DENSO ROBOT

Vertical articulated

V*-D/-E SERIES

Horizontal articulated

H*-D/-E SERIES

Cartesian coordinate

XYC-4D SERIES

Vision device

μVision-21 SERIES

PROGRAMMER'S MANUAL II
PAC Library

(Ver. 1.98)

Preface

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

Before operating your robot, read this manual carefully to safely get the maximum benefit from your robot in your assembling operations.

Robot series and/or models covered by this manual

- Vertical articulated robot	V*-D/-E series
- Horizontal articulated robot	H*-D/-E series
- Cartesian coordinate robot	XYC-4D series
- Vision devaicec	μVision-21 series

Robot controller version (Note)

- Applicable to up to Ver. 1.95 of the RC5 type controller.

Note: The robot controller version is indicated in the main software ver. column of the controller setting table affixed on the controller. It can also be confirmed from the teaching pendant by reading the ROM version column displayed by “basic screen” - “F6 setting” - “F6 maintenance” - “F2 version”.

Important

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

How the documentation set is organized

The documentation set consists of the following books. If you are unfamiliar with this robot and option(s), please read all books and understand them fully before operating your robot and option(s).

GENERAL INFORMATION ABOUT ROBOT

Provides the packing list of the robot and outlines of the robot system, robot unit, and robot controller.

INSTALLATION & MAINTENANCE GUIDE

Provides instructions for installing the robot components and customizing your robot, and maintenance & inspection procedures.

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.

SETTING-UP MANUAL

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

WINCAPSII GUIDE

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

PROGRAMMER'S MANUAL (I), (II) - this book -

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

RC5 CONTROLLER
INTERFACE MANUAL

Describes the RC5 controller, interfacing with external devices, system- and user-input/output signals, and I/O circuits.

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. These tables provide detailed description and recovery ways.

OPTIONS MANUAL

Describes the specifications, installation, and use of optional devices.

How this book is organized

This book is just one part of the documentation set. This book consists of SAFETY PRECAUTIONS and chapters one through five.

SAFETY PRECAUTIONS

Defines safety terms, safety related symbols and provides precautions that should be observed. Be sure to read this section before operating your robot.

Commands Listed in Alphabetical Order

Commands Listed According to Functions

PAC Library

This chapter provides an explanation of program libraries attached to WINCAPSII, as standard accessories.

Index

Contents

Commands Listed in Alphabetical Order

4-axis	6-axis	Vision device	
⊙	⊙	⊙	Available with all series of robots and vision device.
○	○	○	Available with all series of robots. The command specifications differ between the 4-axis, 6-axis robot, and vision device.
⊙	V1.2		Available with the 4-axis robots and the 6-axis robots of Version 1.2 or later.

Commands	Functions	4-axis	6-axis	Vision device	Refer to:
A					
arrange_button_pos	Specify the button arrangement (position, size, and so on) on the screen.	V1.7	V1.7		105
arrange_button_size	Specifies the button arrangement (position, size, and so on) on the screen.	V1.7	V1.7		105
aspACLD	Changes the internal load condition values. There are the mass of payload, noted in grams (g), and the payload center of gravity, noted in millimeters (mm), for the load condition values. Designate both of them.	⊙	⊙		2
aspChange	Selects the internal mode for proper control setting of motion optimization.	⊙	⊙		4
C					
ClearSrvMonitor	Initializes the pointer of data obtained by the single-joint servo data monitor function. (For Ver. 1.5 or later)	V1.5	V1.6		91
D					
dioSync	Synchronizes with an external device (such as a sequencer) connected to DIO.	⊙	⊙		37
M					
make_LABEL	Creates a title (label).	V1.7	V1.7		103
make_LED	Creates an LED button.	V1.7	V1.7		102
make_PARAM_BOX	Creates a variable button (entry & display box).	V1.7	V1.7		103
make_PB	Creates a PB button.	V1.7	V1.7		102
MotionComp	Judges whether execution of running motion commands is complete. (For Ver. 1.5 or later)	V1.5	V1.6		85
MotionSkip	Aborts running motion commands. (For Ver. 1.5 or later)	V1.5	V1.6		84
mvResetPulseWidth	Restores default encoder pulse counts for positioning allowance.	⊙	⊙		38
mvResetPulseWidthJnt	Resets the encoder pulse count for an allowable positioning error for a specified extended-joint to the default.	V1.5	V1.6		39
mvResetTimeOut	Restores the default motion finish timeout value.	⊙	⊙		40
mvReverseFlip	4-axis figure reverse	⊙	⊙		40
mvSetPulseWidth	Sets the permissible stop pulse width.	⊙	⊙		41
mvSetPulseWidthJnt	Sets the encoder pulse count for an allowable positioning error for a specified extended-joint.	V1.5	V1.6		42
mvSetTimeOut	Sets the timeout value for movement finish.	⊙	⊙		43

4-axis	6-axis	Vision device	
⦿	⦿	⦿	Available with all series of robots and vision device.
○	○	○	Available with all series of robots. The command specifications differ between the 4-axis, 6-axis robot, and vision device.
⦿	V1.2		Available with the 4-axis robots and the 6-axis robots of Version 1.2 or later.

Commands	Functions	4-axis	6-axis	Vision device	Refer to:
N					
ndApra	Performs absolute operation with the tool coordinate system specified (exclusively for four-axis robots).	⦿			16
ndDepa	Performs absolute operation with the tool coordinate system specified (exclusively for four-axis robots).	⦿			17
ndInb	Converts the input of a designated port to a decimal number and treats it as a binary number.	⦿	⦿		5
ndJf	A conditional branch upon receipt of OK/NG from an external device (RS232C input/output).	⦿	⦿		6
ndOnb	Converts a decimal number to a binary number and outputs it from a designated port.	⦿	⦿		7
ndOnbl	Converts a decimal number to a binary and outputs it from a designated port.	⦿	⦿		8
ndVType	Designates the protocol for communication with an external device (RS232C output).	⦿	⦿		14
ndVcom	Communication with an external device (kernel) (RS232C input/output)	⦿	⦿		15
ndVdt	Memorizes a variable transferred from an external device (RS232C input/output).	⦿	⦿		9
ndVis	Transfers a designated 2-digit integer to an external device (RS232C input/output).	⦿	⦿		10
ndVput	Transfers the position posture to an external device (RS232C input/output).	⦿	⦿		11
ndVrst	Initializes an external device (RS232C input/output).	⦿	⦿		12
ndVset	Receives data from an external device (RS232C input/output).	⦿	⦿		13
O					
OffPWM	Releases PWM switching on the specified axis (exclusively for a four-axis robot).	⦿			56
OffSrvLock	Releases servo lock for the specified axis. (Exclusively for four-axis robots)	⦿			54
OnPWM	Performs PWM switching on a specified axis (exclusively for a four-axis robot).	⦿			55
OnSrvLock	Servo-locks a specified axis (exclusively for a four-axis robot).	⦿			53
P					
pltDecCnt	Decreases the count of the total palletizing counter.	⦿	⦿		18
pltGetCnt	Obtains the total palletizing counter.	⦿	⦿		18
pltGetK	Obtains palletizing set value K.	⦿	⦿		19
pltGetK1	Obtains palletizing counter K1.	⦿	⦿		19
pltGetM	Obtains palletizing set value M.	⦿	⦿		20
pltGetM1	Obtains palletizing counter M1.	⦿	⦿		20
pltGetN	Obtains palletizing set value N.	⦿	⦿		21
pltGetN1	Obtains palletizing counter N1.	⦿	⦿		21
pltGetNextPos	Obtains the next position.	⦿	⦿		22
pltGetPLT1END	Obtains a palletizing 1-row completion flag.	⦿	⦿		22
pltGetPLTEND	Obtains a palletizing all-row completion flag.	⦿	⦿		23
pltIncCnt	Increases the total palletizing counter.	⦿	⦿		23
pltInit1	Palletizing initialization template 1.	⦿	⦿		24
pltInitialize	Palletizing initialization	⦿	⦿		25
pltKernel	Palletizing motion (kernel)	⦿	⦿		26

4-axis	6-axis	Vision device	
⊙	⊙	⊙	Available with all series of robots and vision device.
○	○	○	Available with all series of robots. The command specifications differ between the 4-axis, 6-axis robot, and vision device.
⊙	V1.2		Available with the 4-axis robots and the 6-axis robots of Version 1.2 or later.

Commands	Functions	4-axis	6-axis	Vision device	Refer to:
pltLetCnt	Sets the total palletizing counter.	⊙	⊙		27
pltLetK1	Sets the palletizing counter K1.	⊙	⊙		27
pltLetM1	Sets the palletizing counter M1.	⊙	⊙		28
pltLetN1	Sets the palletizing counter N1.	⊙	⊙		28
pltMain1	Palletizing template 1	⊙	⊙		29
pltMain2	Palletizing template 2	⊙	⊙		29
pltMove	Standard palletizing template 1	⊙	⊙		30
pltMove0	Standard palletizing motion 1	⊙	⊙		31
pltResetAll	Resets all palletizing counters.	⊙	⊙		32
pltResetPLT1END	Resets a palletizing 1-row completion flag.	⊙	⊙		32
pltResetPLTEND1	Resets a palletizing all-row completion flag.	⊙	⊙		33
R					
ResetCompControl	Disables the compliance control function.		V1.4		66
ResetCompEralw	Initializes the allowable deviation values of the position and the posture of the tool tip under the compliance control.		V1.4		79
ResetCompJLimit	Initializes the current limit under the compliance control.		V1.4		75
ResetCompRate	Initializes the rate of compliance under the compliance control.		V1.4		71
ResetCompVMode	Disables the velocity control mode under the compliance control.		V1.4		77
ResetCurLmt	Resets the motor current limit of the specified axis.	⊙	V1.2		50
Resetcycloid	Causes transition from the cycloid mode to the ordinary operation mode.	⊙	V1.4		59
ResetCycloidJnt	Cancels the cycloid mode set for a specified extended-joint and restores the normal mode.	V1.5	V1.6		60
ResetDampRate	Initializes the damping rate under the compliance control.		V1.4		81
ResetEralw	Resets the allowable deviation value of the specified axis to the initial value.	⊙	V1.2		52
ResetFrcAssist	Initializes the force assistance under the compliance control.		V1.4		73
ResetFrcLimit	Initializes the rate of force limiting.		V1.4		69
ResetGravity	Disables the balance setting between the limited motor torque and gravity torque, which is made with SetGravity.		V1.2		45
ResetGrvOffset	Disables the gravity offset function.		V1.2		47
ResetVibControl	Returns from the residual vibration reduction control mode to the normal control mode.		V1.4		83
S					
set_button_param	Specifies button attributes (type, color, shape, and so on).	V1.7	V1.7		104
SetCompControl	Enables the compliance control function.		V1.4		64
SetCompEralw	Sets the allowable deviation values of the position and the posture of the tool tip under the compliance control.		V1.4		78
SetCompFControl	Enables the compliance control function. However the gravity offset compensation processing is not active.		V1.4		65
SetCompJLimit	Sets the current limit under the compliance control.		V1.4		74

4-axis	6-axis	Vision device	
⦿	⦿	⦿	Available with all series of robots and vision device.
○	○	○	Available with all series of robots. The command specifications differ between the 4-axis, 6-axis robot, and vision device.
⦿	V1.2		Available with the 4-axis robots and the 6-axis robots of Version 1.2 or later.

Commands	Functions	4-axis	6-axis	Vision device	Refer to:
SetCompRate	Sets the rate of compliance under the compliance control.		V1.4		70
SetCompVMode	Sets the velocity control mode under the compliance control.		V1.4		76
SetCurLmt	Sets the limit of motor current to be applied to the specified axis.	○	V1.2		48
Setcycloid	Causes transition to the cycloid operation mode to suppress overshoot and residual vibration at the end of PTP operation.	⦿	V1.4		57
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.	V1.5	V1.6		58
SetDampRate	Sets the damping rate under the compliance control.		V1.4		80
SetEralw	Modifies the allowable deviation of the specified axis.	⦿	V1.2		51
SetForce_HC	A current limitation library that specifies the thrust (unit: N) of the Z coordinate with the HC robot.	⦿			63
SetForce_HM	A current limitation library that specifies the thrust (unit: N) of the Z coordinate with the HM/HS robot.	⦿			62
SetFrcAssist	Sets the force assistance under the compliance control.		V1.4		72
SetFrcCoord	Selects a coordinate system for the compliance control setting.		V1.4		67
SetFrcLimit	Sets the rate of force limiting.		V1.4		68
SetGravity	Compensates for the static load (gravity torque) applied to each joint and attains balance with gravity torque.		V1.2		44
SetGrvOffset	Compensates the torque of each joint programmed with SetGravity for gravity torque.		V1.2		46
SetMonitorCond	Sets the monitoring conditions for single-joint servo data monitor. (For Ver. 1.5 or later)	V1.5	V1.6		88
SetVibControl	Sets to the residual vibration reduction control mode.		V1.4		82
single_button_set	Creates only one button.	V1.7	V1.7		104
StartSrvMonitor	Starts monitoring single-joint servo data. (For Ver. 1.5 or later)	V1.5	V1.6		89
StopSrvMonitor	Stops monitoring single-joint servo data. (For Ver. 1.5 or later)	V1.5	V1.6		90
T					
tolChange	Tool change	⦿	⦿		34
tolInit1	Tool change initialization template 1	⦿	⦿		35
tolInitialize	Tool change initialization	⦿	⦿		35
tolKernel	Tool change motion (kernel)	⦿	⦿		36
tolMain1	Tool change template 1	⦿	⦿		36
V					
viTran6	Transforms the vision coordinates to robot coordinates (for 6 axes).	⦿	⦿		106

Commands Listed According to Functions

4-axis	6-axis	Vision device	
⊙	⊙	⊙	Available with all series of robots and vision device.
○	○	○	Available with all series of robots. The command specifications differ between the 4-axis, 6-axis robot, and vision device.
⊙	V1.2		Available with the 4-axis robots and the 6-axis robots of Version 1.2 or later.

Classified by functions	Commands	Functions	4-axis	6-axis	Vision device	Refer to:
PAC Library						
Conventional Language	aspACLD	Changes the internal load condition values. There are the mass of payload, noted in grams (g), and the payload center of gravity, noted in millimeters (mm), for the load condition values. Designate both of them.	⊙	⊙		2
	aspChange	Selects the internal mode for proper control setting of motion optimization.	⊙	⊙		4
	ndInb	Converts the input of a designated port to a decimal number and treats it as a binary number.	⊙	⊙		5
	ndJf	A conditional branch upon receipt of OK/NG from an external device (RS232C input/output).	⊙	⊙		6
	ndOnb	Converts a decimal number to a binary number and outputs it from a designated port.	⊙	⊙		7
	ndOnbI	Converts a decimal number to a binary and outputs it from a designated port.	⊙	⊙		8
	ndVdt	Memorizes a variable transferred from an external device (RS232C input/output).	⊙	⊙		9
	ndVis	Transfers a designated 2-digit integer to an external device (RS232C input/output).	⊙	⊙		10
	ndVput	Transfers the position posture to an external device (RS232C input/output).	⊙	⊙		11
	ndVrst	Initializes an external device (RS232C input/output).	⊙	⊙		12
	ndVset	Receives data from an external device (RS232C input/output).	⊙	⊙		13
	ndVType	Designates the protocol for communication with an external device (RS232C output).	⊙	⊙		14
	ndVcom	Communication with an external device (kernel) (RS232C input/output)	⊙	⊙		15
	ndApra	Performs absolute operation with the tool coordinate system specified (exclusively for four-axis robots).	⊙			16

4-axis	6-axis	Vision device	
⊙	⊙	⊙	Available with all series of robots and vision device.
○	○	○	Available with all series of robots. The command specifications differ between the 4-axis, 6-axis robot, and vision device.
⊙	V1.2		Available with the 4-axis robots and the 6-axis robots of Version 1.2 or later.

Classified by functions	Commands	Functions	4-axis	6-axis	Vision device	Refer to:
	ndDepa	Performs absolute operation with the tool coordinate system specified (exclusively for four-axis robots).	⊙			17
Palletizing	pltDecCnt	Decreases the count of the total palletizing counter.	⊙	⊙		18
	pltGetCnt	Obtains the total palletizing counter.	⊙	⊙		18
	pltGetK	Obtains palletizing set value K.	⊙	⊙		19
	pltGetK1	Obtains palletizing counter K1.	⊙	⊙		19
	pltGetM	Obtains palletizing set value M.	⊙	⊙		20
	pltGetM1	Obtains palletizing counter M1.	⊙	⊙		20
	pltGetN	Obtains palletizing set value N.	⊙	⊙		21
	pltGetN1	Obtains palletizing counter N1.	⊙	⊙		21
	pltGetNextPos	Obtains the next position.	⊙	⊙		22
	pltGetPLT1END	Obtains a palletizing 1-row completion flag.	⊙	⊙		22
	pltGetPLTEND	Obtains a palletizing all-row completion flag.	⊙	⊙		23
	pltIncCnt	Increases the total palletizing counter.	⊙	⊙		23
	pltInit1	Palletizing initialization template 1.	⊙	⊙		24
	pltInitialize	Palletizing initialization	⊙	⊙		25
	pltKernel	Palletizing motion (kernel)	⊙	⊙		26
	pltLetCnt	Sets the total palletizing counter.	⊙	⊙		27
	pltLetK1	Sets the palletizing counter K1.	⊙	⊙		27
	pltLetM1	Sets the palletizing counter M1.	⊙	⊙		28
	pltLetN1	Sets the palletizing counter N1.	⊙	⊙		28
	pltMain1	Palletizing template 1	⊙	⊙		29
	pltMain2	Palletizing template 2	⊙	⊙		29
	pltMove	Standard palletizing template 1	⊙	⊙		30
	pltMove0	Standard palletizing motion 1	⊙	⊙		31
	pltResetAll	Resets all palletizing counters.	⊙	⊙		32
	pltResetPLT1END	Resets a palletizing 1-row completion flag.	⊙	⊙		32
	pltResetPLTEND1	Resets a palletizing all-row completion flag.	⊙	⊙		33
Tool Operation	tolChange	Tool change	⊙	⊙		34
	tolInit1	Tool change initialization template 1	⊙	⊙		35
	tolInitialize	Tool change initialization	⊙	⊙		35
	tolKernel	Tool change motion (kernel)	⊙	⊙		36
	tolMain1	Tool change template 1	⊙	⊙		36
Input/Output	dioSync	Synchronizes with an external device (such as a sequencer) connected to DIO.	⊙	⊙		37

4-axis	6-axis	Vision device	
⦿	⦿	⦿	Available with all series of robots and vision device.
○	○	○	Available with all series of robots. The command specifications differ between the 4-axis, 6-axis robot, and vision device.
⦿	V1.2		Available with the 4-axis robots and the 6-axis robots of Version 1.2 or later.

