MAE 263B
Dynamics and Robotics

Lab 2

Exercise 1: Online Robot Programming

Instructions for Online Programming of the Denso Robot

1 Task

Make a program online, e.g., using the Teach Pendant, to stack six geometrically identical blocks, which are placed on a given fixture, to achieve the basic or challenging structure.

2 What you will have in the workcell

- The Denso robot with robot controller
- Six geometrically identical blocks (metal or plastic)[1]
- Block fixture

3 Step-by-step Instructions

3.1 Get the Robot Ready

You should follow all the instructions you have learned from lab 1, especially the safety requirement.

3.2 Register a New Program

The first step is to register a new program.

(1) Step 1: Press [F1 Program] on the basic screen

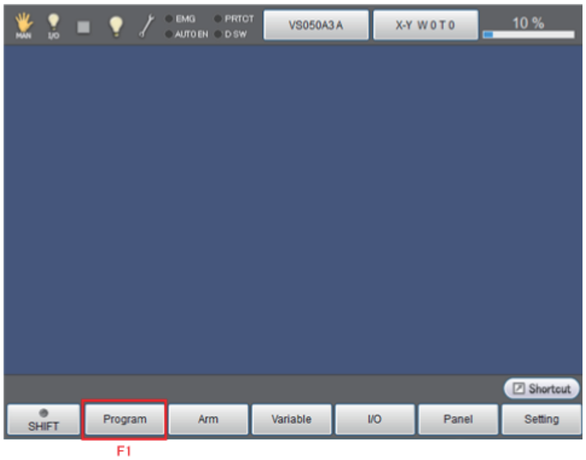


Figure 1: Basic Screen

.

[1] The metal and plastic blocks are exactly the same size; feel free to use either for this lab.

(2) Step 2: Press [F1 New]

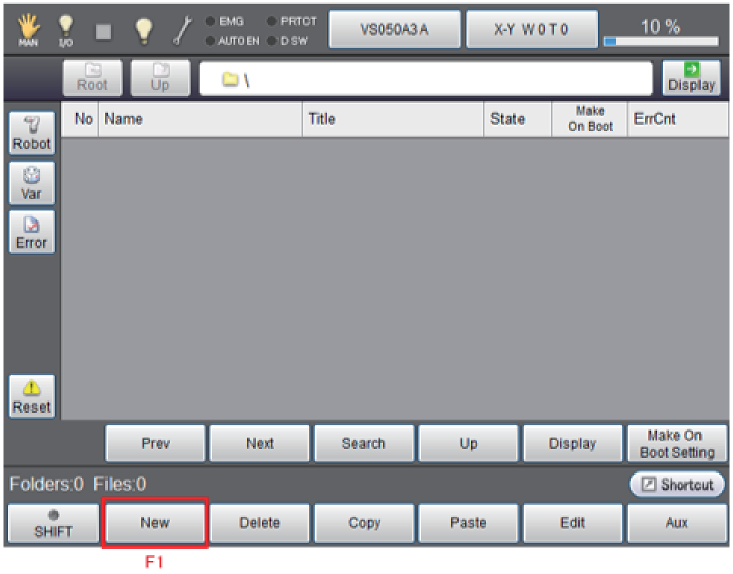
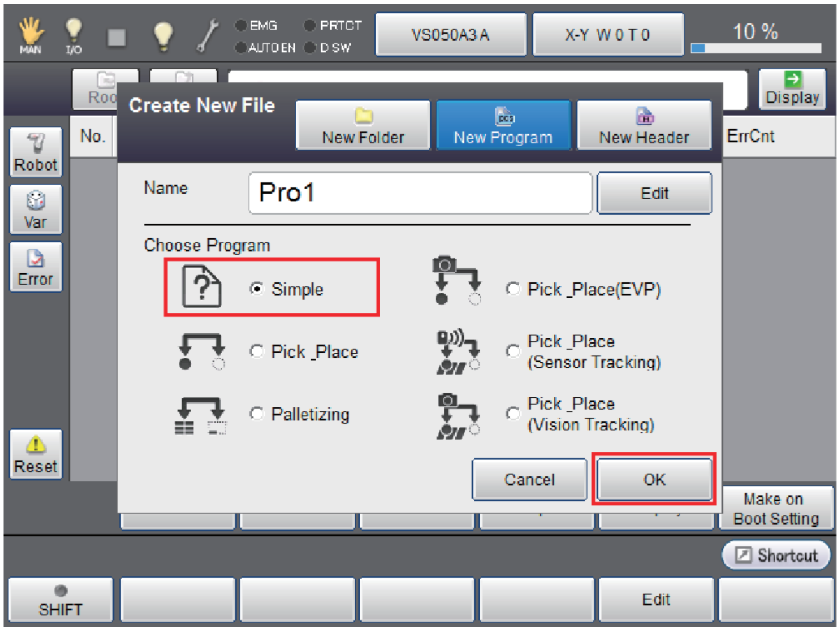


Figure 2: Step 2 of enter a new program name

(3) Step 3: Enter a program name and select [Simple] then press[OK]



Create New File

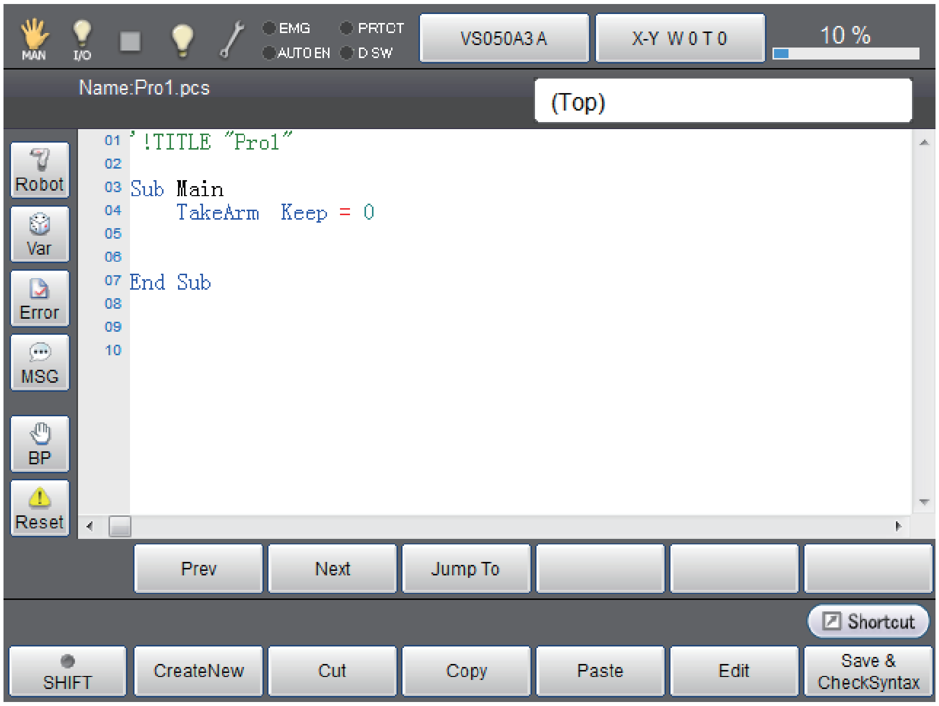
Name Pro1

Choose Program

- ● Simple
- ○ Pick_Place(EVP)
- ○ Pick_Place
- ○ Pick_Place (Sensor Tracking)
- ○ Palletizing
- ○ Pick_Place (Vision Tracking)

Figure 3: Step 3 of enter a new program name

(4) Step 4: This ends the preparation for program editing



```
' !TITLE "Pro1"

Sub Main
    TakeArm Keep = 0


End Sub
```

Figure 4: Step 4 of enter a new program name

For this lab exercise, create a program called Group#_Lab2#; replace the first # with your group number and the second # with A, B, … if you want to keep different versions of the program.

3.3 Teaching

Teaching refers to a method of programming in which you guide a robot through its motion using the teach pendant. In teaching, the robot is taught its motion.

In programming, you can specify positions as constants. However, in order to make the robot accurately learn the relative positional relationship between itself and objective point, you need to move the robot actually on site. Consequently, you write positions as variables in programming and assign actual values to those variables by on-site teaching.

Global Variables Available in Teaching

A variable refers to a program identifier foe a storage location which can contain any number or characters, and which can vary in a program. There are two types of variable: global variables and local variables. While global variables can be refereed from any programs, local variables are effective within respective programs. For teaching process, global variables are available.

The following three types of global variables are available in teaching.

- Pos. (Position variable) (X, Y, Z, RX, RY, RZ, FIG)
- Joint. (Joint variable) (J1, J2, J3, J4, J5, J6, J7, J8)
- Tran. (Homogeneous transform matrix variable) (X, Y, Z, 0x, 0y, 0z, Ax, Ay, Az, FIG)

Teaching to Position Variables

This section describes the method to teach the robot the position variables P1 and P2 in manual mode.

(1) Teaching Robot Position (P1)

Press [F2 Arm] on the teach pendant to display information on each axis of the arm. Press moving direction keys as required while pressing the deadman switch to move robot to the position to be set as P1.

