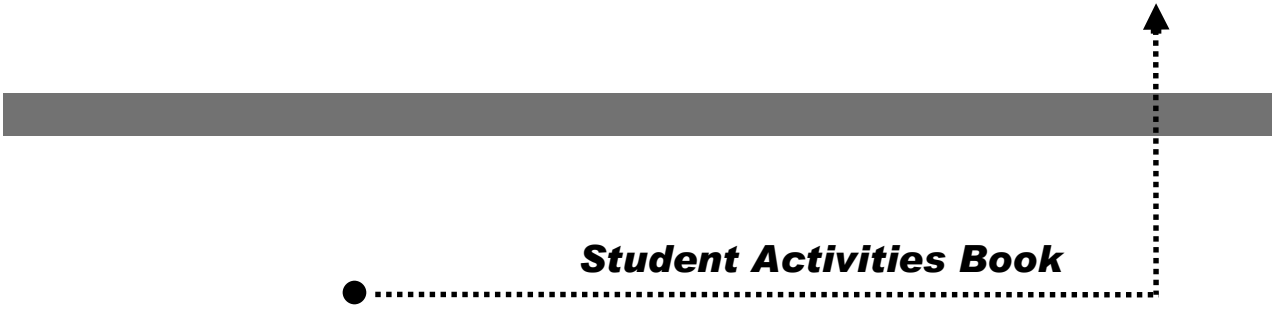
tekLINK

ROBOTICS AND MATERIALS HANDLING 2

RoboCell for SCORBOT-ER 4u



Student Activities Book

Catalog #100356 Rev.01

PRELIMINARY EDITION

intelitek

Catalog No. 100356 Rev. 01
September 2002

Robotics and Materials Handling 2 (SCORBOT-ER 4u) Activities Book

intelitek Inc.
444 East Industrial Park Drive
Manchester, NH 03109-5317
USA
Tel: 603-625-8600
Tel: 800-777-6268
Fax: 603-625-2137

website: www.intelitek.com
email: info@intelitek.com

Table of Contents

About this Activities Book

This Activities Book is a lab manual that contains 15 **Activities**, each of which can be completed in one 45-minute lab session.

At the beginning of each activity you will encounter several lists:

- Objectives are the goals you will achieve.
- SCANS Skills are the competencies you will develop.
- Materials are the specific items you will need for each activity.

The **Overview** introduces you to the subjects you will explore in each activity.

The **Procedures** contain a series of **Tasks**, or operations. The first time an operation is to be performed, instructions are given in a tutorial manner. In subsequent tasks you should be able to perform these operations without guidance.

Many tasks are best performed when each team member takes on a different role. One student may, for example, handle the hardware while another student manages the software. The activities are designed so that team members can switch roles and repeat tasks, thereby allowing everyone more "hands-on" time.

Questions and tables for entering results and observations appear throughout the tasks. Questions for discussion and review conclude each activity. All questions and tables are printed on a set of **Worksheets** supplied with this book. Record your answers in the worksheets, or as directed by your instructor. ***Do not write in this book.***

The **Academics** section at the end of each activity contains enrichment material, such as industrial applications and opportunities, or the scientific background upon which the tekLINK technology is based.

In AMT tekLINKs that include hardware (e.g., panel, robot), you will be directed to perform inventory and safety checks at the beginning of every working session, and to shut down the system properly at the end of each activity.

In AMT tekLINKs that utilize software, it is assumed that you are familiar with the PC and are comfortable working in the Windows/DOS operating environment. However, instructions are explicit enough to allow novices to use the tekLINK's specific software.

Safety

You will work offline in the Robotics and Materials Handling 2 tekLINK so safety should not be an issue. However should your laboratory also include robotics and materials handling hardware, it is imperative that you make sure that the equipment is offline before beginning the tekLINK. Previous users may have left the system online which could cause a potentially dangerous environment for both the human operators and equipment.

Activity 1

Getting Started

Before you begin this activity, **take the Pre-Test** according to your teacher's instructions. Allow 15 minutes for the test.

The purpose of this Pre-Test is to measure your knowledge and skills in the field of Robotics and Materials Handling. *This test will not affect your tekLINK grade!* When you have finished the test, please hand it in to your teacher. Then proceed to this activity.

OBJECTIVES

In this activity you will accomplish the following:

- Measure your knowledge of robotics and materials handling.
- Identify components of RoboCell software.
- Describe the advantages of using simulation software.
- Load and run a robotic program.
- Control viewing options in the simulated robotic cell.

SKILLS

In this activity you will develop the following skills:

- Academic & Employability:
  - Operate lab equipment according to safety regulations.
  - Document inventory and safety procedures for lab set-up and shutdown.
  - Identify historical development of robotics technology.
- Occupational & Technical:
  - Run a sample robotics simulation program.
  - Adjust the views of the robot workcell.

MATERIALS

In this activity you will need the following materials:

- Computer with RoboCell software
- Pre-Test and Pre-Test answer sheet
- Worksheets for Activity 1

The Robotics and Materials Handling 2 tekLINK

In the Robotics and Materials Handling 2 tekLINK, you will learn to program a virtual robot using the SCORBASE programming language, which is part of the RoboCell robotic simulation software package. To After programming, you will observe how a virtual robot performs your program commands.

Simulation software, which is becoming increasingly popular in the world of computer technology, can be used in a number of manners. New automobile drivers, for example, can "experience" tricky road conditions without fear for their safety; jet fighter pilots can attack virtual targets without any risk and at considerably low cost.

In this tekLINK, RoboCell software will enable you to write programs, and then test their execution by using a virtual robot. Such simulation capabilities will help reduce the cost in planning your final industrial robotic system, as well as significantly reduce the risk of accidents or failures.

RoboCell is a software package that integrates SCORBASE robotic software with a 3D Image module. Every project in RoboCell is composed of three files that usually bear the same name with a different extension:

- By default, positions and program files are saved in one combined operation and have the same name with different extensions.
- A file, with the 3DC extension, containing the data regarding the robot cell. After loading a 3D model file, the 3D Image window shows a graphic image of the cell.

When the SCORBASE program is executed, the virtual robot will move within the defined workcell, according to the program positions and commands.

The RoboCell 3D Image Window

The RoboCell 3D Image Window simulates a video camera output screen. The camera is controlled by the user to focus in on a clear, centered view of the cell and robot actions. The user has several options to control the camera, such as:

- Top View - places the camera on top of the cell at the center of the image.

- Redirect Camera - allows the user to define a position that will be in the center of the image.
- Zoom In/Zoom Out - zooms in and out of the image by pressing the right mouse button and moving it forward or backward.

- Rotating the Image - rotates the view of the image by pressing the right mouse button and moving it to the right/left.
- Moving the Camera Up/Down - use the window's scroll bar to adjust the viewing angle of the image.

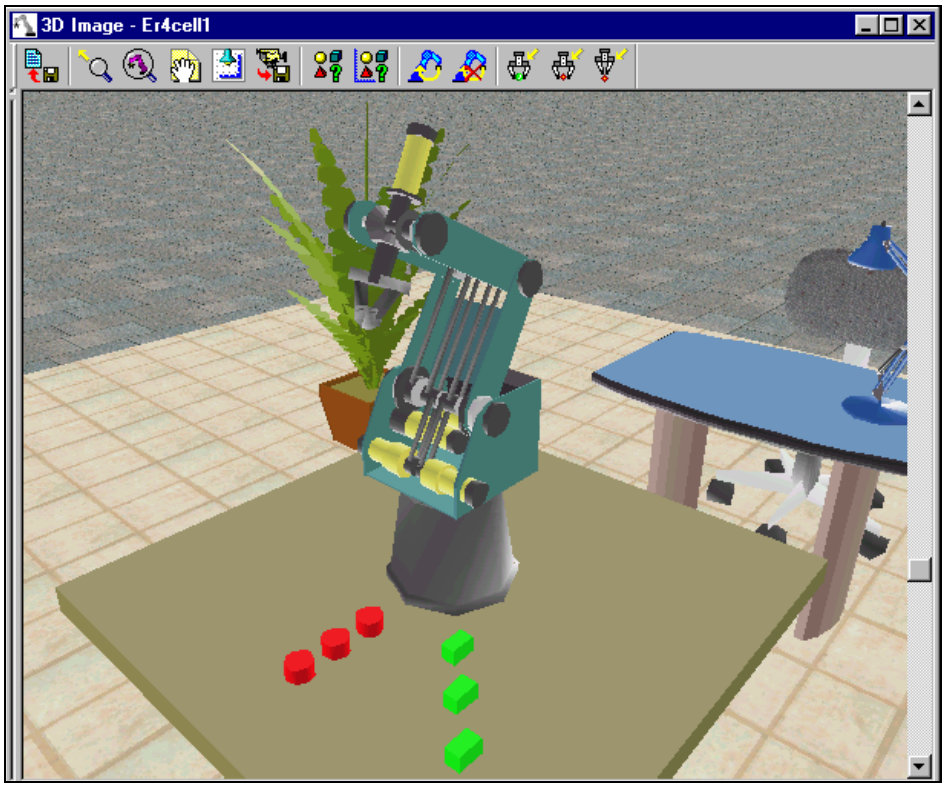


Figure 1-1

SCORBASE Toolbar

Figure 1-2

	Stop	Immediately stops program execution and movement of all axes.
	New	Opens a new, untitled project.
	Open	Opens an existing project.
	Save	Saves the currently active project.
	Run single line	Executes the selected (highlighted) program line.
	Run single circle	Executes the program from the selected (highlighted) program line, to the end of the

		program..
	Run continuously	Executes the program from the selected (highlighted) program line. When the last program line is reached, the program starts again from the first line.
	Search Home	Search Home for all axes
	Control On	Enables servo control of the axes.
	Control Off	Disables servo control of the axes. When control is off, axes cannot be moved.
	Pause	Stops program execution after the current line is executed.
	Charts	Opens the Charts window
	Level 1	Displays list of commands and options at introductory level. Commands related to higher level are disabled.
	Level 2	Displays list of commands and options at advanced level. Commands related to higher level are disabled.
	Pro	Displays list of commands and options at professional level. At this level, all options and commands can be activated.

3D Image Toolbar

Figure 1-3

Reset

Reloads the currently open 3D model file. The robot and all peripheral axes assume their home positions. All objects return to their original positions. The graphic display returns to its default view.

Selecting Reset Model while a robotic program is running will interrupt its execution, but will not stop or reload the program. You must select the SCORBASE Stop command, and then bring the cursor to the first line of the program before resuming program execution.

	Redirect Camera	Allows you to select a different focal point in the graphic display of the cell. To change the center point of the graphic display window, select Redirect Camera and then click on any point in the scene. It now becomes the center point of the graphic display.
	Follow-me Camera	When selected, the camera follows a specific focal point. This function is similar to Redirect Camera, only automatic and continuous. This function is particularly useful for following the motions of the robot gripper. But it can also be used to track any object in the cell, such as a cube that is being moved.
	Drag	Allows you to move the robot screen in a desired direction. Clicking on this icon and "dragging" the mouse to the left will drag the entire scene within the window to the left.
	Top View	Displays an overhead view of the cell.
	Save Camera	Saves the current view of the cell. The graphic display will show this view whenever you select Reset Model or Restore Camera Position, or when the 3D Image file is loaded.
	Names	When selected, a label appears on the object showing its name.
	Positions	When selected, a label appears on the object showing its position. The coordinates indicate the center point of the object relative to the cell's point of origin.
	Path	When selected, a line showing the path of the gripper will be drawn on the screen whenever the robot moves. Clear this menu option to stop the display of the path.
	Clear Path	Removes the robot path drawn on the screen.

Send Robot to Object

Moves the robot (gripper) to an object in the cell. Be sure the gripper is open before using this command.

First select Send Robot to Object, then click on the target object. By default, the gripper will move to a point that is 10 mm above the object's position.

Send Robot to Point

Moves the robot (gripper) to any location in the cell. It is similar to the Send Robot to Object command, but allows you to send the robot to any point on any object in the cell. When you click on an object, such as the table, the target point is the point where you click, not the object's position.

First select Send Robot to Point, then click on the target point. By default, the gripper will move to a point that is 10 mm above the point selected.

Send Robot to Above Point

Moves the robot (gripper) to a point above any selected location in the cell. First select Send Robot Above Point, then click on the target point. By default, the gripper will move to a point that is 150 mm above the target point.

PROCEDURES

Task 1-1: Running RoboCell and Opening the Demo 3D model File

1. Turn on the computer.
2. Run **RoboCell** by doing one of the following:

   - Click **Start | Programs | RoboCell for ER 4u | RoboCell for ER 4u**.
   - Click on the **RoboCell for ER 4u** icon.

   The opening screen appears.

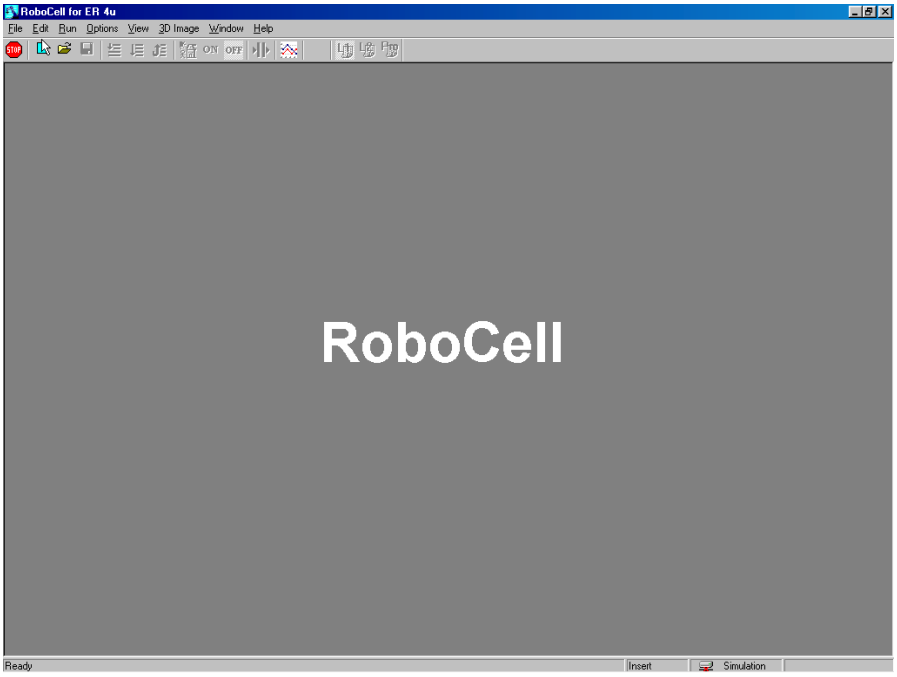


Figure 1-4

In order to create a new robotic program or run an existing program, you must open a project. A RoboCell project contains a program (SBP file), user-defined positions (PNT file), a 3D cell image file (3DC file) and a project data file (WS file). Throughout this tekLINK, the term "project" will refer to the program positions and image files saved by the user as one entity.

3. Load an existing RoboCell project as follows:

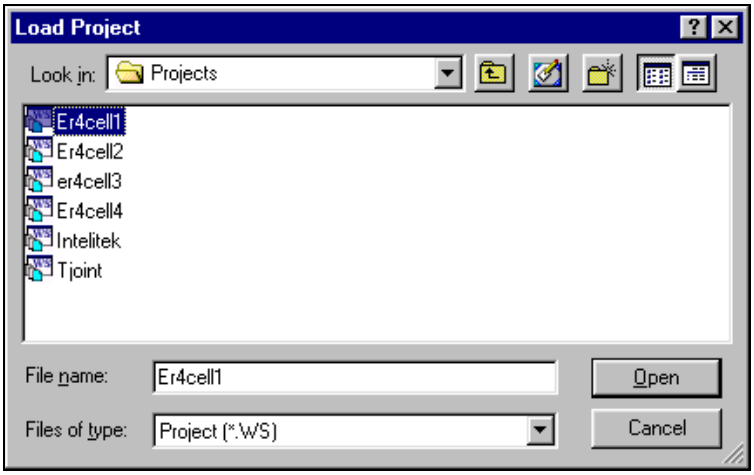


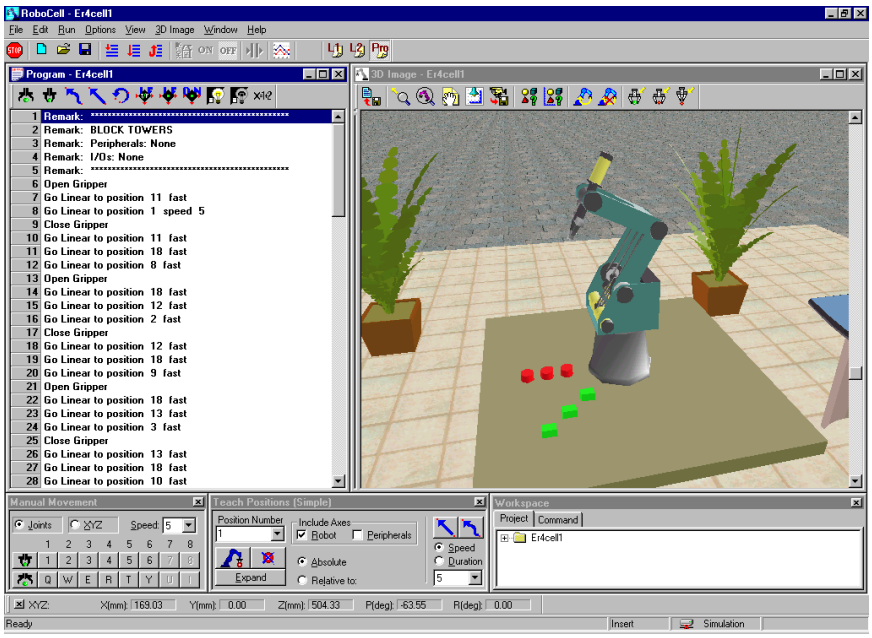
Figure 1-5

- Select **File | Open Project** or click on the **Open Project** icon. 
- Select the file **Er4cell1**.

  This file is a *.ws file, which contains both a robot program, saved positions and 3D Image of a robotic cell.
- Select Open to confirm.