Classified by functions	Commands	Functions	4-axis	6-axis	Vision device	Refer to:
Arm Movement	mvResetPulseWidth	Restores default encoder pulse counts for positioning allowance.	⦿	⦿		38
	mvResetPulseWidthJnt	Resets the encoder pulse count for an allowable positioning error for a specified extended-joint to the default.	V1.5	V1.6		39
	mvResetTimeOut	Restores the default motion finish timeout value.	⦿	⦿		40
	mvReverseFlip	4-axis figure reverse	⦿	⦿		40
	mvSetPulseWidth	Sets the permissible stop pulse width.	⦿	⦿		41
	mvSetPulseWidthJnt	Sets the encoder pulse count for an allowable positioning error for a specified extended-joint.	V1.5	V1.6		42
	mvSetTimeOut	Sets the timeout value for movement finish.	⦿	⦿		43
	SetGravity	Compensates for the static load (gravity torque) applied to each joint and attains balance with gravity torque.		V1.2		44
	ResetGravity	Disables the balance setting between the limited motor torque and gravity torque, which is made with SetGravity.		V1.2		45
	SetGrvOffset	Compensates the torque of each joint programmed with SetGravity for gravity torque.		V1.2		46
	ResetGrvOffset	Disables the gravity offset function.		V1.2		47
	SetCurLmt	Sets the limit of motor current to be applied to the specified axis.	○	V1.2		48
	ResetCurLmt	Resets the motor current limit of the specified axis.	⦿	V1.2		50
	SetEralw	Modifies the allowable deviation of the specified axis.	⦿	V1.2		51
	ResetEralw	Resets the allowable deviation value of the specified axis to the initial value.	⦿	V1.2		52
	OnSrvLock	Servo-locks a specified axis (exclusively for a four-axis robot).	⦿			53
	OffSrvLock	Releases servo lock for the specified axis. (Exclusively for four-axis robots)	⦿			54
	OnPWM	Performs PWM switching on a specified axis (exclusively for a four-axis robot).	⦿			55
	OffPWM	Releases PWM switching on the specified axis (exclusively for a four-axis robot).	⦿			56

4-axis	6-axis	Vision device	
⊙	⊙	⊙	Available with all series of robots and vision device.
○	○	○	Available with all series of robots. The command specifications differ between the 4-axis, 6-axis robot, and vision device.
⊙	V1.2		Available with the 4-axis robots and the 6-axis robots of Version 1.2 or later.

Classified by functions	Commands	Functions	4-axis	6-axis	Vision device	Refer to:
	Setcycloid	Causes transition to the cycloid operation mode to suppress overshoot and residual vibration at the end of PTP operation.	⊙	V1.4		57
	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.	V1.5	V1.6		58
	Resetcycloid	Causes transition from the cycloid mode to the ordinary operation mode.	⊙	V1.4		59
	ResetCycloidJnt	Cancels the cycloid mode set for a specified extended-joint and restores the normal mode.	V1.5	V1.6		60
	SetForce_HM	A current limitation library that specifies the thrust (unit: N) of the Z coordinate with the HM/HS robot.	⊙			62
	SetForce_HC	A current limitation library that specifies the thrust (unit: N) of the Z coordinate with the HC robot.	⊙			63
	SetCompControl	Enables the compliance control function.		V1.4		64
	SetCompFControl	Enables the compliance control function. However the gravity offset processing is not executed.		V1.4		65
	ResetCompControl	Disables the compliance control function.		V1.4		66
	SetFrcCoord	Selects a coordinate system for the force limiting setting.		V1.4		67
	SetFrcLimit	Sets the rate of force limiting.		V1.4		68
	ResetFrcLimit	Initializes the rate of force limiting.		V1.4		69
	SetCompRate	Sets the rate of compliance under the compliance control.		V1.4		70
	ResetCompRate	Initializes the ratio of compliance under the compliance control.		V1.4		71
	SetFrcAssist	Sets the force assistance under the compliance control.		V1.4		72
	ResetFrcAssist	Initializes the force assistance under the compliance control.		V1.4		73
	SetCompJLimit	Sets the current limit under the compliance control.		V1.4		74
	ResetCompJLimit	Initializes the current limit under the compliance control.		V1.4		75
	SetCompVMode	Sets the velocity control mode under the compliance control.		V1.4		76

4-axis	6-axis	Vision device	
⊙	⊙	⊙	Available with all series of robots and vision device.
○	○	○	Available with all series of robots. The command specifications differ between the 4-axis, 6-axis robot, and vision device.
⊙	V1.2		Available with the 4-axis robots and the 6-axis robots of Version 1.2 or later.

Classified by functions	Commands	Functions	4-axis	6-axis	Vision device	Refer to:
	ResetCompVMode	Disables the velocity control mode under the compliance control.		V1.4		77
	SetCompEralw	Sets the allowable deviation values of the position and the posture of the tool tip under the compliance control.		V1.4		78
	ResetCompEralw	Initializes the allowable deviation values of the position and the posture of the tool tip under the compliance control.		V1.4		79
	SetDampRate	Sets the damping rate under the compliance control.		V1.4		80
	ResetDampRate	Initializes the damping rate under the compliance control.		V1.4		81
	SetVibControl	Sets to the residual vibration reduction control mode.		V1.4		82
	ResetVibControl	Returns from the residual vibration reduction control mode to the normal control mode.		V1.4		83
	MotionSkip	Aborts running motion commands. (For Ver. 1.5 or later)	V1.5	V1.6		84
	MotionComp	Judges whether execution of running motion commands is complete. (For Ver. 1.5 or later)	V1.5	V1.6		85
Single-Joint Servo Data Monitor	SetMonitorCond	Sets the monitoring conditions for single-joint servo data monitor. (For Ver. 1.5 or later)	V1.5	V1.6		88
	StartSrvMonitor	Starts monitoring single-joint servo data. (For Ver. 1.5 or later)	V1.5	V1.6		89
	StopSrvMonitor	Stops monitoring single-joint servo data. (For Ver. 1.5 or later)	V1.5	V1.6		90
	ClearSrvMonitor	Initializes the pointer of data obtained by the single-joint servo data monitor function. (For Ver. 1.5 or later)	V1.5	V1.6		91
Libraries for Creating Operation Panel Screen	make_PB	Creates a PB button.	V1.7	V1.7		102
	make_LED	Creates an LED button.	V1.7	V1.7		102
	make_LABEL	Creates a title (label).	V1.7	V1.7		103
	make_PARAM_BOX	Creates a variable button (entry & display box).	V1.7	V1.7		103
	single_button_set	Creates only one button.	V1.7	V1.7		104
	set_button_param	Specifies button attributes (type, color, shape, and so on).	V1.7	V1.7		104
	arrange_button_size	Specifies the button arrangement (position, size, and so on) on the screen.	V1.7	V1.7		105
	arrange_button_pos	Specify the button arrangement (position, size, and so on) on the screen.	V1.7	V1.7		105

4-axis	6-axis	Vision device	
⦿	⦿	⦿	Available with all series of robots and vision device.
○	○	○	Available with all series of robots. The command specifications differ between the 4-axis, 6-axis robot, and vision device.
⦿	V1.2		Available with the 4-axis robots and the 6-axis robots of Version 1.2 or later.

Classified by functions	Commands	Functions	4-axis	6-axis	Vision device	Refer to:
Vision	viTran6	Transforms the vision coordinates to robot coordinates (for 6 axes).	⦿	⦿		106

1 Using the Program Library

To use the program library, you must run a program bank (tool) from the PAC manager in WINCAPSII and install necessary libraries.
To operate the program bank, refer to Owner's Manual (WINCAPSII).

2 Criterion of the Program Library

The standard program library is divided into the following classes which are provided in the program bank of WINCAPSII.

Class Name	Description
Conventional Language	Provides similar functions to those of the conventional language.
Palletizing	Provides palletizing functions.
Tool Operation	Provides tool operation related functions.
Input/Output	Provides DIO and RS232C input/output related functions.
Arm Motion	Provides arm motion related functions.
Vision	Provides vision operation related functions.
Ver. 1.2 compatible	Provides the version 1.2 compatible library that can be used in Controller Software Version 1.2* or earlier. This class contains five libraries--ndVcom, pltMove, pltMove0, ResetCurLmt and ResetEralw. Use libraries in other classes except for those five libraries. **Note:** If in Version 1.2* or earlier any of those five libraries not in this class but in other classes are used, a compilation error will result.
Vision-21	Provides Vision-21 operation related functions.
Operation Panel Screen	Provides functions for creating the operation panel screen in the teach pendant. **Note:** For creating the operation panel screen, sample programs are also provided. See section 8.1.

3 Conventional Language

Provides similar functions to those of the conventional language which were used in the RC3 controller model.

aspACLD (Library)

Function

Changes the internal load condition values. There are the mass of payload, noted in grams (g), and the payload center of gravity, noted in millimeters (mm), for the load condition values. Designate both of them. (See **Note1**.)

Format

aspACLD (<Mass of payload>, <Payload center of gravity coordinate X >, <Payload center of gravity coordinate Y>, <Payload center of gravity coordinate Z>)

Note1: For 4-axes robot in Ver.1.9 or later, <Payload center of gravity coordinate Z> is replaced with <Inertia of payload (kgcm^2)>

Explanation

(Example for 6-axces robot)

The mass of payload is the mass of load (end-effector and workpiece) mounted on the 6th axis of the robot. This unit is designated as (g).
For the load center of gravity position, designate the payload center of gravity using the TOOL0 coordinates. The unit is millimeters (mm). (See **Note1**.)
The reference position of the TOOL0 coordinates is in the center of the 6th axis flange. For component Y, the direction is from the flange center to the pinhole of Φ6H7 (orientation vector direction). For component Z, the direction is vertical to the flange surface through the flange center (approach vector direction). For component X, the direction of the X axis (normal vector direction) is the right-hand coordinate system, when the orientation vector is set to the Y axis and the approach vector is set to the Z axis. Refer to the PROGRAMMER'S MANUAL I, "4.7 Control Set of Motion Optimization in User Preferences."

Even if you change only one of the four values of the mass of payload, the payload center of gravity X, the payload center of gravity Y, and payload center of gravity Z ,describe all of the 4 values again.

It takes about 0.1 sec. to switch the load condition values. Frequently switching the load condition may cause operational delays. Do not change the mode during pass motion while near an obstacle because the path locus may shift. This may delay switching if you change the load condition values.

Macro Definition

Requires file <pacman.h>

Related Terms

Refer to the PROGRAMMER'S MANUAL I, " 4.7 Control Set of Motion Optimization in User Preferences."

Example

```
CALL aspACLD(8500,-50,100,80)  'Sets the internal payload conditions.
                               'Mass of payload:8500(g), Payload center of gravity
                               'component X: -50(mm), component Y: 100(mm),
                               'component Z: 80(mm)
```

Notes

(1) For the mass of payload, designate it with a numerical value of the specified range for each robot type. If you designate a value out of this range, the error message “60d2. The payload setting value exceeds the permissible value” will be displayed.

(2) For the payload center of gravity, enter it so that it satisfies the specified range for each robot type. If it is out of this range, the error message “60d2. The payload setting value exceeds the permissible value” will be displayed.

(3) When setting the internal payload condition, observe the following rule relative to the external payload condition. If not, the error message "60d2 The payload setting value exceeds the permissible value" will be displayed.

0.5 x External payload condition ≤ Internal payload condition ≤ External payload condition

aspChange (Library)

Function

Selects the internal mode for proper control setting of motion optimization.

Format

aspChange (<Mode>)

Explanation

This statement switches the mode for control setting of motion optimization.

<Setting value>
0 → Invalid
1 → Valid only for PTP
2 → Valid only for CP
3 → Valid for both PTP and CP.

It takes about 0.1 sec. to switch the load condition values. Frequently switching the load condition may cause operational delays. Do not change the mode during pass motion, near an obstacle because the path locus may shift. This may delay switching if you change the load condition values.

Macro Definition

Requires file <pacman.h>

Related Terms

Refer to the PROGRAMMER'S MANUAL I, " 4.7 Control Set of Motion Optimization in User Preferences."

Example

```
CALL aspChange(1)          'Sets the internal mode in the control sets of motion
                           'optimization to 1.
```

Notes

For <Mode>, designate it with a numerical value between 0 and 3. If it is out of this range, the error message "6003 Beyond the valid value range" will be displayed.

ndInb (Library)

Function

Converts the input of a designated port to a decimal number and treats it as a binary number.

Format

ndlnb (<Integer variable number>, <Least significant digit output port number>, <Most significant digit output port number>)

Explanation

This statement provides a similar function to an INB instruction in the conventional language.
The system reads the status of a designated input port signal and converts it to a decimal number by treating it as a binary number.
The system assigns a converted value to the integer variable.

Related Terms

ndOnb, ndOnbI

Remarks

You can define the same function with DEFIO and IN instruction in the PAC language. Use DEFIO and IN instructions because this method is more effective.
For input ports, use 16 ports or less in series. If you designate more than 16 ports, no processing will be done.

Example

```
CALL ndInb(1,552,567)     'Reads the status of input ports 552 to 567 as a 16-bit
                          'binary, converts it to decimal, and assigns it
                          'to the integer variable (1).
```

ndJf (Library)

Function

A conditional branch upon receipt of OK/NG from an external device (RS232C input/output).

Format

ndJf (<2-digit integer>, <Determination argument>)

Explanation

This statement provides a similar function to a JF instruction in the conventional language.
The system transfers a 2-digit integer to an external device and calculates a response result to execute a conditional branch in the program.
If the response from the external device is OK, the system proceeds to the next step. If it is not good, it branches to a labeled step.

Related Terms

ndVcom, ndVType

Example

```
#include <Pacman.h>
PROGRAM PRO1
DEFINT JF_VAL = 0
     FLUSH
     CALL ndVType(1)         'Sets the protocol (Conventional = 0/ New = 1).
     CALL ndVrst             'Initializes the external device.
     CALL ndVset(0)          'Clears the received data (VDT) to 0.
     CALL ndVis(3)           'Starts up the external device (Sends 03).
     CALL ndJf(3,JF_VAL)     'Obtains a response result from the external device
                             '(Sends 03).
     IF JF_VAL = TRUE THEN   'Receives data, if the response is OK (TRUE).
     CALL ndVset(3)          'Receives 10 data from the external device (Sends 03).
     CALL ndVdt(pacPOS,1)    'Assigns the data received from the external device
                             'to variable (P1).
     CALL ndVdt(pacJNT,1)    'Assigns the data received from the external device
                             'to variable (J1).
     CALL ndVdt(pacTRN,1)    'Assigns the data received from the external device
                             'to variable (T1).
     END IF
END
```

ndOnb (Library)

Function

Converts a decimal number to a binary number and outputs it from a designated port.

Format

ndOnb (<Integer value>, <Least significant digit output port number>, <Most significant digit output port number>)

Explanation

This statement provides a similar function to an ONB instruction in the conventional language.
The system converts an integer to a binary number and outputs it from a designated port.
Use the ndOnbI Library for the ONB instruction to format type I variable value output.

Related Terms

ndInb, ndOnbI

Remarks

For output ports, use 16 ports or less in series. When you designate more than 16 ports, if low order port > high order port, or if you designate a port not available for output, no processing will be done.

Example

```
CALL ndOnb(15,769,784)       'Converts a decimal 15 to a 16-bit binary and
                             'outputs it to output ports 769 to 784.
```

ndOnbI (Library)

Function

Converts a decimal number to a binary and outputs it from a designated port.

Format

ndOnbI (<Integer variable number>,<Least significant digit output port number>, <Most significant digit output port number>)

Explanation

This statement provides a similar function to an ONB instruction in the conventional language.
The system converts an integer to a binary number and outputs it from a designated port.
Use the ndOnb Library for the ONB instruction to format type I variable value output.

Related Terms

ndInb, ndOnb

Remarks

For output ports, use 16 ports or less in series. When you designate more than 16 ports, if low order port > high order port or if you designate a port not available for output, no processing will be done.

Example

```
CALL ndOnbI(1,769,784)      'Converts the value of integer variable I (number 1)
                            'to a 16-bit binary and outputs it to output ports
                            '769 to 784.
```

ndVdt (Library)

Function

Memorizes a variable transferred from an external device (RS232C input/output).

