DENSO ROBOT

SUPPLEMENT

Extended-Joints Support (H*-D, VS-D)
Main Software Enhancement
(Version 1.5 & Version 1.6)

Preface

Thank you for purchasing the DENSO high-speed, high-accuracy assembly robot.

This book is a supplement to those manuals listed on the next page. It describes "Extended-joints support" (option for the H∗-D series & VS-D series) that enables you to control up to 2 extended joints with the robot controller and "Main software enhancement provided in Version 1.5 & Version 1.6."

DENSO products covered by this manual

Extended-joints support

- Horizontal articulated robots, H∗-D series (Ver. 1.5 or later)
- Vertical articulated robots, VS-D series (Ver. 1.6 or later)

Main software enhancement

- Horizontal articulated robots, H∗-D series and XYC-4D series (Ver. 1.5 or later)
- Vertical articulated robots, V∗-D series (Ver. 1.6 or later)

Important

To ensure operator safety, be sure to read the precautions and instructions in "SAFETY PRECAUTIONS" on pages 1 through 9.

How the H*-D documentation set is organized

The H*-D documentation set consists of the following seven books. If you are unfamiliar with this robot series, please read all seven books and understand them fully before operating your robot.

SUPPLEMENT - this book -

Describes newly added features--"Extended-joints support" and "Main software enhancement provided in Version 1.5 and Version 1.6."

BEGINNER'S GUIDE

Introduces you to the DENSO robot. Taking an equipment setup example, this book guides you through running your robot with the teach pendant, making a program in WINCAPSII, and running your robot automatically.

INSTALLATION & MAINTENANCE GUIDE

Provides an explanation of the robot outline, instructions for installing the robot components, and maintenance & inspection procedures.

The optional devices include an operating panel, a teach pendant, PC teaching system "WINCAPSII," a floppy disk drive, μVision board, Ethernet board, and DeviceNet board.

SETTING-UP MANUAL

Describes how to set-up or teach your robot with the teach pendant or operating panel.

WINCAPSII GUIDE (that comes with WINCAPSII)

Provides instructions on how to use the teaching system WINCAPSII which runs on the PC connected to the robot and its controller for developing and managing programs.

PROGRAMMER'S MANUAL

Describes the PAC programming language, program development steps in PAC, and command specifications.

ERROR CODE TABLES

List error codes that will appear on the teach pendant, operating panel, or PC screen if an error occurs in the robot series or WINCAPSII.

SAFETY PRECAUTIONS

Be sure to observe all of the following safety precautions.

Strict observance of these warning and caution indications are a MUST for preventing accidents, which could result in bodily injury and substantial property damage. Make sure you fully understand all definitions of these terms and related symbols given below, before you proceed to the text itself.

⚠ **WARNING**	Alerts you to those conditions, which could result in serious bodily injury or death if the instructions are not followed correctly.
⚠ **CAUTION**	Alerts you to those conditions, which could result in minor bodily injury or substantial property damage if the instructions are not followed correctly.

Terminology and Definitions

Maximum space: Refers to the volume of space encompassing the maximum designed movements of all robot parts including the end-effector, workpiece and attachments. (Quoted from the RIA* Committee Draft.)

Restricted space: Refers to the portion of the maximum space to which a robot is restricted by limiting devices (i.e., mechanical stops). The maximum distance that the robot, end-effector, and workpiece can travel after the limiting device is actuated defines the boundaries of the restricted space of the robot. (Quoted from the RIA Committee Draft.)

Motion space: Refers to the portion of the restricted space to which a robot is restricted by software motion limits. The maximum distance that the robot, end-effector, and workpiece can travel after the software motion limits are set defines the boundaries of the motion space of the robot. (The "motion space" is Denso-proprietary terminology.)

Operating space: Refers to the portion of the restricted space (or motion space in Denso) that is actually used by the robot while performing its task program. (Quoted from the RIA Committee Draft.)

Task program: Refers to a set of instructions for motion and auxiliary functions that define the specific intended task of the robot system. (Quoted from the RIA Committee Draft.)

(*RIA: Robotic Industries Association)

1. Introduction

This section provides safety precautions to be observed during installation, teaching, inspection, adjustment, and maintenance of the robot.

2. Installation Precautions

2.1 Insuring the proper installation environment

2.1.1 For standard type

The standard type has not been designed to withstand explosions, dust-proof, nor is it splash-proof. Therefore, it should not be installed in any environment where:

(1) there are flammable gases or liquids,

(2) there are any shavings from metal processing or other conductive material flying about,

(3) there are any acidic, alkaline or other corrosive gases,

(4) there is cutting or grinding oil mist,

(5) it may likely be submerged in fluid,

(6) there is sulfuric cutting or grinding oil mist, or

(7) there are any large-sized inverters, high output/high frequency transmitters, large contactors, welders, or other sources of electrical noise.

2.1.2 For dust-proof, splash-proof type

The dust-proof, splash-proof type is an IP54-equivalent dust-proof and splash-proof structure, but it has not been designed to withstand explosions.

Note that the robot controller is not a dust- or splash-proof structure. Therefore, when using the robot controller in an environment exposed to mist, put it in an optional protective box.

The dust-proof, splash-proof type should not be installed in any environment where:

(1) there are any flammable gases or liquids,

(2) there are any acidic, alkaline or other corrosive gases,

(3) there are any large-sized inverters, high output/high frequency transmitters, large contactors, welders, or other sources of electrical noise,

(4) it may likely be submerged in fluid,

(5) there are any grinding or machining chips or shavings,

(6) any machining oil other than DENSO authorized oil is in use, or

Note: DENSO authorized oil: Yushiron Oil No. 4C (non-soluble)

(7) there is sulfuric cutting or grinding oil mist.

2.2 Service space

The robot and peripheral equipment should be installed so that sufficient service space is maintained for safe teaching, maintenance, and inspection.

2.3 Control devices outside the robot's restricted space

The robot controller, teach pendant, and operating panel should be installed outside the robot's restricted space and in a place where you can observe all of the robot's movements when operating the robot controller, teach pendant, or operating panel.

2.4 Positioning of gauges

Pressure gauges, oil pressure gauges and other gauges should be installed in an easy-to-check location.

2.5 Protection of electrical wiring and hydraulic/pneumatic piping

If there is any possibility of the electrical wiring or hydraulic/pneumatic piping being damaged, protect them with a cover or similar item.

2.6 Positioning of emergency stop switches

Emergency stop switches should be provided in a position where they can be reached easily should it be necessary to stop the robot immediately.

(1) The emergency stop switches should be red.

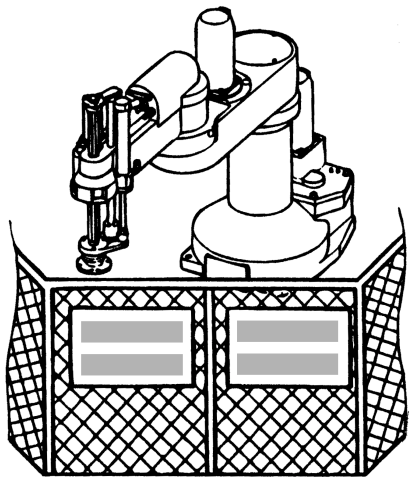
(2) Emergency stop switches should be designed so that they will not be released after pressed, automatically or mistakenly by any other person.

(3) Emergency stop switches should be separate from the power switch.

2.7 Positioning of operating status indicators

Operating status indicators should be positioned in such a way where workers can easily see whether the robot is on temporary halt or on an emergency or abnormal stop.

2.8 Setting-up the safety fence or enclosure



A safety fence or enclosure should be set up so that no one can easily enter the robot's restricted space. If it is impossible, utilize other protectors as described in Section 2.9.

(1) The fence or enclosure should be constructed so that it cannot be easily moved or removed.

(2) The fence or enclosure should be constructed so that it cannot be easily damaged or deformed through external force.

(3) Establish the exit/entrance to the fence or enclosure. Construct the fence or enclosure so that no one can easily get past it by climbing over the fence or enclosure.

(4) The fence or enclosure should be constructed to ensure that it is not possible for hands or any other parts of the body to get through it.

(5) Take any one of the following protections for the entrance/ exit of the fence or enclosure:

1) Place a door, rope or chain across the entrance/exit of the fence or enclosure, and fit it with an interlock that ensures the emergency stop device operates automatically if it is opened or removed.

2) Post a warning notice at the entrance/exit of the fence or enclosure stating "In operation--Entry forbidden" or "Work in progress--Do not operate" and ensure that workers follow these instructions at all times.

When making a test run, before setting up the fence or enclosure, place an overseer in a position outside the robot’s restricted space and one in which he/she can see all of the robot’s movements. The overseer should prevent workers from entering the robot's restricted space and be devoted solely to that task.

2.9 Positioning of rope or chain

If it is not possible to set up the safety fence or enclosure described in Section 2.8, hang a rope or chain around the perimeter of the robot’s restricted space to ensure that no one can enter the restricted space.

(1) Ensure the support posts cannot be moved easily.

(2) Ensure that the rope or chain’s color or material can easily be discerned from the surrounds.

(3) Post a warning notice in a position where it is easy to see stating "In operation--Entry forbidden" or "Work in progress --Do not operate" and ensure that workers follow these instructions at all times.

(4) Set the exit/entrance, and follow the instructions given in Section 2.8, (3) through (5).

2.10 Setting the robot's motion space

The area required for the robot to work is called the robot's operating space.

If the robot's motion space is greater than the operating space, it is recommended that you set a smaller motion space to prevent the robot from interfering or disrupting other equipment.

Refer to the "INSTALLATION & MAINTENANCE GUIDE" Chapter 4.

2.11 No robot modification allowed

Never modify the robot unit, robot controller, teach pendant or other devices.

2.12 Cleaning of tools

If your robot uses welding guns, paint spray nozzles, or other end-effectors requiring cleaning, it is recommended that the cleaning process be carried out automatically.

2.13 Lighting

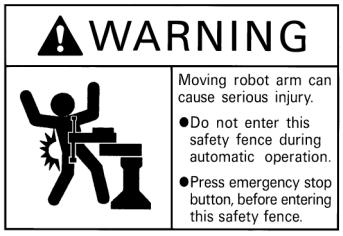
Sufficient illumination should be assured for safe robot operation.

2.14 Protection from objects thrown by the end-effector

If there is any risk of workers being injured in the event that the object being held by the end-effector is dropped or thrown by the end-effector, consider the size, weight, temperature and chemical nature of the object and take appropriate safeguards to ensure safety.

2.15 Affixing the warning label

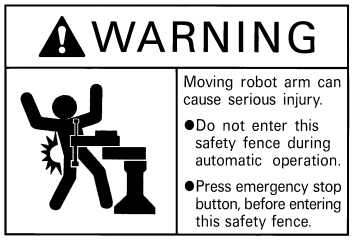
Place the warning label packaged with the robot on the exit/entrance of the safety fence or in a position where it is easy to see.



3. Precautions while robot is running

⚠ Warning

Touching the robot while it is in operation can lead to serious injury. Please ensure the following conditions are maintained and that the cautions listed from Section 3.1 onwards are followed when any work is being performed.



1) Do not enter the robot's restricted space when the robot is in operation or when the motor power is on.
2) As a precaution against malfunction, ensure that an emergency stop device is activated to cut the power to the robot motor upon entry into the robot's restricted space.
3) When it is necessary to enter the robot's restricted space to perform teaching or maintenance work while the robot is running, ensure that the steps described in Section 3.3 "Ensuring safety of workers performing jobs within the robot's restricted space" are taken.

3.1 Creation of working regulations and assuring worker adherence

When entering the robot's restricted space to perform teaching or maintenance inspections, set "working regulations" for the following items and ensure workers adhere to them.

(1) Operating procedures required to run the robot.

(2) Robot speed when performing teaching.

(3) Signaling methods to be used when more than one worker is to perform work.

(4) Steps that must be taken by the worker in the event of a malfunction, according to the contents of the malfunction.

(5) The necessary steps for checking release and safety of the malfunction status, in order to restart the robot after robot movement has been stopped due to activation of the emergency stop device

(6) Apart from the above, any steps below necessary to prevent danger from unexpected robot movement or malfunction of the robot.

1) Display of the control panel (See Section 3.2 on the following page)
2) Assuring the safety of workers performing jobs within the robot's restricted space (See Section 3.3 on the following page)
3) Maintaining worker position and stance

   Position and stance that enables the worker to confirm normal robot operation and to take immediate refuge if a malfunction occurs.

4) Implementation of measures for noise prevention
5) Signaling methods for workers of related equipment
6) Types of malfunctions and how to distinguish them

Please ensure “working regulations” are appropriate to the robot type, the place of installation and to the content of the work.

Be sure to consult the opinions of related workers, engineers at the equipment manufacturer and that of a labor safety consultant when creating these “working regulations”.

3.2 Display of operation panel

To prevent anyone other than the worker from accessing the start switch or the changeover switch by accident during operation, display something to indicate it is in operation on the operating panel or teach pendant. Take any other steps as appropriate, such as locking the cover.

3.3 Ensuring safety of workers performing jobs within the robot's restricted space

When performing jobs within the robot’s restricted space, take any of the following steps to ensure that robot operation can be stopped immediately upon a malfunction.

(1) Ensure an overseer is placed in a position outside the robot’s restricted space and one in which he/she can see all robot movements, and that he/she is devoted solely to that task.
   ① An emergency stop device should be activated immediately upon a malfunction.
   ② Do not permit anyone other than the worker engaged for that job to enter the robot’s restricted space.

(2) Ensure a worker within the robot's restricted space carries the portable emergency stop switch so he/she can press it (the robot stop button on the teach pendant) immediately if it should be necessary to do so.

3.4 Inspections before commencing work such as teaching

Before starting work such as teaching, inspect the following items, carry out any repairs immediately upon detection of a malfunction and perform any other necessary measures.

(1) Check for any damage to the sheath or cover of the external wiring or to the external devices.

(2) Check that the robot is functioning normally or not (any unusual noise or vibration during operation).

(3) Check the functioning of the emergency stop device.

(4) Check there is no leakage of air or oil from any pipes.

(5) Check there are no obstructive objects in or near the robot's restricted space.

3.5 Release of residual air pressure

Before disassembling or replacing pneumatic parts, first release any residual air pressure in the drive cylinder.

3.6 Precautions for test runs

Whenever possible, have the worker stay outside of the robot's restricted space when performing test runs.

3.7 Precautions for automatic operation

(1) At start-up

Before the robot is to be started up, first check the following items as well as setting the signals to be used and perform signaling practice with all related workers.

1) Check that there is no one inside the robot's restricted space.

2) Check that the teach pendant and tools are in their designated places.

3) Check that no lamps indicating a malfunction on the robot or related equipment are lit.

(2) Check that the display lamp indicating automatic operation is lit during automatic operation.

(3) Steps to be taken when a malfunction occurs

Should a malfunction occur with the robot or related equipment and it is necessary to enter the robot's restricted space to perform emergency maintenance, stop the robot's operation by activating the emergency stop device. Take any necessary steps such as placing a display on the starter switch to indicate work is in progress to prevent anyone from accessing the robot.

3.8 Precautions in repairs

(1) Do not perform repairs outside of the designated range.

(2) Under no circumstances should the interlock mechanism be removed.

(3) When opening the robot controller's cover for battery replacement or any other reasons, always turn the robot controller power off and disconnect the power cable.

(4) Use only spare tools authorized by DENSO.

4. Daily and periodical inspections

(1) Be sure to perform daily and periodical inspections. Before starting jobs, always check that there is no problem with the robot and related equipment. If any problems are found, take any necessary measures to correct them.

(2) When carrying out periodical inspections or any repairs, maintain records and keep them for at least 3 years.

5. Management of floppy disks

(1) Carefully handle and store the "Initial settings" floppy disks packaged with the robot, which store special data exclusively prepared for your robot.

(2) After finishing teaching or making any changes, always save the programs and data onto floppy disks.

Making back-ups will help you recover if data stored in the robot controller is lost due to the expired life of the back-up battery.

(3) Write the names of each of the floppy disks used for storing task programs to prevent incorrect disks from loading into the robot controller.

(4) Store the floppy disks where they will not be exposed to dust, humidity and magnetic field, which could corrupt the disks or data stored on them.

Contents

Chapter 4 Other Advanced Features

Commands Listed in Alphabetical Order

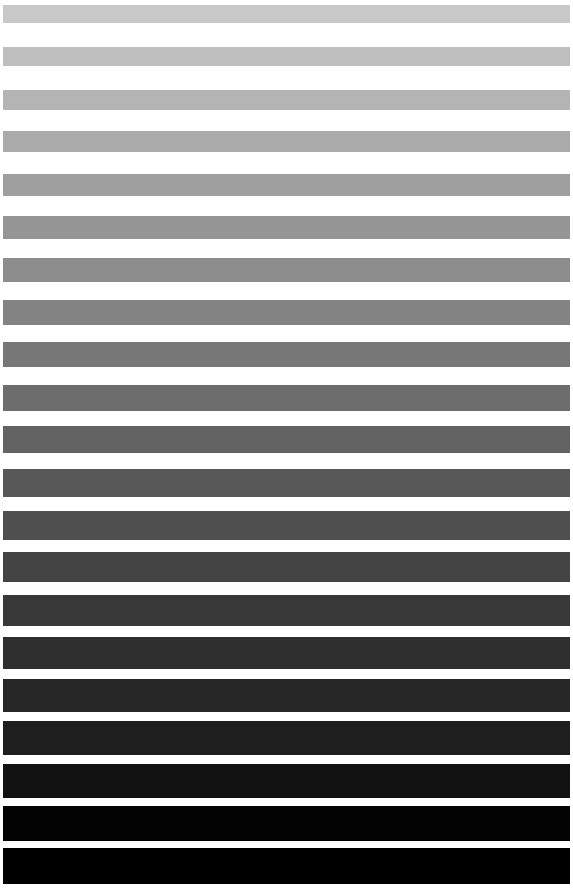
Commands	Functions	Refer to:
A		
ACCEL	Specifies the internal composite acceleration/deceleration of joints included in a currently held arm group.	3-33
arrange_button_pos	Specifies the button arrangement including locations and sizes on a customized operation screen.	3-116
ARRIVE	Defines the motion ratio relative to the programmed full travel distance to the target point in order to make the current program stand by to execute the next step until the robot reaches the defined motion ratio.	3-30
C		
change_bCap	Edits a caption for a specified button.	3-100
change_pCap	Edits a caption for a specified page.	3-101
ClearSrvMonitor	Initializes the pointer of data obtained by the single-joint servo data monitor function.	3-88
com_discom	Releases the RS-232C port from binary transmission.	4-16
com_encom	Enables the RS-232C port only for binary transmission.	4-15
com_state	Gets the status of RS-232C port.	4-17
CURACC	Gets the current internal composite acceleration of joints included in a currently held arm group.	3-39
CURDEC	Gets the current internal composite deceleration of joints included in a currently held arm group.	3-41
CUREXJ	Gets the current angle of an extended-joint into a floating-point variable.	3-28
CURJACC	Gets the current internal acceleration of individual joints included in a currently held arm group.	3-40
CURJDEC	Gets the current internal deceleration of individual joints included in a currently held arm group.	3-42
CURJSPD	Gets the current internal speed of individual joints included in a currently held arm group.	3-38
CURSPD	Gets the current internal composite speed of joints included in a currently held arm group.	3-37
D		
DECEL	Specifies the internal composite deceleration of joints involved in a currently held arm group.	3-35
DESTEXJ	Gets the target position of an extended-joint invoked by the current motion command into a floating-point variable. If the robot is on halt, this command will get the current position (commanded value).	3-29
disp_page	Displays a specified page of a TP operation screen.	3-102
DRIVE	Carries out the relative motion of each joint.	3-24
DRIVEA	Carries out the absolute motion of each joint.	3-25
G		
GetJntData	Gets the internal servo data of a specified joint.	3-92

Commands	Functions	Refer to:
GetSrvData	Gets the internal servo data of robot joints.	3-91
GIVEARM	Releases the currently held arm group.	3-23
I		
inputb	Inputs a single byte of data from the RS-232C port.	4-12
INTERRUPT ON/OFF	Interrupts the current robot motion.	3-43
J		
JACCEL	Specifies the internal acceleration and deceleration of individual joints included in a currently held arm group.	3-34
JDECEL	Specifies the internal deceleration ratio of individual joints included in a currently held arm group.	3-36
JSPEED	Specifies the internal speed of individual joints included in a currently held arm group.	3-32
L		
linputb	Inputs multiple bytes of data from the RS-232C port.	4-14
lprintb	Outputs multiple bytes of data to the RS-232C port.	4-13
M		
make_b_samp1	This is a sample program (Pattern 1) for creating your own operation screen.	3-112
make_b_samp2	This is a sample program (Pattern 2) for creating your own operation screen.	3-113
MotionComp	Judges whether execution of running motion commands is complete.	3-90
MotionSkip	Aborts running motion commands.	3-89
MOVE	Moves the robot flange to the specified coordinates.	3-26
	If specified with an EX option (relative motion of extended-joints) or EXA option (absolute motion of extended-joints), the MOVE can move both the robot flange and the extended-joints synchronously.	
mvResetPulseWidthJnt	Resets the encoder pulse count for an allowable positioning error for a specified extended-joint to the default.	3-47
mvSetPulseWidthJnt	Sets the encoder pulse count for an allowable positioning error for a specified extended-joint.	3-46
P		
POSCLR	Forcibly restores the current position of a joint to 0 mm or 0 degree.	3-45
printb	Outputs a single byte of data to the RS-232C port.	4-11
R		
ResetCurLmt	Releases the drive current limit set for a specified joint motor.	3-50
ResetCycloidJnt	Cancels the cycloid mode set for a specified extended-joint and restores the normal mode.	3-49
ResetEralw	Resets the positioning error allowance of a specified joint to the initial value.	3-51

Commands	Functions	Refer to:
S		
Select_panel	This is a sample program for switching the customized operation screens on the teach pendant.	3-111
set_button_param	Specifies button attributes such as color and shape on a customized operation screen.	3-114
set_button_size	Specifies the button arrangement including locations and sizes on a customized operation screen.	3-115
SetCycloidJnt	Enters a specified extended-joint into the cycloid mode where the controller suppresses the peak of overshoot and residual oscillation that would occur in an end motion.	3-48
SetMonitorCond	Sets the monitoring conditions for single-joint servo data monitor.	3-85
set_button	Sets button parameters.	3-96
set_page	Sets page parameters.	3-98
SPEED	Specifies the internal composite speed of joints included in a currently held arm group.	3-31
StartSrvMonitor	Starts monitoring single-joint servo data.	3-86
StopSrvMonitor	Stops monitoring single-joint servo data.	3-87
T		
TAKEARM	Gets an arm group. Upon execution of this statement, the programmed speed, acceleration and deceleration will be set to 100. If the gotten arm group includes any robot joint, this statement restores the tool coordinates and work coordinates to the origin.	3-20

Chapter 1

Overview



This chapter outlines the extended-joint support system and newly supported functions in Main Software Version 1.5 and Version 1.6.

Chapter 1 Overview

The extended-joint support system, which is provided as an option for the H∗-D and VS-D series, enables the robot controller to control up to two extended-joints in addition to the conventional robot joints. Mainly for supporting the extended-joint related functions, Main Software is upgraded to Version 1.5 (for H∗-D and XYC-4D series) and Version 1.6 (for V∗-D series).

■ Configuration of extended-joint support system

The extended-joint support system is different from the conventional robot system in the following points:

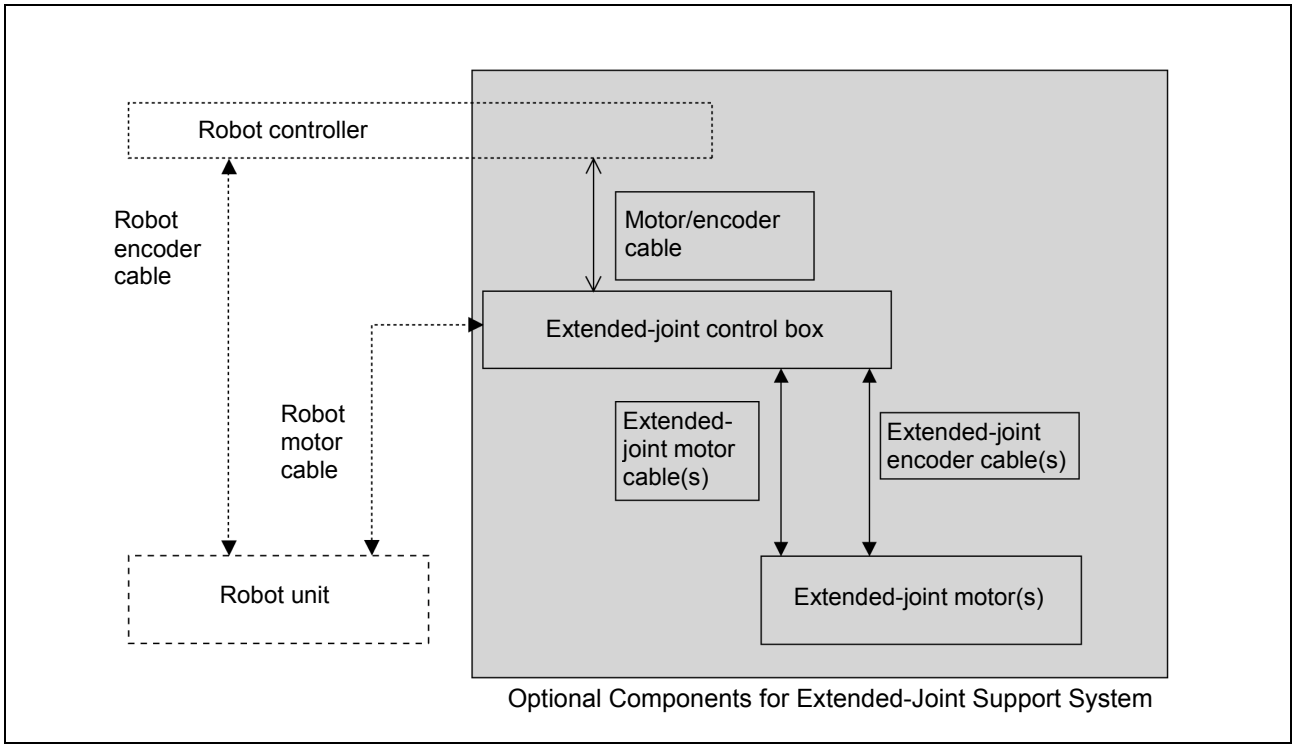
(1) Extended-joint support robot controller which is equipped with an IPM board (factory option) suited to extended-joint motors to be connected

(2) Extended-joint control box (option) that distributes signal and power cables to extended-joint motors & their encoders

(3) Extended-joint motors (option) which are equipped with bus-line encoders (Up to 2 motors, motor type selectable)

(4) Motor/encoder cable (option) between the robot controller and extended-joint control box

(5) Extended-joint motor cables and encoder cables (both option) between the extended-joint control box and motors



Optional Components for Extended-Joint Support System

■ Extended-joint related functions newly added

(1) Extended-joint function

The extended-joint function allows you to control extra joints independently of robot joints through the standard interface of the robot controller (NetwoRC).

(2) Setting extended-joint parameters

To use extended-joints, you need to set extended-joint path parameters and extended-joint servo parameters by using the teach pendant. The former is for motion definitions (including speed, acceleration, and range of motion); the latter is for setting the gain and others of the extended-joint servo system.

After setting the motion conditions of extended-joints and checking the motion of the optional mechanism connected to the extended-joint motors, you need to do gain tuning for the servo system. There are the following two types of tuning methods:

- Auto gain tuning

The robot controller performs acceleration/deceleration operation of the extended-joints according to the default pattern preset in the controller. Based on the motion of the extended-joints in that operation, the controller will estimate the inertia of payload and set the appropriate gain automatically. With this method, you may easily do gain tuning of the extended-joint motors.

- Manual gain tuning

The monitor function of the single-joint servo data monitors the motor speed control value, current motor speed, motor angle deviation, and torque control value. According to the monitored results, you may adjust the gain and torque control filter parameters for optimizing the motion of the extended-joints.

(3) Customizing TP operation screens

Main Software version 1.5 or later allows you to easily customize operation screens on the teach pendant for facilitating control of the robot by the robot controller in stand-alone mode.

■ Other advanced features

In addition to the new extended-joint related functions, Main Software versions 1.5 and 1.6 support the following:

No.	Items	Description
1	Vision control	- Window editing functions enhanced, which allow you to easily handle windows with the teach pendant. - Image analysis features enhanced, which enable model search, labeling, area/center of gravity/major axis angle, and edge finding without the aid of programs.
2	Serial binary transmission	In addition to the conventional ASCII code transmission via the RS-232C port, the new robot controller may support byte-controlled binary data transmission, increasing the types of connectable communications devices.
3	WINCAPSII	The following functions are newly added to WINCAPSII Version 1.5: - Choice of the desired project at the start of WINCAPSII. - "Save Project As" newly added to System Manager. - Import of macro definition file in Variable Manager, DIO Manager, and Vision Manager. - "Permission of CALSET transmission" in Arm Manager. (Only when it is selected, CALSET data may be transmitted to the robot controller.) - Use of hardware accelerator selectable in Arm Manager. - Import and export functions added in Log Manager. (Importing "Servo Joint Log" in CSV format, plotting its graph, and saving it)
4	Operator precedence	The precedence of arithmetic operators, logical operators, and relational operators is defined and one program line may contain those operators together.
5	Field network error indication parameter added	The "10: FieldNetwork ErrDisplay" parameter allows you to choose whether a network error will display "every time" it occurs or at the "first time." (This parameter takes effect in the DeviceNet masters and slaves and the PROFIBUS slaves.
6	Error codes added or modified	Error codes for new extended-joint functions and related ones.

Chapter 2

Robot Components of Extended-Joint Support System

This chapter describes the components and specifications of the extended-joint support system.

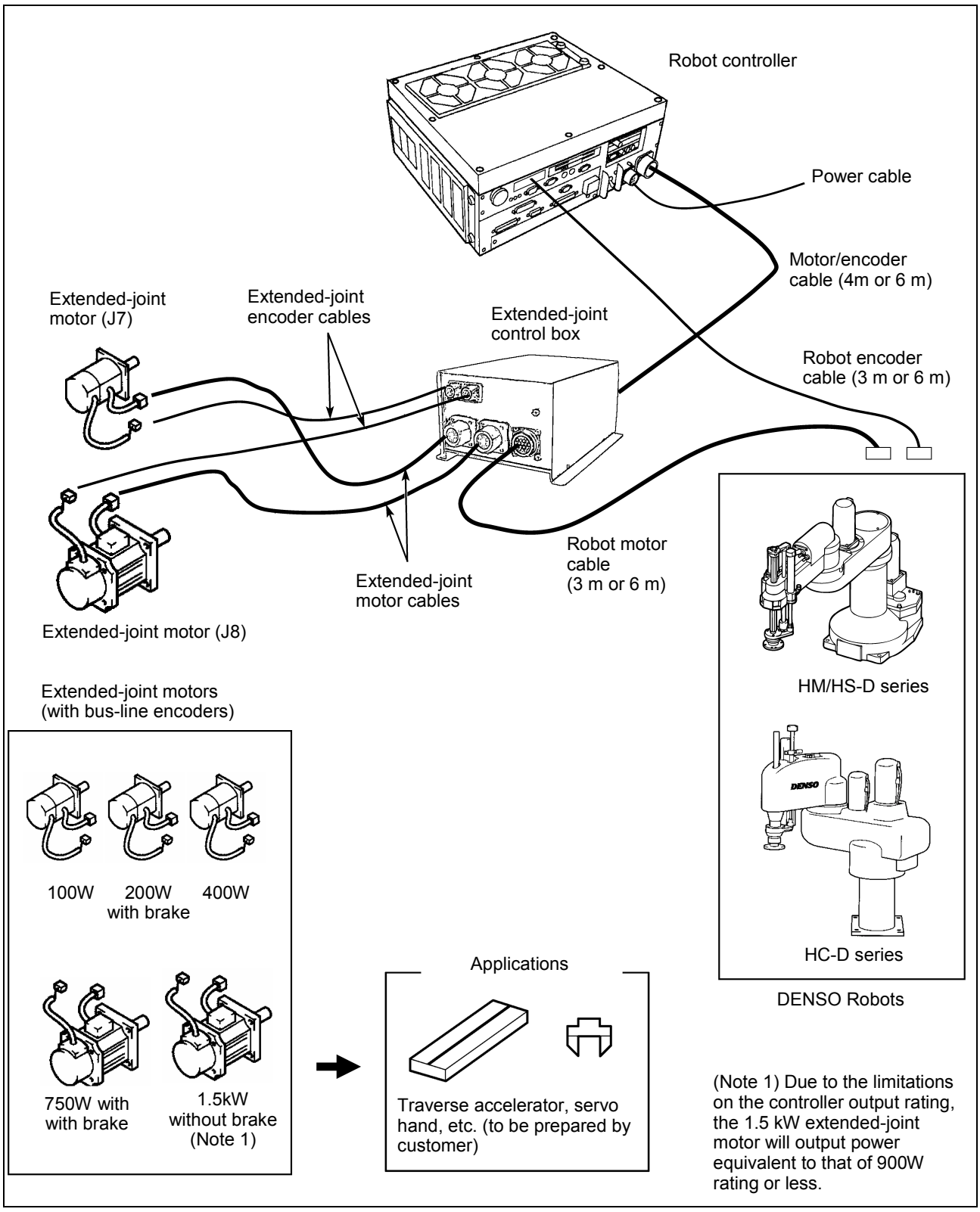
Chapter 2 Robot Components of Extended-Joint Support System

2.1 Configuration of Extended-Joint Support System

You may configure an extended-joint support system (see the illustration given on the next page) by using a robot controller designed for extended-joint motors to be connected and connecting the options (DENSO genuine components) listed below.

- Extended-joint control box
- Extended-joint motors equipped with bus-line encoders
  (Up to 2 motors, motor type selectable)
- Motor/encoder cable (between the robot controller and extended-joint control box)
- Extended-joint motor cables (between the extended-joint control box and extended-joint motors)
- Extended-joint encoder cables (between the extended-joint control box and extended-joint motors)

Extended-Joint Support System



2.2 Extended-Joint Support Robot Controller

(1) Robot controller models

RC5-EAH4A: Robot controller supporting the extended-joints, for MH/HS-D series

RC5-EAHC4A: Robot controller supporting the extended-joints, for HC-D series

RC5-EAVS6A: Robot controller supporting the extended-joints, for VS-D series

(2) IPM boards that enable robot controllers to support extended-joints

When shipped from the factory, an IPM board suited to extended-joint motors to be connected will be built into a robot controller.

Note: On the top of the robot controller is a "Controller Parameter Table" which is labeled "Extended-joint motors and IPM boards" as shown below. Check the label before connecting extended-joint motors.

AXIS	7					
MOTOR	1. 5 KW	750 W	400 W	200 W	100 W	50 W
IPM	L	M	S		SS	

AXIS	8					
MOTOR	1. 5 KW	750 W	400 W	200 W	100 W	50 W
IPM	L	M	S		SS	

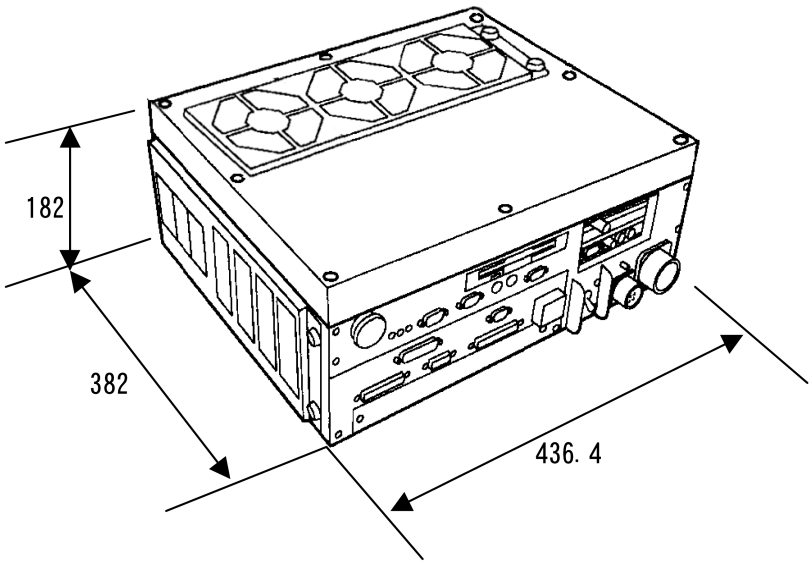
Extended-joint Motors and IPM Boards

(3) Robot controller specifications

The table below lists the specifications of the robot controller capable of controlling the extended-joints.

Items	Specifications
Applicable robot	H*-D series (Extended-joint support system) VS-D series (Extended-joint support system)
Robot controller models	RC5-EAH4A, RC5-EAHC4A, and RC5-EAVS6A
Control system	PTP, CP 3-dimensional linear, and 3-dimensional circular (PTP available only for extended-joints)
No. of controllable axes	H*-D series: Up to 6 axes simultaneously (4 axes for robot and 2 axes for extended-joints) VS-D series: Up to 8 axes simultaneously (6 axes for robot and 2 axes for extended-joints)
Drive system	Full-digital AC servo loop for all axes (joints)
Memory capacity	1.25 MB (Equivalent to 5,000 steps and 13,000 points)
Programming language	DENSO robot language (conforming to SLIM)
Teaching system	1) Remote teaching 2) Numerical input (MDI) 3) Direct teaching

Items		Specifications
External signals (I/O)	Input	20 lines available for user (12 for sequencer, 8 for hand input) and 36 lines reserved by System
	Output	32 lines available for user (24 for sequencer, 8 for hand input) and 33 lines reserved by System
External ports		RS-232C: 1 line, Ethernet: 1 line (option)
Timer function		0.02 to 10 seconds (in increment of 1/60 sec.)
Self-diagnosis function		Overrun, servo loop error, memory error, illegal entry, etc.
Error display		Error codes will display on the external I/O or the operating panel (option). Error messages will display in English on the teach pendant (option).
Power source		3-phase, 200 VAC-15% to 230 VAC+10%, 50/60 Hz, 3.3 kVA
Environmental conditions (in operation)		Temperature: 0 to 40ºC Relative Humidity: 90% RH or less (No condensation allowed)
Degree of protection		IP20
Weight		Approx. 19 kg (excluding cables)



Outer Dimensions of Extended-Joint Support Robot Controller

2.3 Parts Codes of Optional Components for the Extended-Joint Support System

For extended-joint support system, the following components are available as options. Purchase them according to your needs.

For components other than extended-joint related ones, refer to the "INSTALLATION & MAINTENANCE GUIDE."

No.	Components	Parts No.	Remarks
1	Extended-joint motor, 100W (With brake)	410622-1510	
2	Extended-joint motor, 200W (With brake)	410622-1520	
3	Extended-joint motor, 400W (With brake)	410622-1530	
4	Extended-joint motor, 750W (With brake)	410622-1540	
5	Extended-joint motor, 1.5 kW (Without brake) (Note 1)	410622-1550	
6	Extended-joint control box	410181-0070	RC5-EABOX-A
7	Extended-joint cable set (Between extended-joint control box and motor)	410149-0660	Including Nos. 7-1 and 7-2.
7-1	Extended-joint encoder cable (Standard, 3m)	410141-2210	
7-2	Extended-joint motor cable (Standard, 3m)	410141-2130	For 100W, 200W or 400W motor
8	Extended-joint cable set (Between extended-joint control box and motor)	410149-0670	Including Nos. 8-1 and 8-2.
8-1	Extended-joint encoder cable (Standard, 3m)	410141-2210	
8-2	Extended-joint motor cable (Standard, 3m)	410141-2150	For 750W or 1.5 kW Motor (Note 1)
9	Extended-joint cable set (Between extended-joint control box and motor)	410149-0680	Including Nos. 9-1 and 9-2.
9-1	Extended-joint encoder cable (Standard, 6m)	410141-2220	
9-2	Extended-joint motor cable (Standard, 6m)	410141-2140	For 100W, 200W or 400W motor
10	Extended-joint cable set (Between extended-joint control box and motor)	410149-0690	Including Nos. 10-1 and 10-2.
10-1	Extended-joint encoder cable (Standard, 6m)	410141-2220	
10-2	Extended-joint motor cable (Standard, 6m)	410141-2160	For 750W or 1.5kW motor (Note 1)
11	Extended-joint cable set (Between extended-joint control box and motor)	410149-0700	Including Nos. 11-1 and 11-2.
11-1	Extended-joint encoder cable (Enforced, 3m)	410141-2230	
11-2	Extended-joint motor cable (Enforced, 3m)	410141-2170	For 100W, 200W or 400W motor

No.	Components	Parts No.	Remarks
12	Extended-joint cable set (Between extended-joint control box and motor)	410149-0710	Including Nos. 12-1 and 12-2.
12-1	Extended-joint encoder cable (Enforced, 3m)	410141-2230	
12-2	Extended-joint motor cable (Enforced, 3m)	410141-2190	For 750W or 1.5 kW motor (Note 1)
13	Extended-joint cable set (Between extended-joint control box and motor)	410149-0720	Including Nos. 13-1 and 13-2.
13-1	Extended-joint encoder cable (Enforced, 6m)	410141-2240	
13-2	Extended-joint motor cable (Enforced, 6m)	410141-2180	For 100W, 200W or 400W motor
14	Extended-joint cable set (Between extended-joint control box and motor)	410149-0730	Including Nos. 14-1 and 14-2.
14-1	Extended-joint encoder cable (Enforced, 6m)	410141-2240	
14-2	Extended-joint motor cable (Enforced, 6m)	410141-2200	For 750W or 1.5 kW motor (Note 1)
15	Motor/encoder cable (4m)	410141-1550	Between robot controller and extended-joint control box
16	Motor/encoder cable (6m)	410141-1560	Between robot controller and extended-joint control box

(Note 1) Due to the limitations on the controller output rating, the 1.5 kW extended-joint motor will output power equivalent to that of 900W rating or less.

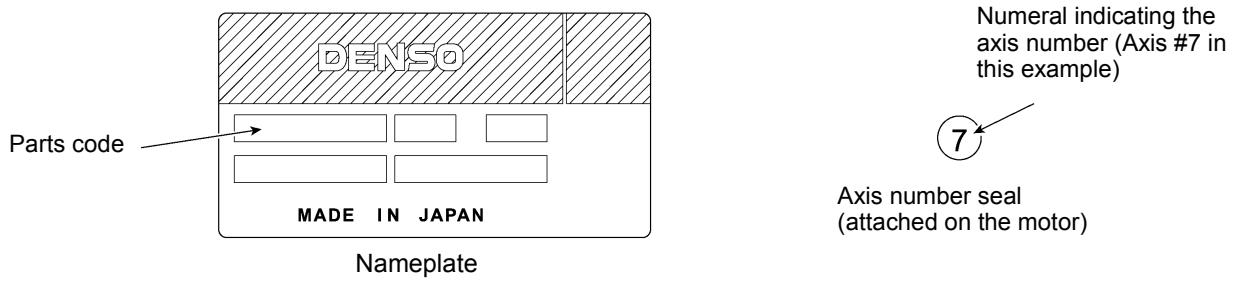
(Note 2) To connect the robot unit, use the robot motor cable and robot encoder cable that come with the robot unit.

2.4 Detailed Specifications of Extended-Joint Related Options

2.4.1 Extended-Joint Motors

The following five types of servomotors are available as options for the extended-joint use.

On those servomotors are nameplates and axis number seals as illustrated below. Before using those motors, be sure to check the parts codes and axis numbers to connect motors to correct axes specified on axis number seals.

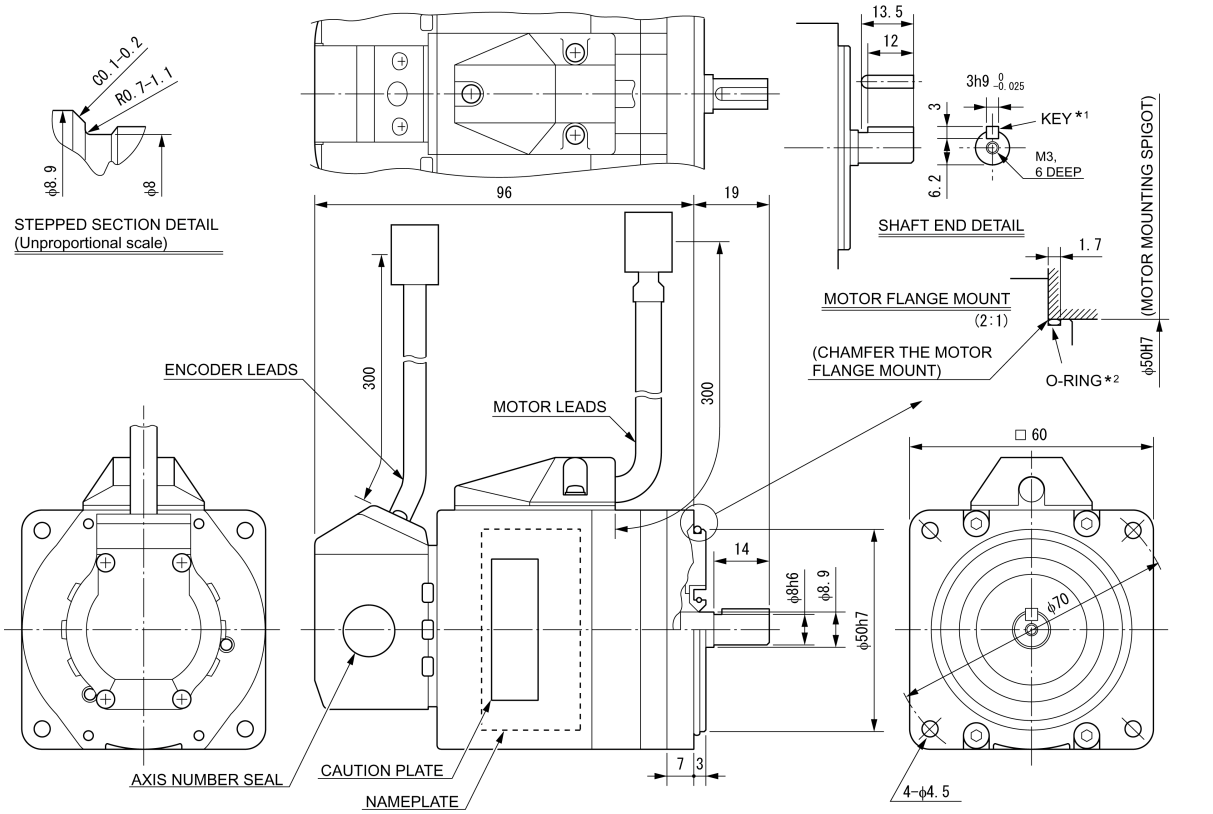


Parts code	Rated output power (W)	Rated revolutions (rpm)	With/without brake	Model
410622-1510	100	3,000	With	MQMA012T2V2
410622-1520	200	3,000	With	MQMA022T3V2
410622-1530	400	3,000	With	MQMA042T3V2
410622-1540	750	3,000	With	MQM082T2V2
410622-1550	1500 (Note 1)	3,000	Without	MQM152T2U2

(Note 1) Due to the limitations on the controller output rating, the 1.5 kW extended-joint motor will output power equivalent to that of 900W rating or less.

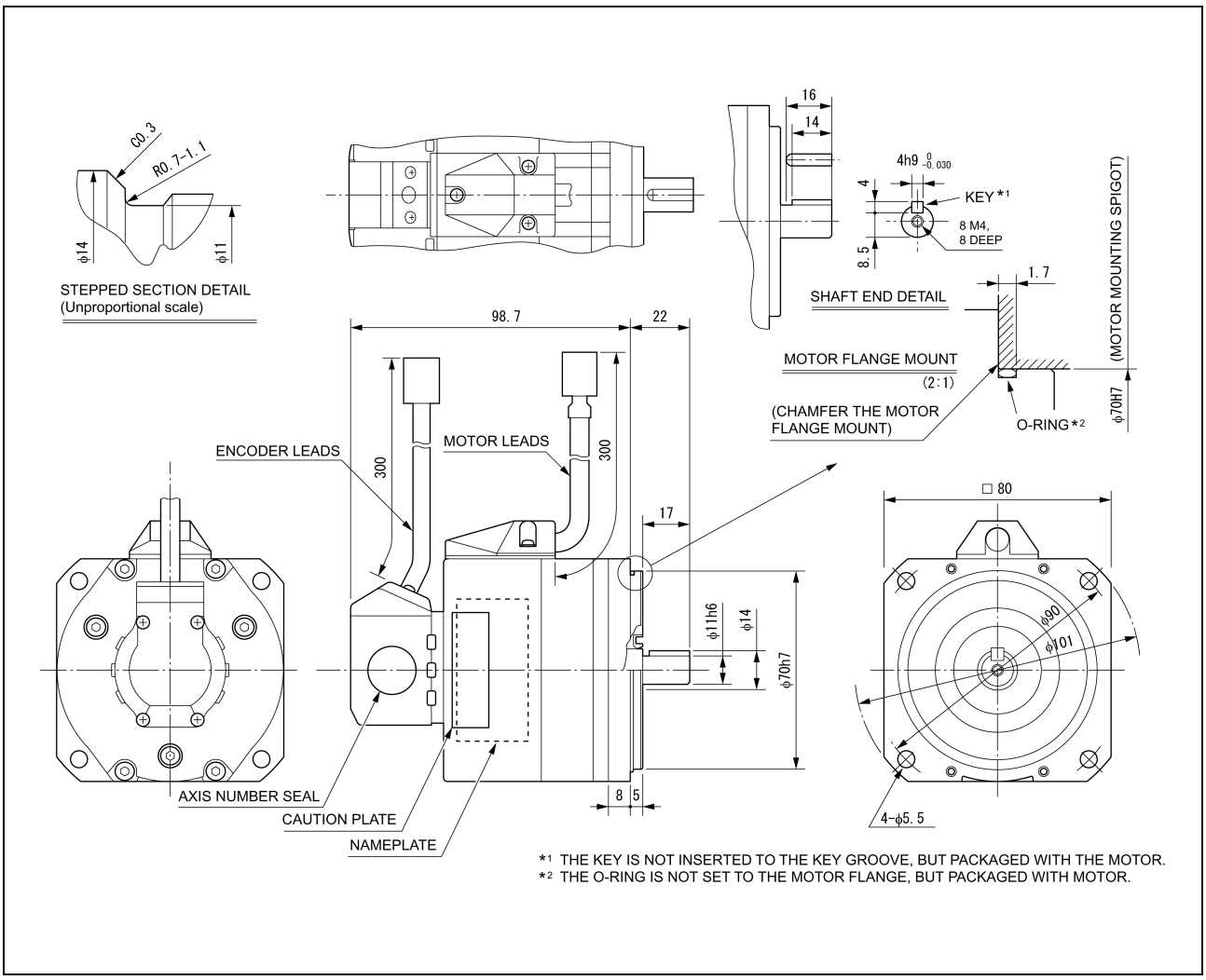
2.4.1.1 External dimensions of extended-joint motors

■ 410622-1510 (100W, with brake) Model: MQMA012T2V2



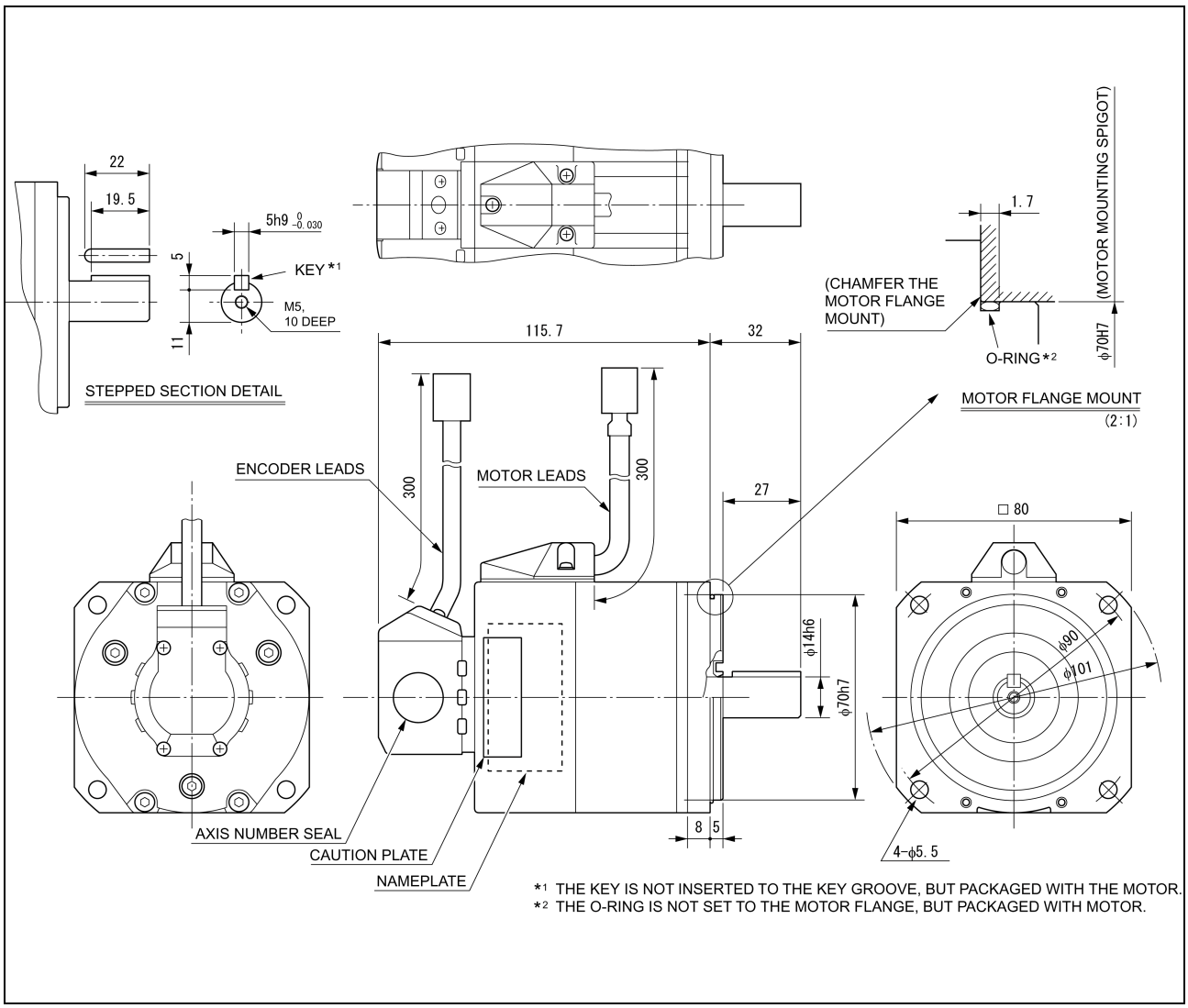
*1 THE KEY IS NOT INSERTED TO THE KEY GROOVE, BUT PACKAGED WITH THE MOTOR.
*2 THE O-RING IS NOT SET TO THE MOTOR FLANGE, BUT PACKAGED WITH MOTOR.