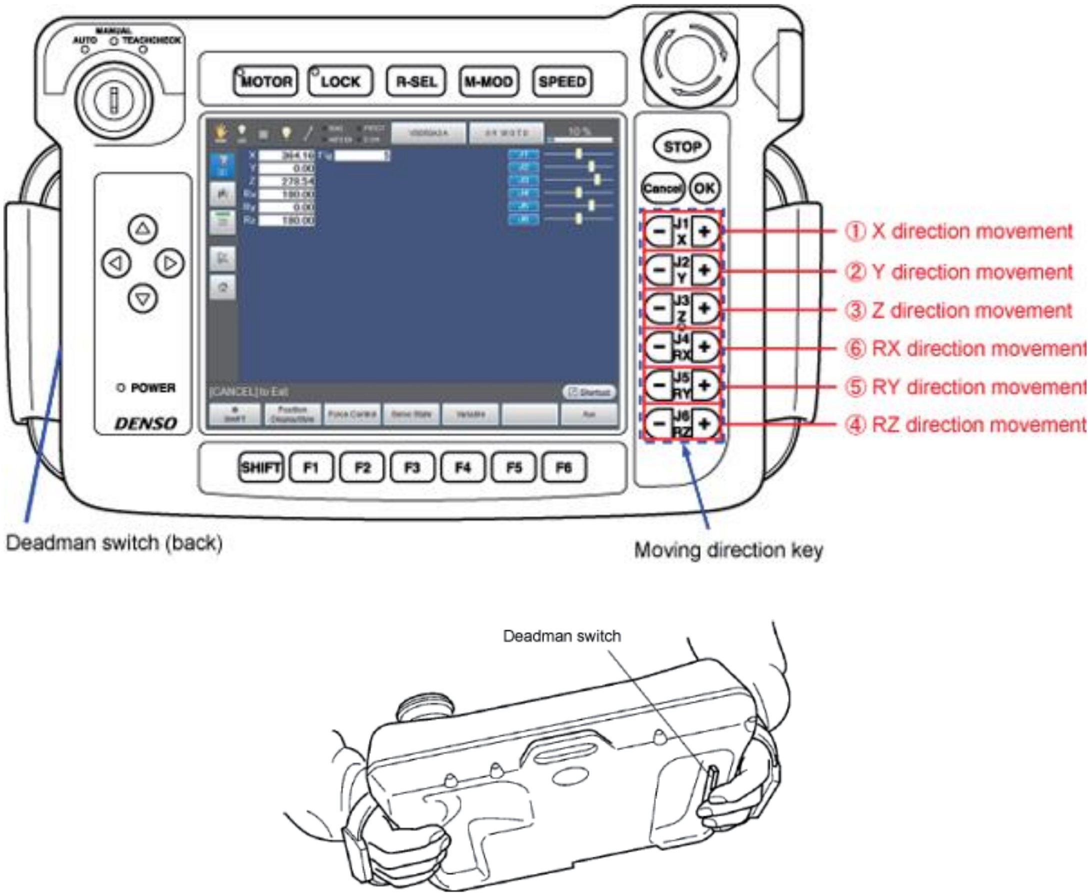


Figure 5: Teaching robot position

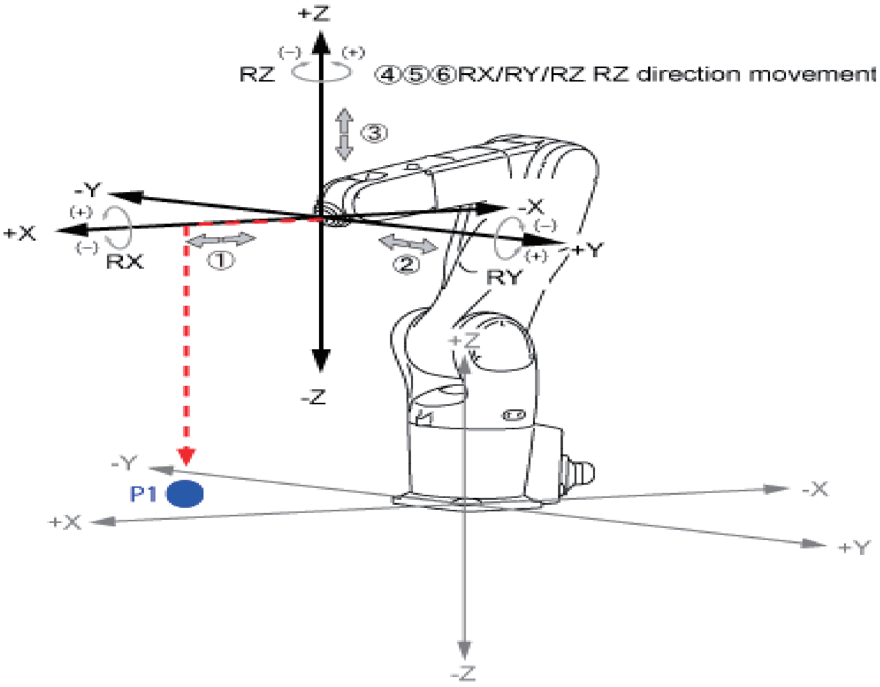
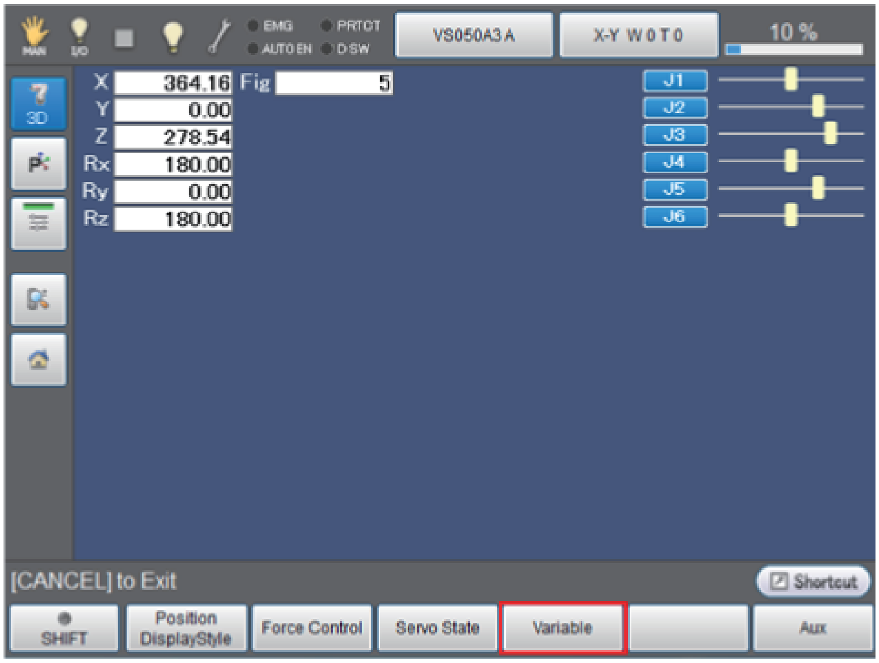


Figure 6: Movement of X-Y mode

(2) Saving the Teaching Value in Variable P1.

Follow the procedure below to save the teaching value in [VarName P1].

Step 1: Press [F4 Variable].



X 364.16
Fig 5
Y 0.00
Z 278.54
Rx 180.00
Ry 0.00
Rz 180.00

[CANCEL] to Exit

F4

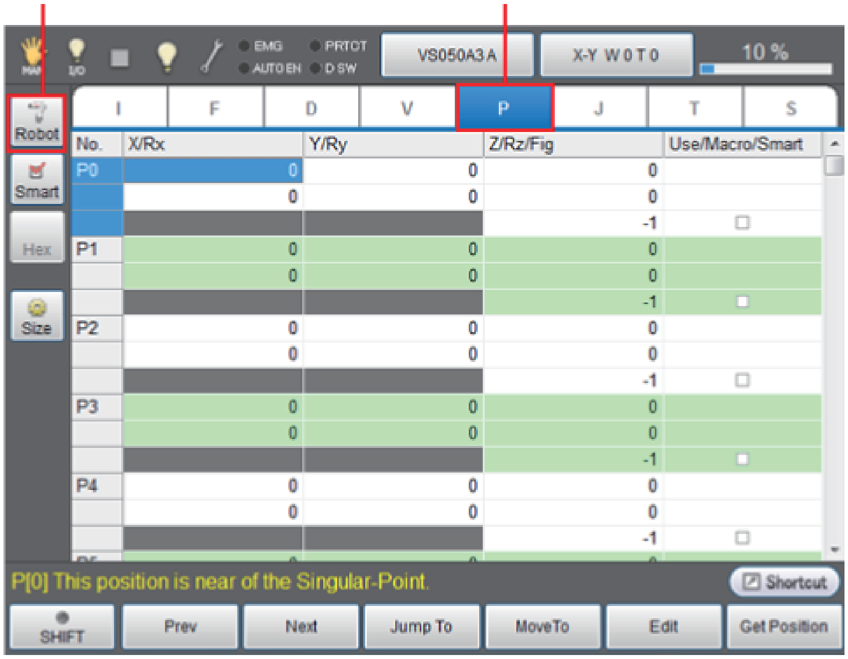
Figure 7: Step 1 of saving teaching value

Step 2: Select variable type in the variable window.

This procedure is for saving position in P variables; hence, press [P variable tab] on the screen.

Robot button

P variable tab



No.	X/Rx	Y/Ry	Z/Rz/Fig	Use/Macro/Smart
P0	0	0	0	
	0	0	0	
			-1	□
P1	0	0	0	
	0	0	0	
			-1	□
P2	0	0	0	
	0	0	0	
			-1	□
P3	0	0	0	
	0	0	0	
			-1	□
P4	0	0	0	
	0	0	0	
			-1	□

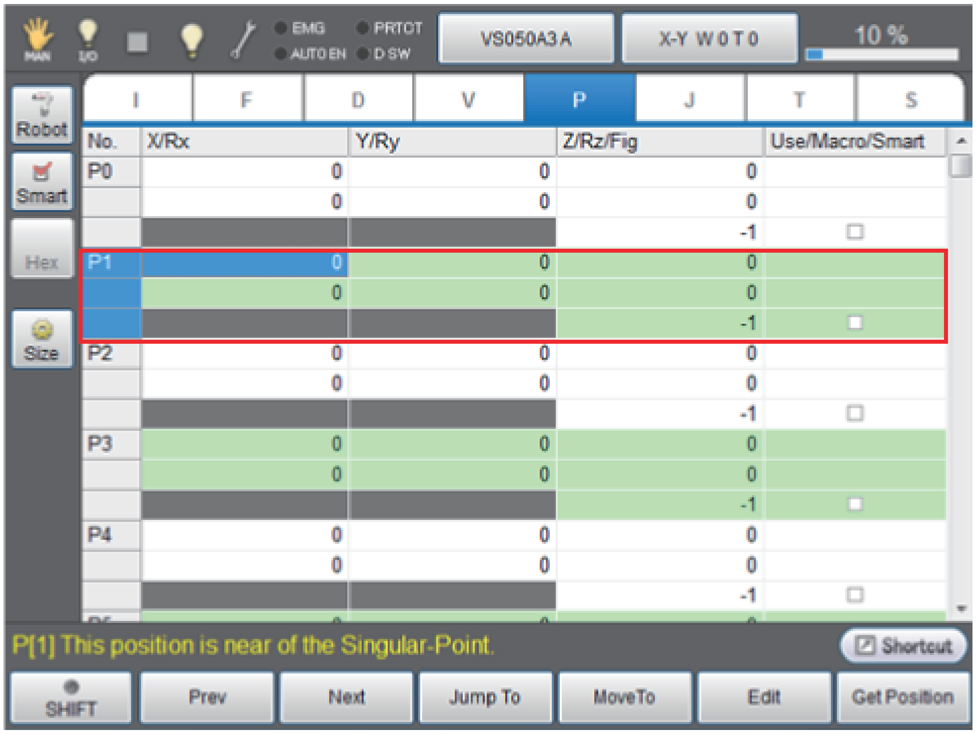
P[0] This position is near of the Singular-Point.

Figure 8: Step 2 of saving teaching value

TIP: If robot arm information is displayed on the right side of screen when variable window opens, the information can be deleted by pressing [Robot] button.

Step 3: Select the field of [VarName P1] by using cursor key or jog dial.

This can be also selected by directly pressing any of data field for variables name p1 on the screen. 12 types of data per one variable will be displayed in P variable screen as shown in the figure. If one of 12 types of data are highlighted when selecting [VarName p1], this means that [VarName P1] is selected.



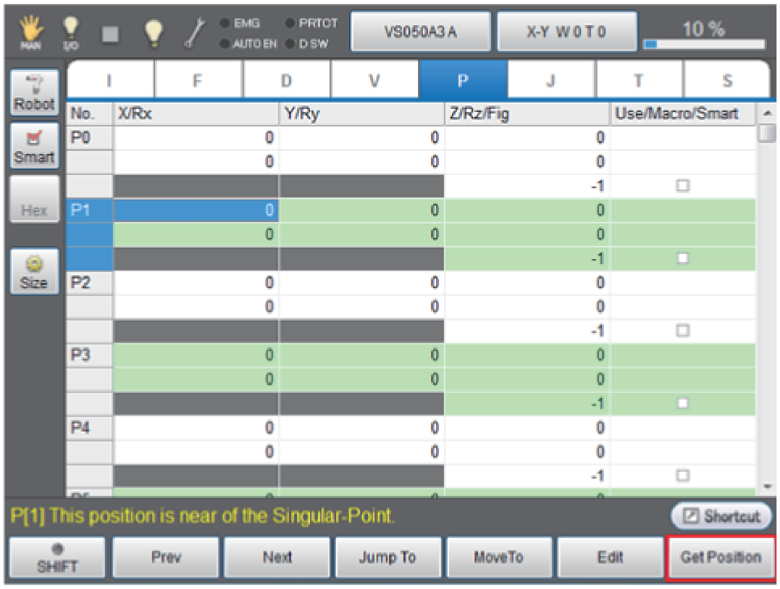
No.	X/Rx	Y/Ry	Z/Rz/Fig	Use/Macro/Smart
P0	0	0	0	
	0	0	0	
			-1	□
P1	0	0	0	
	0	0	0	
			-1	□
P2	0	0	0	
	0	0	0	
			-1	□
P3	0	0	0	
	0	0	0	
			-1	□
P4	0	0	0	
	0	0	0	
			-1	□

P[1] This position is near of the Singular-Point.