Format

ndVdt (<Storage variable type>, <Storage variable number>)

Explanation

This statement provides a similar function of a VDT instruction in the conventional language.
The system assigns the data transferred from an external device to a storage variable number, in a designated storage variable type format (P/J/T).

Macro Definition

Requires file <pacman.h>

Related Terms

ndVcom, ndVset

Example

```
#include <pacman.h>
PROGRAM PRO1
DEFINT JF_VAL = 0
   FLUSH
   CALL ndVType(1)          'Sets the protocol (Conventional = 0/ New = 1).
   CALL ndVrst              'Initializes the external device.
   CALL ndVset(0)           'Clears the received data (VDT) to 0.
   CALL ndVis(3)            'Starts up the external device (Sends 03).
   CALL ndJf(3,JF_VAL)      'Obtains a response result from the external
                            'device (Sends 03).
   IF JF_VAL = TRUE THEN    'Receives data, if the response is OK (TRUE).
      CALL ndVset(3)        'Receives 10 data from the external device (Sends 03).
      CALL ndVdt(pacPOS,1)  'Assigns the data received from the external device
                            'to variable (P1).
      CALL ndVdt(pacJNT,1)  'Assigns the data received from the external device
                            'to variable (J1).
      CALL ndVdt(pacTRN,1)  'Assigns the data received from an external device
                            'to variable (T1).
   END IF
END
```

ndVis (Library)

Function

Transfers a designated 2-digit integer to an external device (RS232C input/output).

Format

ndVis (<2 digit integer>)

Explanation

This statement provides a similar function to a VIS instruction in the conventional language.
The robot transfers a designated 2-digit integer to an external device after checking the preparation status of the external device.

Related Terms

ndVcom, ndVType

Example

```
#include <pacman.h>
PROGRAM PRO1
DEFINT JF_VAL = 0
   FLUSH
   CALL ndVType(1)          'Sets the protocol (Conventional = 0/ New = 1))
   CALL ndVrst              'Initializes the external device.
   CALL ndVset(0)           'Clears the received data (VDT) to 0.
   CALL ndVis(3)            'Starts up the external device (Sends 03).
   CALL ndJf(3,JF_VAL)      'Obtains a response result from the external device
                            '(Sends 03).)
   IF JF_VAL = TRUE THEN    'Receives data, if the response is OK (TRUE).
      CALL ndVset(3)        'Receives 10 data from the external device (Sends 03).
      CALL ndVdt(pacPOS,1)  'Assigns the data received from the external device
                            'to variable (P1).
      CALL ndVdt(pacJNT,1)  'Assigns the data received from the external device
                            'to variable (J1).
      CALL ndVdt(pacTRN,1)  'Assigns the data received from the external device
                            'to variable (T1).
      END IF
END
```

ndVput (Library)

Function

Transfers the position posture to an external device (RS232C input/output).

Format

ndVput (<Storage variable type>, <Storage variable number>)

Explanation

This statement provides a similar function to a VPUT instruction in the conventional language.
If <Position variable number> is negative:
The system transfers the robot's current position coordinates (the home position coordinates in the tool coordinate system when the tool is defined.) and a posture or contents of the position variable to an external device.
If <Position variable number> is positive:
The system transfers a storage variable type and contents of the position variable designated with a storage variable number, to the external device.

Macro Definition

Requires file <pacman.h>

Related Terms

ndVcom, ndVType

Example

```
#include <pacman.h>
PROGRAM PRO65
     FLUSH
     CALL   ndVType(1)         'Sets the protocol (Conventional = 0/ New = 1)
     CALL   nsVrst             'Initializes the external device.
     CALL   ndVis(4)           'Starts up the external device (Sends 03).
     P1 = (1,2,3,4,5,6,7)
     J1 = (11,12,13,14,15,16)
     T1 = (21,22,23,24,25,26,27,28,29,30)
     CALL   ndVput(pacPOS,1)   'Sends the data of variable "P1"
     CALL   ndVput(pacJNT,1)   'Sends the data of variable "J1"
     CALL   ndVput(pacTRN,1)   'Sends the data of variable "T1"
END
```

ndVrst (Library)

Function

Initializes an external device (RS232C input/output).

Format

ndVrst

Explanation

This statement provides a similar function to a VRST instruction in the conventional language.
It also directs the external device to initialize.

Related Terms

ndVcom, ndVType

Example

```
#include <pacman.h>
PROGRAM PRO1
DEFINT JF_VAL = 0
   FLUSH
   CALL ndVType(1)            'Sets the protocol (Conventional = 0/ New = 1)
   CALL ndVrst                'Initializes the external device.
   CALL ndVset(0)             'Clears the received data (VDT) to 0.
   CALL ndVis(3)              'Starts up the external device (Sends 03).
   CALL ndJf(3,JF_VAL)        'Obtains a response result from the external device
                              '(Sends 03).
   IF JF_VAL = TRUE THEN      'Receives data, if the response is OK (TRUE).
      CALL  ndVset(3)         'Receives 10 data from the external device (Sends 03).
      CALL  ndVdt(pacPOS,1)   'Assigns the data received from the external device
                              'to a variable (P1).
      CALL  ndVdt(pacJNT,1)   'Assigns the data received from the external device
                              'to variable (J1).
      CALL  ndVdt(pacTRN,1)   'Assigns the data received from the external device
                              'to variable (T1).
      END IF
END
```

ndVset (Library)

Function

Receives data from an external device (RS232C input/output).

Format

ndVset (<2-digit integer>)

Explanation

This statement provides a similar function to a VSET instruction in the conventional language.
The system receives data from an external device and adds (assigns) it to an internal variable after the robot sends a designated 2-digit integer to the external device. If you designate <2-digit integer> to “0”, the system initializes an internal variable with “0”.

Related Terms

ndVcom, ndVType

Example

```
#include <pacman.h>
PROGRAM PRO1
DEFINTJF_VAL = 0
   FLUSH
   CALL ndVType(1)            'Sets the protocol (Conventional = 0/ New = 1)
   CALL ndVrst                'Initializes the external device.
   CALL ndVset(0)             'Clears the received data (VDT) to 0.
   CALL ndVis(3)              'Starts up the external device (Sends 03).
   CALL ndJf(3,JF_VAL)        'Obtains a response result from the external device
                              '(Sends 03).
   IF JF_VAL = TRUE THEN      'Receives data, if the response is OK (TRUE).
      CALL  ndVset(3)         'Receives 10 data from the external device (Sends 03).
      CALL  ndVdt(pacPOS,1)   'Assigns the data received from the external
                              'device to variable (P1).
      CALL  ndVdt(pacJNT,1)   'Assigns the data received from the external
                              'device to variable (J1).
      CALL  ndVdt(pacTRN,1)   'Assigns the data received from the external
                              'device to variable (T1).
      END IF
END
```

ndVType (Library)

Function

Designates the protocol for communication with an external device (RS232C output).

Format

ndVType (<2-digit integer>)

Explanation

This statement designates the communication protocol for ndVis, ndJf, ndVset, ndVrst, ndVput, and ndVcom. The initial value of each library is set in the new protocol.
(Old protocol = 0/new protocol = 1)

Related Terms

ndVis, ndJf, ndVset, ndVput, ndVcom

Example

```
#include <pacman.h>
PROGRAM PRO1
DEFINTJF_VAL = 0
   FLUSH
   CALL ndVType(1)            'Sets the protocol (Conventional = 0/ New = 1)
   CALL ndVrst                'Initializes the external device.
   CALL ndVset(0)             'Clears the received data (VDT) to 0.
   CALL ndVis(3)              'Starts up the external device (Sends 03).
   CALL ndJf(3,JF_VAL)        'Obtains a response result from the external device
                              '(Sends 03).
   IF JF_VAL = TRUE THEN      'Receives data, if the response is OK (TRUE).
      CALL  ndVset(3)         'Receives 10 data from the external device (Sends 03).
      CALL  ndVdt(pacPOS,1)   'Assigns the data received from the external
                              'device to variable (P1).
      CALL  ndVdt(pacJNT,1)   'Assigns the data received from the external
                              'device to variable (J1).
      CALL  ndVdt(pacTRN,1)   'Assigns the data received from the external
                              'device to variable (T1).
      END IF
END
```

ndVcom (Library)

Function

Communication with an external device (kernel) (RS232C input/output)

Format

ndVcom (<Function code>, <Array variable>)

Explanation

This is a kernel program for communication with an external equipment and is required for using the communication libraries (ndVis, ndJf, ndVset, ndVrst, ndVput, and ndVType).

For Controller System Version 1.2* or earlier, use ndVcom contained in the version 1.2 compatible class. If in Version 1.2* or earlier ndVcom not in the version 1.2 compatible class but in classes 1 to 6 is used, a compilation error will result.

To check the controller software version, refer to the SETTING-UP MANUAL, Chapter 5, Section 5.7, [F6 Set]–[F6 Maint.]–[F2 Version].

Note: This program is not automatically executed by multitasking operation.
If you execute this by multitasking operation, you need to change this for synchronization using a semaphore etc.

Related Terms

ndVis, ndJf, ndVset, ndVrst, ndVput, ndVdt, ndVType

ndApra (Library)

Function

Performs absolute operation with the tool coordinate system specified (exclusively for four-axis robots).

Format

ndApra (<reference position> <Z coordinate> <stopping accuracy>)

Explanation

Provides a function similar to that of the APRA instruction in the conventional language.

Performs PTP movement by specifying the Z coordinate of the reference position only. The operation varies with the specified stopping accuracy as follows:

0: End operation

1: Path operation

2: Checking arrival at the target position by means of the encoder value

To use a function similar to that of the APRT instruction in the conventional language, calculate the position type value by translating deviation calculation (see 7.9.7 Position Calculation) by referring to this library and use the MOVE instruction.

Related Terms

NdDepa ndAprt ndDrwt

Example

```
PROGRAM PRO1
  TAKEARM
  CALL  ndApra((P0),400.0,1) ' Moves to 400 in Z coordinate at XP0.
  MOVE P,@0 P0               ' Moves to P0.
  CALL  ndDepa(400.0,0)      ' Moves to 400 in Z coordinate at P0.
  GIVEARM
END
```

ndDepa (Library)

Function

Performs absolute operation with the tool coordinate system specified (exclusively for four-axis robots).

Format

ndDepa (< coordinate> <stopping accuracy>)

Explanation

Provides a function similar to that of the DEPA function in the conventional language.

Performs PTP movement from the current position by specifying only the Z coordinate. The operation varies with the stopping accuracy as follows:

0: End operation

1: Path operation

2: Checking arrival at the target position by means of the encoder value

To use a function similar to that of the DRET instruction in the conventional language, calculate the position type value by translating deviation calculation (see 7.9.7 Position Calculation) by referring to this library and use the MOVE instruction.

Related Terms

ndApra

Example

```
PROGRAM PRO1
  TAKEARM
  CALL  ndApra((P0),400.0,1) ' Moves to 400 in Z coordinate at XP0.
  MOVE P,@0 P0               ' Moves to P0.
  CALL  ndDepa(400.0,0)      ' Moves to 400 in Z coordinate at P0.
  GIVEARM
END
```

4 Palletizing

Describes the libraries for palletizing functions.

pltDecCnt (Library)

Function

Decreases the count of the total palletizing counter.

Format

pltDecCnt (<Palletizing number>)

Explanation

The total palletizing counter, designated with a palletizing number, decreases by one (-1).
By editing the palletizing counter, you can palletize a hounds tooth pattern or an arbitrary step pattern.

Related Terms

pltGetCnt, pltIncCnt, pltKernel, pltLetCnt

Example

```
CALL pltDecCnt(1)           'Decreases total palletizing counter No. 1 by
                            'one (-1).
```

pltGetCnt (Library)

Function

Obtains the total palletizing counter.

Format

pltGetCnt (<Palletizing number>, <Cnt save integer variable number>)

Explanation

Obtains the total palletizing counter of a designated palletizing number.

Related Terms

pltDecCnt, pltIncCnt, pltKernel, pltLetCnt

Example

```
CALL pltGetCnt(1, 0)        'Assigns the current counter to I[0].
CALL pltLetCnt(1, I[0] - 1) 'Decreases the counter by 1.
```

pltGetK (Library)

Function

Obtains palletizing set value K.

Format

pltGetK (<Palletizing number>, <K save integer variable number>)

Explanation

This statement obtains palletizing set value K of a designated palletizing number.

Related Terms

pltGetK1, pltLetK, pltKernel

Example

```
CALL pltGetK(1,10)          'Assigns palletizing value K of palletizing No. 1
                            'to integer variable 10.
```

pltGetK1 (Library)

Function

Obtains palletizing counter K1.

Format

pltGetK1 (<Palletizing number>, <K1 save integer variable number>)

Explanation

This statement obtains palletizing counter K1 of a designated palletizing number.

Related Terms

pltGetK, pltLetK1, pltKernel

Example

```
CALL pltGetK1(1,10)         'Assigns palletizing counter K1 of palletizing
                            'No. 1 to integer variable 10.
```

pltGetM (Library)

Function

Obtains palletizing set value M.

Format

pltGetM (<Palletizing number>, <M save integer variable number>)

Explanation

This statement obtains palletizing set value M of a designated palletizing number.

Related Terms

pltGetM1, pltLetM, pltKernel

Example

```
CALL pltGetM(1,10)          'Assigns palletizing value M of palletizing No. 1
                            'to integer variable 10.
```

pltGetM1 (Library)

Function

Obtains palletizing counter M1.

Format

pltGetM1 (<Palletizing number>, <M1 save integer variable number>)

Explanation

This statement obtains palletizing counter M1 of a designated palletizing number.

Related Terms

pltGetM, pltLetM1, pltKernel

Example

```
CALL pltGetM1(1,10)         'Assigns palletizing counter M1 of palletizing
                            'No. 1 to integer variable 10.
```

pltGetN (Library)

Function

Obtains palletizing set value N.

Format

pltGetN (<Palletizing number>, <N save integer variable number>)

Explanation

This statement obtains palletizing set value N of a designated palletizing number.

Related Terms

pltGetN1, pltLetN, pltKernel

Example

```
CALL pltGetN(1,10)           'Assigns palletizing value N of palletizing No. 1
                             'to integer variable 10.
```

pltGetN1 (Library)

Function

Obtains palletizing counter N1.

Format

pltGetN1 (<Palletizing number>, <N1 save integer variable number>)

Explanation

Obtains palletizing counter N1 of a designated palletizing number.

Related Terms

pltGetN, pltLetN1, pltKernel

Example

```
CALL pltGetN1(1,10)          'Assigns palletizing counter N1 of palletizing
                             'No. 1 to integer variable 10.
```

pltGetNextPos (Library)

Function

Obtains the next position.

Format

pltGetNextPos (<Palletizing number>, <Next position obtaining position type variable>)

Explanation

This statement obtains the next position. When this library is executed, the palletizing counter increases.
Therefore, when you obtain the next position but do not execute it, you must decrease the pltDecCnt and so on.

Related Terms

pltMove, pltKernel

Example

```
CALL pltGetNextPos(1,10)     'Assigns a position immediately following
                             'palletizing No. 1 to integer variable 10.
```

pltGetPLT1END (Library)

Function

Obtains a palletizing 1-row completion flag.

Format

pltGetPLT1END (<Palletizing number>, <1-row completion flag save integer number>)

Explanation

Stores a palletizing 1-row completion flag into I[<1-row completion flag save integer number>].

Related Terms

pltGetPLTEND, pltResetPLT1END, pltKernel

Example

```
CALL pltGetPLT1END(1,10)     'Assigns a 1-row completion flag of palletizing
                             'No. 1 to integer variable 10.
```

pltGetPLTEND (Library)

Function

Obtains a palletizing all-row completion flag.

Format

pltGetPLTEND (<Palletizing number>, <All row-completion flag save integer number>)

Explanation

This statement stores a palletizing all row completion flag into I[<All-row completion flag save integer number>].

Related Terms

pltGetPLT1END, pltResetPLTEND, pltKernel

Example

```
CALL pltGetPLTEND(1,10)     'Assigns an all-row completion flag of palletizing
                            'No. 1 to integer variable 10.
```

pltIncCnt (Library)

Function

Increases the total palletizing counter.

Format

pltIncCnt (<Palletizing number>)

Explanation

This statement increases the total palletizing counter of a designated palletizing number by +1 from the current value.
By editing the palletizing counter, you can palletize a hounds tooth pattern or arbitrary step pattern.

Related Terms

pltDecCnt, pltGetCnt, pltKernel, pltLetCnt

Example

```
CALL pltIncCnt(1)           'Increases the total palletizing counter of
                            'palletizing No. 1 by one (+1).
```

pltInit1 (Library)

Function

Palletizing initialization template 1.

Format

pltInit1

Explanation

This is an example of pltInitialize.

For the meanings of arguments, refer to the arguments used with the pltInitialize commands.

Related Terms

pltInitialize

pltInitialize (Library)

Function

Palletizing initialization

Format

pltInitialize (<Palletizing number>, <Lateral partition number>, <Longitudinal partition number>, <Pile number>, <Approach length>, <Depart length>, <Pallet height>, <Pallet 4-corner type 1P variable number>, <Pallet 4-corner type 2P variable number>, <Pallet 4-corner type 3P variable number>, <Pallet 4-corner type 4P variable number>)

Explanation

This statement is an initialization program for defining palletizing motion.
It is more effective to use this program rather than the pltResetAll Library, if you initialize only the palletizing counter.

Note: Without execution of pltInitialize, no palletizing motion will be executed.