Several windows and dialog boxes appear on the screen. The program appears in the window in the upper left corner of your screen. The 3D cell image appears on the right side of the screen.

In the 3D Image window, a robot stands on a table. In front of the



robot are three red cylinders and three green cubes.

Figure 1-6

Task 1-2: Identifying Components of RoboCell Software

1. From the following figure, identify the following in the RoboCell software screen and label them on the picture shown in your worksheet:
   - RoboCell window
   - RoboCell toolbar
   - RoboCell menu
   - Program window
   - Program toolbar
   - 3D Image window
   - 3D Image toolbar
   - Workspace window
   - Manual Movement dialog box
   - Teach Positions dialog box
   - XYZ View
   - Status bar

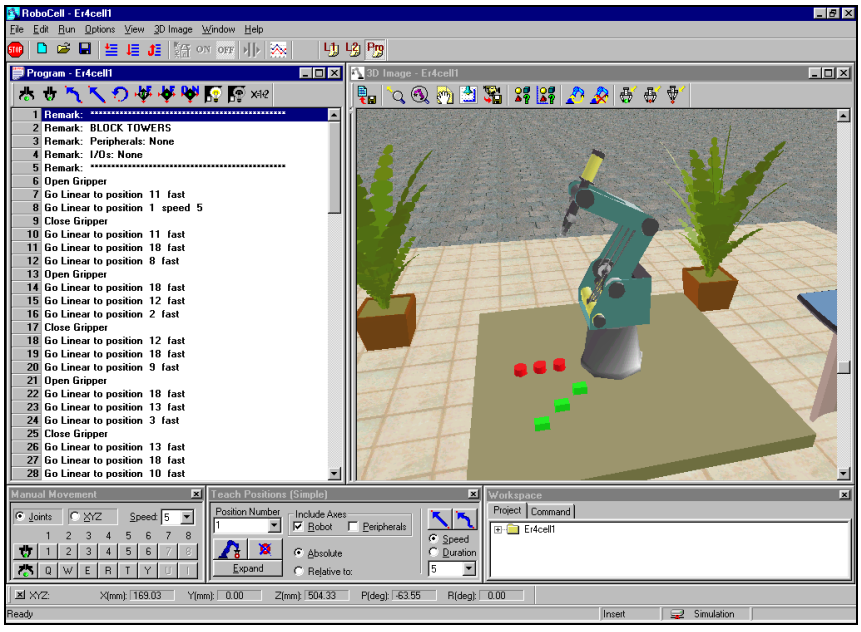


Figure 1-7

Task 1-3: Running a Program

A SCORBASE program is composed of simple commands and positions. The program commands can be executed line-by-line, one cycle at a time or non-stop continuously. In this task, you will order the robot to execute one cycle of the program and stop at the end.

1. From the Program window, click on the first line of the program.
2. From the RoboCell toolbar, click on the **Run a Single Cycle** icon. 

   The robot moves to the nearest cylinder, picks it up and places it on the table between the cylinders and the cubes. Then it moves to the second cylinder picks it up and places it on top of the first. Afterwards it places the third cylinder on the previously stacked two cylinders. It then repeats this process with the green cubes. When all cylinders and cubes are stacked, the robot systematically unstacks them.

   Program execution will take some time. Answer the following questions as you watch the program execution. Should you wish to pause the robot at any time, simply click on the **Pause** icon. Reclick on the Run Single Cycle icon to restart the program. 

Q *Describe what the robot does with the red cylinders.*

Q *Describe what the robot does with the green cubes.*

Q *Describe what the robot then does with the green cubes and red cylinders.*

Q *Describe what the robot does after placing the last cube.*

Q *What will the robot do if **Run a Continuous Cycle** execution mode is selected?* 

Task 1-4: Adjusting the View of the Robot Workcell

Imagine that the image shown in the 3D Image window is the output of a video camera installed in the robot workcell. In this task, you will control the camera position, zoom and output.

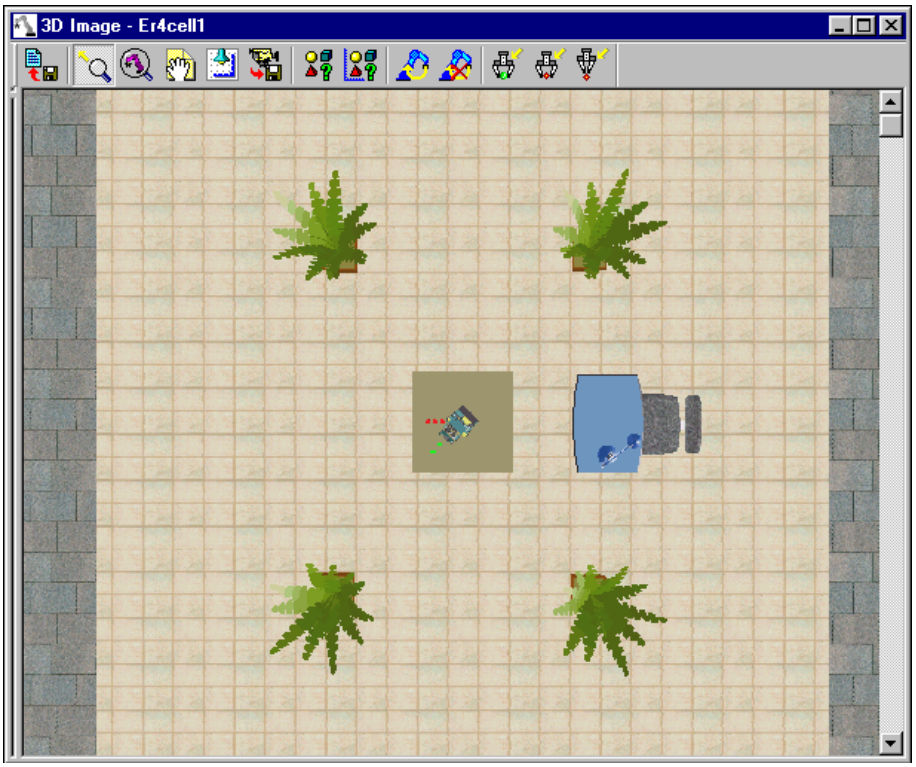
1. Click on the Run a Single Cycle icon in the RoboCell toolbar.

   The program starts again. While the SCORBASE program is being executed (and the robot is moving), you will use some of the camera control options. Controlling the camera will not interfere with the program execution.

   The remaining steps in this task will all be performed from the 3D Image menu.

2. Select **3D Image | Top View** or click the **Top View** icon. 

   This command places the camera in the center of the cell ceiling facing



   downwards.

   Figure 1-8

3. Place the cursor anywhere inside the 3D Image window.

4. Hold down the right mouse button and move the mouse forward. The cursor turns into a magnifying glass.

   Your action is similar to a video camera zoom in feature. The zooming in is performed so that the center of the cell stays in the center of the camera output image.

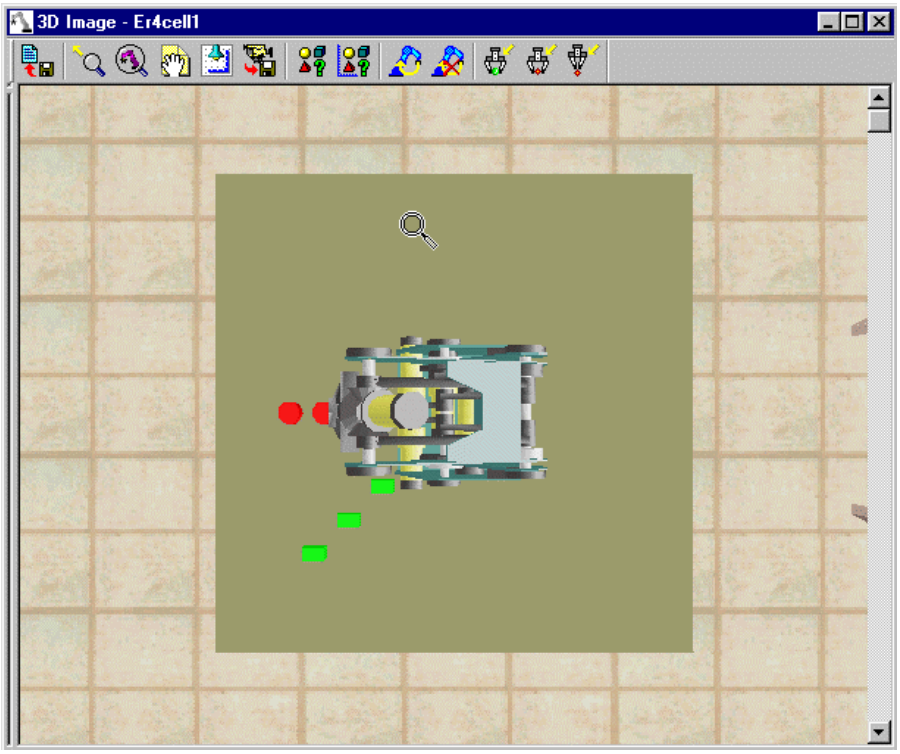


Figure 1-9

5. Hold down the right mouse button and move the mouse backward.

   This action is similar to the zoom out feature.

6. From the 3D Image menu, select **3D Image | Camera | Redirect Camera** or click the **Redirect Camera** icon. 

   The cursor turns into a magnifying glass with an arrow.

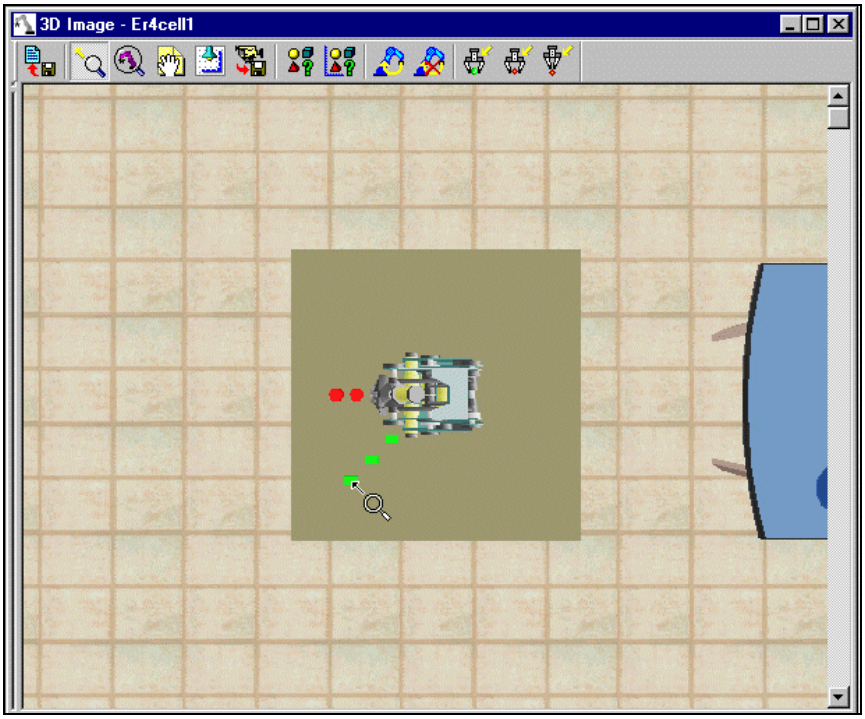


Figure 1-10

7. Click on one of the green cubes.

   The selected box has now been moved to the center of the window. This feature enables you to determine the object (or position) that will be in the center of the image.

8. Deactivate the Redirect Camera feature by re-clicking on the depressed icon.

9. Place the cursor anywhere inside the 3D Image window.

10. Hold down the right mouse button and move the mouse forward.

    Now the zoom in feature is performed so that the selected box stays in the center of the camera output image.

11. Hold down the right mouse button and move the mouse backward.

12. Place the mouse cursor anywhere in the window.

13. Hold down the right mouse button and move to the right and left.

This rotates the displayed image. The center of rotation is the position selected when you executed the Redirect Camera command.

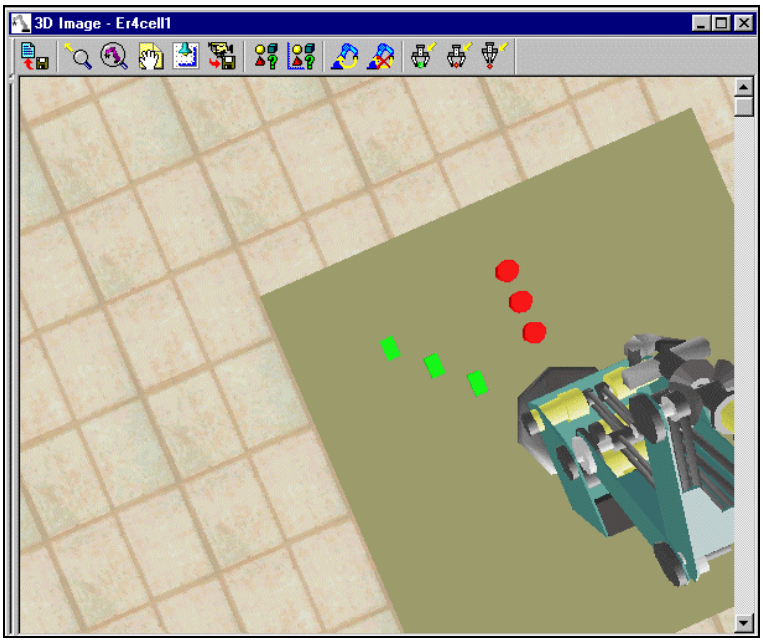


Figure 1-11

14. Move the scroll bar up and down.

This moves the camera up and down, changing the viewing angle.

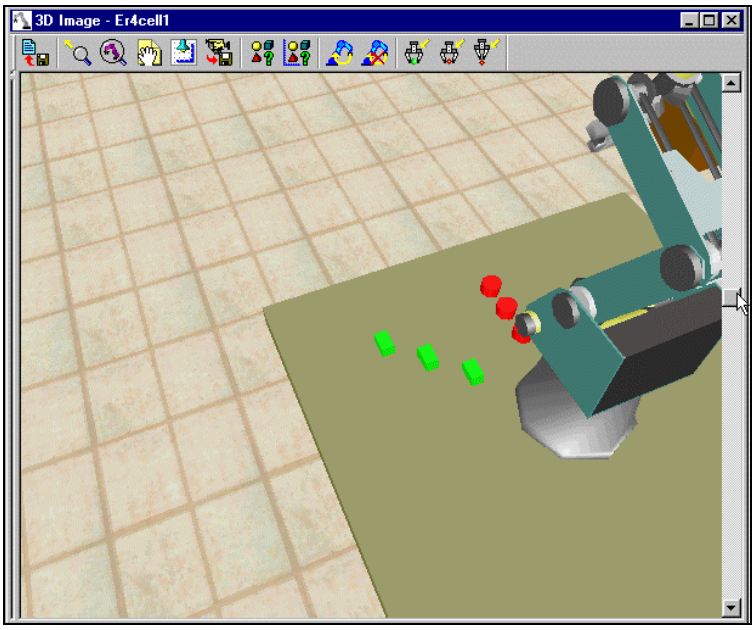


Figure 1-12

Throughout this tekLINK, you will learn to adjust the viewing tools in order to find the most comfortable viewing angle and position for you.

Task 1-5: Team Discussion and Review

The only viewing tool not discussed in this activity is the **Follow-Me Camera** feature. 

Q *What do you think will happen if this feature is selected?*

Hint: After selecting this tool, click on one of the objects in the window (cylinder or cube) and run the program.

Task 1-6: Shut Down

Exit RoboCell software by doing one of the following:

- Select File | Exit.
- Press Alt + F4
- From the Title-bar, click on the application icon and select Close.

ACADEMICS

History

Robots and Automated Machines

The term robot originates from the Czech word robota, meaning "compulsory labor." It was first used in the 1921 play R.U.R. (Rossum's Universal Robots) by the Czech novelist and playwright Karel Capek. The word robot has been used since to refer to a machine that performs work to assist people or work that humans find difficult or undesirable.

The concept of automated machines dates to antiquity with myths of mechanical beings brought to life. Automata, or manlike machines, also appeared in the clockwork figures of medieval churches, and 18th-century watchmakers were famous for their clever mechanical creatures.

The development of the multijointed artificial arm, or manipulator, led to the modern robot. A primitive arm that could be programmed to perform specific tasks was developed by the American inventor George Devol, Jr., in 1954. In 1975 the American mechanical engineer Victor Scheinman, while a graduate student at Stanford University in California, developed a truly flexible multipurpose manipulator known as the Programmable Universal Manipulation Arm (PUMA). PUMA was capable of moving an object and placing it with any orientation in a desired location within its reach. The basic multijointed concept of the PUMA is the template for most contemporary robots.

from "Robot," Microsoft® Encarta® 97 Encyclopedia. © 1993-1996 Microsoft Corporation. All rights reserved.

Activity 2

Recording XYZ Positions

OBJECTIVES

In this activity you will accomplish the following:

- Define the term Tool Center Point (TCP).
- Define the robot Cartesian axes system.
- Record several robot positions.
- Teach several robot positions.
- Program and execute a basic robot program.

SKILLS

In this activity you will develop the following skills:

- Academic & Employability:
  - Operate lab equipment according to safety regulations.
  - Document inventory and safety procedures for lab set-up and shutdown.
  - Identify industrial applications of simulation software.
- Occupational & Technical:
  - Use robotics control software to manually move the robot along the XYZ axes.
  - Write a robotics program to run simulation and describe results.
  - Analyze procedures for safety and quality.
  - Utilize troubleshooting skills to improve the production process.
  - Identify and use components of robotics simulation software.
  - Describe quality issues resulting from improper procedures.

MATERIALS

In this activity you will need the following materials:

- Computer with RoboCell software
- Diskette or personal subdirectory on computer hard drive
- Worksheets for Activity 2

Control Over the Robot's TCP Position

A robot can be defined as a computer controlled machine that is programmed to move, manipulate objects, and accomplish work while interacting with its environment. The great advantage of robots is their ability to perform repetitive tasks quicker, cheaper and more accurately than humans can.