■ 410622-1520 (200W, with brake) Model: MQMA022T3V2



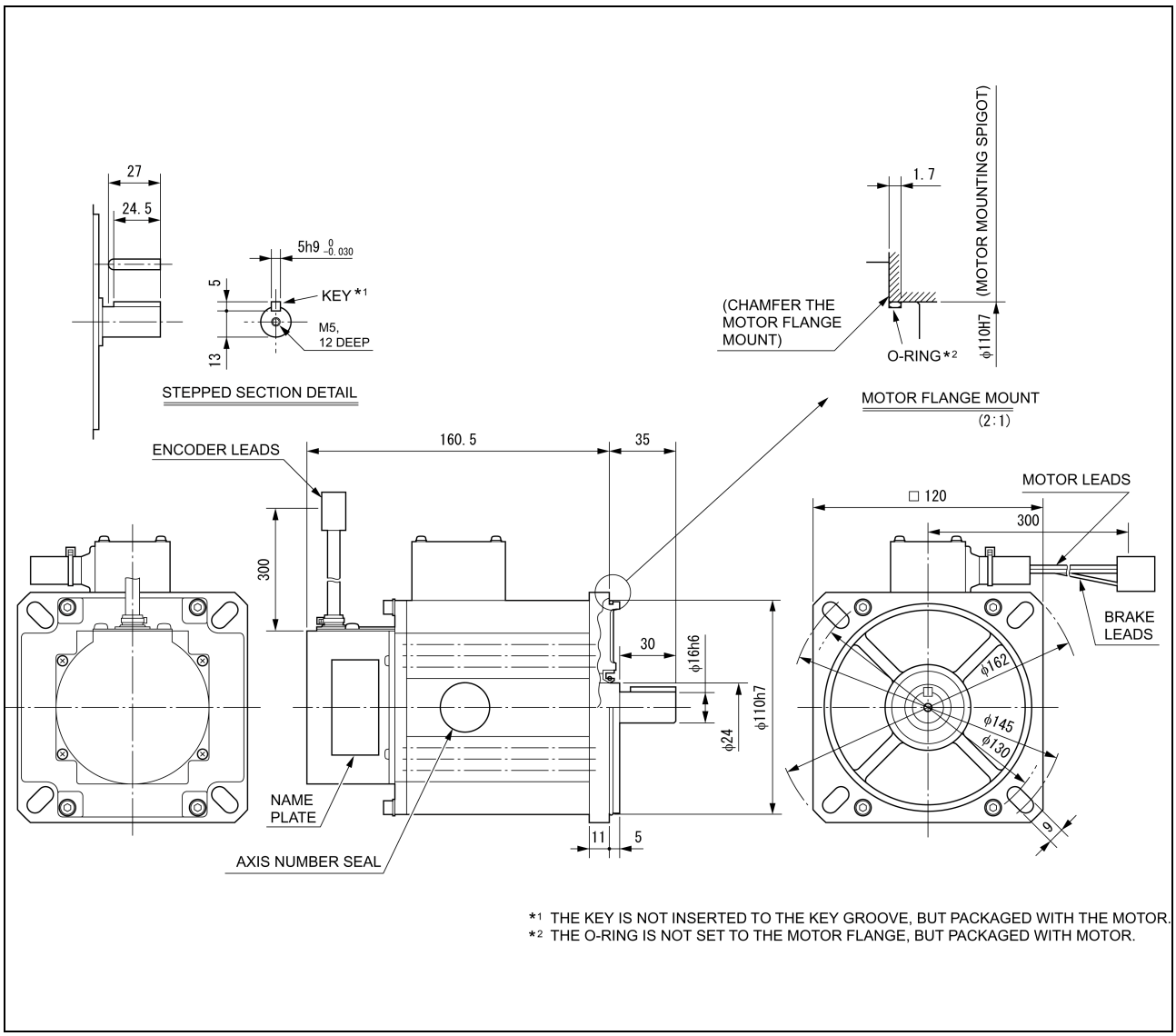
*1 THE KEY IS NOT INSERTED TO THE KEY GROOVE, BUT PACKAGED WITH THE MOTOR.
*2 THE O-RING IS NOT SET TO THE MOTOR FLANGE, BUT PACKAGED WITH MOTOR.

■ 410622-1530 (400W, with brake) Model: MQMA042T3V2



*1 THE KEY IS NOT INSERTED TO THE KEY GROOVE, BUT PACKAGED WITH THE MOTOR.
*2 THE O-RING IS NOT SET TO THE MOTOR FLANGE, BUT PACKAGED WITH MOTOR.

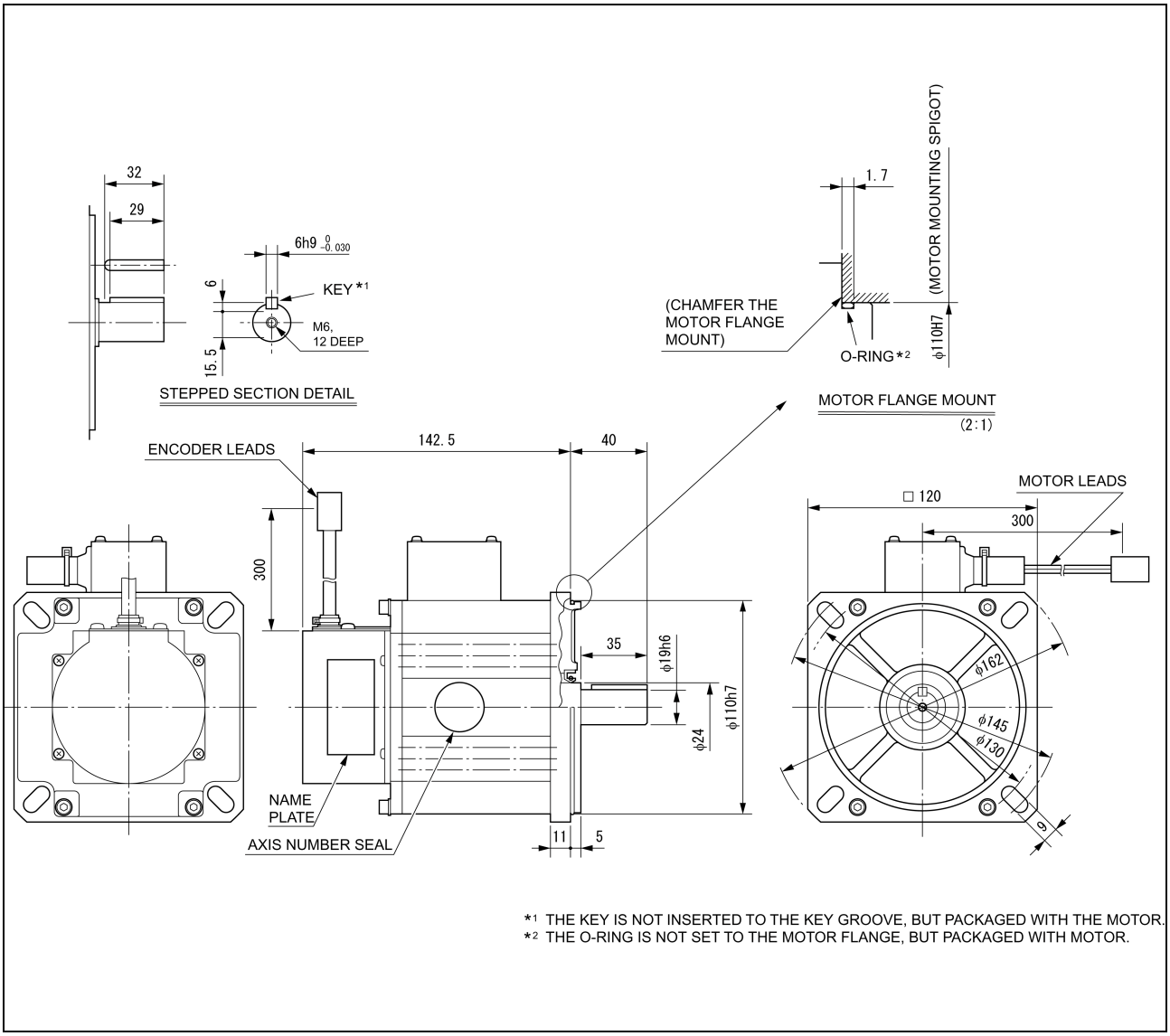
■ 410622-1540 (750W, with brake) Model: MQM082T2V2



*1 THE KEY IS NOT INSERTED TO THE KEY GROOVE, BUT PACKAGED WITH THE MOTOR.
*2 THE O-RING IS NOT SET TO THE MOTOR FLANGE, BUT PACKAGED WITH MOTOR.

■ 410622-1550 (1.5kW, without brake) Model: MQM152T2U2

Note: Due to the limitations on the controller output rating, the 1.5kW extended-joint motor will output power equivalent to that of 900W rating or less.



*1 THE KEY IS NOT INSERTED TO THE KEY GROOVE, BUT PACKAGED WITH THE MOTOR.
*2 THE O-RING IS NOT SET TO THE MOTOR FLANGE, BUT PACKAGED WITH MOTOR.

2.4.1.2 Specifications of Extended-Joint Motors

(1) General specifications

Items (Unit)				410622-1510 (100W, with brake)	410622-1520 (200W, with brake)	410622-1530 (400W, with brake)	410622-1540 (750W, with brake)	410622-1550 (1.5 kW, without brake) (Note 1)
Rated output (W)				100	200	400	750	900 (Note 1)
Rated rotation speed (rpm)				3000	3000	3000	3000	3000
Maximum rotation speed (rpm)				5000	5000	5000	4500	4500
Rated output torque (N*m)				0.32	0.64	1.3	2.4	2.9 (Note 1)
Maximum peak output torque (N*m)				0.95	1.91	3.82	6.9	9.5 (Note 1)
Rated input current (A) rms, typical				1.0	1.6	2.5	4.3	5.6 (Note 1)
Rotor Inertia ($x10^{-4}$ kg*m^2)				0.12	0.42	0.72	2.85	2.82
Maximum rush current (A) (0 to peak) typical				4.3	6.8	10.5	18.9	43.4
Allowable stress or load (N)	In assembly or installation (Note 2)	Radial stress		147	392	392	686	686
		Thrust stress	A-direction	88	147	147	294	294
			B-direction	117.6	196	196	392	392
	In Operation	Radial load		98	284.2	303.8	441	392
		Thrust load		58.8	98	98	147	147
Mass (kg)				0.90	1.6	2.2	5.4	4.8
Recommended load-inertia moment ratio				30 times or less of the servomotor inertia moment (When the system requires higher response, decrease the load inertia moment)				
Environmental conditions		Heat resistance (Allowable temperature)		Operation: 0 to +40ºC Storage: -20ºC to +80ºC				
		Humidity resistance (Allowable relative humidity)		90% or less RH (Without condensation)				
		Vibration resistance		49m/s^2 max. (8 hours, 20 Hz to 3 kHz sweep in each of X, Y, and Z axes. 24.5 m/s^2 max. when the motor is stopped.) 24.5 m/s^2 max. at the resonance frequency (10,000,000 times in each of X, Y, and Z axes)				
		Installation orientation		Motor can be installed in either the horizontal or vertical direction.				

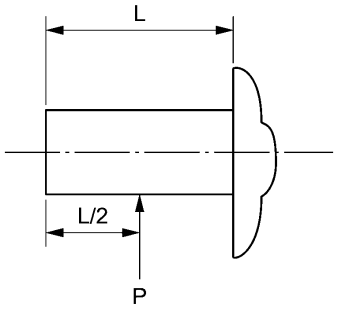
Items (Unit)		410622-1510 (100W, with brake)	410622-1520 (200W, with brake)	410622-1530 (400W, with brake)	410622-1540 (750W, with brake)	410622-1550 (1.5 kW, without brake) (Note 1)
Specification of brake (Note 3)	Static friction torque (N*m)	0.29 min.	0.64 min.	1.27 min.	2.45 min.	Without brake
	Armature pull time (Typical) (ms)	50 max.	50 max.	50 max.	50 max.	
	Armature release time (Typical) (ms) (Note 4)	15 max.	20 max.	20 max.	20 max.	
	Release voltage (VDC)	1 min.	1 min.	1 min.	1 min.	
	Rated exciting voltage (VDC)	24 ±2.4	24 ±2.4	24 ±2.4	24 ±2.4	
	Rated exciting current (Amp. DC) (Cold start at 24 VDC typical)	0.29 max.	0.23 max.	0.27 max.	0.86 max.	
	Allowable work energy for every braking (J) (Typical value)	137	196	196	392	
	Allowable gross work load (J) (Typical)	44.1×10^3	147×10^3	147×10^3	490×10^3	

(Note 1) Due to the limitations on the controller output rating, the 1.5 kW extended-joint motor will output power equivalent to that of 900W rating or less.

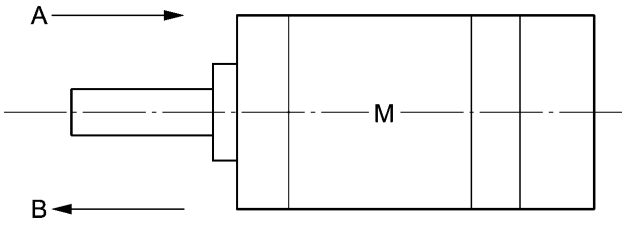
(Note that the values of each item of this specification are equivalent to those of the 900W servomotor.)

(Note 2) The radial stress (P) position and thrust stress direction in assembling or installing of the motor are shown below.

Radial stress (P) position



Thrust stress direction



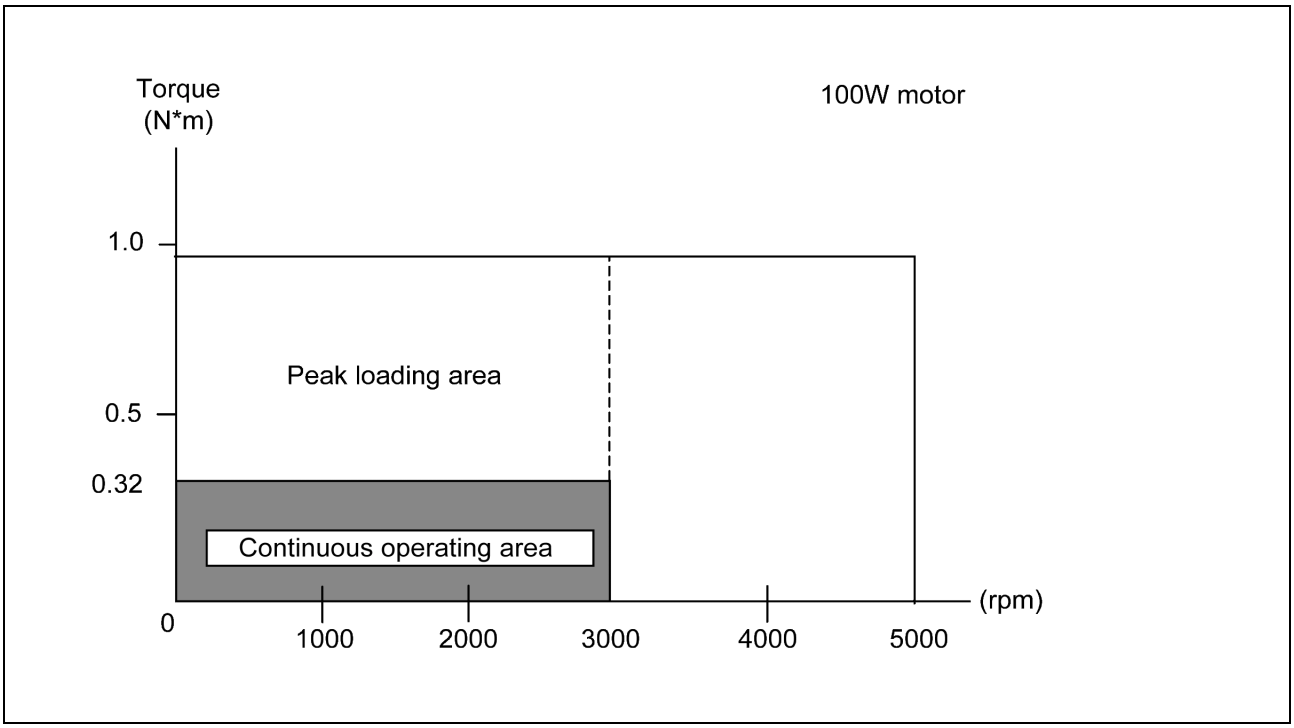
(Note 3) The brake backlash angle at the factory shipment is within ± 1º.

(Note 4) The armature release time refers to the time length from when the braking current is cut off until no DC current flows across the surge absorber coil used as dummy load.

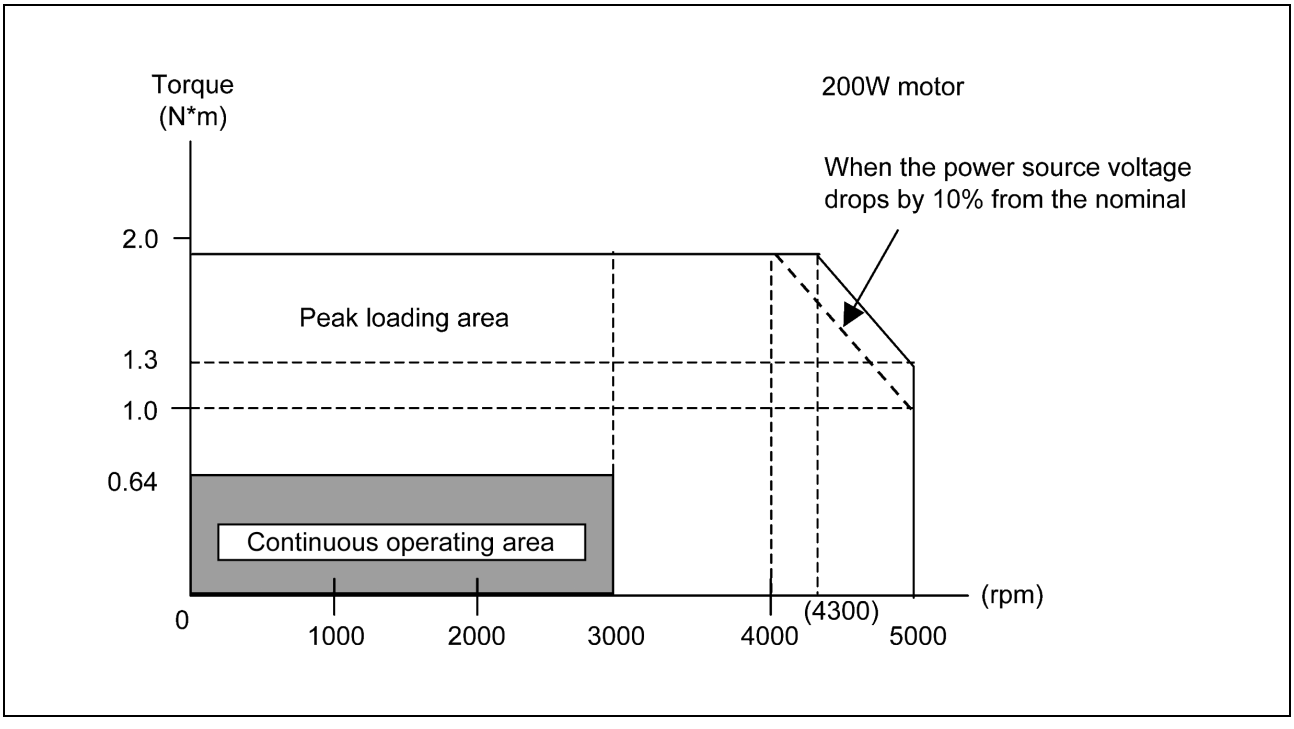
(2) S vs. T characteristic curves of extended-joint motors (Revolution speed vs. Torque characteristic)

The torque characteristics of extended-joint motors are shown below in typical values.

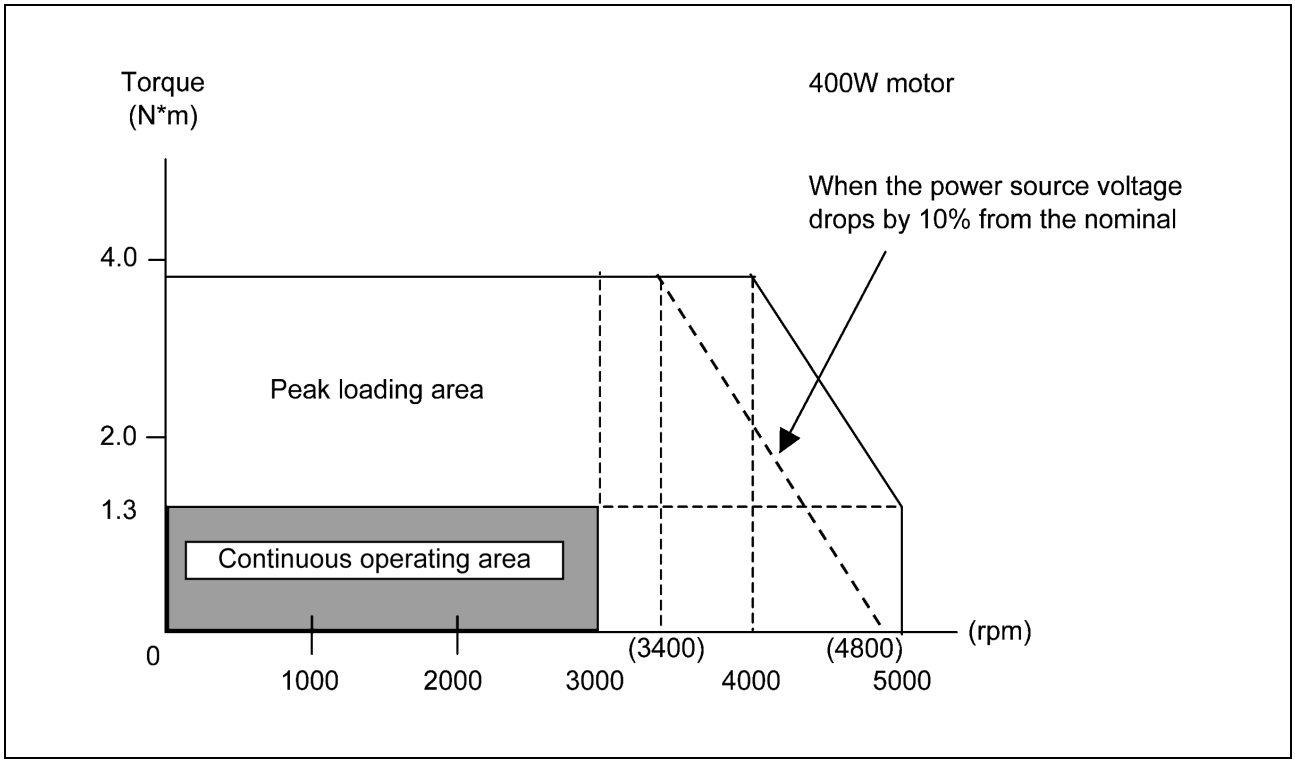
■ 410622-1510 (100W, with brake)



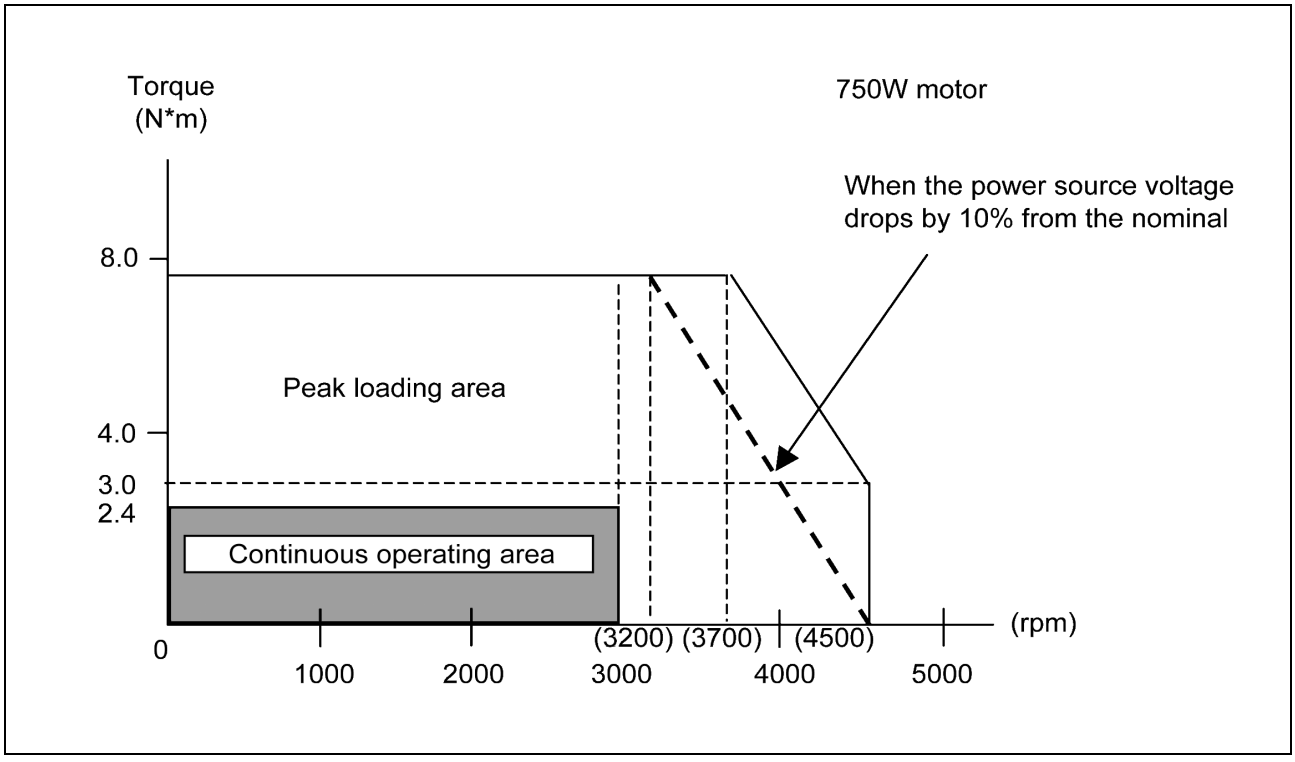
■ 410622-1520 (200W, with brake)



■ 410622-1530 (400W, with brake)

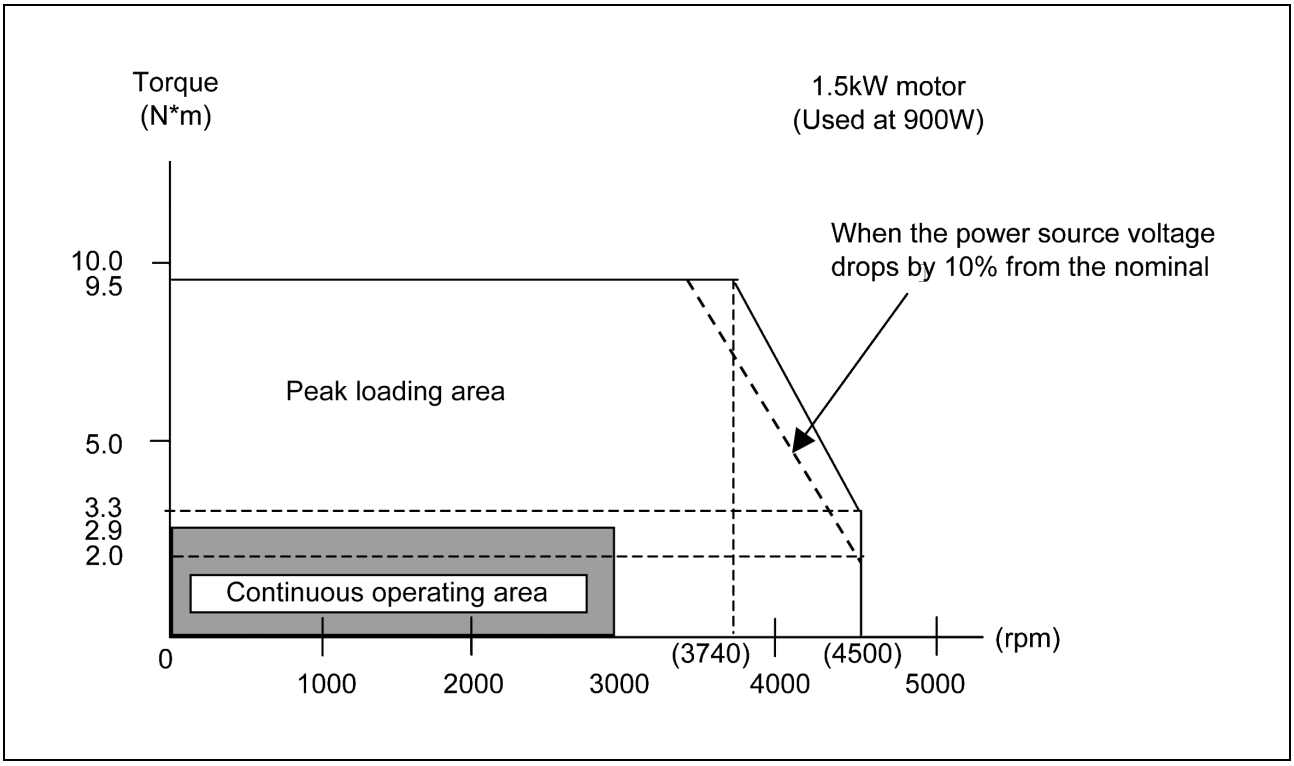


■ 410622-1540 (750W, with brake)



■ **410622-1550 (1.5kW, without brake)**

Note: Due to the limitations on the controller output rating, the 1.5 kW extended-joint motor will output power equivalent to that of 900W rating or less.



2.4.1.3 Safety rules for using extended-joint motors

Note: This section gives important points only for handling extended-joint motors. For the general precautions for handling robots, refer to manuals prepared for the robot series.

[1] Motor handling notes

(1) When moving, wiring, maintaining motors, confirm that the controller power has been shut down.

(2) Take care not to damage or crush motor cables or let them undergo much stress or heavy load.

(3) When a motor is running, never reach out towards the rotating part.

(4) Do not touch a motor and its peripheral devices when the motor power is on and for a while after it is cut off, since they are HOT.

(5) When carrying a motor, hold the motor body not the cables, shafts or encoders.

(6) Do not put heavy materials on motors or hang them onto motors.

(7) Avoid letting motors undergo strong shock or impact.

(8) If an earthquake or any other disaster happens, confirm the installation conditions of motors and machines for safety before starting motors.

(9) Do not disassemble or modify motors.

[2] Precautions for installing motors

(1) Install specified extended-joint motors. Do not install motors other than specified ones.

(2) The robot controller and extended-joint motors will operate in combination. According to the specified capacity, select the proper motors.

(3) When hooking up motors to an extended-joint control box, be sure to check the motor models and joint numbers which are labeled on motors.

(4) When routing motor cables, observe the following to prevent them from undergoing stress:

- Protect the cable clamps and connector pins from excessive stress due to bending or cable weight.
- When installing extended-joint motors on moving mounts, be sure to fasten their cables to those mounts at the connector ends and route the relay cables through cable bearers to minimize excessive bending stress.
- Make the radius of each cable loop as large as possible.

(5) Observe the following notes relating to the allowable loads of motor output shafts:

- Design mechanisms so that the radial or thrust load that will apply to motor shafts at installation or in operation will be kept within the allowable range specified for each motor model.
- To keep radial load that would be caused by small misalignment between shafts within the allowable range, use flexible couplings exclusively designed for servomotors and as rigid as possible.
- If you want to use rigid couplings, install them with extra care. If applied in installation, excessive bending load will break motor shafts or shorten the service life of bearings.

(6) When installing or removing a coupling to/from the end of a motor shaft, do not apply any direct impact onto the motor shaft with a hammer. An encoder fitted on the other end of the shaft will be damaged.

Align two shafts with each other sufficiently. Misalignment will produce vibration, resulting in damaged bearings.

(7) Never use motors in an environment where they may be subjected to water splash, corrosive gases, or inflammable gases. Do not use them in the vicinity of the inflammable. (Extended-joint motors have not been designed to withstand explosions, dust-proof, nor are they splash-proof.)

(8) Be sure to install and set up motors so that they will never cause a fire or bodily injury even if an earthquake or any other disaster occurs.

(9) The shaft ends of extended-joint motors are lubricated with grease (SHELL Albania No. 2). Take into account that grease would influence plastics.

[3] Motor storage notes

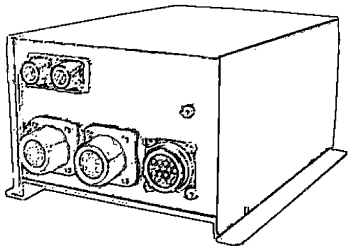
(1) Do not store motors in an environment where they may be subjected to rain, water splash, harmful gases or liquids.

(2) Shield motors from direct sunlight and store them inside the specified temperature/humidity range (-20°C to 80°C, RH 90% or less, without dew condensation).

2.4.2 Extended-Joint Control Box

The extended-joint control box distributes signal and power cables to extended-joint motors & their encoders, as well as feeding them to a robot unit.

Up to two extended-joint motors may be connected as 7th- and 8th-joints.



2.4.2.1 Specifications of the extended-joint control box

The table below lists the specifications of the extended-joint control box.

The extended-joint control box may distribute power up to two motors. Note that it is limited in the total capacity of motors connectable, depending upon the robot model already connected.

To connect extended-joint motors to the control box, you need to use dedicated cables listed in Section 2.3 "Parts Codes of Optional Components for the Extended-Joint Support System."

Specifications of the Extended-Joint Control Box (Model: RC5-EABOX-A)

Robots and controllers		Total output capacity for extended joints, when both those joints and the robot unit operate simultaneously: (Note 1)	Total output capacity for extended joints, when the robot unit is on halt: (Note 2)	
Robot type	Robot controller type		If extended-joints are used as traverse accelerators: (Note 3)	If no extended-joints are used as traverse accelerators:
HM/HS-D	RC5-EAH4A	1250W	1450W	3000W
HC-D	RC5-EAHC4A	1650W	1750W	3000W
VS-D	RC5-EAVS6A	1850W	2170W	2620W

(Note 1) When both the robot unit and extended-joint motors operate simultaneously, the extended-joint control box may output the total of 1250W or 1650W to the extended joints.

Connection example: If the output capacity for extended-joints is 1250W, the extended-joint control box may connect up to two joints of 750W and 400W each.

(Note 2) If extended-joint motors are to be driven only when the robot is on holt, these output capacities may apply.

(Note 3) The total output capacity for extended-joint motors is further limited if extended-joints are used as traverse accelerators of the robot so that they receive reaction force from the robot unit.

⚠ **CAUTION**

(1) Use the extended-joint control box with specified motors whose total capacity is less than the allowable limit.

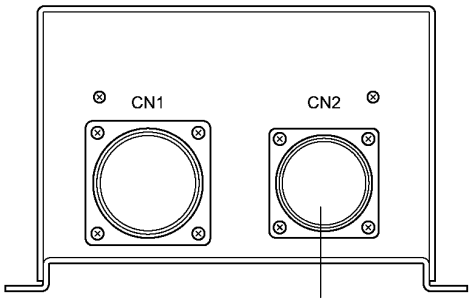
If the motors exceed the limit, the robot controller will cause an error. In the worst case, the internal circuits in the robot controller may be damaged.

(2) Even with the specified capacity of motors, the robot controller may cause a power error, overcurrent, overvoltage, overload and other errors, depending upon the use conditions of the extended-joint motors.

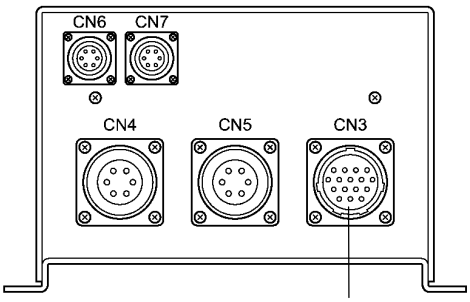
You may solve those problems by adjusting acceleration, deceleration, maximum speed, and gain of each extended-joint servo loop.

2.4.2.2 Connector names of the extended-joint control box

The connector names of the extended-joint control box are shown below.



CN2 (for VS-D)



CN3 (for H∗-D)

Connectors

Connector No.	Indication	Name
CN1	CONTROLLER	Controller connector
CN2	ROBOT (V)	VS robot motor connector
CN3	ROBOT (H)	4-axis robot motor connector

Connector No.	Indication	Name
CN4	MOTOR J7	Extended-joint J7 motor connector
CN5	MOTOR J8	Extended-joint J8 motor connector
CN6	ENCODER J7	Extended-joint J7 encoder connector
CN7	ENCODER J8	Extended-joint J8 encoder connector

⚠ CAUTION

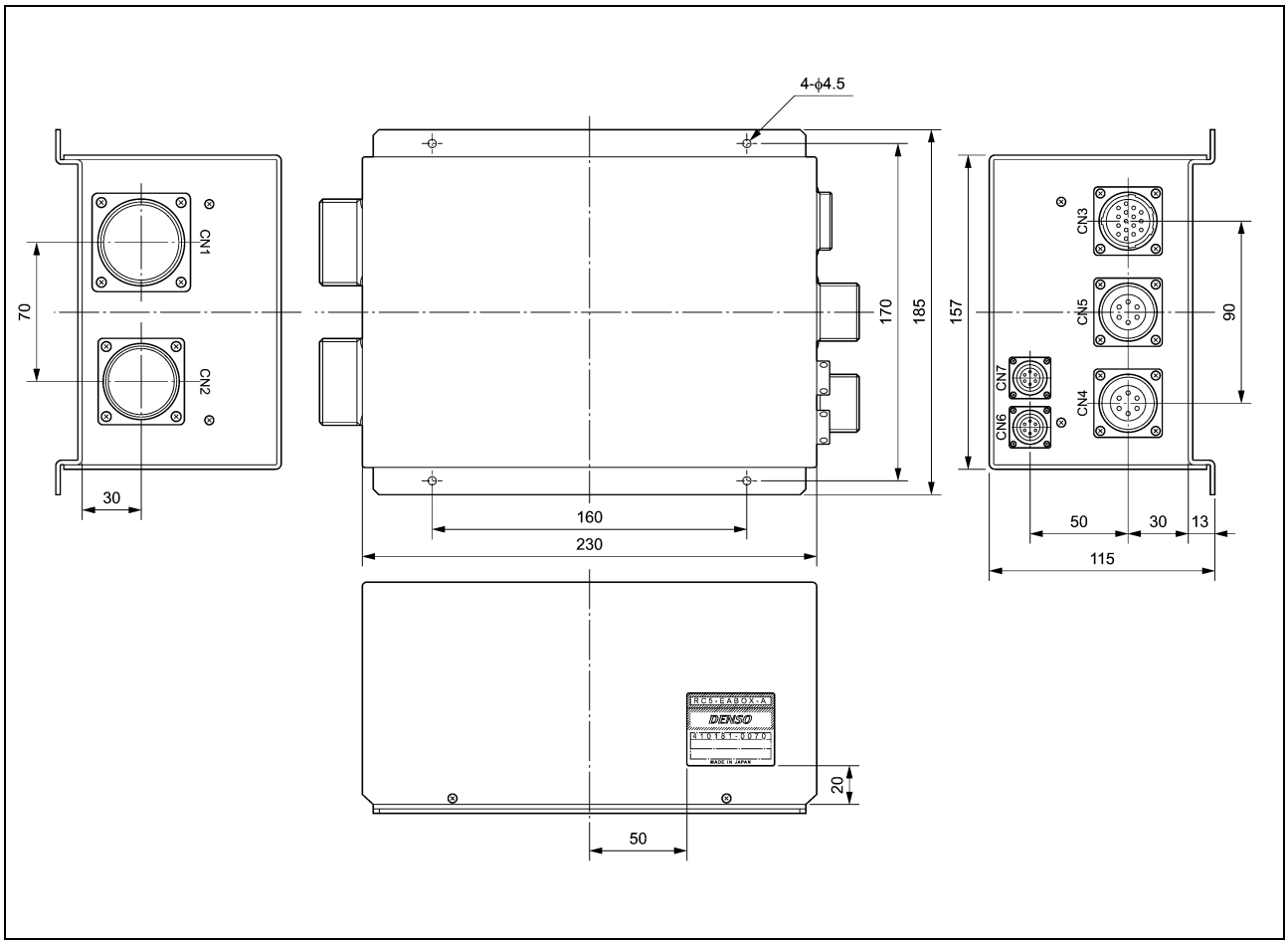
(1) Cover connectors not to be used, with dust caps that come with the extended-joint control box.

If any conductive materials accumulate on those connectors or any pointed conductive object is inserted to them when the robot controller is ON, the robot controller would be damaged.

(2) After turning the robot controller off, connect or disconnect cables to/from the extended-joint control box. If the robot controller is on, doing so may damage the internal circuits of the robot controller.

2.4.2.3 Outer dimensions of the extended-joint control box

The figure below shows the external dimensions of the extended-joint control box.



2.4.2.4 Setting up the extended-joint control box

[1] Installing the extended-joint control box

Install the extended-joint control box to a place where you may easily maintain or inspect the facilities, by bolting through holes provided in the box (shown in Subsection 2.4.2.3).

> ⚠ **CAUTION**
>
> **(1) The extended-joint control box is a floor- or panel-mount type. Always secure it on the floor or other equipment panel. Unsecured control box may cause errors or result in a damaged robot controller.**
>
> **(2) The installation environments and requirements for the extended-joint control box are listed in the table given below.**

Installation Environments and Requirements for the Extended-Joint Control Box

Items	Environments and Requirements
Ambient temperature	In operating: 0 to 40°C In storage/transportation: -10 to 60°C
Humidity	In operating: 90% or below (Dew condensation not allowed) In storage/transportation: 75% or below (Dew condensation not allowed)
Vibration	In operating: Max. 0.5G In storage/transportation: Max. 2.5G
Environments	Do not install the extended-joint control box in any environment where: - there are flammable gases or liquids, - there are any shavings from metal processing or other conductive material flying about, - there are any acidic, alkaline or other corrosive gases, - there is cutting or grinding oil mist, - there is sulfuric cutting or grinding oil mist, - there are any large-sized inverters, high output/high frequency transmitters, large contactors, welders, or other sources of electrical noise, or - it may be directly subjected to water, oil or shavings.
Work space	- Sufficient service space should be maintained for safe inspection and disassembly. - Sufficient wiring space (at least 200 mm from the rear of the extended-joint control box) should be maintained. Secure cables to the mount of the control box or to the beams of facilities to prevent their connectors from directly undergoing the cable weight.
Grounding conditions	D-type grounding (Grounding resistance: Max. 100Ω)

[2] Connecting cables to the extended-joint control box

A hook-up example of the extended-joint support system is illustrated below, where the HM-D series is connected.

Refer to this example and the connector names given in Subsection 2.4.2.2.

⚠ CAUTION

(1) When using only one extended-joint, be sure to cover the other encoder connector not in use with an encoder dummy cap that comes with the extended-joint control box.

(If an extended-joint motor is connected to the J7 encoder connector, cover the J8 with the dummy cap; and vice versa.)

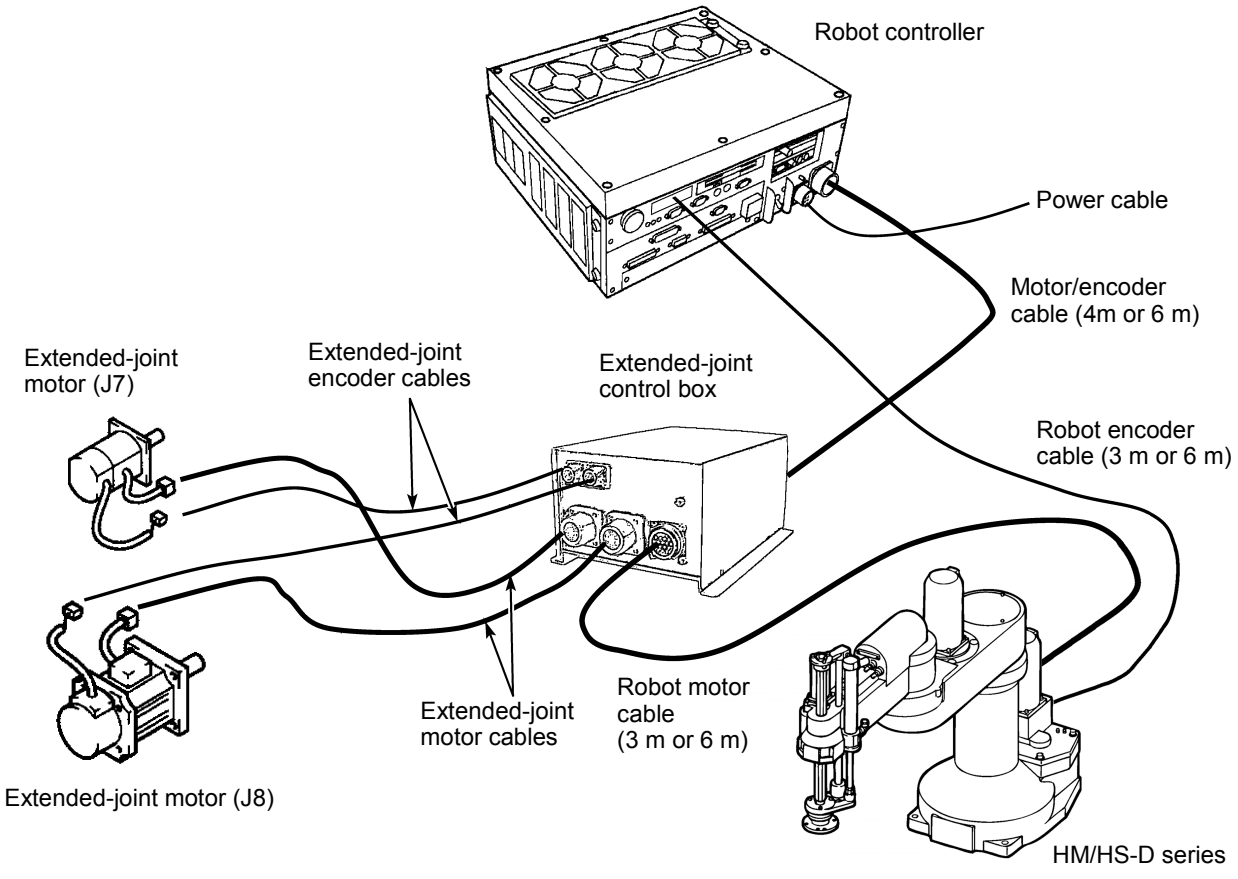
Without an encoder dummy cap connected, the extended-joint support system will not operate normally.

(2) Use DENSO-authorized cables given in Section 2.3 "Parts Codes of Optional Components for the Extended-Joint Support System."

Never modify those cables or use cables other than DENSO-authorized ones to configure an extended-joint support system.

(3) Always ground the shield conductors of extended-joint cables; otherwise, malfunctions may result due to noises.

Connection Example



Chapter 3

Extended-Joint Related Function

This chapter covers new features relating to the extended-joints.

Chapter 3 Extended-Joint Related Function

3.1 Extended-Joint Function

The extended-joint function allows you to control extra joints independently of robot joints through the standard interface of the robot controller (NetwoRC).

To support the extended-joint function, Main Software version 1.5 or later enhances the following functions:

Added functions	Extended functions
• Operating extended-joints manually • Getting extended-joint positions • Operating extended-joints by specifying a desired variable • Operating extended-joints by program (Several commands are extended and added.) • Setting the boundless rotation of extended-joints	• Command execution from the specified line • Continue function • Safe start (SS) function

NOTE: Before using these functions, you need to set various parameters of extended-joints properly.
Refer to Section 3.2 "Setting the Extended-Joint Parameters."

3.1.1 Operating Procedures of the Extended-Joint Function

This section describes the operating procedures of the extended-joint function:

[1] Operating extended-joints manually

[2] Getting extended-joint positions

[3] Operating extended-joints by specifying a desired variable

[4] Operating extended-joints by program

[5] Setting the boundless rotation of extended-joints

[6] Command execution from the specified line

[7] Continue function (for extended-joints)

[8] Safe start (SS) Function (for extended-joints)

[1] Operating extended-joints manually

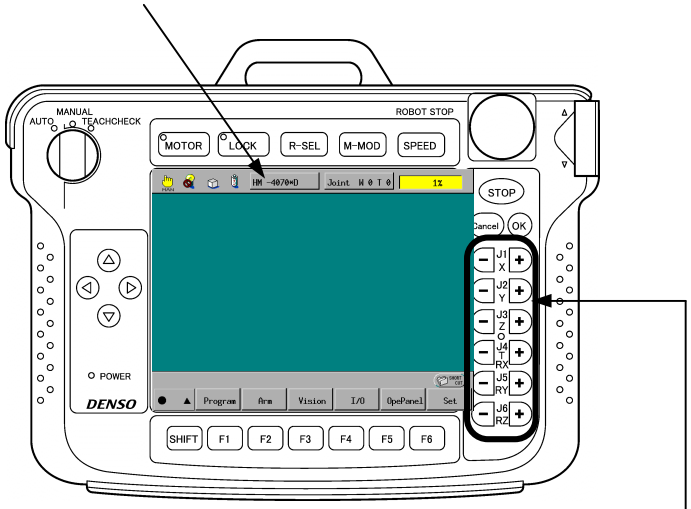
This subsection describes items relating to the extended-joints. For other details, refer to the SETTING-UP MANUAL, Section 3.2 "Manual Mode."

Extended-joints may operate only in Joint mode, independently of each other.

■ ***From the teach pendant***

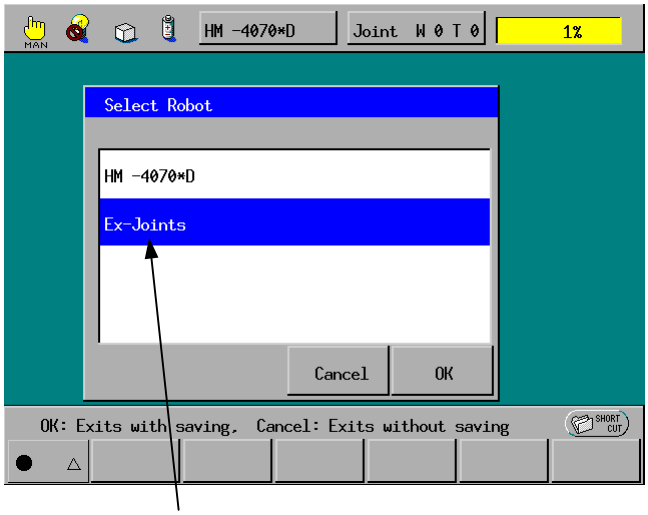
Step 1 Set the mode switch to the MANUAL position and press the MOTOR key to turn the motor on.

Step 2 On the top screen, press the robot select button (or R-SEL key).



The Select Robot window will display.

Step 3 Choose the "Ex-Joints" and then press the OK button.



Extended-joints

If you change the robot selection as shown at left, assignment of joints to the arm traverse keys in manual mode operation will be switched.
(Choosing the "Ex-Joints" will switch the joint assignment made for the Arm traverse keys from robot joints to extended-joints.)

Note: Extended-joints may operate only in Joint mode. Choosing the Ex-Joints in the above window will automatically switch to Joint mode.

NOTE: In the Select Robot window, the "Ex-Joints" will appear only when any extended-joint is set to "Valid" in the [Servo Parameters for Ex-Joint] window.

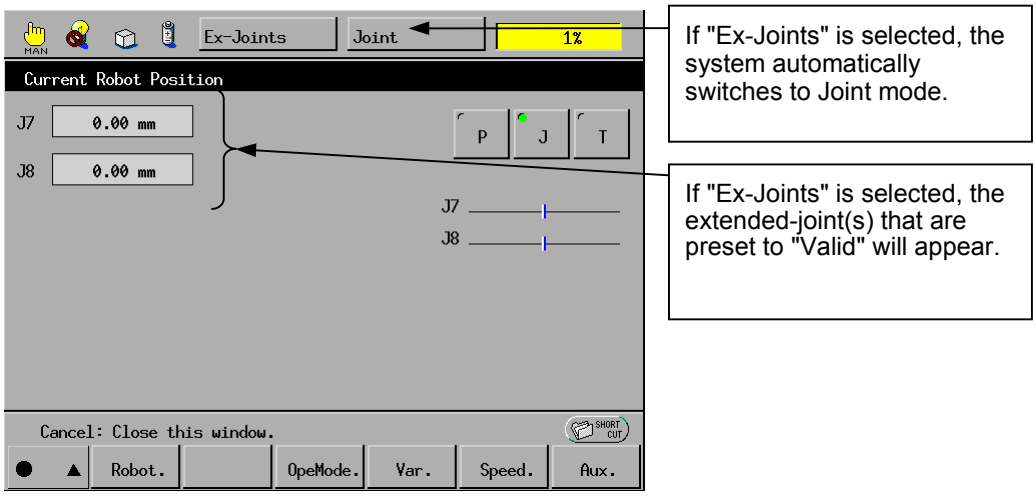
If no "Ex-Joints" appears, access [F2 Arm]—[F12 Maint.]—[F7 ExJoints]—[F8 ExServo] and then enable the desired extended-joint.

For more details, refer to Section 3.2 "Setting the Extended-Joint Parameters."

Step 4

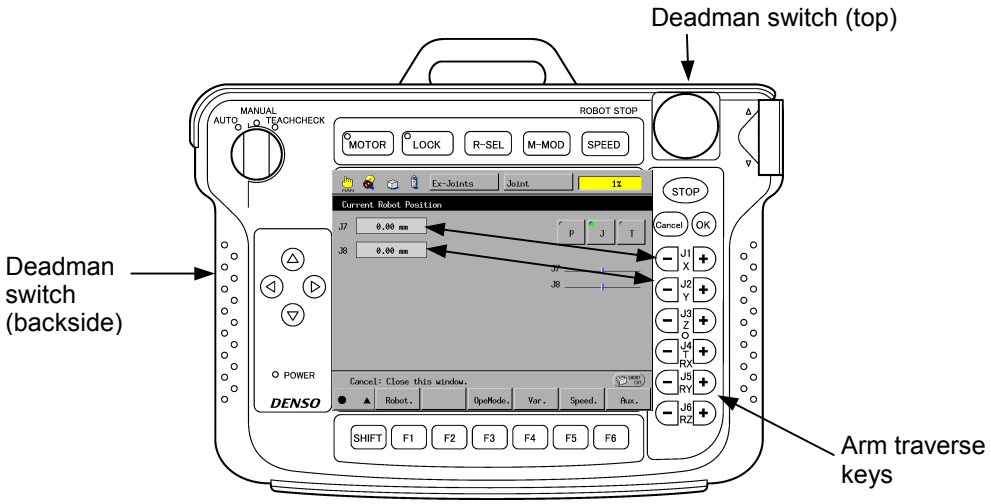
On the top screen, press [F2 Arm] to display the Current Robot Position window as shown below.

Note: Without displaying this window, you may operate the extended-joints manually.



Step 5

While holding down the deadman switch, press either of the arm traverse keys that are assigned the extended-joint you want to move.



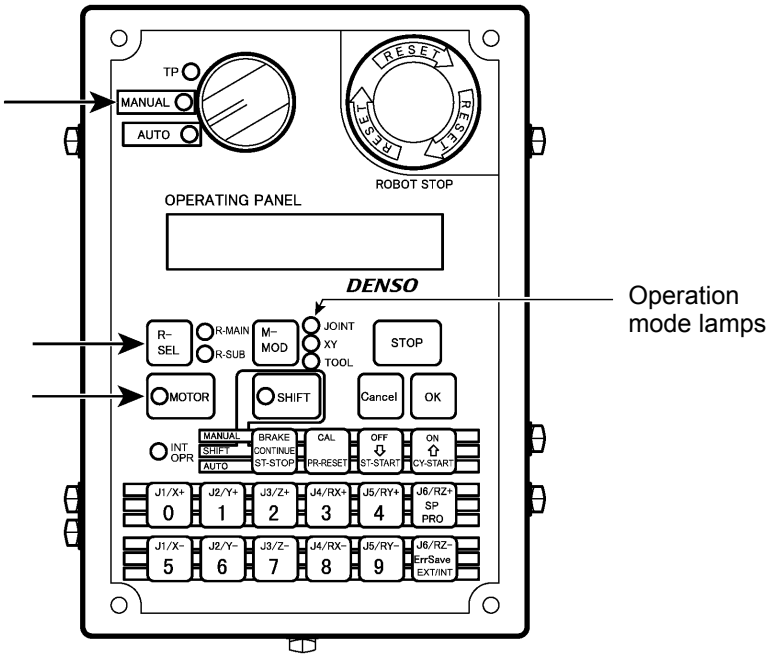
Note: Assignment of joints to the arm traverse keys is arranged from the top line first, as shown above. In this figure, you may move the 7th joint by using the [+ J1 X] and [- J1 X] keys.

■ ***From the operating panel***

Step 1 Set the mode switch to the MANUAL position and press the MOTOR key.

Make sure that the MOTOR lamp comes on.

Step 2 Press the R-SEL key to select the R-SUB. Only when the R-SUB lamp is on, the extended-joints may be driven from the operating panel.



NOTE: The extended-joints work only in Joint mode. Choosing the R-SUB in the above window will automatically switch to Joint mode.

Step 3 While holding down the deadman switch, press either of the arm traverse keys that are assigned the extended-joint you want to move.

NOTE: Assignment of joints to the arm traverse keys on the operating panel is the same as that on the teach pendant.

(Example) If the 7th and 8th joints are set to "Valid" in the [Servo Parameters for Ex-Joint] window (and other joints are not) and the R-SUB is selected with the R-SEL key, then you may move the 7th and 8th joints by using the [+ J1 X][-J1 X] and [+ J2 X][- J2 X], respectively.