Macro Definition

mcApprVal	:	F type variable number to assign the approach length.
McDepVal	:	F type variable number to assign the depart length.
McPltH	:	A type F variable number to assign the pallet height.

Note: If these definitions compete with the user's variable number, define the macro again.

Related Terms

pltKernel

Example

```
CALL pltInitialize(0,4,3,1,50,50,50,52,53,54,55)
                  'Initializes palletizing parameters; Palletizing number:0,
                  'Lateral partition number:4, Longitudinal partition number:3
                  'Pile number:1, Approach length:50 mm, Depart length:50 mm,
                  'Pallet height:50 mm, Pallet 4-corner type 1P:52, 2P:53,
                  '3P:54, 4P:55.
```

pltKernel (Library)

Function

Palletizing motion (kernel)

Format

pltKernel (<Palletizing number>, <Action number>, <Action argument>, <Error number>)

Explanation

This statement is a kernel program for palletizing motion. This library is required to use all other palletizing libraries.

Note: This program is not automatically executed by multitasking operation. If you execute this by multitasking, you need to change this for synchronization using a semaphore.

Macro Definition

mcPaltMax: Maximum pallet number
You can easily increase pallets in number by defining this macro again.

Note: If you increase the number of pallets, the system consumes the local variable area.

Related Terms

pltInitialize

Remarks

- Only other libraries call this library. The user does not directly call this.
- For directly using this, you should fully understand the contents of the program.

pltLetCnt (Library)

Function

Sets the total palletizing counter.

Format

pltLetCnt (<Palletizing number>, <All counter setting values>)

Explanation

This statement can set the total palletizing counter of a designated palletizing number.
By editing the palletizing counter, you can palletize a hounds tooth pattern or arbitrary step pattern.

Related Terms

pltDecCnt, pltGetCnt, pltIncCnt, pltKernel

Example

```
CALL pltLetCnt(1,12)        'Sets the total palletizing counter of palletizing
                            'No. 1 to 12.
```

pltLetK1 (Library)

Function

Sets the palletizing counter K1.

Format

pltLetK1 (<Palletizing number>, <K1 setting value>)

Explanation

This statement can set the counter K1 of a designated palletizing number.
By editing the palletizing counter, you can palletize a hounds tooth pattern or arbitrary step pattern.

Related Terms

pltGetK1, pltKernel

Example

```
CALL pltLetK1(1,1)     'Sets palletizing counter K1 of palletizing No. 1 to 1.
```

pltLetM1 (Library)

Function

Sets the palletizing counter M1.

Format

pltLetM1 (<Palletizing number>, <M1 setting value>)

Explanation

This statement can set the counter M1 of a designated palletizing number.
By editing the palletizing counter, you can palletize a hounds tooth pattern or arbitrary step pattern.

Related Terms

pltGetM1, pltKernel

Example

```
CALL pltLetM1(1,4)     'Sets palletizing counter M1 of palletizing No. 1 to 4.
```

pltLetN1 (Library)

Function

Sets the palletizing counter N1.

Format

pltLetN1 (<Palletizing number>, <N1 setting value>)

Explanation

This statement can set the counter N1 of a designated palletizing number.
By editing the palletizing counter, you can palletize a hounds tooth pattern or arbitrary step pattern.

Related Terms

pltGetN1, pltKernel

Example

```
CALL pltLetN1(1,3)          'Sets palletizing counter N1 of palletizing
                            'No. 1 to 3.
```

pltMain1 (Library)

Function

Palletizing template 1

Format

pltMain1

Explanation

This statement is the basic work statement to pick up a part and move it to the assembling position.

Macro Definition

pltIndex palletizing number

Related Terms

pltMain2

Remarks

You may modify the template to meet your needs.

pltMain2 (Library)

Function

Palletizing template 2

Format

pltMain2

Explanation

This statement is the basic work statement to pick up a part and move it to the assembling position. Before picking up a part, the robot checks that the previous pick-up was correctly done.

Macro Definition

pltIndex	:	palletizing number
ChuckNG	:	pick-up NG signal (DIO number)

Related Terms

pltMain1

Remarks

You may modify the template to meet your needs.

pltMove (Library)

Function

Standard palletizing template 1

Format

pltMove (<Palletizing number>)

Explanation

This statement executes standard palletizing, as defined with pltInitialize. You can insert a bypass point according to comments.

For Controller System Version 1.2* or earlier, use ndVcom contained in the version 1.2 compatible class. If in Version 1.2* or earlier ndVcom not in the version 1.2 compatible class but in classes 1 to 6 is used, a compilation error will result.

To check the controller software version, refer to the SETTING-UP MANUAL, Chapter 5, Section 5.7, [F6 Set]–[F6 Maint.]–[F2 Version].

Macro Definition

mcNextPos	:	A type P variable number to store the next position
mcApprLen	:	A type F variable number to store the approach length
mcDepLen	:	A type F variable number to store the depart length

Note: If these definitions compete with the user's variable number, define the macro again.

Related Terms

pltInitialize, pltMove0, pltKernel

Remarks

The palletizing counter counts when the system obtains the next position. Therefore, when the system obtains but does not operate, you need to use pltDecCnt to decrease the counter.

Example

```
CALL pltMove(1)        'Carries out palletizing motion of palletizing No. 1.
```

pltMove0 (Library)

Function

Standard palletizing motion 1

Format

pltMove0

Explanation

This statement is a specialized program to Index=0 of pltMove (Index).

If you change the motion of each pallet, add this program to change the line of #DEFINE pltIndex 0 to the pallet number and then change the program name.

For Controller System Version 1.2* or earlier, use ndVcom contained in the version 1.2 compatible class. If in Version 1.2* or earlier ndVcom not in the version 1.2 compatible class but in classes 1 to 6 is used, a compilation error will result.

To check the controller software version, refer to the SETTING-UP MANUAL, Chapter 5, Section 5.7, [F6 Set]–[F6 Maint.]–[F2 Version].

Macro Definition

mcNextPos	:	A type P variable number to store the next position
mcApprLen	:	A type F variable number to store the approach length
mcDepLen	:	A type F variable number to store the depart length

Note: If these definitions compete with the user's variable number, define the macro again.

PltIndex	:	pallet number

Related Terms

pltInitialize, pltMove, pltKernel

Remarks

The palletizing counter counts when the system obtains the next position. Therefore, when the system obtains but does not operate, you need to use pltDecCnt to decrease the counter.

Example

```
CALL pltMove0          'Carries out palletizing motion of palletizing No. 0.
```

pltResetAll (Library)

Function

Resets all palletizing counters.

Format

pltResetAll (<Palletizing number>)

Explanation

This statement resets all palletizing counters.
This has almost the same meaning as pltLetCnt (<Palletizing number>, 0). However, this library also resets the row completion flag.
When redefinition of palletizing motion is not required and only a counter is reset, use this library. Do not use pltInitialize.

Related Terms

pltInitialize, pltResetPLTEND, pltKernel

Example

```
CALL pltResetAll(1)    'Resets all palletizing counters of palletizing No. 1.
```

pltResetPLT1END (Library)

Function

Resets a palletizing 1-row completion flag.

Format

pltResetPLT1END (<Palletizing number>)

Explanation

This statement resets a palletizing 1-row completion flag.

Related Terms

pltGetPLT1END, pltResetPLTEND, pltKernel

Example

```
CALL pltResetPLT1END(1)     'Resets a palletizing 1-row completion flag of
                            'palletizing No. 1.
```

pltResetPLTEND1 (Library)

Function

Resets a palletizing all-row completion flag.

Format

pltResetPLTEND1 (<Palletizing number>)

Explanation

This statement resets a palletizing all-row completion flag.

Related Terms

pltGetPLTEND, pltResetPLT1END, pltKernel

Example

```
CALL pltResetPLT1END1(1)     'Resets a palletizing all-row completion flag of
                             'palletizing No. 1.
```

5 Tool Operation

Describes the libraries for tool operation.

tolChange (Library)

Function

Tool change

Format

tolChange (<Tool loading number>, <Tool number>)

Explanation

This library executes the standard tool change motion, as defined in tolInitialize.

Related Terms

tolInitialize, tolKernel

Remarks

The tool coordinate system change instruction in PAC (ChangeTool) is not to practically change the tool position and not to load the tool but only to change the tool coordinate system.

Example

```
CALL tolChange(1,2)         'Assigns tool No.2 to tool load number 1.
```

tolInit1 (Library)

Function

Tool change initialization template 1

Format

tolInit1

Explanation

This is an example of tolInitialize.
For the meanings of arguments, refer to the arguments used with the tolInitialize commands.

Related Terms

tolInitialize, tolKernel

tolInitialize (Library)

Function

Tool change initialization

Format

tolInitialize (<Tool loading number>, <Tool number>, <Chuck point type P variable number>, <Approach length>, <Chuck DIO number>, <Unchuck DIO number>)

Explanation

This statement defines the standard tool change motion.

Related Terms

tolInit1, tolKernel

Example

```
CALL tolInitialize(0,1,50,50,40,41)  'Tool loading number:0, Tool number:1,
                                     'Chuck point type:P50, Approach length:50mm
                                     'Chuck DIO number:40, Unchuck DIO number:41
```

tolKernel (Library)

Function

Tool change motion (kernel)

Format

tolKernel (<Tool loading number>, <Tool number>, <Action number>, <Action argument>)

Explanation

This statement is a kernel program for tool change. This program is required to use all other tool change libraries.

Macro Definition

mcToolMax	:	maximum number of tools.
McTools:	:	maximum available number of tools for multiple hands.

Related Terms

tolInitialize

tolMain1 (Library)

Function

Tool change template 1

Format

tolMain1

Explanation

This statement is an example of tool change.
When a tool number is designated in 8-bit number from ioINB, the tool is changed to the designated one.

Macro Definition

ioINB : The first number of a tool number designation IO

Related Terms

tolChange, tolInitialize, tolKernel

6 Input/Output

Describes the libraries for DIO and RS232C input/output.

dioSync (Library)

Function

Synchronizes with an external device (such as a sequencer) connected to DIO.

Format

dioSync (<Data receive permission signal number>, <Data receive finish signal number>)

Explanation

This statement is the standard procedure used to synchronize with an external device (such as a sequencer) connected to DIO.

Example

```
CALL dioSync(220,221)       'Synchronizes with the connected external device
                            'by assigning a data receive permission signal and
                            'data receive finish signal to IO220 and IO221,
                            'respectively.
```

7 Arm Movement

Describes the libraries for arm motion.

mvResetPulseWidth (Library)

Function

Restores default encoder pulse counts for positioning allowance.

Format

mvResetPulseWidth

Explanation

Restores default encoder pulse counts (20) for positioning allowance for all axes.

An encoder pulse count for positioning allowance refers to convergence accuracy for a joint which is applied at execution of a motion command with @E option.

Macro Definition

Requires file <pacman. h>

Example

```
CALL mvResetPulseWidth      'Restores default encoder pulse counts for all axes.
```

mvResetPulseWidthJnt (Library) (Version 1.7 or later)

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.
```

mvResetTimeOut (Library)

Function

Restores the default motion finish timeout value.

Format

mvResetTimeOut

Explanation

This statement sets the value of the movement finish timeout to a default value of 5600 (millisec.) (5.6 sec.).
The movement finish timeout is the limit value of the time which is taken for the axis to enter the stop permissible pulse width, when the axis moves during encoder value checking motion (@E designation).
Even if the axis does not enter the permissible pulse width within the limit value, the alarm “6651 check instruction time over” will be displayed.

Macro Definition

Requires file <pacman. h>

Example

```
CALL mvResetTimeOut          'Restores the default motion finish timeout value.
```

mvReverseFlip (Library)

Function

4-axis figure reverse

Format

mvReverseFlip (<Base position>, <No check position>, <Decision base value>)

Explanation

The system compares the 4-axis value of <Reference position> with that of <Non inspection position>. This statement reverses FLIP/NONFLIP if the 4-axis value exceeds <Determination reference value>.

Example

```
CALL mvReverseFlip(P0,P1,90) 'Compares base position P0 with non-inspection
                             'position P1. If the absolute difference exceeds 90,
                             'the system reverses 4-axis figure.
```

mvSetPulseWidth (Library)

Function

Sets the permissible stop pulse width.

Format

mvSetPulseWidth (<Stop permissible pulse width J1>, <Stop permissible pulse width J2>, < Stop permissible pulse width J3>, < Stop permissible pulse width J4>, < Stop permissible pulse width J5>, < Stop permissible pulse width J6>)

Explanation

This statement sets the permissible stop pulse width.
The permissible stop pulse width is an error pulse used to judge that the motor has stopped when the controller confirms the encoder value (by designation of @E).

Macro Definition

Requires file <pacman. h>

Example

call mvSetPulseWidth (10 10 10 10 10 10) sets 10 pulses to stop accuracy range for all axes.

Notes

Set the pulse width to 1 or more. If 0 is designated, the default value (20) is designated. If a negative value is designated, the error “6003 a value exceeds a effective value zone” will be displayed. If a smaller value is designated, the error “6651 check instruction time over” will be displayed.

mvSetPulseWidthJnt (Library) (Version 1.7 or later)

Function

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

Format

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

Explanation

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 Terms

`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.
```

mvSetTimeOut (Library)

Function

Sets the timeout value for movement finish.

Format

mvSetTimeOut (<Movement finish timeout>)

Explanation

This statement sets the timeout value for movement finish (unit□ms).
The movement finish timeout is the limit value of the time which is taken for an axis to enter the permissible pulse width to stop when the axis moves during encoder value checking motion (@E designation).
Even if the axis does not enter the permissible pulse width within the limit value, the alarm “6651 check instruction time over” will be displayed.

Macro Definition

Requires file <Pacman.h>

Example

call mvSetTimeOut (3000) Sets the timeout time to 3 (s).

Notes

Set the timeout value to 1 or more. If 0 is designated, the default value (5600) is designated. If a negative value is designated, the error “6003 A value exceeds an effective value zone” will be displayed. If a smaller value is designated, the error “6651 check instruction time over” will be displayed.

SetGravity (Library) (Version1.2 or later)

Function

Compensates for the static load (gravity torque) applied to each joint and attains balance with gravity torque.

Format

SetGravity

Explanation

Each joint of the robot undergoes downward static load (gravity torque) due to earth gravity. The effects of the gravity torque will vary depending upon the mass of payload (end-effector and workpiece), the payload center of gravity, and robot figures.

If you limit the motor output torque by setting its drive current limit so that the limited torque becomes lower than the gravity torque, then the robot will move down towards the earth. To prevent it, this statement compensates the limited torque for the gravity torque, keeping the balance of torque.

Macro Definition

Requires <pacman.h> file.

Related Terms

SetCurLmt, ResetGravity, SetGrvOffset

Notes

(1) Write this command in a TAKEARMed task that has obtained arm-semaphore. If this command is executed without arm-semaphore obtained, Error 21F7 will result.

(2) Set the mass of payload and the payload center of gravity accurately. Otherwise, the robot may move down due to gravity if you set a low current limit value (e.g., less than 30). For the entry procedure of the mass of payload and the payload center of gravity, refer to p.4-15, " 4.7 Setting the Master Control Parameters in User Preferences."

(3) If you do not know the accurate mass of payload or its center of gravity or if the robot moves down in spite of accurate settings, then use the gravity offset function (SetGrvOffset) that compensates for the gravity compensation value.

(4) If you set the gravity offset setting to "1" on the teach pendant, the gravity offset function becomes enabled. The setting made on the teach pendant will take effect immediately following the completion of calibration after the robot controller is powered on.

Example

```
CALL SetGravity        'Enables gravity compensation function.
Delay 100              'Waits for gravity compensation to take effect.
CALL SetCurLmt (2,30)  'Sets the current limit value of the 2nd axis to 30%.
```

ResetGravity (Library) (Version 1.2 or later)

Function

Disables the balance setting between the limited motor torque and gravity torque, which is made with SetGravity.

Format

ResetGravity

Explanation

This command disables the balance setting between the motor torque limited by the current limit function and gravity torque.

Macro Definition

Requires <pacman.h> file.

Related Terms

ResetCurLmt, SetEralw

Notes

(1) Write this command in a TAKEARMed task that has obtained arm-semaphore. If this command is executed without arm-semaphore obtained, Error 21F7 will result.

(2) If this command is executed when the current limit function is enabled, Error 665b will result. Disable the current limit function and then try it again.

(3) If you set the gravity offset setting to "0" on the teach pendant, the gravity offset function becomes disabled. If you do it when the current limit function is enabled, Error 665b will result, as in step (2).

Example

```
CALL ResetGravity
```

SetGrvOffset (Library) (Version1.2 or later)

Function

Compensates the torque of each joint programmed with SetGravity for gravity torque.

Format

SetGrvOffset

Explanation

Each joint of the robot undergoes downward static load (gravity torque) due to earth gravity. Although gravity compensation command SetGravity allows you to adjust the balance between the limited torque and gravity torque, the balance may be off-balance due to the difference between the mass of payload you set and the actual one.

This offset function presumes the gravity torque when the robot is on halt and calculates the gravity offset value.

Macro Definition

Requires <pacman.h> file.

Related Terms

SetCurLmt, SetGravity, ResetGrvOffset

Notes

(1) Write this command in a TAKEARMed task that has obtained arm-semaphore. If this command is executed without arm-semaphore obtained, Error 21F7 will result.

(2) This command should be executed when the motor power is on and the robot is on halt. If it is executed when the motor power is off, Error 6006 will result. If it is executed when the robot is in motion, Error 600B will result.