A robot system is made up of the following elements: the manipulator arm, end effector (the gripper or tool mounted on the end of the arm), robot controller and a computer for programming the robot.

Figure 2-1

In RoboCell, the end effector of the virtual robot will always be a gripper (Figure 2-2). In industry, however, real robot systems can perform specific tasks by connecting special tools to the robot's arm in place of the gripper. Building the optimal tool for a robot is probably one of the most difficult and creative tasks of a robotic engineer. Tools enable robots to perform tasks such as welding, painting, screwing and carrying objects.

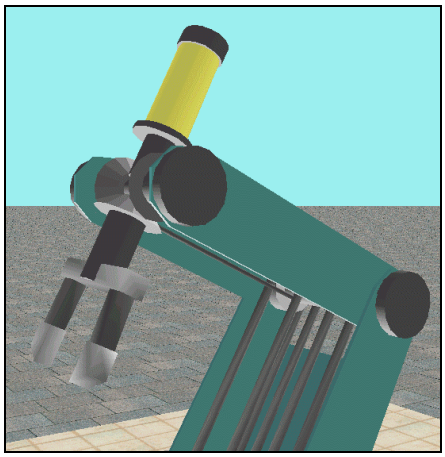


Figure 2-2

Control over the robot is therefore based on controlling the location, position and movements of the tool attached to the robot arm. The control system monitors and controls the location and speed of one point in particular, known as the **Tool Center Point** or, in short, **TCP**.

Cartesian (XYZ) Coordinate System

A robot's TCP position is specified by a Cartesian, or XYZ, coordinate system. In a Cartesian coordinate system, each point has a singular and unique name that is made up of three numbers (also known as the point's **coordinates**). The first number represents the distance of the point along the X-axis, the second the distance of the point along the Y-axis and the third along the Z. The sign of the coordinates (+ or -) indicates its direction along each axes.

Figure 2-3 shows a position in a three axes system.

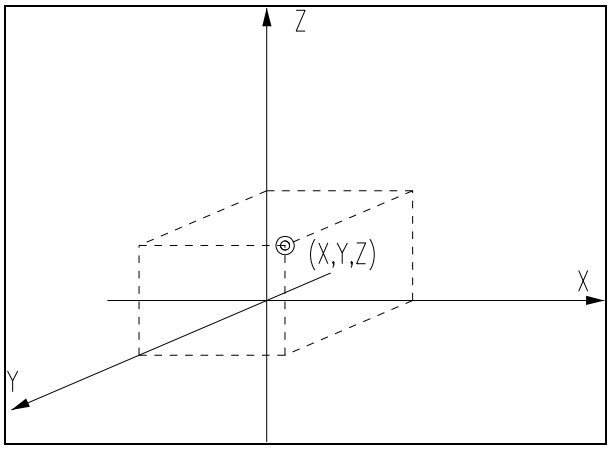


Figure 2-3

Note that in an XYZ coordinate system, the coordinates of the origin (where each of the axes intersect) are (0,0,0). Two different points will always have different coordinates.

In the robot system, the center of the robot base (on the table) is defined as the origin. Therefore, moving the TCP:

- Up/down is considered movement along the Z-axis.
- Right/left is considered movement along the Y-axis.
- Away from/towards the robot base is considered movement along the X-axis.

Record and Teach Commands

Although the terms **Teach** and **Record** are often used interchangeably in robotics, RoboCell software makes the following distinction:

- The **Record Position** command is used to record the current TCP position as a position to be used for programming. Use this command if you do not know the accurate coordinates of a position you want to use in your program. As you will see during the coming activities, moving the tool to an exact desired position can be quite difficult.
- The **Teach Position** command is used to designate a TCP position by entering the coordinates (and angles) of the tool. Use this command if you know the accurate coordinates of a position you want to use in your program.

The Teach Position function is most effectively used for position modification, that is, for changing, only one of the coordinates of a position. You can use it to adjust a position's location. Or you can use the coordinates of an existing position to create a new position whose locations differs only slightly from the first.

In this activity, you will use the Record command to record position #1. Then you will modify the recorded position coordinates and use them to Teach the next three positions.

Teach Positions Dialog Box

In RoboCell, you will record and teach positions using the Teach Positions dialog box. Figure 2-4 shows the Simple version of that dialog box.

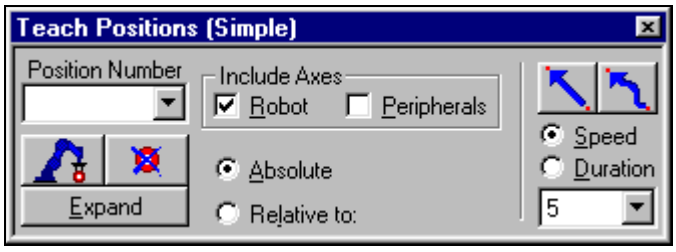


Figure 2-4

Figure 2-5 shows the Expanded version of that dialog box.

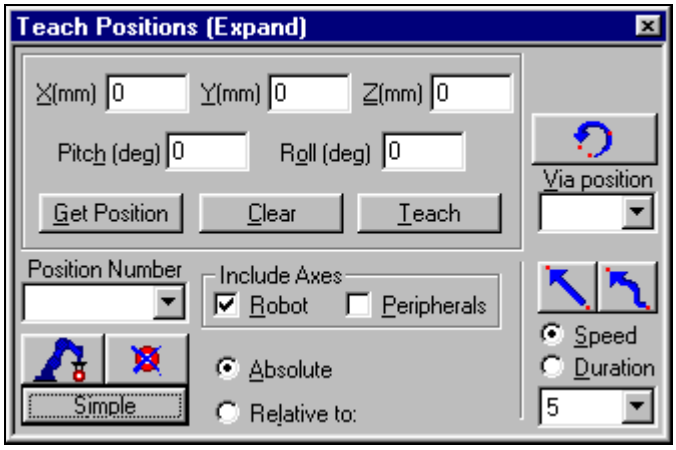
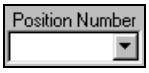
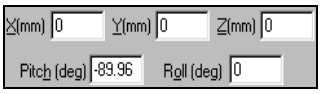


Figure 2-5

In this activity, you will explore the following functions of this dialog box:

Function/Icon/Field	Description
Position Number 	Allows you to assign a numerical name for a position.
Record 	Records the current robot position (in joint coordinates) to the position displayed in the position number field.
Expand 	Opens the Teach Positions (Expand) dialog box.
Go Position 	Executes the *Go Position* command. Sends the axes to a selected position.
X(mm), Y(mm), Z(mm), Pitch (deg.), Roll (deg.) 	Fields for displaying or changing the Cartesian coordinates of the selected position.
Get Position 	Displays the Cartesian coordinates of the selected position.
Teach 	Teaches position using Cartesian coordinates system.
Simple 	Returns to Simple teach position dialog box.

Moving a Cube by Recording Four Positions

In this activity's robotic cell, a cube will initially be located on a table next to a robot. You will program the robot to move the cube backwards 100 millimeters to a new position (shown in Figure 2-6).

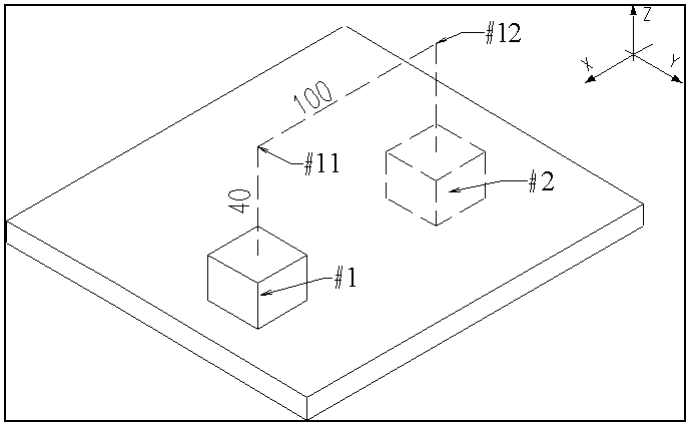


Figure 2-6

As you may notice from the figure, positions numbering is done in such a manner as to help you better remember specific locations. All positions on the table level are given one digit numbers (e.g. positions #1 and #2). Positions directly above the one digit positions are named by adding ten to the number of the position they are above (positions #11 and #12).

In order for the robot to move the cube, you must do the following:

- Find the location of the cube in the Cartesian system and define its location as position #1.
- Define the next three positions (#11, #12, and #2) through which the robot will move the tool (with the cube).
- Write a program that will instruct the robot how to perform the job. The program will do the following:

  Open the gripper

  Move to position #11 fast (the tool is not sent directly to position #1; instead it will approach the cube from a position above it)

  Move to position #1 slow (to prevent damage and increase accuracy)

  Close the gripper

  Move back to position #11 fast

  Move to position #12 fast

  Move to position #2 slow

  Open the gripper

  Move back to position #12 fast

The Remark Command

User comments are commonly inserted into robot programs using the **RE Remark** command. Remarks do not affect program execution and are useful for program execution, maintenance and debugging.

In this tekLINK, every program you write will begin with four remarks, creating a "headline" documenting the contents of the program. The following example shows how all programs should begin:

```
1   Remark: ************************************
2   Remark: ACT#
3   Remark: ************************************
4   Remark: ACTIVITY NAME
```

Remarks are added using the Remark dialog box (Figure 2-7).

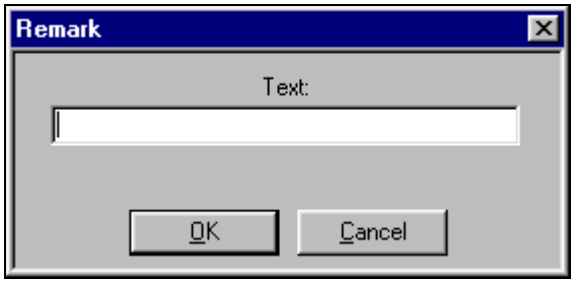


Figure 2-7

PROCEDURES

Task 2-1: Running RoboCell and Importing a 3D Image File

1. Run RoboCell for ER 4u.
2. Every SCORBASE program is part of a project, which can also include the user-defined positions, project data and 3D model files. In order to write your first SCORBASE program, you must open a new project.

   To open a new project, do one of the following:

   - Select **File | New Project**.
   - Click on the **New Project** icon. 
   - Press [Ctrl] + N.

The RoboCell screen now appears as shown Figure 2-8.

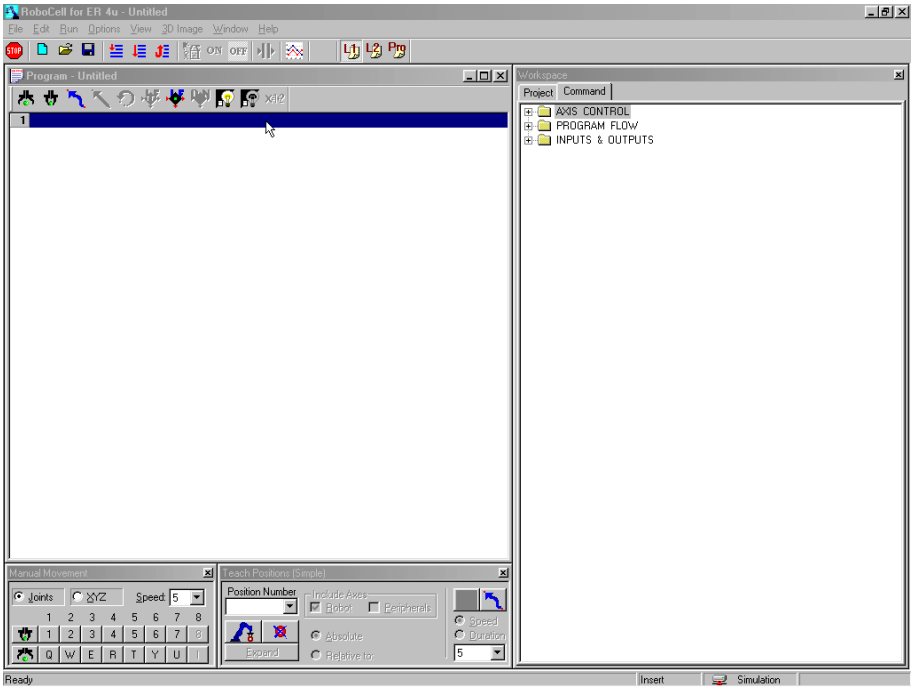


Figure 2-8

3. You now need to open a 3D image model for this project.

Select **File | Import 3D Model**.

The Import 3D Model dialog box opens.

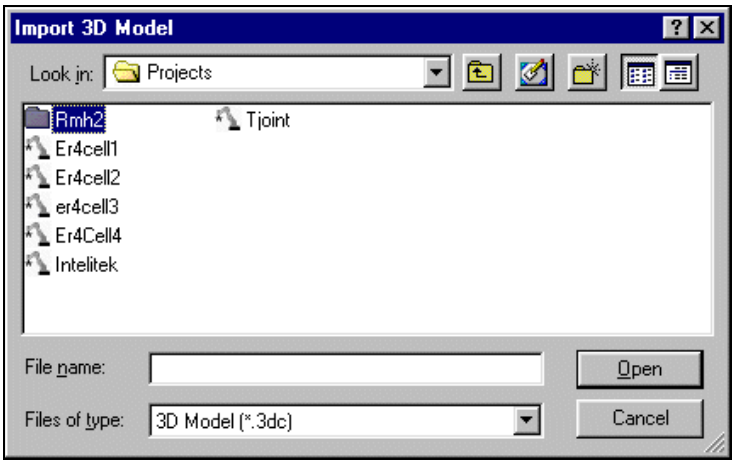


Figure 2-9

- Double click on the RMH2 folder to open that folder.
- Select the 3D model file **ACT02.3DC** and click OK.

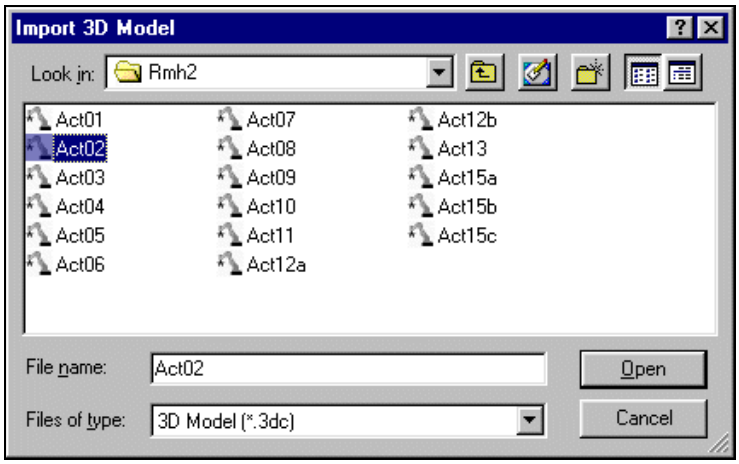


Figure 2-10

A 3D image opens, displaying the SCORBOT-ER 4u robot centered on a table with a red cube placed in front of it.

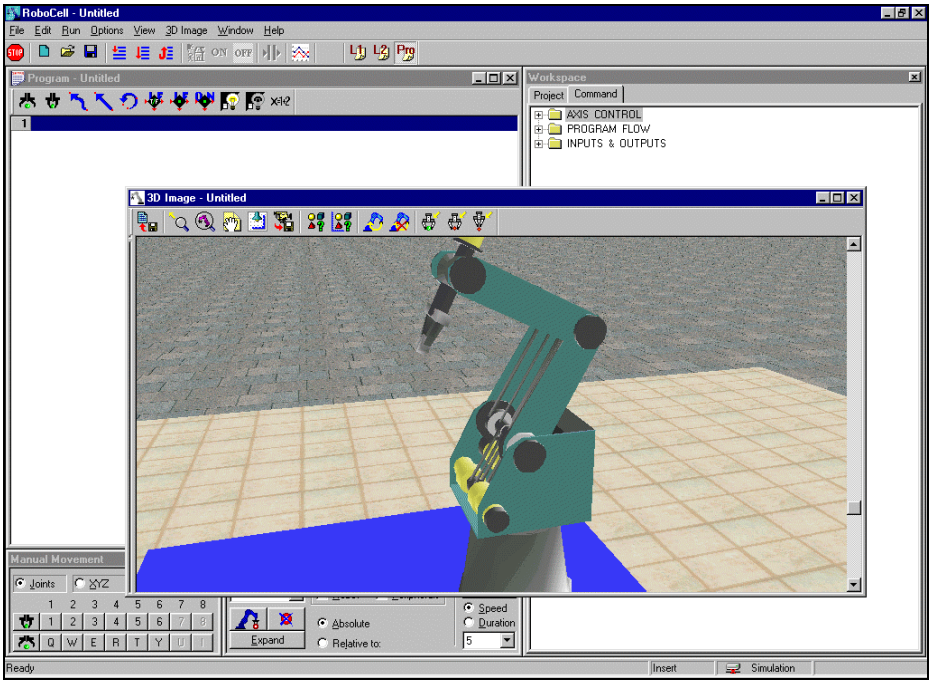


Figure 2-11

4. You now need to arrange the windows in positions optimal for viewing the 3D Image and teaching/recording positions.

5. Select **Window | Simulation & Teach** to arrange the windows.

You screen should now look like Figure 2-12.