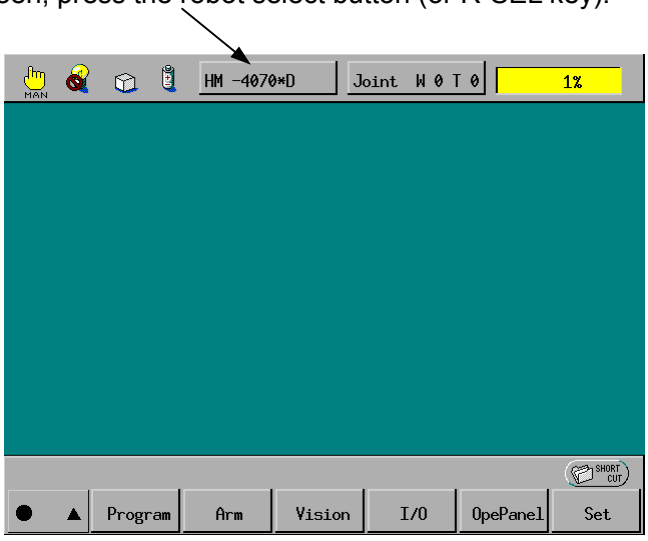
[2] Getting extended-joint positions

Get the positions of extended-joints into floating-point variables, joint by joint.

■ ***From the teach pendant***

Step 1

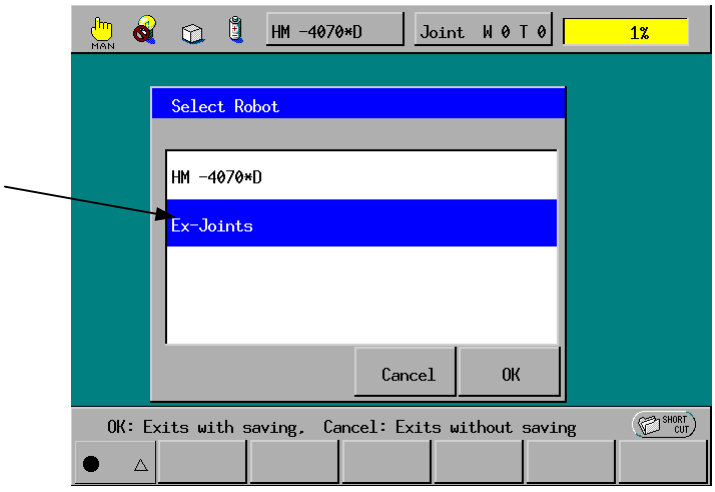
On the top screen, press the robot select button (or R-SEL key).



The Select Robot window will display.

Step 2

Choose the "Ex-Joints" and then press the OK button.



NOTE: In the Select Robot window, the "Ex-Joints" will appear only when any extended-joint is set to "Valid" in the [Servo Parameters for Ex-Joint] window.

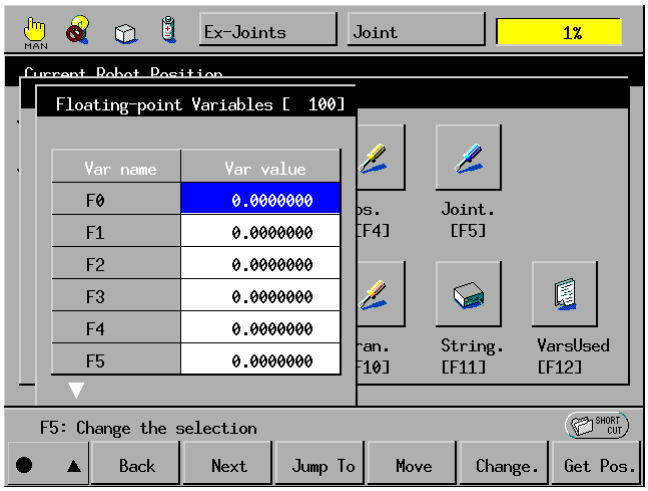
If no "Ex-Joints" appears, access [F2 Arm]—[F12 Maint.]—[F7 ExJoints]—[F8 ExServo] and then enable the desired extended-joint.

For more details, refer to Section 3.2 "Setting the Extended-Joint Parameters."

Step 3

From the top screen of the teach pendant, access [F2 Arm]—[F4 Var.]—[F2 Float.] to call up the Floating-point Variables window.

Select a variable name to which you want to get a position value.



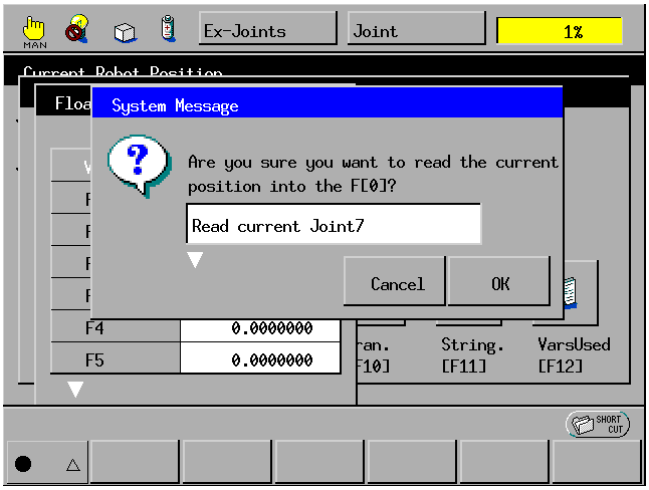
This example will get the extended-joint position value into variable F0.

NOTE: The [Move] and [Get Pos.] function buttons for the floating-point variables will display only when the "Ex-Joints" is selected in the Select Robot window.

Step 4

Press [F6 Get Pos.].

The window below will appear. Select a joint whose position should be read in, and then press [OK].



This example will read in the position of the 7th joint.

NOTE: Only extended-joints that are set to "Valid" in the [Servo Parameters for Ex-Join] window may be read into floating-point variables. No robot joints can be read.

The above procedure reads the current position of the 7th joint into floating-point variable F0.

■ ***From the operating panel***

Step 1

Set the mode switch to the MANUAL position.

Confirm that the MANUAL lamp comes on.

Step 2

Press the SHIFT key. Make sure that the SHIFT lamp comes on. Then press the BRAKE/CONTINUE/ST-STOP key.

Step 3

Use the [↑] and [ ] keys to choose [Set CurPos F] on the LCD screen, then press the OK key.

Step 4

Selecting a variable name to which you want to get a position value.

If "F?" appears on the LCD, enter the name of a floating-point variable by using the numeric keys and then press the OK key.

Step 5

Selecting a joint whose position should be read in.

If "JointNo" appears on the LCD, enter the number of a floating-point variable by using the numeric keys and then press the [OK] key.

NOTE: Only extended-joints that are set to "Valid" in the [Servo Parameters for Ex-Join] window may be read into floating-point variables. No robot joints can be read.

Step 6

Reading in the joint position.

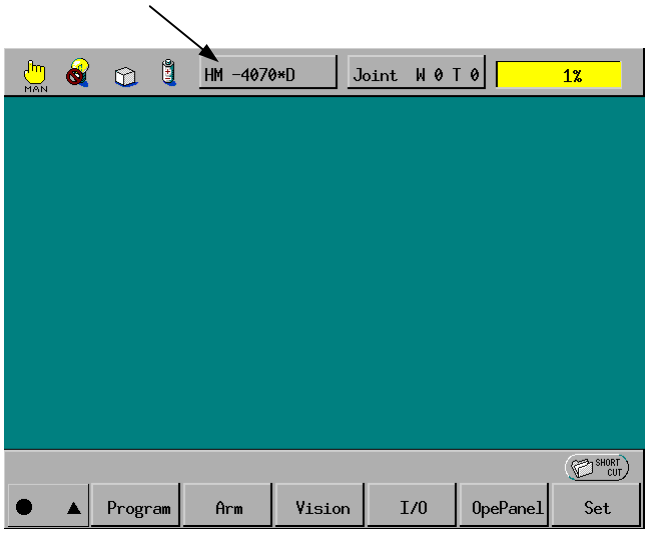
If "SetCurPos" appears on the LCD, press the OK key. The "SetVarVal OK" appears on the LCD, showing that the position has been successfully read into the specified floating-point variable.

[3] Operating extended-joints by selecting a desired variable

You can operate the extended-joints manually by specifying a desired floating-point variable for each joint.

Step 1

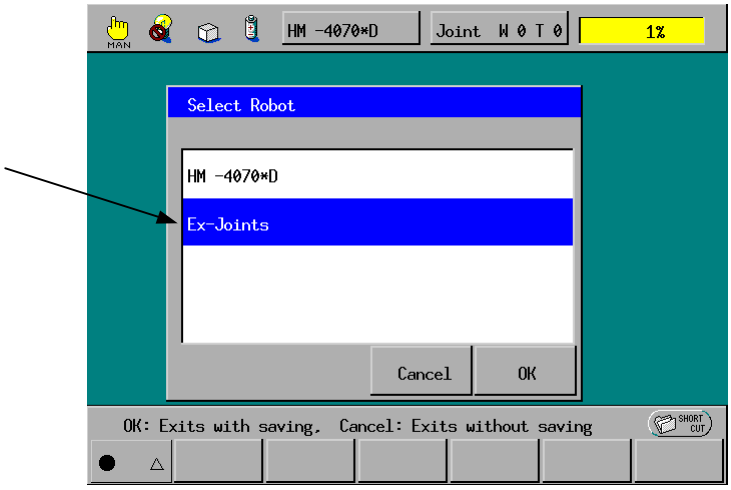
On the top screen, press the robot select button (or R-SEL key).



The Select Robot window will display.

Step 2

Choose the "Ex-Joints" and then press the OK button.



NOTE: In the Select Robot window, the "Ex-Joints" will appear only when any extended-joint is set to "Valid" in the [Servo Parameters for Ex-Joint] window.

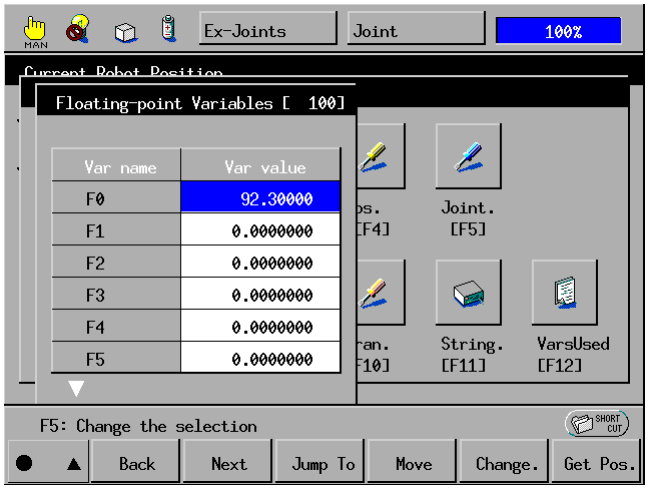
If no "Ex-Joints" appears, access [F2 Arm]—[F12 Maint.]—[F7 ExJoints]—[F8 ExServo] and then enable the desired extended-joint.

For more details, refer to Section 3.2 "Setting the Extended-Joint Parameters."

Step 3

From the top screen of the teach pendant, access [F2 Arm]—[F4 Var.]—[F2 Float.] to call up the Floating-point Variables window.

Select a variable having the desired movement value.



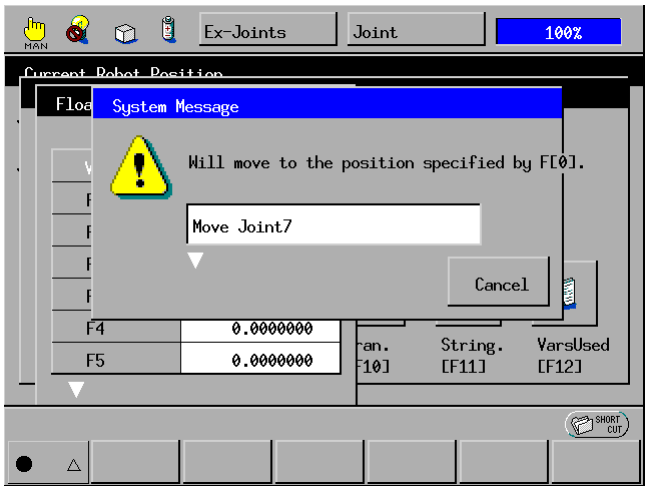
This example is to move an extended-joint to the position 92.30 assigned to floating-point variable F0.

NOTE: The [Move] and [Get Pos.] function buttons for the floating-point variables will display only when the "Ex-Joints" is selected in the Select Robot window.

Step 4

Press [F4 Move].

The window below will appear. Select an extended-joint you want to move by using the floating-point variable specified in Step 3.

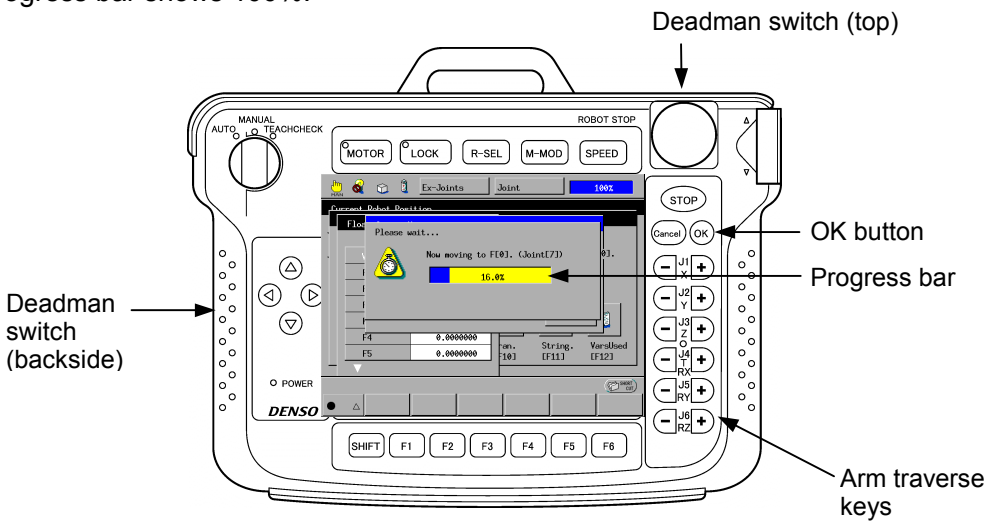


This example is to move the 7th joint to the position 92.30 assigned to floating-point variable F0.

NOTE: Only extended-joints that are set to "Valid" in the [Servo Parameters for Ex-Join] window may be read into floating-point variables. No robot joints can be read.

Step 5

While holding down the deadman switch, keep on pressing the OK button until the progress bar shows 100%.



The above procedure moves the 7th joint to the position 92.30 assigned to floating-point variable F0.

Note: If an extended-joint(s) you want to move by using variables is contained in an arm group and any task holding its semaphore is active, then the extended-joint cannot be moved by using variables. For more details on the arm group semaphore, refer to the next subsection "[4] Operating extended-joints by program."

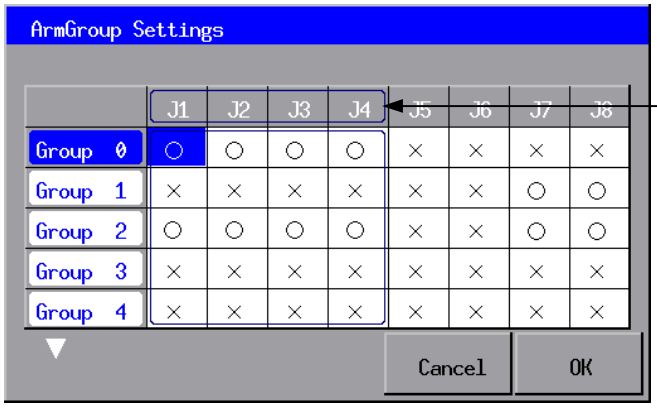
[4] Operating extended-joints by program

To operate extended-joints by program, you need to set up an arm group.

■ *Concept of an arm group*

"Arm Group" is a semaphore to control joints in a group as a single unit. Only when a task holds an "arm group" semaphore (herein called "arm group" and expressed as "Group n"), it can control robot joints and extended-joints. An arm group prevents more than one task from executing motion commands to a same joint at the same time.

Example of Arm Group Setting (For 4-axis robots)



In the above example:

Group 0: Only robot joints involved.

Group 1: Only extended-joints (7th and 8th) involved.

Group 2: Robot joints and extended-joints (7th and 8th) involved.

By activating a task holding the "arm group" semaphore with extended-joints involved, you may operate the extended-joints.

In the above setting example, a task holding Group 1 can control the 7th and 8th extended-joints only.

■ *Getting an arm group*

To make tasks get an arm group, give a TAKEARM command an arm group number as an argument as shown below.

```
PROGRAM PRO1
TAKEARM 1
   :
   :
END
```

Program PRO1 gets Arm Group 1 by TAKEARM command with an argument set to 1.

■ *Restrictions on the application of arm groups*

If more than one task involves joints shared in their arm groups, those tasks cannot be *simultaneously* executed.

Tasks involving arm groups without shared joints can be *simultaneously* executed.

Example: Getting an Arm Group in Programs

	J1	J2	J3	J4	J5	J6	J7	J8
Group 0	○	○	○	○	×	×	×	×
Group 1	×	×	×	×	×	×	○	○
Group 2	○	○	○	○	×	×	○	○
Group 3	×	×	×	×	×	×	×	×
Group 4	×	×	×	×	×	×	×	×

```
PROGRAM PRO0        PROGRAM PRO1        PROGRAM PRO2
TAKEARM 0           TAKEARM 1           TAKEARM 2
    :                   :                   :
    :                   :                   :
END                 END                 END
```

Arm Groups 0 and 1 involve no shared joint, so PRO0 and PRO1 can be executed simultaneously. (Robot joints and extended-joints can be simultaneously controlled by different programs.)

Arm Groups 0 and 2 involve shared joints (J1 to J4), so PRO0 and PRO2 cannot be executed simultaneously. When the later TAKEARM command is executed, an error will occur.

■ *Releasing the currently held arm group*

Executing a GIVEARM command releases the currently held arm group.

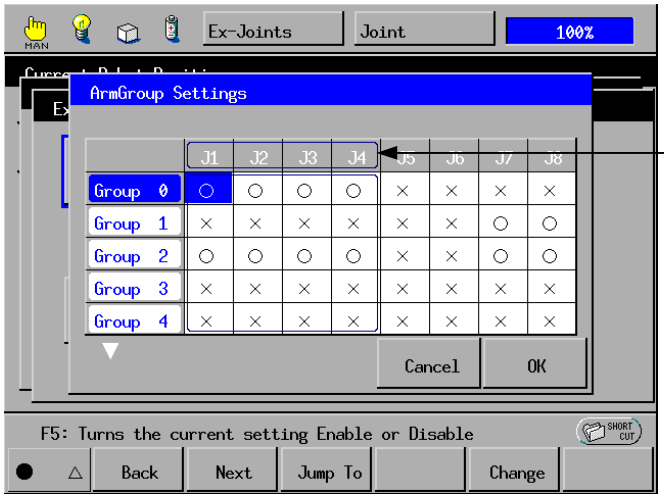
The currently held arm group will be automatically released also if a program stops due to occurrence of an error or because of normal program end.

NOTE: No arm group will be released by a Halt or Step stop signal.

■ ***Setting up an arm group***

Step 1 Call up the Arm Group Settings window by accessing:

[F2 Arm]—[F12 Maint.]—[F7 ExJoints]—[F1 ArmGroup]



Robot joints enclosed in this box

F5

Step 2 Move the cursor to a joint to be grouped and then press [F5 Change].

Step 3 After marking joints to be grouped or not with "O" or "X", respectively, press [OK].

Notes for setting up an arm group

(1) New settings will go into effect when the controller is turned off and then on after the settings is changed.

(2) Arm Group 0 cannot be accessed.

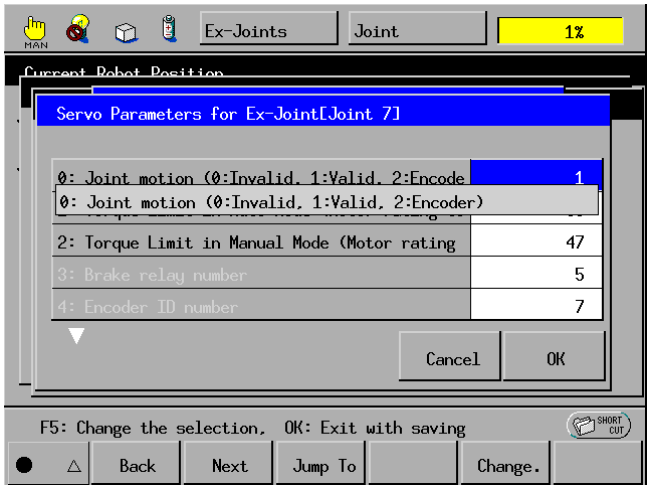
(3) Robot joints may be set all to either "Enabled" or "Disabled." For example, in a 4-joint robot, all four joints are either "Enabled" or "Disabled."

(4) Extended-joints that you want to enable in an arm group should be set to "Valid" in the [Servo Parameters for Ex-Joint] window beforehand. If not, an error will occur when you are attempting to set up an arm group.

Access to the [Servo Parameters for Ex-Joint] window is:

[F2 Arm]—[F12 Maint.]—[F7 ExJoints]—[F8 ExServo]

For more details, refer to Section 3.2 "Setting the Extended-Joint Parameters."



[5] Boundless rotation of extended-joints

The boundless rotation function suppresses errors that could occur if an extended-joint keeps on rotating in the same direction by a relative motion command (DRIVE or MOVE with EX option).

You need to set the boundless rotation parameter to [1: Boundless] in the [Path parameter for Ex-Joint] window.

Notes for allowing boundless rotation on extended-joints

(1) An absolute motion command (DRIVEA or MOVE with EXA option) cannot drive extended-joints whose boundless rotation is allowed.

(2) Extended-joints whose boundless rotation is allowed cannot be moved by using variables.

(3) When an extended-joint keeps on rotating in the same direction, the current value might suddenly jump to a large minus value.

(4) In a DRIVEA or MOVE with EXA option, up to 7 digits of value may be specified for boundless rotation. If a value exceeding 7 digits is specified, the actual rotation amount will differ from the specified one.

For example
If DRIVE (5, 11111115555) is specified, 11111115555 will be internally interpreted as 1.111111*E+10 and 5555 will be ignored due to the definition of a single precision floating point number.

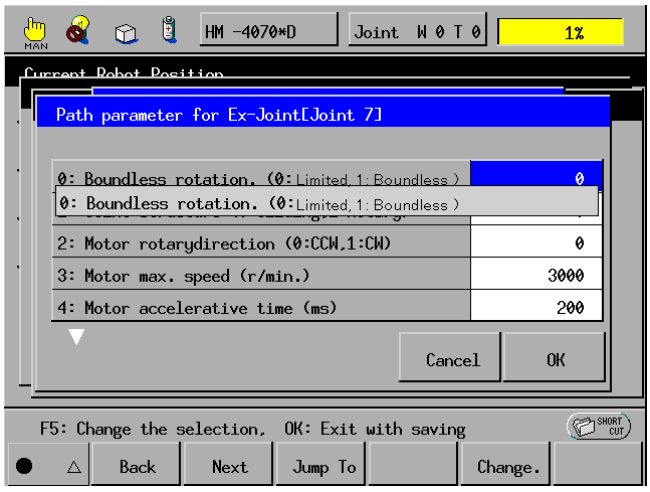
(5) If a large value is specified as the amount of movement at one time in boundless rotation, then the "Out of range" error will occur. The quantum of movement depends on the gear ratio.

■ *Allowing boundless rotation*

From the top screen of the teach pendant, access:
[F2 Arm]—[F12 Maint.]—[F7 ExJoints]—[F7 ExPath]

In the [Path parameter for Ex-Joint] window, set the boundless rotation parameter to [1: Boundless].

For more details, refer to Section 3.2 "Setting the Extended-Joint Parameters."



[6] Command execution from the specified line for extended-joints

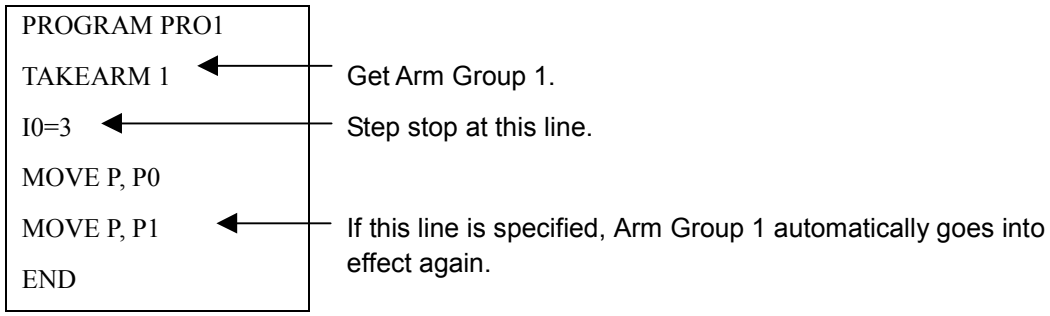
The specifications about command execution from the specified line are described in the SETTING-UP MANUAL, Section 3.3 "Teach Check Mode (TP), [4]."

This subsection describes only what is different from those specifications.

■ *Making effective the arm group previously obtained by TAKEARM*

If a command on the specified program line is executed, an arm group obtained by a TAKEARM command on the earlier line will automatically take effect.

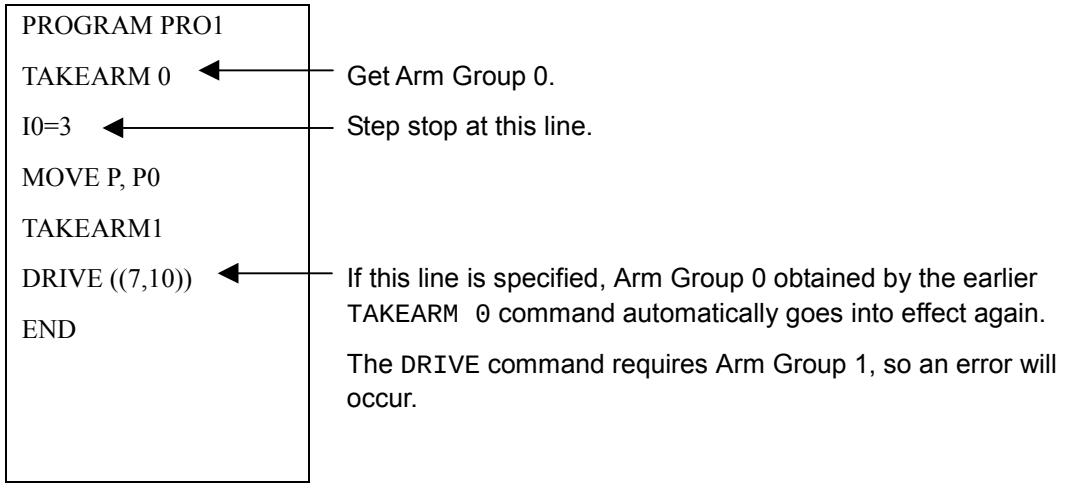
Example



Notes for executing a command from the specified line

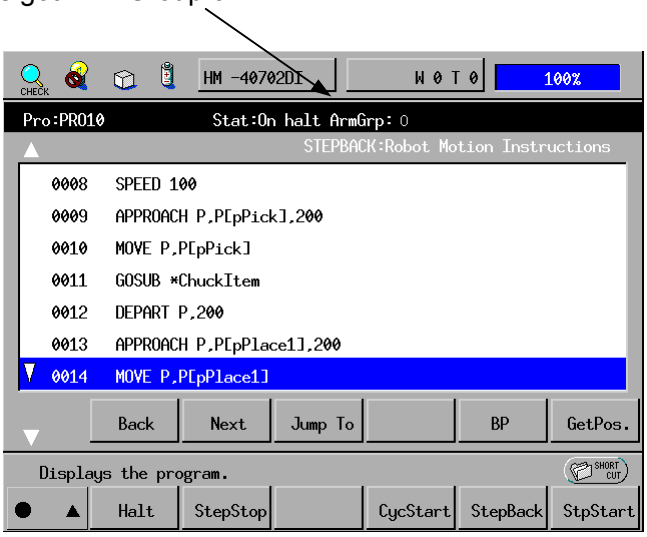
When using more than one arm group in a single task, be careful with the case given in the following sample:

Example



You may check an active arm group currently held by a task by displaying the task.

This program has got Arm Group 0.



[7] Continue function for extended-joints

The specifications of the continue function are described in the SETTING-UP MANUAL, Subsection 3.4.5.

This subsection describes what is different from those specifications.

■ *Auto position correction for extended-joints*

You may choose extended-joints you want to correct in position when Continue Start is executed.

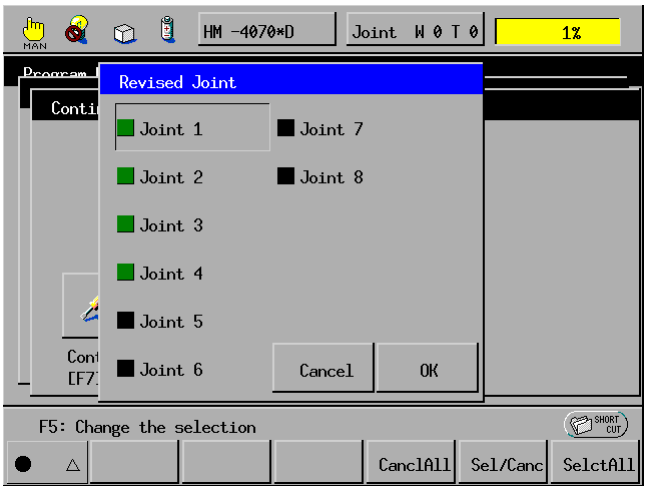
This function is useful for those extended-joints incapable of rotating in the reverse direction. For details, refer to the SETTING-UP MANUAL, Subsection 3.4.5.

■ *Choosing extended-joints to be corrected in position*

To call up the Revised Joint window, access:

Top Screen—[F1 Program]—[F6 Aux.]—[F7 Continue]—[F8 ReviseJnt]

Window for Specifying Extended-Joints to be Corrected in Position



Choose joints to be corrected in position and then press [OK].

In the above example, robot joints (1 to 4) are selected so that those joints only will be corrected in position at Continue Start.

Notes

All robot joints (1 to 4) should be selected as a single unit. You cannot select any out of them.

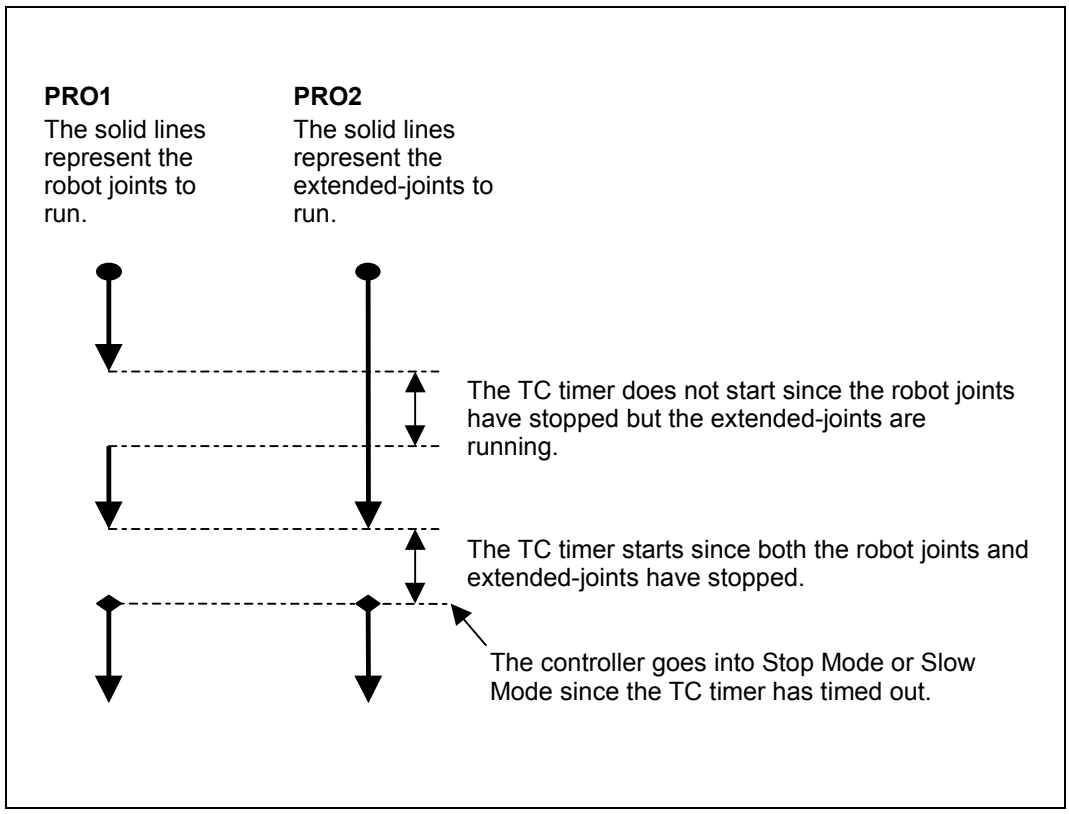
[8] SS (Safe Start) function for extended-joints

The specifications of the SS (Safe start) function are described in the SETTING-UP MANUAL, Subsection 3.4.6.

This subsection describes what is different from those specifications.

■ *Detecting TC time*

This function detects TC time if both the robot joints and extended-joints are not in motion for the specified period in the currently running program.



■ *Slow Mode*

If the TC timer times out, the speed of the subsequent operation of robot joints and extended-joints will become low.

■ *Stop Mode*

If the TC timer times out, the program will execute "Continue stop," stopping all the robot joints and extended-joints.

3.1.2 Commands Supporting Extended-Joints

NOTE: Commands described in this subsection may be used for general robots, not exclusively used for extended-joints.

TAKEARM (Statement)

Function

Gets an arm group. Upon execution of this statement, the programmed speed, acceleration and deceleration will be set to 100. If the gotten arm group includes any robot joint, this statement restores the tool coordinates and work coordinates to the origin.

Syntax

```
TAKEARM[<ArmGroupNumber>][<KEEP=DefaultValue>]
```

Description

This section explains the functions relevant to extended-joints. For details, refer to the PROGRAMMER'S MANUAL, Section 14.3 ”TAKEARM."

If <ArmGroupNumber> is omitted, the TAKEARM gets the semaphore of Arm Group 0 in which only robot joints are enabled.

If a task holding no arm group attempts to execute any of the following motion commands, an error will occur. Before executing those commands, get the arm semaphore (Arm Group) by using the TAKEARM command.

Commands Requiring Arm Group

Commands	Requires:
HOME, TOOL, WORK, APPROACH, DEPART, DRAW, GOHOME, MOVE, ROTATEH, ROTATE, CHANGETOOL, CHANGEWORK, DRIVE, DRIVEA, SPEED, JSPEED, ACCEL, JACCEL, DECEL, JDECEL, INTERRUPT, LETENV, Motion optimization library, and Arm motion library	Arm group involving robot joints
DRIVE, DRIVEA, SPEED. JSPEED. ACCEL, JACCEL, DECEL, JDECEL, INTERRUPT, LETENV, and POSCLR	Arm group that may or may not involve robot joints

Notes

(1) The DRIVE and DRIVEA commands require an arm group involving a joint(s) to move.

Example: DRIVE (7,10)

To move the 7th joint, the task should hold the arm group involving the 7th joint.

(2) The MOVE command with EX (EXA) option requires the arm group involving robot joints and extended-joints which are invoked by the EX (EXA) option.

Example: MOVE P, P0 EX ((7,10))

The arm group should involve robot joints as well as the 7th joint.

(3) The speed setting commands will only change the speed of joints involved in an active arm group currently held in the task.

(4) The LETENV command requires an arm group involving joints associated with parameters to be changed.

Related commands

GIVEARM

Examples

Shown below are program samples for the TAKEARM command when arm groups are arranged as listed below (for 4-axis robots).

	J1	J2	J3	J4	J5	J6	J7	J8
Group 0	○	○	○	○	×	×	×	×
Group 1	×	×	×	×	×	×	○	○
Group 2	○	○	○	○	×	×	○	○
Group 3	×	×	×	×	×	×	×	×
Group 4	×	×	×	×	×	×	×	×

```
PROGRAM PRO1
TAKEARM 1                'Get Arm Group 1 (7th extended-joint involved).
DRIVEA (7,100)           'Move the 7th extended-joint to an angle of 100
                         'degrees.
END


PROGRAM PRO2
TAKEARM 2                'Get Arm Group 2 (all robot joints and 7th
                         'extended-joint involved).
MOVE P,P0 EX ((7,10))    'Simultaneously move robot joints and 7th
                         'extended-joint.
DRIVEA (7,100)           'Move the 7th extended-joint to an angle of
                         '100 degrees.
END
```

Notes

(1) One program cannot hold more than one different arm group. However, it can get the same arm group again in one program.

```
Example:  TAKEARM 0
          MOVE P, P0
          TAKEARM 0      'Can get Arm Group 0 again, even if this
                         'program has got it.
          TAKEARM 1      'Error occurs since TAKEARM attempts to
                         'get Arm Group 1 while Arm Group 0 has been
                         'held.
```

(2) If TAKEARM with KEEP option being set to "1" gets an arm group, the arm speed will not be initialized and remain as that of the old arm group.

Example: If arm groups are arranged as listed below and the three programs exist:

	J1	J2	J3	J4	J5	J6	J7	J8
Group 0	○	○	○	○	×	×	×	×
Group 1	×	×	×	×	×	×	○	○
Group 2	○	○	○	○	×	×	○	○
Group 3	×	×	×	×	×	×	×	×
Group 4	×	×	×	×	×	×	×	×

```
PROGRAM PRO1
TAKEARM 1
SPEED 10
DRIVE(7,10)
END
```

```
PROGRAM PRO2
TAKEARM 2
SPEED 20
DRIVE(7,10)
END
```

```
PROGRAM PRO3
TAKEARM 1 keep=1
DRIVE(7,10)
END
```

First run PRO1. The 7th-joint will move at speed 10.

Next run PRO2. The 7th-joint will move at speed 20.

Then run PRO3. Since the KEEP option is "1", the 7th-joint will move at speed 10 keeping the speed defined by SPEED command for Arm Group 1 in PRO1.

GIVEARM (Statement)

Function

Releases the currently held arm group.

Syntax

```
GIVEARM
```

Description

This subsection explains the functions relevant to extended-joints. For others, refer to the PROGRAMMER'S MANUAL, Section 14.2 "GIVEARM."

GIVEARM releases the currently held arm group, allowing other tasks to get the arm group semaphore.

If a task that holds no arm group attempts to execute the GIVEARM command, an error will result.

NOTE:

In any of the following cases, the program or task automatically releases the arm semaphore (arm group), so you may omit the GIVEARM command.

- **When an END command executes and normally terminates a program. (Excluding an END written at the end of a called program.)**
- **When a program or task stops due to any error or by execution of a KILL command.**

  (Note that an arm group will not be released by Halt or Step stop.)

Related commands

TAKEARM

Examples

```
PROGRAM PRO1
TAKEARM 1       'Get Arm Group 1.
GIVEARM         'Release Arm Group 1 currently held.
END

PROGRAM PRO2
TAKEARM 2       'Get Arm Group 2.
END             'Stop the program normally and release Arm
                'Group 2 currently held.
```

Notes

If a GIVEARM command is issued, the controller will execute it after completion of all arm motions invoked by the current arm group. Even if you specify a pass motion preceding a GIVEARM, therefore, the controller will ignore it so that no pass motion will execute.

DRIVE (Statement)

Function

Carries out the relative motion of each joint.

Syntax

```
DRIVE[<@PassStartOffset>](<JntNumber>,<RelativeDistance>)
[,(<JntNumber>,<RelativeDistance>)...][,<MotionOption>]
[,NEXT]
```

Description

DRIVE moves a joint specified by <JntNumber> towards the angle (in degrees) specified by <RelativeDistance>.

If <RelativeDistance> is positive, the joint moves in the positive direction; if negative, the joint moves in the negative direction.

To execute this command, you need to get an arm group involving a joint(s) to be moved.

If a same joint is specified repeatedly, the last specification takes effect.

For details about <@PassStartOffset>, <MotionOption>, and NEXT, refer to the PROGRAMMER'S GUIDE, Section 12.1 "DRIVE."

Related commands

DRIVEA, MOVE with EX or EXA option

(Commands applicable to the extended-joints include MOVE with EX or EXA option, DRIVE and DRIVEA only.)

Example

	J1	J2	J3	J4	J5	J6	J7	J8
Group 0	○	○	○	○	×	×	×	×
Group 1	×	×	×	×	×	×	○	○
Group 2	○	○	○	○	×	×	○	○
Group 3	×	×	×	×	×	×	×	×
Group 4	×	×	×	×	×	×	×	×

```
PROGRAM PRO1
TAKEARM1                'Arm Group 1 involves 7th and 8th joints.
DRIVE (7,30),(8,50)     'Operate 7th and 8th joints
END
```

In the above example, to move the 7th and/or 8th extended-joint, the program needs to get Arm Group 1 or Arm Group 2.

Notes

If a numerical value is set to <@PassStartOffset> for the specified extended-joint, the joint will move in pass motion regardless of the entered value.

DRIVEA (Statement)

Function

Carries out the absolute motion of each joint.

Syntax

```
DRIVE[<@PassStartOffset>](<JntNumber>,<JntCoordinates>)
[,(<JntNumber>,<JntCoordinates>)...][,<MotionOption>]
[,NEXT]
```

Description

DRIVE moves a joint specified by <JntNumber> towards the angle (in degrees) specified by <JntCoordinates>.

To execute this command, it is necessary to get an arm group including a joint(s) to be moved.

If a same joint is specified repeatedly, the last specification takes effect.

For details about <@PassStartOffset>, <MotionOption>, and NEXT, refer to the PROGRAMMER'S GUIDE, Section 12.1 "DRIVEA."

Related commands

DRIVE, MOVE with EX or EXA option

(Commands applicable to the extended-joints include MOVE with EX or EXA option, DRIVE and DRIVEA only.)

Example

	J1	J2	J3	J4	J5	J6	J7	J8
Group 0	○	○	○	○	×	×	×	×
Group 1	×	×	×	×	×	×	○	○
Group 2	○	○	○	○	×	×	○	○
Group 3	×	×	×	×	×	×	×	×
Group 4	×	×	×	×	×	×	×	×

```
PROGRAM PRO1
TAKEARM1                  'Arm Group 1 involves 7th and 8th joints.
DRIVEA (7,30),(8,50)      'Operate 7th and 8th joints
END
```

In the above example, to move the 7th and/or 8th extended-joint, the program needs to get Arm Group 1 or Arm Group 2.

Notes

(1) If a numerical value is set to <@PassStartOffset> for the specified extended-joint, the joint will move in pass motion regardless of the entered value.

(2) This command is not applicable to any joints specified for boundless rotation.

MOVE (Statement)

Function

Moves the robot flange to the specified coordinates.

If specified with an EX option (relative motion of extended-joints) or EXA option (absolute motion of extended-joints), the MOVE can move both the robot flange and the extended-joints synchronously. (i.e. Synchronized start and stop from/at the specified positions is possible.)

Syntax

```
MOVE<InterpolationMethod>,[@<PassStartOffset>]<Pose>
[<EX/EXAoption>][,[@<PathStartOffset>]<Pose>[<EX/EXAoption
>]…][,<MotionOption>][, NEXT]
```

Description

This subsection explains the functions relevant to extended-joints. For others, refer to the PROGRAMMER'S MANUAL, Section 12.1 "MOVE."

The MOVE moves the joints from the current positions to the coordinates specified by <Pose>.

The syntax of an EX or EXA option is shown below.

<EX/EXAoption> syntax

```
EX((<JntNumber>,<RelativeDistance>)[,(<JntNumber>,
<RelativeDistance>)…])
```

```
EXA((<JntNumber>,<AxisCoordinates>)[,(<JntNumber>,
<AxisCoordinates>)…])
```

To <JntNumber>, you can specify only an extended-joint, never specify any robot joint.

For details about other options, refer to the PROGRAMMER'S MANUAL, Section 12.1 "MOVE."

Related commands

DRIVE, DRIVEA

(Commands applicable to the extended-joints include MOVE with EX or EXA option, DRIVE and DRIVEA only.)

Example

	J1	J2	J3	J4	J5	J6	J7	J8
Group 0	○	○	○	○	×	×	×	×
Group 1	×	×	×	×	×	×	○	○
Group 2	○	○	○	○	×	×	○	○
Group 3	×	×	×	×	×	×	×	×
Group 4	×	×	×	×	×	×	×	×

```
PROGRAM PRO1
TAKEARM 2                          'Get Arm Group 2 involving both robot
                                   'joints and extended-joints.
MOVE P, P0 EX ((7,30),(8,10))      'Move robot flange to P0 as well as
                                   'moving 7th and 8th extended-joints to
                                   'relative coordinates synchronously.
MOVE P, P1 EXA ((7,30))            'Move robot flange to P1 as well as
                                   'moving 7th extended-joint to absolute
                                   'coordinates synchronously.
END
```

Notes

(1) The MOVE with EXA option (absolute motion) is not applicable to any extended-joints specified for boundless rotation.

(2) Given below is an example handling a POSE array.

Example: `MOVE P,P[3 TO 5]EX((7,30))`

The above sample will produce the same result as the following; that is, the motion of the 7th joint will synchronize with that of the robot flange to P5.

```
MOVE P,P3,NEXT
MOVE P,P4,NEXT
MOVE P,P5 EX((7,30))
```

CUREXJ (Statement)

Function

Gets the current angle of an extended-joint into a floating-point variable.

Syntax

```
CUREXJ(<JntNumber>)
```

Description

CUREXJ reads an angle detected by the encoder of an extended-joint specified by <JntNumber> into a floating-point variable. If the specified joint is in motion, this command will get the current angle of the joint detected when this command executes.

To get the target angle of an extended-joint, use a DESTEXJ command.

Related commands

CURJNT, CURPOS, CURTRN, and DESTEXJ

Example

```
PROGRAM PRO1
DIM lf1 AS SINGLE
TAKEARM 1
DRIVEA (7,100)        'Move the 7th joint to an angle of 100 degrees.
lf1 = CUREXJ(7)       'Assign current position of 7th joint to lf1.
END
```

Notes

When the machine is locked (Machine Lock state), the CUREXJ command will get not an angle detected by an encoder but a virtual angle (commanded angle).

DESTEXJ (Statement)

Function

Gets the target position of an extended-joint invoked by the current motion command into a floating-point variable. If the robot is on halt, this command will get the current position (commanded value).

Syntax

```
DESTEXJ(<JntNumber>)
```

Description

DESTEXJ reads the target position of an extended-joint invoked by the current motion command and specified by <JntNumber> into a floating-point variable.

To get a position detected by the encoder of an individual joint, use a CUREXJ command.

Related commands

CUREXJ, DESTJNT, DESTPOS, and DESTTRN

Example

```
PROGRAM PRO1
DIM lf1 AS SINGLE
TAKEARM 1
DRIVEA (7,100),NEXT      'Move 7th joint to an angle of 100 degrees.
                         'NEXT advances the process to the next command
                         'before the motion is completed.
lf1 = DESTEXJ(7)         'Assign target angle 100 of previously
                         'commanded motion into lf1.
END
```

Notes

If a robot motion is stopped by entering Stop motion command (Refer to INTERRUPT ON/OFF stated later), the DESTEXJ will get the joint angle where the motion is stopped.

Example

```
INTERRUPT ON
DRIVEA(7,100)            'Interrupt signal turns ON during motion.
                         'The 7th joint stops and the process advances
                         'to the next command.
INTERRUPT OFF
F1=DESTEXJ(7)            'The value of F1 will not be 100 but the value
                         'of the angle where the joint stopped.
```

ARRIVE (Statement)

Function

Defines the motion ratio relative to the programmed full travel distance to the target point in order to make the current program stand by to execute the next step until the robot reaches the defined motion ratio.

Syntax

```
ARRIVE<MotionRatio>
```

Description

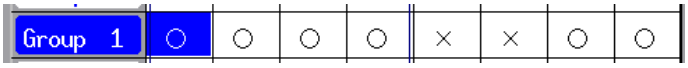
This subsection explains the functions relevant to extended-joints. For others, refer to the PROGRAMMER'S MANUAL, Section 12.1 "ARRIVE."

ARRIVE takes effect only for robot joints. Therefore, when operating only extended-joints by DRIVE or DRIVEA, this ARRIVE command does not work.

Related commands

Example

If an arm group is set in 4-joint robots as shown below, the program sample below works as follows.




```
PROGRAM PRO1
TAKEARM 1
DRIVE(1,30)(7,30),NEXT
ARRIVE 50
MOVE P,P0 EX((7,30)),NEXT
ARRIVE 50
DRIVE(7,30),NEXT
ARRIVE 50
END
```

Since the previous command involves robot joints also, the ARRIVE works.

Synchronized motion with extended-joints. So the ARRIVE will works.

Since the previous DRIVE command involves an extended-joint only, the ARRIVE does not work and the process advances to the next line.

SPEED (Statement)

Function

Specifies the internal composite speed of joints included in a currently held arm group.

Syntax

```
SPEED<SpeedRatio>
```

Description

The <SpeedRatio> should be the target ratio (%) of the maximum internal composite speed of joints in a currently held arm group. The entry range is from 0.1 to 100 of a real number.

Note: If a value 0.1 or less but greater than 0 is specified to <SpeedRatio>, no error will result, but the actual speed may be different from the specified speed.

The actual speed is equal to (external speed x internal speed ÷ 100).

If the SPEED is specified, the JSPEED, ACCEL, DECEL, JACCEL and JDECEL will be set automatically.

Example: If you write SPEED 50, the following will be set automatically:

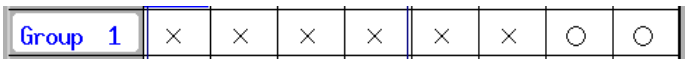
```
JSPEED 50 (same value as SPEED)
ACCEL 25 (SPEED*SPEED÷100)
JACCEL 25 (SPEED*SPEED÷100)
DECEL 25 (SPEED*SPEED÷100)
JDECEL 25 (SPEED*SPEED÷100)
```

If you write SPEED without getting any arm group beforehand, an error will result.

Related commands

ACCEL, DECEL, and JSPEED

Example




```
PROGRAM PRO1
TAKEARM 1        'Get Arm Group 1 (including 7th and 8th joints).
SPEED 100        'Specify composite speed of joints (7th and 8th)
                 'involved in Arm Group 1.
END
```

JSPEED (Statement)

Function

Specifies the internal speed of individual joints included in a currently held arm group.

Syntax

```
JSPEED<JointSpeedRatio>
```

Description

The <JointSpeedRatio> should be the target ratio (%) of the maximum internal speed of individual joints in a currently held arm group. The entry range is from 0.1 to 100 of a real number.

Note: If a value 0.1 or less but greater than 0 is specified to <JointSpeedRatio>, no error will result, but the actual speed may be different from the specified speed.

The actual speed is equal to (external speed x internal speed ÷ 100).

If the JSPEED is specified, the JACCEL and JDECEL will be set automatically.

Example: If you write JSPEED 50, the following will be set automatically:

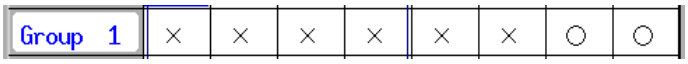
```
JACCEL 25 (JSPEED*JSPEED÷100)
JDECEL 25 (JSPEED*JSPEED÷100)
```

If you write JSPEED without getting any arm group beforehand, an error will result.

Related commands

JACCEL, JDECEL and SPEED

Example




```
PROGRAM PRO1
TAKEARM 1          'Get Arm Group 1 (involving 7th and 8th joints).
JSPEED 100         'Specify speed of joints (7th and 8th) involved
                   'in Arm Group 1.
END
```

ACCEL (Statement)

Function

Specifies the internal composite acceleration/deceleration of joints included in a currently held arm group.

Syntax

```
ACCEL<AccelerationRatio>[,<DecelerationRatio>]
```

Description

The <AccelerationRatio> or <DecelerationRatio> should be the target ratio (%) of the maximum internal composite acceleration or deceleration of joints in a currently held arm group, respectively. The entry range is from 0.0001 to 100 of a real number.

Note: If a value 0.0001 or less but greater than 0 is specified, no error will result, but the actual acceleration may be different from the specified one.

The actual acceleration/deceleration is equal to (external x internal acceleration or deceleration ÷ 100).

If the ACCEL is specified, the JACCEL will be set automatically.

Example: If you write ACCEL 50, the JACCEL 50 (same value as ACCEL) will be set automatically:

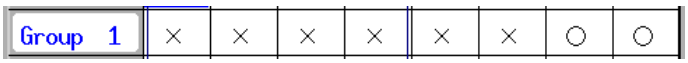
```
JACCEL 25 (JSPEED*JSPEED÷100)
JDECEL 25 (JSPEED*JSPEED÷100)
```

If you write ACCEL without getting any arm group beforehand, an error will result.

Related commands

DECEL and SPEED

Example




```
PROGRAM PRO1
TAKEARM 1          'Get Arm Group 1 (involving 7th and 8th joints).
ACCEL 100          'Specify composite acceleration ratio of joints
                   '(7th and 8th) involved in Arm Group 1.
END
```

JACCEL (Statement)

Function

Specifies the internal acceleration and deceleration of individual joints included in a currently held arm group.

Syntax

```
JACCEL<JointAccelerationRatio>[,<JointDecelerationRatio>]
```

Description

The <JointAccelerationRatio> or <JointDecelerationRatio> should be the target ratio (%) of the maximum internal acceleration or deceleration of individual joints in a currently held arm group, respectively. The entry range is from 0.0001 to 100 of a real number.

Note: If a value 0.0001 or less but greater than 0 is specified, no error will result, but the actual acceleration may be different from the specified one.

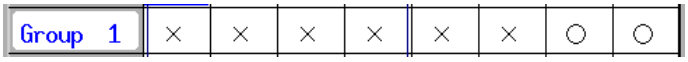
The actual acceleration/deceleration is equal to (external x internal acceleration or deceleration ÷ 100).

If you write JACCEL without getting any arm group beforehand, an error will result.

Related commands

JDECEL, JSPEED, and SPEED

Example




```
PROGRAM PRO1
TAKEARM 1         'Get Arm Group 1 (involving 7th and 8th joints).
JACCEL 100        'Specify acceleration ratio of joints (7th and 8th)
                  'involved in Arm Group 1.
END
```

DECEL (Statement)

Function

Specifies the internal composite deceleration of joints involved in a currently held arm group.

Syntax

```
DECEL<DecelerationRatio>
```

Description

The <DecelerationRatio> should be the target ratio (%) of the maximum internal composite deceleration of joints in a currently held arm group. The entry range is from 0.0001 to 100 of a real number.

Note: If a value 0.0001 or less but greater than 0 is specified, no error will result, but the actual deceleration may be different from the specified one.

The actual deceleration is equal to (external x internal deceleration ÷ 100).

If the DECEL is specified, the JDECEL will be set automatically.

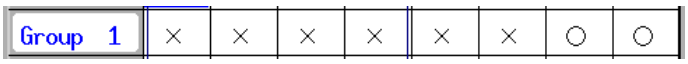
Example: If you write DECEL 50, the JDECEL 50 (same value as DECEL) will be set automatically:

If you write DECEL without getting any arm group beforehand, an error will result.

Related commands

ACCEL and SPEED

Example




```
PROGRAM PRO1
TAKEARM 1        'Get Arm Group 1 involving 7th and 8th joints.
DECEL 100        'Specify deceleration ratio of joints (7th and 8th)
                 'in Arm Group 1.
END
```

JDECEL (Statement)

Function

Specifies the internal deceleration ratio of individual joints included in a currently held arm group.

Syntax

`JDECEL<DecelerationRatio>`

Description

The `<DecelerationRatio>` should be the target ratio (%) of the maximum internal deceleration of individual joints in a currently held arm group. The entry range is from 0.0001 to 100 of a real number.

> **Note: If a value 0.0001 or less but greater than 0 is specified, no error will result, but the actual deceleration may be different from the specified one.**

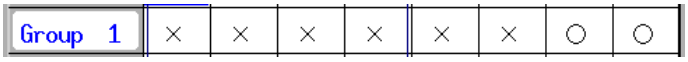
The actual deceleration is equal to (external x internal deceleration ÷ 100).

If you write `JDECEL` without getting any arm group beforehand, an error will result.

Related commands

JACCEL, JSPEED, and SPEED

Example




```
PROGRAM PRO1
TAKEARM 1        'Get Arm Group 1 involving 7th and 8th joints.
JDECEL 100       'Specify deceleration ratio of joints (7th and 8th)
                 'involved in Arm Group 1.
END
```

CURSPD (Statement)

Function

Gets the current internal composite speed of joints included in a currently held arm group.