(3) If the robot attitude is greatly changed after execution of this command, execute this command again.

(4) If the current limit reset value in User Preferences is set to any value other than “1”, “3”, “5”, or “7”, the compensation value will be reset to “0” when you turn on the motor power.

Example

```
CALL SetGrvOffset
```

ResetGrvOffset (Library) (Version 1.2 or later)

Function

Disables the gravity offset function.

Format

ResetGrvOffset

Explanation

Disables the gravity offset function which has been enabled with SetGrvOffset.

Macro Definition

Requires <pacman.h> file.

Related Terms

SetGrvOffset

Notes

(1) Write this command in a TAKEARMed task that has obtained arm-semaphore. If this command is executed without arm-semaphore obtained, Error 21F7 will result.

(2) This command should be executed when the robot is on halt. If it is executed when the robot is in motion, Error 600B will result. This command is executable even when the motor power is off.

Example

```
CALL ResetGrvOffset
```

SetCurLmt (Library) (Version 1.2 or later)

Function

Sets the limit of motor current to be applied to the specified axis.

Format

SetCurLmt (<AxisNumber>, <Value>)

Explanation

Limits the value of motor current (torque) to be applied to the axis specified by <AxisNumber> to the value specified by <Value>. This command is useful when you want to limit torque that a workpiece will undergo during insertion or butting jobs.

The maximum value of <Value> is 100 which refers to the motor rating current. If any value exceeding the allowable limit for each axis is specified, the value will be automatically limited to that allowable limit.

Set a value of 1 or above. If 0 or a negative number is set, Error 6003 will result.

Macro Definition

Requires <pacman.h> file.

Related Terms

ResetCurLmt, SetGravity, SetGrvOffset, SetEralw

Notes

(1) When the motor current is limited with SetCurLmt, the robot cannot move at the maximum speed or acceleration. Use SetCurLmt only at steps that need the current limit. When using SetCurLmt, decrease the acceleration.

(2) If a workpiece bumps against something at high speed even if the driving force is controlled by limiting the motor current, the impact is considerable due to the inertia of the workpiece, end-effector and axis. Set the current limit just before the workpiece comes into contact with the object and reduce the speed.

(3) Set the current limit when the robot is on halt. If it is set during a pass motion, an error is likely to occur.

(4) Write this command in a TAKEARMed task that has obtained arm-semaphore. If this command is executed without arm-semaphore obtained, Error 21F7 will result.

(5) When setting the current limit, be sure to enable the gravity compensation function. If the current limit is set when the gravity compensation function is disabled, Error 665a will result. For the gravity compensation function, refer to SetGravity.

6-axis (6) If the current limit reset value in User Preferences is set to any value other than"1," the current limit will be reset when you turn on the motor power. To make the current limit function effective immediately after switching on the motor power, set the current limit reset value to "1."

6-axis (7) Set the mass of payload and the payload center of gravity accurately. Otherwise, the robot may move down due to gravity if you set a low current limit value (e.g., less than 30). For the entry procedure of the mass of payload and the payload center of gravity, refer to PART 1, Section 5.7, "Setting the Master Control Parameters in User Preferences."

6-axis (8) If the current limit reset value is set to "1," the robot might move down due to gravity the moment you turn on the motor power. Reset the current limit by executing ResetCurLmt when the motor power is off and then switch on the motor power.

6-axis (9) If you do not know the accurate mass of payload or its center of gravity or if the robot moves down in spite of accurate settings, then use the gravity offset function (SetGrvOffset) that compensates for the gravity compensation value.

Example

6-axis

```
CALL SetGravity        'Enables the gravity offset

CALL SetGrvOffset      'Compensates the gravity offset value

CALL SetEralw (2, 20)  'Sets the allowable deviation of the 2nd axis
                       to '20 degree

CALL SetCurLmt (2, 30) 'Sets the current limit of the 2nd axis to
    30%
```

4-axis

```
CALL SetEralw (2, 20)  'Sets the allowable deviation of the 2nd
    axis
                       'to 20 degree

CALL SetCurLmt (2, 30) 'Sets the current limit of the 2nd axis to
    30%
```

ResetCurLmt (Library) (Version 1.2 or later)

Function

Resets the motor current limit of the specified axis.

Format

ResetCurLmt (<AxisNumber>)

Explanation

`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

Requires <pacman.h> file.

Related Terms

SetCurLmt, ResetEralw

Notes

(1) When resetting the current limit, this command carries out deviation elimination process. If there is angle deviation due to external force, the time required for deviation elimination process will vary depending upon the set speed and acceleration. To shorten the time, set higher speed and acceleration.

(2) The command can be executed even when the motor power is off. For resetting the current limit when motor power is off, run the following program after finishing the task that is obtaining 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)          'Resets the current limit of all the axes.
```

SetEralw (Library) (Version 1.2 or later)

Function

Modifies the allowable deviation of the specified axis.

Format

SetEralw (<AxisNumber>, <Value>)

Explanation

Sets the allowable deviation of the axis specified by <AxisNumber>. Use this command if angle deviation occurs due to external force when the current limit function is enabled.

The <Value> is the arm joint angle and specified in degrees.

"Allowable deviation value" refers to the allowable range of the "Error 611*J* Excessive deviation" which will occur for safety if the servo deviation exceeds the specified value.

During assembling operation with the current limit enabled, if servo deviation occurs due to external force, the above error may occur. To avoid this, you may use this command temporarily to increase the allowable deviation value.

This command can also be used to reduce the allowable deviation value for helping quick detection of the downward movement of the robot due to gravity when the current limit function is enabled.

Macro Definition

Requires <pacman.h> file.

Related Terms

SetCurLmt, ResetEralw

Notes

(1) Run this command in a TAKEARMed task which has obtained arm-semaphore. If the command is executed without arm-semaphore obtained, Error 21F7 will result.

Example

```
CALL SetEralw(2,20)         'Sets the permissible deviation of the 2nd axis
                            'to 20 degrees.
```

ResetEralw (Library) (Version 1.2 or later)

Function

Resets the allowable deviation value of the specified axis to the initial value.

Format

ResetEralw(<AxisNumber>)

Explanation

Resets the allowable deviation value of the axis specified by <AxisNumber> to the initial value.
If <AxisNumber> is specified to "0," the allowable deviation values of all the axes will be reset.

[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

Requires <pacman.h> file.

Related Terms

ResetCurLmt, 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) Run this command in a TAKEARMed task that has obtained arm-semaphore. If this command is executed without arm-semaphore obtained, Error 21F7 will result.

(3) Like this command, execution of ResetCurLmt will also reset the allowable deviation values to the initial values.

Example

```
CALL ResetEralw(0)          Returns the allowable deviation values of all axes
                            to initial values.
```

OnSrvLock (Library)

Function

Servo-locks a specified axis (exclusively for a four-axis robot).

Format

OnSrvLock (<specified axis>)

Explanation

Provides a function similar to that of the ON SVLOCK instruction in the conventional language.
Servo lock means that robot arms are controlled and their positions are held.

Macro Definition

The pacman.h file is necessary.

Related Term

OffSrvLock

Notes

Set servo lock as the robot stops. If it is set during path operation, an error may occur.

Execute this command on a task that obtains robot control conditions (TAKEARM). If robot control conditions are not obtained, error message "21F7 Arm semaphore cannot be fetched." is reported.

Example

```
CALL  OnSrvLock(1)          ' Servo lock for axis 1
CALL  OnSrvLock(0)          ' Servo lock for all axes
```

OffSrvLock (Library)

Function

Releases servo lock for the specified axis. (Exclusively for four-axis robots)

Format

OffSrvLock (<specified axis>)

Explanation

Provides a function similar to the OFF SVLOCK instruction in the conventional language.

Servo lock refers to the state where the robot arm is controlled to keep its position. When it is released, the robot arm is not kept in its position but moved by an external force applied to it.

Macro Definition

The <pacman.h> file is necessary.

Related Term

OnSrvLock

Notes

(1) No operation command can be executed for an axis for which servo lock is released.

(2) Set release of servo lock while the robot is in stopped state. If set during path operation, an error may result.

(3) Execute this command in the task obtaining the robot control right (TAKEARM). If the robot control right has not been obtained, an error "21F7 Arm semaphore cannot be obtained" will occur.

(4) If bit 2 of the value set for "25: Current limit reset" in the operating conditions is "0" (initial value), servo lock release is reset (to cause servo lock) upon motor power on. To validate servo lock release immediately after motor power on, set "+2" as the current limit reset value.

Note□Setting example of operating condition "25: Current limit reset"

	PWM	SVLock	Curlmt	
	*	*	*	
If only SVLock is effective	0	1	0	= 2
If all is effective	1	1	1	= 7

Example

```
CALL  OffSrvLock (1)    ' Releases servo lock for axis 1.
```

OnPWM (Library)

Function

Performs PWM switching on a specified axis (exclusively for a four-axis robot).

Format

OnPWM (<specified axis>)

Explanation

PWM switching means that robot arms are controlled and their positions are held.

Macro Definition

The pacman.h file is necessary.

Related terms

OffPWM

Notes

(1) Set PWM switching as the robot stops. If it is set during path operation, an error may occur.

(2) Execute this command on a task that obtains robot control conditions (TAKEARM). If robot control conditions are not obtained, error message "21F7 Arm semaphore cannot be fetched." is reported.

Example

```
CALL  OnPWM(1)' PWM switching on 1 axis
CALL  OnPWM(1)' PWM switching on all axes
```

OffPWM (Library)

Function

Releases PWM switching on the specified axis (exclusively for a four-axis robot).

Format

OffPWM (<specified axis>)

Explanation

PWM switching means that robot arms are controlled and their positions are held. When PWM switching is released, robot arm positions are not held, and they are displaced if external force is applied.

Macro Definition

The pacman.h file is necessary.

Related Terms

OnSrvLock

Notes

(1) Operation command cannot be executed on an axis with PWM switching released.

(2) Set PWM switching release as the robot stops. If it is set during path operation, an error may occur.

(3) Execute this command on a task that obtains robot control conditions (TAKEARM). If robot control conditions are not obtained, error message "21F7 Arm semaphore cannot be fetched." is reported.

(4) When bit 3 of the "25: current limitation reset" set value is set to 0 (initial value) as a use condition, PWM switching release is reset (PWM switching) at motor power-on. To validate PWM switching release just after the motor power is turned on, set +4 to the "current limitation reset" set value.

Note: Use condition "25: current limitation reset" setting example

	PWM	SVLock	Curlmt	
	*	*	*	
PWM only valid	0	1	0	= 2
All valid	1	1	1	= 7

Example

CALL OffPWM(1)' PWM switching release on axis 1

Setcycloid (Library)

Function

Causes transition to the cycloid operation mode to suppress overshoot and residual vibration at the end of PTP operation.

Format

Setcycloid

Explanation

In the cycloid operation mode, speed change during deceleration can be smoothed. Therefore, the overshoot and residual vibration upon stopping can be reduced.

The operation time increases slightly in the cycloid operation mode. Be sure to check the cycle time.

This mode is valid when another program is executed after Setcycloid execution. For resetting, either execute Resetcycloid or set to the automatic mode after setting the manual mode once.

Macro Definition

The <pacman.h> file is necessary.

Related Term

Resetcycloid

Example

```
CALL  Setcycloid' Causes transition to the cycloid operation mode.
```

SetCycloidJnt (Library) (Version 1.7 or later)

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.

Format

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

Explanation

(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 Terms

`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.
```

Resetcycloid (Library)

Function

Causes transition from the cycloid mode to the ordinary operation mode.

Format

Resetcycloid

Explanation

Causes transition from the cycloid operation mode to the ordinary operation mode. If it is desired to set the cycloid operation mode again, execute Setcycloid.

Macro Definition

The <pacman.h> file is necessary.

Related Terms

Setcycloid

Notes

(1) Execute this command in a task holding robot control conditions (TAKEARM). If robot control conditions are not held, an error message "21F7 Arm semaphore cannot be fetched." is reported.

Example

```
CALL  Resetcycloid     ' Resets the cycloid operation mode.
```

ResetCycloidJnt (Library) (Version 1.7 or later)

Function

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

Format

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

Explanation

`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 Terms

`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.
```

SetCPSpdMode (Library) (Version 1.8 or later)

Function

Keeps the TCP speed (Tool end speed in CP motion) constant or restores it to the default.

Format

```
SetCPSpdMode<SetValue>
```

Explanation

If you specify a CP motion involving the rotation of the robot hand, then the system will automatically decrease the TCP speed according to the rotation angle by default. Setting 1 to `<SetValue>` may keep the TCP speed constant as long as the rotation angle is less than the specified value.

Example

```
CALL SetCPSpdMode(0)    'Restore to the default.

CALL SetCPSpdMode(1)    'Set the TCP speed to"1."
```

Notes

If `<SetValue>` is set to `1`, the rotation speed will vary depending upon the robot motion in order to keep the TCP speed constant. If you specify such a motion that will exceed the rotation speed limit, then the system will issue a warning message and run the robot while decreasing the TCP speed.

SetForce_HM (Library)

Function

A current limitation library that specifies the thrust (unit: N) of the Z coordinate with the HM/HS robot.

Format

SetForce_HM (<thrust unit >)

Explanation

Limits the thrust by limiting the motor current on the Z axis. The unit is N (Newton). The maximum thrust extremely varies depending on the impact force generated at contact with peripherals. The allowable thrust is a value assumed when the impact force at collision is almost 0 (collision at too low speed). The thrust also varies depending on the Z-axis position by the mechanical friction; therefore, this value should be assumed to be a typical value (not a guaranteed value). Like the current limitation library (SetCurlmt), a value that exceeds the allowable value is limited to the allowable value.

Macro Definition

The <pacman.h> file is necessary.

Related terms

SetForce_HC, SetCurlmt, ResetCurlmt

Notes

(1) Since the motor current is limited during current limitation, operation is impossible at the highest speed and highest acceleration. For current limitation, use only the required steps. To use the current limitation, lower the acceleration.

(2) When the thrust is limited by the current limitation, if work collides at high speed, an impact force occurs due to the inertia between the work and hand. Set the current limitation before setting work, and lower the speed.

(3) Set the current limitation as the robot stops. If it is set during path operation, an error may occur.

(4) Execute this function on a task that obtains the robot control right (TAKEARM). If the robot control right is not obtained, error message "21F7 Arm semaphore cannot be fetched." is reported.

(5) When bit 1 of the "25: current limitation reset" set value is set to 0 (initial value) as a use condition, the servo lock release is reset (servo-lock) at motor power-on. To validate servo lock release just after the motor power is turned on, set +1 to the "current limitation reset" set value. For details on setting, see Notes in "OffSrvlock."

(6) When releasing this command, execute ”ResetCurlmt(3).”

(7) Even if setting is performed in detail, the specified thrust may not be output for a reason such as friction.

Example

```
CALL  SetForce_HM(10.0)Specify thrust '10.0(N).
```

SetForce_HC (Library)

Function

A current limitation library that specifies the thrust (unit: N) of the Z coordinate with the HC robot.

Format

SetForce_HC (<thrust unit : N >)

Explanation

Limits the thrust by limiting the motor current on the Z axis. The unit is N (Newton). The maximum thrust extremely varies depending on the impact force generated at contact with peripherals. The allowable thrust is a value assumed when the impact force at collision is almost 0 (collision at too low speed). The thrust also varies depending on the Z-axis position by the mechanical friction; therefore, this value should be assumed to be a typical value (not a guaranteed value). Like the current limitation library (SetCurlmt), a value that exceeds the allowable value is limited to the allowable value.

Macro Definition

The pacman.h file is necessary.

Related Terms

SetForce_HM, SetCurlmt, ResetCurlmt

Notes

(1) Since the motor current is limited during current limitation, operation is impossible at the highest speed and highest acceleration. For current limitation, use only the required steps. To use the current limitation, lower the acceleration.

(2) When the thrust is limited by the current limitation, if work collides at high speed, an impact force occurs due to the inertia between the work and hand. Set the current limitation before setting work, and lower the speed.

(3) Set the current limitation as the robot stops. If it is set during path operation, an error may occur.

(4) Execute this function on a task that obtains the robot control right (TAKEARM). If the robot control right is not obtained, error message "21F7 Arm semaphore cannot be fetched." is reported.

(5) When bit 1 of the "25: current limitation reset" set value is set to 0 (initial value) as a use condition, the servo lock release is reset (servo-lock) at motor power-on. To validate servo lock release just after the motor power is turned on, set +1 to the "current limitation reset" set value. For details on setting, see Notes in "OffSrvlock."

(6) When releasing this command, execute ”ResetCurlmt(3).”

(7) Even if setting is performed in detail, the specified thrust may not be output for a reason such as friction.

Example

