

DENSO ROBOT

SUPPLEMENT

Main System Software Version 1.98

Copyright © DENSO WAVE INCORPORATED, 2003

All rights reserved. No part of this publication may be reproduced in any form or by any means without permission in writing from the publisher.

Specifications are subject to change without prior notice.

All products and company names mentioned are trademarks or registered trademarks of their respective holders.

Preface

DENSO WAVE has updated main system software designed for DENSO robot series from Version 1.95 to Version 1.98.

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.

Products covered by this manual

Robot series configured with RC5 robot controller Version 1.98 or later

Contents

1. Saving Error Codes	2
1.1 What is an error code saving feature?.....	2
1.2 Items to be set and ring buffer	2
1.3 Configuring the error code saving feature.....	3
1.4 Commands for the error code saving	4
2. Commands Added.....	5
2.1 Commands added to the multitasking control statements.....	5
SUSPENDALL (Statement)	5
KILLALL (Statement)	6
CONTINUERUN (Statement)	6
ROBOTSTOP (Statement)	7
2.2 Commands added to the robot control statements.....	8
MOTOR{ON OFF} (Statement)	8
EXTSPEED (Statement)	9
EXECAL (Statement)	9
2.3 Commands added to the system information statements	10
CHGEXTMODE (Statement)	10
CHGINTMODE (Statement)	11
CUROPTMODE (Statement).....	11
SYSSTATE (Statement)	12
2.4 Commands added to the error management	13
SETERR (Statement)	13
GETERR (Function)	13
CLRERR (Statement).....	14
GETERRLVL (Function)	14
3. Error Codes Added or Modified in version 1.98.....	15
4. Rendering Object Images by Arm Manager in WINCAPSII	16
4.1 Creating a palette base object	17
4.2 Creating a palette object.....	20
4.3 Creating a work object	23
4.4 Copying a combination object (pallet + works).....	23
4.5 Copying a combination object (pallet base + pallets + works)	24
4.6 Creating a conveyer object and its tool objects	24

1. Saving Error Codes

Refer to the SETTING-UP MANUAL, Chapter 5, Section 5.7 "Enabling extension functions" (p. 5-177).

1.1 What is an error code saving feature?

This feature saves an error code into an integer variable area (that serves as a ring buffer) if an error occurs.

Using SETERR or GETERR command newly added to version 1.98, you may designate error codes to be written or read into/from the ring buffer.⁷

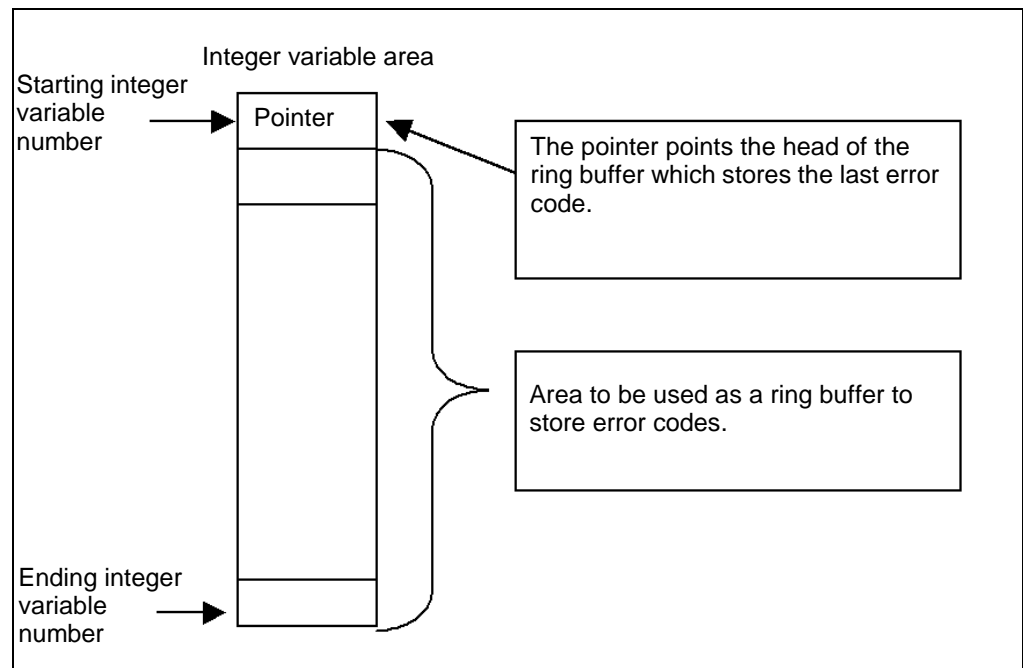
To use this feature, you need to add the required function and configure the controller using the system extension with the teach pendant as given in Section 1.3.

1.2 Items to be set and ring buffer

To use the error code saving feature, you need to:

- Enable the error storage.
- Specify the starting and ending variable numbers of integer variables to be used as a storage area of error codes.

TIP: An integer variable area to be used for storing error codes serves as a ring buffer as shown below. Since the area of the starting integer variable number is used for a pointer, the ring buffer starts with the starting variable number plus 1.



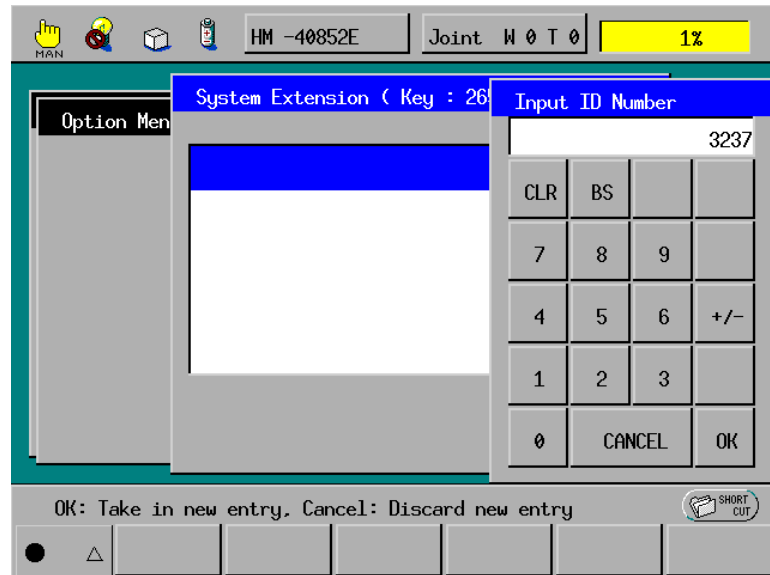
1.3 Configuring the error code saving feature

- (1) Call up the System Extension window.

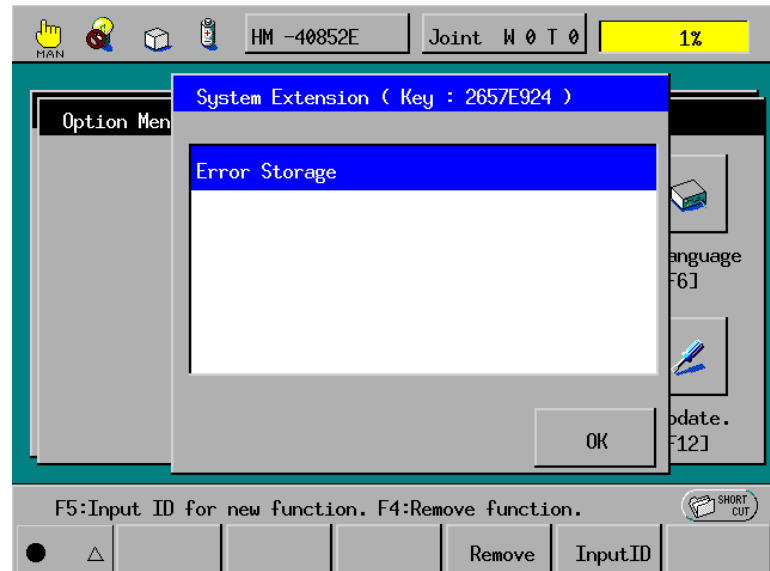
Access: [F6 Set]—[F7 Options.]—[F8 Extnsion]

- (2) Press [F5 Input ID].