Syntax

CURSPD

Description

Executing this command in a task holding no arm group will return a value of 100.

Related commands

CURACC, CURDEC, CURJACC, CURJDEC, and CURJSPD

Example

```
PROGRAM PRO1
TAKEARM 1          'Get Arm Group 1.
SPEED 70           'Specify composite speed of joints included in
                   'Arm Group 1 at 70.
I0=CURSPD          'Return current speed 70 of Arm Group 1 to I0.
END
```

CURJSPD (Statement)

Function

Gets the current internal speed of individual joints included in a currently held arm group.

Syntax

CURJSPD

Description

Executing this command in a task holding no arm group will return a value of 100.

Related commands

CURACC, CURDEC, CURJACC, CURJDEC, and CURSPD

Example

```
PROGRAM PRO1
TAKEARM 1         'Get Arm Group 1.
JSPEED 70         'Specify speed of individual joints in Arm Group 1
                  'at 70.
I0=CURJSPD        'Return current speed 70 of Arm Group 1 to I0.
END
```

CURACC (Statement)

Function

Gets the current internal composite acceleration of joints included in a currently held arm group.

Syntax

CURACC

Description

Executing this command in a task holding no arm group will return a value of 100.

Related commands

CURDEC, CURJACC, CURJDEC, CURJSPD, and CURSPD

Example

```
PROGRAM PRO1
TAKEARM 1      'Get Arm Group 1.
ACCEL 70       'Specify composite acceleration of joints included in
               'Arm Group 1 at 70.
I0=CURACC      'Return current acceleration 70 of Arm Group 1 to I0.
END
```

CURJACC (Statement)

Function

Gets the current internal acceleration of individual joints included in a currently held arm group.

Syntax

CURJACC

Description

Executing this command in a task holding no arm group will return a value of 100.

Related commands

CURACC, CURDEC, CURJDEC, CURJSPD, and CURSPD

Example

```
PROGRAM PRO1
TAKEARM 1       'Get Arm Group 1.
JACCEL 70       'Specify acceleration of individual joints included
                'in Arm Group 1 at 70.
I0=CURJACC      'Return current acceleration 70 of Arm Group 1 to I0.
END
```

CURDEC (Statement)

Function

Gets the current internal composite deceleration of joints included in a currently held arm group.

Syntax

CURDEC

Description

Executing this command in a task holding no arm group will return a value of 100.

Related commands

CURACC, CURJACC, CURJDEC, CURJSPD, and CURSPD

Example

```
PROGRAM PRO1
TAKEARM 1       'Get Arm Group 1.
DECEL 70        'Specify composite deceleration of joints included
                'in Arm Group 1 at 70.
I0=CURDEC       'Return current deceleration 70 of Arm Group 1 to I0.
END
```

CURJDEC (Statement)

Function

Gets the current internal deceleration of individual joints included in a currently held arm group.

Syntax

```
CURJDEC
```

Description

Executing this command in a task holding no arm group will return a value of 100.

Related commands

CURACC, CURDEC, CURJACC, CURJSPD, and CURSPD

Example

```
PROGRAM PRO1
TAKEARM 1     'Get Arm Group 1.
JDECEL 70     'Specify current deceleration of individual joints
              'included in Arm Group at 70.
I0=CURJDEC    'Return current deceleration 70 of Arm Group 1 to I0.
END
```

INTERRUPT ON/OFF (Statement)

Function

Interrupts the current robot motion.

Syntax

```
INTERRUPT{ON/OFF}
```

Description

INTERRUPT ON and INTERRUPT OFF are used as a pair. In program lines sandwiched by the pair, if an interrupt skip signal turns ON during execution of motion commands, then the robot controller will interrupt execution of the current motion command and proceeds to the next program step.

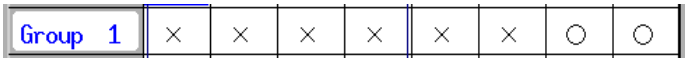
To execute the INTERRUPT command, the task must have gotten an arm group semaphore.

Without INTERRUPT ON written earlier, even if an interrupt skip signal turns ON, the controller will not interrupt execution of any motion commands.

If the program comes to a stop or GIVEARM command is executed, then INTERRUPT will be set to OFF automatically.

Related commands

Example




```
PROGRAM PRO1
TAKEARM 1               'Get Arm Group 1 involving 7th and 8th
                        'extended-joints.
INTERRUPT ON
DRIVE (7,100), (8,30)   'If interrupt skip signal turns ON during
                        'execution of this motion command between
                        'INTERRUPT ON/OFF program lines, then the
                        'controller interrupts the command and
                        'proceeds to the next program step.
INTERRUPT OFF
END
```

Notes

(1) If the controller executes any relative motion command immediately following an interrupt skip, then the subsequent relative motion will start from the position where the robot stops. In program lines sandwiched by the INTERRUPT pair, therefore, use an absolute motion command.

(2) Turning an interrupt skip signal ON interrupts all motion commands sandwiched by the INTERRUPT pair and stops all robot motions. Therefore carefully design your robot system for multitasking if you use the INTERRUPT.

Program Example

	J1	J2	J3	J4	J5	J6	J7	J8
Group 0	○	○	○	○	×	×	×	×
Group 1	×	×	×	×	×	×	○	○
Group 2	○	○	○	○	×	×	○	○
Group 3	×	×	×	×	×	×	×	×
Group 4	×	×	×	×	×	×	×	×

```
PRO1                                  PRO2
TAKEARM 0                             TAKEARM 1
INTERRUPT ON                          INTERRUPT ON
MOVE P,P0      'Motion command        DRIVE(7,30)      'Motion command
  .                                     .
  .                                     .
  .                                     .
INTERRUPT OFF                         INTERRUPT OFF
END                                   END
```

When the controller is running PRO1 and PRO2 programs concurrently in multitasking mode, if an interrupt skip signal turns ON, then the controller will interrupt all commands (MOVE in PRO1 and DRIVE in PRO2 in the above example) running in both programs and make the programs simultaneously proceed to the next step.

POSCLR (Statement)

Function

Forcibly restores the current position of a joint to 0 mm or 0 degree.

Syntax

```
POSCLR<JntNumber>
```

Description

POSCLR forcibly restores the angle of a joint specified by <JntNumber> to 0 mm or 0 degree.

This command is applicable only to joints specified for boundless rotation.

Use this command if a joint keeps on rotating in the same direction so that any of the following happens:

- The current position value becomes too large to handle.
- The current position value jumps to a large negative value (due to overflow or wrap-around of a variable value).

To execute this command, you need to get an arm group including a joint whose position is to be restored to its origin.

Related commands

Example

```
PROGRAM PRO1
TAKEARM 1
DRIVEA (7,100)   'Move 7th joint to an angle of 100 degrees.
POSCLR 7         'Force restore the current angle of the 7th
                 'joint to 0 degree.
END
```

Notes

(1) This command is not applicable to robot joints (Only to extended-joints).

(2) The controller runs this command after the robot has completely stopped. Therefore, any pass motion command written preceding POSCLR will cause no pass motion.

(3) The Step Back function cannot return the program control back to any command written preceding the POSCLR command.

mvSetPulseWidthJnt (Library)

Function

Sets the encoder pulse count for an allowable positioning error for a specified extended-joint.

Syntax

`mvSetPulseWidthJnt(<JntNumber>,<AllowablePulseCount>)`

Description

This library sets a value specified by `<AllowablePulseCount>` for an extended-joint specified by `<JntNumber>`.

If the actual count of positioning error pulses issued by a joint motor encoder is within `<AllowablePulseCount>` at execution of a motion command with `@E` option, the controller will recognize that the joint is on halt.

Macro definition

File <pacman.h> is required.

Related commands

`mvResetPulseWidthJnt` and `mvSetPulseWidth`

Notes

This `mvSetPulseWidthJnt` library is applicable to 7th and 8th extended-joints only. For robot joints, use the `mvSetPulseWidth` library.

Example

```
CALL mvSetPulseWidthJnt(7,10)     'Set allowable error pulse count
                                  'for 7th extended-joint at 10.
```

mvResetPulseWidthJnt (Library)

Function

Resets the encoder pulse count for an allowable positioning error for a specified extended-joint to the default.

Syntax

```
mvResetPulseWidthJnt(<JntNumber>)
```

Description

This library resets the current encoder pulse count specified for an allowable positioning error for an extended-joint specified by <JntNumber> to the default value of 20.

For details about the encoder pulse count for an allowable positioning error,

Macro definition

File <pacman.h> is required.

Related commands

mvResetPulseWidth and mvSetPulseWidthJnt

Notes

This mvResetPulseWidthJnt library is applicable to 7th and 8th extended-joints only. For robot joints, use the mvResetPulseWidth library.

Example

```
CALL mvResetPulseWidthJnt(7)    'Reset allowable error pulse count
                                'for 7th extended-joint to the default.
```

SetCycloidJnt (Library)

Function

Enters a specified extended-joint into the cycloid mode where the controller suppresses the peak of overshoot and residual oscillation that would occur in an end motion.

Syntax

```
SetCycloidJnt(<JntNumber>)
```

Description

(1) In the cycloid mode, the controller will automatically reduce the deceleration ratio currently set for an extended-joint specified by <JntNumber> so as to suppress the peak of overshoot and residual oscillation. The extended-joint will smoothly arrive at the target position.

(2) In the cycloid mode, the controller will take longer operation time than the normal mode. Use this library only when the robot operation can allow timing margins for all related operations.

(3) Once specified, the cycloid mode will remain in effect for the specified extended-joint even driven by any other programs. To reset the cycloid mode, use the ResetCycloidJnt library.

(4) When the specified extended-joint is synchronized with the robot operation, this library will not be called, so the extended-joint will operate according to the robot settings.

(5) If more than one extended-joint runs synchronously, all of those joints will enter the cycloid mode once any one of them has entered the cycloid mode by SetCycloidJnt. This case cannot select joints to be involved.

Macro definition

File <pacman.h> is required.

Related commands

ResetCycloidJnt, ResetCycloid, and SetCycloid

Notes

Before calling this library, you need to hold an arm group semaphore. The TAKEARM should execute beforehand.

This SetCycloidJnt library is applicable to extended-joints only. For robot joints, use the SetCycloid library.

Example

```
Call SetCycloidJnt(8) 'Make 8th extended-joint enter cycloid mode.
```

ResetCycloidJnt (Library)

Function

Cancels the cycloid mode set for a specified extended-joint and restores the normal mode.

Syntax

```
ResetCycloidJnt(<JntNumber>)
```

Description

`ResetCycloidJnt` cancels the cycloid mode set for a joint specified by `<JntNumber>` and restores the normal mode. To set the cycloid mode again, the `SetCycloidJnt` should execute again.

Macro definition

File <pacman.h> is required.

Related commands

`ResetCycloid`, `SetCycloidJnt`, and `SetCycloid`

Notes

Before calling this library, you need to hold an arm group semaphore. The `TAKEARM` should execute beforehand.

This `ResetCycloidJnt` library is applicable to extended-joints only. For robot joints, use the `ResetCycloid` library.

Example

```
CALL ResetCycloidJnt (8) 'Release cycloid mode set for 8th
                         'extended-joint.
```

ResetCurLmt (Library for Ver.1.2 or later)

Function

Releases the drive current limit set for a specified joint motor.

Syntax

`ResetCurLmt(<JntNumber>)`

Description

`ResetCurLmt` releases the drive current limit set for the motor of a joint specified by `<JntNumber>`. The motor drive current limit and positioning error allowances will revert to the defaults.

[For Ver. 1.4 or earlier] If you set "0" to `<JntNumber>`, the drive current limit set for all joints will revert to the default.

[For Ver. 1.5 or later] If you set "0" to `<JntNumber>`, the drive current limit set for all joints involved in an arm group semaphore held by the current task running `ResetCurLmt`, will revert to the default.

Macro definition

File <pacman.h> is required.

Related commands

`ResetEralw` and `SetCurLmt`

Notes

(1) When running this library to release the current limit, the controller will remove preset error allowances concurrently. If any external reaction causes any angle error in joint positioning, then the controller's processing time for the removal will vary depending upon the speed and acceleration/deceleration ratio previously set. If you need to reduce the processing time, set the ratio as high as possible.

(2) This library is executable even if the motor power is off. To do so, run the program as written below after terminating a current task holding an arm semaphore.

```
PRO999
TAKEARM
CALL ResetCurLmt(0)
END
```

(3) [For Ver. 1.4 or earlier] Write this library in a TAKEARMed task that has got robot arm semaphore. If you specify any joints not in the arm semaphore to `<JntNumber>`, then error [21F7 Cannot take arm semaphore] will result.

[For Ver. 1.5 or later] Write this library in a TAKEARMed task that has got an arm group. If you specify any joints not included in the arm group to `<JntNumber>`, then error [27D* Cannot take J* semaphore] will result.

Example

```
CALL ResetCurLmt(0)     'Release the drive current limit set for all
                        'joints.
CALL ResetGrvOffset     'Reset compensation for gravity torque applied
                        'to all joints.
```

ResetEralw (Library for Ver. 1.2 or later)

Function

Resets the positioning error allowance of a specified joint to the initial value.

Syntax

```
ResetEralw(<JntNumber>)
```

Description

`ResetEralw` resets the positioning error allowance set for a joint specified by `<JntNumber>` to the initial value.

[For Ver. 1.4 or earlier] If you set "0" to `<JntNumber>`, the positioning error allowance set for all joints will revert to the default.

[For Ver.1.5 or later] If you set "0" to `<JntNumber>`, the positioning error allowance set for all joints involved in an arm group semaphore held by the current task running `ResetEralw`, will revert to the default.

Macro definition

File <pacman.h> is required.

Related commands

`ResetCurLmt` and `SetEralw`

Notes

(1) [For Ver. 1.4 or earlier] Write this library in a TAKEARMed task that has got robot arm semaphore. If you specify any joints not in the arm semaphore to `<JntNumber>`, then error [21F7 Cannot take arm semaphore] will result.

[For Ver. 1.5 or later] Write this library in a TAKEARMed task that has got an arm group. If you specify any joints not included in the arm group to `<JntNumber>`, then error [27D* Cannot take J* semaphore] will result.

(2) Running the current limit release library `ResetCurLmt` also resets the positioning error allowance to the initial value.

Example

```
CALL ResetEralw(0)     'Reset positioning error allowance of all
                       'joints to initial value.
```

3.2 Setting the Extended-Joint Parameters

To use extended-joints, you need to set extended-joint parameters beforehand. There are two types of extended-joint parameters as described below, which can be set by using the teach pendant.

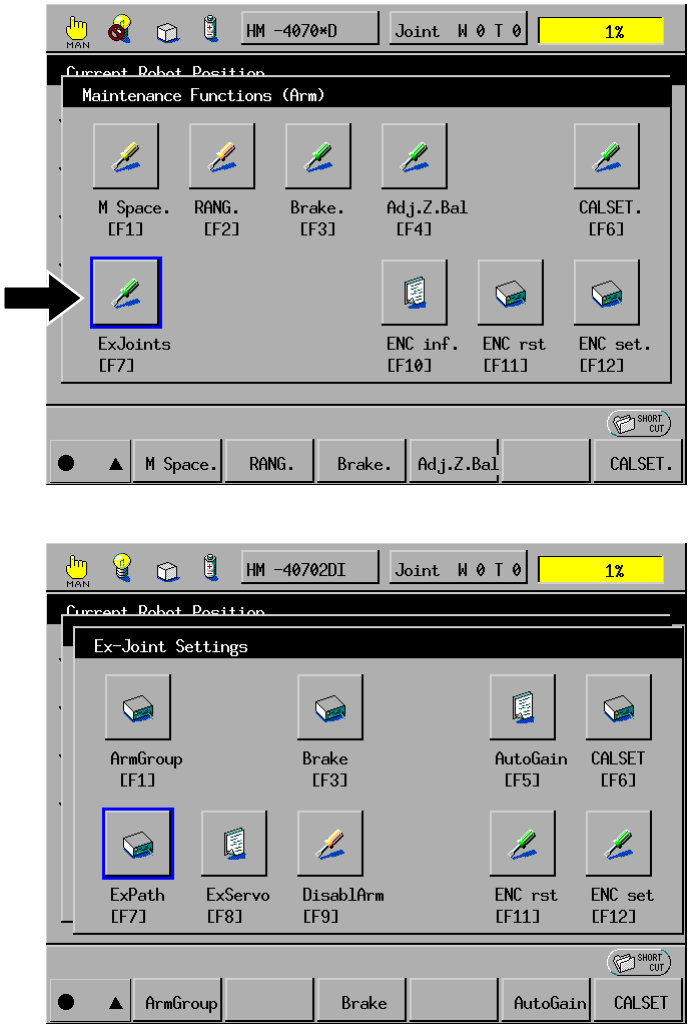
(1) Extended-joint path parameters, which are provided for motion definitions (including speed, acceleration, and range of motion) of extended-joints.

(2) Extended-joint servo parameters, which are provided for setting the gain and others of extended-joint servo system.

3.2.1 Calling up "Path Parameters for Ex-Joint" and "Servo Parameters for Ex-Joint" Windows

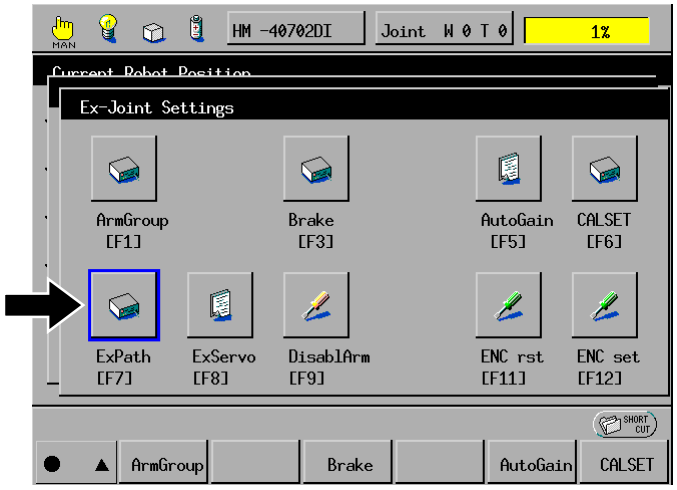
(1) Calling up the "Ex-Joint Settings" window

Top Screen—[F2 Arm]—[F12 Maint.]—[F7 ExJoints]



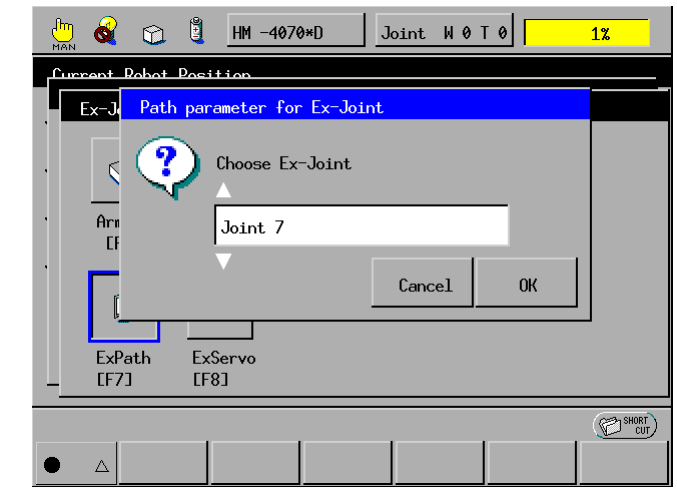
"Ex-Joint Settings" Window

(2) Changing the extended-joint path parameters

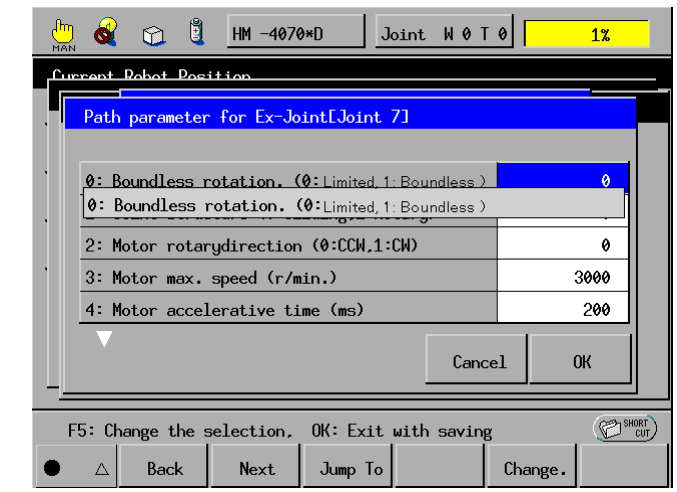


On the "Ex-Joint Settings" window shown above, press [F7 ExPath].

The "Path Parameters for Ex-Joint" window will appear as shown below. Choose the target joint (J7 in this example) by using the cursor keys or jog dial.



Press [OK]. The "Path Parameters for Ex-Joint" window will appear as shown below. Change the path parameters and press [OK]. (For detailed setting procedure, refer to Subsection 3.2.2.)



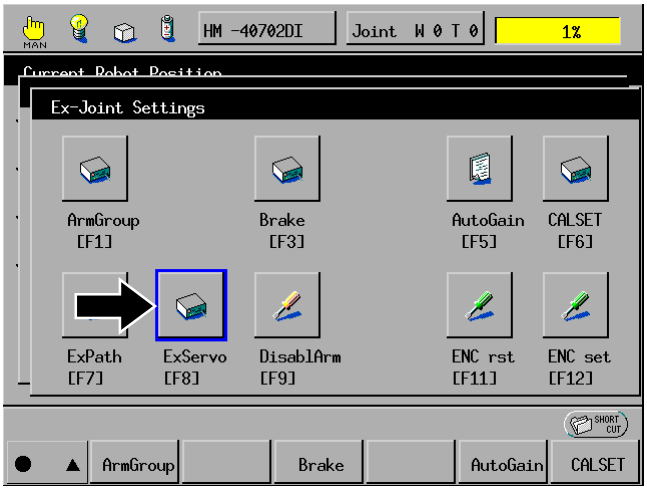
The path parameters are listed in Table 3.1 below.

NOTE: Some parameters will take effect after the controller power is turned off and then on.

Table 3.1 Extended-Joint Path Parameters

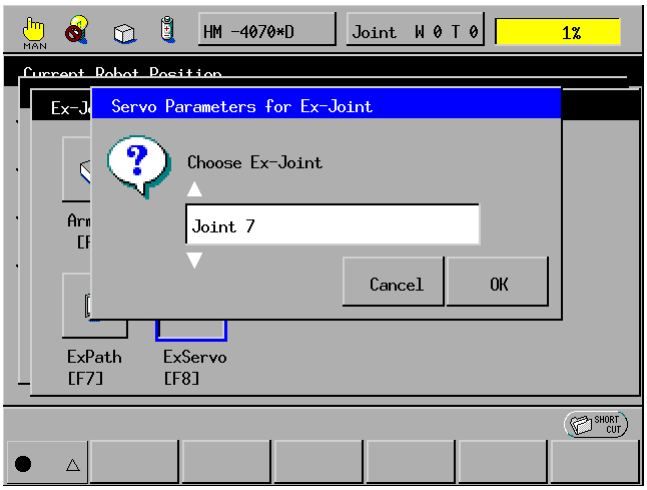
Parameter name	Entry range	Factory default	Unit	Description	Remarks	Controller restart
Boundless rotation (0: Limited, 1: Boundless)	0 or 1	0		If you want to rotate the motor 16384 times or more in the same direction, set this parameter to 1.	If 1 is set, the encoder joint number setting switch must be set to Unlatch. (Refer Sub-section 3.2.4.4.)	Needed
Joint structure (0: Sliding, 1: Rotary)	0 or 1	1		If your optional mechanism to be connected to the specified motor has a linear joint, then set 0; if a rotary joint, set 1.		Needed
Motor rotation direction (0: CCW, 1: CW)	0 or 1	0		To convert the CCW rotation of the specified motor (when viewed from the load side) to the positive direction movement of the connected mechanism, set 0; to convert it to the negative one, set 1.		Needed
Motor max. speed (rpm)	1 to 5000	3000	rpm	Set the maximum speed of the specified motor.		Needed
Motor acceleration time (ms)	1 min.	200	ms	Set the motor acceleration time required for the specified motor to reach the maximum speed.		Needed
Gear ratio or lead (mm/r)	0.00001 min.	100	For lead: mm/r	For rotary joints, set the deceleration ratio (motor rotation/joint rotation). For linear joints, set the lead (movement) per motor rotation.	Up to 100,000 may be set. But if a large value is set, the entered value may be different from the displayed one due to overflow.	Needed
Motion limit detection (0: Invalid, 1: Valid)	0 or 1	1		To make the controller check the motion limit and issue an error if the specified joint is out of the range, set 1.		Needed
Positive motion limit (deg.) (mm)		360	For rotary joints: degrees For linear joints: mm	Set the positive motion limit.		Not needed
Negative motion limit (deg.) (mm)		-360	For rotary joints: degrees For linear joints: mm	Set the negative motion limit.		Not needed
CALSET position		0	For rotary joints: degrees For linear joints: mm	Set the CALSET reference position.		Not needed

(3) Changing the extended-joint servo parameters

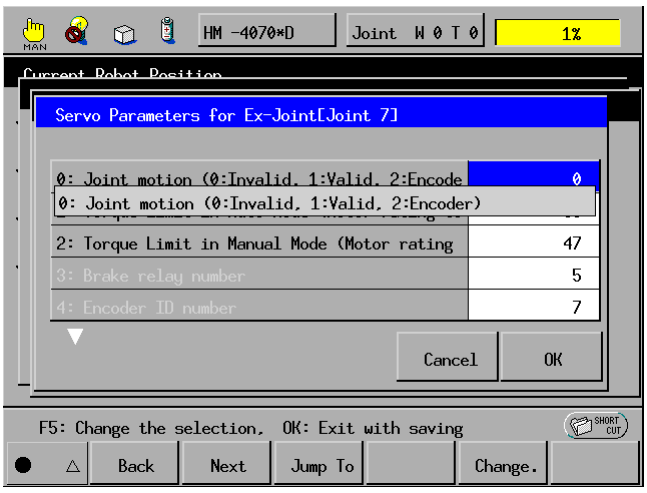


On the "Ex-Joint Settings" window shown above, press [F8 ExServo].

The "Servo Parameters for Ex-Joint" window will appear as shown below. Choose the target joint (J7 in this example) by using the cursor keys or jog dial.



Press [OK]. The "Servo Parameters for Ex-Joint" window will appear as shown below. Change the servo parameters and press [OK]. (For detailed setting procedure, refer to Subsection 3.2.2.)



The servo parameters are listed in Table 3.2 below.

NOTE: Some parameters will take effect after the controller power is turned off and then on.

Table 3.2 Extended-Joint Servo Parameters

Parameter name	Entry range	Factory default	Unit	Description	Remarks	Controller restart
Joint motion (0: Invalid, 1: Valid, 2: Encoder)	0 to 2			To connect and drive a specified motor, set 1; to use the encoder only, set 2.	If "2: Encoder" is selected, turning the motor on will release the brake. CAUTION: If any unbalanced load is applied, the joint will move towards the load.	Needed
Torque limit in auto mode (Motor rating torque %)	0 to 400	300	%	Set the torque limit value to be applied in auto mode.		Not needed
Torque limit in manual mode (Motor rating ratio %)	0 to 400	150	%	Set the torque limit value to be applied in manual mode.		Not needed
Brake relay number	0 to 8			Displays the motor brake relay number.	No change allowed.	
Encoder ID number	1 to 8			Displays the encoder ID number.	No change allowed.	
Power module slot number	1 to 8			Displays the power module slot number.	No change allowed.	
Positional loop gain	1 min.	64		Set the response of the position control system. Increasing the value will decrease the positioning time.	The positioning loop gain can be converted in unit by Formula 3.2.3-1 given in Subsection 3.2.3.	Not needed
Positional loop feed forward gain (%)	0 to 100	0	%	Set the loop forward gain of the position control system. Increasing the value will decrease a positioning error and increase the response, but overshoot will easily occur.		Not needed
Positioning error allowance (pulse)	1 min.	30000		Set the allowable value of positioning error. If a positioning error exceeding this allowable value occurs, an error will result.	Set the value that meets Formula 3.2.3-2 given in Subsection 3.2.3.	Not needed

Parameter name	Entry range	Factory default	Unit	Description	Remarks	Controller restart
Speed proportional gain	0 min.	200 (for 200W or less) 400 (for 400W or greater)		Set the response of the speed control system. Increasing the value will enable you to set a higher value of the positional loop gain.	The speed loop proportional gain can be converted to the speed response frequency in Hz by Formula 3.2.3-3 given in Subsection 3.2.3.	Not needed
Speed integral gain	0 min.	10		Set the integral compensation gain of the speed control system. Increasing the value will converge the speed deviation at the time of stop faster.	The speed loop integral gain can be converted to the time constant by Formula 3.2.3-4 given in Subsection 3.2.3.	Not needed
Filter parameter	0 to 15	8		Set the primary delay filter band in the torque instruction section. Increasing the value will decrease the time constant of the low-pass filter.		Not needed
Torque offset setting (Motor rating ratio %)	0 to 100	0	%	Set the torque offset value of the torque instruction value. If the motor undergoes any unbalanced load (movement towards the load), this offset will compensate it.	If you enable the gravity offset in auto gain tuning, the torque offset value will be automatically set.	Not needed
Motor capacity (1: 50W, 2: 100W, 3: 200W, 4: 400W, 5: 750W, 6: 1.5kW)	1 to 6			Display the connected motor capacity	No change allowed.	

3.2.2 Detailed Description of Extended-Joint Parameter Setting

The extended-joint path parameters and extended-joint servo parameters should be set with extended-joint motors being connected.

(1) Resetting the encoder of an extended-joint motor

The encoder is not connected with a backup battery at the time of shipment, so the error message "J* encoder system down" or "J* encoder speed exceeded" will appear.

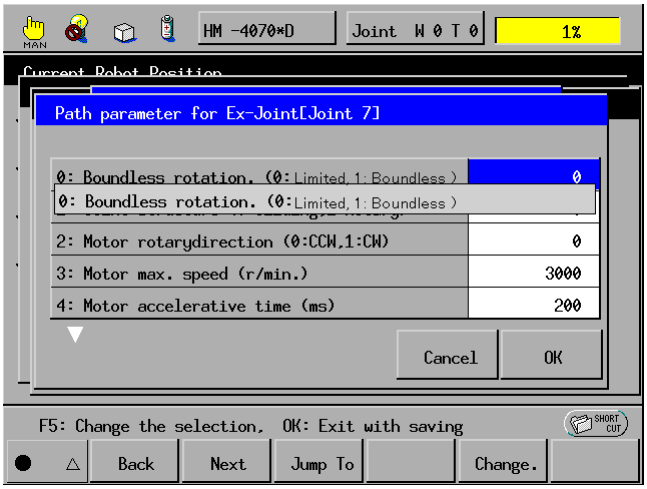
If this happens, reset the encoder (refer to Subsection 3.2.4.3) and restart the robot controller.

NOTE: When resetting the encoder, connect only the motor cable to the extended-joint control box and be sure to disconnect the robot cable from it. This is to prevent the robot motor from getting reset.

(2) Setting the path parameters

For the calling-up procedure of the "Path Parameters for Ex-Joint" window, refer to Subsection 3.2.1.

(2-1) Boundless rotation

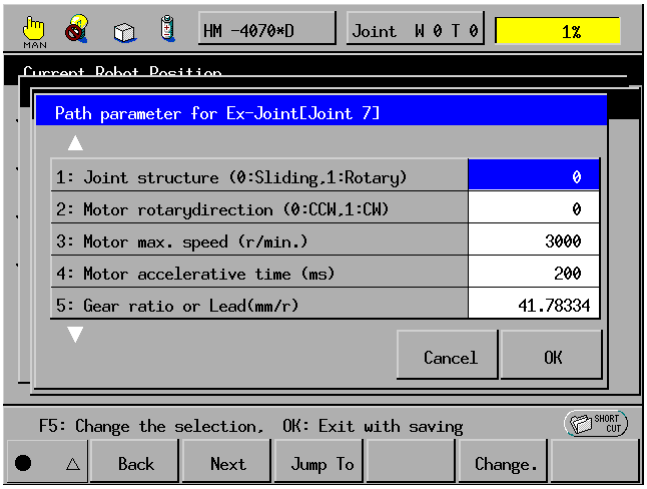


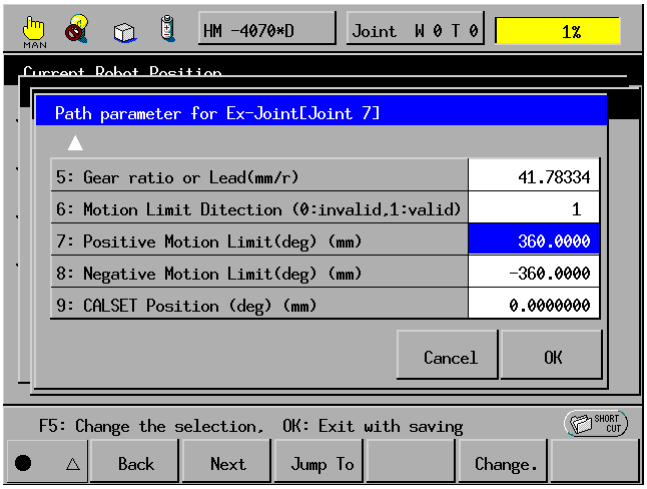
To rotate the motor 16,384 times or more in the same direction, you need to set the boundless rotation to "1: Boundless."

Specifying the boundless rotation requires the overflow error of the encoder to be set to "Unlatch." See the "Joint Number Setting Switch" window in Subsection 3.2.4.4 "Setting a joint ID to an extended-joint encoder."

NOTE: When setting the encoder ID, be sure to connect only the motor whose encoder joint ID should be set, to the extended-joint control box. Disconnect the robot cable and other motor cables from the extended-joint control box. The encoder ID setting operation will set all encoders being connected to the same ID.

(2-2) Setting the motion conditions





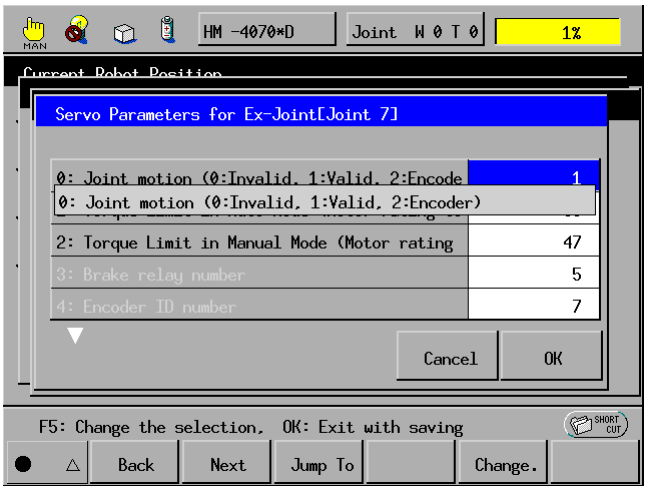
Set the motion-relation parameters--joint structure, motor rotation direction, motor maximum speed, motor acceleration time, gear ratio or lead, motion limit detection, positive motion limit, negative motion limit, and CALSET reference position.

(3) Setting the extended-joints servo parameters

For the calling-up procedure of the "Servo Parameters for Ex-Joint" window, refer to Subsection 3.2.1.

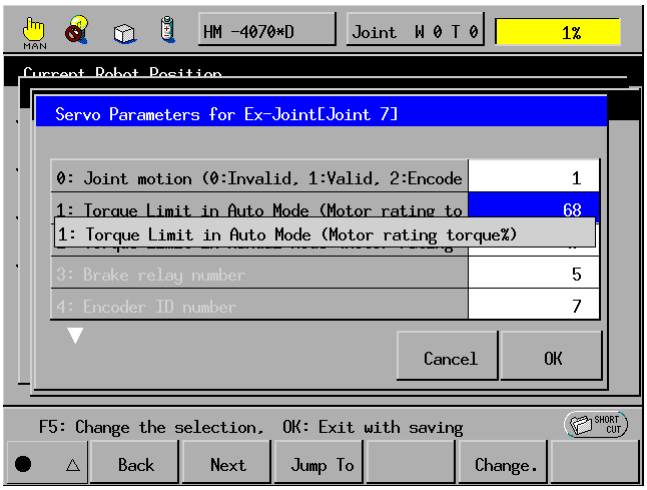
(3-1) Setting the joint motion

Set the joint motion to "1: Valid."



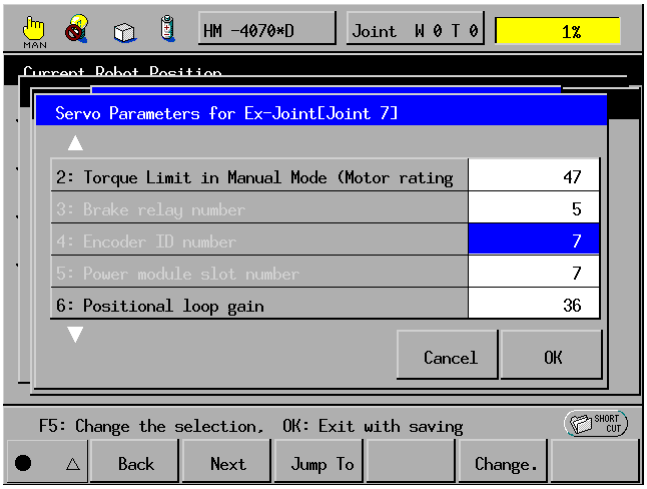
(3-2) Setting the torque limits

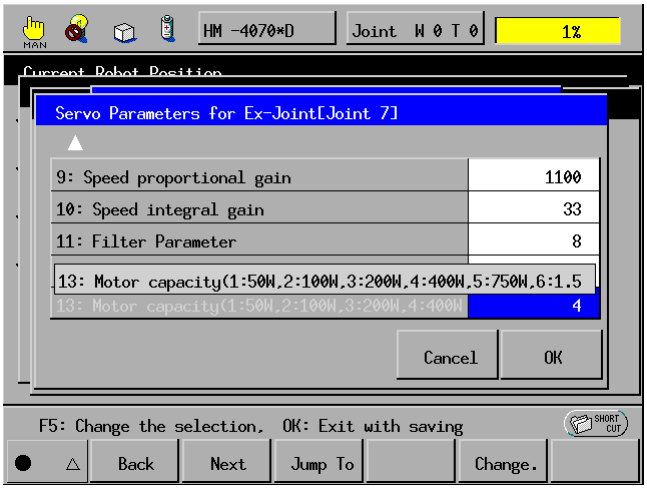
Set the torque limits in auto mode and in manual mode.



(3-3) Checking the encoder ID number, power module slot number, and motor capacity

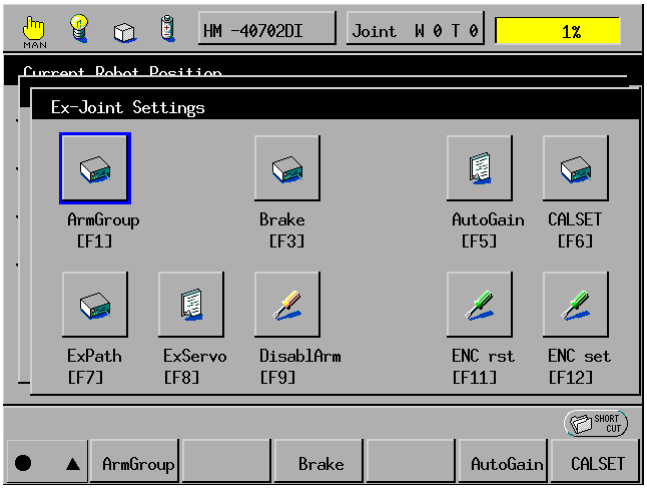
Checks that the encoder ID number and power module slot number match the joint number. Also, check that the motor capacity is selected correctly.





(4) Registering extended-joints in an arm group

Register extended-joints into an arm group. Refer to Subsection 3.1.1 "Operating Procedures of the Extended-Joint Function, [4] Operating extended-joints by program."



After completion of steps (1) to (4), restart the robot controller.

(5) Checking the wiring

(5-1) Checking the brake wiring

If the extended-joint motor has a brake, release the brake and check that the brake of the specified joint will be released. For the brake releasing procedure, refer to Subsection 3.2.4.2 "Releasing and locking an extended-joint brake."

NOTE: Releasing the brake of a robot joint may drop the arm. Take care not to release the brakes of robot joints.

(5-2) Checking the encoder wiring

After releasing the brake of the extended-joint motor, apply external force to the motor and check that the data of the joint corresponding to the motor will change in the Current Robot Position window of the teach pendant.

(5-3) Checking the motor wiring

Turn the extended-joint motor on, set the motor speed at SP10, and check that you may drive the extended-joint manually in Joint mode.

If the motor vibrates abnormally or stops due to any error, check the wiring of the motor. If the wiring is correct, gradually decrease the positional loop gain and speed proportional gain of the extended-joint servo parameters.

(6) Executing CALSET

Release the brake of the extended-joint motor and move the optional mechanism connected to the motor to the CALSET reference position. Then execute CALSET in the CALSET reference position, referring to Subsection 3.2.4.1 "Performing CALSET operation on extended-joints."

NOTE: Performing the CALSET operation on a robot joint will change the reference angle of the robot. Take care not to execute CALSET on any robot joint.

After completion of CALSET operation, execute CAL operation.

(7) Checking the motion of the optional mechanism connected to the extended-joint motor

Run the optional mechanism connected to the motor manually in Joint mode and check that an error will be detected if the mechanism exceeds the positive or negative motion limit.

Also check that the actual movement amount matches the values displayed in the Current Robot Position window of the teach pendant. If not, check the gear ratio and lead.

3.2.3 Gain Tuning of Extended-Joints

In Subsection 3.2.2, you have set the motion conditions of extended-joints and checked the motion of the optional mechanism connected to the extended-joint motors manually in Joint mode. After that, proceed to the gain tuning for the servo system.

Tune the servo system according to the following two types of tuning methods:

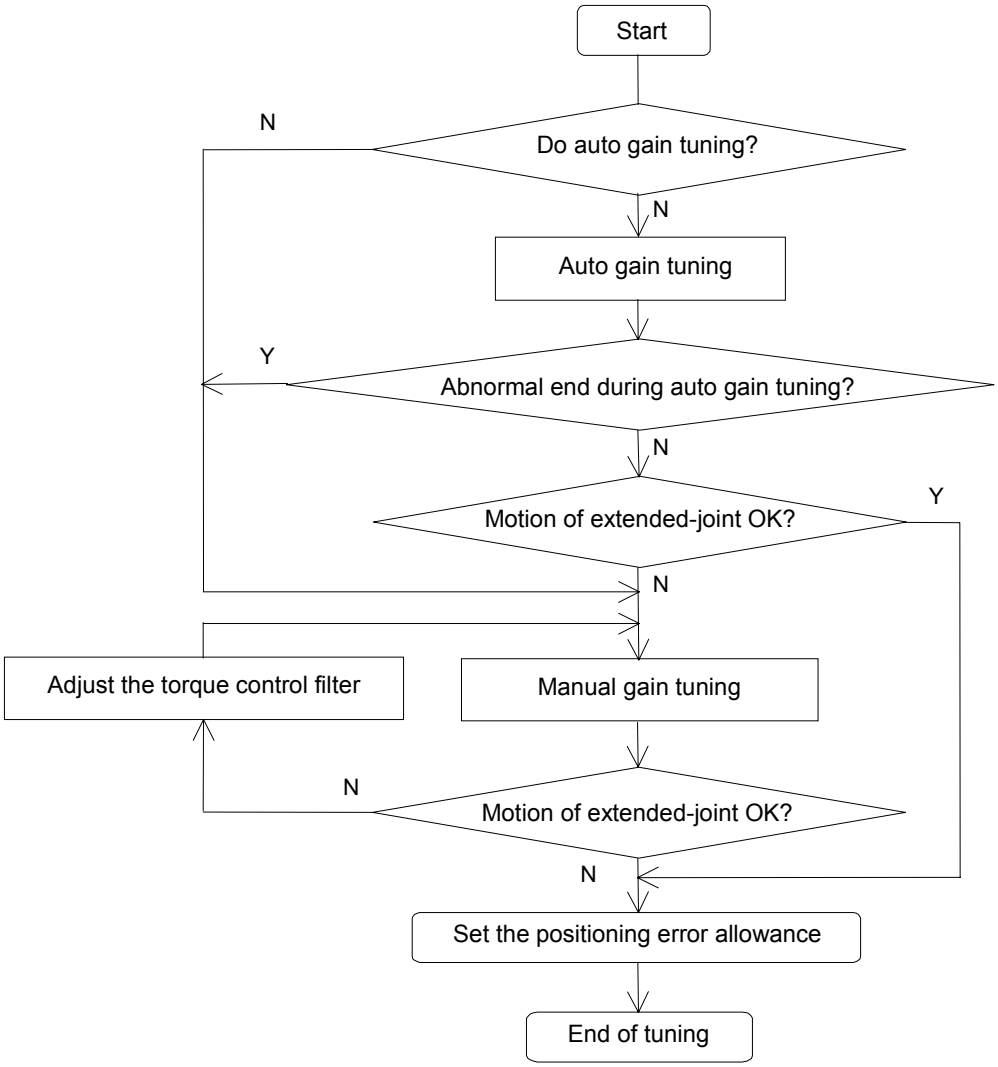
(1) Auto gain tuning

The controller performs acceleration/deceleration operation of the extended-joints according to the default pattern preset in the controller. Based on the motion of the extended-joints in that operation, the controller will estimate the inertia of payload and set the appropriate gain automatically.

(2) Manual gain tuning

The monitor function of the single-joint servo data monitors the motor speed control value, current motor speed, motor angle deviation, and torque control value. According to the monitored results, you may adjust the gain and torque control filter parameters for optimizing the motion of the extended-joints.

Follow the next flowchart to tune the servo system.



3.2.3.1 Auto gain tuning

To implement auto gain tuning, your optional mechanism to be connected to the extended-joint motor should satisfy the requirements given in [1] below. Otherwise, some errors may occur and the auto gain tuning process may be interrupted. If such happens, implement manual gain tuning.

[1] Requirements for implementing auto gain tuning

(1) The inertia of payload should be within 15 times that of the motor and should not deviate greatly.

(2) The rigidity of the torque transmission mechanism (including motor and coupling) to be connected to the extended-joint motor should be high.

(3) The backlash in the torque transmission mechanism should be minimized.

(4) Rotating the motor in CCW and CW directions alternately two times each direction should result in no problem.

[2] Auto gain tuning procedure

(1) Turn the motor power on and perform CAL

NOTE: If in Auto mode or Teach check mode, switch to Manual mode.

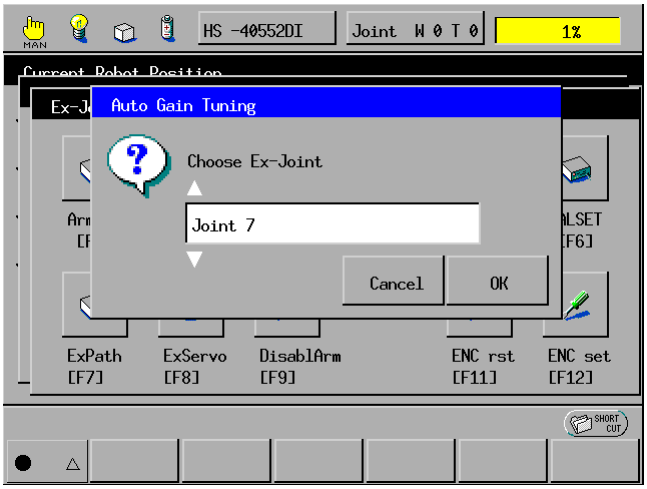
(2) Get out of the motion range so that there will be no problem even if the motor rotates in CCW and CW directions alternately two times each direction.

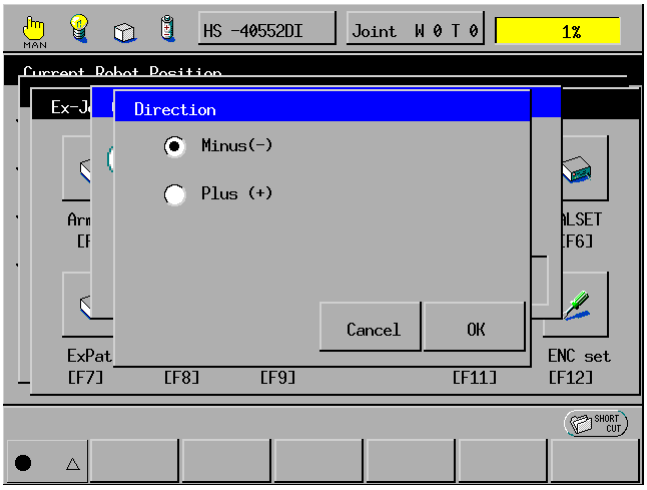
(3) On the teach pendant, call up the Ex-Joint Settings window.

Access: Top Screen—[F2 Arm]—[F12 Maint.]—[F7 ExJoints]

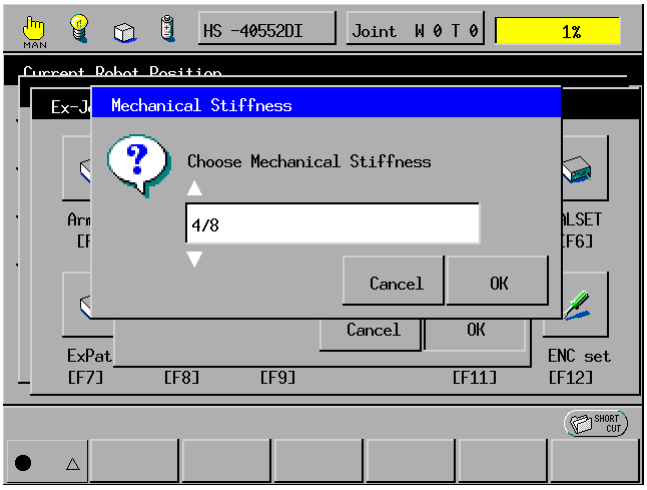
(4) Press [F5 Auto Gain] to call up the Auto Gain Tuning window as shown below.

Choose the joint number that should undergo auto gain tuning and the motor rotation direction.



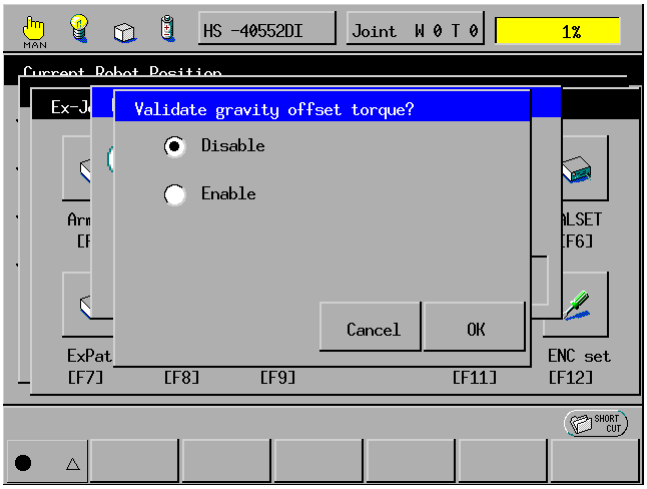


(5) Select the mechanical rigidity, referring to the rigidity reference values listed below.



Types of Torque Transmission Mechanisms	Mechanical Rigidity
Ball screw direct connection	4 to 8
Ball screw with transmission mechanism	3 to 7
Timing belt	3 to 6
Gear or rack & pinion	2 to 6
Other mechanism with low rigidity	1 to 3

(6) Select whether the gravity offset torque should be enabled or disabled.



If an unbalanced load applies to the motor, be sure to enable the gravity offset torque.

NOTE: If you disable gravity offset torque when the motor undergoes any unbalanced load, then the extended-joint will drop in the gravity direction, causing an error. To implement auto gain tuning when an unbalanced load applies to the motor, be sure to enable the gravity offset torque.

If you enable the gravity offset torque for auto gain tuning, the controller will automatically calculate the torque offset included in extended-joint servo parameters. On the Ex-Joint Settings window, press [F8 ExServo] to call up the "Servo Parameters for Ex-Joint" window and then press [OK] to save the calculated torque offset value.

NOTE: If you turn the controller power off without saving the calculated torque offset value, then the value will be lost and the previous value will resume.

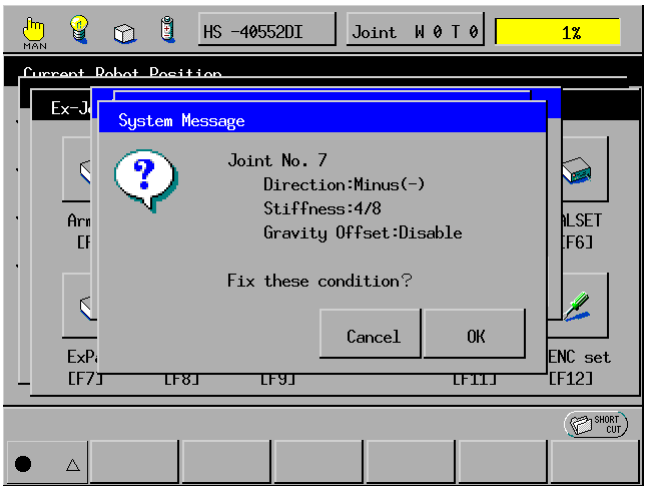
(7) Hold down the deadman switch through all processes of auto gain tuning. Releasing it will interrupt auto gain tuning.

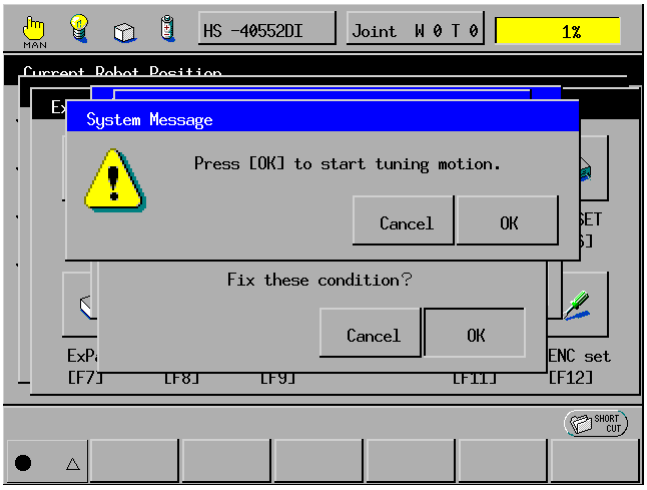
NOTE: During auto gain tuning, do not press any key on the teach pendant except for the deadman switch. Doing so will interrupt auto gain tuning.

NOTE: If the joint motion has been set to [2:Encoder] on the "Servo Parameters for Ex-Joint" window, then the error message "Not executable" will appear during auto gain tuning.

(8) In the dialog box shown below, confirm the conditions and press [OK]. In the confirmation window, press [OK]. Then auto gain tuning will start. The motor rotates in CCW and CW directions alternately two times each direction in two sequences to calculate a temporary servo loop gain.

After that, the motor will repeat the sequence up to 8 times to fine-tune the gain. If the gain is fixed within the eight sequences, auto gain tuning will complete.





(9) After eight sequences of the above fine tuning operation, any of the following messages may display:

"Auto gain tuning warning 1": Overshoot found at the end of motion.

"Auto gain tuning warning 2": Slow settlement found at the end of motion.

"Auto gain tuning warning 3": Low-level oscillation found during motion.

If any of the above messages displays but there is no problem with the joint motion, then finish the gain tuning. If any abnormal noise or vibration is noted and there are some problems with the motion, then change the mechanical rigidity. After that, retry auto gain tuning or proceed to manual gain tuning.

