the matching between the power module capacity and the connected motor's output. 2. Check the matching between the arm file being used and the robot model. 3. When the joint is used as an extended-joint, check that the extended-joint path parameters for the motor are correctly set.
614A	J2 power module capacity failure	4	The J2 power module does not match the motor parameter values.	Same as above
614B	J3 power module capacity failure	4	The J3 power module does not match the motor parameter values.	Same as above
614C	J4 power module capacity failure	4	The J4 power module does not match the motor parameter values.	Same as above
614D	J5 power module capacity failure	4	The J5 power module does not match the motor parameter values.	Same as above
614E	J6 power module capacity failure	4	The J6 power module does not match the motor parameter values.	Same as above
614F	J7 power module capacity failure	4	The J7 power module does not match the motor parameter values.	Same as above
6150	J8 power module capacity failure	4	The J8 power module does not match the motor parameter values.	Same as above

6.2 Error Codes Added

The table below lists the error codes added in Ver. 2.2 or later.

Code	Message	Level	Description	Remedy
1218	FlashRom BCC error	4	A BCC error has occurred in the flash ROM on the DeviceNet board.	Power the controller off and on for restart.
1219	Parameter information area error	4	A data error has occurred in the parameter information area on the DeviceNet board.	Power the controller off and on for restart.
121A	Robot control area error	4	A data error has occurred in the Robot Controller's control area on the DeviceNet board.	Power the controller off and on for restart..
121D	Scan list data table error	4	A data error has occurred in the scan list data table on the DeviceNet board.	Power the controller off and on for restart.
121E	Scan list mapping information area error	4	A data error has occurred in the scan list mapping information area on the DeviceNet board.	Power the controller off and on for restart.
129D	CC-Link check sum error	5	A check-sum error has occurred in the flash ROM on the CC-Link Remote Device board.	Power the controller off and on for restart.
129E	CC-Link board access failure in DPRAM	4	The robot cannot access the DPRAM on the CC-Link Remote Device board.	Power the controller off and on for restart.
2035	Data area remains undefined	2	The data area is not yet defined by I/O commands.	1. Define the data area properly and then try again. 2. Make sure that the data area is properly defined before the strobe signal is turned on.
220F	I/O device changed	5	The status or configuration of I/O devices has been changed since the last operation. (Example: A new DeviceNet Slave board has been mounted.)	Update the I/O status and configuration settings and then restart the controller.
7396	An integer type variable cannot be used	3	Integer variables cannot be used in approximate comparison operation.	Change the variable type to single-precision or double-precision real.

Code	Message	Level	Description	Remedy
7799	The maximum number of TSR was exceeded.	3	The number of active supervisory tasks has exceeded its limit (32).	Modify the program so that the number of active supervisory tasks does not exceed 32.
779B	The maximum number of files was exceeded.	3	The number of files created in the controller has exceeded the limit: 256 for PAC files and 256 for the combination of header files and operation panel files.	Decrease the number of files and then try again.
7780	The max number of folders was exceeded.	3	The number of folders created has exceeded the limit (256).	Decrease the number of folders below the limit and then try again.

SUPPLEMENT

Main System Software Version 2.2*

First Edition November 2004

DENSO WAVE INCORPORATED

11F20C

The purpose of this manual is to provide accurate information in the handling and operating of the robot. Please feel free to send your comments regarding any errors or omissions you may have found, or any suggestions you may have for generally improving the manual.