The numeric keypad will appear where you enter the necessary ID code--3237 for the error code saving feature.

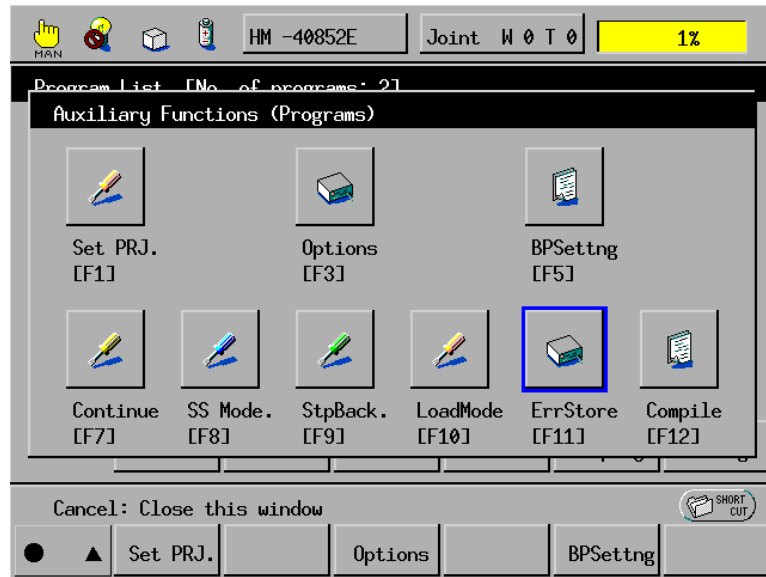


- (3) Press the OK button in the window above to add the error code saving feature.

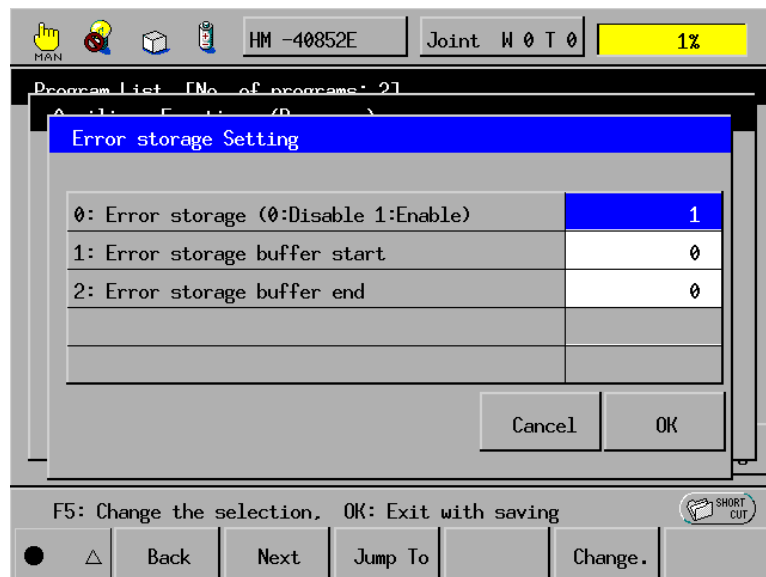


- (4) Call up the Auxiliary Functions (Programs) window and press the [F11 ErrStore] button.

Access: [F1 Program]—[F6 Aux.]—[F11 ErrStore]



- (5) The error code saving feature setting dialog will appear where you can configure the feature.



Item	Configuration
0: Error storage	To enable the feature, enter "1."
1: Error storage buffer start	Set the start integer variable number in the error code saving ring buffer.
2: Error storage buffer end	Set the end integer variable number in the error code saving ring buffer.

1.4 Commands for the error code saving

SETERR and GETERR

Refer to Section 2.4 "Commands added to the error management."

2. Commands Added

This section describes commands (statements and functions) newly added to version 1.98.

2.1 Commands added to the multitasking control statements

Refer to the PROGRAMMER'S MANUAL I, Chapter 14.

2.1.1 Task control statements

SUSPENDALL (Statement)

Function Suspends all running programs except supervisory tasks.

Syntax `SUSPENDALL`

Description This statement suspends all tasks except supervisory tasks, makes them enter the "Continue Stop" state, and turns off the "Robot-in-operation" output signal.

Related commands `SUSPEND`, `KILLALL`, `ROBOTSTOP`, and `CONTINUERUN`

Coding example

```

PROGRAM TSR1
-----
SUSPENDALL           'Immediately stop all tasks and enter "Continue Stop" status.
-----
CONTINUERUN          'Continue start.
END

```

Notes Programs stopped by the `SUSPENDALL` statement can be restarted from the subsequent steps by the `RUN` command. They can also be continue-started by the `CONTINUERUN` command.

The `RUN` or `CONTINUERUN` command for restarting a suspended task should be executed after the motion has completely stopped. Otherwise, a "Command speed limit over" error or such errors may occur.

Once the `SUSPENDALL` statement executes, the controller cannot restart any program for 0.5 second after the execution, just as when any of forced stop commands, e.g. "Robot stop" input signal is entered from the teach pendant or any external equipment.

KILLALL (Statement)

Function	Forcibly terminates all tasks except supervisory tasks. (Functionally equivalent to the "Program reset" command)
Syntax	KILLALL
Description	This statement forcibly terminates all tasks except supervisory tasks and turns off the "Robot-in-operation" output signal.
Related commands	KILL, SUSPENDALL, and ROBOTSTOP

Coding example

```

PROGRAM TSRL
-----
KILLALL           'Terminate all tasks and enter the program reset state.
-----
END

```

Notes	<p>Programs terminated by this statement can be no longer restarted.</p> <p>Once the KILLALL statement executes, the controller cannot restart any program for 0.5 second after the execution, just as when any of forced stop commands, e.g. "Robot stop" input signal is entered from the teach pendant or any external equipment.</p>
--------------	--

CONTINUERUN (Statement)

Function	Continue-runs tasks.
Syntax	CONTINUERUN
Description	Restarts all continue-stopped tasks from the subsequent steps.
Related commands	SUSPENDALL and ROBOTSTOP

Coding example

```

PROGRAM TSRL
-----
SUSPENDALL        'Suspend all tasks and enters "continue stop" state.
-----
CONTINUERUN       'Start to continue-run the all the tasks stopped.
-----
END

```

Notes	This statement can be executed only in a supervisory task when the Continue Start is permitted.
--------------	---

ROBOTSTOP (Statement)

Function Stops the robot.

Syntax ROBOTSTOP

Description This statement stops all tasks except supervisory tasks, makes them enter the "Continue Stop" state, shuts down the motor driving power, and turns off the "Robot-in-operation" output signal.

Related commands SUSPENDALL and CONTINUERUN

Coding example

```
PROGRAM TSR1
-----
ROBOTSTOP           'Stop the robot.
-----
END
```

Notes The difference between this statement and SUSPENDALL is that this statement shuts down the motor driving power and changes the related output signal states.

Once the ROBOTSTOP statement executes, the controller cannot restart any program for 0.5 second after the execution, just as when any of forced stop commands, e.g. "Robot stop" input signal is entered from the teach pendant or any external equipment.

Note that if the "Continue" parameter* is set to Disabled ("0"), this statement does not continue-stop the tasks but halts them.

*The "Continue" parameter is shown on the Continue Parameters Setting window that can be called up by pressing the [F1: Program]—[F6 Aux.]—[F7 Continue] on the teach pendant.

2.2 Commands added to the robot control statements

Refer to the PROGRAMMER'S MANUAL I, Chapter 12.

2.2.1 Motor power control statements

MOTOR {ON | OFF} (Statement)

Function Turns the motor power on or off.

Syntax MOTOR{ON|OFF}

Description This statement turns the power to the robot motor on or off.