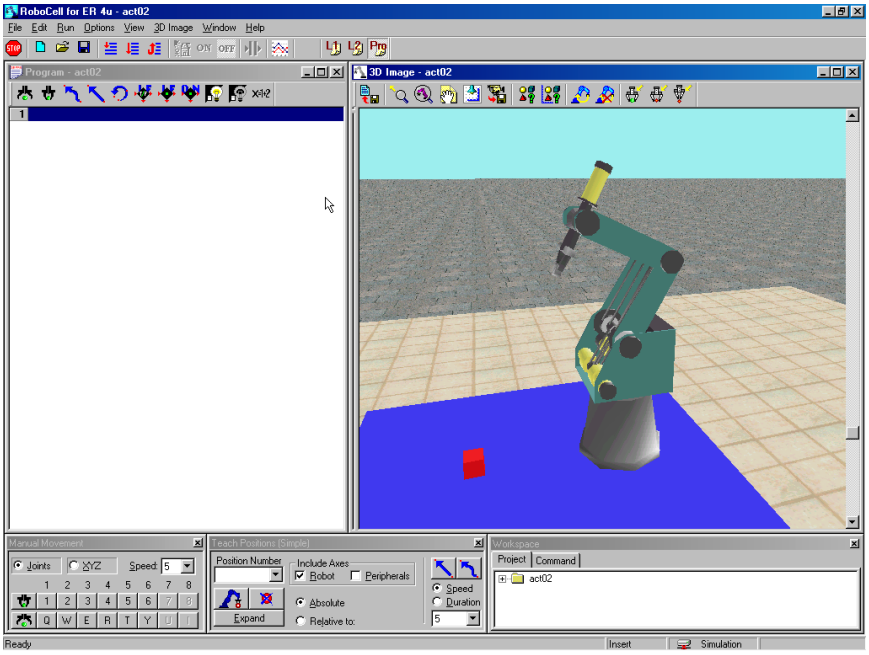


Figure 2-12

6. Select **3D Image | Top View** or click on the **Top View** icon. 

This feature displays an overhead view of the cell.

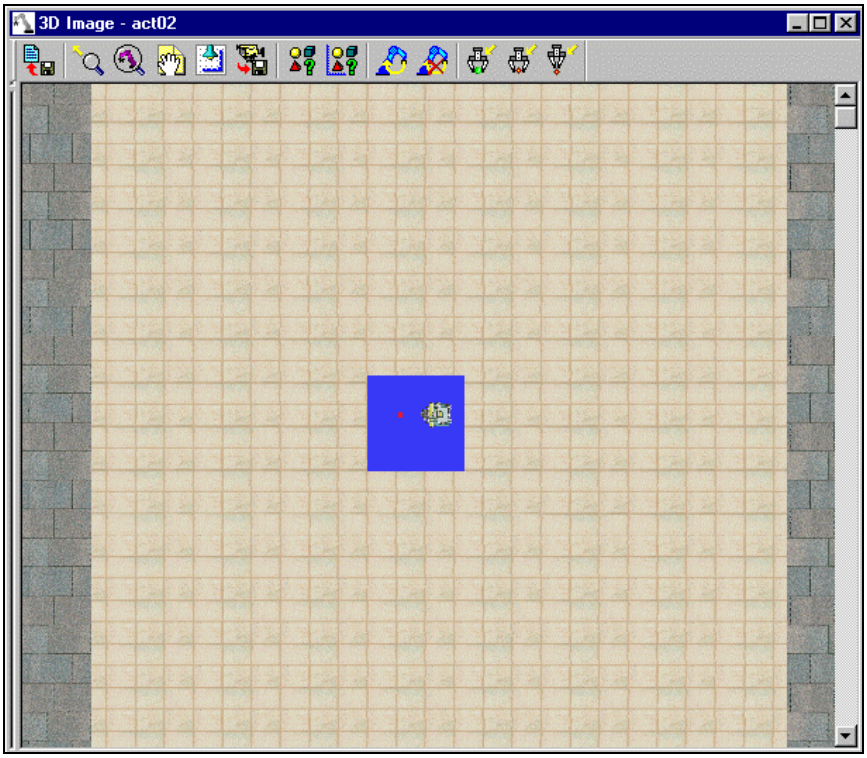


Figure 2-13

7. Select **3D Image | Show Origin** to see the point of origin (0,0) of the cell (at table level). The positions of all objects in the cell are defined as relative to this point of origin.

   If the X and Y labels don't appear unobstructed in the window, zoom in/out or change the viewing angle to seem them (Figure 2-).

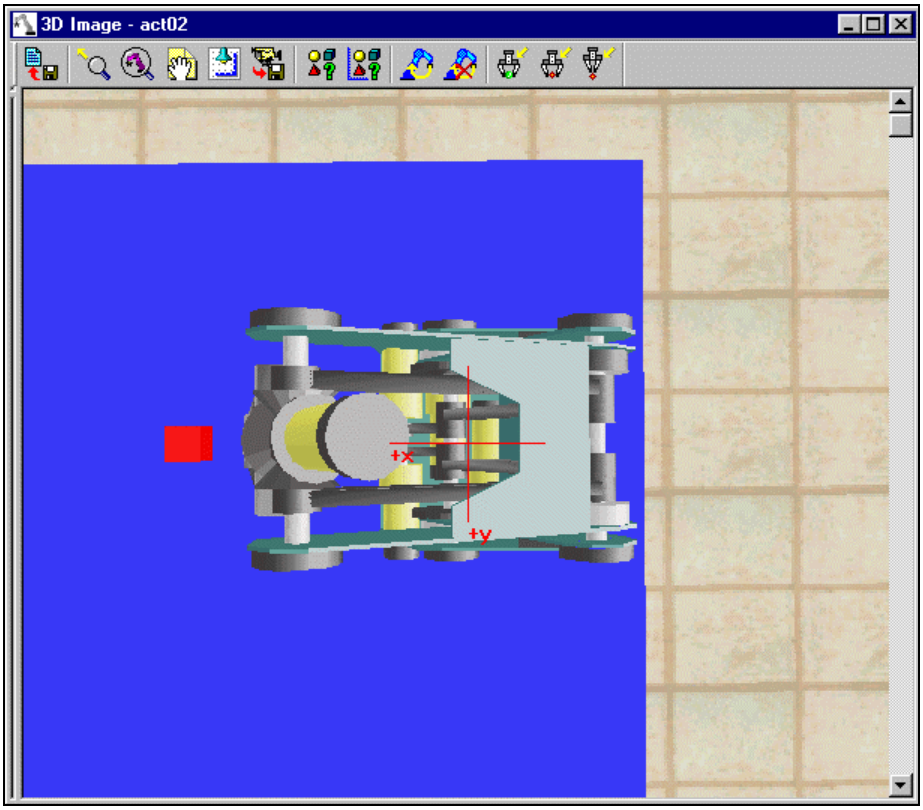


Figure 2-14

8. Select **3D Image | Camera | Redirect Camera** or click on the **Redirect Camera** icon. 

   The cursor turns into a magnifying glass with an arrow in its left-hand corner.

   Click on the red cube to make it the focal point of the window.

   This feature allows you to select a different focal point in the 3D Image window.

9. Deactivate the redirect camera function by doing one of the following:

   - Reselect the menu option.
   - Re-click on the Redirect Camera icon.

10. Use the zoom features and the scrollbar to see a clear view of the cube and the robot gripper.

1. Click in the Manual Movement dialog box (in the lower left-hand corner of the screen) to make it active.

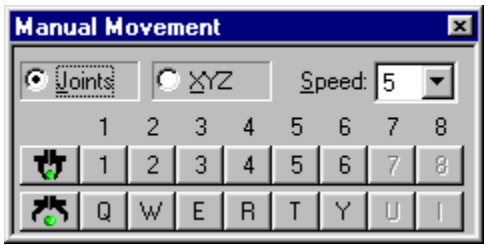


Figure 2-15

The Manual Movement dialog box allows you to assume direct control of the robot and peripheral axes. By clicking with the mouse on the buttons in the dialog box, or pressing keys on the keyboard, you can move the axes.

2. In the Manual Movement dialog box, do the following:

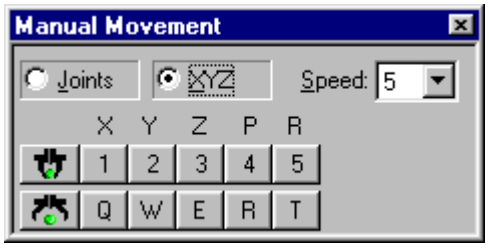


Figure 2-16

- Select **XYZ**. This option, located on the left side of the Manual Movement dialog box, allows you to move the TCP along the X, Y and Z axes.
- Click the **Open Gripper** button to open the robot gripper. 
- Take a moment to familiarize yourself with the Manual Movement dialog box.

  The following chart explains how clicking the buttons (or pressing the keys on the keyboard) moves the robot:

Keys	TCP Motion
1 / Q	TCP moves along the X+ and X- axes.
2 / W	TCP moves along the Y+ and Y- axes.
3 / E	TCP moves along the Z+ and Z- axes.
4 / R	TCP rotates around the Y-axis.
5 / T	TCP rotates around the Z-axis.

In this setup, the cube was placed so that its Y coordinate is zero. Since the initial Y coordinate of the gripper is also zero you only need to manipulate the robot along the X and Z-axes.

3. Using the buttons/keys in the Manual Movement dialog box, you will now move the TCP along the X and Z axes until the cube is between the open jaws of the gripper. Be sure that the gripper tips do not touch the table or the cube.

   - Click **1** to move the tool slightly forward (along the X-axis).
   - Click **E** to lower the tool (along the Z-axis). Make sure that the tool is always slightly above the cube's upper plane.
   - When you are close to the cube click in the speed field from the Manual Movement dialog box and set the speed to 1 (the slowest speed).
   - Move the tool up/down and backward/forward until the cube is located between the gripper's open jaw. Using the viewing tools, rotate the picture to be sure.
   - Click the **Close Gripper** button to close the gripper and make sure that the cube is in the center of its open jaws.

     
   - Then click the Open Gripper button to open the robot gripper.

   The robot is now in position #1 and you will record this position.

4. Click in the Teach Positions (Simple) dialog box, located to the right of the Manual Movement dialog box, to make it an active window and then do the following:

   - In the Position Number field, enter **1**.
   - Then click the **Record Position** button to record the robot's current position as position #1.

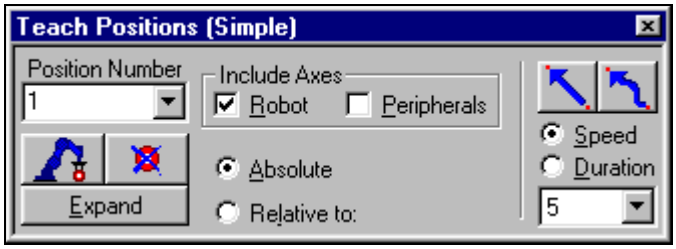


Figure 2-17

Position #1 is now recorded. You will now teach the next three positions required for the required task.

- Click the **Expand** button. 

  The Teach Positions (Simple) dialog box will expand to a version (Expand) containing more options.

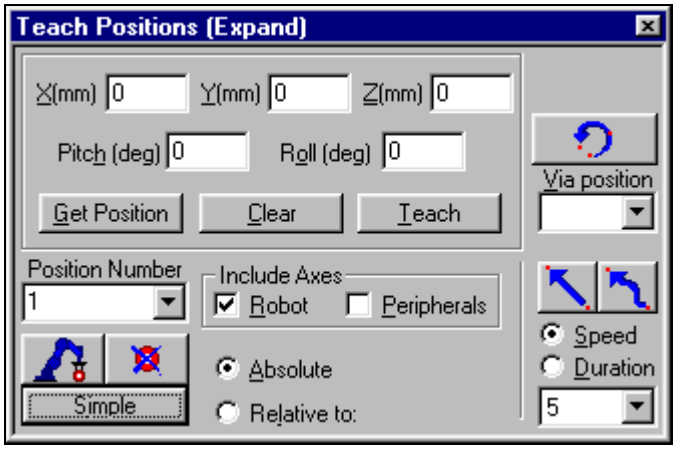


Figure 2-18

- Click the **Get Position** button. 

  The coordinates of the current TCP position (recorded as position #1) appear in corresponding fields in the dialog box.

  Note that your coordinates may slightly vary from the ones shown in the figure.

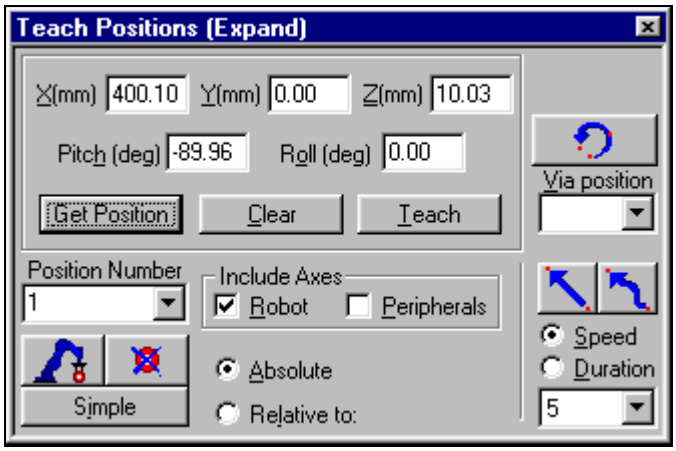


Figure 2-19

5. In your worksheet, record the X, Y and Z coordinates of position #1 in the table.

Position #	X	Y	Z
1			
11			
2			
12			

6. In Table 2-2 in your worksheet, copy the X and Y coordinates of position #1 to position #11.
7. Add 40 to the Z coordinate of position #1 and record it in Table 2-2 as the Z coordinate of position #11.
8. In the Teach Positions (Expand) dialog box, do the following to teach the coordinates for position #11:
   - In the Z (mm) field, enter the new value for Z you recorded in the table and leave all other coordinates intact.
   - In the Position Number field, enter **11**.
   - Click the **Teach** button.

   You have now taught position #11.

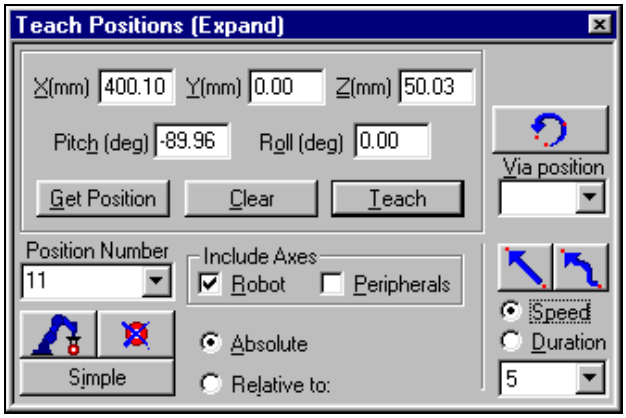


Figure 2-20

9. Click the **Go Position** button. 

   The TCP moves to the new position.

Figure 2-21

10. In the Position Number field re-enter/select 1.

Click Go Position again. The robot moves back to the position previously recorded as position #1.

11. Using the Position Number field and the Go Position button, return the robot to position #11.

12. In the Teach Positions (Expand) dialog box, do the following to teach the coordinates for position #2:

- In the Position Number field, enter/select 1 and click Go Position.
- Click Get Position. The coordinates of position #1 are displayed.
- In Table 2-2 on your worksheet, copy the Y and Z coordinates of position #1 to position #2.
- Subtract 100 from the X coordinate of position #1 and record it in the table as the X coordinate for position #2.
- In the X (mm) field, enter the new X coordinate and leave all other coordinates intact.
- In the Position Number field, enter **2**.
- Click the Teach button to teach this position as position #2.

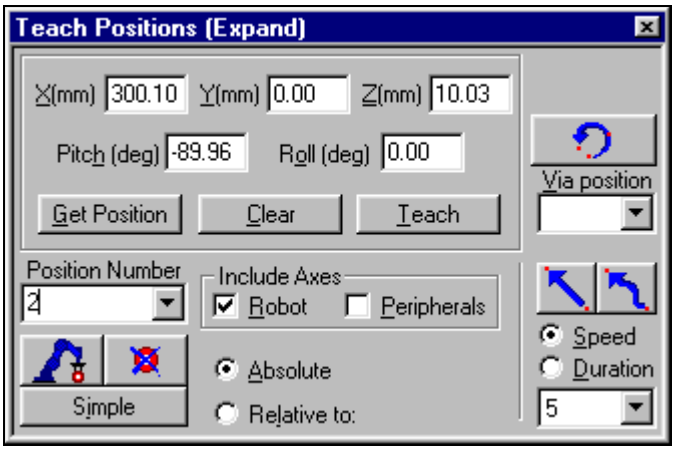


Figure 2-22

- Make sure that the robot gripper is open.
- Click Go Position. The robot moves to the newly recorded position #2.

13. In the Teach Positions (Expand) dialog box, do the following to teach the coordinates for position #12:

- Make sure that the Position Number field is set to 2 and click the Get Position button. The coordinates of position #2 appear in the box.
- In Table 2-2 on your worksheet, copy the X and Y coordinates of position #2 to position #12.
- Add 40 to the Z coordinate of position #2 and record it in the table as the Z coordinate for position #12.
- In the Z (mm) field, enter the new Z coordinate and leave all other coordinates intact.
- In the Position Number field, enter **12**.
- Click the Teach button to teach this position as position #12.
- Click Go Position. The robot moves to the newly recorded position #12.
- You have now recorded all four positions.
- Click Simple to minimize the dialog box.

14. You will now save the positions that you recorded into a project file.

- Select **File | Save Project As**.

  The Save Project dialog box opens.

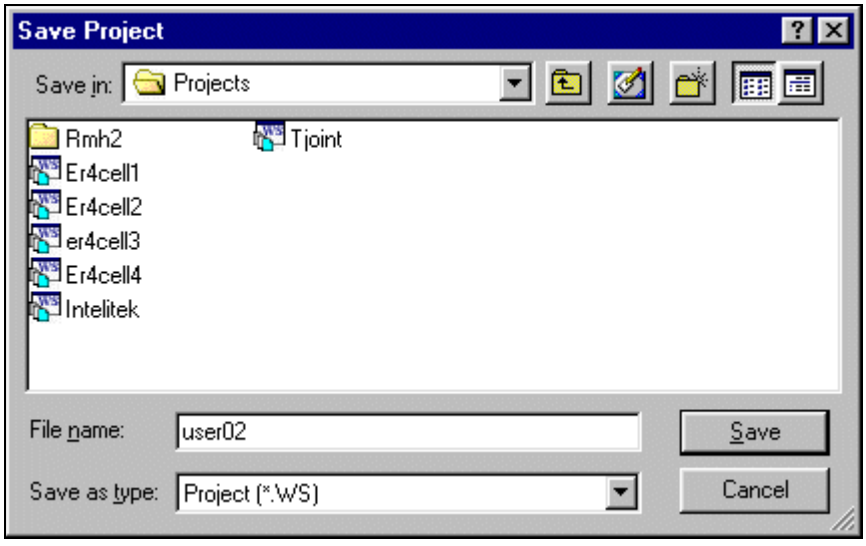


Figure 2-23

- Save the file as **USER2** (where USER is replaced by four characters which identify either the individual student or team; e.g., `JFOX5, BLUE5`).