Figure 9: Step 3 of saving teaching value

Step 4: Confirm that [VarName P1] is selected.

Step 5: Press [F6 Get Position].



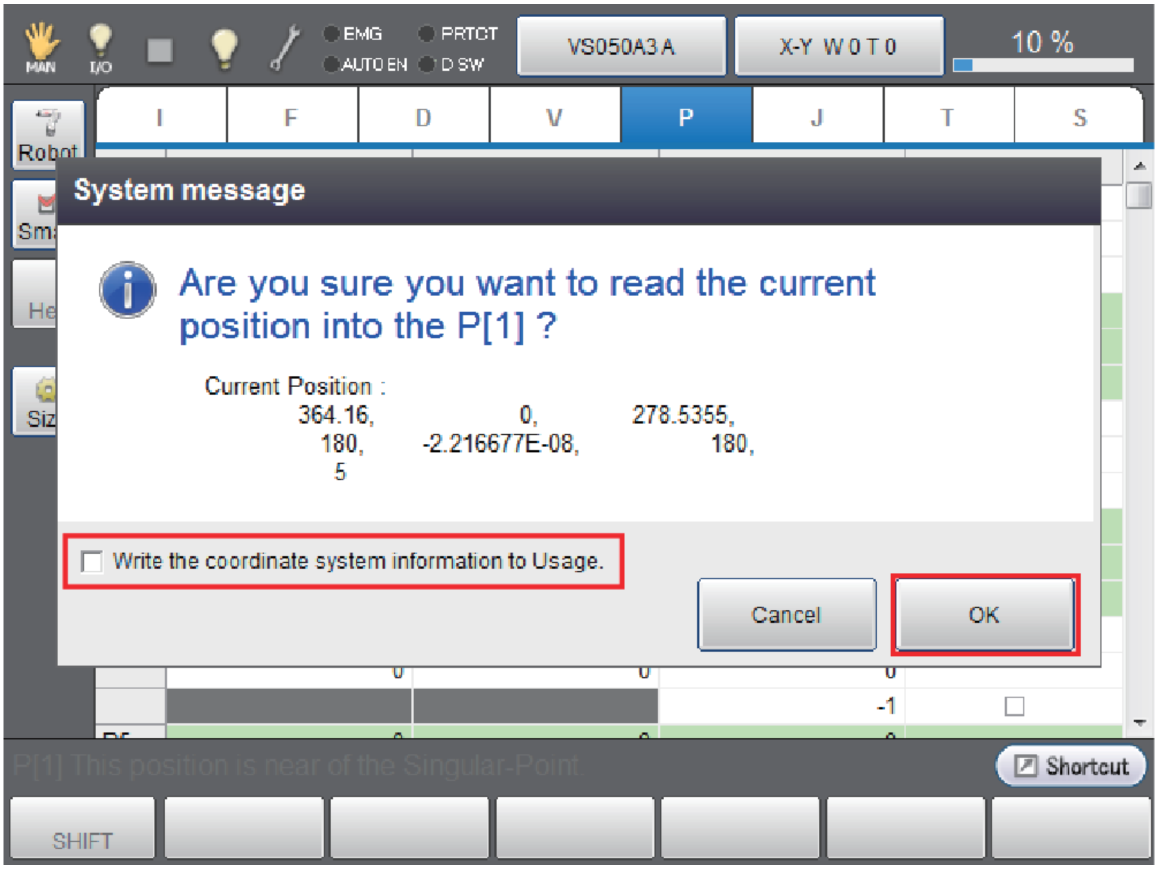
No.	X/Rx	Y/Ry	Z/Rz/Fig	Use/Macro/Smart
P0	0	0	0	
	0	0	0	
			-1	□
P1	0	0	0	
	0	0	0	
			-1	□
P2	0	0	0	
	0	0	0	
			-1	□
P3	0	0	0	
	0	0	0	
			-1	□
P4	0	0	0	
	0	0	0	
			-1	□

P[1] This position is near of the Singular-Point.

F6

Figure 10: Step 5 of saving teaching value

Step 6: Check the system message. If you agree the message, select a chechbox of "Write the coordinate system information to Usage". And then press [OK]. (This function is available in Ver.1.8.* or higher.)



System message

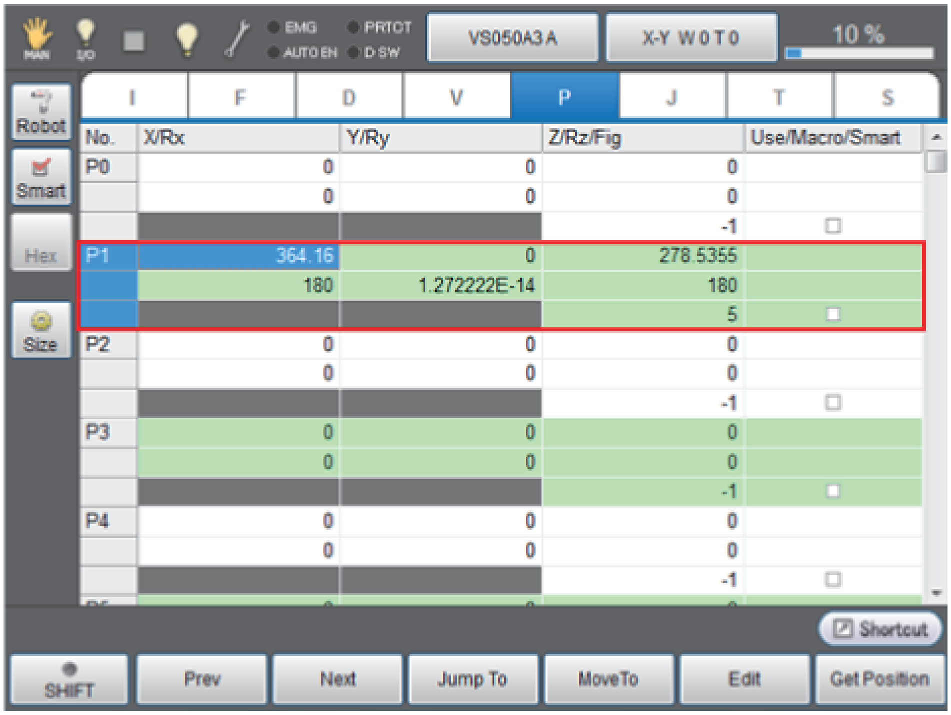
Are you sure you want to read the current position into the P[1] ?

Current Position :
364.16, 0, 278.5355,
180, -2.216677E-08, 180,
5

☐ Write the coordinate system information to Usage.

P[1] This position is near of the Singular-Point

The current position is loaded as variable P1. Coordinate system information is written in Usage (Use/Macro/Smart) cell.



No.	X/Rx	Y/Ry	Z/Rz/Fig	Use/Macro/Smart
P0	0	0	0	
	0	0	0	
			-1	□
P1	364.16	0	278.5355	
	180	1.272222E-14	180	
			5	□
P2	0	0	0	
	0	0	0	
			-1	□
P3	0	0	0	
	0	0	0	
			-1	□
P4	0	0	0	
	0	0	0	
			-1	□

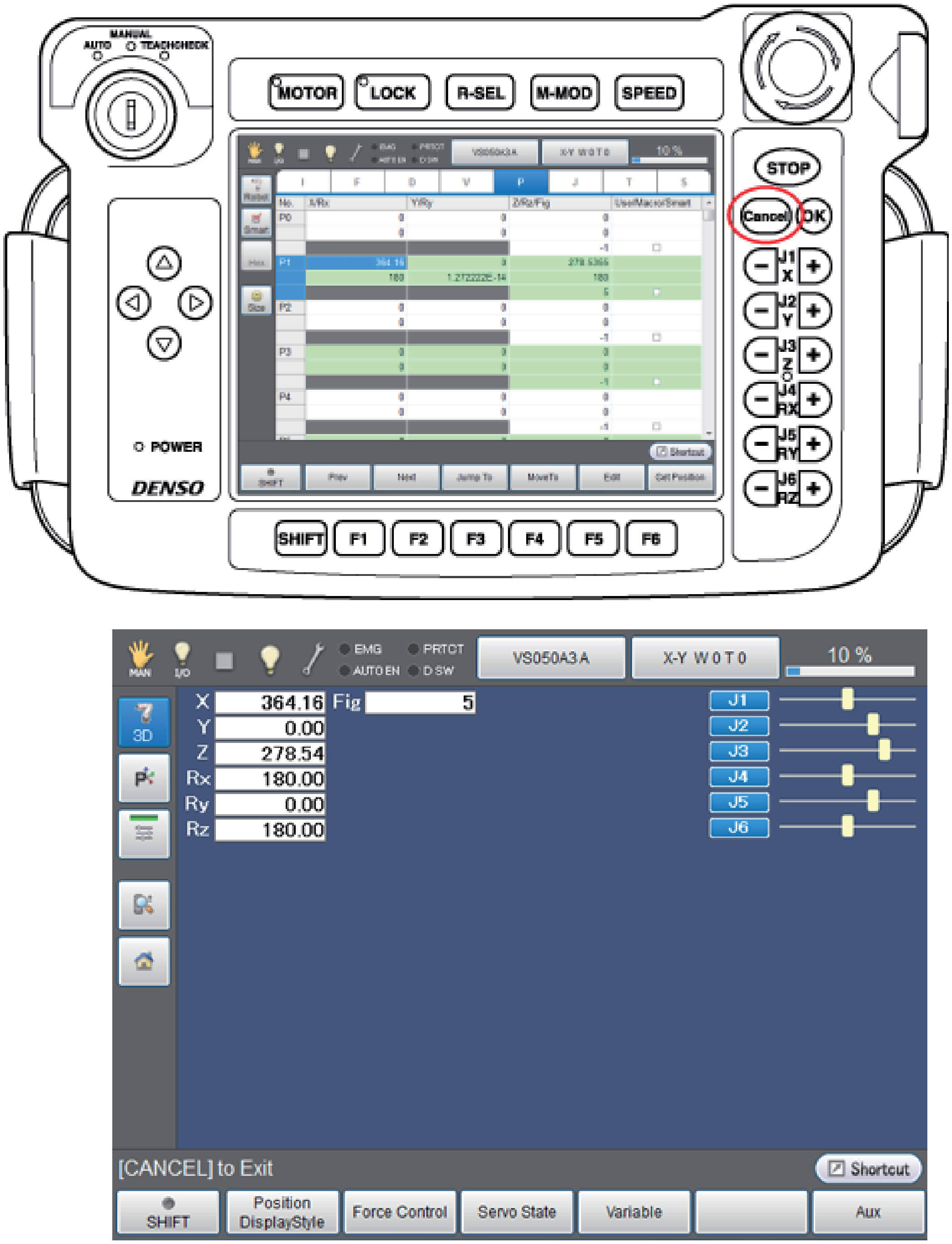
Figure 11: Step 6 of saving teaching value

TIP

- If any data is already written in the Usage cell, a new coordinate system information will overwrite it.
- When a controller is rebooted, the check box of “Write the coordinate system information to Usage” remains selected. (Ver.1.13. * or higher)
- To clear a coordinate system information in a Usage cell, select desired Usage cell, and press [F5 Edit].