Related commands INIT

Coding example

```
PROGRAM PRO1
-----
MOTOR ON           "Turn on the motor power.
-----
MOTOR OFF          "Turn off the motor power.
-----
END
```

Notes

The MOTOR OFF statement is functionally equivalent to the motor off key on the teach pendant. That is, executing this statement stops the robot motion and program execution and then brings tasks into the "Continue stop" state.

If the "296: Motor command setting" is modified from "0" to "1" on the User Preferences window*, executing the MOTOR OFF stops the motor but does not stop the program being executed.

The MOTOR OFF statement cannot execute when the robot is in operation. Attempting to do so will result in an error, quitting the program execution.

However, the "296: Motor command setting" does not affect the execution of the MOTOR ON statement.

*The User Preferences window can be called up by pressing the [F2 Arm]—[F6 Aux.]—[F7 Config.].

2.2.2 Speed control statements

EXTSPEED (Statement)

Function Sets the external speed.

Syntax `EXTSPEED <External Speed>`

Description This statement sets the reduced ratio (%) of the internal speed (programmed speed).
 Tip: In older system software versions, the external speed (reduced ratio) can be set only from the teach pendant or external equipment. In version 1.98, it can be set also in programs by using this statement.

Related commands `SPEED`

Coding example

```
PROGRAM PRO1
-----
EXTSPEED 100           'Set the external speed at 100.
-----
END
```

2.2.3 Calibration statements

EXECAL (Statement)

Function Executes CAL operation.

Syntax `EXECAL`

Description This statement executes CAL operation.

Related commands `INIT`

Programming example

```
PROGRAM PRO1
-----
EXECAL                 'Execute CAL operation.
-----
END
```

Notes In the robots requiring no CAL operation such as the E series of robots, this statement produces nothing.

2.3 Commands added to the system information statements

Refer to the PROGRAMMER'S MANUAL I, Chapter 19.

2.3.1 Operation mode control statements

CHGEXTMODE (Statement)

Function Switches from internal to external auto mode.

Syntax CHGEXTMODE

Description This statement switches the operation mode from internal to external auto mode.

Related commands INIT and CHGINTMODE

Coding example

```

PROGRAM TSR1
-----
CHGEXTMODE           'Switch to the external auto mode.
-----
CHGINTMODE           'Switch to internal auto mode.
-----
END
    
```

Notes This statement cannot switch from the manual mode or teach check mode to the external auto mode.

This statement can execute only when a supervisory task is running and no user tasks are running.

CHGINTMODE (Statement)

Function Switches from external to internal auto mode.

Syntax CHGINTMODE

Description This statement switches the operation mode from external to internal auto mode.

Related commands INIT and CHGEXTMODE

Coding example

```

PROGRAM TSR1
-----
CHGEXTMODE           'Switch to the external auto mode.
-----
CHGINTMODE           'Switch to internal auto mode.
-----
END

```

Notes This statement cannot switch from the manual mode or teach check mode to the internal auto mode.

This statement can execute only when a supervisory task is running and no user tasks are running.

CUROPTMODE (Statement)

Function Gets the current operation mode.

Syntax CUROPTMODE

Description This statement gets the current operation mode as a value (any of 1 to 4 shown below).

1: Manual, 2: Teach check, 3: Internal auto 4: External auto

Coding example

```

PROGRAM PRO1
-----
I1 = CUROPTMODE      'Get the current operation mode.
-----
END

```

SYSSTATE (Statement)

Function

Gets the system status of the robot controller.

Syntax

SYSSTATE

Description

This statement gets the system status of the robot controller. The status data differs depending upon the I/O line assignment. Listed below are data that can be obtained.

Bit 0	Robot-in-operation signal
1	Robot failure signal
2	Servo ON signal
3	Robot initialization complete signal (in the I/O standard mode) Robot power on complete signal (in the I/O compatible mode)
4	Auto mode signal
5	External mode signal
6	Dead battery warning signal
7	Robot warning signal
8	Continue start permitted signal
9	SS mode signal
10	Robot stop signal
11	Enable Auto signal
12 to 15	Reserved.
16	Program start reset signal (in the I/O compatible mode)
17	CAL complete signal (in the I/O compatible mode)
18	Teaching signal (in the I/O compatible mode)
19	Single-cycle end signal (in the I/O compatible mode)
20 to 23	Reserved.
24	Command processing complete signal (in the I/O standard mode)
25 to 31	Reserved.

Coding example

```
PROGRAM TSR1
```

```

I1 = SYSSTATE           'Get the system status of robot controller.
IF (I1 AND &h0082) THEN 'If any failure or warning occurs,
CLRERR                  'clear the error.
END IF

```

```
END
```

2.4 Commands added to the error management

Refer to the PROGRAMMER'S MANUAL I, Chapter 18.

SETERR (Statement)

Function Saves a specified error code into an integer variable area (to be used as a ring buffer).

Syntax SETERR <Error code>

Description This statement saves an error code specified by <Error code> into an integer variable area declared by the error code saving feature, and counts up the pointer address by one.

Related commands CLRERR, GETERRRLVL and GETERR

Coding example

```
PROGRAM PRO1
-----
SETERR 100          ' Save error code "100" into the ring buffer.
-----
END
```

Notes The error code saving feature should be configured beforehand (refer to Section 1.3).
Be careful with some commands or teach pendant operation that may change integer variable areas declared by the error code saving feature.

GETERR (Function)

Function Gets the error code from the ring buffer declared by the error code saving feature.

Syntax GETERR(<Expression>)

Description This function reads out the error code from the ring buffer declared by the error code saving feature.

To read out the latest error code, set "0" in the argument.

Related commands CLRERR, GETERRRLVL and SETERR

Coding example

```
PROGRAM PRO1
-----
I1 = GETERR(0)      'Get the latest error code from the ring buffer.
-----
END
```

Notes The error code saving feature should be configured beforehand (refer to Section 1.3).
Be careful with some commands or teach pendant operation that may change integer variable areas declared by the error code saving feature.
If the error code saving feature has not been configured, this function reads out the error code from the regular error log area.

CLRERR (Statement)

Function	Clears the current error.
Syntax	CLRERR
Description	This statement clears the error that occurs currently.
Related commands	ERR, SETERR and GETERR
Coding example	

```

PROGRAM TSR1
-----
I1 = SYSSTATE           'Get the system status.
IF (I1 AND &H0082) THEN 'If a failure or warning occurs,
CLRERR                  'clear the error.
END IF
-----
END

```

Notes	This statement is exclusively executable from a supervisory task.
--------------	---

GETERRLVL (Function)

Function	Gets the error level.
Syntax	GETERRLVL(<Numeric expression>)
Description	This function gets the level of the error code specified by <Numeric expression>.
Related commands	CLRERR, SETERR and GETERR

Coding example

```

PROGRAM PRO1
-----
I1 = GETERRLVL(&H6001)   'Get the error level of error code "6001."
-----
END

```

Notes	If the related error has not occurred, this function returns "-1."
--------------	--

3. Error Codes Added or Modified in version 1.98

Refer to the ERROR CODE TABLES.

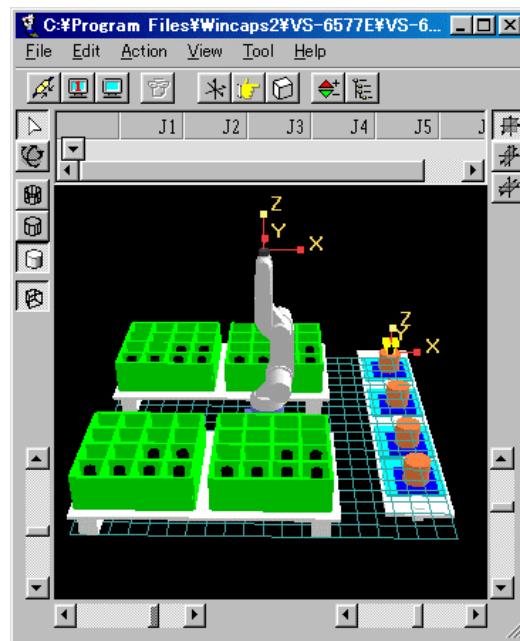
Code	Message	Level	Description	Recovery
279C	Can't set when DETECT is valid.	2	Hand I/O interrupt and DETECT are not enabled simultaneously.	Disable the DETECT.
279D	Can't set. (Hand I/O interrupt is valid.)	2	DETECT and Hand I/O interrupt are not enabled simultaneously.	Disable the hand I/O interrupt.
77B7	Motor off command while robot is running.	3	The motor OFF command cannot execute when the robot is running.	Correct your program so as not to execute the motor off command when the robot is running.
77B8	Motor on command with deadman switch off.	3	In Robot System "Type A," the motor ON command cannot execute when the deadman switch is released in Manual mode or Teach check mode.	When executing the motor ON command in Manual mode or Teach check mode, hold down the deadman switch.