```
CALL  SetForce_HC(10.0)Specify thrust '1(N).
```

SetCompControl (Library) (Version 1.4 or later)

Function

Enables the compliance function (dedicated command for 6-axis).

Format

SetCompControl

Explanation

Enables the compliance function. Enables the compliance conditions set by SetFrcLimit, SetCompRate, and SetFrcCoord.

Macro Definition

<pacman.h> is required.

Related Terms

SetFrcLimit, SetCompRate, SetFrcCoord, ResetCompControl, SetCompFControl

Notes

(1) You will receive an error "60f5 Cannot execute compliance control", when this library is executed while the gravity offset is disabled and the current limiting is enabled. Execute again after you enable the gravity offset and disable the current limiting. See ResetCurLmt and SetGravity for disabling the current limiting and enabling the gravity offset respectively.

(2) The compliance control will not be enabled while motors are off. The compliance control will be disabled when you turned off motors under the compliance control.

(3) Execute this command in a task holding robot control conditions (TAKEARM). If robot control conditions are not held, an error message "21F7 Arm semaphore cannot be fetched." is reported.

(4) Execute while your robot is stopping. Executing this library in a path motion will cause an execution after stop. When you receive an error "600b Robot is in motion" after you execute this library while your robot is in motion, wait until your robot stops after issuing a command such as Delay.

(5) When your robot moves due to an external force, you may receive an error "611* Excessive deviation". If this is the case, change the allowable deviation. Use SetEralw to change the allowable deviation.

(6) Do not execute this library when a force such as a contact force is applied to your robot. Use SetCompControl to enable the compliance control function while a force is applied to your robot.

(7) When a robot posture changes largely after you execute SetCompControl, an error may be generated in the compensation value for the gravity offset and your robot may move toward the direction of gravity. If the posture changes largely in the course of the compliance control, use ResetCompControl to disable the compliance control and execute SetCompControl again to enable the compliance control.

Example

```
CALL SetFrcCoord (1) 'Sets the force limiting coordinate system
CALL SetFrcLimit (100, 0, 100, 100, 100, 100)
                     'Sets the force limiting rate
CALL SetCompControl  'Enables the compliance control function
CALL SetEralw (1, 90)     'Sets the allowable deviation
```

SetCompFControl (Library) (Version 1.4 or later)

Function

Enables the compliance control function (dedicated command for 6-axis).

Format

SetCompFControl

Explanation

Enables the compliance control function as SetCompControl. However, the gravity offset compensation is not executed as SetCompControl.

Macro Definition

<pacman.h> is required

Related Terms

SetCompControl

Notes

(1) You will receive an error "60f5 Cannot execute compliance control", when this library is executed while the gravity offset is disabled and the current limiting is enabled. Execute again after you enable the gravity offset and disable the current limiting.

(2) The compliance control will not be enabled while motors are off. The compliance control will be disabled when you turned off motors under the compliance control.

(3) Execute this command in a task holding robot control conditions (TAKEARM). If robot control conditions are not held, an error message "21F7 Arm semaphore cannot be fetched." is reported.

(4) Execute while your robot is stopping. Executing this library in a path motion will cause an execution after stop. When you receive an error "600b Robot is in motion" after you execute this library while your robot is in motion, wait until your robot stops after issuing a command such as Delay.

(5) Set the tip load exactly. Your robot may fall in the direction of gravity when the tip load setting and the actual tip load differ. You can execute SetGrvOffset to prevent a fall due to gravity.

Example

```
CALL SetGrvOffset      'Calculates the gravity offset compensation value
CALL SetFrcCoord (1)   'Sets the force limiting coordinate
CALL SetFrcLimit (100, 0, 100, 100, 100, 100)
                       'Sets the force limiting rate
CALL SetCompFControl   'Enables the compliance control function
```

ResetCompControl (Library) (Version 1.4 or later)

Function

Disables the compliance control function (dedicated command for 6-axis).

Format

ResetCompContrl

Explanation

Disables the compliance control function.

Macro Definition

<pacman.h> is required.

Related Terms

SetCompControl

Notes

(1) Execute this command in a task holding robot control conditions (TAKEARM). If robot control conditions are not held, an error message "21F7 Arm semaphore cannot be fetched." is reported.

(2) Execute while your robot is stopping. Executing this library in a path motion will cause an execution after stop. When your robot stops suddenly, and you receive an error "612* Over current" after you execute this library while your robot is in motion, Set 1 to <Set value> or wait until your robot stops after issuing a command such as Delay.

(3) You will receive an error "60f9 Abnormal operation in disabling compliance control", if you execute a step back or a program reset after a momentary stop during executing this library.

(4) When you use SetEralw to change the allowable deviation values while the compliance control is enabled, the allowable deviation values will return to their initial values. The allowable deviation values may not be reset to the initial values when an error occurs while executing ResetCompControl. If this is the case, use ResetEralw to initialize the allowable deviation values after disabling the compliance control.

(5) You may receive an error "608* J* Directed speed limit was exceeded". If this is the case, use aspChange to change the optimal load capacity mode to 2 or 3 before you execute ResetCompControl and resume the optimal load capacity mode to the previous value after the execution.

Example

```
CALL ResetCompControl  'Disables the compliance control function
CALL ResetEralw        'Initializes the allowable deviation values
```

SetFrcCoord (Library) (Version 1.4 or later)

Function

Selects a force limiting coordinate system (dedicated command for 6-axis).

Format

SetFrcCoord(<Set value>)

Explanation

Selects a coordinate system for force limiting values specified by SetFrcLimit and SetCompRate. You can use a set value 0 for the base coordinate system, a set value 1 for the tool coordinate system, and a set value 2 for the work coordinate system of your robot.

Macro Definition

<pacman.h> is required.

Related Terms

SetFrcLimit, SetCompRate, SetFrcCoord, ResetCompControl

Notes

(1) Execute this command in a task holding robot control conditions (TAKEARM). If robot control conditions are not held, an error message "21F7 Arm semaphore cannot be fetched." is reported.

(2) This library is not available while the compliance control is enabled. When you execute this library under the compliance control, you will receive an error "60fa Compliance control is enabled".

(3) When you specify 1 for <Set value> to select the tool coordinate system, the tool coordinate will be the tool coordinate for enabling the compliance control (executing SetCompControl). When you use the changetool command to change the tool coordinate while the compliance control is enabled, the force limiting coordinate will not be changed.

(4) When you specify 2 for <Set value> to select the work coordinate system, the work coordinate will be the work coordinate for enabling the compliance control (executing SetCompControl). When you use the changetool command to change the work coordinate while the compliance control is enabled, the force limiting coordinate will not be changed.

(5) The set value will be initialized to 0 (the base coordinate system) after the controller is turned on.

Example

```
        CALL SetFrcCoord (1)   'Sets the force limiting coordinate system to the tool
                               'coordinate
        Changetool 2           'Sets the tool coordinate to tool2
        CALL SetFrcLimit (100, 0, 100, 100, 100, 100)
                               'Sets the force limiting rate
        CALL SetCompControl    'Sets the force limiting rate in Y direction of the tool
2
                               'coordinate system to 0% and enables the compliance
control
```

SetFrcLimit (Library) (Version 1.4 or later)

Function

Sets the force limiting rates (dedicated command for 6-axis).

Format

SetFrcLimit (<Limiting rate along X>, <Limiting rate along Y>, <Limiting rate along Z>, <Limiting rate about X>, <Limiting rate about Y>, <Limiting rate about Z>)
Setting ranges from 0 to 100. Up to two decimal places are valid.

Explanation

Sets the force limiting rates along and about X, Y, and Z axes of a coordinate system specified by SetFrcCoord.

Macro Definition

<pacman.h> is required.

Related Terms

ResetFrcLimit, SetFrcCoord, SetCompControl

Notes

(1) Execute this command in a task holding robot control conditions (TAKEARM). If robot control conditions are not held, an error message "21F7 Arm semaphore cannot be fetched." is reported.

(2) This library is not available while the compliance control is enabled. When you execute this library under the compliance control, you will receive an error "60fa Compliance control is enabled".

(3) All the set values for along and around the X, Y and Z axes will be initialized to 100 after the controller is turned on.

Example

```
        CALL SetFrcCoord (1)   'Sets the force limiting coordinate system to the tool
                               'coordinate
        CALL SetFrcLimit (100, 0, 100, 100, 100, 100)
                               'Sets the force limiting rates
        CALL SetCompControl    'Sets the force limiting rate in Y direction of the tool
                               'coordinate system to 0% and enables the compliance
 control
```

ResetFrcLimit (Library) (Version 1.4 or later)

Function

Initializes the force limiting rates (dedicated command for 6-axis).

Format

ResetFrcLimit

Explanation

Initializes the force limiting rates. All rates along and about X, Y, and Z axes are set to 100%.

Macro Definition

<pacman.h> is required.

Related Terms

SetFrcLimit

Notes

(1) Execute this command in a task holding robot control conditions (TAKEARM). If robot control conditions are not held, an error message "21F7 Arm semaphore cannot be fetched." is reported.

(2) This library is not available while the compliance control is enabled. When you execute this library under the compliance control, you will receive an error "60fa Compliance control is enabled".

Example

```
CALL ResetCompControl  'Disables compliance control
CALL ResetFrcLimit     'Initializes the force limiting rates
```

SetCompRate (Library) (Version 1.4 or later)

Function

Sets the compliance rates under the compliance control (dedicated command for 6-axis).

Format

SetCompRate (<Compliance along X>, <Compliance along Y>, <Compliance along Z>, <Compliance about X>, <Compliance about Y>, <Compliance about Z>)

Explanation

Sets the compliance rates along and about X, Y, and Z axes of a coordinate system specified by SetFrcCoord.
Setting ranges from 0 to 100 and 0 gives the maximum compliance. Up to two decimal places are valid.

Macro Definition

<pacman.h> is required.

Related Terms

ResetCompRate, SetFrcCoord, SetCompControl

Notes

(1) Execute this command in a task holding robot control conditions (TAKEARM). If robot control conditions are not held, an error message "21F7 Arm semaphore cannot be fetched." is reported.

(2) This library is not available while the compliance control is enabled. When you execute this library under the compliance control, you will receive an error "60fa Compliance control is enabled".

(3) All the set values for along and around the X, Y and Z axes will be initialized to 100 after the controller is turned on.

Example

```
        CALL SetFrcCoord (1)    'Sets the force limiting coordinate system to the tool
                                'coordinate
        CALL SetCompRate (100, 0, 100, 100, 100, 100)
                                'Sets the compliance rate
        CALL SetCompControl     'Sets the compliance rate in Y direction of the tool
                                'coordinate system to 0% and enables the compliance
 control
```

ResetCompRate (Library) (Version 1.4 or later)

Function

Initializes the compliance rates (dedicated command for 6-axis).

Format

ResetCompRate

Explanation

Initializes the compliance rates along and about X, Y, and Z axes to 100%.

Macro Definition

<pacman.h> is required.

Related Terms

SetCompRate

Notes

(1) Execute this command in a task holding robot control conditions (TAKEARM). If robot control conditions are not held, an error message "21F7 Arm semaphore cannot be fetched." is reported.

(2) This library is not available while the compliance control is enabled. When you execute this library under the compliance control, you will receive an error "60fa Compliance control is enabled".

Example

```
CALL ResetCompControl  'Disables the compliance control
CALL ResetCompRate     'Initializes the compliance rates
```

SetFrcAssist (Library) (Version 1.4 or later)

Function

Sets the force assistance under the compliance control (special compliance control function library) (dedicated command for 6-axis).

Format

SetFrcAssist (<Force assistance along X>, <Force assistance along Y>, <Force assistance along Z>, <Moment assistance about X>, <Moment assistance about Y>, <Moment assistance about Z>)

Explanation

Sets the force assistance along and the moment assistance about X, Y, and Z axes of a coordinate system specified by SetFrcCoord. The maximum set value is 10% of the maximum force limiting value.
The unit for the force setting is [N]. The unit for the moment setting is [Nm]. Up to one decimal place is valid.

Macro Definition

<pacman.h> is required.

Related Terms

ResetFrcAssist, SetFrcCoord, SetCompControl

Notes

(1) Execute this command in a task holding robot control conditions (TAKEARM). If robot control conditions are not held, an error message "21F7 Arm semaphore cannot be fetched." is reported.

(2) Your robot may move toward the direction to which the force assistance and the moment assistance are applied. If this is the case, reduce the set values.

(3) All the set values for along and around the X, Y and Z axes will be initialized to 0 after the controller is turned on.

Example

```
        CALL SetFrcCoord (1)   'Sets the force limiting coordinate system to the tool
                               'coordinate
        CALL SetFrcAssist (-30, 0, 0, 0, 0, 0)
                               'Sets the force assistance to 30 [N] toward -X direction
        CALL SetFrcLimit (0, 100, 100, 100, 100, 100)
                               'Sets the force limiting rates
        CALL SetCompControl    'Enables the compliance control function.  Force limiting
                               'in X direction is 0% and a force of 30 [N] is applied
toward
                               '-X direction
```

ResetFrcAssist (Library) (Version 1.4 or later)

Function

Initializes the force assistance (special compliance control function library) (dedicated command for 6-axis).

Format

ResetFrcAssisit

Explanation

Initializes the force assistance along and the moment assistance about X, Y, and Z axes of a coordinate system specified by SetFrcCoord are set to 100%.

Macro Definition

<pacman.h> is required.

Related Terms

SetFrcAssist

Notes

(1) Execute this command in a task holding robot control conditions (TAKEARM). If robot control conditions are not held, an error message "21F7 Arm semaphore cannot be fetched." is reported.

Example

```
       CALL ResetCompControl  'Disable the compliance control
       CALL ResetFrcAssist    'Initializes the force assistance and the moment 
assistance
```

SetCompJLimit (Library) (Version 1.4 or later)

Function

Sets the current limit under the compliance control (special compliance control function library) (dedicated command for 6-axis).

Format

SetCompJLimit (<J1 current limit>, <J2 current limit>, <J3 current limit>, <J4 current limit>, <J5 current limit>, <J6 current limit>)

Explanation

Sets the current limit under the compliance control. The rated current of a motor corresponds to 100. When you use SetFrcLimit to set 0 to all the directions, the motor currents are limited to values less than the setting.
Setting ranges from 0 to 100.

Macro Definition

<pacman.h> is required.

Related Terms

ResetCompJLimit, SetCompControl

Notes

(1) Execute this command in a task holding robot control conditions (TAKEARM). If robot control conditions are not held, an error message "21F7 Arm semaphore cannot be fetched." is reported.

(2) The set values will be initialized and all the current limits will be 0 after the controller is turned on.

(3) When you use SetCompJLimit to set relatively large values, your robot may present an oscillation resulting in an error. If this is the case, use SetCompRate and SetDumpRate to adjust the compliance.

Example

```
CALL SetFrcCoord (1)   'Sets the force limiting coordinate system to the tool
                       'coordinate
CALL SetCompJLimit (30, 0, 0, 0, 0, 0)
                       'Sets the current limit for J1 to 30%
CALL SetFrcLimit (0, 100, 100, 100, 100, 100)
                       'Sets the force limiting rates
CALL SetCompControl    'Enables the compliance control function.
```

ResetCompJLimit (Library) (Version 1.4 or later)

Function

Initializes the current limit under the compliance control (special compliance control function library) (dedicated command for 6-axis).

Format

ResetCompJLimit

Explanation

Initializes the current limit under the compliance control and set 0 to all axes.

Macro Definition

<pacman.h> is required.

Related Terms

SetCompJLimit, SetCompControl

Notes

(1) Execute this command in a task holding robot control conditions (TAKEARM). If robot control conditions are not held, an error message "21F7 Arm semaphore cannot be fetched." is reported.

Example

```
CALL ResetCompControl  'Disables the compliance control
CALL ResetCompJLimit   'Initializes the current limits
```

SetCompVMode (Library) (Version 1.4 or later)

Function

Sets the velocity control mode under the compliance control (special compliance control function library) (dedicated command for 6-axis).

Format

SetCompVMode

Explanation

Enables the compliance velocity control mode when SetCompControl is executed.

Macro Definition

<pacman.h> is required.

Related Terms

ResetCompVMode

Notes

(1) Execute this command in a task holding robot control conditions (TAKEARM). If robot control conditions are not held, an error message "21F7 Arm semaphore cannot be fetched." is reported.

Example

```
CALL SetCompVMode      'Enables the compliance velocity control mode
CALL SetCompControl    'Enables the compliance control
```

ResetCompVMode (Library) (Version 1.4 or later)

Function

Disables the velocity control mode under the compliance control (special compliance control function library) (dedicated command for 6-axis).

Format

ResetCompVMode

Explanation

Disables the compliance velocity control mode when SetCompControl is executed.

Macro Definition

<pacman.h> is required.

Related Terms

SetCompVMode

Notes

(1) Execute this command in a task holding robot control conditions (TAKEARM). If robot control conditions are not held, an error message "21F7 Arm semaphore cannot be fetched." is reported.

Example

```
CALL ResetCompVMode    'DIsables the compliance velocity control mode
CALL SetCompControl    'Enables the compliance control
```

SetCompEralw (Library) (Version 1.4 or later)

Function

Sets the allowable deviation values of the position and the posture of the tool tip under the compliance control (dedicated command for 6-axis).

Format

SetCompEralw (<Allowable deviation along X>, <Allowable deviation Y>, <Allowable deviation Z>, <Allowable deviation X>, <Allowable deviation Y>, <Allowable deviation Z>)

Explanation

Sets the allowable deviation values of the position and the posture of the tool tip under the compliance control. The unit for the allowable deviation along X, Y, and Z is (mm), and the unit for the allowable deviation about X, Y, and Z is (degree). Up to one decimal place is valid.

Macro Definition

<pacman.h> is required.

Related Terms

ResetCompEralw, SetFrcCoord

Notes

(1) Execute this command in a task holding robot control conditions (TAKEARM). If robot control conditions are not held, an error message "21F7 Arm semaphore cannot be fetched." is reported.

(2) The initial values of allowable deviation are 100 (mm) along X, Y, and Z and 30 (degree) about X, Y, and Z. The maximum value about X, Y, and Z axes is 175 (degree). If you set larger values than the maximum value, you will receive an error "6003 Valid range was exceeded".

(3) If the position or the posture deviation of the tool tip exceeds the allowable value, you will receive an error "60f8 Position deviation under compliance control error".

(4) The coordinate system used for setting the deviation is the one set by SetFrcCoord. When you specify as SetFrcCoord (1), the setting coordinate system for SetCompEralw is the tool coordinate system.

Example