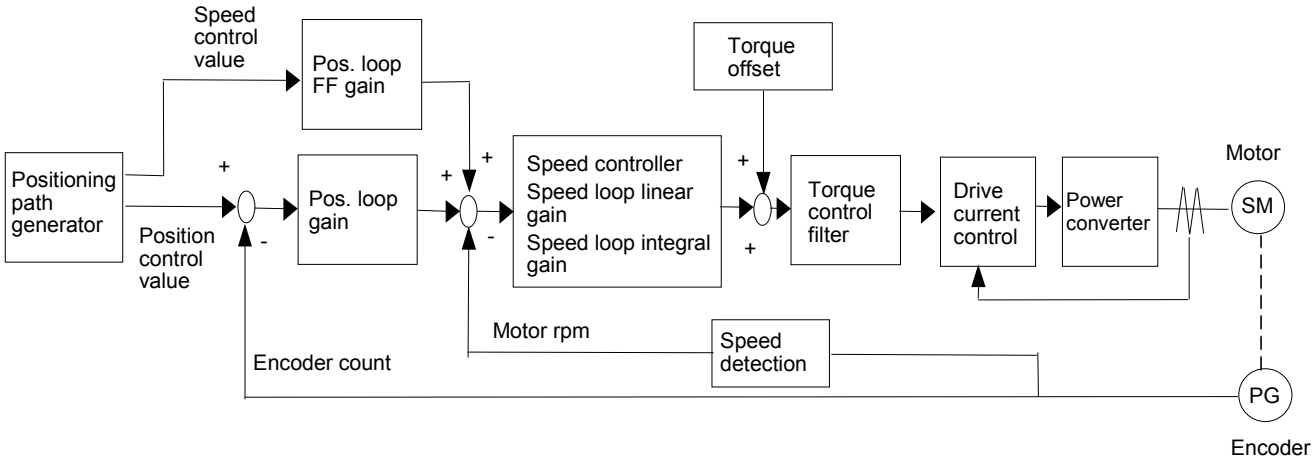
(10) If you set higher mechanical rigidity for transmission mechanism having lower rigidity and vise versa, then an error may occur during auto gain tuning. Change the mechanical rigidity setting and retry auto gain tuning.

3.2.3.2 Manual gain tuning

You can manage the following parameters for manual gain tuning:

(1) Positioning loop gain
(2) Positioning loop feed forward gain
(3) Positioning error allowance
(4) Speed linear gain
(5) Speed loop integral gain
(6) Torque control filter
(7) Torque offset

The block diagram for the servo system is shown below.



An electric servo loop system consists of the three feed back systems--positioning loop, speed control loop, and drive current loop. The inner the loop is, the quicker response required. If the response of an inner loop is not sufficiently high for an outer loop, then the overall system response degrades and vibrations or oscillations may occur in the extended-joint support system.

In this system, the innermost loop is the drive current loop and the outermost loop, the positioning loop.

You need to do gain tuning for the positioning loop and speed control loop. The drive current loop is designed to have sufficiently high response for all applications allowable to the extended-joint support system.

[1] Parameter details

(1) Positioning loop gain

Set the response of the positioning loop. The positioning loop gain is a dimensionless number, so it may be converted to the (1/s) unit according to the following formula:

Positioning loop gain x 125/256 (1/s) **(Formula 3.2.3-1)**

For example, positioning loop gain 32 is equivalent to 15.625 (1/s).

Increasing the positioning loop gain will reduce the positioning time. However, increasing the gain exceeding the natural oscillation frequency of the connected mechanism will easily bring vibration or overshoot. If the natural oscillation frequency is 20 Hz, for instance, set the positioning loop gain to 20 (1/s), that is, approx. 41.

(2) Positioning loop feed forward gain

Set the speed feeding forward value of the positioning loop. Increasing the value will reduce the positioning error and increase the system response. Setting 100 may reduce the positioning error to almost 0 in constant speed operation. However, setting an excessively high value may easily cause vibration or overshoot in the system.

(3) Positioning error allowance

Set the positioning error allowance. If the actual positioning error exceeds the specified allowance, an error will occur. The positioning error allowance should satisfy the following formula:

[Positioning error allowance] > [Maximum motor speed (rpm)] x (1.0-[Positioning loop forward gain (%)] x 0.01)/[Positioning loop gain] x 524288/1875 **(Formula 3.2.3-2)**

(4) Speed control linear gain

Set the response of the speed control system. Increasing the value will make it possible to set a high value to the positioning loop gain, thereby increasing the system response. The speed control linear gain to be set may be converted to the speed response frequency (in Hz) according to the following formula:

Speed response frequency (Hz) = [Speed control linear gain] / [Motor rotor inertia (kgm^2) + Load inertia converted at motor joint (kgm^2)] x [Drive current loop gain] / (2π) **(Formula 3.2.3-3)**

The drive current system gain is as listed below.

Motor Model	Motor Rating	Drive Current System Gain
MQM012T2V	100 W	3.942E-05
MQM022T2V	200 W	9.381E-05
MQM042T2V	400 W	1.264E-04
MQM082T2V	750 W	3.038E-04
MQM152T2U	1.5 kW	2.785E-04

(5) Speed control integral gain

Set the integral compensation gain of the speed control system. You may convert the integral gain of the speed control loop into integral speed loop gain time constant (ms) according to the following formula:

Integral speed loop gain time constant (ms) = 0.5 x [Speed control linear gain] / [Speed loop integral gain] (Formula 3.2.3-4)

Increasing the value will decrease the integral time constant, making the speed error converge faster at the end of joint motion. However, increasing the value for the connected transmission mechanism having lower rigidity will decrease the convergence of residual oscillation at the end of joint motion.

(6) Torque control filter

This value sets the band of the linear delay component for the torque control filter. The table below lists the relationship between the set and the band.

Filter Set Value	3	4	5	6	7	8	9	10	11	12	13	14	15
Band (Hz)	775	540	422	341	280	230	188	152	121	92	66	43	21

(7) Torque offset

This value gives an offset to the torque control value of the extended-joint support system. If the motor undergoes any unbalanced load due to the force of gravity, setting this value will compensate the torque caused by the unbalanced load. The maximum offset value you can set is equal to the rated output torque of the motor.

If you set a large torque offset at once, the connected mechanism may move in the preset direction immediately after the motor power is turned on. Gradually change the torque offset while confirming the current torque value and positioning error waveform in the next item "[2] Monitor of single-joint servo data."

As described in Subsection 3.2.3.1 "Auto gain tuning," the torque offset value will be automatically set if you enable the gravity offset torque in auto gain tuning.

[2] Monitor of single-joint servo data

This function allows you to monitor a specified joint servo data currently set in the robot controller with graphs in real-time.

(1) Monitoring capability

This function is capable of handling up to 1,250 samples of data at once. If the sampling interval is set to 1 ms, then you may monitor the servo data for 1.25 seconds; if 8 ms, you may monitor it for 10 seconds.

The following five types of data may be monitored, two types at a time:

1) Motor speed control value (rpm)

   Shows the sampled control value of motor speed.

2) Current motor speed (rpm)

   Shows the sampled current motor speed.

3) Motor angle deviation (pulse)

   Shows the deviation between the actual motor angle and motor control angle.

4) Torque control value (%)

   Shows the substantial torque control value; that is, (Torque control value - the Torque offset value). The unit is a ratio to the rated motor torque (%).

5) Motor current (%)

   Shows the currently maximal motor drive current between the 3-phase driving lines. The unit is a ratio to the motor rated current.

(2) Defining the monitoring terms

To define the monitoring terms, call the single-joint servo data monitor definition library `SetMonitorCond` in your program. For details, refer to Subsection 3.2.5 "`SetMonitorCond`."

Once monitoring starts, the monitoring terms already defined can not be changed until the monitoring cycle has completed. Define all necessary monitoring terms before starting a monitoring cycle.

(3) Starting and stopping the monitoring cycle

To start monitoring, run the library `StartSrvMonitor`. To end it, run the library `StopSrvMonitor`. To clear the data collected in the monitoring cycle, run the library `ClearSrvMonitor`. For details, refer to Subsection 3.2.5, "`StartSrvMonitor`," "`StopSrvMonitor`," and "`ClearSrvMonitor`."

If the total number of data samples monitored in a monitoring cycle is 1250 or less, all data may be monitored. If it exceeds 1250 samples, the last 1250 data samples before the end of the cycle may be monitored and other data will be discarded.

If any error occurs and the motor being monitored is turned OFF during monitoring, then a maximum of 850 samples before the OFF and 400 samples after that may be monitored.

(4) Graphing the monitored data

Log Manager in WINCAPSII allows you to graph the monitored data on a PC screen. For details, refer to Subsection 4.3.8, "New Features in Log Manager."

(4.1) Read monitored data

To read the monitored data (as Servo Joint Log) into Log Manager, first establish connection between the PC and robot controller in Communications Setting Manager and choose "Import" from the File menu in Log Manger to call up the "Receive Table" dialog box. From the table, select the <Servo Joint Log> and receive it.

(4.2) Plot the graph of monitored data

In Log Manager, choose "Servo Joint Graph" from the Tools menu. Adjust the scale and offset of the graph and check the graphed data.

(5) Saving the monitored data into a file

Log Manager in WINCAPSII allows you to save the monitored data into a CSV file. For details, refer to Subsection 4.3.8, "New Features in Log Manager."

To save the data into a CSV file, choose "Export"—"Servo Joint Data" from the File menu.

[3] Operating procedure for manual gain tuning

(1) Initializing positioning loop gain

First set the positioning loop gain to almost the same value of that calculated from natural frequency of the connected mechanism.

If the natural frequency is 20 Hz, set the positioning loop gain to 41 which equals to 20 × 256/125 as calculated by Formula 3.2.3-1. If the natural frequency is unknown, use the default value 64.

(2) Tuning torque

If almost constant, unbalanced load (e.g., force of gravity) applies to the motor, then set the torque offset calculated from the load.

(3) Obtaining the limit of speed linear loop gain

While increasing the speed linear loop gain gradually, find the upper limit of the loop gain at which the connected mechanism will start producing abnormal noises or oscillations.

(4) Checking the effect of the torque control filter setting

First set the torque control filter parameter to "0" and find the upper limit of speed linear loop gain again. If the speed linear loop gain obtained here is lower than the previous one obtained in step (3), reset it to 8 (default).

(5) Determining the appropriate speed linear loop gain

Apply 80% of the limit obtained in steps (3) and (4) to the speed linear loop on the connected mechanism.

(6) Tuning the speed Integral loop gain

Gradually increase the speed integral loop gain so that the positioning time and peaks of overshoot and undershoot will be minimized to optimize the connected mechanism.

(7) Tuning the positioning loop gain

If the connected mechanism is still oscillatory after carrying out the procedure in step (6), then decrease the positioning loop gain.

If you decrease the positioning time further after tuning in steps (3) through (6), then gradually increase the positioning loop gain to the extent that no noise or oscillation will be produced.

(8) Tuning the positioning loop feed forward gain

If you further decrease the positioning time of the connected mechanism, gradually increase the positioning loop feed forward gain to the extend that no oscillation will be produced.

(9) Checking operations of the connected mechanism in full motion range and in full speed range

Run the connected mechanism in the full motion range while changing the speed gradually. If any abnormal noises or vibrations occur at some particular points, then check whether the mechanism slides evenly.

If any abnormal noise occurs in some particular speed, tune the torque control filter parameter again and check whether the abnormal noise decreases.

If tuning-up of the mechanism and torque control filter parameter cannot suppress abnormal noises, then decrease the speed linear loop gain and speed integral loop gain in the same proportion. (It is convenient to use the quick tuning function for the speed control system gain described in the next item [4].)

If any vibration occurs in some particular speed, decrease the speed integral loop gain, positioning loop gain, and/or positioning loop feed forward gain.

NOTE: Servomotors recommended earlier in this manual will issue torque ripple 4 times per rotation. The torque transmission mechanism may also issue torque ripple specific times per rotation at its output shaft.

Therefore, the frequency of the torque ripple may vary according to the speed so as to become equal to the natural frequency of the connected mechanism.

If vibrations are large in some specific speed, decrease the speed integral loop gain, positioning loop gain and/or positioning loop feed forward gain as well as stated above.

[4] Quick tuning function for speed control system gain

As expressed in Formula 3.2.3-4, the ratio of the speed linear loop gain to speed integral loop gain makes the integral speed loop gain time constant. For fine tuning of the speed control system gain, therefore, change the speed linear loop gain and speed integral loop gain in the same proportion. This simultaneous and proportional adjustment of those gains is "Quick tuning function for speed control system gain." You may use this function with the teach pendant.

(1) Calling up the Quick Loop Gain Tuning screen using the teach pendant

Access: Top Screen—[F2 Arm]—[F6 Aux.]—[F7 Config.]—[F3 Jump To]

Select #59.

(2) Setting a value to "Gain Decreasing Ratio (J*)" for the joint (J*) to be tuned

The gain decreasing ratio is called Tuning Ratio. The robot controller automatically modifies the current speed linear loop gain and speed integral loop gain by the number of "Tuning Ratio" times.

The relationship between the value to be set and the tuning ratio is listed below.

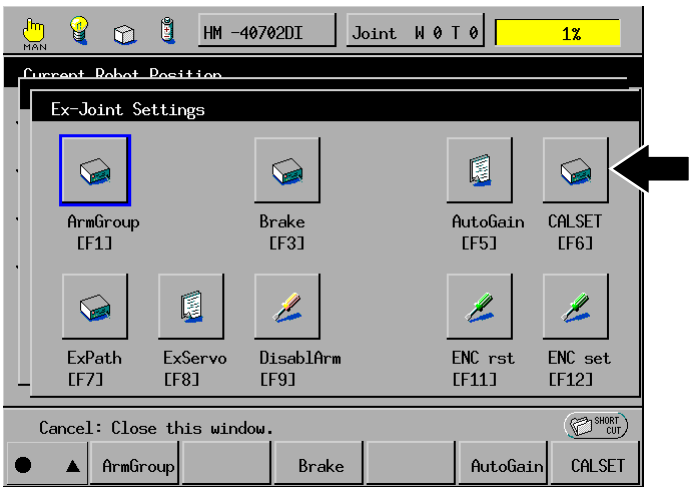
Set Value	-5	-4	-3	-2	-1	0	1	2	3	4	5
Tuning Ratio	1.5	1.4	1.3	1.2	1.1	1	0.9	0.8	0.7	0.6	0.5

3.2.4 Extended-Joints Exclusive Operations

3.2.4.1 Performing CALSET operation on an extended-joint

(1) Calling up the Ex-Joint Settings window

Access: Top Screen—[F2 Arm]—[F12 Maint.]—[F7 ExJoints]



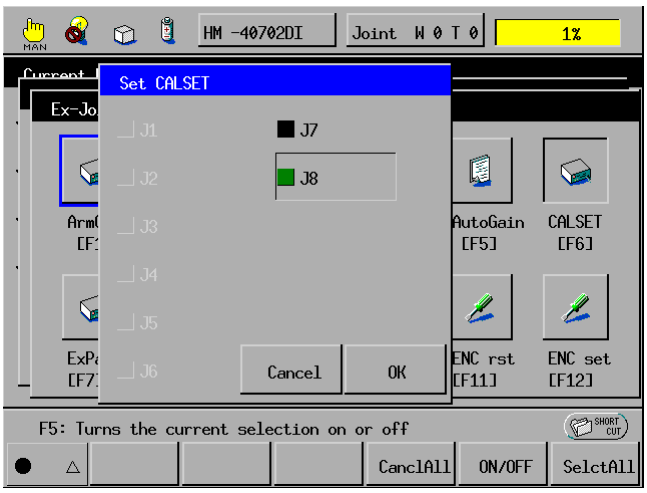
F6

(2) Calling up the Set CALSET window

In the Ex-Joint Settings window, press [F6 CALSET] to call up the Set CALSET window.

(3) CALSETing a specified extended-joint

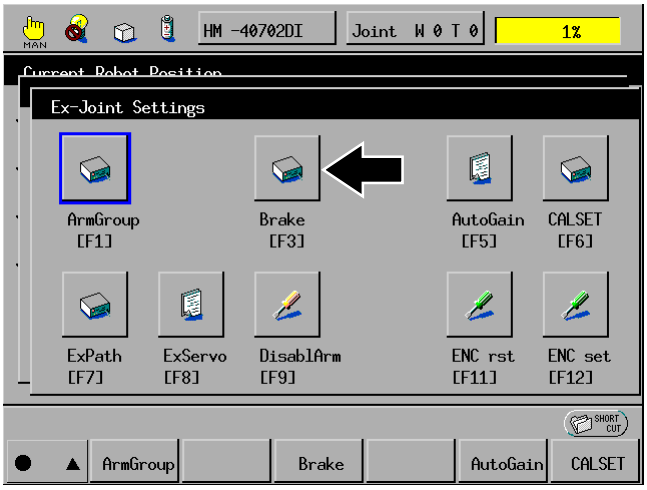
Select a joint to be CALSET and press [OK]. CALSET on the selected joint will start.



3.2.4.2 Releasing and locking an extended-joint brake

(1) Calling up the Ex-Joint Settings window

Access: Top Screen—[F2 Arm]—[F12 Maint.]—[F7 ExJoints]



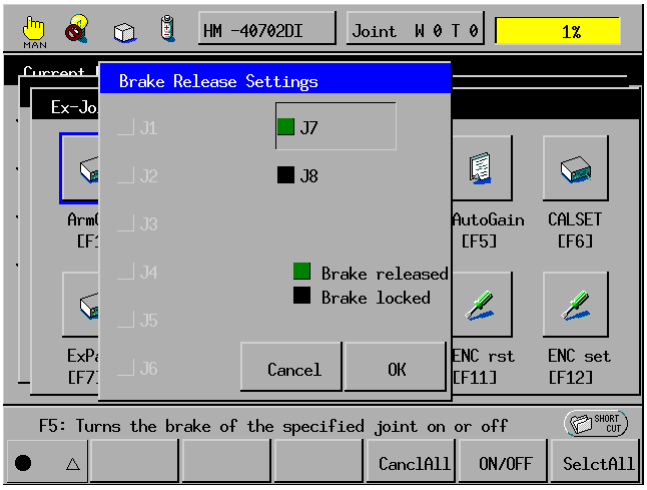
F3

(2) Calling up the Brake Release Settings window

In the Ex-Joint Settings window, press [F3 Brake] to call up the Brake Release Settings window. When a brake is released, the joint displays in green; when locked, it displays in black.

(3) Selecting a target joint to be released or locked

Select a target joint and press [F5 ON/OFF] to change the indicator color. To release the brake, turn the indicator color green; to lock it, turn it black.

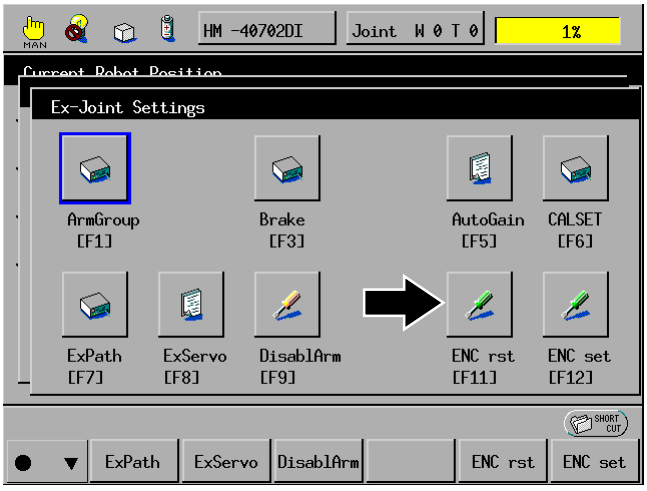


F5

3.2.4.3 Resetting an extended-joint encoder

(1) Calling up the Ex-Joint Settings window

Access: Top Screen—[F2 Arm]—[F12 Maint.]—[F7 ExJoints]

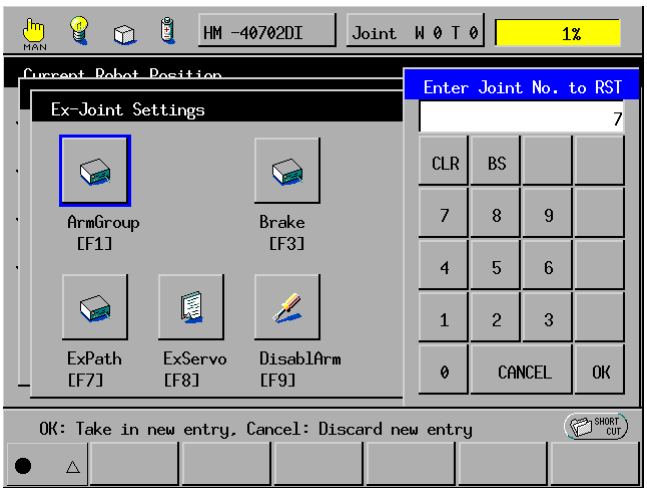


F11

(2) Entering the joint number of an encoder to be reset

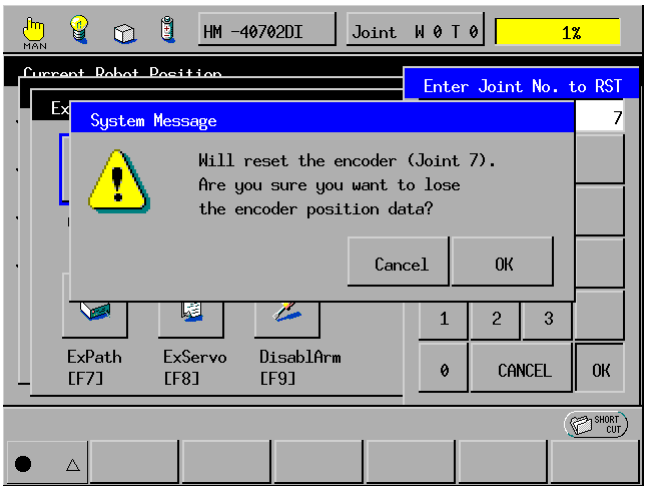
In the Ex-Joint Settings window, press [F11 ENC rst], and the numeric keypad for entering the joint number will appear as shown below.

Enter the joint number of an encoder to be reset and then press [OK]. Entering a robot joint number will cause an error.



(3) Confirming the specified joint number to be reset

The following confirmation message will appear. Check the specified joint number and press [OK].



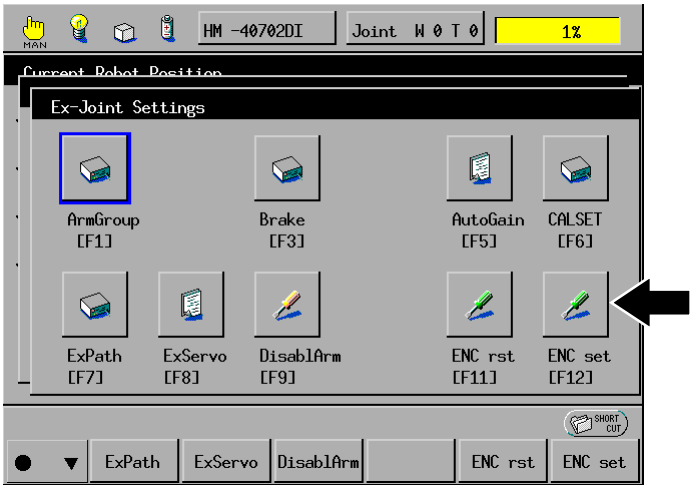
3.2.4.4 Setting a joint ID to an extended-joint encoder

NOTE: When setting a joint ID, be sure to connect only the motor whose encoder joint ID should be set, to the extended-joint control box. Disconnect the robot cable and other motor cables from the extended-joint control box. The encoder ID setting operation will set a same ID to all encoders being connected.

NOTE: Disconnecting other encoders will cause the robot controller to issue an encoder error. Press the [LOCK] button on the teach pendant to place the controller in machine lock state and then clear the error.

(1) Calling up the Ex-Joint Settings window

Access: Top Screen—[F2 Arm]—[F12 Maint.]—[F7 ExJoints]

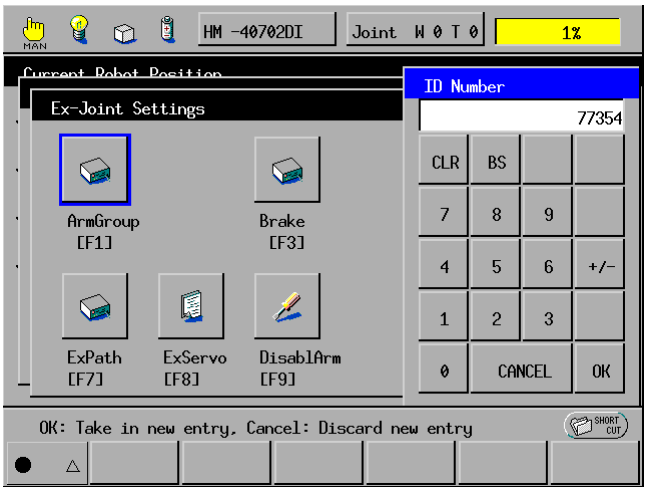


F12

(2) Entering a password

In the Ex-Joint Settings window, press [F12 ENC set], and the numeric keypad for entering a password will appear as shown below.

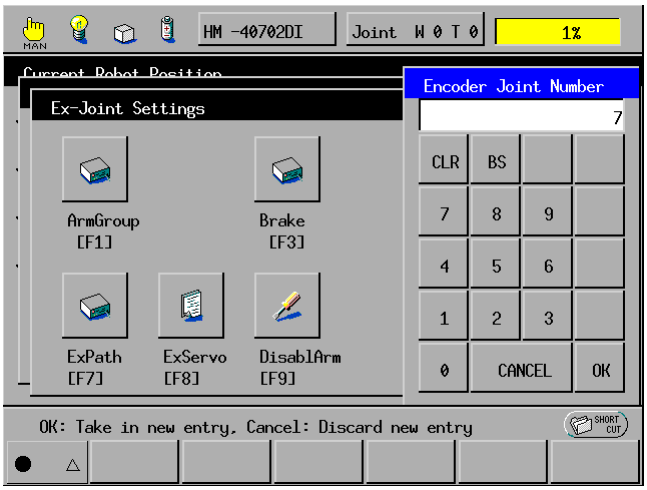
Enter the password "77354" and then press [OK].



(3) Entering an extended-joint ID number

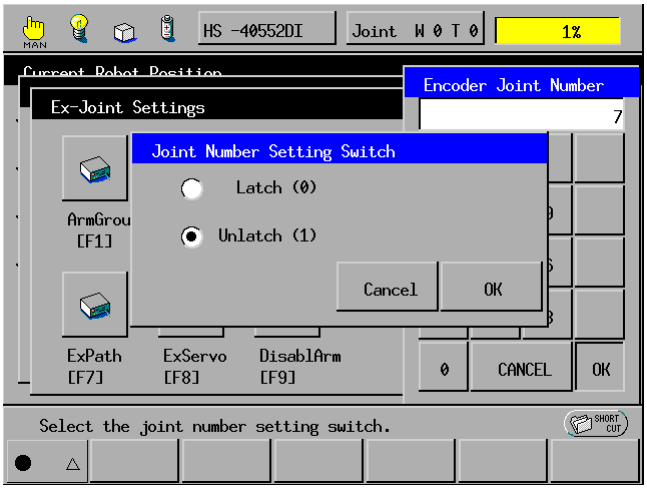
The numeric keypad for entering an extended-joint ID number will appear as shown below.

Enter an extended-joint ID number to be set and then press [OK].



(4) Latching/Unlatching the joint number setting switch

NOTE: In the window shown below, usually select "Latch" and press [OK]. Only when you want to rotate the motor 16,384 times or more in the same direction, select "Unlatch" and press [OK]. Also you need to set the boundless rotation to "1: Boundless" in the "Path Parameters for Ex-Joint" window.



(5) Confirming the specified joint ID number to be set

The confirmation message will appear. Check the specified joint ID number and press [OK].

If more than one encoder is connected, specifying a robot joint number will result in an error.

3.2.4.5 Enabling/disabling the robot arms

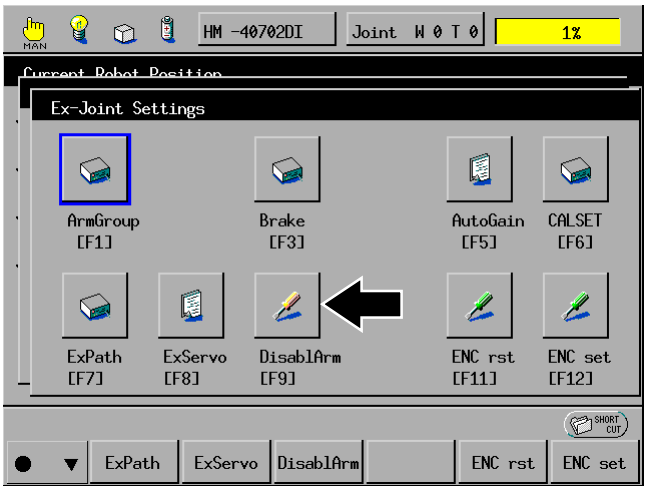
In adjustment of extended-joints, if you want to turn only extended-joint motors on for operational check without turning robot arm motors on, then disable the robot arm.

NOTE: Once you change this robot arm setting, be sure to restart the robot controller. Otherwise, an encoder-related error may occur.

NOTE: If the robot arm is disabled, turning the motor power on will not start the robot arm motors.

(1) Calling up the Ex-Joint Settings window

Access: Top Screen—[F2 Arm]—[F12 Maint.]—[F7 ExJoints]



F9

(2) Enabling/disabling the robot arm

In the Ex-Joint Settings window, press [F9 DisableArm], and the current arm setting will appear as shown below. Select the desired setting and press [OK].

3.2.5 Extended-Joint Parameter Setting Commands

NOTE: Commands described in this section are used also for general robots.

[1] Single-Joint Servo Data Monitor Commands (Library)

SetMonitorCond

Function

Sets the monitoring conditions for single-joint servo data monitor. (For Ver. 1.5 or later)

Syntax

```
SetMonitorCond(<JntNumber>,<MonitorData1>,<MonitorData2>,
<SampInterval>)
```

Description

`SetMonitorCond` sets the joint number to be monitored, monitor data (up to 2 types allowed per command), and sampling interval in ms as monitoring conditions.

The following five types of data may be monitored, two types at a time, by specifying `<MonitorData1>` and `<MonitorData2>`:

`<MonitorData1>` and `<MonitorData2>`	Data to be monitored
0	Motor speed control value in rpm
1	Current motor speed (Actual speed) in rpm
2	Motor torque control value (excluding torque offset) in ratio (%) to the rated value
3	Motor rotation angle error (Motor angle control value - Actual motor angle value) in pulses
4	Motor current absolute value (Maximum value out of three absolute values detected from all 3 phases of the motor.) in ratio (%) to the rated value

`<SampInterval>` must be set in ms as an integer between 1 and 8.

Macro definition

Not needed.

Related commands

`ClearSrvMonitor`, `StartSrvMonitor`, and `StopSrvMonitor`

Notes

(1) If this library executes following the monitor start library `StartSrvMonitor`, the error "6001: Not executable" will result. Be sure to set the monitoring conditions before starting monitor.

(2) If any of the joint number, data types, and sampling interval entered is wrong, the error message "The entered value is out of the range." will result. Correct those monitoring conditions you entered.

Example

```
CALL SetMonitorCond(7,0,3,4)  'For getting speed control value and
                              'motor angle error of J7 every 4 ms.
CALL StartSrvMonitor          'Start monitoring data.
```

StartSrvMonitor

Function

Starts monitoring single-joint servo data. (For Ver. 1.5 or later)

Syntax

```
StartSrvMonitor
```

Description

`StartSrvMonitor` fetches a maximum of 1250 samples of single-joint servo data until `StopSrvMonitor` executes.

Macro definition

Not needed.

Related commands

`ClearSrvMonitor`, `SetMonitorCond`, and `StopSrvMonitor`

Notes

(1) If the total number of data samples monitored in a monitoring cycle is 1250 or less, all data may be monitored. If it exceeds 1250 samples, the last 1250 data samples before the end of the cycle may be monitored and other data will be discarded.

(2) If any error occurs and the motor being monitored is turned OFF during monitoring, then a maximum of 850 samples before the OFF and 400 samples after that may be monitored.

(3) No data may be monitored when the target motor is off. Execute this command with the motor power on.

Example

```
CALL SetMonitorCond(7,0,3,4)  'For getting speed control value and
                              'motor angle error of J7 every 4 ms.
CALL StartSrvMonitor          'Start monitoring data.
```

StopSrvMonitor

Function

Stops monitoring single-joint servo data. (For Ver. 1.5 or later)

Syntax

```
StopSrvMonitor
```

Description

In duration from execution of `StartSrvMonitor` to that of `StopSrvMonitor`, a maximum of 1250 samples of data may be obtained.

Macro definition

Not needed.

Related commands

`ClearSrvMonitor`, `SetMonitorCond`, and `StartSrvMonitor`

Notes

(1) If the total number of data samples monitored in a monitoring cycle is 1250 or less, all data may be monitored. If it exceeds 1250, the last 1250 data samples before the end of the cycle may be monitored and other data will be discarded.

(2) If any error occurs and the motor being monitored is turned OFF during monitoring, then a maximum of 850 samples before the OFF and 400 samples after that may be monitored.

Example

```
CALL StopSrvMonitor      'End monitoring data.
```

ClearSrvMonitor

Function

Initializes the pointer of data obtained by the single-joint servo data monitor function. (For Ver. 1.5 or later)

Syntax

`ClearSrvMonitor`

Description

`ClearSrvMonitor` initializes the pointer of data already obtained and starts monitoring new data up to 1250 samples.

Macro definitions

Not needed.

Related commands

`ClearSrvMonitor`, `SetMonitorCond`, and `StartSrvMonitor`

Notes

In duration from execution of `ClearSrvMonitor` to that of `StopSrvMonitor`, if the total number of data samples monitored is 1250 or less, all data may be monitored. If it exceeds 1250, the last 1250 samples before the end of the monitoring cycle may be monitored and other data will be discarded.

Example

```
CALL StartSrvMonitor     'Start monitoring data.
     .
     .
     .
CALL ClearSrvMonitor     'Clear monitored data after
                         'execution of StartSrvMonitor.
     .
     .
     .
CALL StopSrvMonitor      'Stop monitoring data.
                         '(Data between ClearSrvMonitor and
                         'StopSrvMonitor processes are monitored.)
```

[2] Operation Termination Commands

MotionSkip

Function

Aborts running motion commands. (For Ver. 1.5 or later)

Syntax

```
MotionSkip
```

Description

`MotionSkip` aborts motion commands running in the task in which the `MotionSkip` executes.

Macro definition

Not needed.

Related commands

`GetJntData` and `GetSrvData`

Notes

(1) Execute this command in a TAKEARMed task that holds an arm semaphore. If not in a TAKEARMed task, the error "Not executable" will result.

(2) Executing `MotionSkip` in a robot motion task will abort robot joint motion commands. Executing it in an extended-joint motion task will abort extended-joint motion commands.

If `MotionSkip` executes in a motion task holding an arm group involving both robot joints and extended-joints, then both the robot and extended-joint motions will be aborted.

Example

```
defjnt lj1
defsng lf1
move p,P1,next
lj1=GetSrvState(2)       'Get errors of each joint rotation angle.
lf1=ABS(JOINT(2,lj1))    'Select rotation error of J2.
if lf1 > 10000 then
   CALL MotionSkip       'If the rotation error of J2 exceeds 10000
                         '(in pulses), then abort motion commands.
endif
```

MotionComp

Function

Judges whether execution of running motion commands is complete. (For Ver. 1.5 or later)

Syntax

`MotionComp(<MotionCommandComplete>)`

Description

If `MotionComp` judges that execution of running motion commands is complete, then it returns "1" in `<MotionCommandComplete>`.

This command checks motion commands running in the task in which the `MotionComp` executes. It is not applicable to motion commands in any other tasks.

If a motion command has an encoder value check option, then `MotionComp` will interpret the moment when the encoder count is converged within the positioning error allowance as completion of the motion command. For other operations, if motion control to the servo loop disappears, then `MotionComp` will judge that the command is complete.

Macro definition

Not needed.

Related commands

`GetJntData`, `GetSrvData`, and `MotionSkip`

Notes

(1) Execute this command in a TAKEARMed task that holds an arm semaphore. If not in a TAKEARMed task, the error "Not executable" will result.

(2) Executing `MotionComp` in a robot motion task will judge whether robot motion commands are complete. Executing it in an extended-joint motion task will judge whether extended-joint motion commands are complete.

If `MotionComp` executes in a motion task holding an arm group involving both robot joints and extended-joints, then completion of both the robot and extended-joint motions will be judged.

(3) When the motion is on Halt, `MotionComp` will interpret it as operation being in progress.

(4) If you use a local variable for `<MotionCommandComplete>`, the local variable must be reset to "0" beforehand.

Example

```
defint comp=0             'Initialize motion command completion status.
defjnt lj1
defsng lf1
move p,P1,next
DO
  lj1=GetSrvState(2)      'Get error of each joint rotation.
  lf1=ABS(JOINT(2,lj1))   'Select the rotation error of J2.
  if lf1 > 10000 then
    CALL MotionSkip       If the rotation error of J2 exceeds 10000 (in pulses),
                          'then abort motion commands and end the loop.
    EXIT DO
  endif
  CALL MotionComp(comp)
LOOP UNTIL comp=1         'Loop until the end of motion commands.
```

[3] Internal Servo Data Get Commands

GetSrvData

Function

Gets the internal servo data of robot joints. (For Ver. 1.5 or later)

Syntax

```
<InternalServoData> = GetSrvData(<DataNumber>)
```

Description

`GetSrvData` gets the internal servo data specified by `<DataNumber>` into `<InternalServoData>`.

`<InternalServoData>` is a joint type data of robot. `<DataNumber` should be any of the following:

<DataNumber>	<InternalServoData>
1	Current motor speed (Actual speed) in rpm
2	Motor rotation angle error in pulses
4	Motor current absolute value in ratio (%) to the rated value
5	Motor torque control value (excluding torque offset) in ratio (%) to the rated value
8	Joint position or angle control value in mm or degrees
17	Tool-end speed (3 position elements only in the work coordinates) in mm/s
18	Tool-end positioning speed (3 position elements only in the work coordinates) in mm
19	Tool-end speed (3 position elements only in the tool coordinates) in mm/s
20	Tool-end positioning speed (3 position elements only in the tool coordinates) in mm

Macro definition

Not needed.

Related commands

`GetJntData`

Notes

(1) Data numbers other than those given above are reserved. Do not use any other number other than the above, although no error will result if you specify any number up to 30.

(2) If you attempt to fetch the servo data when the single-joint servo data monitor is running, the fetching process may become very slow. Take care when using the single-joint servo data monitor.

(3) If you change `<DataNumber>`, the modification may take time. Do not change it so frequently.

(4) Execute this command in a TAKEARMed task that holds an arm semaphore. If not in a TAKEARMed task, the error "21F7: Cannot take arm semaphore" will result.

Example

```
defjnt vel
defsng absv,xvel,yvel,zvel
vel=GetSrvData(17)                          'Get tool-end speed.
xvel=JOINT(1,vel)                           'Select X component in work coordinates.
yvel=JOINT(2,vel)                           'Select Y component in work coordinates.
zvel=JOINT(3,vel)                           'Select Z component in work coordinates.
absv = SQR(xvel*xvel+yvel*yvel+zvel*zvel) 'Calculate total tool-end speed.
```

GetJntData

Function

Gets the internal servo data of a specified joint. (For Ver. 1.5 or later)

Syntax

```
<JntInternalServoData> = GetJntData(<DataNumber>,<JntNumber>)
```

Description

GetJntData gets the internal servo data (specified by <DataNumber>) of a joint specified by <JntNumber> into <JntInternalServoData>.

<JntInternalServoData>.is a floating point type data of the specified joint. <DataNumber> should be any of the following:

<DataNumber>	<JntInternalServoData>
1	Current motor speed (Actual speed) in rpm
2	Motor rotation angle error in pulses
4	Motor current absolute value in ratio (%) to the rated value
5	Motor torque control value (excluding torque offset) in ratio (%) to the rated value
8	Joint position or angle control value in mm or degrees

Macro definition

Not needed

Related commands

GetSrvData

Notes

(1) Data numbers other than those given above are reserved. Do not use any other number other than the above, although no error will result if you specify any number up to 30.

(2) If you attempt to fetch the servo data when the single-joint servo data monitor is running, the fetching process may become very slow. Take care when using the single-joint servo data monitor.

(3) If you change <DataNumber>, the modification may take time. Do not change it so frequently.

(4) Execute this command in a TAKEARMed task that holds an arm semaphore. If not in a TAKEARMed task, the error "21F7: Cannot take arm semaphore" will result.

Example

```
defsng vel
vel=GetJntData(1,7)      'Get motor speed of J7.
```

3.3 Customizing TP Operation Screens

Main software version 1.5 or later allows you to easily customize your own operation screens on the teach pendant for facilitating control of the robot by the robot controller in stand-alone mode.

In PAC language, you may program your own control buttons in size, position, and color and paste them onto the Teach Pendant screen.

Once the PAC program in which you have defined your own screens runs, those screens go into effect and remain in effect as long as you do not clear them, even if you restart the robot system or controller.

■ Buttons and screens

You may customize buttons and screens up to 100 and 10, respectively.

■ Commands for creating TP operation screens

`set_button`	Sets button parameters (incl. an attribute for choosing visible/invisible)
`set_page`	Sets page parameters (incl. an attribute for choosing visible/invisible)
`change_bCap`	Edits captions on buttons
`change_pCap`	Edits captions on pages
`disp_page`	Displays a specified page

■ Parameters set by commands

Button parameters	• Index • Display position (Upper left X coordinate, upper left Y coordinate, lower right X coordinate, and lower right Y coordinate) • Button type (Touch switch with variation in shape, value display box, value entry box, digital switch, lamp with variation in shape, and page switching button) • Status (reserved) • Background color • Text color • Enable/disable flag • Visible/invisible flag • Captions • Variable type • Variable number • I/O number • Page number • Modification flag (reserved) • Result (reserved)
Page parameters	• Index • Screen type (fixed) • Status (reserved) • Background color • Text color • Enable/disable flag • Visible/invisible flag • Captions • Modification flag (reserved) • Result (reserved)

3.3.1 Programming a TP operation screen

Program a TP operation screen as follows:

(1) Setting button parameters

Use `set_button` command to specify button parameters for a button.

(Example) Create a button numbered 1 with background (7) set to black (0)

```
set_button 1,7,0
```

(2) Setting page parameters

Use `set_page` command to specify page parameters for a page.

(Example) Create a page numbered 2 with background (3) set to red (4)

```
set_page 2,3,4
```

(3) Setting a button caption

Use `change_bCap` command to specify a desired button caption.

(Example) Specify caption "Setup" for button numbered 2

```
change_bCap 2,"Setup"
```

(4) Setting a page caption

Use `change_pCap` command to specify a desired page caption.

(Example) Specify caption "Screen 3" for page numbered 3

```
change_pCap 3,"Screen 3"
```

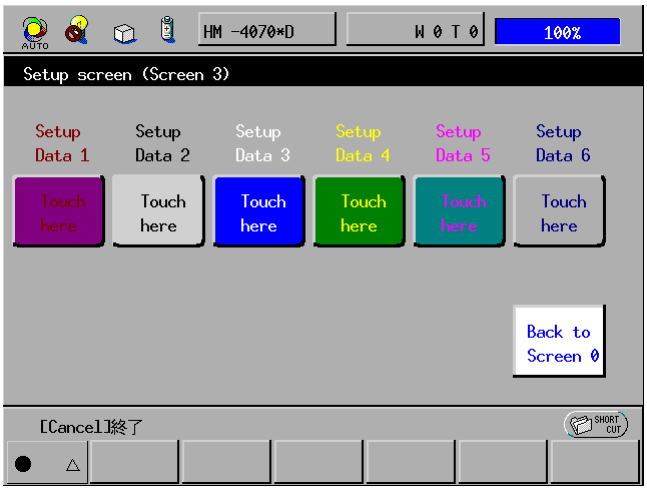
(5) Displaying a specified page

Use `disp_page` command to display the desired page.

(6) Displaying a programmed TP operation screen

From the top screen of the teach pendant, choose [F9: OpePanel] to display a TP operation screen you have programmed.

TP operation screen sample



3.3.2 TP Operation Screen Customizing Commands

set_button

Function

Sets button parameters.

Syntax

```
set_button <ButtonNumber>,<ParameterType>,<NewValue>
```

`<ButtonNumber>` Number indicating the button location in all button arrangement on a TP operation panel.

`<ParameterType>` Button attributes including color, position and others. (See the table below.)

`<NewValue>` Parameter value for making new settings (See the table below.)

<ParameterType>	Description	<NewValue>
1	Upper left X coordinate	0 to 640 in dots
2	Upper left Y coordinate	0 to 350 in dots
3	Lower right X coordinate	0 to 640 in dots
4	Lower right Y coordinate	0 to 350 in dots
5	Button type	0: None 1: Label 2: Line 17: 2D button (Change variable) 18: 3D button (Change variable) 19: 3D button (Change variable) 20: Circle (Change variable) 33: 2D LED (lamp) 34: Circle LED (lamp) 35: 3D button (Change IO)
6	Button status	Reserved.
7	Background color	0: Black 1: Blue 2: Green 3: Cyan 4: Red 5: Magenta 6: Brown 7: Light gray 8: Gray 9: Light blue 10: Light green 11: Light cyan 12: Light red 13: Light magenta 14: Yellow 15: White

<ParameterType>	Explanation	<NewValue>
8	Text color	0: Black 1: Blue 2: Green 3: Cyan 4: Red 5: Magenta 6: Brown 7: Light gray 8: Gray 9: Light blue 10: Light green 11: Light cyan 12: Light red 13: Light magenta 14: Yellow 15: White
9	Usable state	0: Disable 1: Enable
10	Visible/invisible state	0: Invisible 1: Visible
11	Variable type (Fixed)	1: Integer
12	Variable number	Variable number that may be changed by the change variable button.
13	I/O number	Variable number that may be changed by the change I/O button. (128 to 511)
14	Display page number	Page number in which the buttons are displayed.

Explanation

set_button changes the current value of a parameter specified by <ParameterType> to <NewValue> to modify the specifications of a button specified by <ButtonNumber>.

*Sample of Button type

1: Label

2: Line

3: 2D button (Change variable)

4: 3D button (Change variable)

5: 3D button (Change variable)

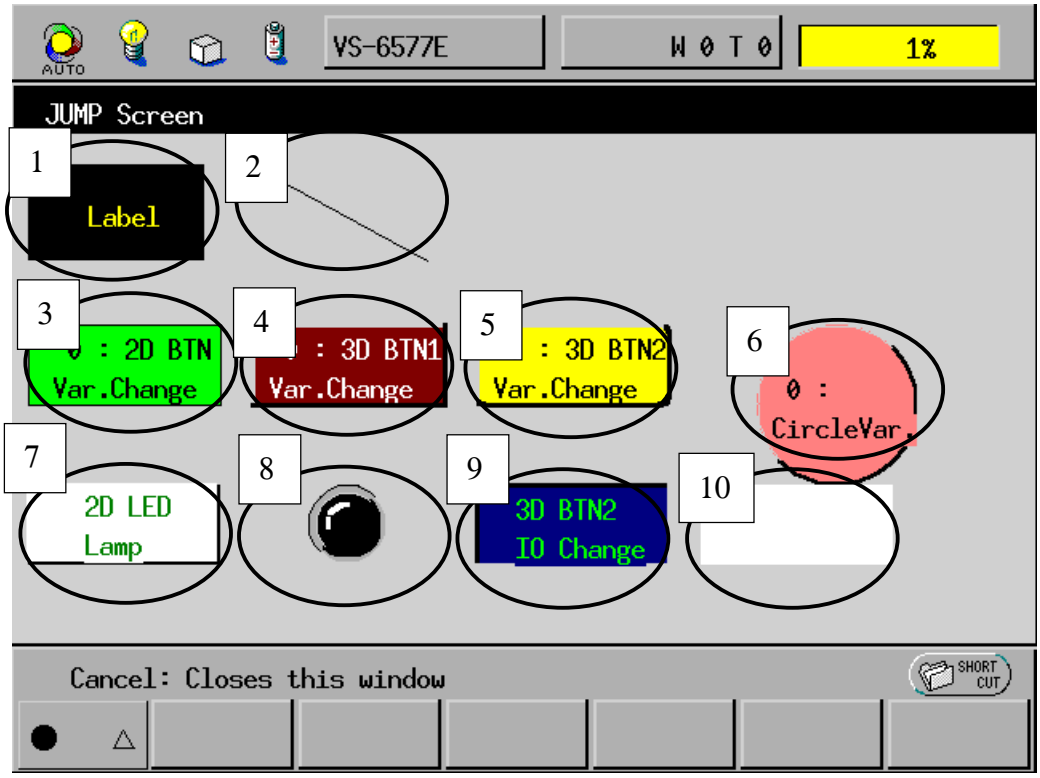
6: Circle

7: 2D LED

8: Circle LED

9: 3D button (Change IO)

10: Box



Example

```
'!TITLE "<Title>"
PROGRAM sample1
          .
          .
          .
        set_button (btn_no),(1),(minx)
        set_button (btn_no),(2),(miny)
        set_button (btn_no),(3),(maxx)
        set_button (btn_no),(4),(maxy)
          .
          .
          .
end
```

set_page

Function

Sets page parameters.

Syntax

```
set_page <PageNumber>,<ParameterType>,<NewValue>
```

`<PageNumber>` Number indicating a page out of all pages arranged on a TP operation panel.

`<ParameterType>` Page attributes including color, position and others. (See the table below.)

`<NewValue>` Parameter value for making new settings (See the table below.)

<ParameterType>	Description	<NewValue>
1	Page type	0 (Fixed)
2	Button status	Not used.
3	Background color	0: Black 1: Blue 2: Green 3: Cyan 4: Red 5: Magenta 6: Brown 7: Light gray 8: Gray 9: Light blue 10: Light green 11: Light cyan 12: Light red 13: Light magenta 14: Yellow 15: White
4	Text color (Not used in Ver. 1.5 or 1.6)	0: Black 1: Blue 2: Green 3: Cyan 4: Red 5: Magenta 6: Brown 7: Light gray 8: Gray 9: Light blue 10: Light green 11: Light cyan 12: Light red 13: Light magenta 14: Yellow 15: White
5	Usable state	0: Disable 2: Enable
6	Visible/invisible state	0: Invisible 1: Visible

Description

set_page changes the current value of a parameter specified by <ParameterType> to <NewValue> to modify the specifications of a page specified by <PageNumber>.

Example

```
'!TITLE "<Title>"
PROGRAM sample2
         .
         .
         .
        set_page panel_no, P_BGCOLOR,GRAY    'Set background color of the
                                             'page.
        set_page panel_no, P_USESTATE,ON     'Enable page.
        set_page panel_no, P_VISSTATE,ON     'Make page visible.
         .
         .
         .
end
```

change_bCap

Function

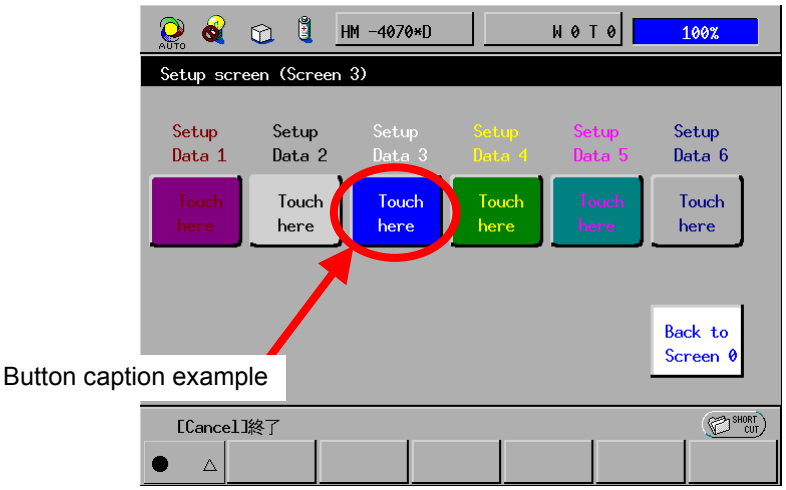
Edits a caption for a specified button.

Syntax

```
change_bCap <ButtonNumber>,<Caption>
```

`<ButtonNumber>` Number indicating the button location in all button arrangement on a TP operation panel.

`<Caption>` Character string to be displayed on the center of a button.



Description

`change_bCap` displays a character string specified by `<Caption>` on the center of a button specified by `<ButtonNumber>`.

Example

```
'!TITLE "<Title>"
PROGRAM sample3
          .
          .
        bcap4 = "Cut workpiece"
        btn_no = 3
          .
          .
          .
        change_bCap btn_no,bcap4
          .
          .
          .
end
```

change_pCap

Function

Edits a caption for a specified page.

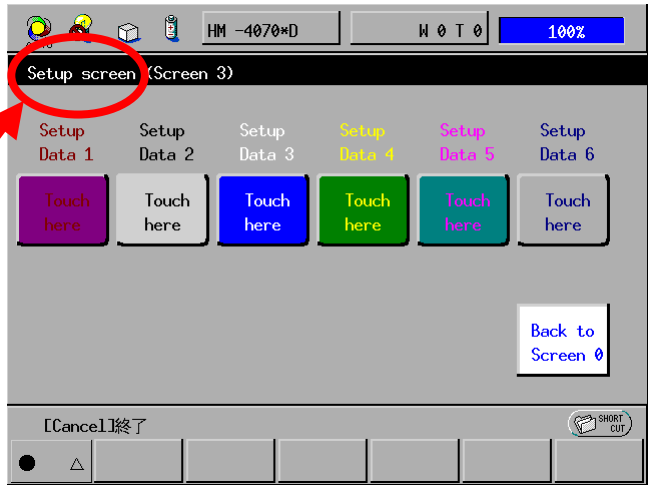
Syntax

```
change_pCap <PageNumber>,<Caption>
```

<PageNumber> Number indicating a page out of all pages arranged on a TP operation panel.

<Caption> Character string to be displayed in the title bar of a page.

Page caption example



Description

change_pCap displays a character string specified by <Caption> in the title bar of a page specified by <PageNumber>.

Example

```
'!TITLE "<Title>"
PROGRAM sample4
          .
          .
          .
        pcap4 = "Cut workpiece"
        page_no = 3
          .
          .
          .
        change_pCap page_no,pcap4
          .
          .
          .
end
```

disp_page

Function

Displays a specified page of a TP operation screen.

Syntax

```
disp_page <PageNumber>
```

<PageNumber>	Number indicating a page out of all pages arranged on a TP operation panel.

Description

disp_page displays the page specified by <PageNumber> on the TP operation screen.