(3) Teaching Robot Position (P2) and Saving it to [VarName P2]

Step 1: Press [Cancel] button once to go back to [Current Robot Position] window



X 364.16 Fig 5
Y 0.00
Z 278.54
Rx 180.00
Ry 0.00
Rz 180.00

[CANCEL] to Exit

Figure 12: Step 1 of teaching robot position (P2)

Step 2: Press moving direction keys as required while pressing the deadman switch to move robot to the position to be set as P2

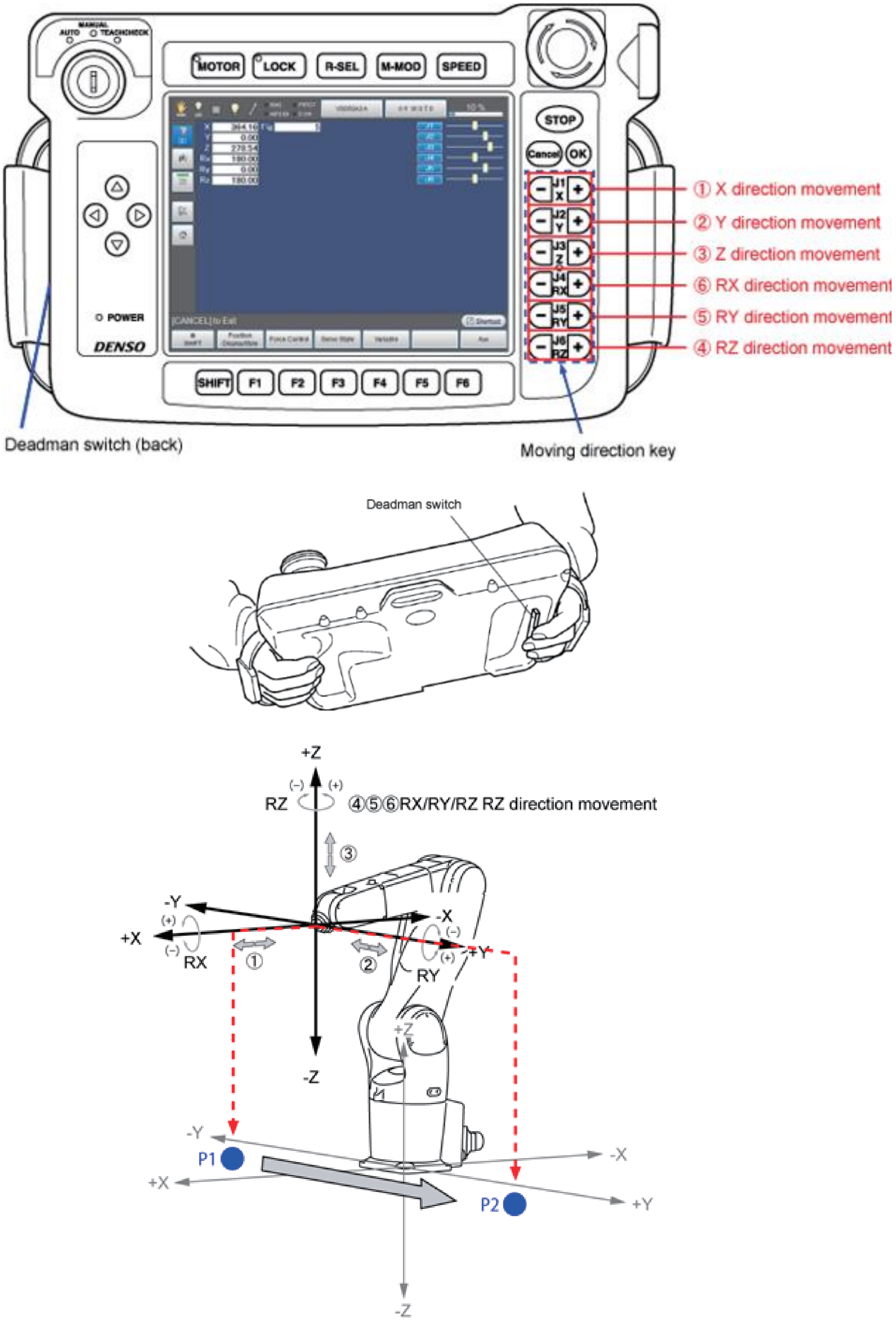


Figure 13: Step 2 of teaching robot position (P2)

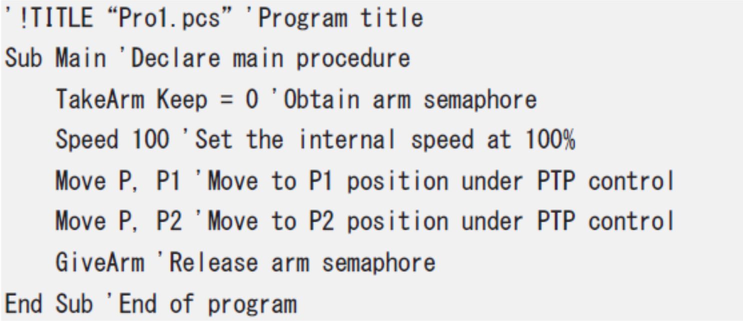
Step 3: Follow the procedure described in “Saving the teaching value in VarName P1” to save the value of P2 in [VarName P2].

Now, teaching of P1 and P2 is completed.

3.4 Entering Program Codes

Except using the jog manually save the position of point P1 and P2, we could create a program to move from point P1 to P2. In this step, you will create a program to mode from P1 to P2. Enter the program codes listed in the table below.

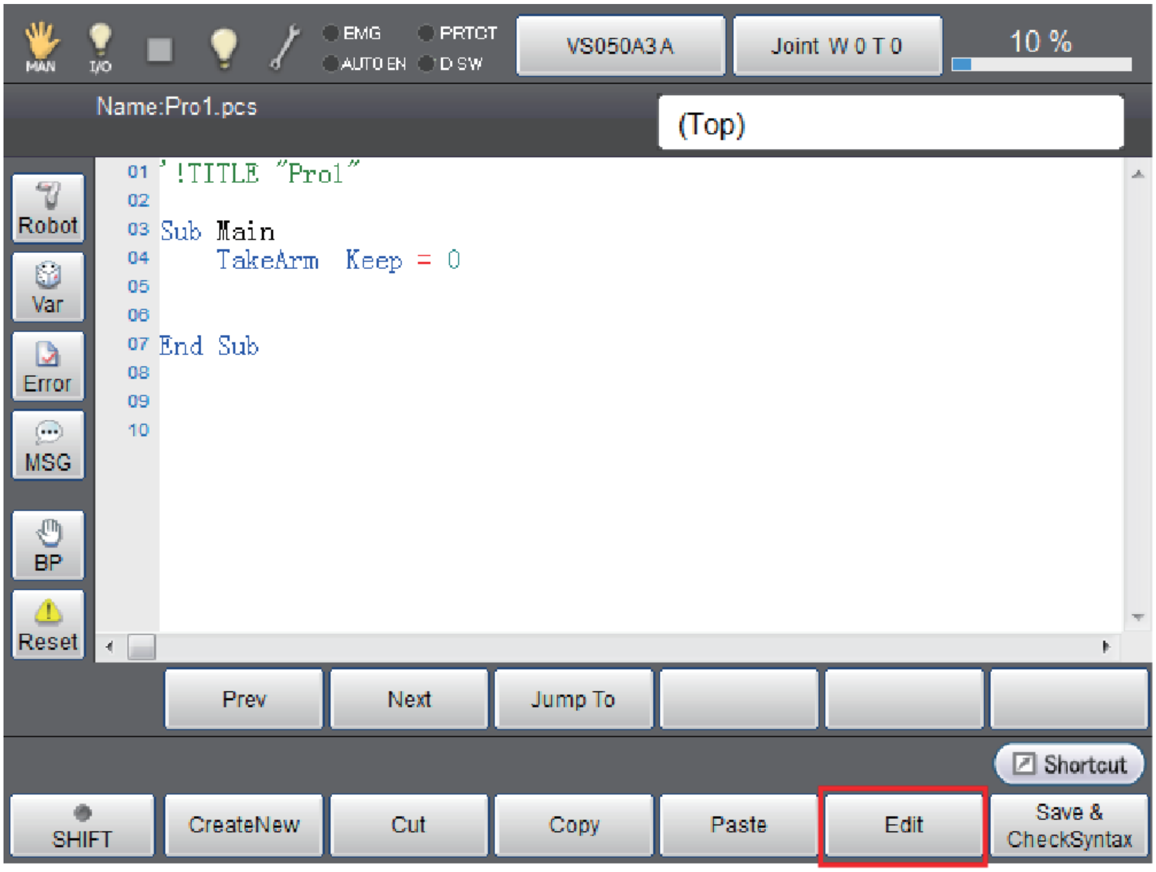
Program name PRO.1pcs



```
'!TITLE “Pro1.pcs” 'Program title
Sub Main 'Declare main procedure
    TakeArm Keep = 0 'Obtain arm semaphore
    Speed 100 'Set the internal speed at 100%
    Move P, P1 'Move to P1 position under PTP control
    Move P, P2 'Move to P2 position under PTP control
    GiveArm 'Release arm semaphore
End Sub 'End of program
```

(1) Step 1: Move the cursor to the 5th line on the program edit window by using Up/Down key or jog dial.

(2) Step 2: Press [F5 Edit]



Name:Pro1.pcs

(Top)

```
'!TITLE "Pro1"

Sub Main
    TakeArm  Keep = 0


End Sub
```

F5

Keyboard appears.

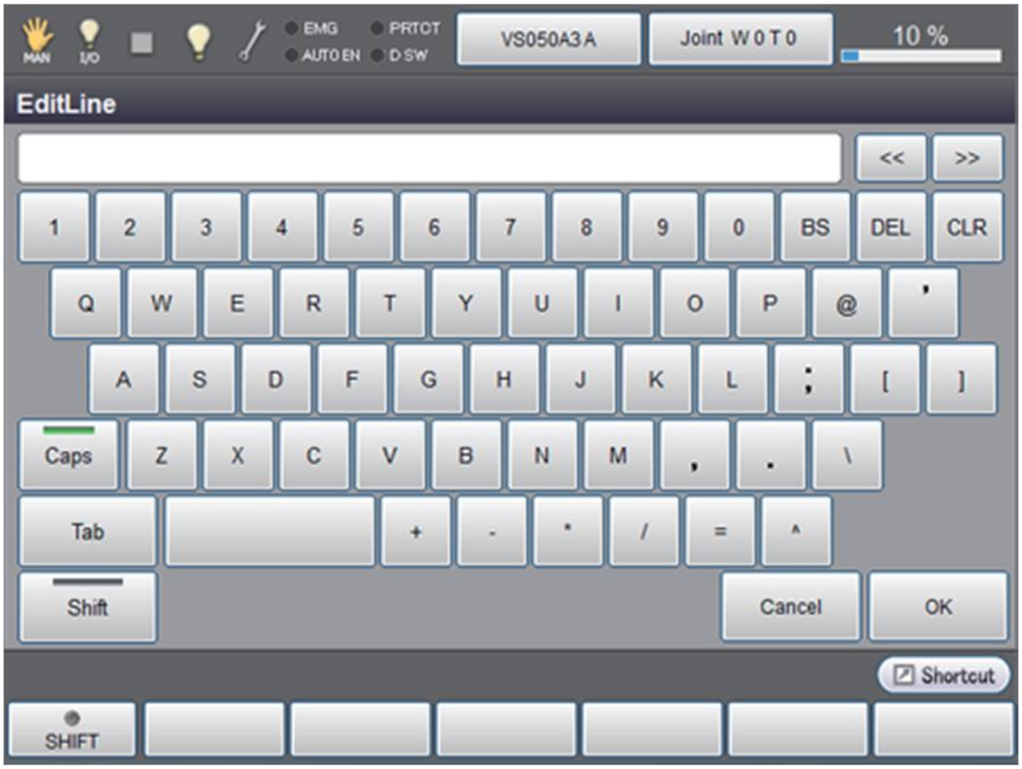
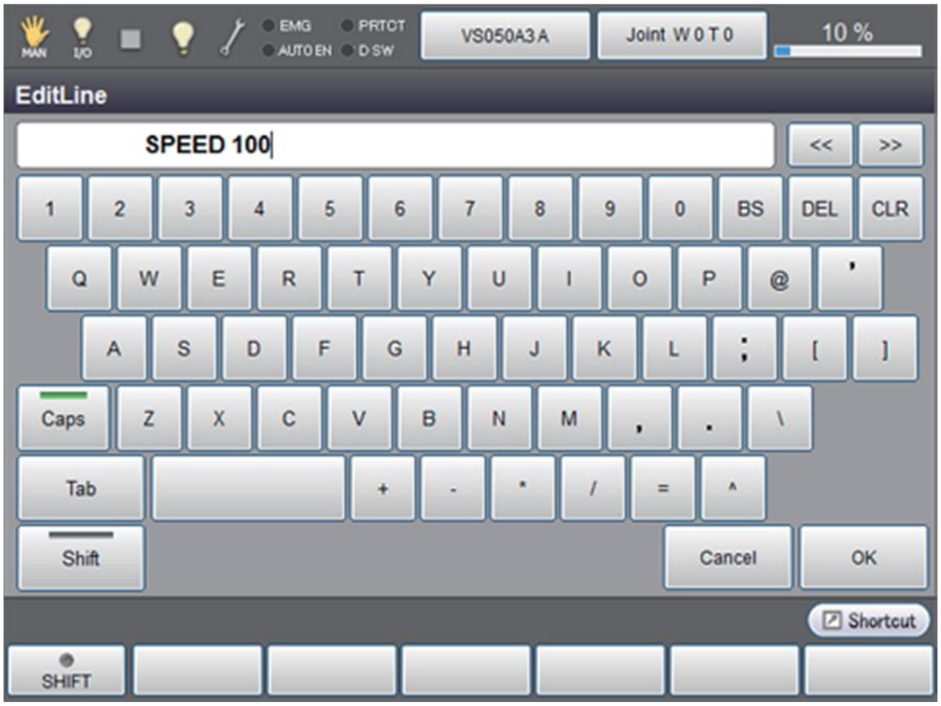