```
       CALL SetFrcCoord (2)   'Sets the force limiting coordinate system to the work
                              'coordinate system
       CALL SetFrcLimit (100, 0, 100, 100, 100, 100)
                              'Sets the force limiting rate for the Y direction of the
work
                              'coordinate system to 0%
       CALL SetCompEralw (10, 150, 10, 5, 5, 5)
                              'Sets the allowable deviation values along X and Z of the
work
                              'coordinate to 10 (mm), along Y to 150 (mm), and about X,
Y,
                              'and Z to 5 (degree).
```

ResetCompEralw (Library) (Version 1.4 or later)

Function

Initializes the allowable deviation values of the position and the posture of the tool tip under the compliance control (dedicated command for 6-axis).

Format

ResetCompEralw

Explanation

Initializes the allowable deviation values of the position and the posture of the tool tip under the compliance control. The initial values of allowable deviation are 100 (mm) along X, Y, and Z and 30 (degree) about X, Y, and Z.

Macro Definition

<pacman.h> is required.

Related Terms

SetCompEralw

Notes

(1) Execute this command in a task holding robot control conditions (TAKEARM). If robot control conditions are not held, an error message "21F7 Arm semaphore cannot be fetched." is reported.

Example

```
CALL ResetCompEralw
```

SetDampRate (Library) (Version 1.4 or later)

Function

Sets the damping rates under the compliance control (dedicated command for 6-axis).

Format

SetDampRate (<Damping rate along X>, <Damping rate along Y>, <Damping rate along Z>, <Damping rate about X>, <Damping rate about Y>, <Damping rate about Z>)

Explanation

Sets the damping rates along and about X, Y, and Z axes of a coordinate system specified by SetFrcCoord.
Setting ranges from 0 to 100 and 0 gives the maximum damping. Up to two decimal places are valid.

Macro Definition

<pacman.h> is required.

Related Terms

ResetDampRate, SetFrcCoord, SetCompRate

Notes

(1) Execute this command in a task holding robot control conditions (TAKEARM). If robot control conditions are not held, an error message "21F7 Arm semaphore cannot be fetched." is reported.

(2) This library is not available while the compliance control is enabled. When you execute this library under the compliance control, you will receive an error "60fa Compliance control is enabled".

(3) All the set values for along and around the X, Y and Z axes will be initialized to 100 after the controller is turned on.

(4) Execute this function on a task that obtains the robot control right (TAKEARM). If the robot control right is not obtained, error message "21F7 Arm semaphore cannot be fetched." is reported.

(5) If you execute SetCompRate after SetDampRate, the compliance rates are altered to the setting by SetCompRate.

Example

```
        CALL SetFrcCoord (1)   'Sets the force limiting coordinate system to the tool
                               'coordinate system
        CALL SetCompRate (100, 0, 100, 100, 100, 100)
                               'Sets the compliance rate
        CALL SetDampRate (100, 20, 100, 100, 100, 100)
                               'Sets the damping rate
        CALL SetCompControl    'Sets the compliance rate and the damping rate in Y
direction
                               'of the tool coordinate system to 0% and 20% respectively
and
                               'enables the compliance control
```

ResetDampRate (Library) (Version 1.4 or later)

Function

Initializes the damping rates under the compliance control (dedicated command for 6-axis).

Format

ResetDampRate

Explanation

Initializes the damping rates along and about X, Y, and Z axes to 100.

Macro Definition

<pacman.h> is required.

Related Terms

SetDampRate

Notes

(1) Execute this command in a task holding robot control conditions (TAKEARM). If robot control conditions are not held, an error message "21F7 Arm semaphore cannot be fetched." is reported.

(2) This library is not available while the compliance control is enabled. When you execute this library under the compliance control, you will receive an error "60fa Compliance control is enabled".

Example

```
CALL ResetDampRate     'Initializes the damping rates
```

SetVibControl (Library) (Version 1.4 or later)

Function

Sets to the residual vibration reduction control mode (dedicated command for 6-axis).

Format

SetVibControl

Explanation

Switches to the residual vibration reduction control mode and reduces the residual vibration when robot stops its motion.

Macro Definition

<pacman.h> is required.

Related Terms

ResetVibControl

Notes

(1) Execute this command in a task holding robot control conditions (TAKEARM). If robot control conditions are not held, an error message "21F7 Arm semaphore cannot be fetched." is reported.

(2) If you execute SetVibControl while using SetEralw to change the allowable angle deviation, the allowable angle deviation will be initialized. Use SetEralw to set the allowable angle deviation again.

(3) The path precision will decrease due to motion follow-up delay in the residual vibration reduction control mode. Avoid this mode when the path precision is required.

(4) When you execute SetVibControl during a path motion, the library is executed after the motion stops. Thus, this will be an execution after stop.

Example

```
CALL SetVibControl     'Sets to the residual vibration reduction control mode
```

ResetVibControl (Library) (Version 1.4 or later)

Function

Returns from the residual vibration reduction control mode to the normal control mode.

Format

ResetVibConrol

Explanation

Returns from the residual vibration reduction control mode to the normal control mode.

Macro Definition

<pacman.h> is required.

Related Terms

ResetVibControl

Notes

(1) Execute this command in a task holding robot control conditions (TAKEARM). If robot control conditions are not held, an error message "21F7 Arm semaphore cannot be fetched." is reported.

(2) If you execute ResetVibControl while using SetEralw to change the allowable angle deviation, the allowable angle deviation will be initialized. Use SetEralw to set the allowable angle deviation again.

(3) When you execute ResetVibControl during a path motion, the library is executed after the motion stops. Thus, this will be an execution after stop.

Example

```
CALL ResetVibControl   'Sets to the normal control mode
```

MotionSkip (Library) (Version 1.5 or later)

Function

Aborts running motion commands.

Format

`MotionSkip`

Explanation

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

Related Terms

`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 (Library) (Version 1.5 or later)

Function

Judges whether execution of running motion commands is complete.

Format

`MotionComp(<MotionCommandComplete>)`

Explanation

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.

Related Terms

`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.
```

ArchMove (Library) (Version 1.9 or later)

Function

Executes an arch motion.

Format

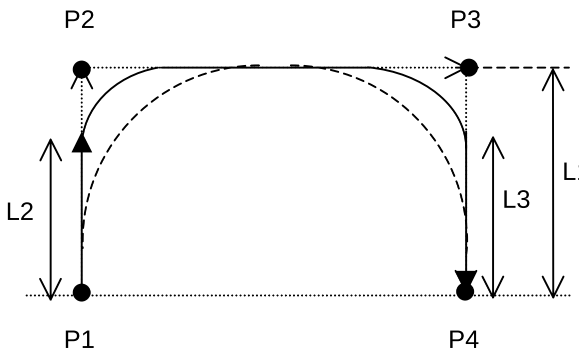
```
ArchMove (<destination (Position variable)>, <Z-axis move>)
```

Explanation

This library makes a pick-and-place motion from current position P1 (pick) to destination position P4 (place) via the vicinity of route positions P2 and P3.

The route position P3 is located above destination position P4 by the vertical move distance (L1) specified by `<Z-axis move>`.

The route position P2 is on the Z axis of source position P1 and has the same Z-axis value as P3.



Arch start position (L2) and end position (L3) should be defined in the SetArchParam library.

Macro Definition

Requires file <pacman.h>

Related Terms

SetArchParam (library)

Notes

(1) Execute this library in a TAKEARMed task that holds an arm semaphore.

(2) An arch motion does not support the position control so that the path of the arm endpoint may vary depending upon the actual robot speed. Be careful with interference with the surrounding facilities when the robot is in arch motion.

(3) The Z-axis value of the source position and that of the destination position are not identical with each other. Otherwise, this library will not function normally.

(4) This library is not available to the work coordinates system. Use it in the base coordinates system.

Example

```
CALL ArchMove (P10,100)     'Moves the arm endpoint to destination P10
                            'with 100 mm Z-axis move. The current arch
                            start and end positions will apply.
```

SetArchParam (Library) (Version 1.9 or later)

Function

Defines the start position of a horizontal movement (Arch start position) in upward movement of the arm endpoint and the end position in downward movement (Arch end position).

Format

```
SetArchParam (<Arch start position>, <Arch end position>)
```

Explanation

This library configures an arch form.

`<Arch start position>` is the distance (mm) from the source position in the direction of the Z axis; `<Arch end position>` is that from the destination position.

The entry range is from 0 to "Maximum reach position (mm) in the Z-axis direction" in increment of 1 mm.

The defaults of `<Arch start position>` and `<Arch end position>` are 0 mm when the robot controller is turned on.

Values defined in this library will be effective until the robot controller is turned off.

Macro Definition

Requires file <pacman.h>

Related Terms

ArchMove

Notes

Specifying any invalid value will cause no syntax error, but it will result in a run-time error at the execution of the `ArchMove`.

Example

```
CALL SetArchParam (10,20)   'Set the arch form so that the arm endpoint
                            'moves upward 10 mm, starts horizontal move
                            'and ends it 20 mm above the destination.
<Program lines hidden>

Call ArchMove (P10,100)     'Execute arch motion towards destination P10
                            'with 100 mm Z-axis vertical move.
```

SetMonitorCond (Library) (Version 1.5 or later)

Function

Sets the monitoring conditions for single-joint servo data monitor.

Format

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

Explanation

`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.

Related Terms

`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 (Library) (Version 1.5 or later)

Function

Starts monitoring single-joint servo data.

Format

```
StartSrvMonitor
```

Explanation

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

Related Terms

`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 (Library) (Version 1.5 or later)

Function

Stops monitoring single-joint servo data.

Format

```
StopSrvMonitor
```

Explanation

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

Related Terms

`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 (Library) (Version 1.5 or later)

Function

Initializes the pointer of data obtained by the single-joint servo data monitor function.

Format

`ClearSrvMonitor`

Explanation

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

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.

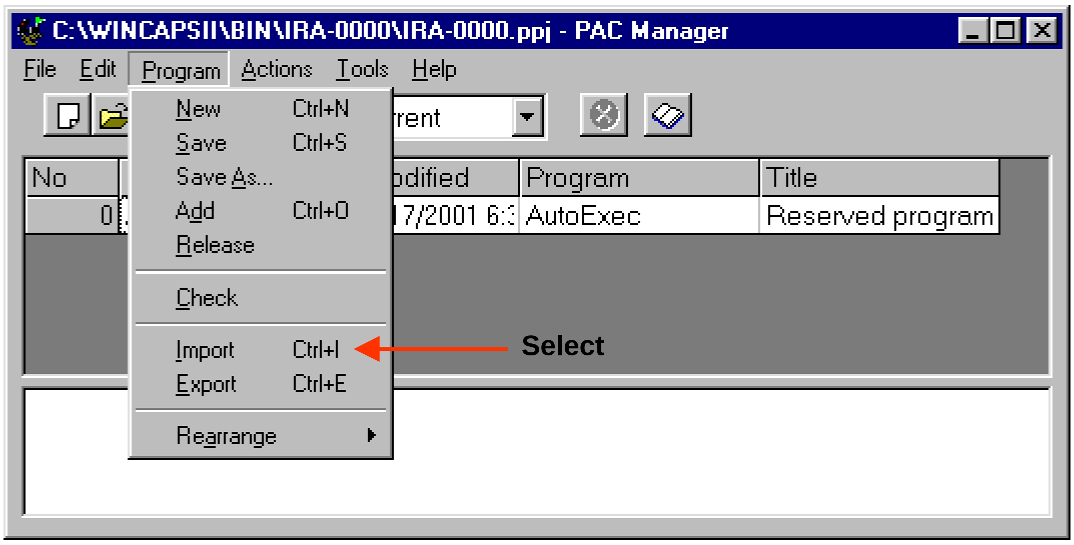
Example

```
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.)
```

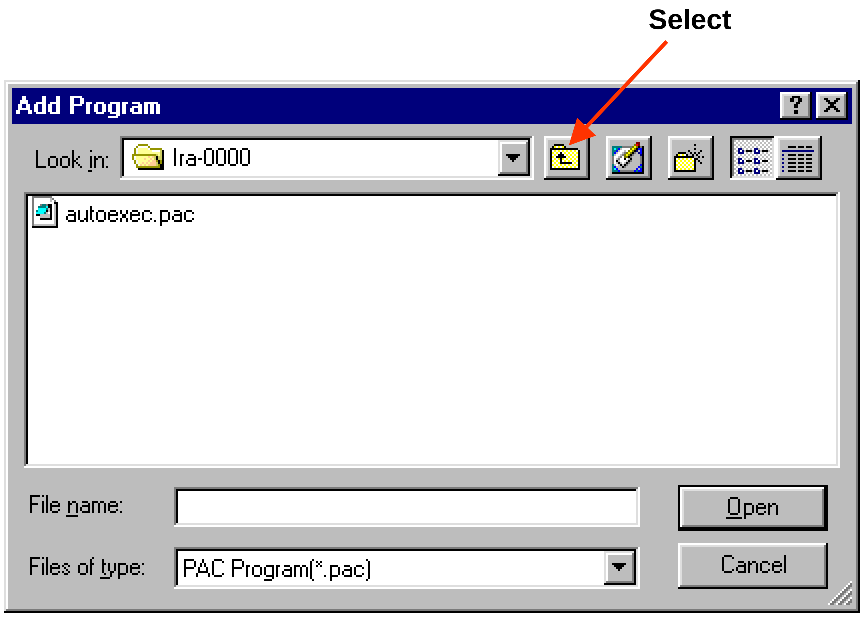
8 Creating TP Easy Operation Panel Screen

8.1 How to Use Sample Programs for Creating Operation Panel Screen

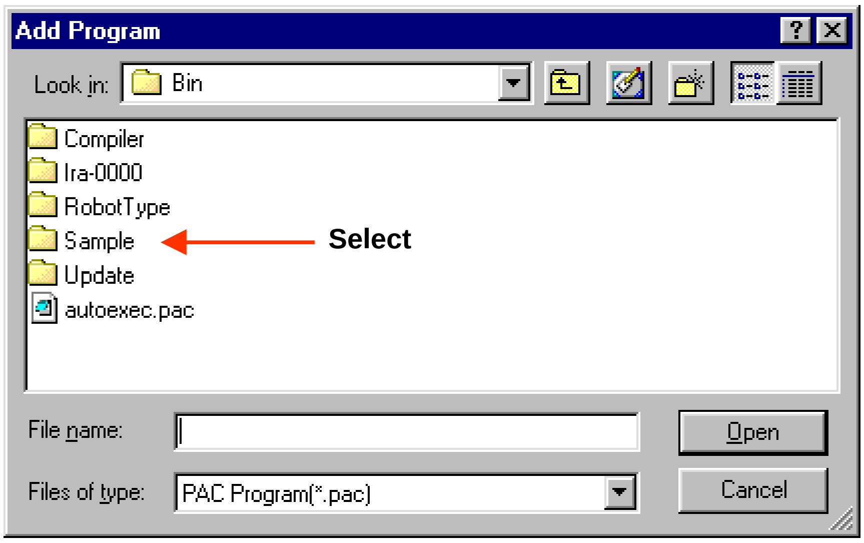
To create the operation panel screen, a sample program is registered in WINCAPS II under \Wincaps2\Sample\make_screen. To import the sample program to the PAC manager, select [Program] → [Import] → [Add Program] to move to the folder above, and aslo select a required sample program.



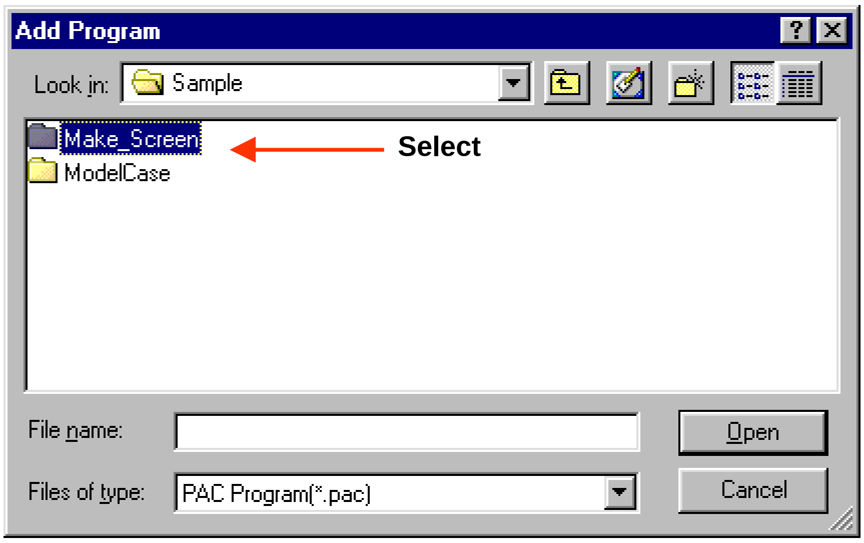
No	odified	Program	Title
0	7/2001 6:3	AutoExec	Reserved program



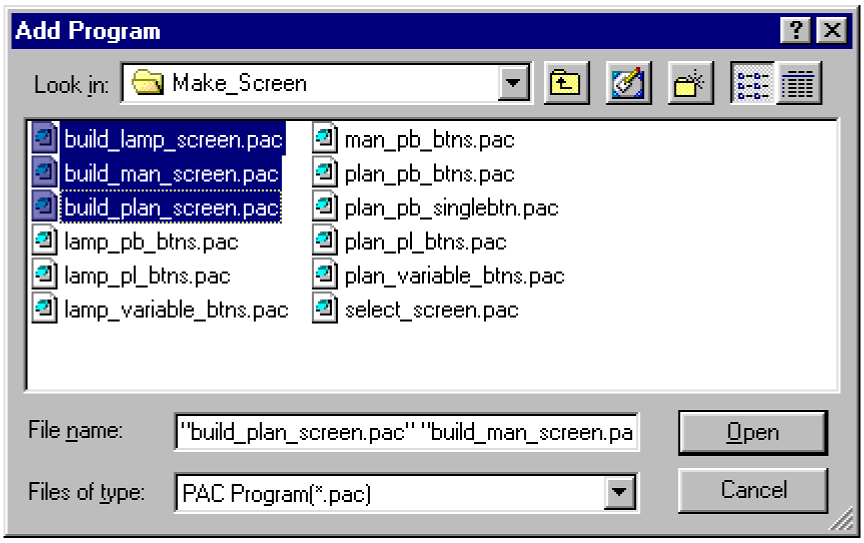
Select a sample folder.