Example

```
'!TITLE "<Title>"
PROGRAM sample3
          .
          .
          .
        'disp_page panel_no    'Display the specified page.
          .
          .
          .
end
```

3.3.3 Sample Program: Creating a TP Operation Panel

Shown below is a sample program for creating a TP operation panel.

```
'! TITLE "<Title>"
PROGRAM BUTTON3D_VAL_3
'Color definition
#define BLACK                0                    'Black
#define BLUE                 1                    'Blue
#define GREEN                2                    'Green
#define CYAN                 3                    'Cyan
#define RED                  4                    'Red
#define MAGENTA              5                    'Magenta
#define BROWN                6                    'Brown
#define LIGHTGRAY            7                    'Light gray
#define GRAY                 8                    'Gray
#define LIGHTBLUE            9                    'Light blue
#define LIGHTGREEN           10                   'Light green
#define LIGHTCYAN            11                   'Light cyan
#define LIGHTRED             12                   'Light red
#define LIGHTMAGENTA         13                   'Light magenta
#define YELLOW               14                   'Yellow
#define WHITE                15                   'White

'Button definition
#define LABEL                1                    'Label
#define LINE                 2                    'Line
#define 2DBUTTON_V           17                   '2D button (Change variable)
#define 3DBUTTON_V           18                   '3D button (Change variable)
#define 3DBUTTON_V2          19                   '3D button 2 (Change variable)
#define CIRCLE               20                   'Circle (Change variable)
#define 2DLED                33                   '2DLED (Lamp)
#define CIRCLELED            34                   'Circle LED (Lamp)
#define 3DBUTTON_IO          35                   '3D button (Change IO)

'Page parameters
#define P_BGCOLOR            3                    'Page background color
#define P_CHARCOLOR          4                    'Page text color
#define P_USESTATE           5                    'Enable/disable
#define P_VISSTATE           6                    'Visible/invisible
```

```
'Button parameters
#define X_UPPERLEFT_P         1                       'Upper left X coordinate
#define Y_UPPERLEFT_P         2                       'Upper left Y coordinate
#define X_LOWERRIGHT_P        3                       'Lower right X coordinate
#define Y_LOWERRIGHT_P        4                       'Lower right Y coordinate
#define B_KIND                5                       'Button type
#define B_BGCOLOR             7                       'Button background color
#define B_FGCOLOR             8                       'Button text color
#define B_USESTATUS           9                       'Button enable/disable
#define B_VISSTATUS           10                      'Button visible/invisible
#define B_VALUEKIND           11                      'Button variable type (Fixed)
#define B_VALUE_NO            12                      'Button variable number
#define B_IO_NO               13                      'Button I/O number
#define B_DISP_PNO            14                      'Button display page number

'Button status
#define ON                    1                       'ON
#define OFF                   0                       'OFF
#define I_VAL                 1                       'Integer

'Button address
#define IO_PB_ADRS            170                     'I/O number assigned to the 1st button
                                                      'on a TP operation panel.

defint btn_no,minx,maxx,miny,maxy,loopcnt
defint enable,visible,var_type,var_index
defint panel_no,io_no,io_adrs,btn_adrs
defstr panel_cap
defstr bcap0,bcap1,bcap2,bcap3,bcap4,bcap5,bcap6,bcap7
      panel_no = 3

      panel_cap = "Setup screen (Screen 3)"
      loopcnt = 0

      change_pCap panel_no,panel_cap                  'Set page title.
      set_page panel_no,P_BGCOLOR,GRAY                'Set page background color.
      set_page panel_no,P_USESTATE,ON                 'Enable page.
      set_page panel_no,P_VISSTATE,ON                 'Make page visible.
```

```
'Resetting all parameters
      btn_adrs = 30
      io_no = IO_PB_ADRS
      reset io[128 to 133]
      enable = ON
      visible = ON
      var_type = I_VAL
      var_index = 1

      bcap0 = "plan”+chr$(10)+"Data1"
      bcap1 = "plan”+chr$(10)+"Data2"
      bcap2 = "plan”+chr$(10)+"Data3"
      bcap3 = "plan”+chr$(10)+"Data4"
      bcap4 = "plan”+chr$(10)+"Data5"
      bcap5 = "plan”+chr$(10)+"Data6"
      bcap6 = "Screen0”+chr$(10)+"Back to"
      bcap7 = "here"+chr$(10)+"Touch"

      while loopcnt < 6                                    'Loop 6 times.
            btn_no = btn_adrs + loopcnt
            minx = 10 + ((loopcnt mod 6)*100)
            miny = 50 + ((loopcnt / 6)*100)
            maxx = 100 + ((loopcnt mod 6)*100)
            maxy = 120 + ((loopcnt / 6)*100)
            io_adrs = IO_PB_ADRS + loopcnt

      'Common process
            set_button (btn_no),(1),(minx)
            set_button (btn_no),(2),(miny)
            set_button (btn_no),(3),(maxx)
            set_button (btn_no),(4),(maxy)
            set_button (btn_no),(9),(enable)
            set_button (btn_no),(10),(visible)
            set_button (btn_no),(11),(var_type)
            set_button (btn_no),(12),(var_index)
            set_button (btn_no),(14),(panel_no)
```

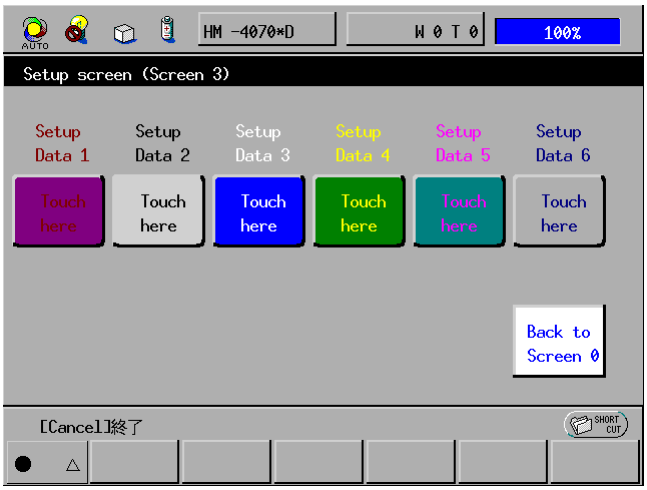
```
'Label display
      select case loopcnt
      case 0
            set_button btn_no,B_KIND,LABEL              'Set button type.
            set_button btn_no,B_IO_NO,io_adrs           'Set I/O address.
            set_button btn_no,B_BGCOLOR,GRAY            'Set background color.
            set_button btn_no,B_FGCOLOR,RED             'Set foreground color.
            change_bCap btn_no,bcap0                    'Set button number.

      case 1
            set_button btn_no,B_KIND,LABEL              'Set button type.
            set_button btn_no,B_IO_NO,io_adrs           'Set I/O address.
            set_button btn_no,B_BGCOLOR,GRAY            'Set background color.
            set_button btn_no,B_FGCOLOR,BLACK           'Set foreground color.
            change_bCap btn_no,bcap1                    'Set button number.

      case 2
            set_button btn_no,B_KIND,LABEL              'Set button type.
            set_button btn_no,B_IO_NO,io_adrs           'Set I/O address.
            set_button btn_no,B_BGCOLOR,GRAY            'Set background color.
            set_button btn_no,B_FGCOLOR,WHITE           'Set foreground color.
            change_bCap btn_no,bcap2                    'Set button number.

      case 3
            set_button btn_no,B_KIND,LABEL              'Set button type.
            set_button btn_no,B_IO_NO,io_adrs           'Set I/O address.
            set_button btn_no,B_BGCOLOR,GRAY            'Set background color.
            set_button btn_no,B_FGCOLOR,YELLOW          'Set foreground color.
            change_bCap btn_no,bcap3                    'Set button number.

      case 4
            set_button btn_no,B_KIND,LABEL              'Set button type.
            set_button btn_no,B_IO_NO,io_adrs           'Set I/O address.
            set_button btn_no,B_BGCOLOR,GRAY            'Set background color.
            set_button btn_no,B_FGCOLOR,LIGHTMAGENTA    'Set foreground. color
            change_bCap btn_no,bcap4                    'Set button number.
```

```
            case 5
                  set_button btn_no,B_KIND,LABEL              'Set button type.
                  set_button btn_no,B_IO_NO,io_adrs           'Set I/O address.
                  set_button btn_no,B_BGCOLOR,GRAY            'Set background color.
                  set_button btn_no,B_FGCOLOR,LIGHTBLUE       'Set foreground color.
                  change_bCap btn_no,bcap5                    'Set button number.
            end select
            loopcnt = loopcnt + 1
      wend

'Display in the lower row
      loopcnt = 0
      while loopcnt < 6                                       'Loop 6 times.
            btn_no = btn_adrs + 10 + loopcnt
            minx = 10 + ((loopcnt mod 6)*100)
            miny = 120 + ((loopcnt / 6)*100)
            maxx = 100 + ((loopcnt mod 6)*100)
            maxy = 190 + ((loopcnt / 6)*100)
            io_adrs = IO_PB_ADRS+6+loopcnt
            var_index = var_index + 1

      'Common process
            set_button (btn_no),(1),(minx)
            set_button (btn_no),(2),(miny)
            set_button (btn_no),(3),(maxx)
            set_button (btn_no),(4),(maxy)
            set_button (btn_no),(9),(enable)
            set_button (btn_no),(10),(visible)
            set_button (btn_no),(11),(var_type)
            set_button (btn_no),(12),(var_index)
            set_button (btn_no),(14),(panel_no)
```

```
      'Label display
            select case loopcnt
            case 0
'                 set_button btn_no,B_KIND,3DBUTTON_V          'Set button type.
                  set_button btn_no,B_KIND,3DBUTTON_V2         'Set button type.
                  set_button btn_no,B_IO_NO,io_adrs            'Set I/O address.
                  set_button btn_no,B_BGCOLOR,MAGENTA          'Set background color.
                  set_button btn_no,B_FGCOLOR,RED              'Set foreground color.
                  var_index = loopcnt
                  set_button btn_no,B_VALUE_NO,var_index
                  change_bCap btn_no,bcap7
            case 1
'                 set_button btn_no,B_KIND,3DBUTTON_V          'Set button type.
                  set_button btn_no,B_KIND,3DBUTTON_V2         'Set button type.
                  set_button btn_no,B_IO_NO,io_adrs            'Set I/O address.
                  set_button btn_no,B_BGCOLOR,LIGHTGRAY        'Set background color.
                  set_button btn_no,B_FGCOLOR,BLACK            'Set foreground color.
                  var_index = loopcnt
                  set_button btn_no,B_VALUE_NO,var_index
                  change_bCap btn_no,bcap7
            case 2
'                 set_button btn_no,B_KIND,3DBUTTON_V          'Set button type.
                  set_button btn_no,B_KIND,3DBUTTON_V2         'Set button type
                  set_button btn_no,B_IO_NO,io_adrs            'Set I/O address.
                  set_button btn_no,B_BGCOLOR,BLUE             'Set background color.
                  set_button btn_no,B_FGCOLOR,WHITE            'Set foreground color.
                  var_index = loopcnt
                  set_button btn_no,B_VALUE_NO,var_index
                  change_bCap btn_no,bcap7
            case 3
'                 set_button btn_no,B_KIND,3DBUTTON_V          'Set button type.
                  set_button btn_no,B_KIND,3DBUTTON_V2         'Set button type.
                  set_button btn_no,B_IO_NO,io_adrs            'Set I/O address.
                  set_button btn_no,B_BGCOLOR,GREEN            'Set background color.
                  set_button btn_no,B_FGCOLOR,YELLOW           'Set foreground color.
                  var_index = loopcnt
                  set_button btn_no,B_VALUE_NO,var_index
                  change_bCap btn_no,bcap7
            case 4
'                 set_button btn_no,B_KIND,3DBUTTON_V          'Set button type.
                  set_button btn_no,B_KIND,3DBUTTON_V2         'Set button type.
                  set_button btn_no,B_IO_NO,io_adrs            'Set I/O address.
                  set_button btn_no,B_BGCOLOR,CYAN             'Set background color.
                  set_button btn_no,B_FGCOLOR,LIGHTMAGENTA     'Set foreground color
                  var_index = loopcnt
                  set_button btn_no,B_VALUE_NO,var_index
                  change_bCap btn_no,bcap7
```

```
            case 5
'                 set_button btn_no,B_KIND,3DBUTTON_V          'Set button type.
                  set_button btn_no,B_KIND,3DBUTTON_V2         'Set button type.
                  set_button btn_no,B_IO_NO,io_adrs            'Set I/O address.
                  set_button btn_no,B_BGCOLOR,GRAY             'Set background color.
                  set_button btn_no,B_FGCOLOR,LIGHTBLUE        'Set foreground color.
                  var_index = loopcnt
                  set_button btn_no,B_VALUE_NO,var_index
                  change_bCap btn_no,bcap7

            end select
            loopcnt = loopcnt + 1
      wend

'Creating a 3D button for returning to screen 0
            io_adrs = 128
            minx = 510
            maxx = 600
            miny = 250
            maxy = 320
            btn_no = 92
            set_button (btn_no),(1),(minx)
            set_button (btn_no),(2),(miny)
            set_button (btn_no),(3),(maxx)
            set_button (btn_no),(4),(maxy)
            set_button (btn_no),(9),(enable)
            set_button (btn_no),(10),(visible)
            set_button (btn_no),(11),(var_type)
            set_button (btn_no),(12),(var_index)
            set_button (btn_no),(14),(panel_no)
            set_button btn_no,B_KIND,3DBUTTON_IO             'Set button type.
            set_button btn_no,B_IO_NO,io_adrs                'Set I/O address.
            set_button btn_no,B_BGCOLOR,BLUE                 'Set background color.
            set_button btn_no,B_FGCOLOR,WHITE                'Set foreground color.
            change_bCap btn_no,bcap6                         'Set button number.

            disp_page panel_no                               'Display specified screen.

END
```

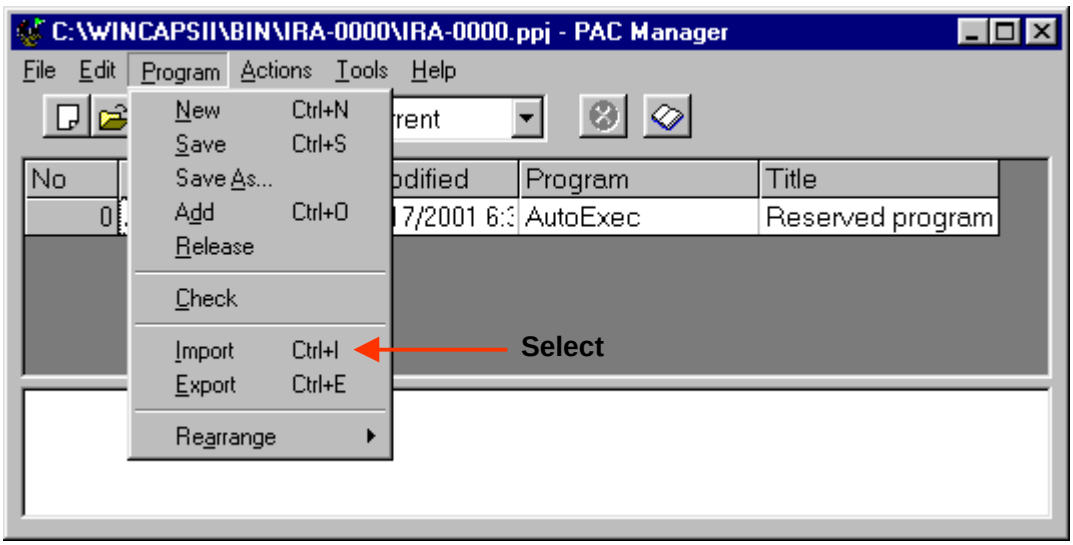
TP Operation Panel Sample: Result of the above sample program

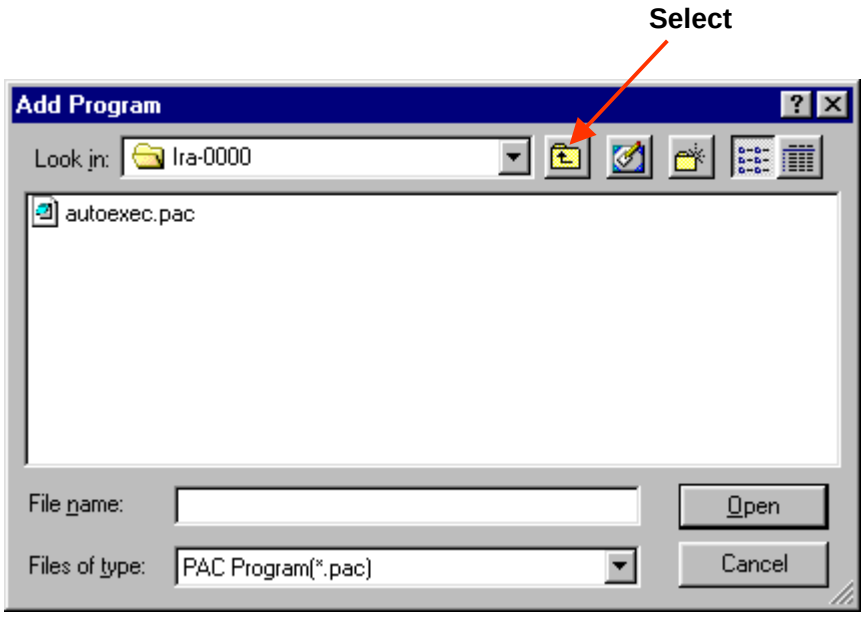


22.7 Creating TP Easy Operation Panel Screen

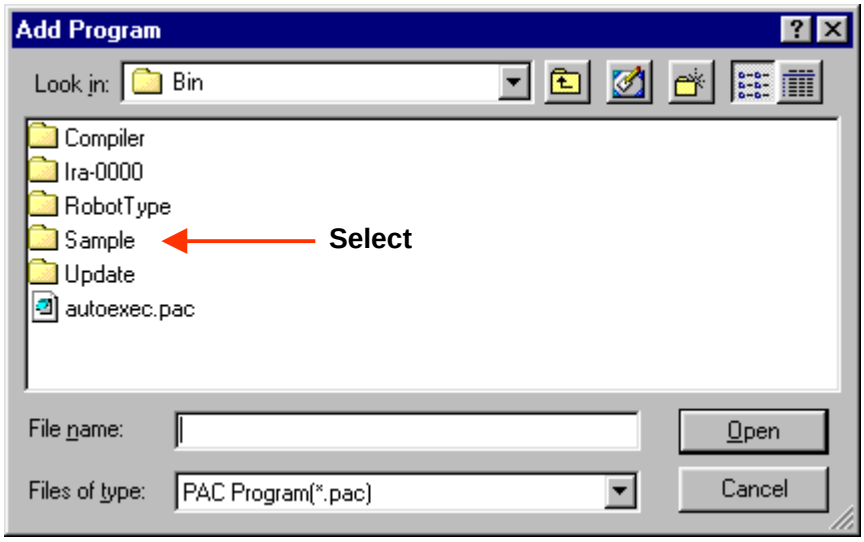
22.7.1 Sample Programs for Creating Operation Panel Screen

To create the operation panel screen, a sample program is registered in WINCAPS II under \Wincaps2\Sample\make_screen. To import the sample program to the PAC manager, select [Program] → [Import] → [Add Program] to move to the folder above, and aslo select a required sample program.

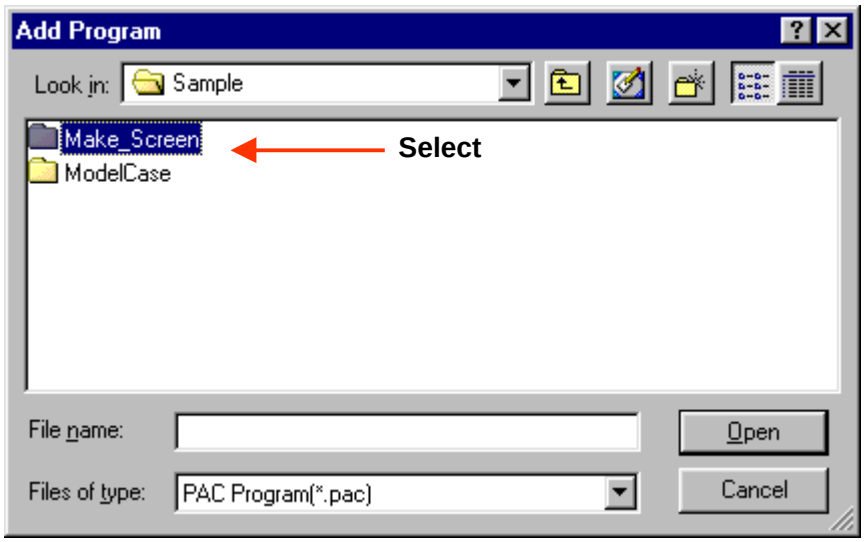




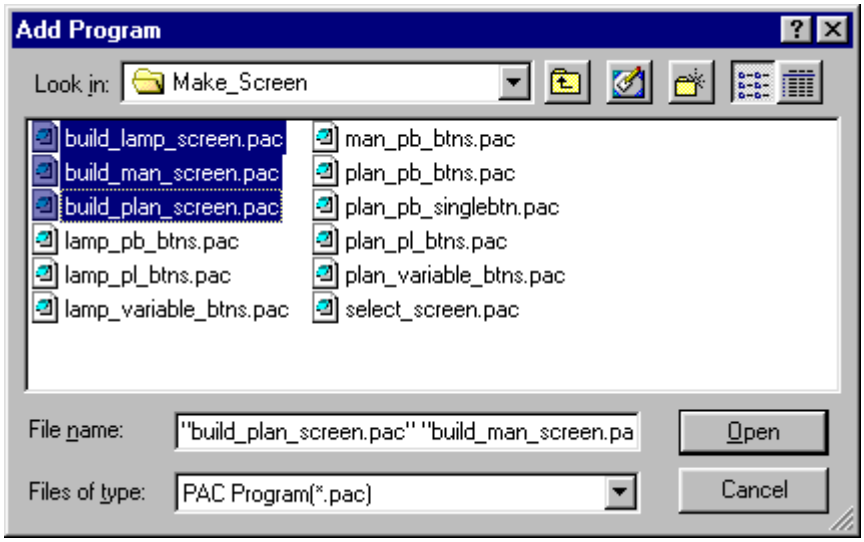
Select a sample folder.



Select the make_screen folder.



Select a prorgam to be imported.



Select_Screen (Sample program) (Ver.1.7 or later)

Function

A use example of general-purpose operation screen switching processing

Explanation

This program controls screen switching in the general-purpose operation function.

This program monitors the I/O numbers of the screen switching buttons defined on each operation panel screen. When I/O is turned on, the corresponding operation panel screen is displayed.

This program is a sample; so, change it according to the actual processing.

Macro definitions

<button.h> is necessary.

Related commands

`Build_Lamp_Screen, Build_Man_Screen, and Build_Plan_Screen`

Build_Lamp_Screen (Sample program) (Ver.1.7 or later)

Function

Creates the lamp screen below. (A screen number is required as an argument.)

Explanation

This program calls various button creation programs and creates the lamp screen below.

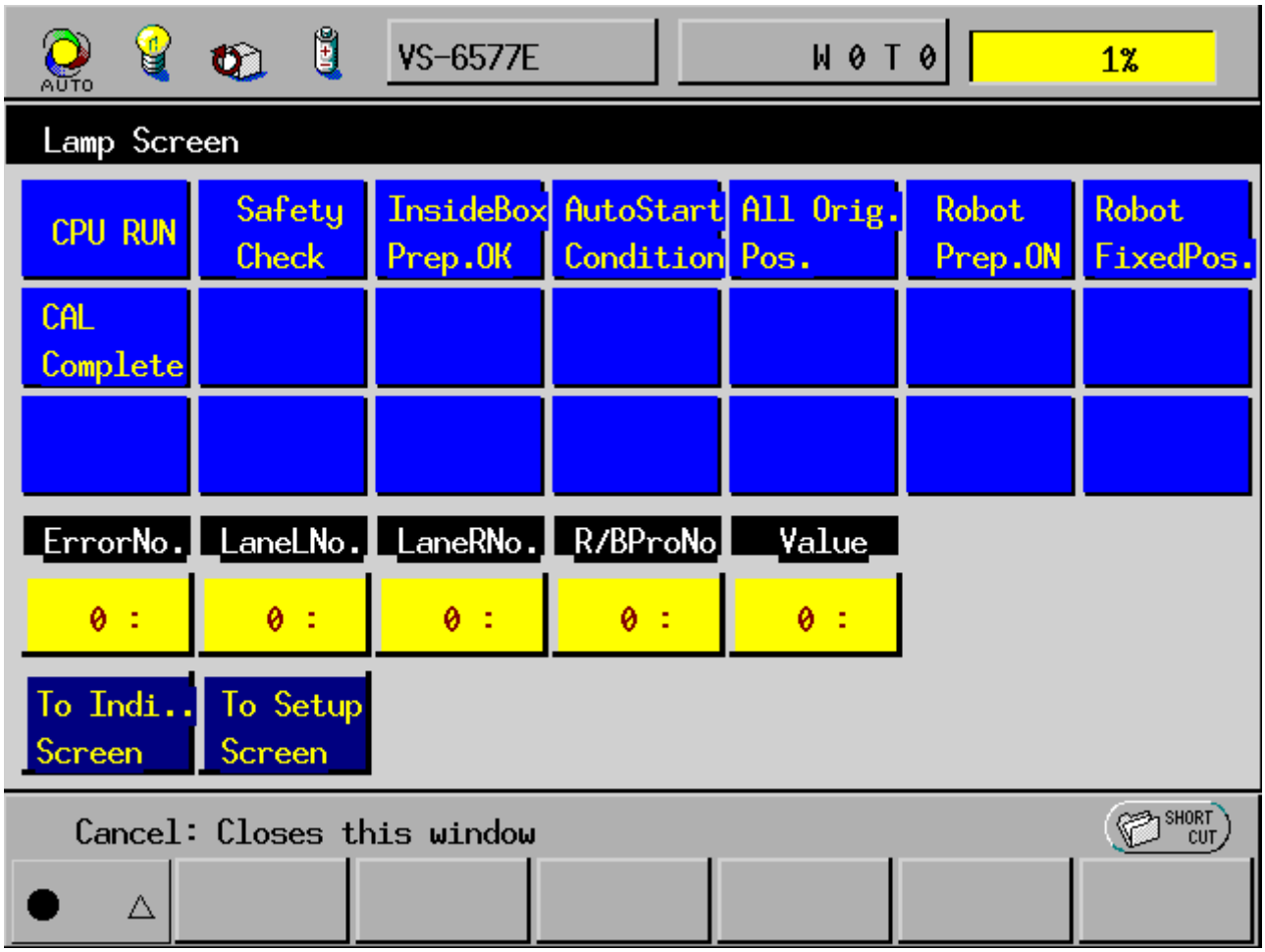
This program is a sample; so, change it according to the actual processing.

Macro definitions

<button.h> is necessary.

Related commands

`Lamp_pl_btns, Lamp_variable_btns, Lamp_pb_btns`



Build_Man_Screen (Sample program) (Ver.1.7 or later)

Function

Creates the individual screen below. (A screen number is required as an argument.)

Explanation

This program calls the PB button creation program and creates the individual screen below.

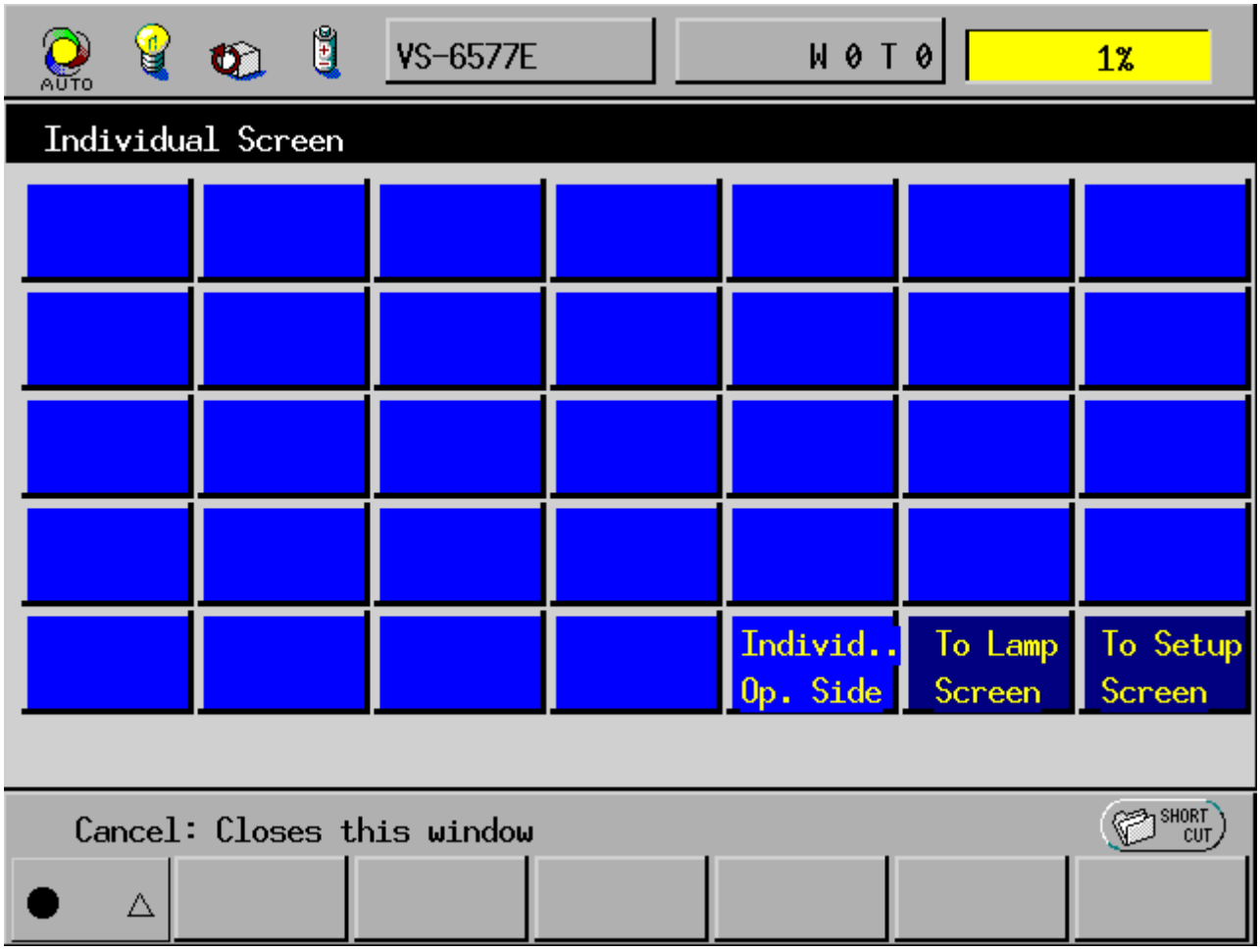
This program is a sample; so, change it according to the actual processing.

Macro definitions

<button.h> is necessary.

Related commands

`Man_pb_btns`



Build_Plan_Screen (Sample program) (Ver.1.7 or later)

Function

Creates the arrangement screen below. (A screen number is required as an argument.)

Explanation

This program calls the buttons creation program and creates the arrangement screen below.

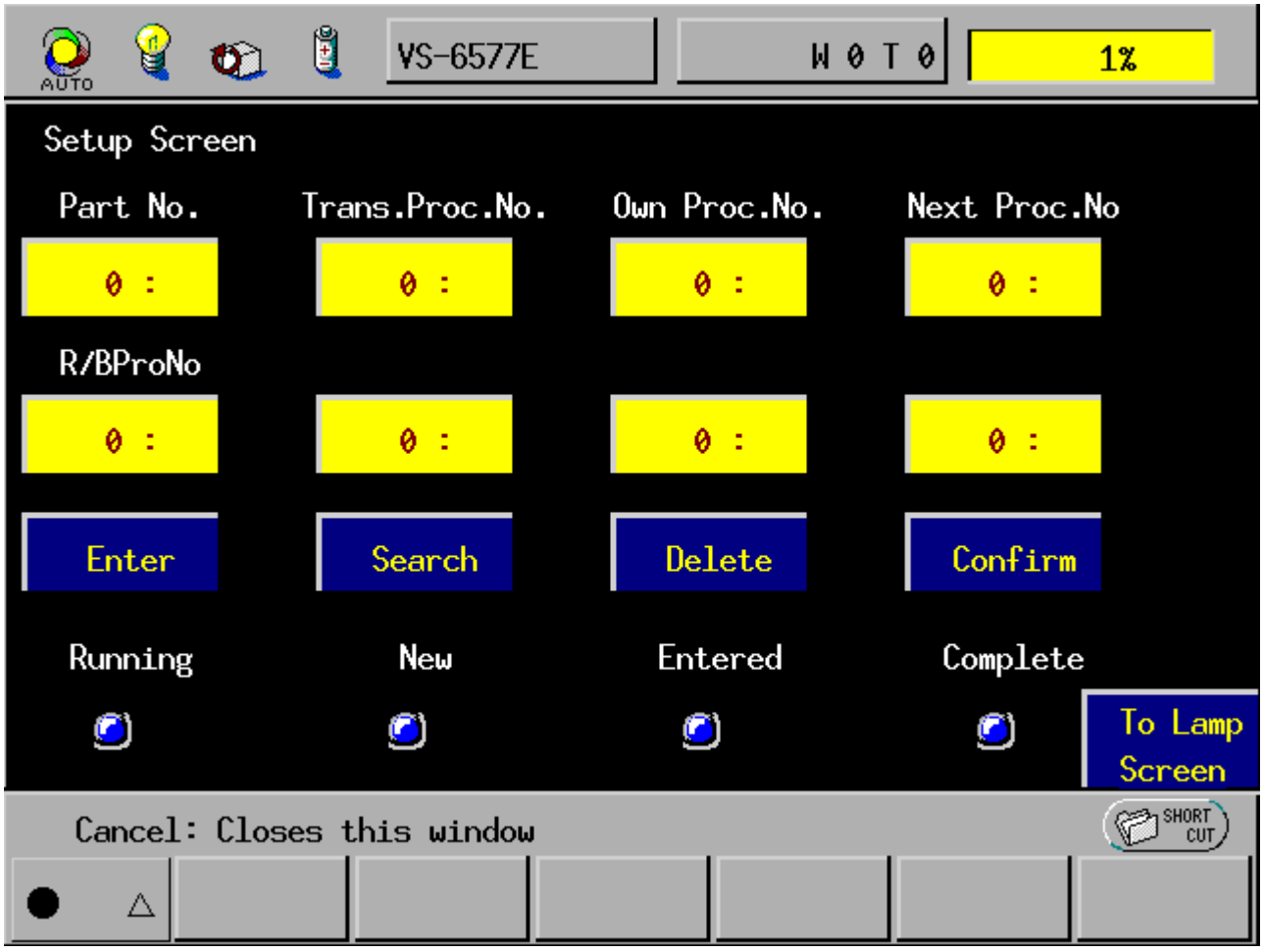
This program is a sample; so, change it according to the actual processing.

Macro definitions

<button.h> is necessary.

Related commands

`Plan_variable_btns, Plan_pb_buns,Plan_pl_btns, Plan_pb_singlebtn`



Lamp_pl_btns (Sample program) (Ver.1.7 or later)

Function

Creates a lamp (LED) on a specified screen. (A screen number is required as an argument.)

Explanation

This program creates a lamp button by entering each parameter required to create the lamp button.

This program is a sample; so, change it according to the actual processing.

Macro definitions

<button.h> is necessary.

Related commands

`make_LED`

Lamp_variable_btns (Sample program) (Ver.1.7 or later)

Function

Creates variable buttons (data display box) on a specified screen. (A screen number is required as an argument.)

Explanation

This program creates variable buttons by entering each parameter required to create them.

This program is a sample; so, change it according to the actual processing.

Macro definitions

<button.h> is necessary.

Related commands

`make_PARAM_BOX, make_LABEL`

Lamp_pb_btns (Sample program) (Ver.1.7 or later)

Function

Creates a PB button (screen switching button) on a specified screen. (A screen number is required as an argument.)

Explanation

This program creates a PB button by entering each parameter required to create a PB button.

This program is a sample; so, change it according to the actual processing.

Macro definitions

<button.h> is necessary.

Related commands

`make_PB`

Man_pb_btns (Sample program) (Ver.1.7 or later)

Function

Creates a PB button on a specified screen. (A screen number is required as an argument.)

Explanation

This program creates a PB button by entering each parameter required to create a button.

This program is a sample; so, change it according to the actual processing.

Macro definitions

<button.h> is necessary.

Related commands

`make_PB`

Plan_variable_btns (Sample program) (Ver.1.7 or later)

Function

Creates a variable entry button (arrangement parameter entry box) on a specified screen. (A screen number is required as an argument.)

Explanation

This program creates a variable button by entering each parameter required to create a button.

This program is a sample; so, change it according to the actual processing.

Macro definitions

<button.h> is necessary.

Related commands

`make_PARAM_BOX, make_LABEL`

Plan_pb_btns (Sample program) (Ver.1.7 or later)

Function

Creates a PB button (data registration processing button) on a specified screen. (A screen number is required as an argument.)

Explanation

This program creates a PB button by entering each parameter required to create a PB button.

This program is a sample; so, change it according to the actual processing.

Macro definitions

<button.h> is necessary.

Related commands

`make_PB`

Plan_pl_btns (Sample program) (Ver.1.7 or later)

Function

Creates a lamp button on a specified screen. (A screen number is required as an argument.)

Explanation

This program creates a lamp button by entering each parameter required to create a lamp button.

This program is a sample; so, change it according to the actual processing.

Macro definitions

<button.h> is necessary.

Related commands

`make_LED, make_LABEL`

Plan_pb_singlebtn (Sample program) (Ver.1.7 or later)

Function

Creates only one button on a specified screen. (A screen number is required as an argument.)

Explanation

This program creates a PB button by entering each parameter required to create a PB button.

This program is a sample; so, change it according to the actual processing.

Macro definitions

<button.h> is necessary.

Related commands

`single_button_set`

22.7.2 Libraries for Creating Operation Panel Screen

make_PB (Library) (Version 1.7 or later)

Function

Creates a PB button.

Syntax

```
make_PB
```

Explanation

This program sets a PB button on the screen on the basis of the specified parameters, display characters, and number of buttons.

Macro definitions

<button.h> is necessary.

Example

```
call make_PB(sysprm(),pb_title(),PB_NUM)
```

make_LED (Library) (Version 1.7 or later)

Function

Creates an LED button.

Syntax

```
make_LED
```

Explanation

This program sets an LED button on the screen on the basis of the specified parameters, display characters, and number of buttons.

Macro definitions

<button.h> is necessary.

Example

```
call make_LED(sysprm(),pl_title(),MAX_PL)
```

make_LABEL (Library) (Version 1.7 or later)

Function

Creates a title (label).

Syntax

```
make_LABEL
```

Explanation

This program sets a title on the screen on the basis of the specified parameters, display characters, and number of buttons.

Macro definitions

<button.h> is necessary.

Example

```
call make_LABEL(sysprm(),pl_title(),PL_NUM)
```

make_PARAM_BOX (Library) (Version 1.7 or later)

Function

Creates a variable button (entry & display box).

Syntax

```
make_PARAM_BOX
```

Explanation

This program sets a variable button on the screen on the basis of the specified parameters and number of buttons.

Macro definitions

<button.h> is necessary.

Example

```
call make_PARAM_BOX(sysprm(),VBTN_NUM)
```

single_button_set (Library) (Version 1.7 or later)

Function

Creates only one button.

Syntax

```
single_button_set
```

Explanation

This program sets only one button by specifying a parameter for a specifeid button number.

Macro definitions

<button.h> is necessary.

Example

```
call single_button_set(btn_no,prm())
```

set_button_param (Library) (Version 1.7 or later)

Function

Specifies button attributes (type, color, shape, and so on).

Syntax

`set_button_param` (<number of displayed buttons>,<number of head buttons>,<displayed button type>,<button character color>,<button background color>,<button use flag>,<button visible flag>,<button correspondence I variable number>,<button support I/O number>,<button display panel number>)

Explanation

This program specifies parameters required to display buttons. In this program, specify 10 parameters in all.

Macro definitions

<button.h> is necessary.

Example

`call set_button_param` (BUTTON_NUM,B_START_NUM,BUTTON_TYPE, CHAR_COLOR,BKGRND_COLOR,BUTTON_USE,BUTTON_VIS,B_VARIABLE_NUM, B_ASSIGN_IO, PANEL_NO)

arrange_button_size (Library) (Version 1.7 or later)

Function

Specifies the button arrangement (position, size, and so on) on the screen.

Syntax

arrange_button_size (<number of displayed buttons>,<button display starting X coordinate>,<button display starting Y coordinate>,<button width>,<button height>,<head button number>,<button horizontal gap>,<button vertical gap>,<button display panel number>)

Explanation

This program uses parameters required to arrange buttons as arguments to determine the button positions on the screen.
This program specifies the button size as an argument parameter for button creation like the arrange_button_pos function.

Macro definitions

<button.h> is necessary.

Example

```
Call arrange_button_size (BUTTON_NUM,ORIGIN_PX,ORIGIN_PY,BUTTON_W,
  BUTTON_H,B_START_NUM,H_GAP,V_GAP,PANEL_NO)
```

arrange_button_pos (Library) (Version 1.7 or later)

Function

Specify the button arrangement (position, size, and so on) on the screen.

Syntax

arrange_button_pos (<number of displayed buttons>,<button display starting X coordinate>,<button display starting Y coordinate>,<number of buttons on screen width>,<number of buttons on screen height>,<head button number>,<horizontal button gap>,<vertical button gap>,<button display panel number>)

Explanation

This program uses parameters required to arrange buttons as arguments to determine the button positions on the screen.

This program specifies the number of buttons on the height of the screen as an argument parameter for button creation like the set_button_size function.

Macro definitions

<button.h> is necessary.

Example

```
call arrange_button_size (BUTTON_NUM,ORIGIN_PX,ORIGIN_PY,PANEL_M,
PANEL_N,,B_START_NUM,H_GAP,V_GAP,PANEL_NO)
```

Chapter 4

Other Advanced Features

This chapter explains enhanced features of the vision control and WINCAPSII and newly supported features such as serial binary transmission, writing of arithmetic, logical and relational operators in a same line, and setting of a field network error display parameter.

Simplified palletizing and newly supported error codes are also described in this chapter.

Chapter 4 Other Advanced Features

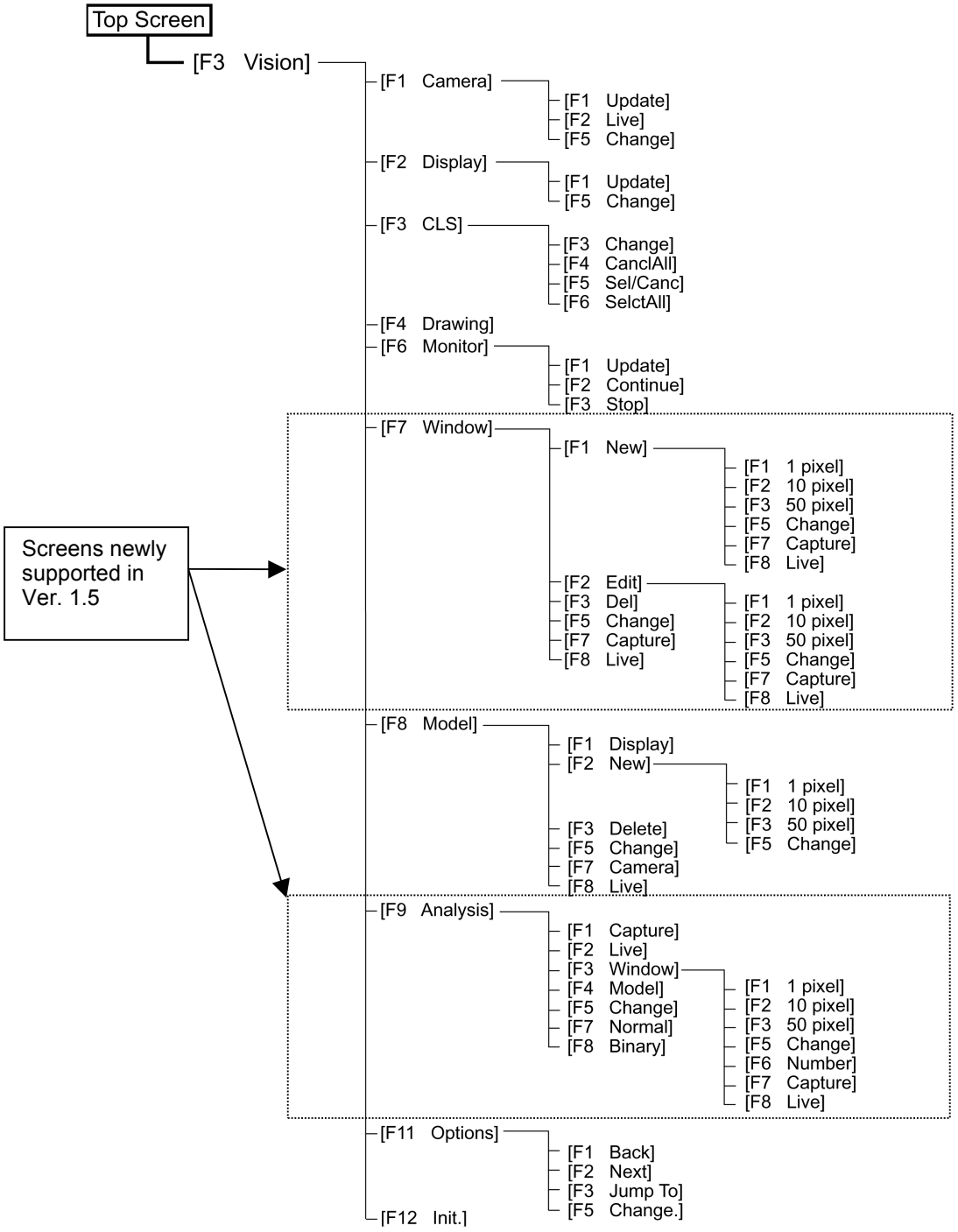
4.1 New Vision Features (Version 1.5 or later)

The following functions have been newly added to the vision control in Main Software Version 1.5 or later:

(1) Creating, editing, and deleting windows, which allows you to easily handle windows with the teach pendant.

(2) Analyzing images (e.g., model search, labeling, area/center of gravity/major axis angle, edge), which allows you to use those analyzing features without the aid of programs, helping you develop programs and process images.

These features are located in the Vision-related menu tree as shown below.

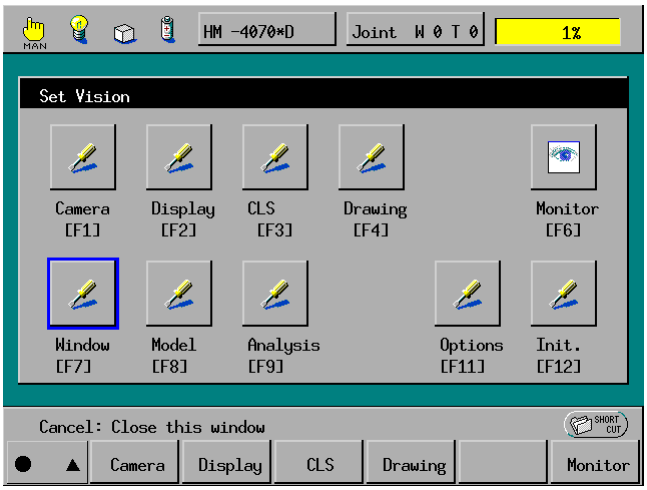


Displaying the Vision Menu

Access: [F3 Vision]

Pressing [F3 Vision] on the top screen will display the Set Vision menu as shown below.

The [F7 Window] and [F9 Analysis] are newly supported in Version 1.5 or later.



NOTE: Before using the vision control [F3 Vision], make sure that:

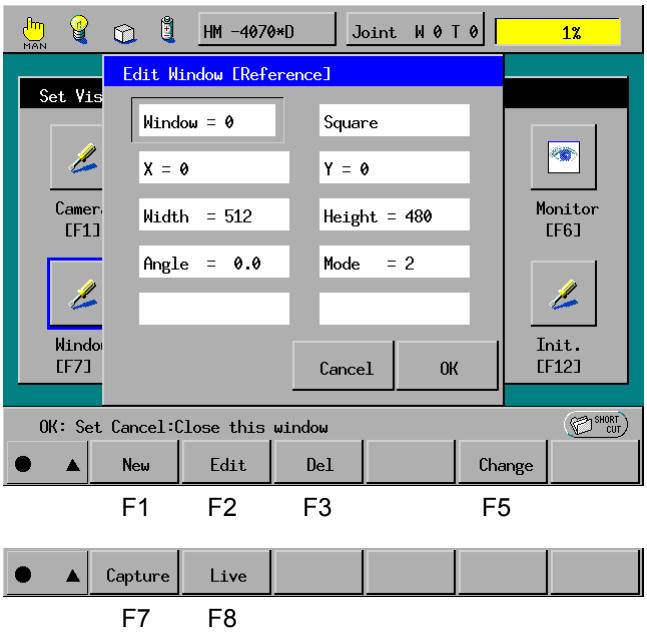
- An optional µVision board is integrated in the robot controller,
- The robot is placed in Manual mode, and
- The vision semaphore is released (no TAKEVIS obtained).

Browsing windows to be used in image analysis

Access: [F3 Vision]—[F7 Window]

Browses the parameter values of windows to be used in image analysis and allows you to monitor the frame of the specified window.

(1) Press [F7 Window] in the Set Vision window, and the Edit Window will display as shown below.



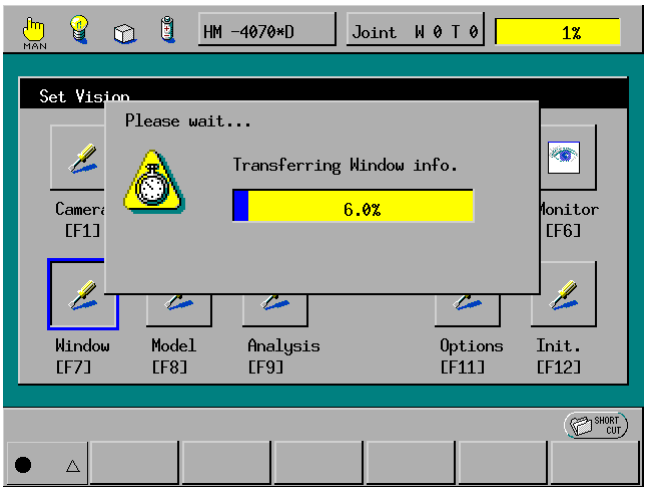
Parameters		Description
Window No.	:	Number assigned to a window (0 to 255)
Window type	:	Shape of a window (Square, line, circle, ellipse, or sector)
X origin	:	X-coordinate origin of a stored window (0 to 511)
Y origin	:	Y-coordinate origin of a stored window (0 to 480)

Other parameters differ depending upon window shapes. For further details regarding window shapes, refer to "WINDMAKE" in the PROGRAMMER'S MANUAL.

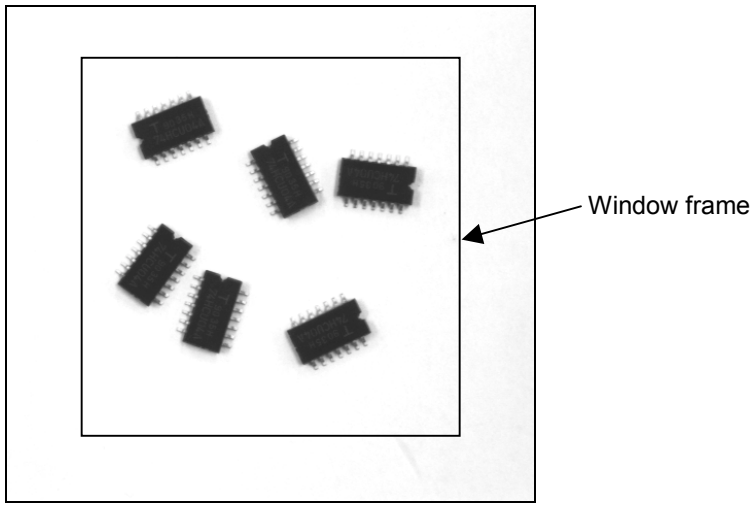
Function keys available	
[F1 New]	Creates, edits, and saves a new window (Edit mode)
[F2 Edit]	Edits a window already stored (Edit mode)
[F3 Del]	Deletes data of the selected window number. The deleted data will be completely lost.
[F5 Change]	Changes the window number.
[F7 Capture]	Captures a camera image and displays it on the process screen.
[F8 Live]	Switches to a camera image.

(2) Press [OK] to display the frame of the selected window.

During execution of "New," "Edit" or "Del," a progress bar will appear as shown below since it takes several seconds to retrieve necessary data from the µVision board.



Shown below is a frame example of a window number selected. (The colors are not the same as the original image colors).



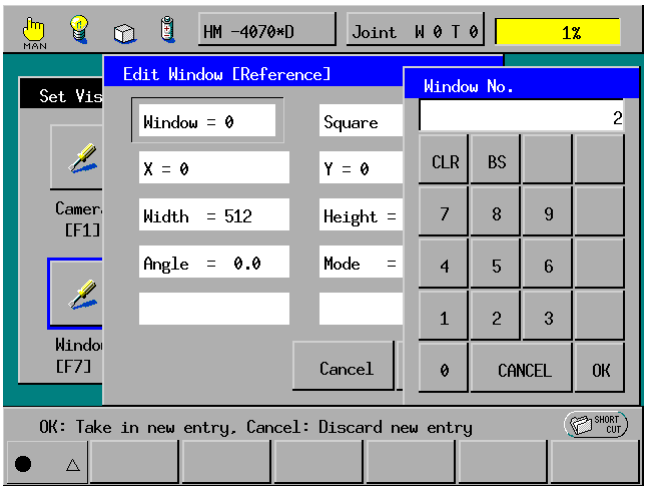
Creating, editing and saving a new window (Edit mode)

Access: [F3 Vision]—[F7 Window]—[F1 New]

Creates, edits and saves a new window.

(1) Press [F1 New] in the Edit Window, and the numerical keypad will appear as shown below.

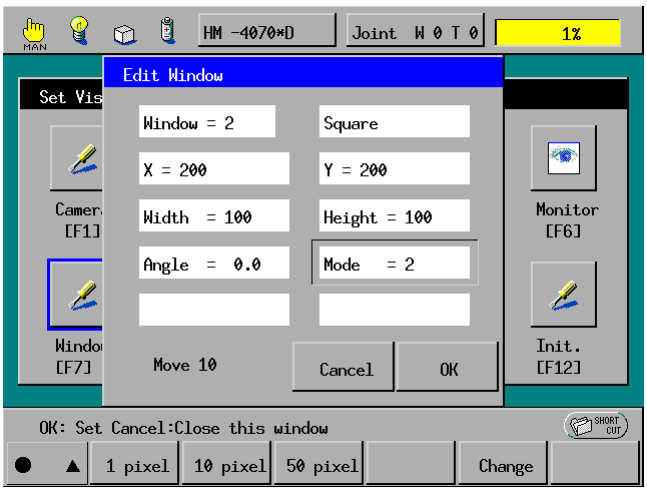
(2) Enter the number of a new window to be created.



Window No. : Number assigned to a window (0 to 255)
Window type : Shape of a window (Square, line, circle, ellipse, or sector)
X origin : X-coordinate origin of a stored window (0 to 511)
Y origin : Y-coordinate origin of a stored window (0 to 480)

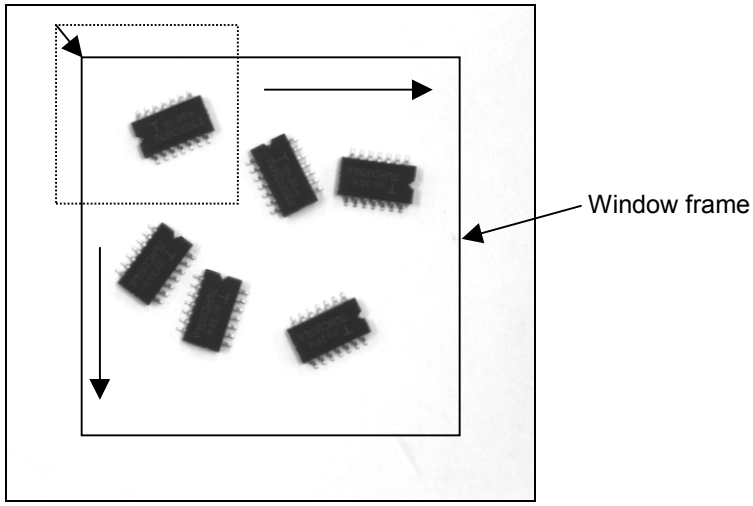
Other parameters differ depending upon window shapes. For further details regarding window shapes, refer to "WINDMAKE" in the PROGRAMMER'S MANUAL.

(3) Press [OK]. The system message will appear and then the following Edit Window will display.



Function keys available	
[F1 1 pixel]	Specifies the movement quantum in units of 1 pixel.
[F2 10 pixel]	Specifies the movement quantum in units of 10 pixels.
[F3 50 pixel]	Specifies the movement quantum in units of 50 pixels.
[F5 Change]	Changes each model data.
[F7 Capture]	Captures a camera image and displays it on the process screen.
[F8 Live]	Switches to a camera image.

(4) You may change the size of the window by modifying the parameters in the window and check the changed size on the monitor.

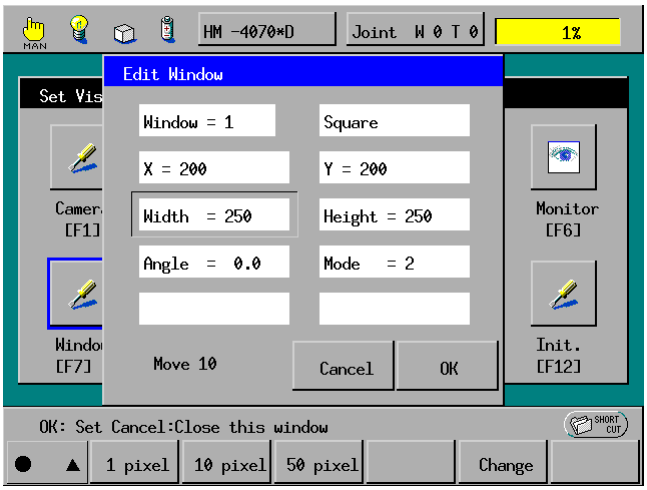


Editing and saving an existing window (Edit mode)

Access: [F3 Vision]—[F7 Window]—[F2 Edit]

Edits and saves a new window.

(1) Press [F2 Edit] in the Edit Window, and the following window will appear.



Window type : Shape of a window (Square, line, circle, ellipse, or sector)
X origin : X-coordinate origin of a stored window (0 to 511)
Y origin : Y-coordinate origin of a stored window (0 to 480)

Other parameters differ depending upon window shapes. For further details regarding window shapes, refer to "WINDMAKE" in the PROGRAMMER'S MANUAL.

(2) Other operating procedure is the same as in [F1 New].