4. Rendering Object Images by Arm Manager in WINCAPSII

Refer to the WINCAPSII GUIDE, Chapter 8, Section 8.6.3, "Object Tree."

NOTE: This chapter does not include any information about new features enhanced in version 1.98. It provides a sample of the object image rendering procedure that WINCAPSII originally features as a facility of Arm Manager, for your reference.

This chapter describes how to create object images including palette bases, palettes, works, conveyer and its tool as shown below.




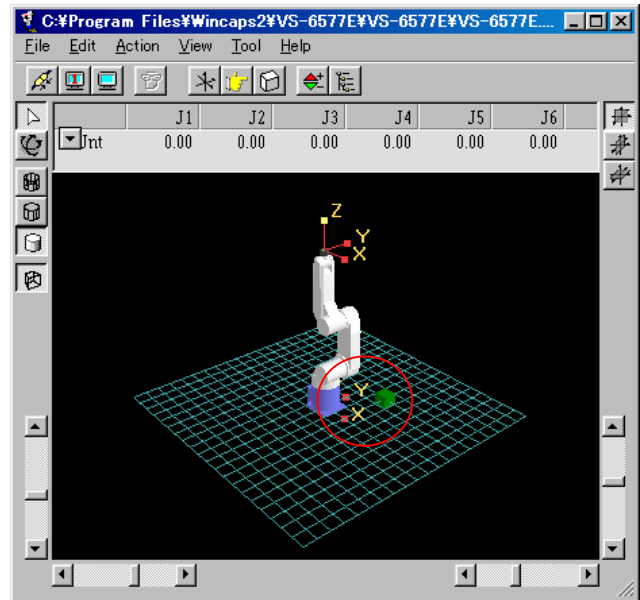
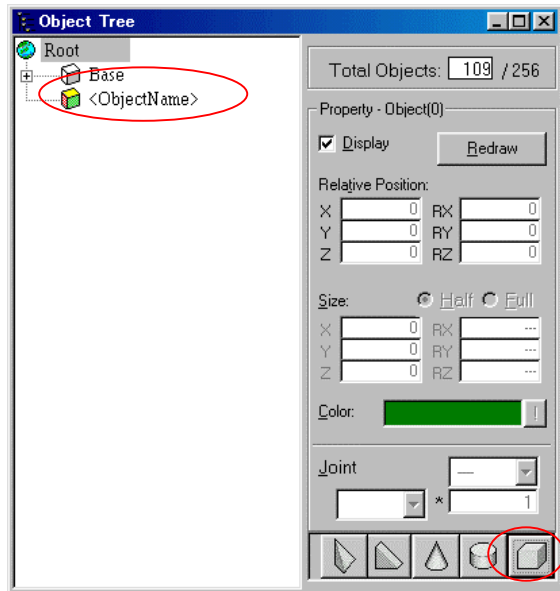
The object image rendering procedure consists of the following steps:

- (1) Creating a palette base
 - Resizing the object
 - Moving the object position
 - Copying the object
- (2) Creating a palette
- (3) Creating a work
- (4) Copying a combined object (palette + work)
- (5) Copying a combined object (palette base + palette + work)
- (6) Creating a conveyer and its tool object, then setting the work object onto the conveyer tool object.

4.1 Creating a palette base object

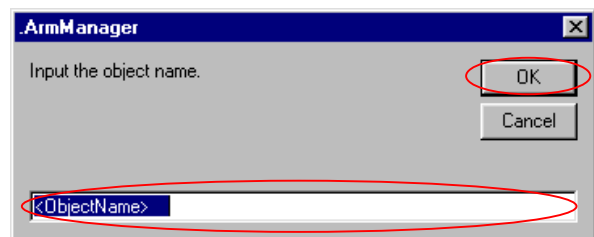
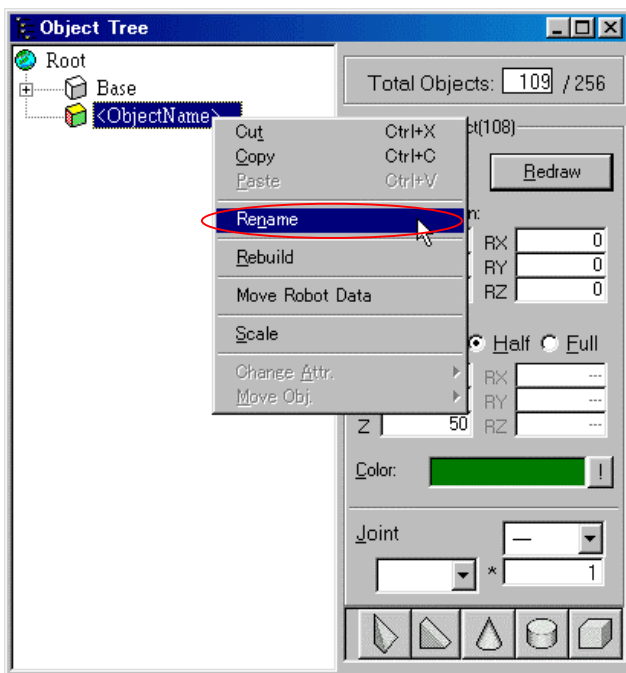
- (1) On the Tool menu, click **Object Tree** to open the Object Tree dialog box.
- (2) First let's create one foot of a palette base object.

Click the  button to create a square base image object. The object will be added to the object tree at the same time.

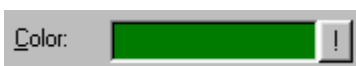


- (3) On the Object Tree, right-click the **<ObjectName>** and click **Rename** in the dropdown menu to give it the desired object name.

Enter the object name you want to use and click **OK**.

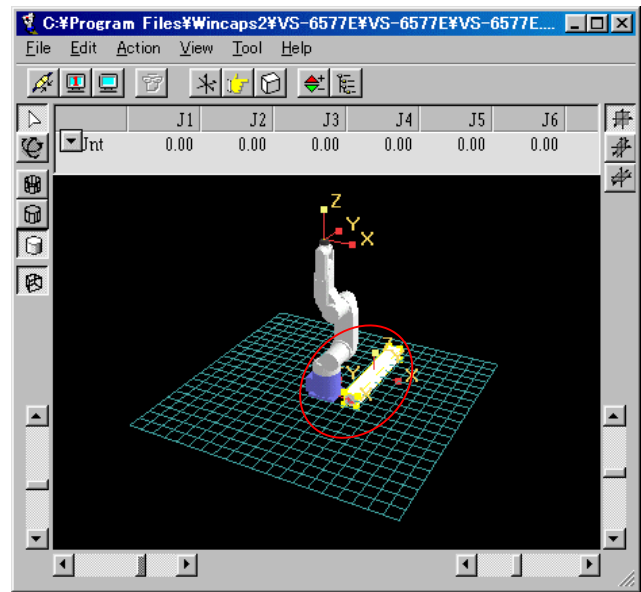
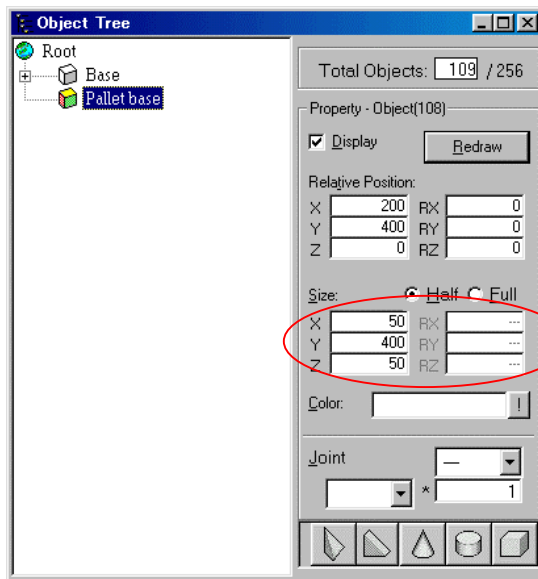


- (4) Change the object color, if you want, by clicking the color buttons.