- If instructed to save the file to your personal diskette, select `a:\` from the drop-down menu in the Save in: (where a is your floppy disk drive).
- If instructed to save the file to a personal subdirectory , be sure to select the directory path from the Save in: (e.g., `c:\blue\blue5`).
- Do not use a file name extension. RoboCell adds the WS extension automatically to create the program and positions files.
- If the file already exists, an error message will appear. Consult with your teacher about whether to overwrite the previous file or save it under a different name.
- Click Save.

The USER2 project file now contains the positions that you just recorded.

15. Click in the Workspace window.

16. The project you just saved now appears in the Project tab window (Figure 2-16).

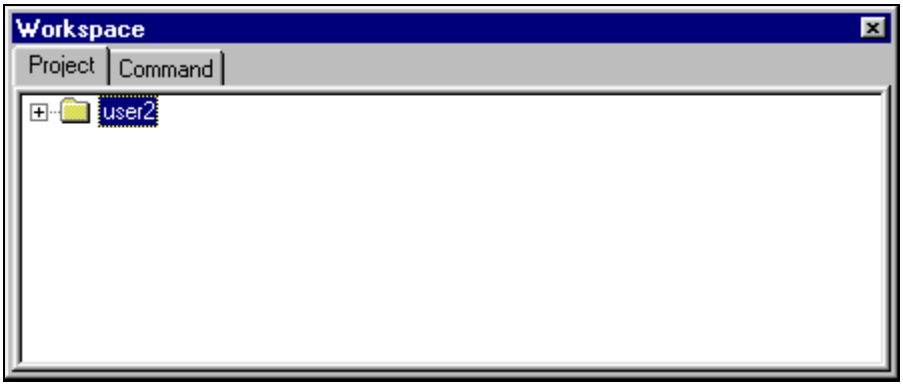


Figure 2-24

17. Open the USER2 tree by double-clicking on USER2 or clicking on the +.

18. Note that the project USER2 is composed of a program file, positions file and a 3D image file.

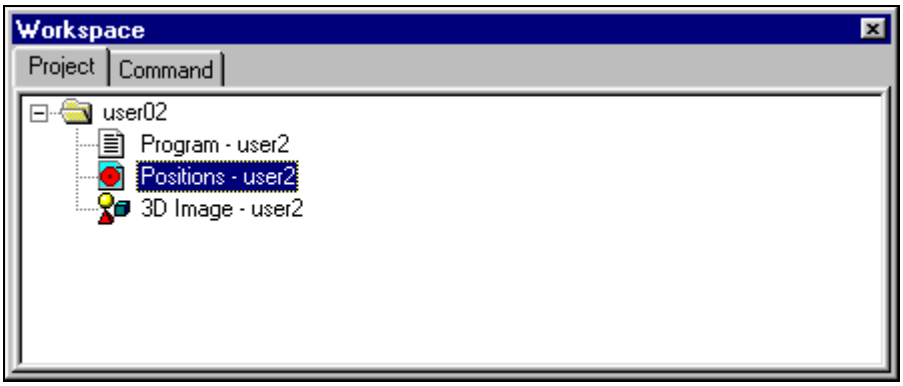
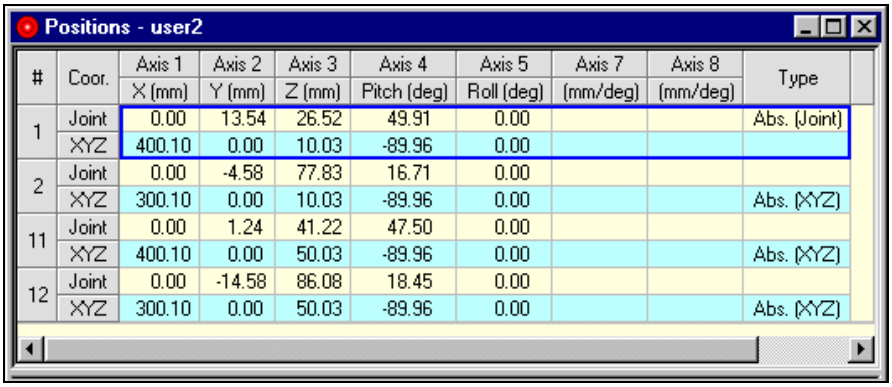


Figure 2-25

19. Double-click on **Positions – user2** to view the positions you just recorded.



#	Coor.	Axis 1 X (mm)	Axis 2 Y (mm)	Axis 3 Z (mm)	Axis 4 Pitch (deg)	Axis 5 Roll (deg)	Axis 7 (mm/deg)	Axis 8 (mm/deg)	Type
1	Joint	0.00	13.54	26.52	49.91	0.00			Abs. (Joint)
	XYZ	400.10	0.00	10.03	-89.96	0.00			
2	Joint	0.00	-4.58	77.83	16.71	0.00			
	XYZ	300.10	0.00	10.03	-89.96	0.00			Abs. (XYZ)
11	Joint	0.00	1.24	41.22	47.50	0.00			
	XYZ	400.10	0.00	50.03	-89.96	0.00			Abs. (XYZ)
12	Joint	0.00	-14.58	86.08	18.45	0.00			
	XYZ	300.10	0.00	50.03	-89.96	0.00			Abs. (XYZ)

Figure 2-26

Q *How do the coordinates shown in the window compare with the coordinates you recorded in Table 2-2?*

Q *What other types of information does the window present?*

Q *Why can't you order the robot to move the cube directly from position #1 to position #2?*

20. Close the Positions window.

21. Return the robot to its initial position by doing one of the following:

- Select **3D Image | Reset Model**.
- Click on the **Reset Model** icon. 

The Reset Model function returns all objects in the robotic cell to their original positions. The robot and peripheral objects assume their home positions. And the 3D Image returns to its default view.

Task 2-3: Programming

In this task, you will program the robot according to the specifications designated in the Overview.

1. Select **Window | Teach & Edit**.

   The RoboCell screen divides into two primary windows, as shown in Figure 2-27. The left window is blank and will contain the untitled program you are about to write. On the right is the Workspace window with the Command tab open. The Command List contains three categories, under which all SCORBASE commands are listed.

   To select a command, open a category and then double-click on the command. It will then be added after the cursor in your current program. If the command requires parameters, a relevant dialog box will open. Some commands have shortcut icons in the Program window toolbar.

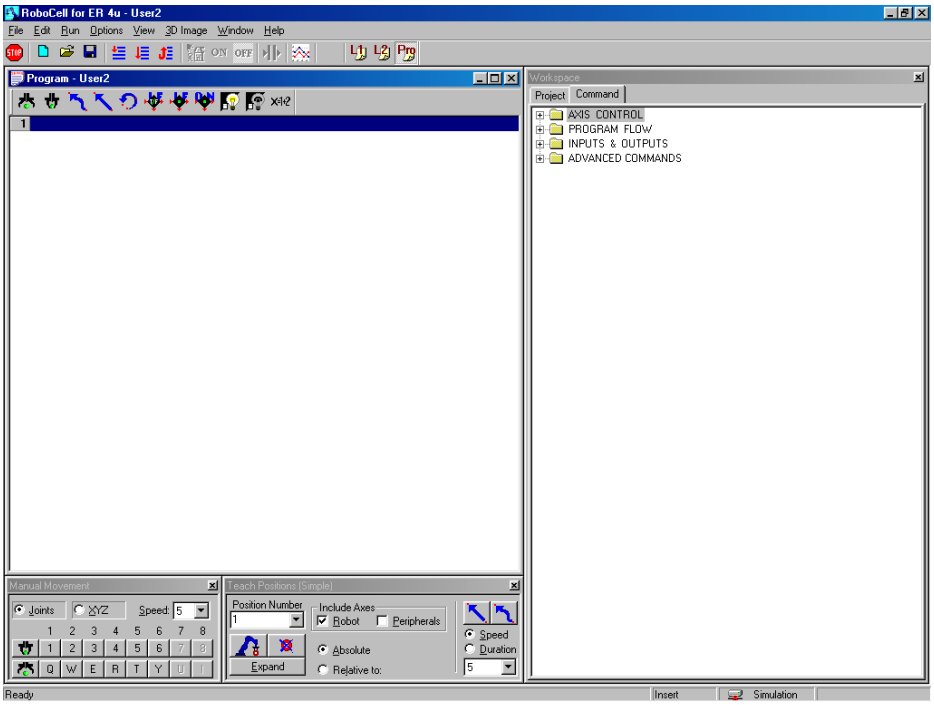


Figure 2-27

2. From the Program Flow section of the Command List, double click on **Remark**.

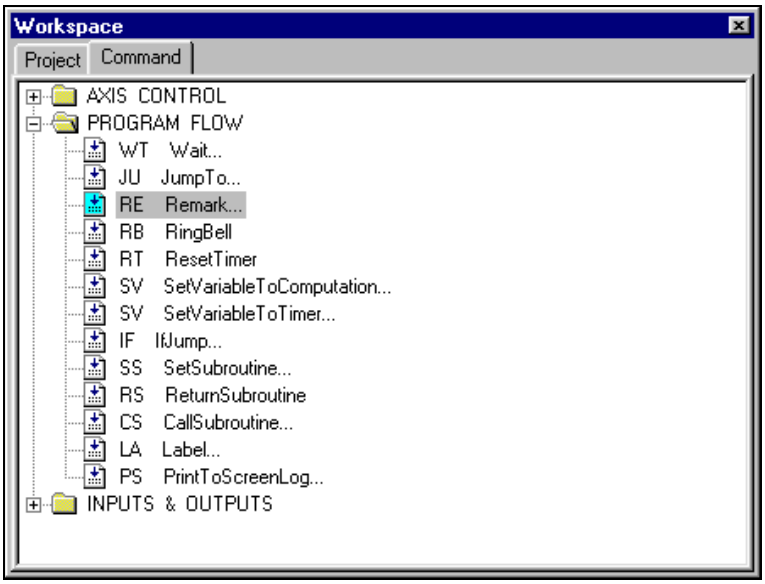


Figure 2-28

The Remark dialog box opens.

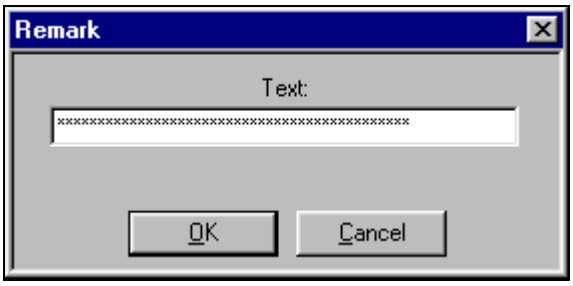


Figure 2-29

3. In the Text field, enter a line of asterisks (**********).

   This "headline" will start every program you write.

   Click OK to accept.

4. Click again on the Remark command.

   In the Text field, enter **Activity 2** to indicate the activity number in which the program was created.

   Click OK to accept.

5. Add a remark indicating the name of the activity**, Recording XYZ Positions**.

   Click OK to accept.

6. Again add a remark with a line of asterisks.

   Click OK to accept.

7. Click on the **Open Gripper** icon in the Program window. 

   The Open Gripper command appears in the Program window.

8. Click on the **Go Position** icon in the Program window.

   The Go to Position dialog box opens.

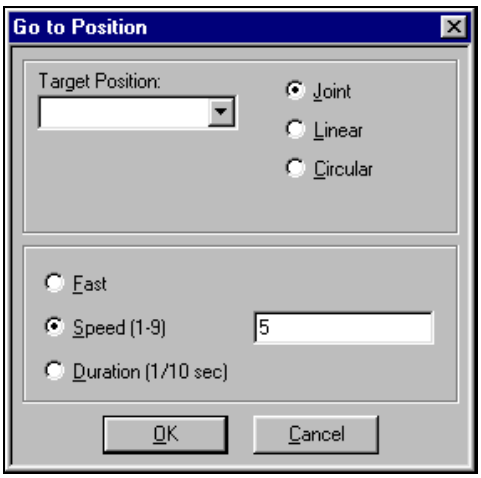


Figure 2-30

- In the Target Position field, enter/select **11** (position #11).
- Select **Fast** for the speed.
- Then click OK to close the dialog box.

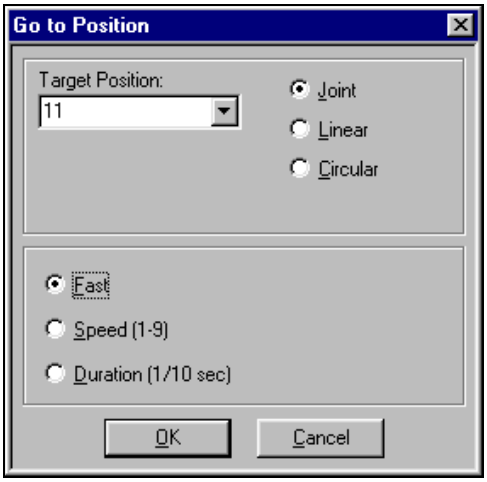


Figure 2-31

9. Add the Go Position command again, setting the target position to 1 and the speed to 1 (the slowest speed).
10. Click on the **Close Gripper** icon in the Program window. 
11. Click on Go Position. Set the Target Position to 11 and the speed to fast.
12. Click on Go Position. Set the Target Position to 12 and the speed to fast.
13. Add a command moving the robot to position #2 in speed 1.

14. Double click on Open Gripper from the Command List.

15. Add a command moving the robot to position #12 fast.

16. Compare the program you just wrote with the following:

```
1     Remark: ***************************
2     Remark: Activity 2
3     Remark: Recording XYZ Positions
4     Remark: ***************************
5     Open Gripper
5     Go to Position 11 fast
7     Go to Position 1 speed 1
8     Close Gripper
9     Go to Position 11 fast
10    Go to Position 12 fast
11    Go to Position 2 speed 1
12    Open Gripper
13    Go to Position 12 fast
```

17. Save the project by doing one of the following:

- Select **File | Save**.
- Click on the **Save** icon. 

Task 2-4: Running the Program

1. Select **Window | Run Screen**.

Your screen divides such that the program appears on the left and the 3D Image appears on the right (Figure 2-32).

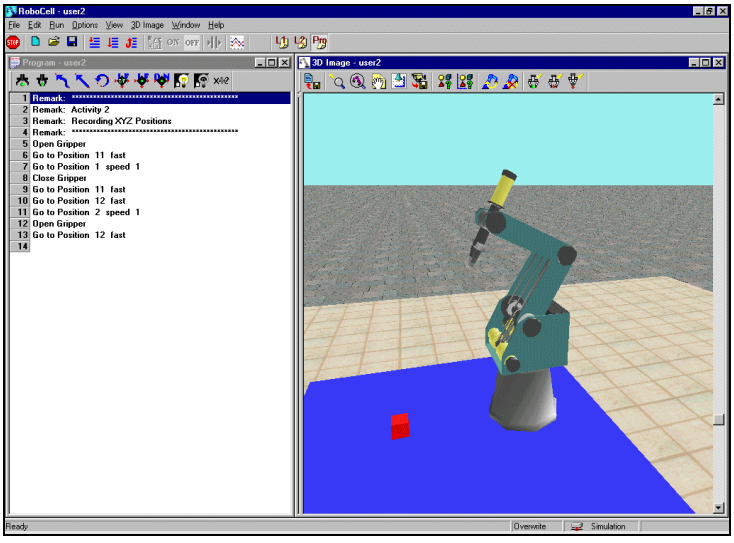


Figure 2-32

2. Click on the first line of the program.

3. Click on the **Run Single Line** icon or press F6. 

As you learned in the previous activity, one line of the program will be executed each time you select this icon or press F6.

4. Continue running the program to the end by clicking repeatedly on the Run Single Line icon.
5. When you have run a full cycle of the program, select 3D Image | Reset Model.

   The cell will return to its initial position.
6. Click on the first line in the program to ensure program execution will start from the first command (Open Gripper).
7. Click the Run Single Cycle icon or press F7.

   A full single cycle of the program will be executed each time you press this icon/key.

Task 2-5: Team Discussion and Review

Q *What would happen if you clicked the Run Single Cycle icon again, without resetting the 3D model?*

Task 2-6: Shut Down

1. Exit RoboCell.

ACADEMICS

Industrial Applications

Simulation Software

Whatever the application in robotics, it is essential to ensure that the investment, training and operation in any factory automation project is both thoroughly researched and accurately implemented.

With the introduction of simulation software, these prerequisites can be achieved with considerable savings to the small and medium sized customer, without sacrificing quality, but achieving economy of scale.

Simulation provides an efficient interactive graphical environment in which to improve the way industrial robots are programmed. Ever-increasing numbers of robot installations are now being planned using computer simulation. Only a few years ago the cost of technology to achieve these aims was prohibitive for all but the largest organizations. However, simulation, calibration and programming of industrial robots is possible even on a standard low-cost PC compatible computer.

Simulation software was primarily intended as a visualization aid for those engineers and managers involved in the process of designing and

debugging new or existing robot installations. Benefits such as the ability to detect off-line collisions between robots and objects, and ability to evaluate and optimize the time taken for a sequence of movements off-line have proved major incentives for the investment in robot simulation.

Simulation has now a major impact on the type of software supplied with robots to users, with most major manufacturers taking a strong interest in marketing their own branded simulations or off-line programming systems. The graphical interface and user friendliness of simulation software has highlighted the inadequacy of today's user-hostile text based robot operating systems. Simulation software is capable of running many different robots from different suppliers in the same workcell. It is now possible to use a graphical simulation system used as a robot operating system with a simple "point and click" selection of robot target positions. New robot tasks can be programmed in a matter of minutes instead of days or weeks.

http://www.rosl.com/taisem.htm

Activity 3

Programming a Continuous Cycle

OBJECTIVES

In this activity you will accomplish the following:

- Write a program that can be executed continuously.
- Adapt a written program for a different task.