Function keys available	
[F1 1 pixel]	Specifies the movement quantum in units of 1 pixel.
[F2 10 pixel]	Specifies the movement quantum in units of 10 pixels.
[F3 50 pixel]	Specifies the movement quantum in units of 50 pixels.
[F5 Change]	Changes each model data.
[F7 Capture]	Captures a camera image and displays it on the process screen.
[F8 Live]	Switches to a camera image.

Analyzing images

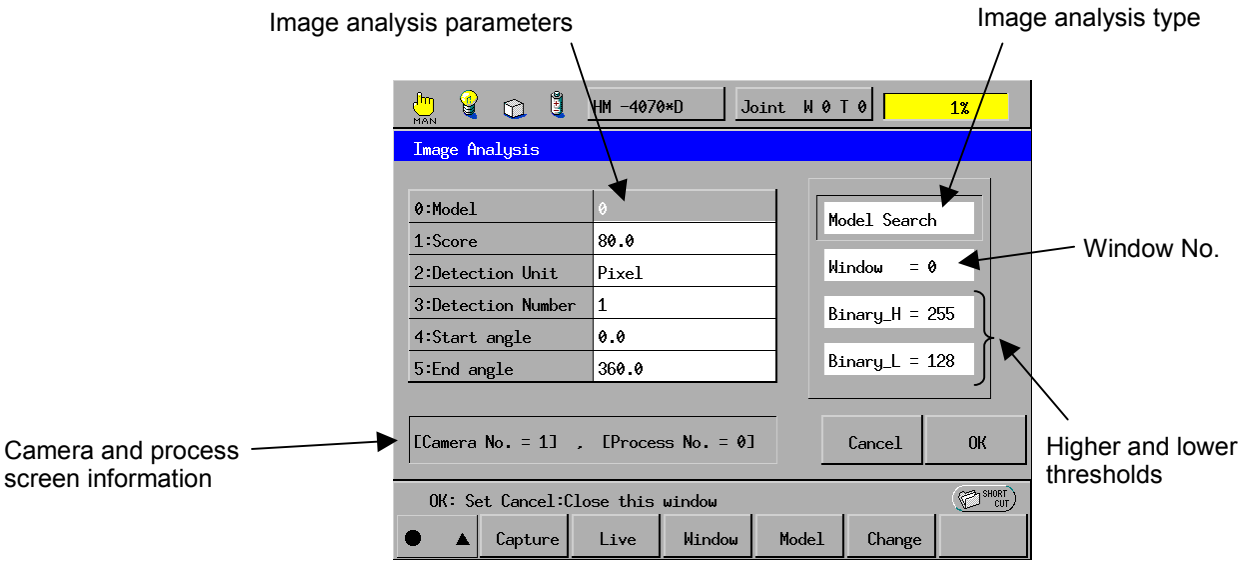
Access: [F3 Vision]—[F9 Analysis]

Allows you to analyze images temporarily from the teach pendant without setting up corresponding programs.

Types of image analysis

Image analysis functions	Image processing instructions	Remarks
Model search	SHMODEL	
Labeling	BLOB	
Edge finding	VISEDGE	
Area/Center of gravity/Major axis angle	VISMEASURE	Extracts features such as area, center of gravity, and major axis angle.
QR code	VISREADQR	Reads QR codes.
Filter processing	VISFILTER	Filters input screens.
Circle search	SHCIRCLE	
Corner search	SHCORNER	

(1) Press [F9 Analysis] in the Set Vision window, and the Image Analysis window will display as shown below.



Window No. : Number of the target window to be processed.

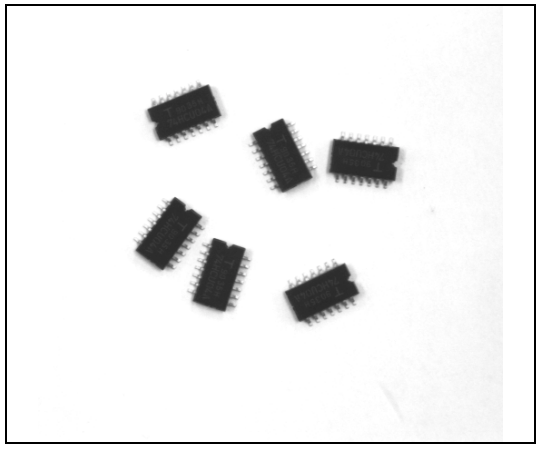
Binary vision parameters : Specifies the upper and lower limits for binary vision.

Function keys available	
[F1 Capture]	Captures a camera image and displays it on the process screen.
[F2 Live]	Switches to a camera image.
[F3 Window]	Sets the shape and size of a temporary window.
[F4 Model]	Temporarily displays the image of the model to be searched in model search.
[F5 Change]	Changes parameter values.
[F7 Normal]	Switches to the normal vision where a grayscale image displays in 256-tone.
[F8 Binary]	Switches to the binary vision where a binary image displays in 2-tone.

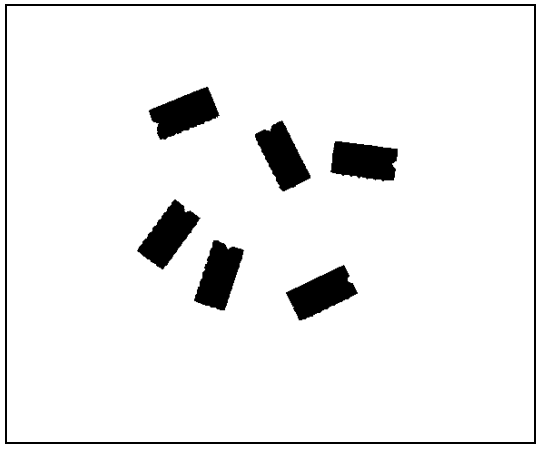
(2) Select the desired image analysis.

Pressing [F8 Binary] will show a binary image based on the higher and lower threshold values you have entered. A camera live image may also display as a binary image in real-time.

Normal vision



Binary vision



4.2 Serial Binary Transmission (Version 1.5 or later)

The conventional robot controllers can transmit only ASCII codes via the RS-232C ports. The new controller may support byte-controlled binary data transmissions, increasing the types of connectable communications devices.

4.2.1 Using Serial Binary Transmission

Connect the robot controller to external equipment with an RS-232C interface cable and use the following commands:

■ Data input/output commands

(1)	`printb`	Output a single byte of data.
(2)	`inputb`	Input a single byte of data.
(3)	`lprintb`	Output multiple bytes of data.
(4)	`linputb`	Input multiple bytes of data.
(5)	`com_encom`	Enable COM port.
(6)	`com_discom`	Disable COM port.
(7)	`com_state`	Get COM port status.

■ Transmission system

(1) Transmission method : RS-232C
(2) Transmission port : Controller RS-232C port, μVision board RS-232C port
(3) Pin array : Same as conventional
(4) Transmission status :

Controller RS-232C port	Variable, same as the previous controllers
μVision board RS-232C port	Transmission speed : 9600bps Bit length : 8 Parity bit : None Stop bit : 1 Flow control : None

4.2.2 Serial Binary Transmission Commands

printb

Function

Outputs a single byte of data to the RS-232C port.

Format

```
printb #<portnumber>,<integervarnumber>
```

`<portnumber>` Output port number
(1: Controller RS-232C port, -1: µVision RS-232C port)

`<integervarnumber>` Integer variable number where output data is stored

Explanation

This command outputs the lower byte of data assigned to `<integervarnumber>` to the port specified by `<portnumber>`.

Example

```
'!TITLE "<Title>"
PROGRAM sample
            .
            .
            .
            printb #1,I10           'Output the lower byte of data stored in I10
                                    'to RS-232C port
            .
            .
            .
end
```

inputb

Function

Inputs a single byte of data from the RS-232C port.

Format

```
inputb #<portnumber>,<integervarnumber>
```

`<portnumber>` Input port number
(1: Controller RS-232C port, -1: μVision RS-232C port)

`<integervarnumber>` Integer variable number where input data is to be stored

Explanation

This command stores a single byte of data inputted from the specified port, into `<integervarnumber>`.

NOTE: If no data exists in the specified port, executing this command will result in an error. Before execution of this command, transfer data from external equipment. To check whether any data exists or not, use `com_state` command.

Example

```
'!TITLE "<Title>"
PROGRAM sample
            .
            .
            .
            inputb #1,I10          'Store data inputted from RS-232C port into I10
            .
            .
            .
end
```

lprintb

Function

Outputs multiple bytes of data to the RS-232C port.

Format

```
lprintb #<portnumber>,<arrayheadelement>,<outputbytes>
```

<portnumber> Output port number
(1: Controller RS-232C port, -1: μVision RS-232C port)

<arrayheadelement> Head element of an array where output data is stored

<outputbytes> Number of bytes to be outputted

Explanation

This command outputs the specified number of bytes of data from the specified element of an array where the output data is stored, to the specified port.

Example

```
'!TITLE "<Title>"
PROGRAM sample
            .
            .
            .
            lprintb #1,I64,30       'Output the lower byte of data from I64 to
                                    'I93 in succession to RS-232C port.
            .
            .
            .
end
```

linputb

Function

Inputs multiple bytes of data from the RS-232C port.

Format

```
linputb #<portnumber>,<arrayheadelement>,<inputbytes>
```

`<portnumber>`	Input port number (1: Controller RS-232C port, -1: μVision RS-232C port)
`<arrayheadelement>`	Head element of an array where input data is to be stored
`<inputbytes>`	Number of bytes to be inputted

Explanation

This command inputs the specified number of bytes of data from the specified element of an array, to the specified port.

NOTE: If no data exists in the specified port, executing this command will result in an error. Before execution of this command, transfer data from external equipment. To check whether any data exists or not, use `com_state` command.

Example

```
'!TITLE "<Title>"
PROGRAM sample
            .
            .
            .
            linputb #1,I64,30       'Input data from I64 to I93 in succession from RS-232C
                                    'port.
            .
            .
            .
end
```

com_encom

Function

Enables the RS-232C port only for binary transmission.

Format

```
com_encom #<portnumber>
```

Explanation

This command discriminates binary data from ASCII data in binary transmission between the PC and robot controller e.g., in WINCAPSII.

After binary transmission is completed, you need to release the RS-232C port by using com_discom command.

NOTE: If data is transferred from the PC (e.g., in WINCAPSII) to the robot controller after execution of this command, then the controller will treat it as binary data and will not close the RS-232C port occupied by binary transmission.

Example

```
'!TITLE "<Title>"
PROGRAM sample
            .
            .
            .
            com_encom #1            'Make port exclusive for binary transmission
            .
            .
            .
end
```

com_discom

Function

Releases the RS-232C port from binary transmission.

Format

```
com_discom #<portnumber>
```

Explanation

This command disables com_encom command to release the RS-232C dedicated to binary transmission for other uses.

NOTE: Executing this command clears the RS-232C port once for preventing data confusion between ASCII and binary data.

Example

```
'!TITLE "<Title>"
PROGRAM sample
            .
            .
            .
            com_discom #1           'Release port from binary transmission
            .
            .
            .
end
```

com_state

Function

Gets the status of RS-232C port.

Format

```
com_state #<portnumber>,<integervar>
```

`<integervar>` Integer variable where the port status is stored

Explanation

This command gets bytes of data remaining in the transmission buffer, into the integer variable specified by `<integervar>`.

Note that -1 will be returned if a transmission port error occurs.

Example

```
'!TITLE "<Title>"
PROGRAM sample
            .
            .
            .
            com_state #1,I280       'Gets data remaining in transmission buffer,
                                    'into I280
            .
            .
            .
end
```

4.3 What's New in WINCAPSII (Versions 1.5 & 1.6)

The table below lists features newly added in WINCAPSII Version 1.5 and Version 1.6.

4.3.1 Overview

No.	Items	Description
1	Required operating environments	New specifications of the personal computer, memory capacity, and hard disk, which are required for running WINCAPSII Version 1.5 or later smoothly.
2	Installation of WINCAPSII	Note added for installing WINCAPSII.
3	Starting WINCAPSII (Version 1.5 or later)	Choice of the desired project at the start of WINCAPSII. (In the earlier versions, no choice is allowed. The most recently used project opens automatically.)
4	System Manager (Version 1.5 or later)	(1) New user interface that makes it easier to choose a robot when you create a new project. (2) "Save Project As" newly added to the File menu.
5	• Variable Manager • DIO Manager • Vision Manager (Version 1.5 or later)	Import of a macro definition file created in other projects into the currently active project.
6	Arm Manager (Version 1.5 or later)	"Permission of CALSET transmission" and "Use hardware accelerator" newly added to the Display tab of Options window. (1) Only when the "Permission of CALSET transmission" is selected, CALSET data (RANG, CALSET values) may be transmitted. This is to avoid unexpected transmission of CALSET data. (2) If enabled, the hardware accelerator will make drawing faster on screens. Depending upon graphics drivers installed in PCs, the hardware accelerator may cause Arm Manager to fail to draw images correctly.
7	Log Manager (Version 1.5 or later)	(1) Import function added Imports "Servo Joint Log" saved in CSV format and plots the graph. (2) Export function added Saves "Servo Joint Log" in CSV format and allows it to be opened from other applications (e.g., Excel).
8	DIO Manager (Version 1.6 or later)	(1) "I/O Allocation" box added to the Hardware tab in the Options dialog box (called by selecting the Tool command) in DIO Manager, which is associated with newly supported optional I/O hardware. (2) I/O hardware-related parameters added in DIO Manager

4.3.2 Required operating environments

Refer to the INSTALLATION & MAINTENANCE GUIDE, page 3-14.

The following operating environments are required for running WINCAPSII smoothly.

Table 3-5 Operating Environments for the PC Teaching System Software

CPU	Pentium or higher capacity
OS	Windows 95 or upper version (See Note 1.)
Memory	32 MB or more (64 MB recommended)
Hard disk	A free area of 80 MB or more is required at installation.
Monitor resolution	640 × 480 or higher

Note 1 WINCAPSII cannot run properly on earlier versions of Windows 95.

The version of Windows 95 can be checked with [Control Panel – System – Information]. If A, B or C is not displayed (no symbol) at the end of the version information (4.00, 95B), update your Windows 95 with the Windows 95 Service Pack 1 that is available from the Microsoft's web site.

4.3.3 Note Added for Installing WINCAPSII

Refer to the WINCAPSII GUIDE, page 2-1.

Note (3) given below has been added.

Note (1)	If the WINCAPSII software is already installed on the computer, first uninstall the existing WINCAPSII software and then reinstall it. For uninstall procedure, refer to "2.1.2 Uninstall".
Note (2)	Always install or uninstall the software after quitting all the applications currently running. Shared components in operation cannot be installed or uninstalled. If a shared component that WINCAPSII is to use is being used by another application, the install/uninstall process may fail.
Note (3)	**During installation of WINCAPSII, the message "Error has occurred during file copying" may appear depending upon the PC environments. This is because any other application is using the file to which WINCAPSII is attempting to overwrite. Ignore the message and continue installation.**

4.3.4 Starting WINCAPSII (Version 1.5 or later)

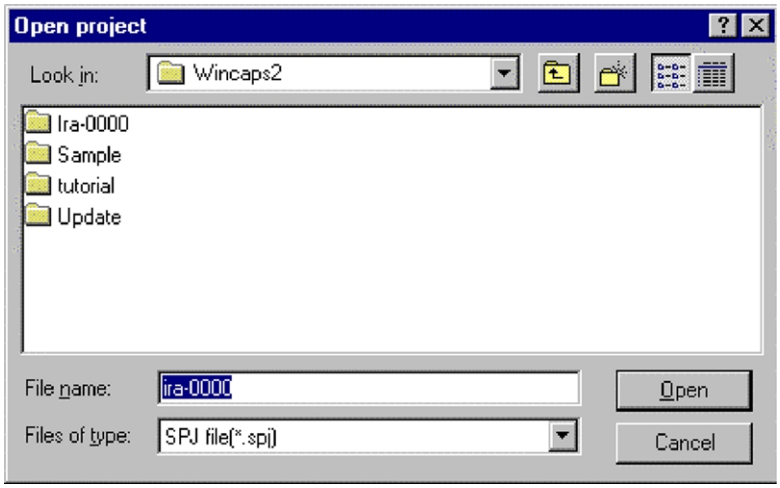
Refer to the WINCAPSII GUIDE, page 3-1.

■ Choice of the desired project at the start of WINCAPSII

In Ver. 1.4 or earlier, if you select Start—Programs on the Windows screen and choose "DENSO System Manager," then the Password window will display and then the project most recently used will open.

In Ver. 1.5, choosing "DENSO System Manager" at the second start or later will display the "Open project" dialog box where you may choose a desired project.

The "File name" in the "Open project" dialog box shows the file most recently used, by default.



"Open project" dialog box

4.3.5 New Features in System Manager (Version 1.5 or later)

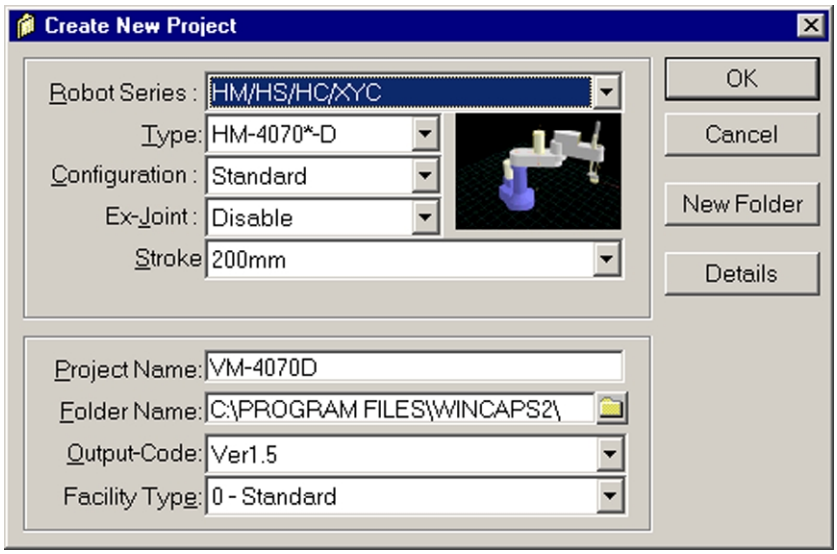
Refer to the WINCAPSII GUIDE, page 4-9.

■ New user interface that makes it easier to choose a robot when you create a new project

In Ver. 1.4 or earlier, the "Create New Project" window has a "Robot Type" selection box.

In Ver. 1.5, the window has "Robot Series" and "Type" selection boxes as shown below, helping you choose your robot series and model.

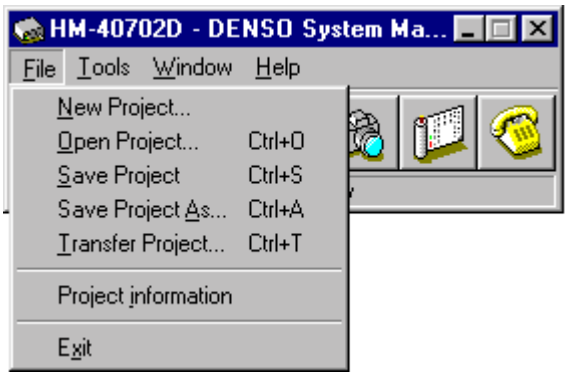
In addition, the "Configuration" selection box is newly provided so that you may select ANSI or other environments.



■ "Save Project As" newly added to the File menu

The "Save Project As" is newly added to the File menu of System Manager.

Choosing it will save the active project with a different name, location, or file format. Files used in the active project will be also copied into the new project.



"File Menu" of System Manager

4.3.6 New Features in Variable Manager, DIO Manager and Vision Manager (Version 1.5 or later)

Refer to the WINCAPSII GUIDE, pages 6-13, 7-8, 9-10.

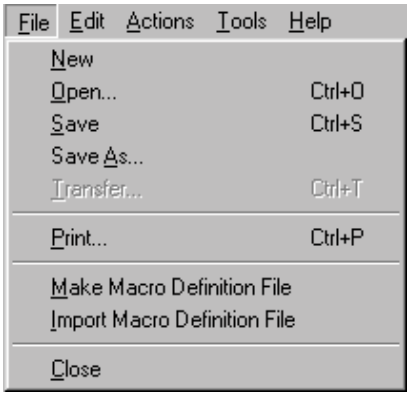
■ Import of a macro definition file created in other projects into the currently active project

The "Import Macro Definition File" command is newly added to the File menu in Variable Manager, DIO Manager, and Vision Manager as shown below.

If you choose the "Import Macro Definition File," each Manager operates as follows:

(1) In Variable Manager

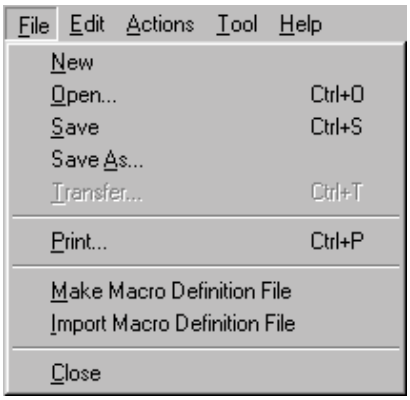
Opens macro definition files (var_tab.h) and shows them in the "Use" and "Macro name" columns of each variable.



File menu of Variable Manager

(2) In DIO Manager

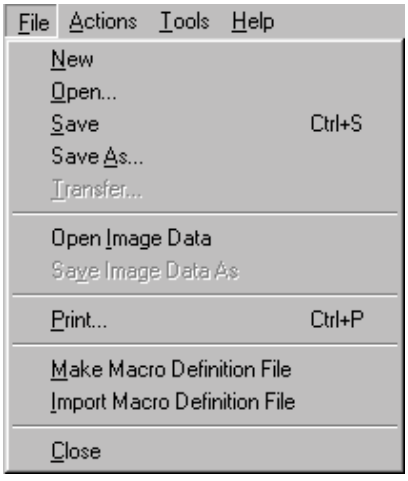
Opens macro definition files (dio_tab.h) and shows them in the "Use" and "Macro name" columns on the screen.



File menu of DIO Manager

(3) In Vision Manager

Opens macro definition files (vis_tab.h) and shows the in the macro editing window.

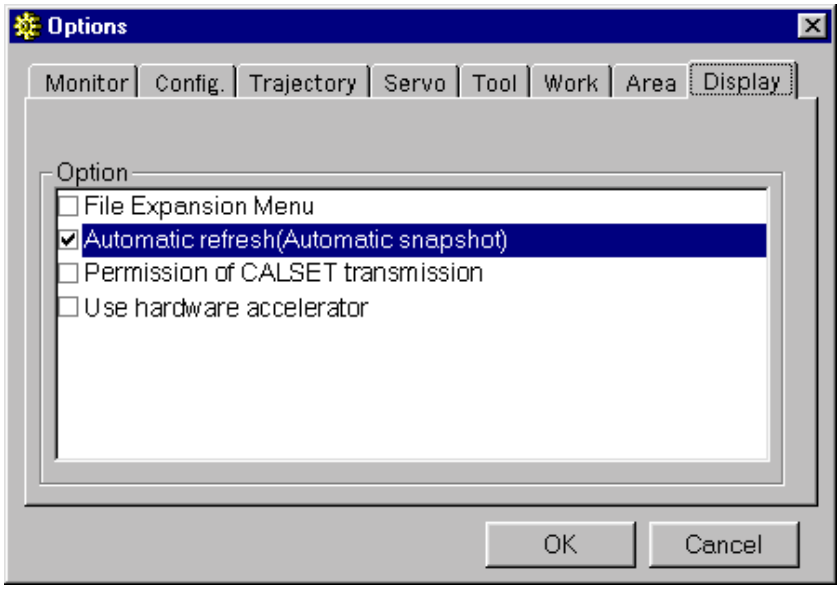


File menu of Vision Manager

4.3.7 New Features in Arm Manager (Version 1.5 or later)

Refer to the WINCAPSII GUIDE, page 8-22.

■ "Permission of CALSET transmission" and "Use hardware accelerator" newly added to the Display tab of Options window



(1) Only when the "Permission of CALSET transmission" is selected, CALSET data (RANG, CALSET values) may be transmitted. This is to avoid unexpected transmission of CALSET data.

This option is not selected by default.

Note: The RANG and CALSET values differ in each robot. When transmitting them, make sure that correct data is set up for transmission. If the robot controller receives incorrect data, the robot will not run normally.

For details about "CALSET," refer to the INSTALLATION & MAINTENANCE GUIDE.

(2) You may choose the use of hardware accelerator.

If enabled, the hardware accelerator will make Arm Manager draw faster on screens according to the "Graphic hardware accelerator" settings of your PC.

If you modify this setting, restart WINCAPSII to make it go into effect.

Note: Depending upon graphics drivers installed in PCs, the hardware accelerator may cause Arm Manager to fail to display correctly in some cases. If such occurs, disable the hardware accelerator.

4.3.8 New Features in Log Manager (Version 1.5 or later)

Refer to the WINCAPSII GUIDE, page 10-10.

■ Import and export of "Servo Joint Log" in CSV format

The "Import" and "Export" commands are newly added to the File menu in Log Manager as shown below.

The Import command reads in the "Servo Joint Log" saved in CSV format and plots the graph.

The Export command saves the "Servo Joint Log" in CSV format and allows it to be opened from other applications (e.g., Excel).

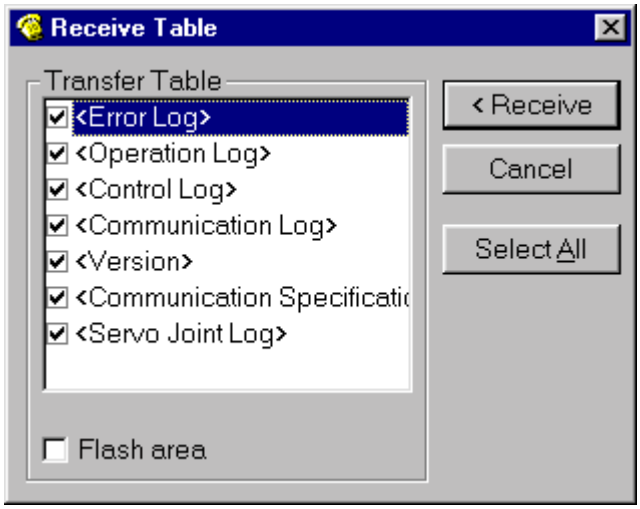
File menu of Log Manager

Plotting the graph of the "Servo Joint Log"

(1) Get the Servo Joint Log according to either of the following two procedures:

Getting "Servo Joint Log" from the robot controller

In the Receive Table box, select <Servo Joint Log> and click the <Receive button.

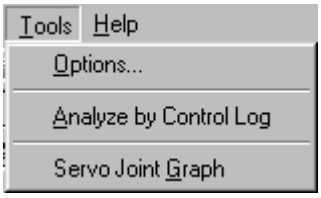


Receive Table box

Reading in data saved with Export command

Choose the Import command from the File menu to specify a CSV format file to be opened.

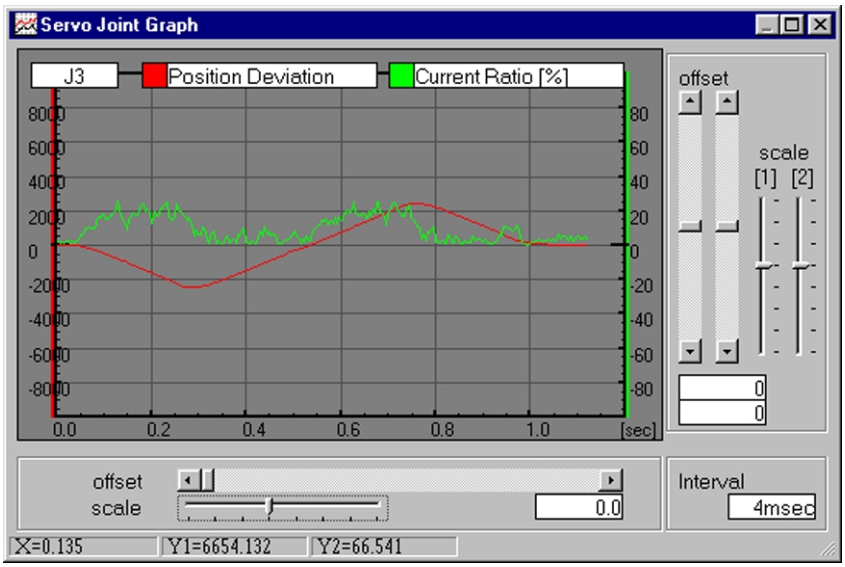
(2) Choose the "Servo Joint Graph" command from the Tools menu to plot the graph.



Tools menu

(3) Shown below is a servo joint graph.

Note: For more details about joint numbers and items of received data, refer to Subsection 3.2.3.2 "[2] Monitor of single-joint servo data."



Servo Joint Graph screen

4.3.9 I/O Hardware-related Settings Added in DIO Manager (Version 1.6 or later)

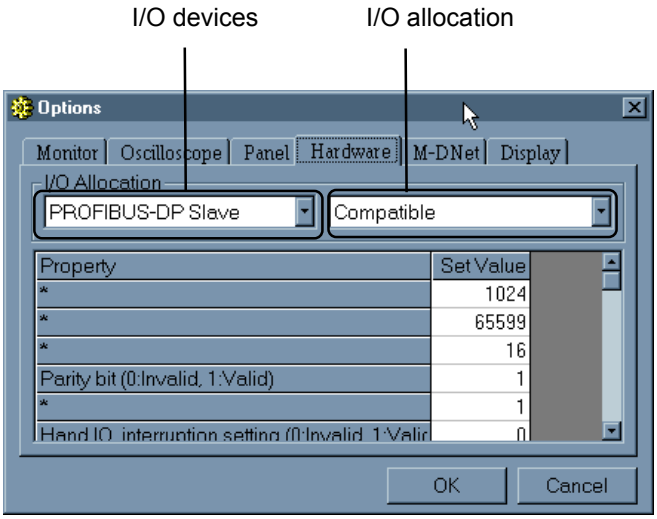
Associated with newly supported optional I/O hardware, the "I/O Allocation" box is newly added to the Hardware tab in the Options dialog box called by selecting the Tool command in DIO Manager.

■ "I/O Allocation" box newly added to the Hardware tab

Refer to the WINCAPSII GUIDE, page 7-25.

This Hardware tab is provided for setting various parameters needed for DIO operation. For detailed about these parameters and element numbers, refer to the PROGRAMMER'S MANUAL, Appendix "8 Setting Parameter Table."

The newly added "I/O Allocation" box consists of the I/O device choice box (left-hand) and allocation choice box (right-hand).



Hardware Tab in the Options Dialog Box

■ I/O hardware-related parameters added in DIO Manager

As listed below, the Parameter Table for DIO Manager (given in the PROGRAMMER'S MANUAL) is modified.

Refer to the PROGRAMMER'S MANUAL, "8. Setting Parameter Table" on page Appendix-28.

■ Dio Manager - Hardware

Parameter name	Macro name	Description
Assignment mode (0: compatible, 1: standard)	IO_ASSIGN	I/O assignment mode setting (0: compatible mode, 1: standard mode)
Parity bit (0: invalid, 1: valid)	IO_PARITY	Setting of program No. selection parity in program start Parity bit (0: invalid, 1: valid)
Hand IO. Interruption setting (0: invalid, 1: valid)	IO_HD_ENABLED	--
DeviceNet. Input slot number	IO_DN_IN_SLOT	--
DeviceNet. Output slot number	IO_DN_OUT_SLOT	--
DeviceNet. Preparation failure detection (0: invalid, 1: valid)	IO_DN_EXERRCHK	--
Dedicated I/O Output Mode (0: invalid, 1: valid)	IO_DEDICTDIO	--
M-Dnet Error Indication (0: Every time, 1: First time)	IO_MDN_ERRDISP	--
PROFIBUS node address (1 to 125)	IO_SPRFI_ADRS	--
PROFIBUS Input Setting (0:8, 1:12, 2:16, 3:20, 4:32)	IO_SPRFI_INTYPE	PROFIBUS input setting 0: 8 bytes Output con 1: 12 bytes Output con 2: 16 bytes Output con 3: 20 bytes Output con 4: 32 bytes Output con
PROFIBUS Output Setting (0:8, 1:12, 2:16, 3:20, 4:32)	IO_SPRFI_OUTTYPE	PROFIBUS output setting 0: 8 bytes Input con 1: 12 bytes Input con 2: 16 bytes Input con 3: 20 bytes Input con 4: 32 bytes Input con

4.4 Operator Precedence (Version 1.5 or later)

In Ver. 1.5, one program line may contain arithmetic operators, logical operators, and relational operators together.

The precedence of those operators is shown below.

Operators	Operations	Precedence
^	Index (Exponent)	High
-	Negation	↑
*, /, ., x	Multiplication, division, inner product, outer product	
MOD	Modulo arithmetic	
+, -	Addition, subtraction	
NOT	Logical negation	
AND	Logical multiplication	
OR	Logical addition	
XOR	Exclusive logical addition	↓
=, =., <>, <, >, <=, >=	Relational operators	Low

When more than one operator occurs at the same level of precedence, the Main Software resolves the expression by proceeding from left to right.

The parentheses ( ) allow you to override operator precedence; that is, operations enclosed with parentheses are first carried out.

Example of operator precedence

1. `if IO128 = ON and IO129 = OFF then`
   (1) (2) (3)

2. `if (IO128 = ON) and (IO129 = OFF) then`
   (1) (2) (3)

In the above examples, the operation order is (1) through (3).

Program samples

[Ver. 1.4 or earlier]

```
if IO128 ON then
       if IO129 OFF then
              move p, p1
       endif
endif
```

[Ver. 1.5 or later]

```
if IO128 ON and IO129 OFF then
       move p, p1
endif
```

NOTE: `NOT` operator

The `NOT` operator is one of the logical bit operators, so it cannot negate any evaluation result of an expression such as I1=I2. (Refer to the PROGRAMMER'S MANUAL, Subsection 7.9.4.)

The following sample cannot negate the evaluation result:

```
if NOT (I1=I2) then…
```

If the result of I1=I2 is Truth, the NOT operator cannot make it False.

To negate the evaluation result, write as follows:

```
if (I1=I2) = FALSE then…
```

4.5 Field Network Error Indication Parameter Added (Version 1.5 or later)

In Main Software Ver 1.5, the "10: FieldNetwork ErrDisplay" parameter is newly added to the I/O Hardware Settings window (Access: [F4 I/O]—[F6 Aux.]—[F1 Set H/W]). This parameter allows you to choose whether a network error will display "every time" it occurs or at the "first time."

This parameter takes effect in the DeviceNet masters and slaves and the PROFIBUS slaves.

The addition of this parameter disables the "8: DeviceNet Setup ErrDisplay" in the I/O Hardware Settings window.

This parameter is set to "0" (EveryTime) by default for safe operation of the facilities. Every time an I/O operation is carried out, an error will display if any.

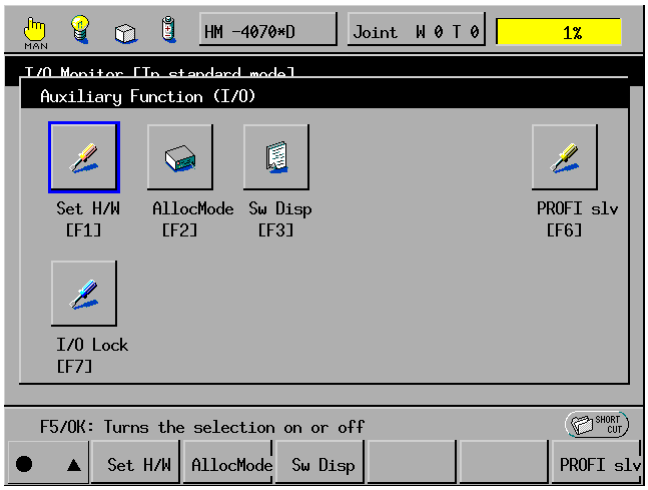
To check program operations using dummy I/Os for adjusting facilities where no connection to the network has been established, set this parameter to "1" (First Time). Doing so will not display errors once detected, allowing you to check program operations.

NOTE: After completion of adjustment, be sure to set this parameter back to "0."

■ Changing the FieldNetwork ErrDisplay parameter

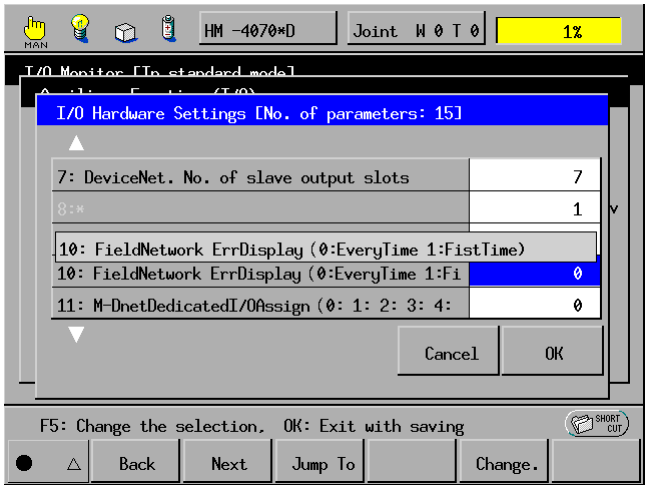
Access: [F4 I/O]—[F6 Aux.]—[F1 Set H/W]

Step 1 Press [F1 Set H/W] in the Auxiliary Function (I/O) window.



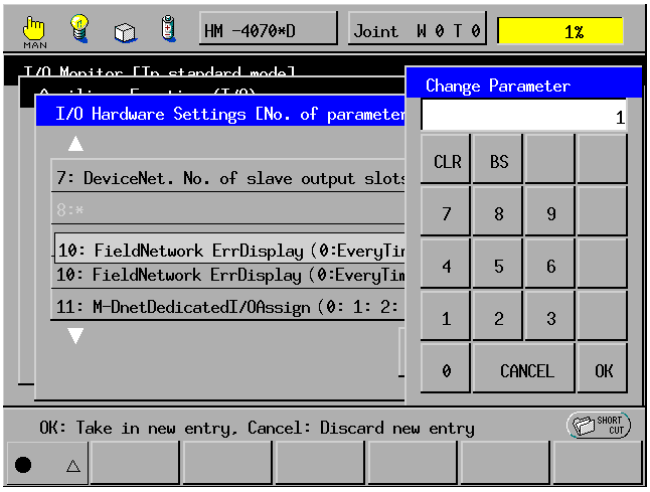
F1

Step 2 Select "10: FieldNetwork ErrDisplay" and press [F5: Change.].

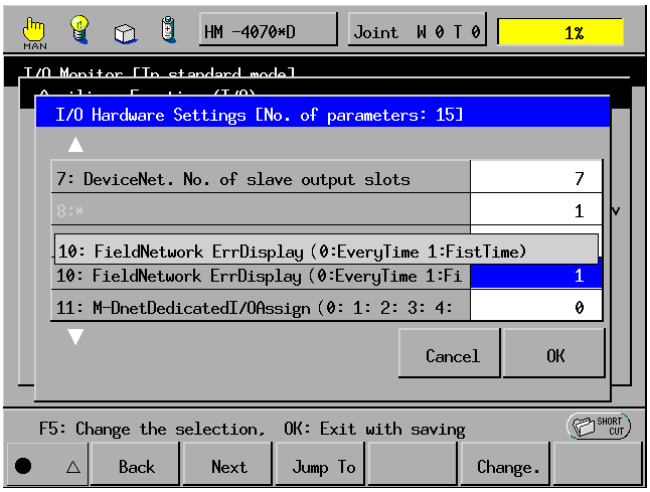


F5

Step 3 Enter "1" in this example and press [OK].



Step 4 Check the newly entered value and press [OK].



Step 5 Following the system message, switch the controller power off and then on.

NOTE: If this message appears, you must switch the controller power off.

4.6 Switching between Standard Mode and Compatible Mode, Modified (Version 1.6 or later)

The switching procedure between the Standard mode and Compatible mode is modified, associated with newly supported optional I/O hardware.

[1] Switching from the Teach Pendant

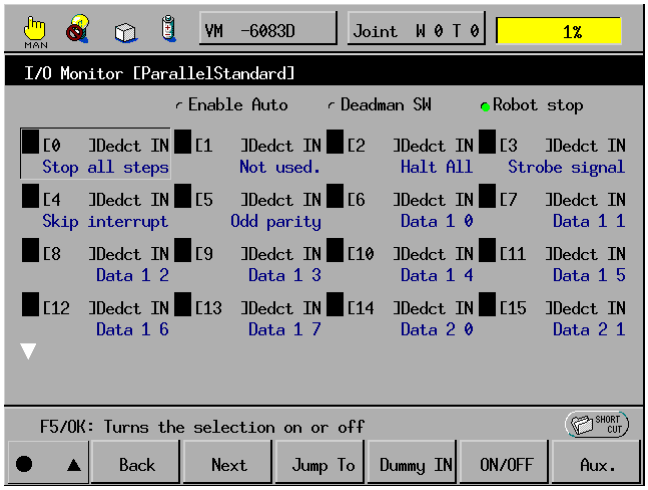
Refer to the INSTALLATION & MAINTENANCE GUIDE, page 5-2.

Follow the procedure below when switching between the Standard and Compatible modes from the teach pendant:

Step 1

Press [F4 I/O] on the top screen.

The I/O Monitor window appears.

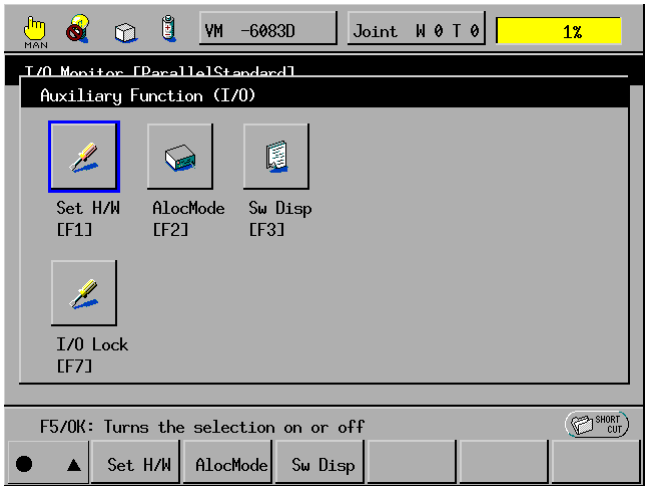


F6

Step 2

Press [F6 Aux.].

The Auxiliary Functions (I/O) window appears.



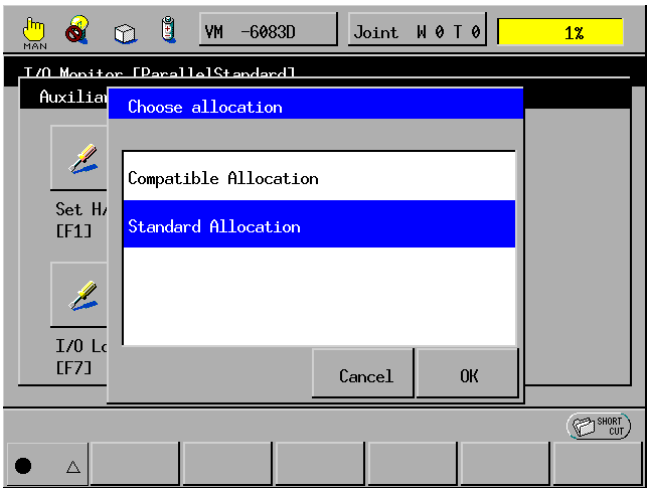
F2

Step 3

Press [F2 AlocMode].

The Choose allocation window appears.

NOTE: If your controller is equipped with a DeviceNet master board, the Dedicated Port Allocation window appears. For details, refer to the User's Manual of the DeviceNet Master Unit, Chapter 3 "I/O Allocation."



Using the jog dial or cursor keys, select the desired allocation mode.

Step 4

Press OK.

The following system message appears, requesting you to restart your controller.

Step 5 Press [OK] in the system message window in Step 4.

The screen returns to the Auxiliary Functions (I/O) window.

Step 6 Turn the controller power OFF.

The screen returns to the Auxiliary Functions (I/O) window.

Step 7 Turn the controller power ON again.

The I/O allocation mode is changed.

[2] Switching from the Computer

Refer to the INSTALLATION & MAINTENANCE GUIDE, page 5-5.

Follow the procedure below when switching from one mode to the other from the computer.

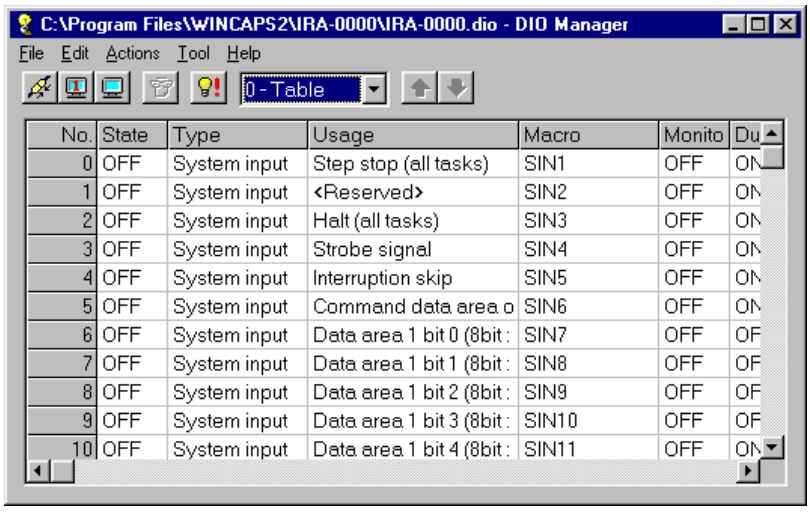
Step 1 Start WINCAPSII on the computer. Log in with Programmer.

Refer to the procedure given in WINCAPSII GUIDE, Chapter 3, Section 3.1.

For details about the user level of Programmer, refer to the WINCAPSII GUIDE, Chapter 1, Section 1.3.

Step 2 Click on the DIO Manager button in System Manager.

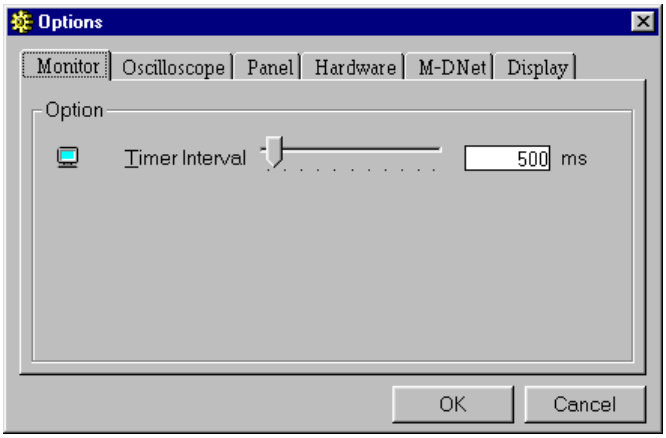
DIO Manager starts to display the DIO Manager window.



No.	State	Type	Usage	Macro	Monito	Du
0	OFF	System input	Step stop (all tasks)	SIN1	OFF	ON
1	OFF	System input	<Reserved>	SIN2	OFF	ON
2	OFF	System input	Halt (all tasks)	SIN3	OFF	ON
3	OFF	System input	Strobe signal	SIN4	OFF	ON
4	OFF	System input	Interruption skip	SIN5	OFF	ON
5	OFF	System input	Command data area o	SIN6	OFF	ON
6	OFF	System input	Data area 1 bit 0 (8bit :	SIN7	OFF	OF
7	OFF	System input	Data area 1 bit 1 (8bit :	SIN8	OFF	OF
8	OFF	System input	Data area 1 bit 2 (8bit :	SIN9	OFF	OF
9	OFF	System input	Data area 1 bit 3 (8bit :	SIN10	OFF	OF
10	OFF	System input	Data area 1 bit 4 (8bit :	SIN11	OFF	ON

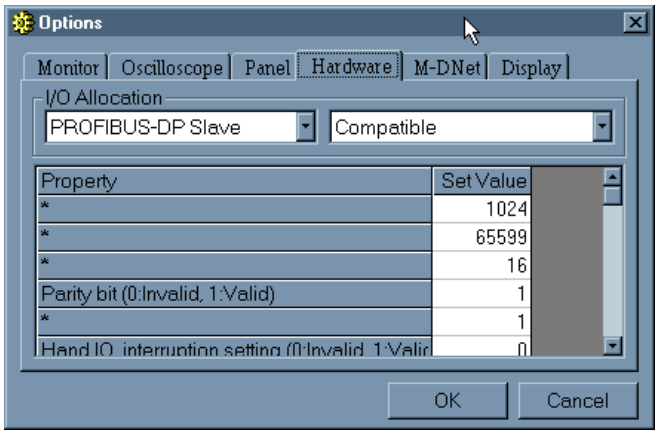
Step 3 From the Tool menu of DIO Manager, select the Options command.

The Options window appears as shown below.

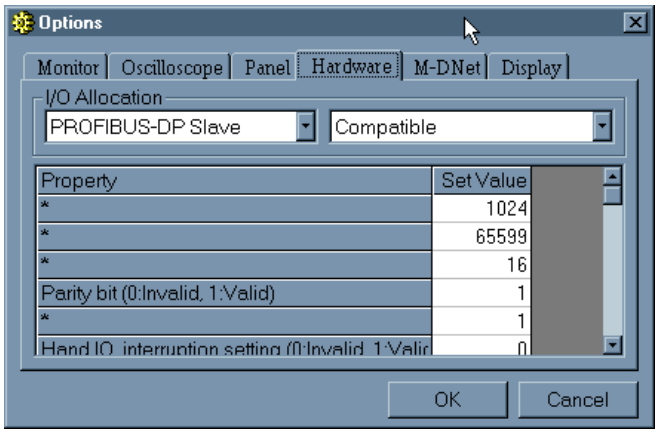


Step 4 Click on the Hardware tab in the Options window.

The hardware settings appear.



Step 5 In the I/O Allocation box, use the right-hand popup menu to choose the desired allocation mode.

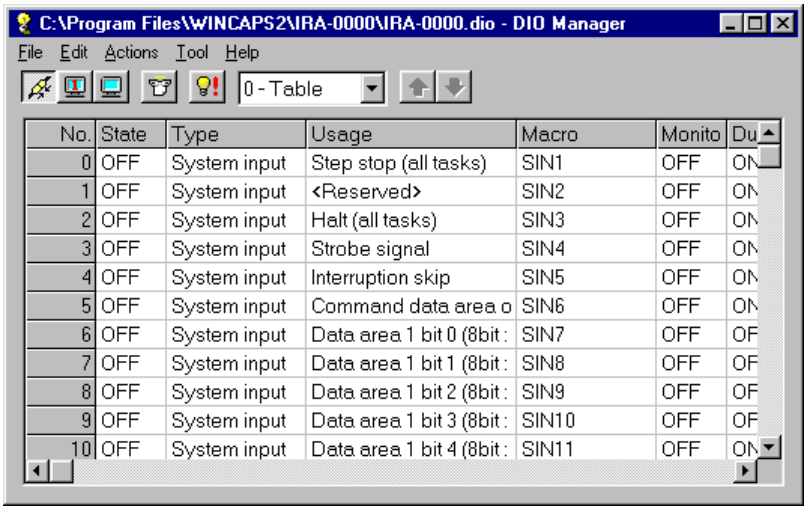


Step 6 Click on OK in the Options window.

The Options window closes.

Step 7 Click on the Connect button to connect the computer to the Robot Controller.

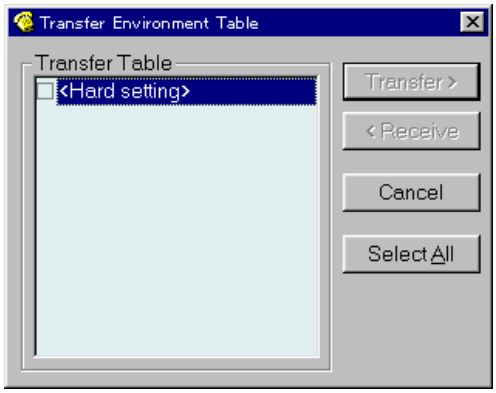
The Connect button seems depressed.



Step 8

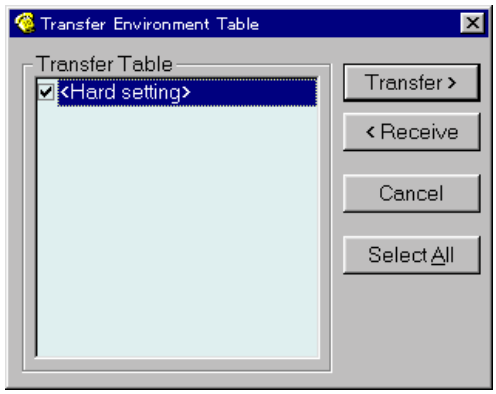
Click on the Transfer button.

The Transfer Environment Table window appears.



Step 9

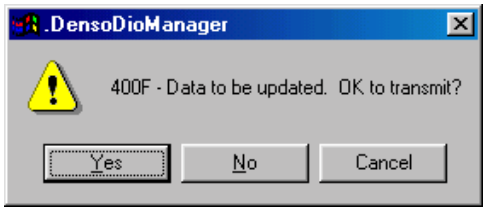
Check off the check box by clicking on the Hard setting field.



Step 10

Click on the Transfer> button.

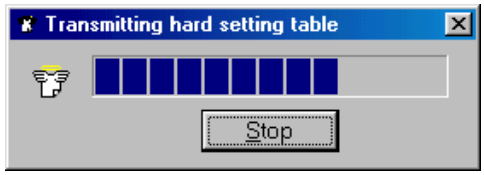
The following message window appears confirming that you are sure to update the data.



Step 11

Click on the Yes button in the message window shown in Step 10.

The Transmitting hard setting table appears displaying a bar graph that indicates the transfer progress.



Step 12

After the Transmitting hard setting table disappears from the screen, turn the controller power switch OFF.

Step 13

Turn the controller power switch ON.

The I/O allocation mode is changed.

4.7 Clearing User Programs and Variables

(Version 1.6 or later)

Version 1.6 or later allows you to delete all user programs stored and clear all global variables to zero.

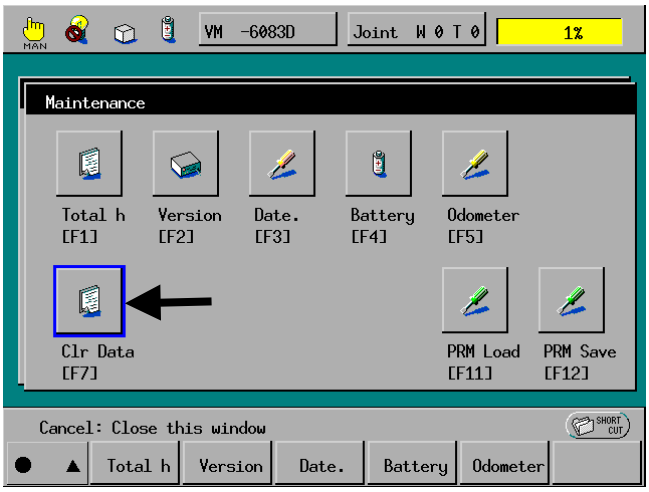
Access: [F6 Set]—[F6 Maint.]—[F7 Clr Data]

Operating procedure

Step 1

On the top screen, press [F6 Set].

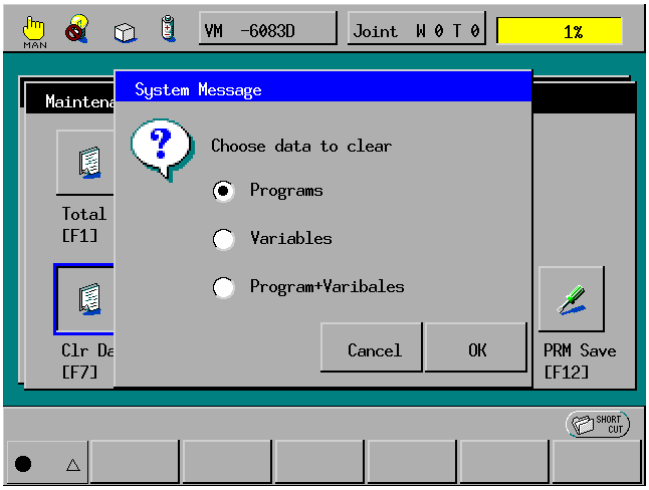
On the Settings (Main) window, press [F6 Maint.] to call up the Maintenance window as shown below.



Press [F7 Clr Data].

Step 2

The choice screen appears where you may choose data type to be cleared.



Choose either one of the following three choices and press OK.