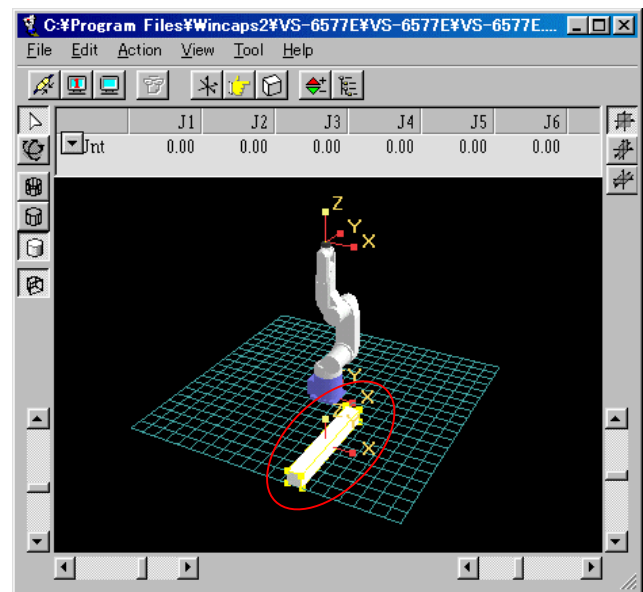
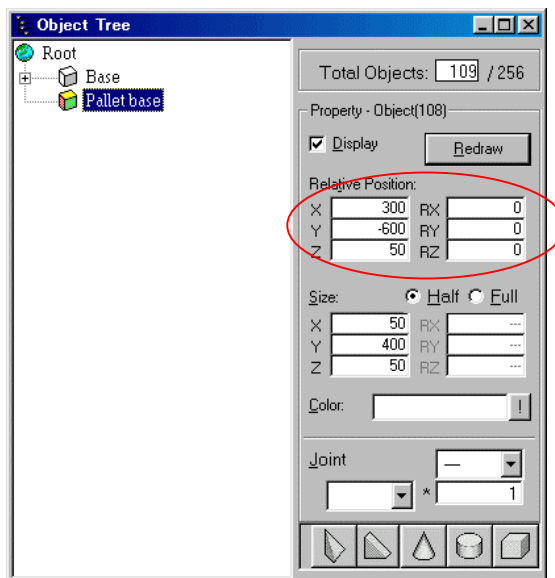


Samples of image object rendering in WINCAPSII

- (5) To change the object size, click the object you want to change in size and enter the desired size (numeric values) into the Size boxes as shown below at left.



- (6) To move the object position, click the object you want to move and enter the target position (numeric values) into the Relative Position boxes as shown below at left.

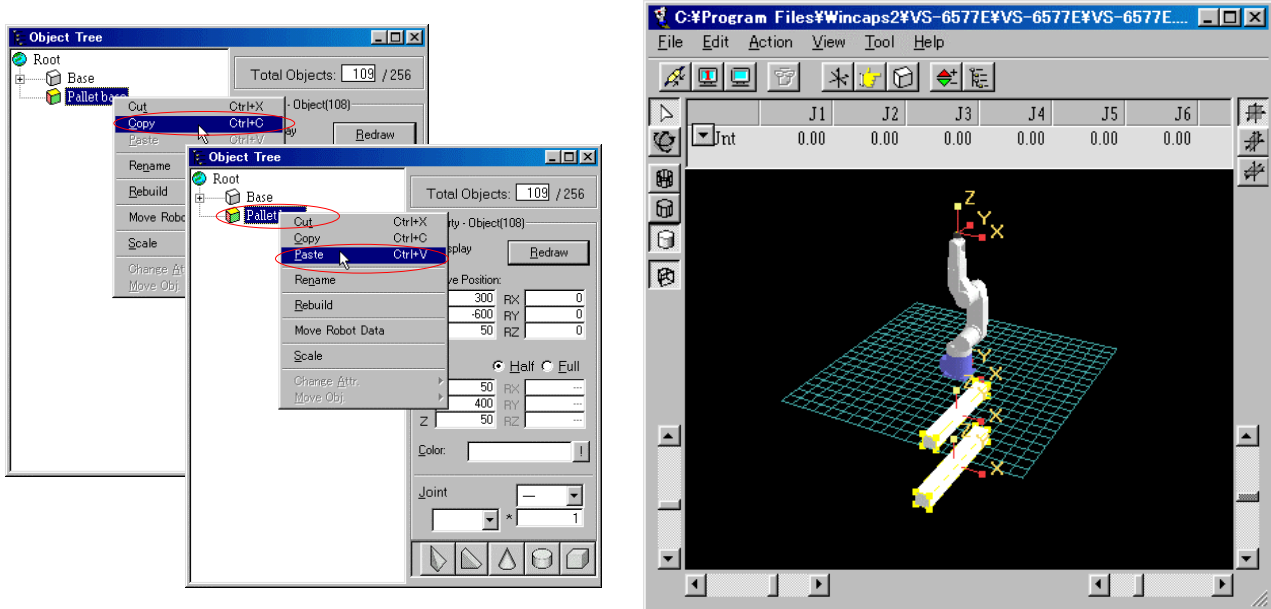


Samples of image object rendering in WINCAPSII

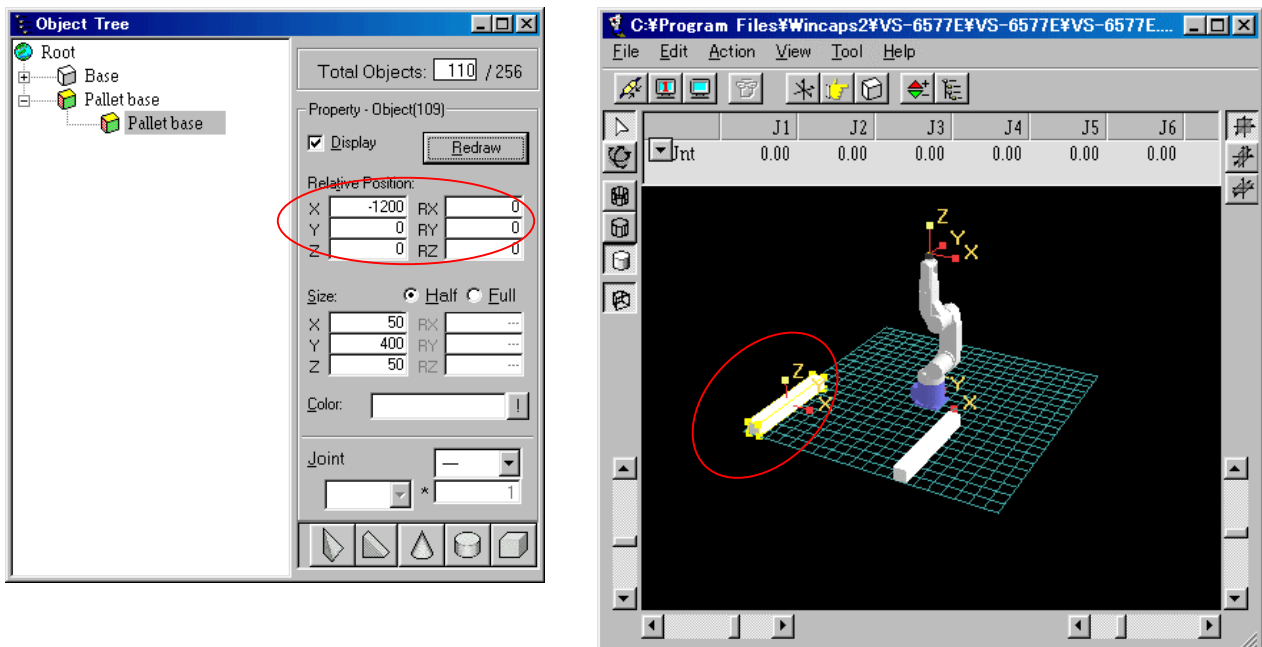
- (7) To create another foot of a pallet base, copy and paste the object in Arm Manager according to these steps:

Right-click the object (the foot created in the steps above) you want to copy and click **Copy** as shown below at left.