Figure 14: Step 2 of entering new code

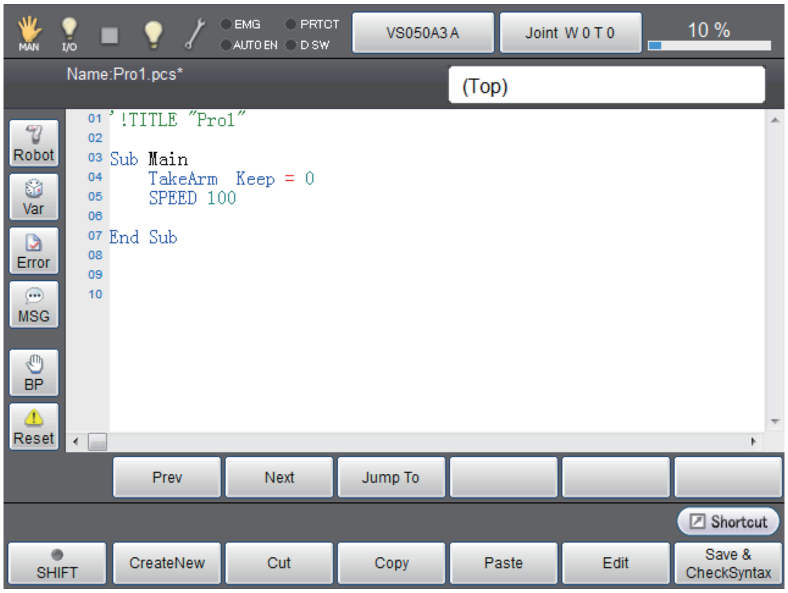
(3) Step 3: Enter [SPEED 100] from the keyboard then press [OK].



SPEED 100

Figure 15: Step 3 of entering new code

(4) Step 4: The program edit window "Pro1.pcs" is displayed and "SPEED 100" is displayed in the 5th line.



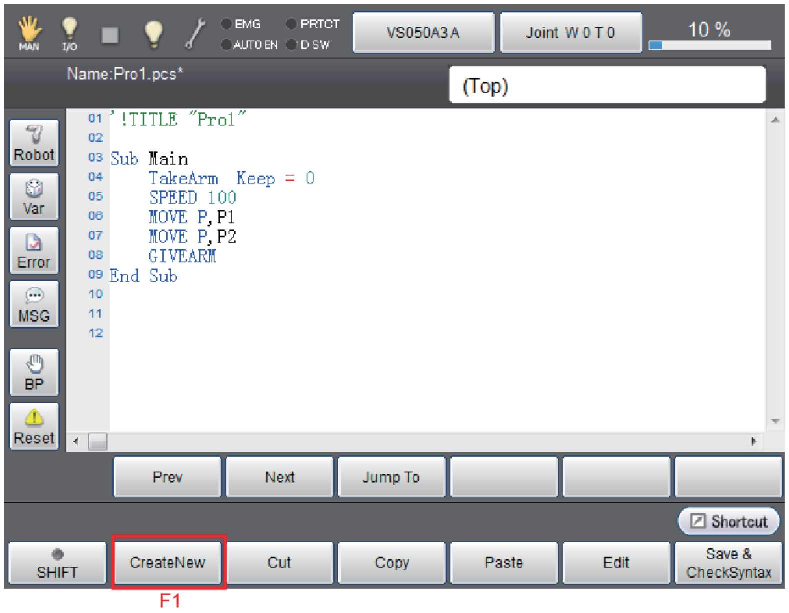
Name:Pro1.pcs*

(Top)

```
01 '!TITLE "Pro1"
02
03 Sub Main
04     TakeArm  Keep = 0
05     SPEED 100
06
07 End Sub
08
09
10
```

Figure 16: Step 4 of entering new code

(5) Step 5: Enter all of the program codes given in the same way used to enter "100". To add a new line, press [F1 Create New]



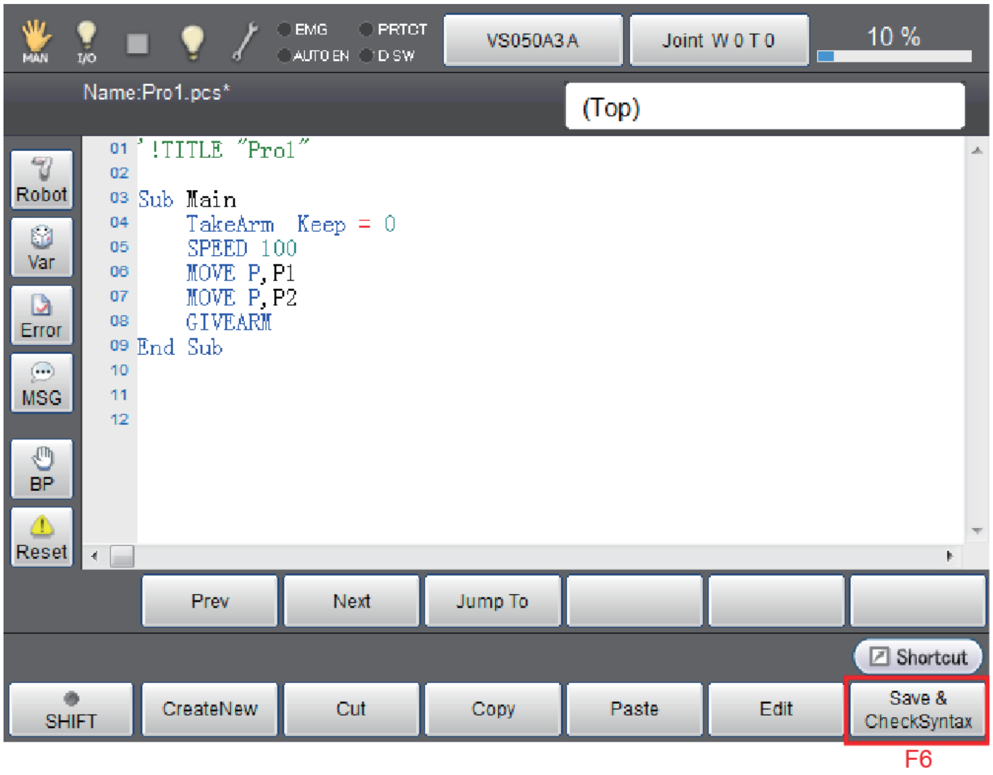
Name:Pro1.pcs*

(Top)

```
01 '!TITLE "Pro1"
02
03 Sub Main
04     TakeArm  Keep = 0
05     SPEED 100
06     MOVE P,P1
07     MOVE P,P2
08     GIVEARM
09 End Sub
10
11
12
```

Figure 17: Step 5 of entering new code

(6) Step 6: After completing entry of all codes, press [F6 Save & CheckSyntax]



Name:Pro1.pcs*

(Top)

```
'!TITLE "Pro1"

Sub Main
    TakeArm  Keep = 0
    SPEED 100
    MOVE P,P1
    MOVE P,P2
    GIVEARM
End Sub
```

Figure 18: Step 6 of entering new code

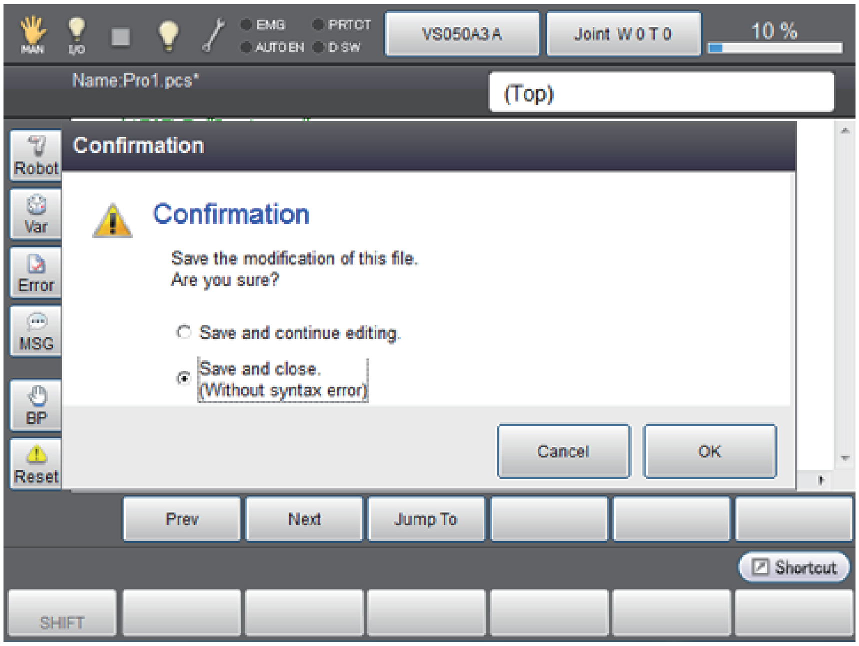
(7) Step 7: Select whether to continue editing after saving data.

Save and continue editing----Go to the program edit window

Save and close (Without syntax error) ----Move to the program list window

In this example, select "Save and close (Without syntax error)" then press [ok]

If an error is detected, syntax error window appears.



Name:Pro1.pcs*

(Top)

Confirmation

Confirmation

Save the modification of this file.
Are you sure?