SKILLS:

In this activity you will develop the following skills:

- Academic & Employability:
  - Operate lab equipment according to safety regulations.
  - Document inventory and safety procedures for lab set-up and shutdown.
  - Identify industrial applications of virtual reality.
- Occupational & Technical:
  - Write a robotics program that implements a continuous cycle.
  - Write commands needed to return the robot to its original position.
  - Modify the design of the program to improve the design process.
  - Implement modifications to meet changes in design criteria.
  - Monitor and analyze the operation of the system to correct the process.
  - Utilize troubleshooting skills to improve the production process.

MATERIALS

In this activity you will need the following materials:

- Computer with RoboCell software
- Diskette or personal subdirectory on computer hard drive
- Worksheets for Activity 3

OVERVIEW

Continuous Cycles

In this activity, you will load the program and cell you saved in the previous activity and modify it such that the robot will also return the cube back to its initial position (position #1). Because the robot and peripheral equipment will always return to their initial positions, you will be able to run a **continuous cycle** of the modified program. A continuous cycle is a cycle that can be operated continuously without having to reset the cell.

You will then load a different robotic cell in which the cube is placed 70 millimeters away from its previous initial position. You will learn how to modify an existing program and/or its recorded positions to accommodate a different robotic 3D model. You will then run this modified program as a continuous cycle, as well.

PROCEDURES

Task 3-1: Running RoboCell and Opening 3D model File

1. Turn on the computer
2. It is important that while running RoboCell, no other programs are running in the background.
3. Run RoboCell.
4. Load the project USER2, which you saved in the previous activity.

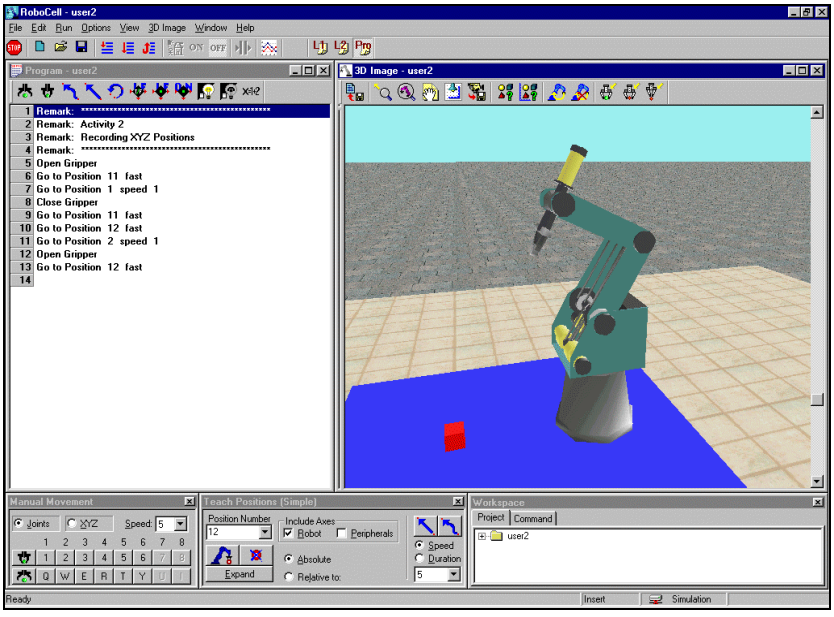


Figure 3-1

5. In this activity, you will modify the program from Activity 2. To ensure that you do not record over your Activity 2 files, resave this project as **USER3a**.

   Select File | Save Project As to save the project as **USER3a**.

6. View the 3D Image from above by doing one of the following:
   - Select 3D Image | Top View.
   - Click on the Top View icon.
7. Redirect the camera to the red cube on the table.

   Don't forget to deactivate the redirect camera function.

8. Use the zoom in features and scroll bars in order to clearly view the cube and robot gripper.

Task 3-2: Running and Modifying the Previous Program

1. From the SCORBASEpro menu, select **Window | Run Screen**.
2. Click the Run Single Cycle icon or press F7.

   Note that the command being executed is highlighted in the Program window.

3. While the robot is moving the cube, fill in the commands executed by the robot in Table 3-1 on your worksheet.

Table 3-1

Line #	Command

Q *Should you record more positions if you want the robot to return the cube to position #1?*

Q *How can you modify the program so that the robot will return the cube to its initial position?*

4. In Table 3-2 of your worksheet, fill in the commands required to move the cube back to position #1.

Table 3-2	
Line #	**Command**

5. Select Window | Teach & Edit to begin programming.

6. Add the commands from the table to your program.

Remember to change the remark commands in the beginning of the program to fit this activity (**Activity 3** and **Programming a Continuous Cycle**).

7. Save the file again by doing one of the following:

- Click **File | Save**.
- Click the **Save** icon. 

Task 3-3: Running the Modified Program

1. Select **Window | Run Screen** to optimize the screen for running a simulated program.

2. Reset the robotic cell (select 3D Image | Reset Model or click on the Reset Model icon).

3. Click on the first line in the program to ensure that the program will run from the beginning.

4. Click the Run Single Cycle icon (F7).

5. Observe the actions of the robot in the 3D Image window.

You have now created a continuous cycle. By programming the robot to return the cube to its initial position, the program can now run

continuously without having to reset the cell between cycles. You will no longer get the impact error messages encountered in the Team Discussion and Review of Activity 2.

6. To run the program in continuous mode, do one of the following:
   - Select **Run | Run continuously**.
   - Click the **Run Continuously** icon 
   - Press F8.
7. After observing the program execution twice, stop the program by doing one of the following:
   - Select **Run | Stop**.
   - Click the **Stop** icon. 
   - Press F9.

Task 3-4: Modifying Positions

RoboCell currently displays a cell and program, in which the robot is commanded to take a cube from its initial position (marked as position #1) and move it back 100 millimeters to a location called position #2.

You will now load a new cell, *ACT03b.3*, in which the cube is initially located 70 millimeters away (on the Y-axis) from its initial position in *ACT02*.

1. Select **File | Import 3D Model**.

   The Import 3D Model dialog box opens.
2. Select the 3D model file **ACT03.3DC** from the RMH2 directory.

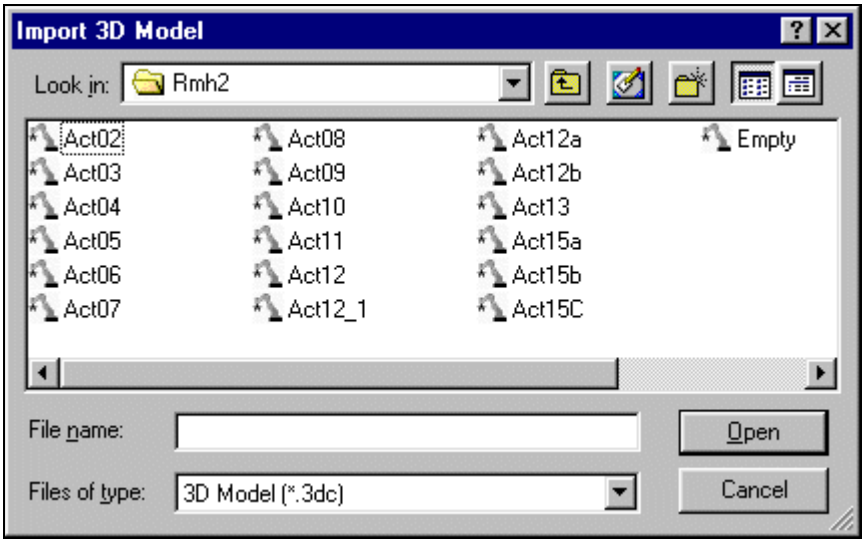


Figure 3-2

As mentioned, the only difference between the cell you loaded just now and the one you previously used (ACT2.3) is that the cube Y-offset is 70 mm.

3. Arrange the windows by selecting **Window | Simulation & Teach**.
4. Save this new project as **USER3b** in the Users directory.
5. Redirect the camera to the red cube.
6. Use the viewing tools to clearly see the cube and the robot gripper from a comfortable angle.

Q *What will happen if you run the current program with this new cell?*

7. Click the Run Single Cycle icon (F7).

Q *Describe the robot's actions.*

Q *Count the changes that should be made in the position coordinates so that the robot will move the cube from the current position back 100 mm.*

Q *Do you need to write an entirely new program?*

You will now modify the four positions so that the robot will move the cube from its new initial position. To modify the previous positions, you need only change the Y coordinate of the four positions.

8. If the Teach Positions dialog box has disappeared, select **Window | Simulation & Teach**.
9. From the Teach Positions dialog box, do the following to modify the positions:
   - In the Position Number field, enter/select the number of the position you wish to change.
   - Make sure the TCP trajectory is clear and the gripper is open.
   - Click Go Position.
   - Click Expand.
   - Click Get Position to display the coordinates of the selected position.
   - Change the Y coordinate of each position from **0**mm to **70**mm.
   - Click the Teach button to teach each new position. The newly recorded positions will replace the previous ones.
   - When done, click Simple to return to the modified version of the Teach Positions dialog box.
10. Save the project with its new positions.

Task 3-5: Saving and Running the Program

1. Reset the robotic cell.
2. Click on the first line in the program to ensure that the program will start running from the beginning.
3. Repeatedly select the Run Single Line icon and observe the actions of the robot.
4. When you have completed running a full cycle of the program, click the Run Continuously icon or press F8 to observe continuous program execution.
5. Press the Stop icon or select **Run | Stop** when you wish to cease program execution.

Task 3-6: Team Discussion and Review

1. Reset the cell.
2. Click on the first line of the program.
3. Zoom in on the cube.
4. Click on the Run Single Line icon.

   When the robot is in position #1, note that the gripper is not aligned with the cube.
5. Click on the Run Single Line icon again.

   Note that the cube turns when the gripper closes.

Q *Why does the cube move when you close the gripper at position #1?*

Task 3-7: Shut Down

1. Exit RoboCell.

Industrial Applications

Virtual Reality

Virtual Reality, (VR), system enables users to move and react in a computer-simulated environment. In some cases various types of devices allow users to sense and manipulate virtual objects much as they would real objects. This natural style of interaction gives participants the feeling of being immersed in the simulated world. Virtual worlds are created by mathematical models and computer programs.

Researchers have been working on virtual-reality devices for many years. In the 1960s Raymond Goertz at Argonne National Laboratory in Argonne, Illinois, and Ivan Sutherland at the Massachusetts Institute of Technology in Cambridge, Massachusetts, demonstrated early versions of HMDs. Goertz, and later Michael Noll of Bell Laboratories, also developed prototype force-feedback devices. In recent years, virtual-reality devices have improved dramatically as the result of various technological advances. Computers now are more powerful, have a higher memory capacity, are smaller, and cost less than in the past. These developments, along with the advent of small liquid-crystal displays (LCDs) that can be used in HMDs, have made it possible for scientists to develop virtual-reality simulations.

Virtual reality is currently used to explore and manipulate experimental data in ways that were not possible before. Therapists use VR to treat sufferers of child abuse and people who are afraid of heights. Muscular dystrophy patients can learn to use a wheelchair through virtual reality

"Virtual Reality,"* Microsoft® Encarta® 97 Encyclopedia. © *1993-1996 Microsoft Corporation. All rights reserved.

Activity 4

Recording Positions by Sending the Robot to Objects

OBJECTIVES

In this activity you will accomplish the following:

- Record positions using simulation software features that send the robot to objects.
- Program the robot to stack three cylinders.

SKILLS

In this activity you will develop the following skills:

- Academic & Employability:
  - Operate lab equipment according to safety regulations.
  - Document inventory and safety procedures for lab set-up and shutdown.
  - Identify applications of robotics technology in the automobile industry.
- Occupational & Technical:
  - Record positions using visual commands.
  - Use advanced software feature to implement simulation.
  - Accurately recording robot positions.
  - Use problem-solving skills to solve a design challenge.
  - Monitor and analyze the operation of the system for quality control.

MATERIALS

In this activity you will need the following materials:

- Computer with RoboCell software
- Diskette or personal subdirectory on computer hard drive
- Worksheets for Activity 4

OVERVIEW

Teaching Positions

In most robotic systems, positions are recording by the following two methods (or a combination of them):

1. Guiding the robot's TCP to the position that should be recorded. When the TCP is in place, the position is recorded using the output of the robot location sensors. You used the Record command to record positions in this manner.
2. Calculating and typing in the coordinates of the positions. This method is based on 3-D geometry and trigonometric calculations. You used the Teach command to record positions in this manner.

In Activity 2, you witnessed the difficulties involved in guiding the robot gripper to a specific position (method #1). This method is frequently used when the TCP is required to move along a path that cannot be defined using basic mathematical tools.

The second method (calculating and typing positions) can be quite problematic. This method's complexity is proportional to the complexity of the tasks the robot must perform. For example, picture how difficult it would be define the path of your hand movements when tying your shoelaces using simple mathematical functions.

Send Robot Commands

In addition to the above two methods, RoboCell software offers a new way to facilitate easy position recording --the Send Robot options:

1. Selecting **Robot | Send Robot to Object**, followed by clicking on an object (including the table), moves the TCP to an object in the cell.

   By default, the open gripper will move to a point 10 millimeters above the object's position so that closing the gripper will grasp the selected object.

2. Selecting **Robot | Send Robot to Point**, followed by clicking on a target point, moves the TCP to any location in the cell. It is similar to the Send Robot to Object command but allows you to send the robot to any point on any object in the cell.

   For example, when you click on any object, such as a cube or table, the target point is the exact point where you click, not the object's position. Be sure to use this option when you want to send the TCP to a specific point on the tabletop.

3. Selecting **Robot | Send Robot Above Point**, followed by clicking on a point, moves the TCP to a point above any selected location in the cell.

   By default, the tool will stop 150 millimeters above the selected object.

Stacking Cylinders Using Send Commands

In this activity, you will program the robot to stack three cylinders using the Send options. As shown in Figure 4-1, each cylinder's height is 35 mm, diameter 35 mm and the distance between each cylinder is 75 mm.

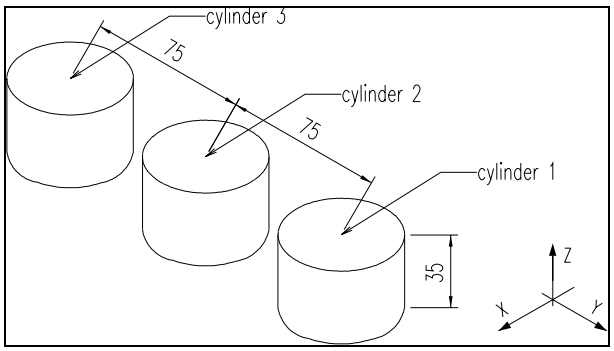


Figure 4-1

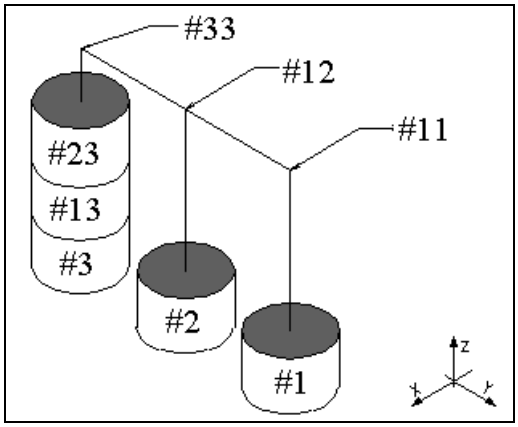


Figure 4-2

Due to safety reasons, the cylinders must be lifted 100 millimeters before being moved along the Y axis. The stacking procedure will be divided into two stages:

1. Cylinder #1 will be placed on cylinder #3.
2. Cylinder #2 will be placed on cylinder #1 (which is already placed on cylinder #3).

PROCEDURES

Task 4-1: Running RoboCell and Opening the 3D model File

1. Turn on the computer and run RoboCell.
2. Create a new program and import the 3D model file **ACT04.3.DC**.
3. From the Graphic Display menu, do the following:
   - Select **View | Redirect Camera**.
   - Click on the center cylinder (cylinder #2).

     Use the viewing features (zoom in/out and scroll bars) to clearly see the cube and the robot gripper.
   - Select **3D Image | Labels | Object Names** or click on the **Names** icon on the 3D Image window. The names of the three cylinders and the robot will be displayed. If the names overlap, zoom out to see the names clearly.

     
   - Select **3D Image | Labels | Object Positions** or click on the **Positions** icon on the 3D Image window.

     The label that previously showed the objects' names will be replaced with the objects' coordinates. Note that because all of the objects are located on the table, the Z coordinate is 0 and only the X and Y coordinates are shown.
   - Select **Labels | Object Names**, or click on the **Positions** icon, to remove the captions.

Task 4-2: Recording Positions

You will now record the eight positions required to define the robot movements, as shown in Figure 4-2.

1. In the Manual Movement dialog box, click the Open Gripper button.
2. Record position #1 by doing the following:
   - From the Graphic Display menu, select **Robot | Send Robot to Object**.

     
   - Then click on cylinder #1 to send the robot to that object.

     The robot gripper automatically moves to a position close to the cylinder. In the new position, closing the gripper would grip cylinder #1 in the gripper jaw.
   - *Do not close the gripper!*
   - In the Teach Positions dialog box, enter 1 in the Position Number field and click Record.

You recorded position #1.

3. Record position #11 by doing the following:
   - From the Graphic Display menu, select **Robot | Send Robot Above Point** and then click on cylinder #1. 
   - In the Teach Positions dialog box, enter **11** in the Position Number field and then click Record.
4. Record positions #2 and #12 the same way you recorded positions #1 and #11.
5. To record position #3, do the following:
   - From the Graphic Display menu, select **Send | Send Robot to Object** and then click on cylinder #3.
   - Using the Teach Positions dialog box, record this position as position #3.
6. To record position #13, do the following:
   - Send the robot to cylinder #1.
   - Close the gripper.
   - Redirect the camera to cylinder #3.
   - Change the viewing angle in the Graphic Display window such that the window is filled with the centered, top surface of the cylinder. (*Use the Top View option*.)
   - You may need to continually redirect the camera to cylinder #3 and adjust the viewing tools.
   - Select **Robot | Send Robot to Point**. 
   - Click on the center of the red circle (the top surface of cylinder #3). Try to be as exact as possible.