Click the target place where the object should be pasted, then click **Paste**.

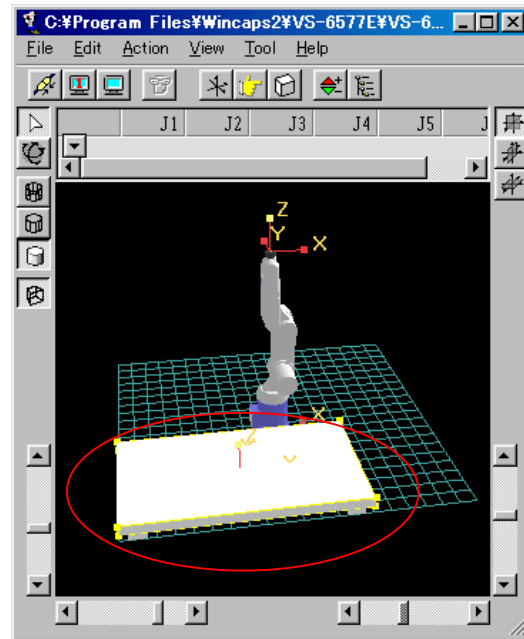
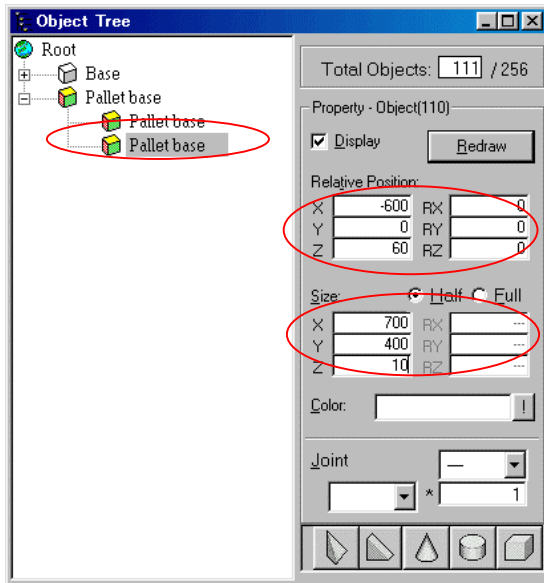


- (8) To move the pasted object in Arm Manager, click the object and enter the target position (numeric values) into the Relative Position boxes as shown below at left.




- (9) Let's create a palette base object.

Copy and paste the object and enter the desired size and target position into the Size and Relative Position boxes, respectively, as shown below at left.

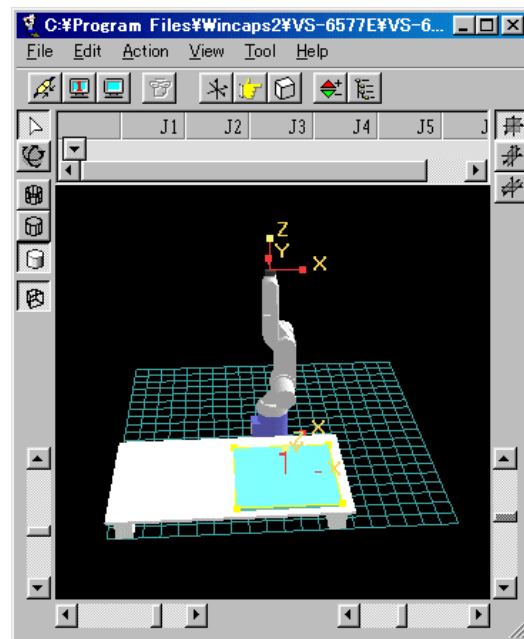
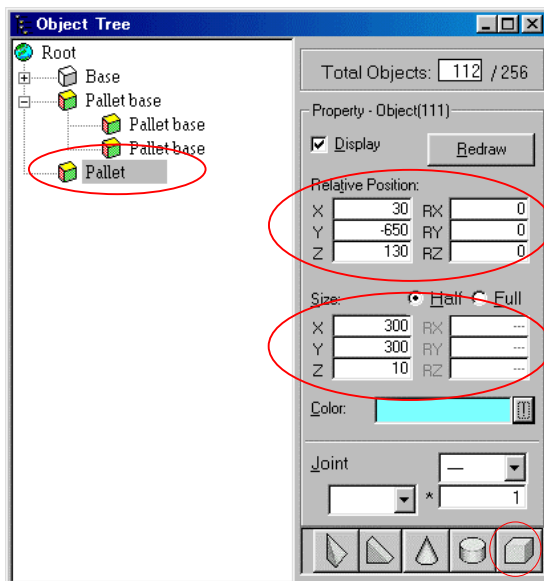


4.2 Creating a palette object

- (1) First let's create a palette bottom plate.

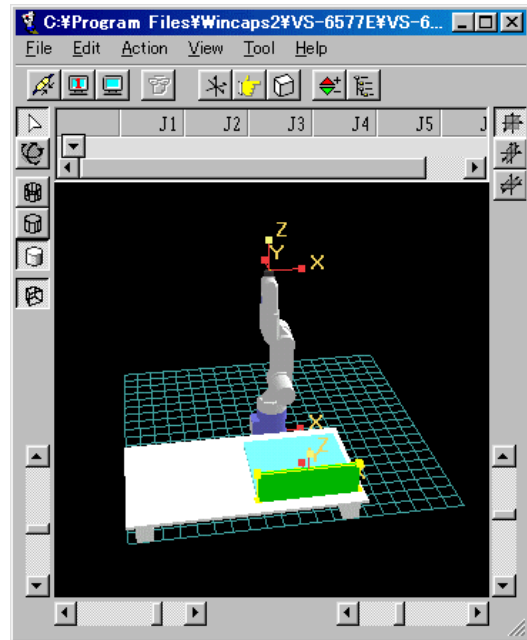
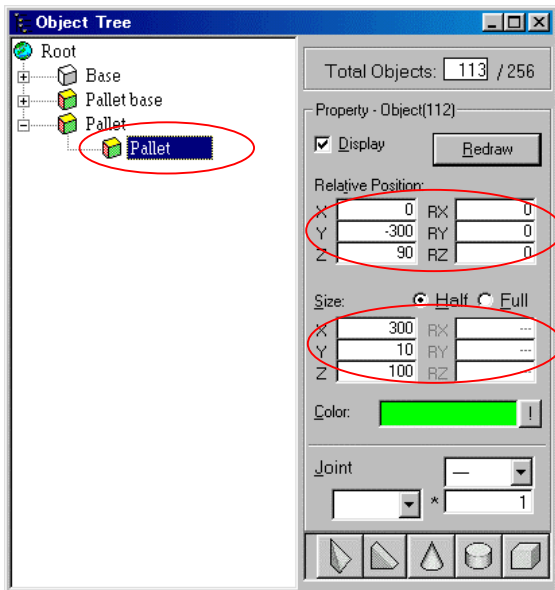
Click the  button in the object tree window to create a square bottom plate object. The object will be added to the object tree at the same time.

- (2) Enter the desired size and the target position into the Size and Relative Position boxes, respectively, to set the pallet bottom plate.



(3) Let's create a partition.