Select the make_screen folder.



Select a prorgam to be imported.



8.2 Sample Programs for Creating Operation Panel Screen

Describes the sample programs which are provided in WINCAPSII.

Select_Screen (Sample program) (Version 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 Definition

<button.h> is necessary.

Related Terms

`Build_Lamp_Screen`, `Build_Man_Screen`, and `Build_Plan_Screen`

Build_Lamp_Screen (Sample program) (Version 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.

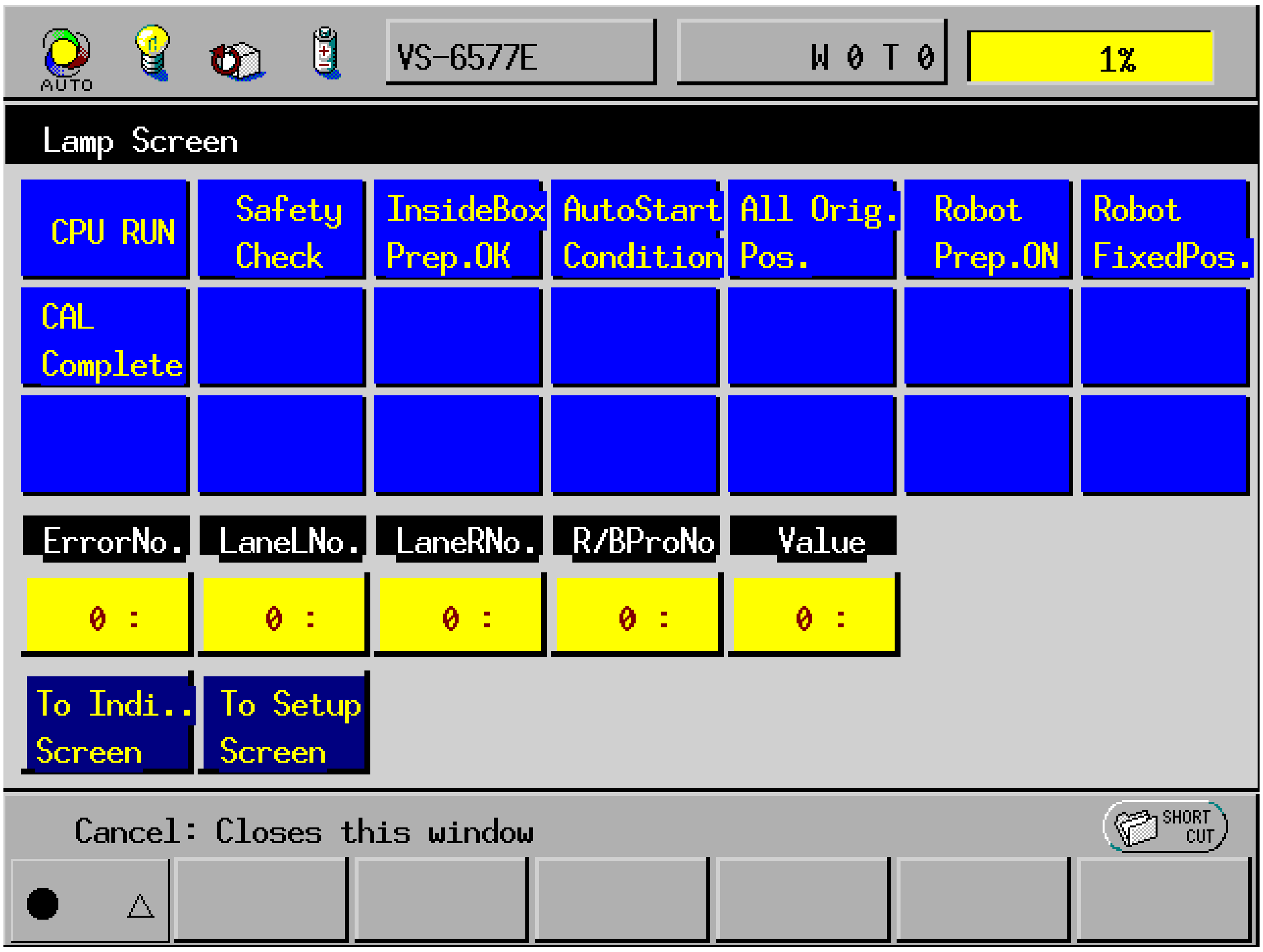
This program is a sample; so, change it according to the actual processing.

Macro Definition

<button.h> is necessary.

Related Terms

`Lamp_pl_btns, Lamp_variable_btns, Lamp_pb_btns`



Build_Man_Screen (Sample program) (Version 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.

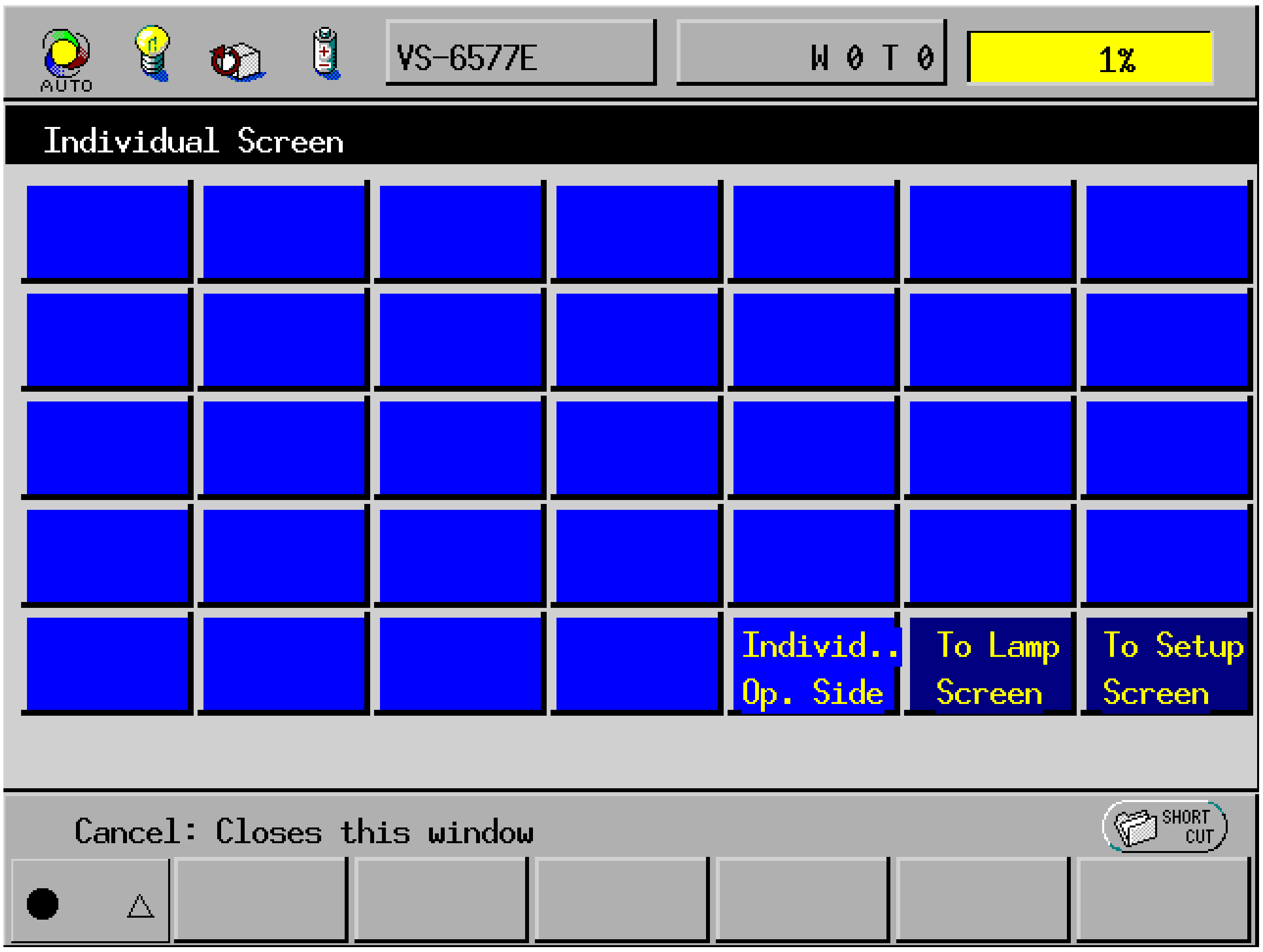
This program is a sample; so, change it according to the actual processing.

Macro Definition

<button.h> is necessary.

Related Terms

`Man_pb_btns`



Build_Plan_Screen (Sample program) (Version 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.

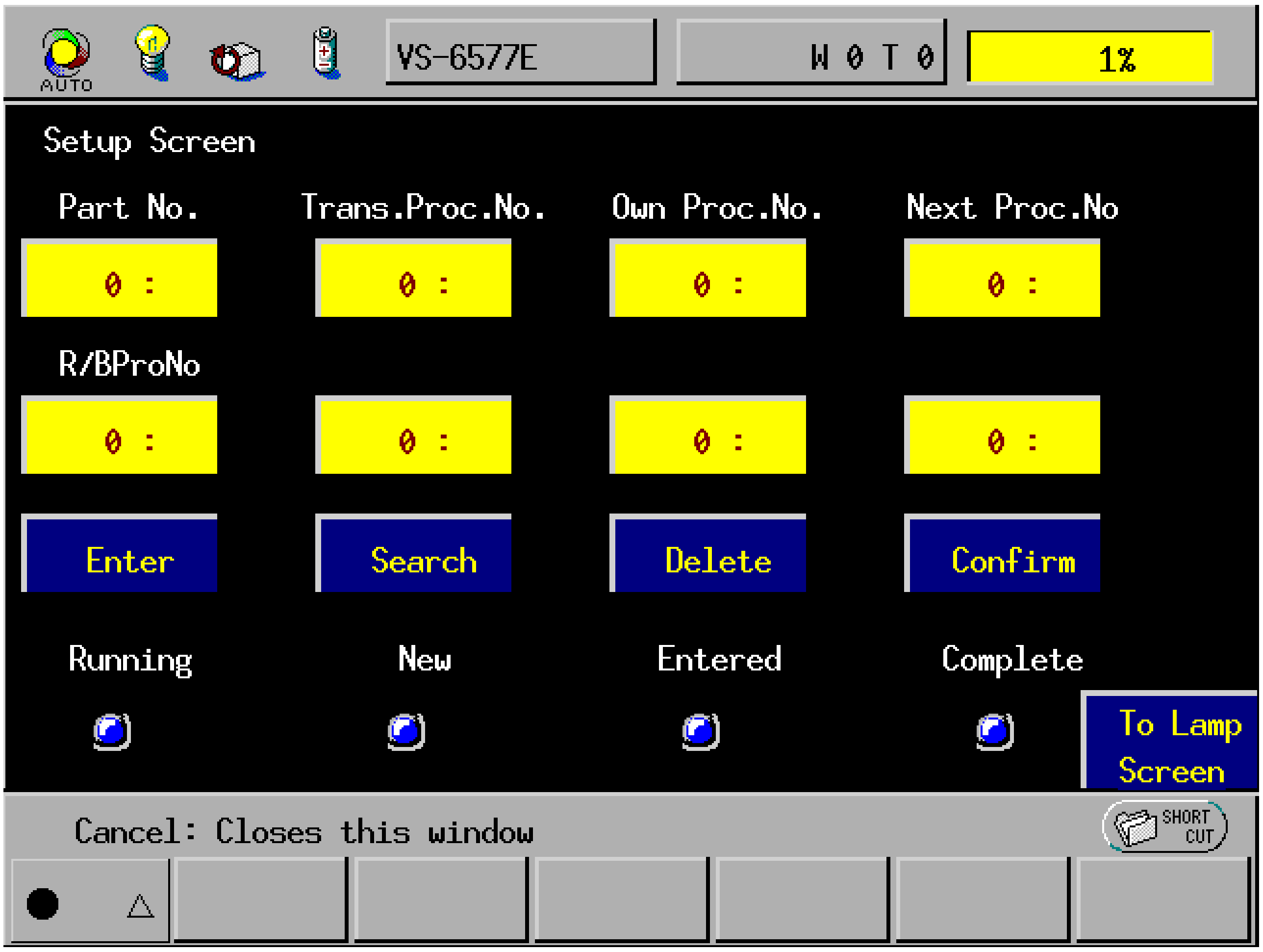
This program is a sample; so, change it according to the actual processing.

Macro Definition

<button.h> is necessary.

Related Terms

`Plan_variable_btns, Plan_pb_btns,Plan_pl_btns, Plan_pb_singlebtn`



Lamp_pl_btns (Sample program) (Version 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 Definition

<button.h> is necessary.

Related Terms

`make_LED`

Lamp_variable_btns (Sample program) (Version 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 Definition

<button.h> is necessary.

Related Terms

`make_PARAM_BOX`, `make_LABEL`

Lamp_pb_btns (Sample program) (Version 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 Definition

<button.h> is necessary.

Related Terms

`make_PB`

Man_pb_btns (Sample program) (Version 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 Definition

<button.h> is necessary.

Related Terms

`make_PB`

Plan_variable_btns (Sample program) (Version 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 Definition

<button.h> is necessary.

Related Terms

`make_PARAM_BOX, make_LABEL`

Plan_pb_btns (Sample program) (Version 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) (Version 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 Definition

<button.h> is necessary.

Related Terms

`make_LED`, `make_LABEL`

Plan_pb_singlebtn (Sample program) (Version 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 Definition

<button.h> is necessary.

Related Terms

`single_button_set`

8.3 Libraries for Creating Operation Panel Screen

Describes the libraries for creating operation panel screen.

make_PB (Library) (Version 1.7 or later)

Function

Creates a PB button.

Format

```
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 Definition

<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.

Format

```
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 Definition

<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).

Format

```
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 Definition

<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).

Format

```
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 Definition

<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.

Format

`single_button_set`

Explanation

This program sets only one button by specifying a parameter for a specifeid button number.

Macro Definition

<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).

Format

`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 Definition

<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.

Format

`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 Definition

<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.

Format

`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 Definition

<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)
```

9 Vision

Describes the libraries for vision functions.

viTran6 (Library) (for the Vision board)

Function

Transforms the vision coordinates to robot coordinates (for 6 axes).

Format

viTran6 (<Calibration number>, <Vision coordinate X>, <Vision coordinate Y>, <Robot coordinate (P type)>)

Explanation

This statement transforms the vision coordinates (X, Y) to the robot absolute coordinates (X, Y, Z).
For execution of this library, it is required to execute vision calibration and transmit the vision calibration data with the controller.
Designate a number which executes vision calibration as a calibration number.
With this coordinate transformation, only the absolute coordinates (X, Y, Z) can be obtained. Figure data must be set separately in order to execute the MOVE instruction.

Example

```
                  'Stores the vision coordinate X to F1.
                  'Stores the vision coordinate Y to F2.
TAKEARM
CHANGETOOL 1      'Sets the fool to which vision calibration was performed.
MOVE P,P10        'Moves to the stand-by position.
P11 = P10         'Copies the figure at the stand-by position.
CALL viTran6(0, F1, F2, P11)
                  'Transforms the vision coordinates to the robot coordinates
                  '(“0” of vision calibration is used).
APPROACH P, P11, 100
MOVE L, P11       'Moves to the position measured with vision.
DEPART L,100
CHANGETOOL 0
```

viTran6S (Library) (for Vision-21)

Function

Transforms the vision coordinates to robot coordinates (for 6 axes).

Format

viTran6S (<Calibration number>, <Vision coordinate X>, <Vision coordinate Y>, <Robot coordinate X>, <Robot coordinate Y>, <Robot coordinate Z>)

Explanation

This statement transforms the vision coordinates (X, Y) to the robot absolute coordinates (X, Y, Z).
For execution of this library, it is required to execute vision calibration and transmit the vision calibration data with the controller.
Designate a number which executes vision calibration as a calibration number.
With this coordinate transformation, only the absolute coordinates (X, Y, Z) can be obtained. Figure data must be set separately in order to execute the MOVE instruction.

Index

Vertical articulated V∗-D/-E SERIES
Horizontal articulated H∗-D/-E SERIES
Cartesian coordinate XYC-4D SERIES
Vision device μVision-21 SERIES

PROGRAMMER'S MANUAL II PAC Library Ver.1.98

First Edition	May 2002
Second Edition	December 2002

DENSO WAVE INCORPORATED
Factory Automation Division

6E**C