     This will define the exact center point of cylinder #3 on which you will stack cylinder #1.
   - Record this position as position #13.

Q *Observe the coordinates for position #13 (Get Position) and explain how the coordinates were calculated based on the cylinder's dimensions.*

7. Open the gripper.
8. To record position #23, first send the robot to cylinder #2 and close the gripper. Then using the Send Robot to Point command, send the robot to the center point of the top surface of cylinder #1 (stacked on top of cylinder #3) and record this position as position #23.

9. Adjust the Z-coordinate of position #23 by adding 100 mm and teach the positions as position #33.

Q *Observe the coordinates for position #23 and explain how the coordinates were calculated based on the cylinder's dimensions.*

Task 4-3: Programming

1. From the SCORBASEpro menu, select **Window | Teach and Edit**.
2. Program the robot to stack the cylinders, using the positions recorded in the previous task.

   Don't forget to document the program using Remark commands.
3. Compare the program you just wrote with the following:

```
 1    Remark: **********************************
 2    Remark: ACT4
 3    Remark: Recording Positions with Send Robot Options
 4    Remark: **********************************
 5    Open Gripper
 6    Go to Position 11 fast
 7    Go to Position 1 speed 5
 8    Close Gripper
 9    Go to Position 11 fast
10    Go to Position 33 fast
11    Go to Position 13 speed 5
12    Open Gripper
13    Go to Position 33 fast
14    Go to Position 12 fast
15    Go to Position 2 speed 5
16    Close Gripper
17    Go to Position 12 fast
18    Go to Position 33 fast
19    Go to Position 23 speed 5
20    Open Gripper
21    Go to Position 33 fast
```

Task 4-4: Running the Program

1. From the SCORBASEpro menu, do the following:
   - Select **File | Save Project As**.
   - Save the file as **USER4**.
   - Select **View | Simulation & Teach**.
2. From the Graphic Display menu, reset the robotic cell.
3. Use the viewing tools to create a comfortable viewing angle of the robotic cell.

4. Click on the first line of the program.
5. Click the Run Single Cycle icon or press F7.

Task 4-5: Team Discussion and Review

Q *Assume that the three cylinders in the cell you just worked with are replaced with three blocks. Will the robot stack the blocks one on top of the other such that the cube bases will overlap one another? If no, suggest a solution.*

Task 4-6: Shut Down

1. Exit RoboCell.
2. Then turn off the computer.

ACADEMICS

Industrial Applications

Robots in the Automobile Industry

Robots weld parts of an automobile together on an automated production line in Fenton, Missouri. Robots are faster and less prone to errors than human workers; as computer and robot technology has become more advanced, robots are increasingly able to perform more complicated tasks.

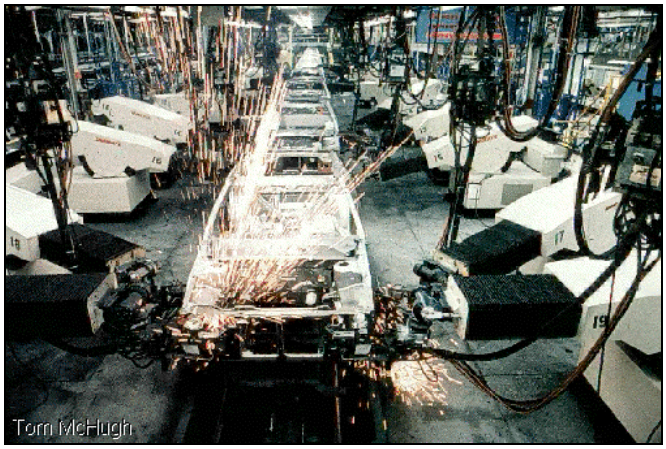


Figure 4-3

Tom McHugh/Photo Researchers, Inc. from "Automation,"* Microsoft® Encarta® 97 Encyclopedia. *© 1993-1996 Microsoft Corporation. All rights reserved.

Activity 5

Defining Roll and Pitch Axes

OBJECTIVES

In this activity you will accomplish the following:

- Define and calculate the TCP roll and pitch angle.
- Program the robot to stack three blocks on top of one another.

SKILLS

In this activity you will develop the following skills:

- Academic & Employability:
  - Operate lab equipment according to safety regulations.
  - Document inventory and safety procedures for lab set-up and shutdown.
  - Identify industrial application of robots.
- Occupational & Technical:
  - Using trigonometry, calculate a corrective angle of rotation.
  - Monitor the operation of the system for quality control.
  - Use troubleshooting skills to improve the production process.
  - Record visually determined coordinates for the robot.
  - Modify the design of the program to improve the design process.
  - Implement modifications to meet changes in design criteria.
  - Monitor and analyze the operation of the system to correct the process.

MATERIALS

In this activity you will need the following materials:

- Computer with RoboCell software
- Diskette or personal subdirectory on computer hard drive
- Worksheets for Activity 5

Degrees of Freedom

One of the ways to characterize robot maneuverability is by counting the robot's **Degrees of Freedom** (**DOF**). A degree of freedom can be defined as the robot's ability to move its tool along or around an axis. Assuming that a robot can move a tool independently along an axis, then it can move a TCP along two or three axes at the same time, enabling higher maneuverability.

An object which is free to move (or rotate) in any direction has six DOF: (see the arrows in Figure 5-1):

1. Along the X axis (in or out).
2. Along the Y axis (left or right).
3. Along the Z axis (up or down).
4. Rotate about the X axis (**Yaw**).
5. Rotate about the Y axis (**Pitch**).
6. Rotate about the Z axis (**Roll**).

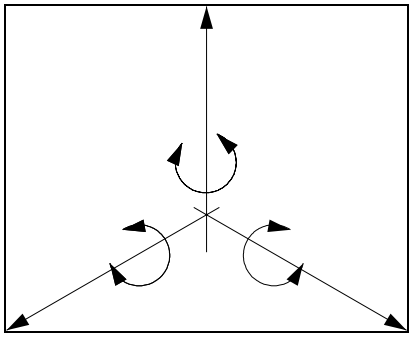


Figure 5-1

So far you have observed that the virtual robot can move the end effector along each of the axes and rotate about the Y and Z axes. Therefore the SCORBOT-ER 4u robot has five out of six possible degrees of freedom.

A full description of the TCP location and position in a RoboCell robotic cell is therefore composed of three coordinates and two angles.

- The **coordinates**, defined in millimeters, describe the distance of the TCP position from the cell origin (as you have already seen).
- The **angles**, defined in degrees, describe the extent of gripper rotation about the Z axis (**roll**) and about the Y axis (**pitch**).

PROCEDURES

Task 5-1: Running RoboCell and Opening the 3D model File

1. Turn on the computer and run RoboCell.
2. Open the program and positions file **USER4** that was saved in the previous activity
3. Import the 3D model file **ACT05.3DC**.

   The cell you loaded is similar to the cell used in Activity 4. In this cell, the cylinders are replaced by cubes with sides of 35 mm.
4. Use the viewing tools to observe the cell from a comfortable viewing angle.
5. Redirect the camera to the center cube (cube #2) and then zoom in.
6. Select Labels | Object Names.
7. Select View | Follow Me Camera.

   Click on cube #1.

   The Follow Me Camera option orders the "camera" to automatically and continuously follow a focal point in the cell, in this case cube #1. The camera will now follow cube #1 as it is lifted/lowered.
8. Select Labels | Hide All Labels.
9. Run a single cycle of the program..

   Pay close attention to the following:

   You may wish to adjust the viewing angle and zoom as the cubes are being stacked.

   - The relative position in which cube #1 is placed on top of cube #3.
   - The angle between the gripper jaw and cube #2 before the gripper closes.
   - The relative position in which cube #2 is placed on top of cube #1.

Q *Describe the relative position in which cube #1 is placed on top of cube #3.*

Q *Describe the angle between the gripper jaw and cube #2 before the gripper closed.*

Q *Describe the relative position in which cube #2 is placed on top of cube #1.*

Q *What is the reason for the misalignment of the cubes?*

Q *Count the positions whose angles must be changed.*

10. As you can see, the position must be adjusted in order for the cubes to be stacked correctly.

This adjustment can be done using one of the following methods:

- Calculating the relative angle and typing it instead of the previous value. You will use this method to modify positions #13 and #23 in Task 5-2.
- Moving the robot TCP to the position whose angle should be adjusted and then adjust the angle manually. Upon completion, the new position is recorded. This method will be used to modify position #3 in Task 5-3.

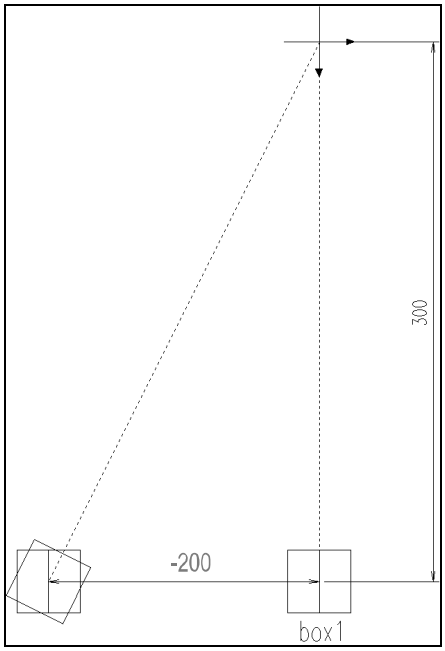


Figure 5-2

Figure 5-2 shows the reason for the shift in the angle in positions #13 and #23.

When the robot grips cube #1 and moves to place it on top of cube #3, the gripper (and the cube) rotate with it. To correct the error, the robot should turn its gripper to keep it in line with cube #3. The amount of rotation needed can be determined by finding the size of the angle whose tangent is calculated by:

$$tg\ a = \frac{-200}{300} = -0.66 = a = -33.69°$$

Q *How many degrees should the gripper be rotated in position #13 so that cube #1 will be aligned with cube #3?*

Q *How many degrees should the gripper be rotated in position #23 so that cube #2 will be aligned with cube #1 and #3?*

Task 5-2: Modifying Positions #13 and #23 by Calculating and Teaching the Roll

1. Change the Remark commands to indicate that the file is now applicable to Activity 5.
2. Then save the file as **USER5**. This file contains the previous program and the previous positions.
3. From the SCORBASEpro menu, select **Window | Simulation & Teach**.
4. Reset the robotic cell.
5. Open the robot gripper.
6. Use the Go Position tool from the Teach Positions dialog box to move the TCP to position #13.
7. From the Graphic Display window, zoom in to better view the relative position of the gripper and cube #3.
8. In the Teach Positions dialog box, do the following:
   - Click the Expand button.
   - Click the Get Position button.
   - The position data is displayed. The three coordinates (in mm) are on top and the two angles are just below them.
   - In the Roll [deg] field, enter **–33.69**.
   - Click the Teach button.
   - Position #13 is modified so that the coordinates of the position remain but the angle of the roll is adjusted (thus the tool is rotated).
   - Click Go Position.
   - The gripper turns and the gripper jaw stop when they are parallel to cube #3.
   - Change the position number to 23.
   - Click Go Position.
   - Note how the gripper turns back.
   - Click Get Position.
   - Modify the roll degree of position #23 to **-33.69**.
   - Click Teach.
   - Click Simple.

Task 5-3: Modifying Position #2

In this task, you will move the gripper to position #3 by manipulating the gripper so that it will approach the cube from above (for example, through position #12).

1. In the Teach Positions dialog box, do the following:
   - With the gripper open, set the position number to 12.
   - Click Go Position to point #12, which was recorded with a Z+ offset from position #2.
   - Set the position number to 2.
   - Click Go Position to move the robot to position #2.
2. From the Graphic Display window, redirect the camera to cube #2 and then zoom in. Try to get a good view from above the cube so that you can clearly see the cube and the gripper.

   Pay special attention to the angle in which the gripper jaw is approaching the cube.
3. In the XYZ mode of the Manual Movement dialog box, click buttons **5** and **T** to rotate the gripper. To increase accuracy, you may wish to slow the speed via the Speed field.
4. When the gripper jaw is parallel with the cube, click Record to record the new position as position #2.
5. From the SCORBASEpro menu, select File | Save to update changes.

   Note that the program is intact; only the roll angle of the three positions changed.

Task 5-4: Running the Program

1. Reset the robotic cell.
2. Using the single cycle icon, run through the entire program.

   Pay close attention to the following:
   - The relative position in which cube #1 is placed on top of cube #3.
   - The angle between the gripper jaw and cube #2.
   - The relative position in which cube #2 is placed on top of cube #1.

Q *Describe the relative position in which cube #1 is placed on top of cube #3.*

Q *Describe the angle between the gripper jaw and cube #2.*

Q *Describe the relative position in which cube #2 is placed on top of cube #1.*

Task 5-5: Team Discussion and Review

Q *Calculate the necessary Roll angle for position #2.*

Q *Hint: Use the following figure.*

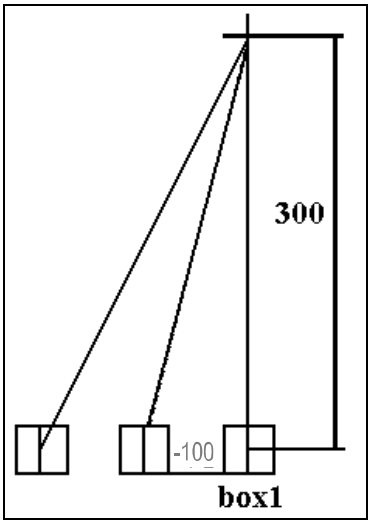


Figure 5-3

Task 5-6: Shut Down

1. Exit RoboCell.
2. Then turn off the computer.

ACADEMICS

Industrial Applications

Uses for Robots

In 1995 about 700,000 robots were operating in the industrialized world. Over 500,000 were used in Japan, about 120,000 in Western Europe, and about 60,000 in the United States. Many robot applications are for tasks that are either dangerous or unpleasant for human beings. In medical laboratories, robots handle potentially hazardous materials, such as blood or urine samples. In other cases, robots are used in repetitive, monotonous tasks in which human performance might degrade over time. Robots can perform these repetitive, high-precision operations 24 hours a day without fatigue. A major user of robots is the automobile industry. General Motors Corporation uses approximately 16,000 robots for tasks such as spot welding, painting, machine loading, parts transfer, and assembly. Assembly is one of the fastest growing industrial applications of robotics. It requires higher precision than welding or painting and depends on low-cost sensor systems and powerful inexpensive computers. Robots are used in electronic assembly where they mount microchips on circuit boards.

Activities in environments that pose great danger to humans, such as locating sunken ships, prospecting for underwater mineral deposits, and active volcano exploration, are ideally suited to robots. Similarly, robots can explore distant planets. NASA's Galileo, an unpiloted space probe, traveled to Jupiter in 1996 and performed tasks such as determining the chemical content of the Jovian atmosphere.

Robots are being used to assist surgeons in installing artificial hips, and very high-precision robots can assist surgeons with delicate operations on the human eye. Research in telesurgery uses robots, under the remote control of expert surgeons, that may one day perform operations in distant battlefields.

"Robot,"* Microsoft® Encarta® 97 Encyclopedia. © *1993-1996 Microsoft Corporation. All rights reserved.

Activity 6

Recording Relative Positions

OBJECTIVES

In this activity you will accomplish the following:

- Teach positions relative to current robot positions.
- Add remarks to a program.
- Pause program execution using the Wait command.
- Program the robot to simulate the immersion of an object in a corroding acid.

SKILLS

In this activity you will develop the following skills:

- Academic & Employability:
  - Operate lab equipment according to safety regulations.
  - Document inventory and safety procedures for lab set-up and shutdown.
  - Identify historical development of robotics industry.
- Occupational & Technical:
  - Write a program that uses relative positions to move the robot through a field.
  - Use visual commands to record the initial cylinder position.
  - Write a program to execute a tank-dipping simulation.
  - Modify the design of the program to improve the design process.
  - Implement modifications to meet changes in design criteria.

MATERIALS

In this activity you will need the following materials:

- Computer with RoboCell software
- Diskette or personal subdirectory on computer hard drive
- Worksheets for Activity 6

Absolute and Relative Positions

So far you have learned how to record **absolute robot positions**. These are positions whose coordinates are fixed. You have had to record/teach all the positions needed to guide the robot to move the tool from one place to another. Using these two methods can be quite exhausting when you need to record many positions for a complex robotic application.

In this activity, you will work with **relative positions** -- positions whose coordinates define a specific offset from another position. A relative position is linked to a reference position. If the coordinates of the reference position change, the relative position moves along with it, maintaining the same offset.

Relative positions are useful when programming the path of the robot for pick-and-place tasks. Intermediate positions along the path can often be defined as relative positions. For example, a relative position defined as a Z-offset of a few centimeters from the pick position will enable the robot to approach and leave the pick location without hitting other equipment in the system. If the pick position has to be adjusted and rerecorded, it will not be necessary to readjust and rerecord the relative position (above it).

Using Relative Positions in Programming

In this activity, you will program a robot to dip a cylinder (product) in four tanks full of corrosive acid, as shown in Figure 6-1.

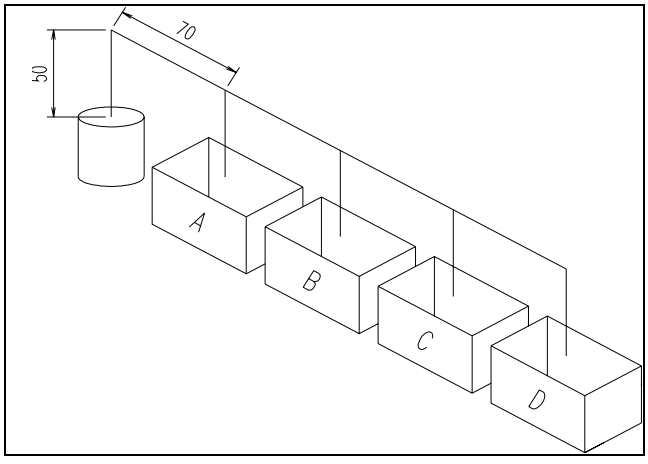


Figure 6-1

The robot will pick the cylinder, dip it in each of the four tanks for a period of five seconds and then place it at its initial position.