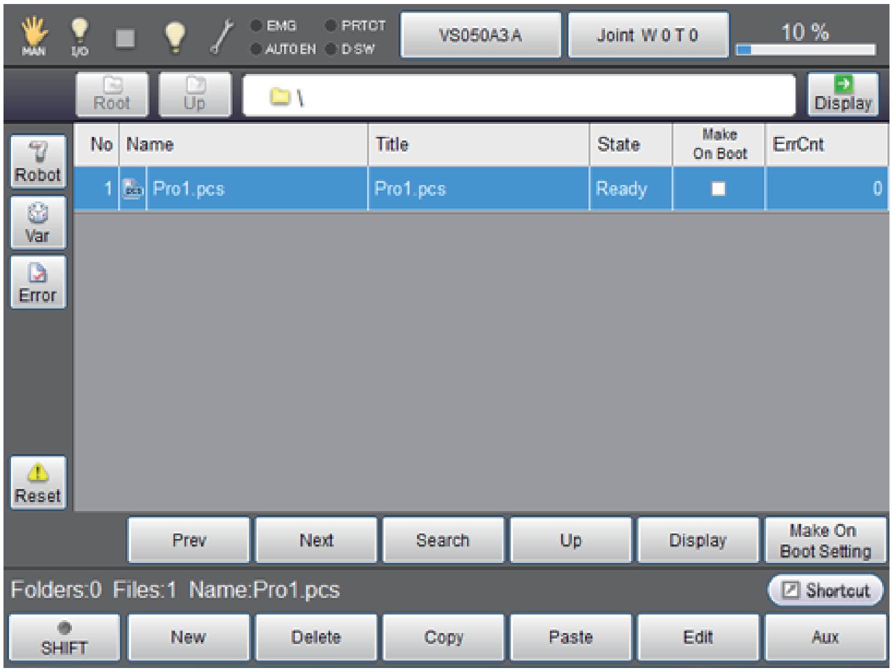
○ Save and continue editing.

● Save and close.
(Without syntax error)

Figure 19: Step 7 of entering new code

When the program is saved successfully, an asterisk at the end of program name will be erased.

(8) Step 8: The display will return to the Program List window



\

No	Name	Title	State	Make On Boot	ErrCnt
1	Pro1.pcs	Pro1.pcs	Ready	□	0

Folders:0 Files:1 Name:Pro1.pcs

Figure 20: Step 8 of entering new code

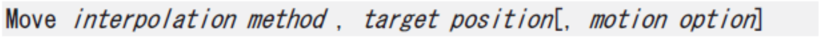
TIP: To execute the syntax check in program list, press [SHIFT]+[F 12 Check Syntax]

3.5 Movement to the Specified Coordinates (Move Command)

Function

This statement moves the robot from the current position to the target position

Syntax



```
Move interpolation method ,  target position[, motion option]
```

“Interpolation method” and “Target position” must be input.” Motion option” is optional.

Interpolation Method

When the robot arm moves, there is not just one path. Various paths can be created with the operation of each axis. The robot can be controlled so that it creates line or circle paths. Interpolation method must be chosen in Move command.

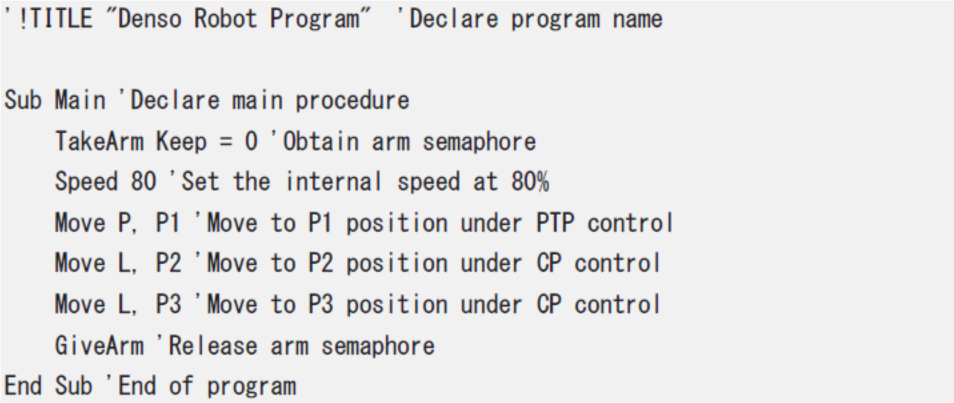
- P: PTP (Point to Point) control
  When moving the robot arm from the current position to the target position, the robot decides the route.
- L: CP (Continuous Path) control—linear interpolation
  When moving the robot arm from the current position to the target position, the robot keeps the position and speed of the hand constant.
- C: CP (Continuous Path) control—arc interpolation
  When moving the robot arm from the current position to the target position, the robot moves its hand along the 3-point curve.
- S: S(Spline) control

When moving the robot arm from the current position to the target position, the robot moves its hand with smooth interpolated motion among specified passing points.

Target Position

Target Position can have any of the position, joint, or homogeneous transform matrix type to which a target position should be assigned. Destination position must be input.

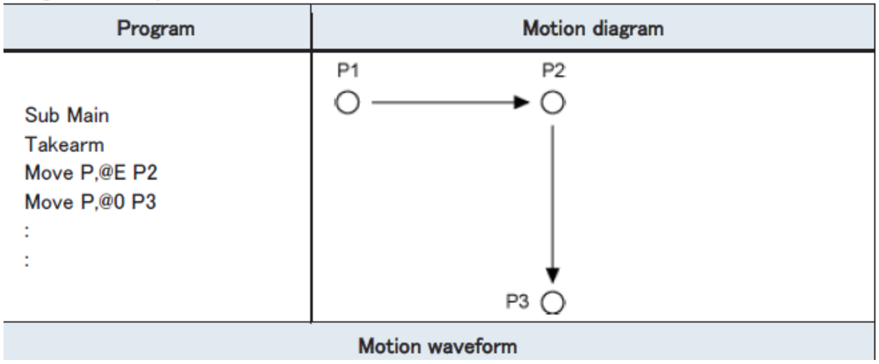
Example:



```
'!TITLE "Denso Robot Program"  'Declare program name

Sub Main 'Declare main procedure
    TakeArm Keep = 0 'Obtain arm semaphore
    Speed 80 'Set the internal speed at 80%
    Move P, P1 'Move to P1 position under PTP control
    Move L, P2 'Move to P2 position under CP control
    Move L, P3 'Move to P3 position under CP control
    GiveArm 'Release arm semaphore
End Sub 'End of program
```

Target position option includes Pass start displacement and Extended-joints option for target position. Pass start displacement is the radius of a sphere whose center is located at the destination position, and it is expressed in units of mm. When the commanded motion value reaches the sphere, control passes to the next one. In other words, this value determines how to stop at the specified point. End motion, encoder value check motion, or pass motion can be selected as control transfer to the next statement.

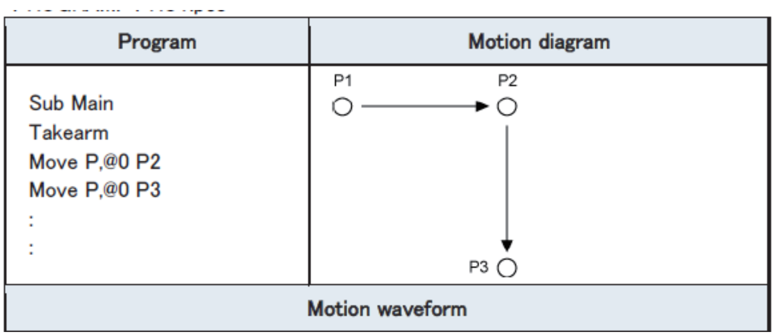
Encoder Value Check Motion



Program	Motion diagram
Sub Main Takearm Move P,@E P2 Move P,@0 P3 : :	
Motion waveform	

The encoder value check motion is to judge that the encoder value has arrived at the taught target position(p2). Although this motion offers highly accuracy of stopping, it takes longer time than the end motion to eliminate the serve deviation.

End Motion

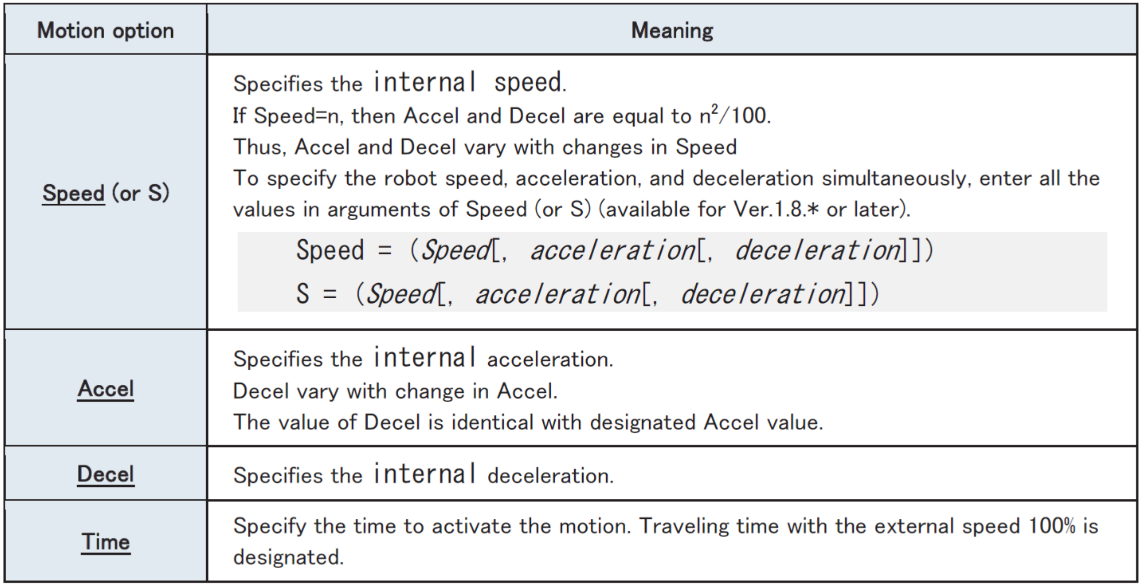


Program	Motion diagram
Sub Main Takearm Move P,@0 P2 Move P,@0 P3 : :	
Motion waveform	

In the end motion, the robot judges that the tool end has arrived at the target position when it reaches the taught position P2 (called as the end position) and the command value to the servo system becomes the target one. The motion to the P3 starts when the commanded motion value reaches P2. When comparing the motor command value to the encoder value, the commanded motion value goes ahead.

Speed Setting

Speed setting option is any of Speed, Accel, Decel, or Time.

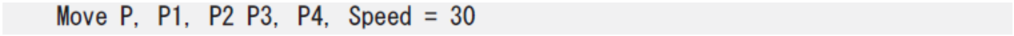


Motion option	Meaning
Speed (or S)	Specifies the internal speed. If Speed=n, then Accel and Decel are equal to n^2/100. Thus, Accel and Decel vary with changes in Speed To specify the robot speed, acceleration, and deceleration simultaneously, enter all the values in arguments of Speed (or S) (available for Ver.1.8.* or later). Speed = (*Speed*[, *acceleration*[, *deceleration*]]) S = (*Speed*[, *acceleration*[, *deceleration*]])
Accel	Specifies the internal acceleration. Decel vary with change in Accel. The value of Decel is identical with designated Accel value.
Decel	Specifies the internal deceleration.
Time	Specify the time to activate the motion. Traveling time with the external speed 100% is designated.

Input Example:

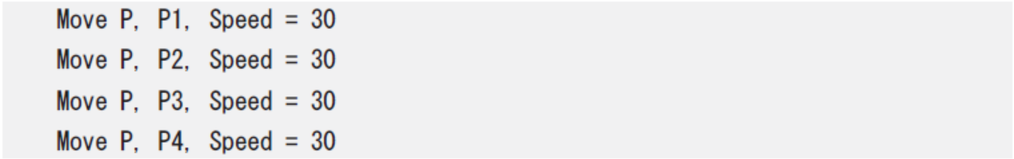
Example 1

A continuous motion specified with two points or more can be written in one line.