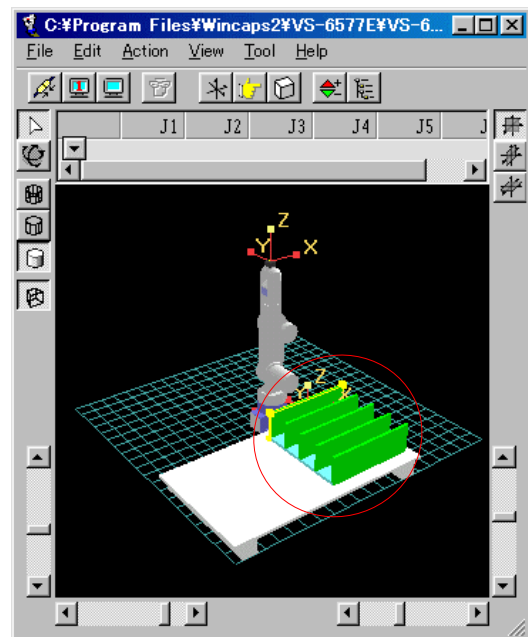
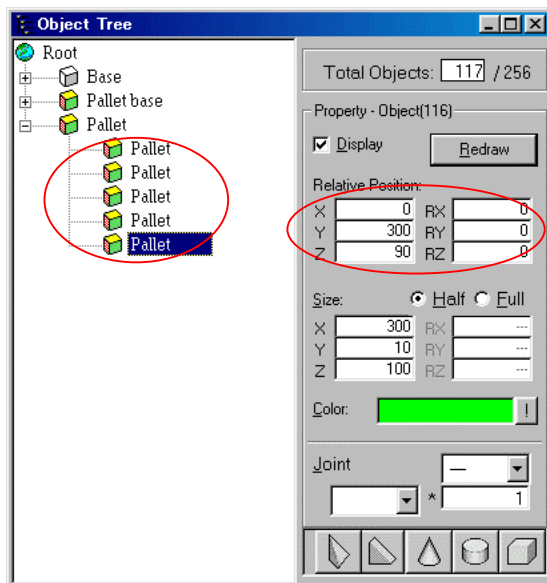
Click **Pallet** in the object tree and enter the desired size and target position into the Size and Relative Position boxes, respectively, based on the bottom plate created.



(4) Let's copy the original partition object created in step (3) to arrange it in parallel with the original partition object.

Copy the original partition object and click the target position to paste it.

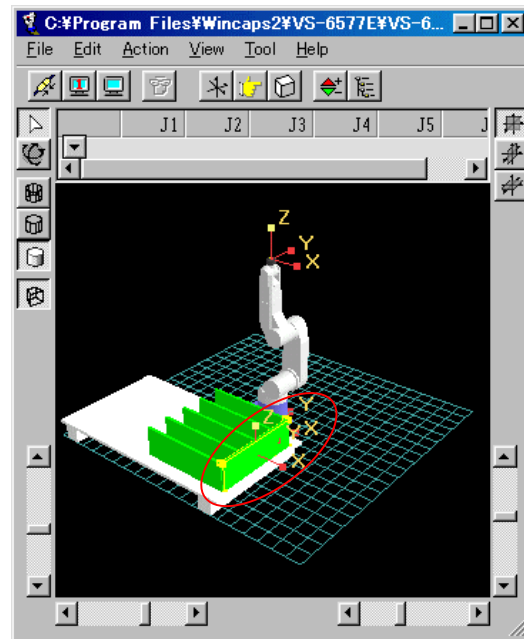
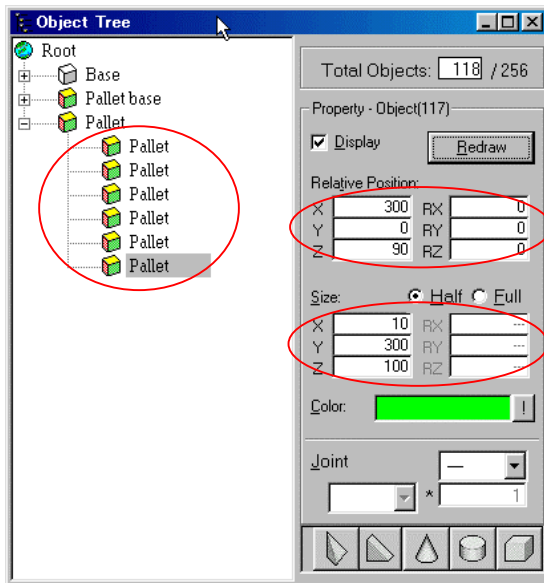
Enter the desired position (numeric values) into the Relative Position boxes to set the partition. Repeat this operation to create the required number of partitions.



Samples of image object rendering in WincapsII

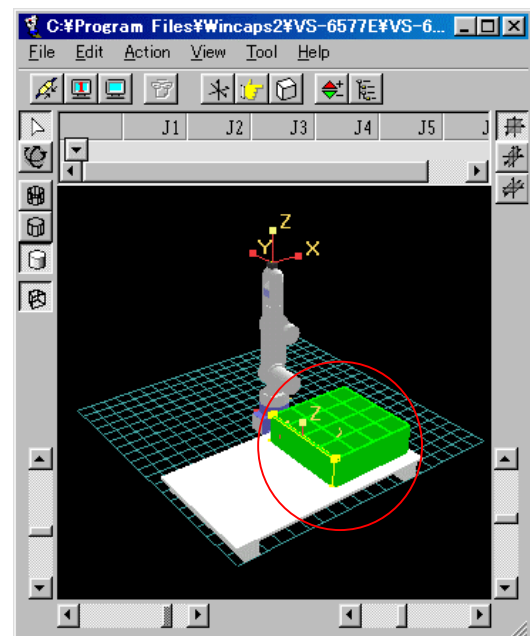
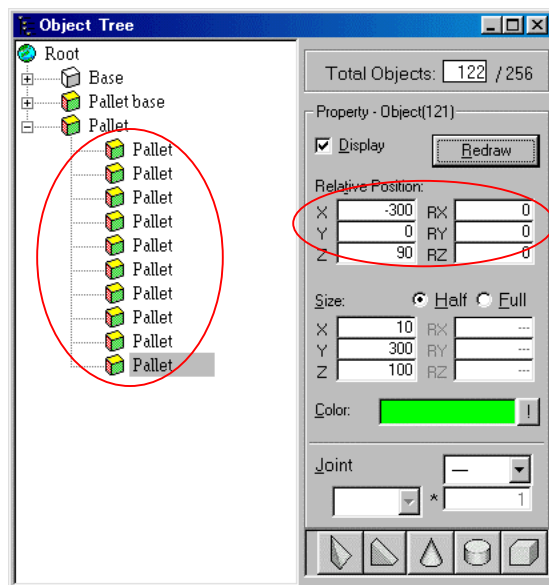
- (5) Let's create a partition perpendicular to the partitions created in steps above.

Click **Pallet** in the object tree and copy the original partition object and click the target position to paste it. Enter the desired position (numeric values) into the Relative Position boxes to set the partition.