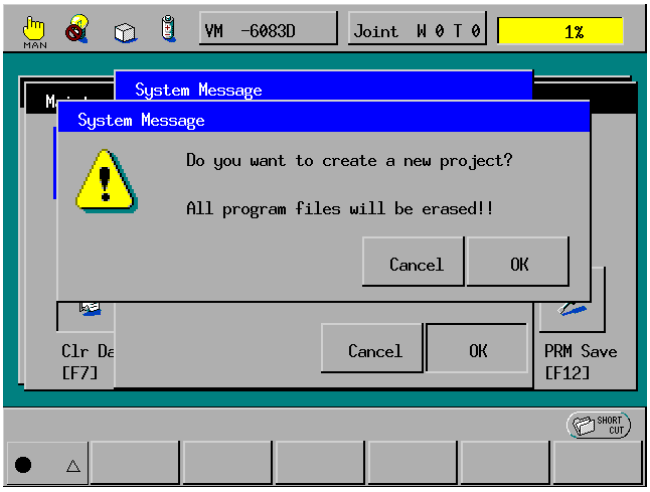
- Programs: Delete all user programs.
- Variables: Clear all global variables to zero.
- Programs + Variables: Delete all user programs and clear all global variables to zero.

Step 3

The following system message appears.

Press OK. Then deleting programs or clearing variables will start.

(Program Deletion Confirmation Message)



(Variable Clearing Confirmation Message)

4.8 Teaching Function Added to the Operating Panel

Refer to the INSTALLATION & MAINTENANCE GUIDE, page 3-1.

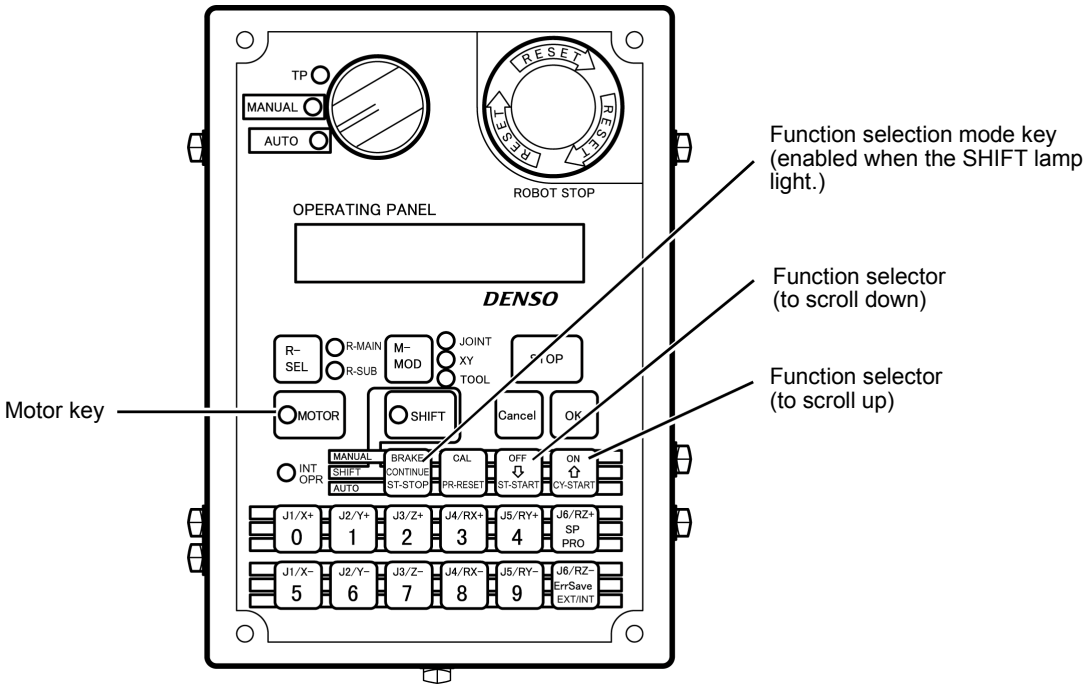
Version 1.6 or later enables the operating panel to support the "Operating the robot arm by specifying a desired variable." This section outlines the teaching functions of the operating panel.

[1] Operating panel functions

With the operating panel, you may run the robot manually, start programs, edit variables, get robot arm positions into variables in teaching, and move the robot arm by specifying a desired variable. Choosing work coordinates or tool coordinates is also possible.

Operating Panel Functions

Version	Function:	Description
Version.1.2 or later	Editing variables	You may edit variables by entering numerical values.
Version.1.4 or later	- Teaching the current position - Choosing work coordinates or tool coordinates	- You may get the current position into P variables, J variable, and T variables. It is used for position teaching. - You may choose work coordinates or tool coordinates.
Version.1.6 or later	Operating the robot arm by specifying a desired variable	You may move the robot arm according to the specified variable. It is used to confirm variables you have preset in teaching.



[2] Operating procedure

According to the procedure below, you may choose the desired function in Manual mode.

Step 1 Turn the mode selector switch to the MANUAL position.

Step 2 Press the SHIFT key.

The SHIFT lamp should come on.

Step 3 Press the function selection mode key to enter the selection mode.

BRAKE
CONTINUE
ST-STOP

The following display appears.

F1:Chg VarVal I

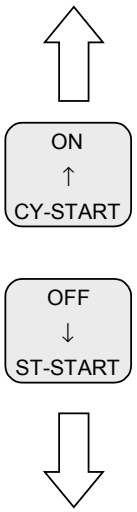
Step 4 Press the following function selectors to scroll the display up or down.

OFF
↓
ST-START

ON
↑
CY-START

F2: Chg VarVal F

Step 5



You may select any of the following functions:

[Chg VarVal I]	Edit integer variables by entering numerical values
[Chg VarVal F]	Edit floating-point variables by entering numerical values
[Chg VarVal D]	Edit double-precision variables by entering numerical values
[Chg VarVal V]	Edit vector variables by entering numerical values
[Chg VarVal P]	Edit position variables by entering numerical values
[Chg VarVal J]	Edit joint variables by entering numerical values
[Chg VarVal T]	Edit variables in homogeneous transform matrix by entering numerical values
[Set VarVal P]	Get the current position into a position variable
[Set VarVal J]	Get the current position into a joint variable
[Set VarVal T]	Get the current position into a variable in homogeneous transform matrix
[Move to Pvar]	Operate the robot by selecting a position variable
[Move to Jvar]	Operate the robot by selecting a joint variable
[Move to Tvar]	Operate the robot by selecting a variable in homogeneous transform matrix

When the desired function is displayed, press the OK key.

To exit from the function selection mode, press the Cancel key.

If any error occurs during the function selection procedure, the operating panel will automatically exit from the function selection mode.

4.9 Wall-Mount Added to the Robot Installation Condition (Version 1.6 or later)

In addition to the floor-mount and overhead-mount types, the wall-mount is newly supported in the robot installation condition.

According to the installation condition, the optimal running conditions will differ.

When shipped, the robot is set to "Floor-mount." You may change the installation condition.

NOTE: Some models are limited in installation condition.

Access: Top Screen—[F2 Arm]—[F6 Aux.]—[F7 Config.]

0: Floor-mount (factory default)

1: Overhead-mount

2: Wall-mount

4.10 Simplified Palletizing

Supplement to the BEGINNER'S GUIDE

Palletizing is explained in the BEGINNER'S GUIDE, Lesson 16. For simpler palletizing, this section provides you with a simplified palletizing template using a palletizing library.

Simplified Palletizing Program "PRO1"

```
'!TITLE "Simplified palletizing program sample"
'
'Approach clearance 50mm, Depart clearance 50mm
'Palletizing target position variable P[40]
'Palletizing counter I[10]
'Stacked-pallets counter I[11]

'N=5 M=3 K=20mm
'            M
'   N   P[54]---------P[55]
'    /             / |
'+  P[52]-------P[53]|
'K  |             | /
'-   -------------/
'
PROGRAM PRO1                                          (1) Program name
  TAKEARM

'------- Get palletizing positions from P[40] -------
'Order of parameters N,M,Stacked pallet height mm,P1,P2,P3,P4,Palletizing points numbers,
'Palletizing counter, Stacked-pallets counter
   CALL xdGetPalt(5,3,20,P[52],P[53],P[54],P[55],P[40],I[10],I[11])   (2) Call library

'---------- Palletizing ----------
   APPROACH P,P[40],@0 50   'Approaching P[40] 50mm upwards           (3) Approaching

   MOVE L,@0 P[40]          'Move down to P[40]                       (4) Down-movement

   DEPART L,50              'Move up by 50mm                          (5) Up-movement

'---------- Count up counters ----------
   I[10] = I[10] + 1        'Increment palletizing counter by one     (6) Count up palletizing counter

   if I[10] > (5*3) then    'If palletizing a layer of pallets (3 rows x 5 columns) finishes   (7) Check completion of palletizing of a layer of pallets

      I[10] = 1             'then reset palletizing counter to initial value   (8) Reset palletizing counter

      I[11] = I[11] + 1     'Increment stacked-pallets counter by one          (9) Count up stacked-pallets counter

      IF  I[11] >= 5 THEN   'If palletizing 5 layers of pallets finishes      (10) Check completion of palletizing of 5 stacks
         I[10] = 1          'then reset stacked-pallets counter to initial value
      END IF                                                                  (11) Reset stacked-pallets counter
   END IF
'----------
END
```

■ Simplified palletizing program "PRO1"

In palletizing explained in the BEGINNER'S GUIDE Lesson 16, you need to execute the pltInitialize library before starting palletizing.

This simplified palletizing program requires no execution of that library. Just executing PRO1 will start palletizing operation.

In simplified palletizing, you need to specify addition and resetting of the palletizing counter and stacked-pallets counter, while in conventional palletizing those counters are automatically controlled inside libraries.

Variables used in PRO1

- Palletizing target position variable (Position variable, P40 in this example)
- Palletizing counter variable (Integer variable, I10 in this example)
- Stacked-pallets counter (Integer variable, I11 in this example)
- Corner partition variables (Position variables, P52 to P55 in this example)

What to do before execution of PRO1

Before start of PRO1, you need to:

- Assign the initial value "1" to each of the palletizing counter I10 and stacked-pallets counter I11 and
- Teach the positions of four corner partitions in the pallet to corner partition variables P1 to P4.

On the following pages are detailed explanation of each part of the program PO1.

(1) Program name

```
'
PROGRAM PRO1
  TAKEARM
```

Change the program name

(2) Call library

```
'------- Get palletizing positions from P[40] --------
'Order of parameters N,M,Stacked pallet height mm,P1,P2,P3,P4,Palletizing points numbers
'Palletizing counter, Stacked-pallets counter
   CALL xdGetPalt (5,3,20,P[52],P[53],P[54],P[55],P[40],I[10],I[11])
```

Setting the following parameters to the called library will assign the target position to the palletizing target position variable specified by the 8th parameter.

1st parameter No. of rows, which should be 1 or greater.
(3 rows in this example)

2nd parameter No. of columns, which should be 1 or greater.
(5 columns in this example)

3rd parameter Height of stacked pallets in mm.
Specify a positive value when increasing the layers of pallets; a negative value when decreasing them.
(20 mm specified in this example)

4th to 7th parameters Position variables to which four corner partition positions of the pallet are assigned.
(P52 to P55 in this example)

8th parameter Palletizing target position variable to which the target position will be assigned. This position may be calculated from the current counter values.
(P40 in this example)

9th parameter Palletizing counter, which should be 1 or greater and M*N or less. According to this value, the corner partition positions may be specified.

10th parameter Stacked-pallets counter, which should be 1 or greater. According to this value, the layer number may be specified.

(3) Approaching

(4) Down-movement

(5) Up-movement

```
'------- Palletizing -------------------------
   APPROACH P,P[40],@0 50     'Approaching P[40] 50mm upwards
   MOVE L,@0 P[40]            'Move down to P[40]
   DEPART L,50                'Move up by 50mm
```

As a result of execution of "(2) Call library," the palletizing target position is assigned to P40. Then some operations should be carried out to P40.

Usually, during those operation, chuck and unchuck processes will be inserted.

(6) Count up palletizing counter

(7) Check completion of palletizing of a layer of pallets

(8) Reset palletizing counter

(9) Count up stacked-pallets counter

(10) Check completion of palletizing of 5 layers of pallets

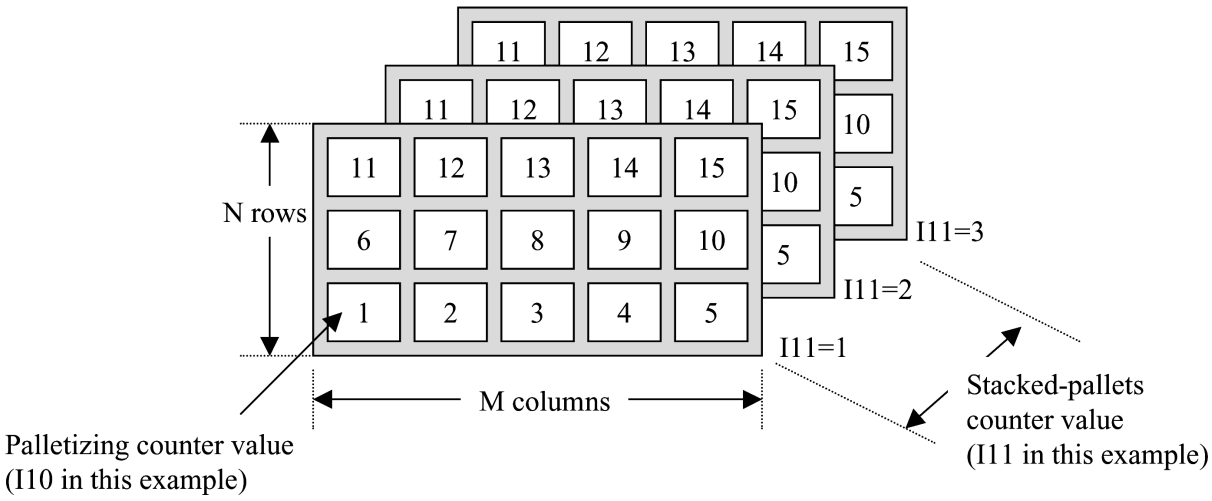
(11) Reset stacked-pallets counter

```
'------- Count up counters-----------------
   I[10] = I[10] + 1            'Increment palletizing counter by one

   if I[10] > (5 * 3) then      'If palletizing a layer of pallets (3 rows x 5 columns) finishes
      I[10] = 1                 'then reset palletizing counter to initial value
      I[11] = I[11] + 1         'Increment stacked-pallets counter by one
      IF  I[11] >= 5 THEN       'If palletizing 5 layers of pallets finishes
          I[10] = 1             'then reset stacked-pallets counter to initial value
      END IF
   END IF
```

This part of the PRO1 counts up the palletizing counter and stacked-pallets counter and checks the completion of palletizing operation for a layer of pallets.

Unlike usual palletizing programs, the simplified palletizing program uses integer variables (I10 and I11 in this example) as a palletizing counter and stacked-pallets counter.

According to the values assigned to I10 and I11, the "(2) Call library" calculates the palletizing target position and assigns it to P40.

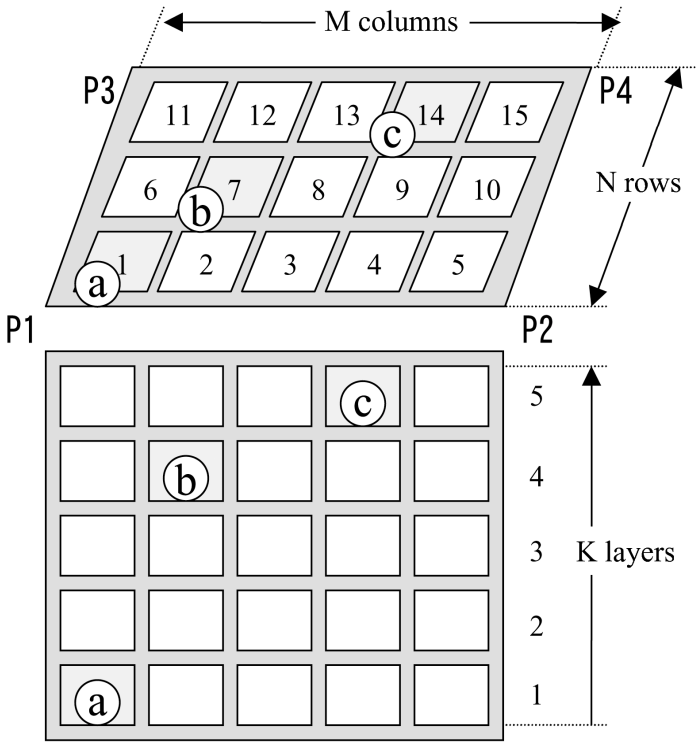


For a single layer of pallet, you may simplify the program further as shown below.

```
'------- Count up counters-----------------
  I[10] = I[10] + 1            'Increment palletizing counter by one

  if I[10] > (5 * 3) then      'If palletizing a layer of pallets (3 rows x 5 columns) finishes
    I[10] = 1                  'then reset palletizing counter to initial value
    I[11] = I[11] + 1          'Increment stacked-pallets counter by one
    IF  I[11] >= 5 THEN        'If palletizing 5 layers of pallets finishes
      I[10] = 1                'then reset stacked-pallets counter to initial value
    END IF
  END IF
```

Delete these lines for a single layer of pallet.

■ Relationship between the palletizing positions and counter values in the simplified palletizing program



If each pallet consists of 3 rows x 5 columns (N=3, M=5), palletizing counter is I10 and stacked-pallets counter is I11, then

Position ⓐ: I10=1, I11=1

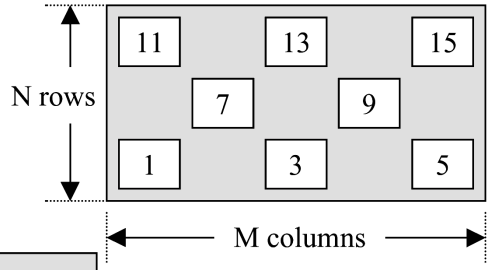
Position ⓑ: I10=7, I11=4

Position ⓒ: I10=14, I11=5

■ Applications of the simplified palletizing program --- Special-purpose palletizing examples ---

(1) Alternate checker-pattern palletizing

Alternate checker-pattern palletizing refers to palletizing to every other partitions as illustrated below. Programming for this is very easy.

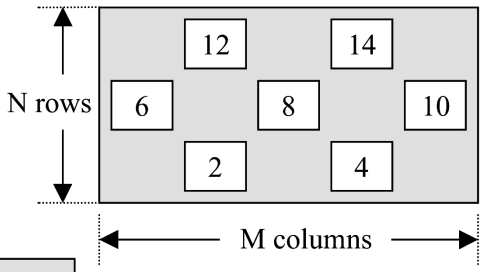


(6) Count up palletizing counter

```
‘---------- Count up counters ----------
  I[10] = I[10] + 1          ‘Increment palletizing counter by one
```

```
‘--------- Count up counters ----------
  I[10] = I[10] + 2          ‘Increment palletizing counter by 2
```

You need to assign "1" to palletizing counter I[10] with the teach pendant beforehand.



(6) Count up palletizing counter

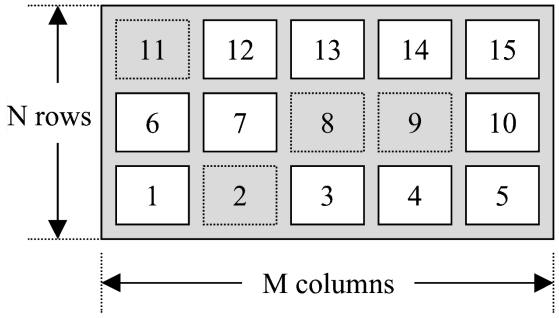
```
‘------ Count up counters ----------
    I[10] = I[10] + 1          ‘Increment palletizing counter by one
 if I[10] > (5 * 3) then       ‘If palletizing a layer of pallets (3 rows x 5 columns) finishes
    I[10] = 1                  ‘then reset palletizing counter to initial value
```

```
‘------Count up counters -----------
    I[10] = I[10] + 2          ‘Increment palletizing counter by 2
 if I[10] > (5 * 3) then       ‘If palletizing a layer of pallets (3 rows x 5 columns) finishes
    I[10] = 2                  ‘then reset palletizing counter to second value
```

You need to assign "2" to palletizing counter I[10] with the teach pendant beforehand.

(2) Skipped palletizing

Skipped palletizing skips arbitrary partitions in palletizing.



No palletizing to 2nd, 8th, 9th or 11th partition.

The above palletizing operation seems complicated, but you may easily program such palletizing just by changing the palletizing counter value that will pass to the library.

```
PROGRAM PRO1
 TAKEARM

'------- Get palletizing positions from P[40] ------
'Order of parameters N,M,Stacked pallet height mm,P1,P2,P3,P4,Palletizing points numbers
'Palletizing counter, Stacked-pallets counter
   CALL xdGetPalt(5,3,20,P[52],P[53],P[54],P[55],P[40],I[10],I[11])
```

```
PROGRAM PRO1
 TAKEARM
 SELECT CASE I[10]
  CASE 2          'If palletizing counter I[10]=2
   I[10] = 3      'then set the counter to 3

  CASE 8,9        'If palletizing counter I[10]=8 or 9
   I[10] = 10     'then set the counter to 10

  CASE 11         'If palletizing counter I[10]=11
   I[10] = 12     'then set the counter to
 END SELECT

'------- Get palletizing positions from P[40] ------
'Order of parameters N,M,Stacked pallet height mm,P1,P2,P3,P4,Palletizing points numbers
'Palletizing counter, Stacked-pallets counter
   CALL xdGetPalt(5,3,20,P[52],P[53],P[54],P[55],P[40],I[10],I[11])
```

4.11 Error Codes Added or Modified

The table below lists error codes newly added or modified in Main Software versions 1.5 and 1.6.

Code	Message	Level	What happens:	What to do:
120A	DeviceNet slave exclusive flag failure	4	The exclusive flag of DeviceNet slave communication processor is not working normally.	Check the connection of the DeviceNet slave board. If the error persists, the board may be defective.
120B	DeviceNet master exclusive flag failure	4	The exclusive flag of DeviceNet master communication processor is not working normally.	Check the connection of the DeviceNet master board. If the error persists, the board may be defective.
1210	DeviceNet internal communications error	4	Communications data is abnormal due to noises.	Restart the controller and then retry the operation.
1220	I/O option board connection error	4	More than one I/O option board is installed. Concurrent access to those boards is impossible.	Turn the controller power off and remove I/O option boards except one board.
126A	CIF board access error	5	Failed to access the CIF board.	Restart the controller.
126B	CIF board DPRAM access error	5	Failed to access the DPRAM on the CIF board.	Restart the controller. If the error persists, the CIF board may be defective.
126C	CIF board initialization error	5	Failed to initialize the CIF board.	↑
126D	CIF board watchdog error	5	An error has occurred in the watchdog timer on the CIF board.	↑
126E	Failed CIF board resetting	5	Failed to reset the CIF board.	↑
126F	Network is not established	4	On the CIF board, no network (e.g., PROFIBUS) has been established.	(1) Check the connection of the line. (2) Check the network settings (e.g., node address). (3) Check lines for breaks.
127A	CIF board failure	5	A CIF board error has occurred.	Restart the controller. If the error persists, the CIF board may be defective.
127B	CIF board message send time out	5	During message transmission from the CIF board, a timeout has occurred.	↑

Code	Message	Level	What happens:	What to do:
127C	CIF board message received time out	5	During message reception on the CIF board, a timeout has occurred.	↑
127D	Communication watchdog is invalid	4	The communications watchdog timer is set to "Disable."	Set the watchdog timer to "Enable"; otherwise, the system cannot check whether the network is established.
127E	Network configuration mismatch	4	The configuration data sent from the master unit mismatches that preset in the slave unit.	Check the network settings (e.g., module type).
127F	CIF board Initializing	4	You attempted to perform any operation that is not allowed during initialization of the CIF board.	Wait for approx. 20 seconds and then retry the operation.
20F1	Semaphore creation error	5	An error has occurred when the controller attempted to create the semaphore.	Restart the controller and then retry the operation.
20F2	Semaphore taking error	5	An error has occurred when the controller attempted to get semaphore.	Restart the controller and then retry the operation.
27AC	Sending system data	2	The SYSTEM data is being sent.	Wait until the transmission of the SYSTEM data is completed.
27BF	Updating system	2	Updating the system.	Wait until updating is completed. Do not turn the controller power off during updating.
27D0	Semaphore release error	5	An error has occurred when the controller attempted to release the semaphore.	Turn the controller power off and then on. Retry the operation.
27D3	Cannot take J1 semaphore	4	(1) A task, that had gotten no arm group containing J1, attempted to execute the J1 motion related command. (2) A task attempted to get an arm group containing J1, but any other task had already gotten it.	(1) Get an arm group containing J1 with the TAKEARM statement, and then execute J1 motion related command. (2) Correct the program so that more than one program will not attempt to get an arm group containing J1 at the same time.

Code	Message	Level	What happens:	What to do:
27D4	Cannot take J2 semaphore	4	(1) A task, that had gotten no arm group containing J2, attempted to execute the J2 motion related command. (2) A task attempted to get an arm group containing J2, but any other task had already gotten it.	(1) Get an arm group containing J2 with the TAKEARM statement, and then execute J2 motion related command. (2) Correct the program so that more than one program will not attempt to get an arm group containing J2 at the same time.
27D5	Cannot take J3 semaphore	4	(1) A task, that had gotten no arm group containing J3, attempted to execute the J3 motion related command. (2) A task attempted to get an arm group containing J3, but any other task had already gotten it.	(1) Get an arm group containing J3 with the TAKEARM statement, and then execute J3 motion related command. (2) Correct the program so that more than one program will not attempt to get an arm group containing J3 at the same time.
27D6	Cannot take J4 semaphore	4	(1) A task, that had gotten no arm group containing J4, attempted to execute the J4 motion related command. (2) A task attempted to get an arm group containing J4, but any other task had already gotten it.	(1) Get an arm group containing J4 with the TAKEARM statement, and then execute J4 motion related command. (2) Correct the program so that more than one program will not attempt to get an arm group containing J4 at the same time.
27D7	Cannot take J5 semaphore	4	(1) A task, that had gotten no arm group containing J5, attempted to execute the J5 motion related command. (2) A task attempted to get an arm group containing J5, but any other task had already gotten it.	(1) Get an arm group containing J5 with the TAKEARM statement, and then execute J5 motion related command. (2) Correct the program so that more than one program will not attempt to get an arm group containing J5 at the same time.
27D8	Cannot take J6 semaphore	4	(1) A task, that had gotten no arm group containing J6, attempted to execute the J6 motion related command. (2) A task attempted to get an arm group containing J6, but any other task had already gotten it.	(1) Get an arm group containing J6 with the TAKEARM statement, and then execute J6 motion related command. (2) Correct the program so that more than one program will not attempt to get an arm group containing J6 at the same time.

Code	Message	Level	What happens:	What to do:
27D9	Cannot take J7 semaphore	4	(1) A task, that had gotten no arm group containing J7, attempted to execute the J7 motion related command. (2) A task attempted to get an arm group containing J7, but any other task had already gotten it.	(1) Get an arm group containing J7 with the TAKEARM statement, and then execute J7 motion related command. (2) Correct the program so that more than one program will not attempt to get an arm group containing J7 at the same time.
27DA	Cannot take J8 semaphore	4	(1) A task, that had gotten no arm group containing J8, attempted to execute the J8 motion related command. (2) A task attempted to get an arm group containing J8, but any other task had already gotten it.	(1) Get an arm group containing J8 with the TAKEARM statement, and then execute J8 motion related command. (2) Correct the program so that more than one program will not attempt to get an arm group containing J8 at the same time.
27E3	Not executable while joint is suspended	2	If suspended, the joint cannot be moved by variables.	Release the suspension of the joint.
2AF1	Encoder reference position error	3	When the controller is turned on, any encoder value is different from that detected when the controller was off. This error may be caused by either of the following: - Encoder(s) defective - The encoder(s) is normal, but the robot arm has been moved after the controller was turned off.	Check the reference position (referring to the "Encoder Reference Position and Its Related Error") or carry out CALSET for all axes.
2AF2	During a software limit check release	2	When the software motion limits are temporarily invalid, you attempted to move joints using variables or get the current arm position into variables.	Make the software motion limits valid and then retry the operation.
53E3	Insufficient memory	4	The amount of memory is insufficient to display the screen you requested.	Turn the controller power off and then on. Retry the operation. If this error occurs during program editing, removing a part of that program may fix this problem.
5797	J7 speed over in Direct Teaching	4	Excessive speed (axis 7) in manual operation occurred during teaching in direct mode.	Same as that for error code 5796.

Code	Message	Level	What happens:	What to do:
5798	J8 speed over in Direct Teaching	4	Excessive speed (axis 8) in manual operation occurred during teaching in direct mode.	↑
6077	Cur. Position J7 soft motion limit over	3	(1) Current position exceeded the software motion limit of the 7th axis in motion (2) The 7th axis exceeded the software motion limit during CP motion for deceleration stop.	Same as that for error code 6076.
6078	Cur. Position J8 soft motion limit over	3	(1) The current position exceeded the software motion limit of the 8th axis in motion. (2) The 8th axis exceeded the software motion limit during CP motion at the deceleration stop.	↑
6087	J7 command speed limit over	3	The CP motion cannot execute with the specified speed because the speed command value of the 7th axis exceeds the limit value.	Same as that for error code 6086.
6088	J8 command speed limit over	3	The CP motion cannot execute with the specified speed because the speed command value of the 8th axis exceeds the limit value.	↑
608F	J7 command speed limit over (servo)	4	The CP motion cannot execute with the specified speed because the speed command value of the 7th axis exceeds the limit value.	Same as that for error code 608E.
6090	J8 command speed limit over (servo)	4	The CP motion cannot execute with the specified speed because the speed command value of the 8th axis exceeds the limit value.	↑
6097	J7 power module failure	5	A power module failure occurred on 7th axis.	Same as that for error code 6096.
6098	J8 power module failure	5	A power module failure occurred on 8th axis.	↑

Code	Message	Level	What happens:	What to do:
60B7	J7 current offset failure	5	Offset of the 7th axis detection current exceeds the reference value.	Inspect or repair the 7th-axis power module.
60B8	J8 current offset failure	5	Offset of the 8th axis detection current exceeds the reference value.	Inspect or repair the 8th-axis power module.
6101	Watchdog error	5	(1) +24V output short-circuited (2) 200 VAC error (3) Low power voltage in the controller (4) Servomotor counter emf error (5) Power board error If this error message appears when you turn the controller power off, it means no error.	(1) Inspect the wiring ends of the controller I/O cables to check that the +24V and 0V pins are not short-circuited or the +24V and output pins are not short-circuited. (2) Check that the 200 VAC source is within the range between 253 and 170 VAC. (3) Check that 200 VAC power cables are connected firmly. Be sure to turn the controller power off when connecting/ disconnecting those cables. (4) Check that the specifications of the hands (incl. works) are compliant with the robot requirements. (5) If the error persists after the controller power is turned off and on, it is necessary to inspect or repair the controller.
6102	Power failure	5	↑	↑
6117	J7 excess error	4	Excessive deviation error on the 7th axis. The servo deviation exceeded the permissible value.	Same as that for error code 6116.
6118	J8 excess error	4	Excessive deviation error on the 8th axis. The servo deviation exceeded the permissible value.	↑
6127	J7 overcurrent	4	Overcurrent error on the 7th axis. The current to the motor exceeded the permissible value.	Same as that for error code 6126.
6128	J8 overcurrent	4	Overcurrent error on the 8th axis. The current to the motor exceeded the permissible value.	↑

Code	Message	Level	What happens:	What to do:
612F	J7 overcurrent (software)	4	Overcurrent error on the 7th axis. The current flown through the motor exceeded the permissible value set in the software.	Same as that for error code 612E.
6130	J8 overcurrent (software)	4	Overcurrent error on the 8th axis. The current flown through the motor exceeded the permissible value set in the software.	↑
6137	J7 encoder cable disconnected	4	The 7th-axis cable is not connected or broken.	Same as that for error code 6136.
6138	J8 encoder cable disconnected	4	The 8th-axis cable is not connected or broken.	↑
6147	J7 power module failure	4	A fuse on the 7th-axis power module was blown.	Same as that for error code 6146.
6148	J8 power module failure	4	A fuse on the 8th-axis power module was blown.	↑
6177	J7 motor overload	4	An overload error occurred on the 7th axis motor.	Same as that for error code 6176.
6178	J8 motor overload	4	An overload error occurred on the 8th axis motor.	↑
6180	Servo communication data error	5	A controller internal error has occurred. (Servo-received data is out of the range.)	(1) Check that the FG wires of the robot unit and controller are grounded properly. (2) Check that no noise sources (e.g., welding machines) are in the vicinity of the robot unit or controller.
61A7	J7 torque limit time over	4	The 7th axis torque command reached its limit and the time exceeded its limit while maintaining that status.	Same as that for error code 61A6.
61A8	J8 torque limit time over	4	The 8th axis torque command reached its limit and the time exceeded its limit while maintaining that status.	↑
61AF	J7 motor lock overload	4	An overload occurred because the 7th axis motor locked.	Same as that for error code 61AE.
61B0	J8 motor lock overload	4	An overload occurred because the 8th axis motor locked.	↑

Code	Message	Level	What happens:	What to do:
61B7	J7 power module overload	4	An overload error occurred in the 7th axis power module of the controller.	Same as that for error code B6.
61B8	J8 power module overload	4	An overload error occurred in the 8th axis power module of the controller.	↑
61EC	Press the deadman switch	3	The deadman switch was turned off during auto gain tuning.	Hold down the deadman switch during auto gain tuning.
6407	J7 encoder acceleration error	5	The 7th-axis encoder exceeded the acceleration limit value.	Same as that for error code 6406.
6408	J8 encoder acceleration error	5	The 8th-axis encoder exceeded the acceleration limit value.	↑
6411	J1 encoder system down failure	5	The 1st-axis encoder system has gone down.	1. Check that the encoder backup battery connector is firmly plugged. 2. If the battery connector has been disconnected for more than 3 minutes, this error will occur. For recovery, reset the encoder and execute CALSET.
6412	J2 encoder system down failure	5	The 2nd-axis encoder system has gone down.	↑
6413	J3 encoder system down failure	5	The 3rd-axis encoder system has gone down.	↑
6414	J4 encoder system down failure	5	The 4th-axis encoder system has gone down.	↑
6415	J5 encoder system down failure	5	The 5th-axis encoder system has gone down.	↑
6416	J6 encoder system down failure	5	The 6th-axis encoder system has gone down.	↑
6417	J7 encoder system down failure	5	The 7th-axis encoder system has gone down.	↑

Code	Message	Level	What happens:	What to do:
6418	J8 encoder system down failure	5	The 8th-axis encoder system has gone down.	↑
6427	J7 encoder data not received	4	A communication error has occurred on the 7th-axis encoder.	Same as that for error code 6426.
6428	J8 encoder data not received	4	A communication error has occurred on the 8th-axis encoder.	↑
6437	J7 encoder counter overflow	5	The multi-rotation data of the 7th-axis encoder has overflowed.	Reset the 7th-axis encoder and execute CALSET.
6438	J8 encoder counter overflow	5	The multi-rotation data of the 8th-axis encoder has overflowed.	Reset the 8th-axis encoder and execute CALSET.
6447	J7 encoder counter error	5	A 7th-axis encoder counter error has occurred.	Same as that for error code 6446.
6448	J8 encoder counter error	5	A 8th-axis encoder counter error has occurred.	↑
6457	J7 encoder G/A counter error	5	↑	Same as that for error code 6456.
6458	J8 encoder G/A counter error	5	↑	↑
6467	J7 encoder phase Rx signal interrupted	4	The 7th-axis cable is not connected or broken.	Same as that for error code 6466.
6468	J8 encoder phase Rx signal interrupted	4	The 8th-axis cable is not connected or broken.	↑
6477	J7 CALSET execution failed	2	CALSET execution error on the 7th axis.	Same as that for error code 6476.
6478	J8 CALSET execution failed	2	CALSET execution error on the 8th axis.	↑
64A7	J7 encoder low battery	2	The 7th axis encoder back-up battery voltage dropped.	Same as that for error code 64A6.
64A8	J8 encoder low battery	2	The 8th axis encoder back-up battery voltage dropped.	↑

Code	Message	Level	What happens:	What to do:
64AF	J7 encoder preset status error	4	The robot controller could not recognize the current position of the 7th-axis encoder.	Restart the robot controller. If Error 6427 also occurs, take its recovery. If not, replace the motor.
64B0	J8 encoder preset status error	4	The robot controller could not recognize the current position of the 8th-axis encoder.	Restart the robot controller. If Error 6428 also occurs, take its recovery. If not, replace the motor.
64B7	J7 encoder CRC check error	4	CRC check failure occurred on the 7th axis encoder data.	Same as that for error code 64B6.
64B8	J8 encoder CRC check error	4	CRC check failure occurred on the 8th axis encoder data.	↑
64C7	J7 encoder framing error	4	Frame configuration error in the 7th axis encoder data.	Same as that for error code 64C6.
64C8	J8 encoder framing error	4	Frame configuration error in the 8th axis encoder data.	↑
64D7	J7 encoder data (software) abnormal	4	Encoder data error (data skip) on 7th axis.	Same as that for error code 64D6.
64D8	J8 encoder data (software) abnormal	4	Encoder data error (data skip) on 8th axis.	↑
64E7	J7 encoder phase Rx not received (CABS)	4	7th axis encoder communication error occurred.	Same as that for error code 64E6.
64E8	J8 encoder phase Rx not received (CABS)	4	8th axis encoder communication error occurred.	↑
64F7	J7 encoder CRC check error (CABS)	4	CRC error on 7th axis encoder data occurred.	Same as that for error code 64F6.
64F8	J8 encoder CRC check error (CABS)	4	CRC error on 8th axis encoder data occurred.	↑
6637	J7 speed limit over	4	Motion exceeded the 7th axis speed limit	Same as that for error code 6636.
6638	J8 speed limit over	4	Motion exceeded the 8th axis speed limit	↑

Code	Message	Level	What happens:	What to do:
6647	J7 acceleration limit over	4	Motion exceeded the 7th axis acceleration limit.	Same as that for error code 6646.
6648	J8 acceleration limit over	4	Motion exceeded the 8th axis acceleration limit.	↑
6677	Dest. pos. out of J7 soft motion limit	3	(1) The motion destination position of the 7th axis is out of the software motion limit. (2) The robot cannot physically perform linear movements in the CP motion for the coordinate entered.	Same as that for error code 6676.
6678	Dest. pos. out of J8 soft motion limit	3	(1) The motion destination position of the 8th axis is out of the software motion limit. (2) The robot cannot physically perform linear movements in the CP motion for the coordinate entered.	↑
66D7	J7 software limit over (Compliance)	3	The 7th-axis software limit was exceeded during compliance control.	Same as that for error code 66D6.
66D8	J8 software limit over (Compliance)	3	The 8th-axis software limit was exceeded during compliance control.	↑
6727	J7 acceleration limit over	4	Operation was executed exceeding the 7th axis acceleration limit.	Same as that for error code 6726.
6728	J8 acceleration limit over	4	Operation was executed exceeding the 8th axis acceleration limit.	↑
6737	J7 acceleration limit over	4	Operation was executed exceeding the 7th axis acceleration limit.	Same as that for error code 6736.
6738	J8 acceleration limit over	4	Operation was executed exceeding the 8th axis acceleration limit.	↑
6747	J7 acceleration limit over	4	Operation was executed exceeding the 7th axis acceleration limit.	Same as that for error code 6746.

Code	Message	Level	What happens:	What to do:
6748	J8 acceleration limit over	4	Operation was executed exceeding the 8th axis acceleration limit.	↑
6757	Execute J7 CALSET	2	CALSET has not been executed on the 7th axis.	Same as that for error code 6756.
6758	Execute J8 CALSET	2	CALSET has not been executed on the 8th axis.	↑
6767	J7 command accel limit over (servo)	4	The CP motion is not available with the specified speed because the acceleration command value exceeds the limit on the 7th axis.	Same as that for error code 6766.
6768	J8 command accel limit over (servo)	4	The CP motion is not available with the specified speed because the acceleration command value exceeds the limit on the 8th axis.	↑
676F	J7 command accel limit over (host)	4	The CP motion is not available with the specified speed because acceleration command value exceeds the limit on the 7th axis.	Same as that for error code 676E.
6770	J8 command accel limit over (host)	4	The CP motion is not available with the specified speed because acceleration command value exceeds the limit on the 8th axis.	↑
6777	J7 encoder speed over	5	When the controller power was OFF, an encoder speed error occurred on the 7th axis.	Same as that for error code 6776.
6778	J8 encoder speed over	5	When the controller power was OFF, an encoder speed error occurred on the 8th axis.	↑
6787	J7 speed over at brake releasing	2	When the brake is OFF, an encoder speed error occurred on the 7th axis.	Same as that for error code 6786.
6788	J8 speed over at brake releasing	2	When the brake is OFF, an encoder speed error occurred on the 8th axis.	↑

Code	Message	Level	What happens:	What to do:
67E7	Boundless rotatory joint isn't available	3	This command or operation cannot execute to a boundless rotatory joint.	Release the joint from the boundless rotation or do not execute this command or operation to the boundless rotatory joint.
67E8	Not set boundless rotatory joint	3	This command or operation cannot execute to a limited rotatory joint.	Set the joint for boundless rotation or do not execute this command or operation to the limited rotatory joint.
6809	Auto gain tuning is interrupted	3	The auto gain tuning process has been interrupted.	Execute auto gain tuning again.
680A	Inertia identification error	3	The inertia identification process is not possible during auto gain tuning	Auto gain tuning is not possible. Implement manual gain tuning.
680B	Auto gain tuning warning 1	1	Overshoot was found at the end of motion during fine adjustment of gain.	To reduce the overshoot, implement manual gain tuning.
680C	Auto gain tuning warning 2	1	Slow settlement was found at the end of motion during fine adjustment of gain.	To reduce the settlement time, implement manual gain tuning.
680D	Auto gain tuning warning 3	1	Low-level oscillation was found during fine adjustment of gain.	To reduce the oscillation, implement manual gain tuning.
680E	Servo data monitor error	4	Failed to monitor servo single-joint.	Clear the error and start monitoring the servo single-joint again.
680F	Auto gain tuning is not executable	3	The auto gain tuning start requirements are not satisfied.	Check the auto gain tuning start requirements and implement auto gain tuning again.
6A91	J1 encoder communication error (bit)	4	The J1 encoder data received is abnormal.	(1) Check that the FG wires of the robot unit and controller are grounded properly. (2) Check that no noise sources (e.g., welding machines) are in the vicinity of the robot unit or controller.
6A92	J2 encoder communication error (bit)	4	The J2 encoder data received is abnormal.	↑
6A93	J3 encoder communication error (bit)	4	The J3 encoder data received is abnormal.	↑
6A94	J4 encoder communication error (bit)	4	The J4 encoder data received is abnormal.	↑

Code	Message	Level	What happens:	What to do:
6A95	J5 encoder communication error (bit)	4	The J5 encoder data received is abnormal.	↑
6A96	J6 encoder communication error (bit)	4	The J6 encoder data received is abnormal.	↑
6A97	J7 encoder communication error (bit)	4	The J7 encoder data received is abnormal.	↑
6A98	J8 encoder communication error (bit)	4	The J8 encoder data received is abnormal.	↑
6AA1	J1 encoder backup error	5	The J1 backup battery does not work so that the internal data has been lost.	(1) Check that the encoder backup battery connector is firmly plugged in. (2) To recover from this error state, you need to reset the related encoder and perform CALSET.
6AA2	J2 encoder backup error	5	The J2 backup has run out so that the internal data has been lost.	↑
6AA3	J3 encoder backup error	5	The J3 backup battery has run out so that the internal data has been lost.	↑
6AA4	J4 encoder backup error	5	The J4 backup battery has run out so that the internal data has been lost.	↑
6AA5	J5 encoder backup error	5	The J5 backup battery has run out so that the internal data has been lost.	↑
6AA6	J6 encoder backup error	5	The J6 backup battery has run out so that the internal data has been lost.	↑
6AA7	J7 encoder backup error	5	The J7 backup battery has run out so that the internal data has been lost.	↑
6AA8	J8 encoder backup error	5	The J8 backup battery has run out so that the internal data has been lost.	↑
6AA9	J1 encoder initialize error	4	An error has occurred during initialization of the J1 encoder.	(1) Check that the FG wires of the robot unit and controller are grounded properly. (2) Check that no noise sources (e.g., welding machines) are in the vicinity of the robot unit or controller.

Code	Message	Level	What happens:	What to do:
6AAA	J2 encoder initialize error	4	An error has occurred during initialization of the J2 encoder.	↑
6AAB	J3 encoder initialize error	4	An error has occurred during initialization of the J3 encoder.	↑
6AAC	J4 encoder initialize error	4	An error has occurred during initialization of the J4 encoder.	↑
6AAD	J5 encoder initialize error	4	An error has occurred during initialization of the J5 encoder.	↑
6AAE	J6 encoder initialize error	4	An error has occurred during initialization of the J6 encoder.	↑
6AAF	J7 encoder initialize error	4	An error has occurred during initialization of the J7 encoder.	↑
6AB0	J8 encoder initialize error	4	An error has occurred during initialization of the J8 encoder.	↑
6AB1	J1 encoder absolute data error	5	The J1 position data may be wrong.	↑
6AB2	J2 encoder absolute data error	5	The J2 position data may be wrong.	↑
6AB3	J3 encoder absolute data error	5	The J3 position data may be wrong.	↑
6AB4	J4 encoder absolute data error	5	The J4 position data may be wrong.	↑
6AB5	J5 encoder absolute data error	5	The J5 position data may be wrong.	↑
6AB6	J6 encoder absolute data error	5	The J6 position data may be wrong.	↑
6AB7	J7 encoder absolute data error	5	The J7 position data may be wrong.	↑
6AB8	J8 encoder absolute data error	5	The J8 position data may be wrong.	↑

Code	Message	Level	What happens:	What to do:
6AB9	J1 encoder error	5	A J1 encoder error has occurred.	(1) Check that the FG wires of the robot unit and controller are grounded properly. (2) Check that no noise sources (e.g., welding machines) are in the vicinity of the robot unit or controller. (3) To recover from this error state, you need to reset the related encoder and perform CALSET.
6ABA	J2 encoder error	5	A J2 encoder error has occurred.	↑
6ABB	J3 encoder error	5	A J3 encoder error has occurred.	↑
6ABC	J4 encoder error	5	A J4 encoder error has occurred.	↑
6ABD	J5 encoder error	5	A J5 encoder error has occurred.	↑
6ABE	J6 encoder error	5	A J6 encoder error has occurred.	↑
6ABF	J7 encoder error	5	A J7 encoder error has occurred.	↑
6AC0	J8 encoder error	5	A J8 encoder error has occurred.	↑
6AC1	J1 encoder over speed error	5	The J1 speed was too high when the power was turned on.	When the robot is on halt, restart the controller.
6AC2	J2 encoder over speed error	5	The J2 speed was too high when the power was turned on.	↑
6AC3	J3 encoder over speed error	5	The J3 speed was too high when the power was turned on.	↑
6AC4	J4 encoder over speed error	5	The J4 speed was too high when the power was turned on.	↑
6AC5	J5 encoder over speed error	5	The J5 speed was too high when the power was turned on.	↑
6AC6	J6 encoder over speed error	5	The J6 speed was too high when the power was turned on.	↑
6AC7	J7 encoder over speed error	5	The J7 speed was too high when the power was turned on.	↑

Code	Message	Level	What happens:	What to do:
6AC8	J8 encoder over speed error	5	The J8 speed was too high when the power was turned on.	↑
6AC9	J1 encoder communication error	4	The J1 encoder data has not been updated correctly.	(1) Check that the FG wires of the robot unit and controller are grounded properly. (2) Check that no noise sources (e.g., welding machines) are in the vicinity of the robot unit or controller.
6ACA	J2 encoder communication error	4	The J2 encoder data has not been updated correctly.	↑
6ACB	J3 encoder communication error	4	The J3 encoder data has not been updated correctly.	↑
6ACC	J4 encoder communication error	4	The J4 encoder data has not been updated correctly.	↑
6ACD	J5 encoder communication error	4	The J5 encoder data has not been updated correctly.	↑
6ACE	J6 encoder communication error	4	The J6 encoder data has not been updated correctly.	↑
6ACF	J7 encoder communication error	4	The J7 encoder data has not been updated correctly.	↑
6AD0	J8 encoder communication error	4	The J8 encoder data has not been updated correctly.	↑
6AD1	J1 encoder data not received	4	Cannot receive J1 encoder data.	(1) Check that the FG wires of the robot unit and controller are grounded properly. (2) Check that no noise sources (e.g., welding machines) are in the vicinity of the robot unit or controller. (3) Check the cables between the robot unit and controller for connection.
6AD2	J2 encoder data not received	4	Cannot receive J2 encoder data.	↑
6AD3	J3 encoder data not received	4	Cannot receive J3 encoder data.	↑
6AD4	J4 encoder data not received	4	Cannot receive J4 encoder data.	↑

Code	Message	Level	What happens:	What to do:
6AD5	J5 encoder data not received	4	Cannot receive J5 encoder data.	↑
6AD6	J6 encoder data not received	4	Cannot receive J6 encoder data.	↑
6AD7	J7 encoder data not received	4	Cannot receive J7 encoder data.	↑
6AD8	J8 encoder data not received	4	Cannot receive J8 encoder data.	↑
6AD9	J1 encoder over heat error	4	The internal temperature of the J1 encoder is too high.	This high temperature state may break the encoder, so follow the steps below. (1) Check the temperature of the operating environment. (2) Check that the specifications of the hand (inc. workpiece) do not exceed the acceptable capacity of the robot. (3) Set timers between motion commands or decrease the speed and/or acceleration. (Before performing the operation again, wait for at least one minute.)
6ADA	J2 encoder over heat error	4	The internal temperature of the J2 encoder is too high.	↑
6ADB	J3 encoder over heat error	4	The internal temperature of the J3 encoder is too high.	↑
6ADC	J4 encoder over heat error	4	The internal temperature of the J4 encoder is too high.	↑
6ADD	J5 encoder over heat error	4	The internal temperature of the J5 encoder is too high.	↑
6ADE	J6 encoder over heat error	4	The internal temperature of the J6 encoder is too high.	↑
6ADF	J7 encoder over heat error	4	The internal temperature of the J7 encoder is too high.	↑
6AE0	J8 encoder over heat error	4	The internal temperature of the J8 encoder is too high.	↑
6AE1	J1 encoder battery low voltage	2	The battery voltage level of the J1 encoder has dropped.	Replace the encoder backup battery of the related joint.
6AE2	J2 encoder battery low voltage	2	The battery voltage level of the J2 encoder has dropped.	↑
6AE3	J3 encoder battery low voltage	2	The battery voltage level of the J3 encoder has dropped.	↑

Code	Message	Level	What happens:	What to do:
6AE4	J4 encoder battery low voltage	2	The battery voltage level of the J4 encoder has dropped.	↑
6AE5	J5 encoder battery low voltage	2	The battery voltage level of the J5 encoder has dropped.	↑
6AE6	J6 encoder battery low voltage	2	The battery voltage level of the J6 encoder has dropped.	↑
6AE7	J7 encoder battery low voltage	2	The battery voltage level of the J7 encoder has dropped.	↑
6AE8	J8 encoder battery low voltage	2	The battery voltage level of the J8 encoder has dropped.	↑
6AE9	J1 encoder overflow warning	4	If the motor keeps running in the current direction, the J1 position encoder counter will be overflown.	If you need to rotate the motor in the same direction further, reset the encoder of the related joint and perform CALSET.
6AEA	J2 encoder overflow warning	4	If the motor keeps running in the current direction, the J2 position encoder counter will be overflown.	↑
6AEB	J3 encoder overflow warning	4	If the motor keeps running in the current direction, the J3 position encoder counter will be overflown.	↑
6AEC	J4 encoder overflow warning	4	If the motor keeps running in the current direction, the J4 position encoder counter will be overflown.	↑
6AED	J5 encoder overflow warning	4	If the motor keeps running in the current direction, the J5 position encoder counter will be overflown.	↑
6AEE	J6 encoder overflow warning	4	If the motor keeps running in the current direction, the J6 position encoder counter will be overflown.	↑
6AEF	J7 encoder overflow warning	4	If the motor keeps running in the current direction, the J7 position encoder counter will be overflown.	↑
6AF0	J8 encoder overflow warning	4	If the motor keeps running in the current direction, the J8 position encoder counter will be overflown.	↑

Code	Message	Level	What happens:	What to do:
6AF3	Interference area detected by J1,2,3	3	The end-effector has came in the defined interference check area.	Run the robot to make the end-effector go out of the interference check area and then try the operation again.
736B	Automatic load not permitted	1	When the Program List window or the Select Variable Type window was opened, you attempted to carry out automatic load.	Try automatic load again with the teach pendant.
73FF	Cannot start program during the HALT process execution.	2	Attempted to start a program during HALT processing. (This error will occur also in Teach Check mode if you start a program immediately after releasing the deadman switch.)	Wait for a while and then restart the program. If HALT is commanded to all tasks, no program can run until all tasks come to stop. If step-stop is commanded to all tasks but any task is stopped by WAIT statement so as not to step-stop, then satisfy the WAIT condition to step-stop the task or stop that task. (If placed in Slow mode with the SS function, the task will come into the same state.) When operating the robot through I/Os, switching the external mode or Auto Enable will cause stop to all tasks, so the program cannot start for a while. Wait for a second or more and start the program.
77D1	Undefined arm group	4	Attempted to get an undefined arm group.	Correct the program. Or define the arm group, turn the controller power off and then on, and start the operation again.
77D2	Arm group has been taken	4	Attempted to get any other different arm group from that already gotten.	Correct the program so as not to get the different arm group in the same program.
77E4	EX(EXA) option can use only exjoint	3	The EX (EXA) option cannot drive the robot joints.	Correct the program so that no robot joint is included in the EX (EXA) option.
77E5	This joint is not available	3	The position of the extended joint disabled cannot be taken.	Correct the program. Or enable the extended joint, turn the controller power off and then on, then start the operation again.

Code	Message	Level	What happens:	What to do:
77E6	Cannot take semaphore of invalid joint	3	Cannot get an arm group containing an extended joint disabled.	↑
77E7	Boundless rotatory joint isn't available	3	This command or operation cannot execute to a boundless rotatory joint.	Release the joint from the boundless rotation or do not execute this command or operation to the boundless rotatory joint.
77E8	Not set boundless rotatory joint	3	This command or operation cannot execute to a limited rotatory joint.	Set the joint for boundless rotation or do not execute this command or operation to the limited rotatory joint.
77EA	Can not start Program in Manual Mode	3	In Manual or Teach Check mode, a privilege task attempted to start a user task.	Stop the privilege task or delete a user task start instruction written in the privilege task.
77EB	Can not execute \"takearm\" by TSR	3	To prohibit robot motions, a privilege task called the TAKEARM command.	Correct the program so that robot motions are written in a user task.
77EC	TSR setting is \"P-TASK No Use\"	2	Attempted to start a privilege task, but the privilege task mode has been disabled.	Enable the privilege task mode and restart the robot controller.
77ED	Cannot start TSR	3	Attempted to run a privilege task in usual tasks.	Correct the program not to attempt to start a privilege task in usual tasks.
77EE	TSR can't be started continuously	1	A privilege task is not allowed to start in succession.	To run a privilege task in succession explicitly, correct the program so that the repetitions loop inside the privilege task.
77EF	Failure to do calibration (INIT)	3	Failed to CAL due to any cause during execution of INIT.	Fix the CAL failure cause and restart the robot controller.
77F0	Failure to turn on the motor power (INIT)	3	Failed to turn the motor on due to any cause during execution of INIT. INIT commands executed in succession have caused a timeout error.	Fix the failure cause and restart the robot controller.
77F1	Command exclusive to TSR	3	In a general task, you attempted to execute a command(s) exclusively allowed for a privilege task.	Correct the program so that the command executes in a privilege task.
77F2	COM-DATA length too short	3	In binary transmission, the data length specified by <inputbytes> in linputb command is too short.	Send data whose length matches <inputbytes> from external equipment to the robot controller.

DENSO ROBOT

Extended-Joints Support (H∗-D, VS-D)
Main Software Enhancement
(Version 1.5 & Version 1.6)

SUPPLEMENT

First Edition December 2000
Second Edition March 2001

DENSO CORPORATION
Industrial Systems Product Division

3C30C

The purpose of this manual is to provide accurate information in the handling and operating of the robot. Please feed 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.