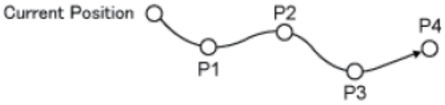


```
    Move P, P1, P2 P3, P4, Speed = 30
```

The above statement is the same as the following.



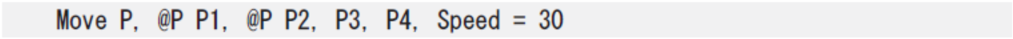
```
    Move P, P1, Speed = 30
    Move P, P2, Speed = 30
    Move P, P3, Speed = 30
    Move P, P4, Speed = 30
```



Note: A single step contains all motions up to P4. A step stop operation, therefore, can not stop the motion in midstream, such as at P1, P2 or P3.

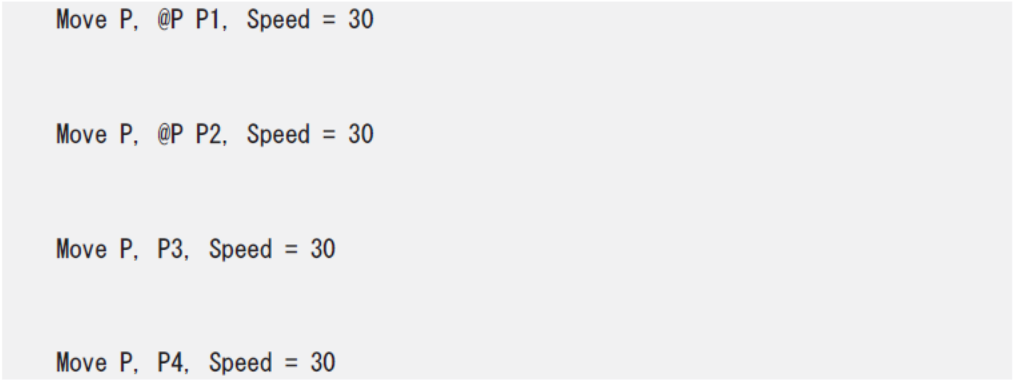
Example 2

Set the target position options for each target position.



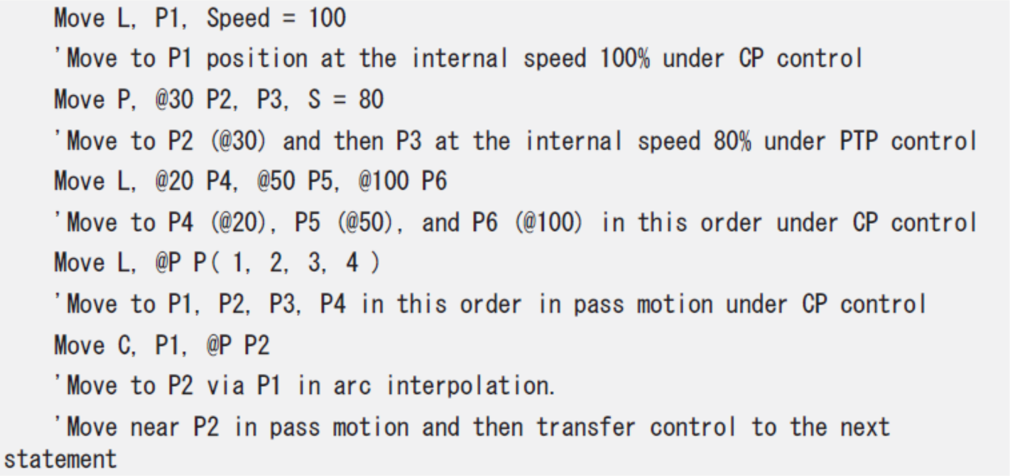
```
    Move P, @P P1, @P P2, P3, P4, Speed = 30
```

The above statement is the same as the following.



```
    Move P, @P P1, Speed = 30


    Move P, @P P2, Speed = 30


    Move P, P3, Speed = 30


    Move P, P4, Speed = 30
```

Example 3:



```
    Move L, P1, Speed = 100
    'Move to P1 position at the internal speed 100% under CP control
    Move P, @30 P2, P3, S = 80
    'Move to P2 (@30) and then P3 at the internal speed 80% under PTP control
    Move L, @20 P4, @50 P5, @100 P6
    'Move to P4 (@20), P5 (@50), and P6 (@100) in this order under CP control
    Move L, @P P( 1, 2, 3, 4 )
    'Move to P1, P2, P3, P4 in this order in pass motion under CP control
    Move C, P1, @P P2
    'Move to P2 via P1 in arc interpolation.
    'Move near P2 in pass motion and then transfer control to the next
statement
```

3.6 Operating the gripper from the program

Electric Gripper Status

Use the command or the electric gripper screen to read the electric gripper status.

-1: Light On

0: Light Off

(1) Emergency stop status (Hand[n].EmgState)

This is the external emergency stop status.

-1: The emergency stop condition is cleared. (Emergency stop input shorted.)

0: An emergency stop condition has

(2) Motor

This is motor power status

-1: Motor power ON

0: Motor power OFF

(3) Running

This is signal indicates that the electric gripper control board is operating.

(4) INPOS

The signal indicates that the electric gripper is at the target position.

(5) Hold

This signal indicates that the fingers are gripping a workpiece.

-1: The workpiece is gripped with the set gripping force.

0: The electric gripper is not gripping.

Note: If the workpiece is gripped at an angle and the fingers then move, the HOLD signal turns OFF.

(6) Origin return completion

This signal indicates that the origin return is complete.

-1: Origin return is complete.

0: Origin return is not complete

(7) Zone

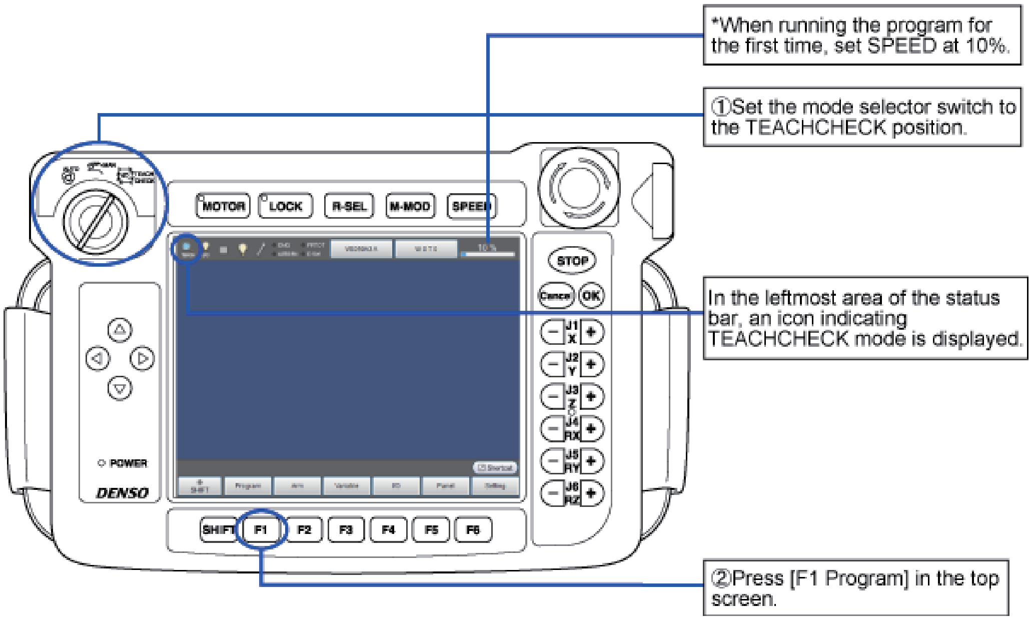
This signal indicates that the electric gripper is gripping inside the specified range. Constant speed movement/grip with ZON (close/open)

-1: The electric gripper is gripping between range specification 1 and range specification 2.

0: The electric gripper is stopped outside the range specification

3.7 Run the Program

Select the Teach Check Mode



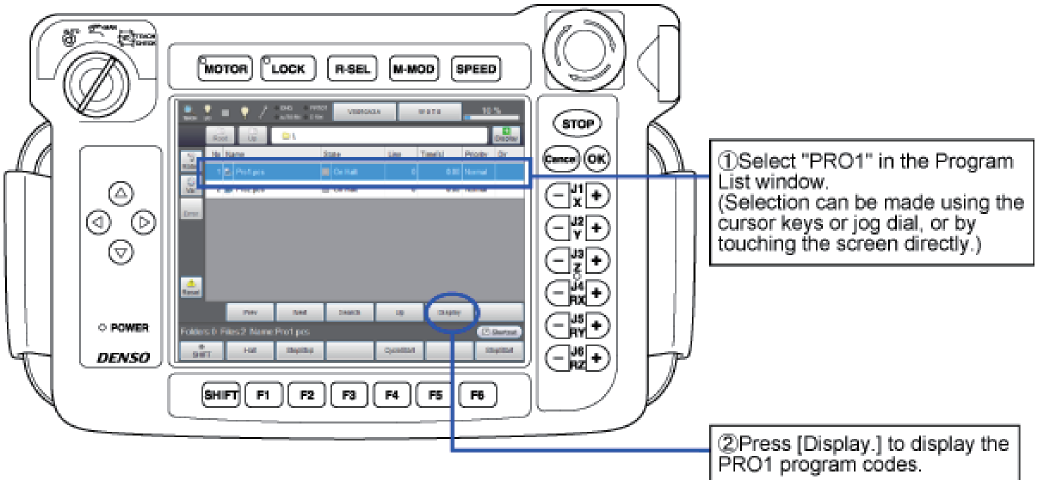
*When running the program for the first time, set SPEED at 10%.

①Set the mode selector switch to the TEACHCHECK position.

In the leftmost area of the status bar, an icon indicating TEACHCHECK mode is displayed.

②Press [F1 Program] in the top screen.

Select a Program to be Executed



①Select "PRO1" in the Program List window.
(Selection can be made using the cursor keys or jog dial, or by touching the screen directly.)

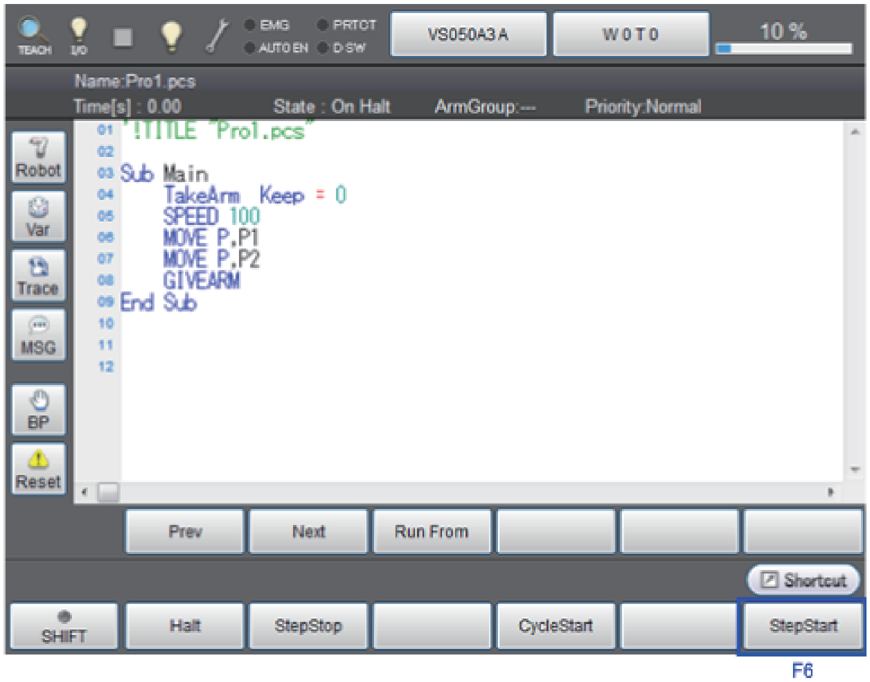
②Press [Display.] to display the PRO1 program codes.