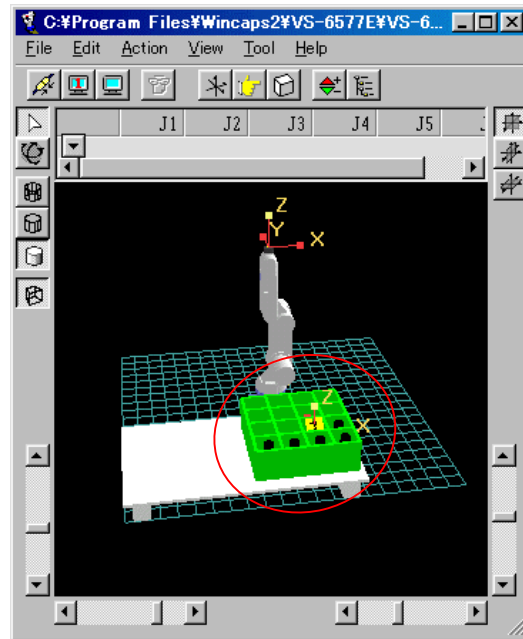
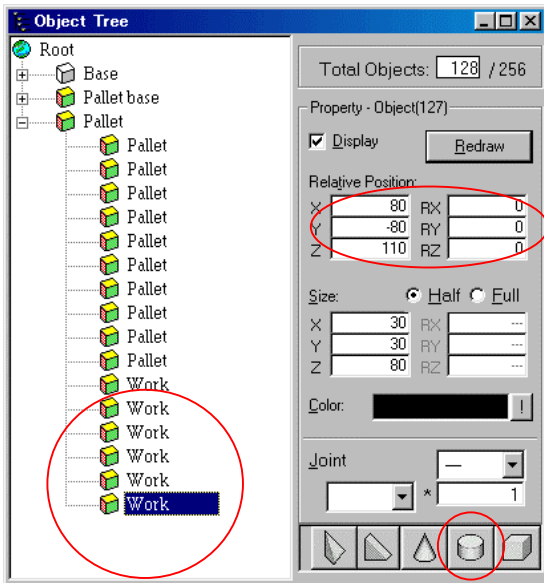
- (6) Let's copy the original partition object created in step (5) to arrange it in parallel with the original partition object.

Copy the original partition object and click the target position to paste it. Enter the desired position (numeric values) into the Relative Position boxes to set the partition. Repeat this operation to create the required number of partitions.

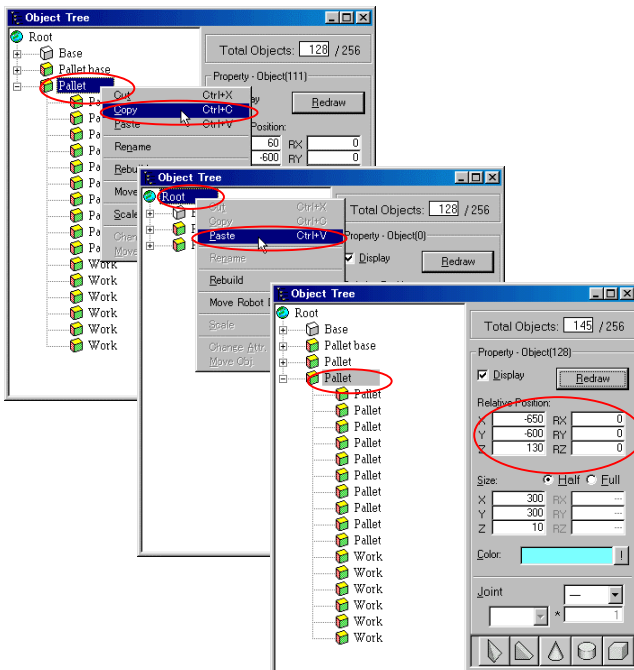


4.3 Creating a work object

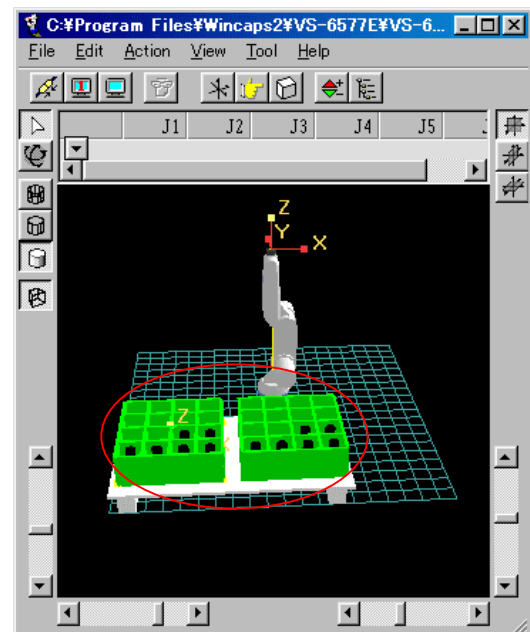
Create a work object by using the  button in the Object Tree window and put it in the partitioned space, in the same way as described in the previous sections.



4.4 Copying a combination object (pallet + works)

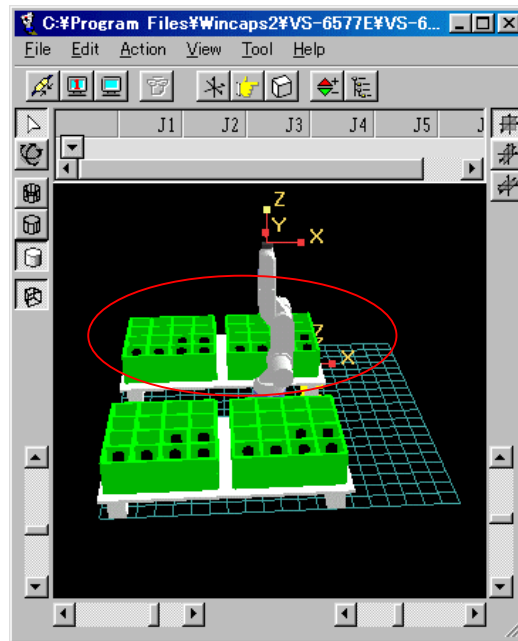


- (1) In the Object Tree, right-click a higher class object containing objects to be copied, and click **Copy**.
- (2) Click the object to which you want to paste the copied object, then click **Paste**.
- (3) Enter the target position (numeric values) into the Relative Position boxes to set a combination object (pallet + works) as shown below.



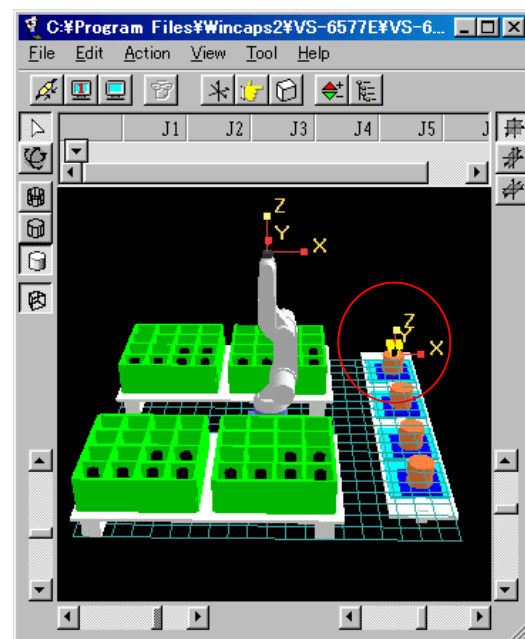
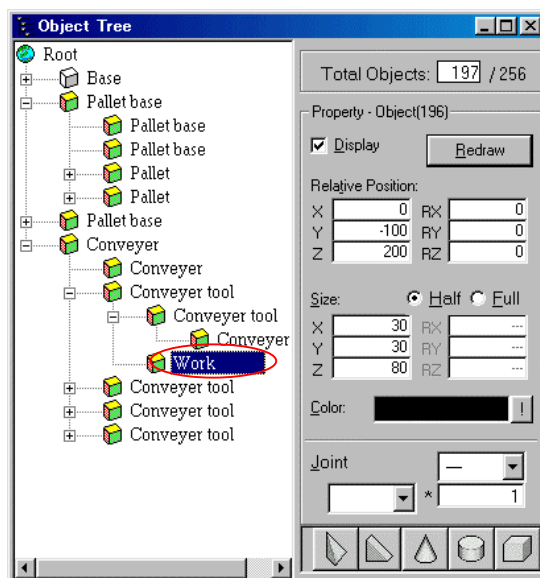
4.5 Copying a combination object (pallet base + pallets + works)

In the same way as described above, copy a combination object of a pallet base, pallets and works onto the desired position.



4.6 Creating a conveyer object and its tool objects

In the same way as described in the previous sections, create a conveyer object and its tool objects and then put works on the conveyer tools.



SUPPLEMENT

Main System Software Version 1.98

First Edition May 2003

DENSO WAVE INCORPORATED
Factory Automation Division

7E50C

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.

In no event will DENSO WAVE INCORPORATED be liable for any direct or indirect damages resulting from the application of the information in this manual.