Step Start: In the step start, the program executes a single step at a time.

(1) Step 1: While holding down the deadman switch, turn the motor power on.

Note: Check that the machine lock is released. The motor power does not turn ON if the machine is locked.

(2) Step 2: Press [F6 StepStart]



Name:Pro1.pcs
Time[s] : 0.00 State : On Halt ArmGroup:--- Priority:Normal

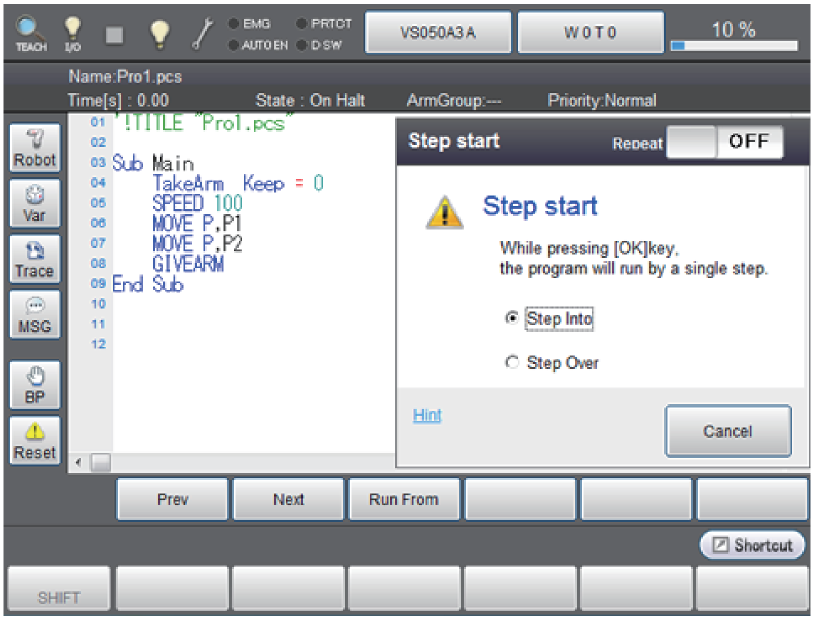
```
'!TITLE "Pro1.pcs"

Sub Main
    TakeArm Keep = 0
    SPEED 100
    MOVE P,P1
    MOVE P,P2
    GIVEARM
End Sub
```

Figure 9: Step 2 of run the program

(3) Step 3: The system message appears on the right side of the screen

- StepIn: Run into the procedure called from current program
- StepOver: Run over the procedure called from current program

- Repeat: If this option is “ON”, StepStart window appears after a cycle of StepStart completes. This option is practical when executing StepStart repeatedly.



Name:Pro1.pcs
Time[s] : 0.00 State : On Halt ArmGroup:--- Priority:Normal

```
!TITLE "Pro1.pcs"

Sub Main
    TakeArm  Keep = 0
    SPEED 100
    MOVE P,P1
    MOVE P,P2
    GIVEARM
End Sub
```

Step start Repeat OFF

Step start

While pressing [OK]key,
the program will run by a single step.

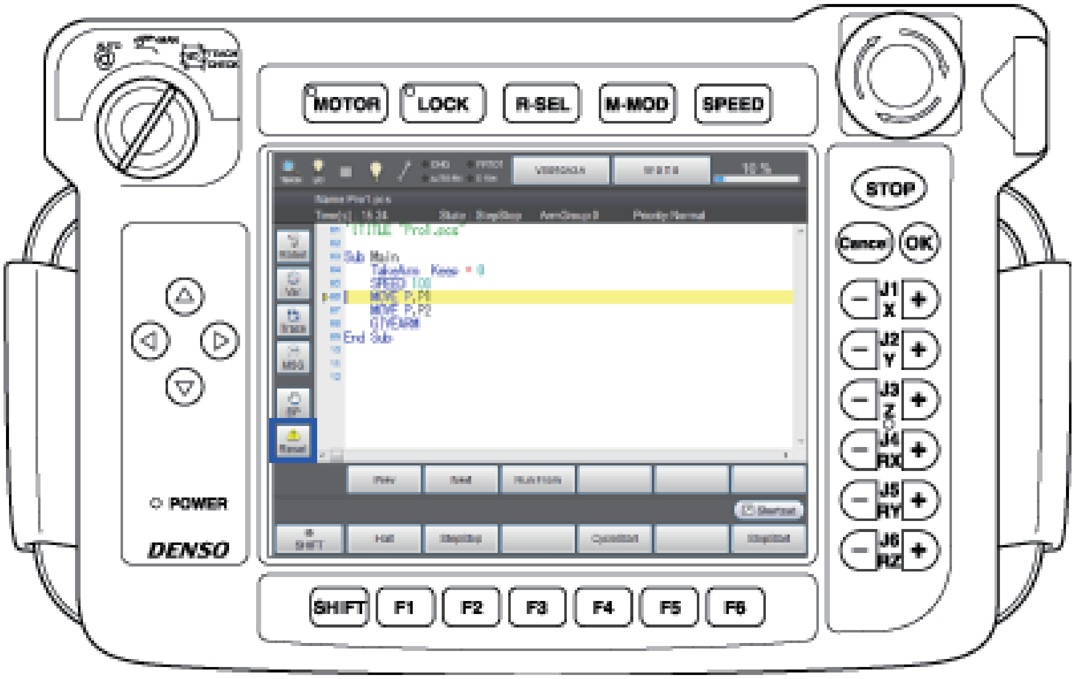
● Step Into

○ Step Over

Figure 10: Step 3 of run the program

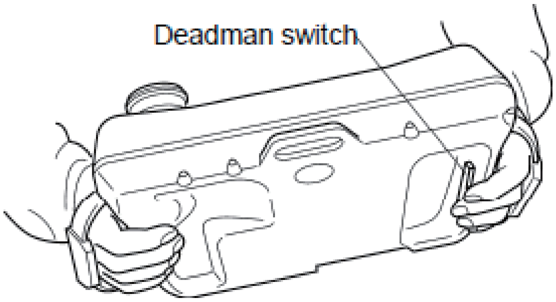
(4) Step 4: To execute a step start of the next line, press OK for a while with holding down the deadman switch.

(5) Step 5: To stop the robot image motion during step start, If the [OK] button is released while step start is running, the program will be suspended state. In TEACHCHECK mode, keep pressing both deadman switch and [OK]button until the execution of program is completed. Robot stops immediately if either of them is released before completing the execution. To reset the program which is under execution, press [RESET]button.



```
'!TITLE "Pro1.pcs"

Sub Main
    TakeArm Keep = 0
    SPEED 100
    MOVE P, P1
    MOVE P, P2
    GIVEARM
End Sub
```

Figure 11: Step 5 of run the program



3.8 Debug the Program

The “Simulating a Program Operation” method enables to simulate the robot motion of created programs by using teach pendant, without moving actual robot.

4 Stacking Blocks

Write a program to stack six geometrically identical Delrin (or metal) blocks in the configuration shown in the figure below. Do the basic one in the figure below, if you want, you could do the challenging one.

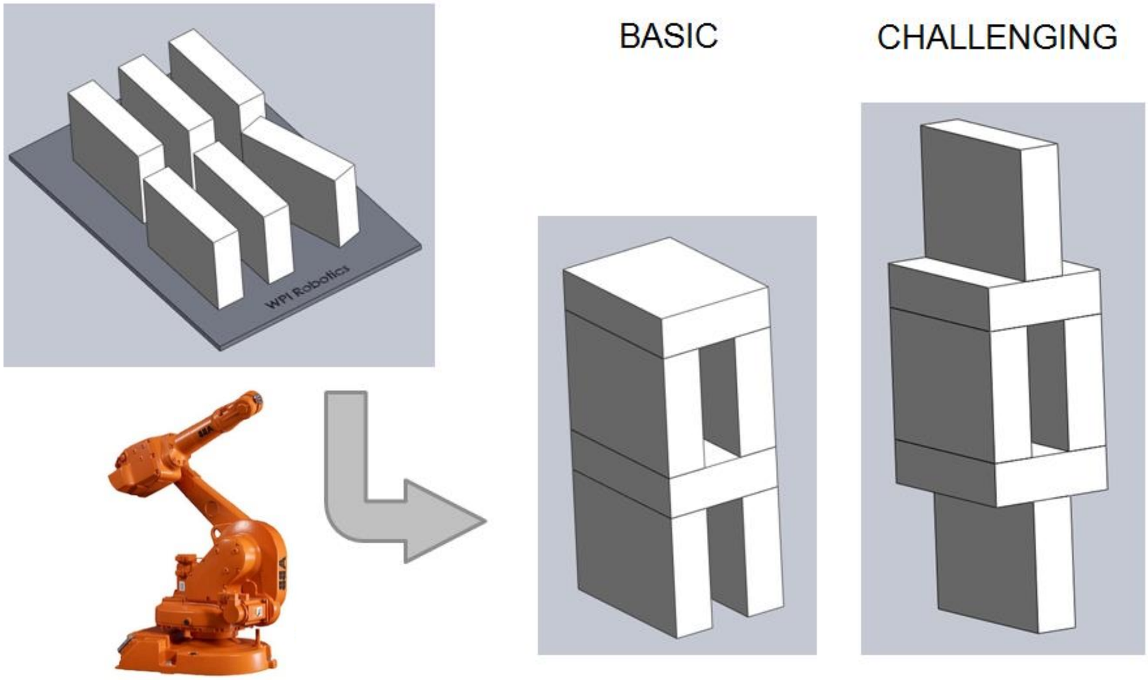


Figure 12: Possible block stacking configurations

Hints:

- Make sure the plate used to hold the blocks is always aligned properly before starting operations. The plate is aligned so that most of the blocks are oriented parallel to the robot's base Y axis. [Use the supplied marks on the table to line things up properly.
- Most of the block moves can be accomplished by gripping them from the top. Consider getting the gripper oriented vertically and then jog it (mainly) using linear moves in the base coordinate system.
- For some of the blocks, you may need more complex movements. Don't try to accomplish too much with a single move instruction. It's important to remember that, by default, the controller will flag an error if you try to rotate an axis more than 90° in a single linear move instruction. Use multiple moves if necessary.

- There may be occasions where it might be more appropriate to use joint moves instead of linear moves. Joint moves are generally less likely to flag errors if you try to accomplish too much in a single move instruction.
- Keep the speeds low – this will give you more time to react if something isn't right. This is particularly important when approaching the blocks to grip them.
- Work on getting the code for moving the first block working correctly. Once you have that written and debugged, you can then copy those lines of code and use them as a template for moving the second block, and so on. Once the lines of code are copied, all you will need to do is select the appropriate (copied) instruction, jog the robot to the correct pose for that instruction, and then click on Modify Position. Then repeat this process for the subsequent poses (moves) for the second block.
- The robot will automatically stop if you jam the gripper and a block against a plate or the table. Depending on how hard the EOAT is jammed, you may have some difficulty in un-jamming the block. First try simply jogging the robot away from the jam. There will most likely be an error you will have to acknowledge before the robot can be moved. If you can't move the EOAT away from the jam, try releasing the block from the gripper – that may make it easier to get the block out. Tapping the block or plate may also work.