You will not see actual tanks in the robotic cell; imagine that they exist.

In this task, a robot is an excellent alternative to human operators for two reasons:

1. The acid can be dangerous to humans.
2. The robot will ensure accurate dipping time for every cylinder.

If you were to use the program commands learned thus far, you would need to record the following 10 absolute positions to perform the task:

Initial cylinder position

50 mm above the initial cylinder position

Inside tank A

50 mm above tank A

Inside tank B

50 mm above tank B

Inside tank C

50 mm above tank C

Inside tank D

50 mm above tank D

In this activity, however, you will work with relative positions to simplify the programming. You will need to record only one absolute position and three relative positions. The absolute position (position #1) will be 50 millimeters above the cylinder, and the relative positions will be:

- Position #2 - 50 mm below the current TCP position.
- Position #3 - 50 mm above the current TCP position.
- Position #4 - 70 mm to the right of the current TCP position.

The program will consist of the following:

Move to position #1

Move to position #2 - move down 50 mm

Close the gripper - grips the cylinder

Move to position #3 - move up 50 mm

Move to position #4 - move to the right 70 mm

Move to position #2 - move down 50 mm to tank A

Wait 5 seconds

Move to position #3 - move up 50 mm out of tank A

Move to position #4 - move to the right 70 mm

Move to position #2 - move down 50 mm to tank B

Wait 5 seconds

Move to position #3 - move up 50 mm out of tank B

Move to position #4 - move to the right 70 mm

Move to position #2 - move down 50 mm to tank C

Wait 5 seconds

Move to position #3 - move up 50 mm out of tank C

The Wait Command

Relative positions will help you guide the robot to the various tanks for dipping. However, you still need to determine how to define the dipping time for each tank.

The **Wait** (**WT**) command provides an excellent solution. The Wait command halts the robot at certain points in the program for a pre-specified time, allowing other machines to perform a task. The time is entered in tenths of seconds, as only integer numbers are allowed. Therefore, the cylinders can be "dipped" in the tanks for a specified amount of time and then the program will begin where it left off.

You will also use the Remark command in this program. In addition to documenting the activity number and name, you will use a remark to denote the next tank in which the cube will be dipped.

PROCEDURES

Task 6-1: Running RoboCell and Opening the 3D model File

1. Turn on the computer and run RoboCell.
2. Open the 3D model file **ACT06.3DC**.
3. Redirect the camera to the red cylinder.
4. Use the viewing tools to clearly see the cylinder and robot gripper from a comfortable viewing angle.

Task 6-2: Recording Positions

1. Open the robot gripper.
2. Send the robot to the red cylinder (using the Send Robot to Object option).
3. Using the Teach Positions dialog box, you will now record position #3 as a relative position, located 50 mm above the current robot position:
   - Click Expand.
   - In the Position Number field, enter **3**.

- Select **Relative To** and select **Current** for the reference position (from the Relative To field).
- In the Z (mm) field, enter **50**.
- Click Teach to teach position #3.
- Click the Go Position button.
- The robot moves up 50 mm.

Q *What would happen if you clicked again on the Go Position button?*

4. Using the Teach Positions dialog box, you will now record position #1 as an absolute position:
   - In the Position Number field, enter **1**.
   - Select **Absolute**.
   - Click the Record button.
   - You have now finished recording position #1.
5. Teach position #2 in the same way you recorded position #3. It should be relative to the Current position with a Z-offset of -50 mm.
6. Teach position #4 as relative to the current position with a Y-offset of +70 mm.

   Make sure that the coordinate fields are (0,0,0) before adding the Y-offset. If they are not, click again on Relative To.
7. From the SCORBASEpro menu, select View | List Positions.

   Select XYZ.

Q *What information does this dialog box provide?*

8. Close the List Positions dialog box.

Task 6-3: Programming

The program you will create contains many similar segments. You will use the Copy & Paste tools to duplicate a segment three times.

1. Select **Window | Teach & Edit**.
2. Enter the necessary Remark commands to document the program.
3. Double click on Open Gripper.
4. Select Go Position #1 fast.
5. Select Go Position #2 speed 5.
6. Double click on Close Gripper.
7. Select Go Position #3 fast.

Now you will program the procedure for dipping the cylinder in tank A.

8. Add a remark stating **START A**, to indicate that the robot is now dipping the cylinder in tank A.
9. Select Go Position #4 fast.
10. Select Go position #2 speed 5.
11. Double click on **WT (Wait)**, also located in the Program Flow section of the Command List.

    The Wait dialog box opens (Figure 6-2).

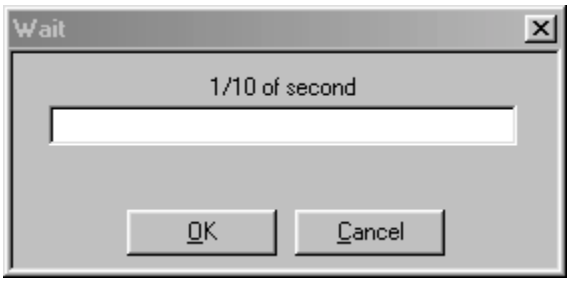


Figure 6-2

Enter **50**. 50 tenths of a second will create a 5 second pause inside the tank.

Click OK to close the dialog box.

12. Select Go Position #3 fast.

    Dipping cylinder A is completed. You must now program the dipping of the cylinder in tanks B, C and D.

13. The procedure for dipping will be identical to the commands you just entered. To simplify programming, use the Copy & Paste tools to copy the program lines starting from the remark and to paste them at the end of the program:
    - In the Program window, click and hold down the mouse button, highlighting lines 10-14.
    - Right click to copy the selected lines, or select Copy from the SCORBASEpro Edit menu.
    - Place the cursor at the end of the program.
    - Right click to paste the lines, or select Paste from the SCORBASEpro Edit menu.
    - Repeat twice more to paste the lines for tanks C and D.
14. Now you still have to change the three last remarks from A to B, C and D.

    Double click on the second START A line.

The Remark dialog box opens.

Change START A to **START B** and click OK.

15. Repeat step #1 to change the third and fourth remarks to START C and START D respectively.

Q *How will you now order the robot to return to its initial position?*

16. Add commands at the end of the program to return the robot to its initial position, place the cylinder there and then move up 50 mm.

17. Compare the program you just assembled with the following:

```
 1    Remark: **********************************
 2    Remark: ACT6
 3    Remark: Recording Relative Positions
 4    Remark: **********************************
 5    Open Gripper
 6    Go to Position 1 fast
 7    Go to Position 2 speed 5
 8    Close Gripper
 9    Go to Position 3 fast
10    Remark: START A
11    Go to Position 4 fast
12    Go to Position 2 speed 5
13    WaiT 50 ( 10 ths of seconds )
14    Go to Position 3 fast
15    Remark: START B
16    Go to Position 4 fast
17    Go to Position 2 speed 5
18    WaiT 50 ( 10 ths of seconds )
19    Go to Position 3 fast
20    Remark: START C
21    Go to Position 4 fast
22    Go to Position 2 speed 5
23    WaiT 50 ( 10 ths of seconds )
24    Go to Position 3 fast
25    Remark: START D
26    Go to Position 4 fast
27    Go to Position 2 speed 5
28    WaiT 50 ( 10 ths of seconds )
29    Go to Position 3 fast
30    Go to Position 1 fast
31    Go to Position 2 speed 5
32    Open Gripper
33    Go to Position 3 fast
```

Task 6-4: Running the Program

1. Save the program and positions as file **USER6**.

2. Reset the robotic cell.

3. Click the Run Single Line icon seven times (until the robot is in position #2).
4. Select **3D Image | Show Path**.

   As mentioned in the Overview, actual tanks do not appear in the cell. However, this option will show on-screen the robot path and help you better visualize the tanks and dipping procedure.
5. Continue executing the program by clicking on the Run Single Line icon. When the program finishes executing line #29, disable the Show Robot Path option by reselecting it from the menu (the checkmark should disappear).

   Make sure that you disable the option but don't clear the robot path.
6. Run the program until its end.

Task 6-5: Team Discussion and Review

- **Q** *The production procedure of another product requires that the product be dipped in tank A, B and D (skipping tank C). Modify the program accordingly.*
- **Q** *Save the program and positions as* ***USER6A****.*

Task 6-6: Shut Down

1. Exit RoboCell.
2. Then turn off the computer.

ACADEMICS

Industrial Applications

The State of the Robotics Industry

In general, there was a significant slump in the mid to late 1980's in industrial robotics. However in the early 1990's, sales and numbers have rebounded to surpass early 1980 numbers and dollars.

From Motion Control Magazine April 1994

Robotics Industries Association said recently robot orders jumped 40% through June, 1993, as the industry posted its best opening half-year ever.... Net new orders received by U.S. based robotics companies totaled 3,640 robots valued at $306.2 million, the highest unit and dollar figures ever.

From the New York Times, Wednesday September 7th pC1 (paraphrased)

In the late 1980's a steep decline in robot orders drove most US companies out of the business. In the first half of 1994 4,335 robots with a total value of $383.5 million. Fanuc is the leader with about $360M in sales this year. Asea Brown Boveri (ABB) is second with sales estimated at $120M. The next several are Japanese: Motoman, Panasonic, Sony and Nachi.

The only major US producer to have survived is Adept Technology with about $50M in sales in a $700M market. The following table is interpreted from a graph in the article.

Net New Orders In US		
Year	**# of Robots**	**$US**
1984	5800	$480M
1985	6200	$380M
1986	5400	$320M
1987	3800	$300M
1988	4000	$325M
1989	4500	$510M
1990	5000	$510M
1991	4000	$410M
1992	5250	$500M
1993	6800	$630M
1994	4335 (6 mos.)	$383M (6 mos.)

From Industry Flash Vol1, No. 4, Dec 5, 1994:

Demand for U.S. Industrial Robots Surging: U.S.-based robotics companies are enjoying the best of times. The Robotics Industries Association (RIA) says surging demand recently led American robotic companies to their best nine-month totals ever.

Through September, new orders totaled 6,218 robots valued at $548 million, a 12 percent increase in units and 13 percent increase in revenue over the previous nine-month period last year. The greatest demand, says the trade group, is coming from U.S. manufacturers which are finally learning what the Japanese have known for years: robots can play a significant role in improving productivity, quality, flexibility and time-to-market. But, even though demand is surging and the U.S. is the world's second largest robotics user with some 53,000 systems, the Japanese have more than seven times as many robots in use, RIA says.

Dowling, Kevin (1996) "Robotics: comp.robotics Frequently Asked Questions" Available as a hypertext document at http://www.frc.ri.cmu.edu/robotics-faq. 90+ pages.

Activity 7

Recording More Relative Positions

OBJECTIVES

In this activity you will accomplish the following:

- Describe the construction of a gravity feeder.
- Describe the use of templates in robotic systems.
- Record positions as relative to other positions.
- Program the robot to load parts from a feeder to a template.

SKILLS

In this activity you will develop the following skills:

- Academic & Employability:
  - Operate lab equipment according to safety regulations.
  - Document inventory and safety procedures for lab set-up and shutdown.
  - Identify industrial application of manipulators.
- Occupational & Technical:
  - Program a feeder and template simulation.
  - Use output commands to indicate when the feeder should release a cylinder.
  - Write a program to execute the simulation.
  - Run simulation and describe the results.
  - Utilize troubleshooting skills to improve the production process.

MATERIALS

In this activity you will need the following materials:

- Computer with RoboCell software
- Diskette or personal subdirectory on computer hard drive
- Worksheets for Activity 7

OVERVIEW

Feeder

In this activity, you will learn about two components often used in modern automated production processes: **feeders** and **templates**. You will also learn a new way to record programming positions.

Figure 7-1 shows a pneumatic feeder. Feeders are used to provide required materials or components to the production process. The feeder shown in Figure 7-1 is supposed to supply the production process with cubes.

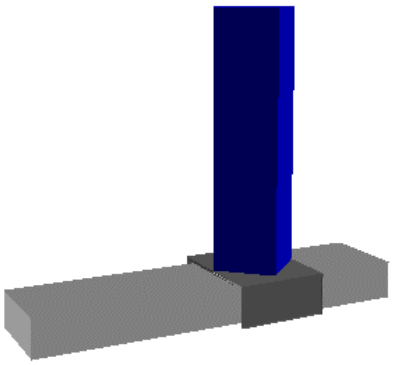


Figure 7-1

In the feeder, the cubes are stacked one on top of the other. When a cube is needed for the production process, the robot pulls out the bottom cube.

The robot feeder is considered an **output** device**.** Output devices receive control signals from the system. Following the removal of the bottom cube, an output is turned on, and another cube is sent to the bottom by the robot controller. Feeders are periodically filled with new cubes, thus providing a continuous cube supply.

A major advantage of the feeder is that the position of the available cubes remains the same. When a cube is taken from the feeder, another cube immediately replaces it. When another cube is needed, the robot needs only to return to exactly the same position and repeat its procedure.

Template

Templates are used to hold and store various products in modern production and manufacturing systems. Figure 7-2 shows a simple template.

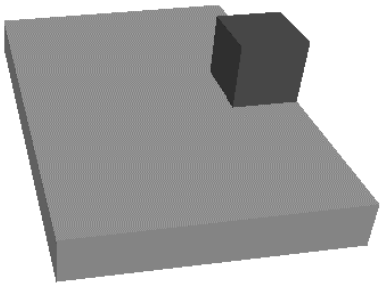


Figure 7-2

A template is a plain platform on which parts can be placed in various positions using special mounting brackets (when needed). The template is fitted with a handle that the robot can grasp in order to move the template (with the part attached) to different positions.

Templates offer a number of advantages:

- Since all parts are placed on a standardized template, the same robots with the same tool can be used to move the templates (and the parts) from one place to another.
- In most cases, templates have identification codes that are used to identify the part placed on the template. The identification codes ensure efficient and productive processing.

Using a Feeder and Template in a Production Process

In this activity, you will load two cylinders from the gravity feeder and place them on a template. When the cylinders are placed on the template, the robot will be able to move the two cylinders in one cycle.

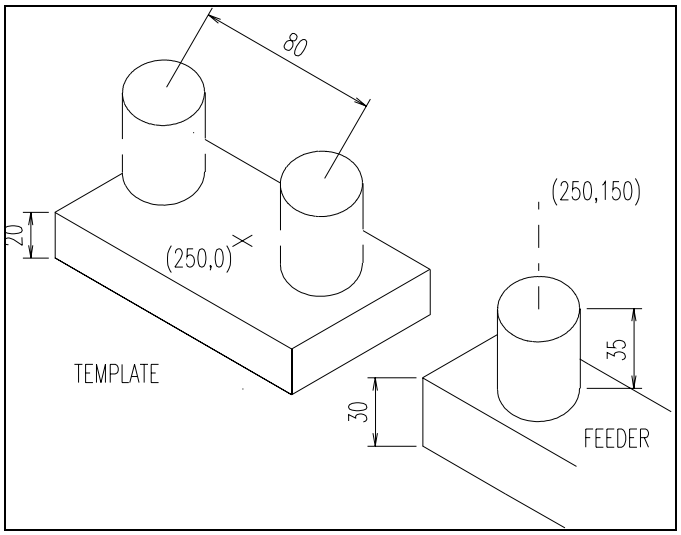


Figure 7-3

The robot working procedure should be as follows:

1 The robot's gripper will move to above the cylinder's initial position (at the feeder's mouth), lower 40 mm, grip the cylinder and move up again.

2 The robot will move to above the destination of cylinder #1, lower 50 mm, open the gripper and then move back up.

3 The robot's gripper will move back to above the new cylinder at the feeder's mouth, lower 40 mm, grip the cylinder and move back up.

4 The robot will move to above the destination of cylinder #2, lower 50 mm and then move back up.

Recording the positions for this task could be quite difficult. If you chose to record the positions as absolute, you would need to record six positions (the initial cylinder location at the mouth of the feeder, the two end positions and a position above each point). Teaching these six positions would be complicated and mistake-prone, primarily because the positions are not provided and would need to be calculated.

For these reasons, you will use the Relative To position recording method. As you saw in the previous activity, this method allows you to record positions as *relative to a given position*. Note that relative positions can be considered as absolute positions provided the reference position remains unchanged.

PROCEDURES

Task 7-1: Running RoboCell and Opening the 3D model File

1. Turn on the computer and run RoboCell.
2. Open the 3D model file **ACT07.3DC**.
3. Click on **View | Dialog Bars | Digital Outputs**. The Digital Outputs window is displayed at the bottom of the screen

   Click on the number **2** to activate the feeder, which calls for a red cylinder.
4. Redirect the camera to the red cylinder, located at the mouth of the feeder.
5. Use the viewing tools to clearly see the cylinder from a comfortable viewing angle.

Task 7-2: Recording Positions

1. Open the robot gripper.
2. Send the robot above the red cylinder.
3. Then send the robot to the red cylinder.

   The robot is now in position #1.
4. Record position #1 as Absolute.
5. Teach position #11 as relative to position #1 with a Z-offset of +40 mm.

6. Record the remaining positions as defined by Table 7-1.

Position #	Description	Relative Dimensions
1	initial cylinder position	use the Send feature
11	above initial cylinder position	Relative to position #1 (z = 40)
2	final position cylinder #1	Relative to #1 (y = -110, z = -10)
12	above final position cylinder #1	Relative to #2 (z = +50)
3	final position cylinder #2	Relative to #2 (y = -80)
13	above final position cylinder #2	Relative to #3 (z = +50)

Table 7-1

Q *Why is the robot ordered to lift the cylinder 40 mm?*

Q *Why should the cylinder be lowered 50 mm to the template?*

Task 7-3: Programming and Running the Program

1. Write the program yourself. Note that large portions of the program will be identical and you can use the cut and paste tools to avoid repetitive programming.

   To write this program, you must use two new program commands, **Turn output on (ON)**, and **Turn output off (OF.)** These commands are found in the Inputs & Outputs section. Double click ON to insert the turn input on command. This tells the program to send an output from the controller to the feeder. This will release the cylinder from the feeder.

   You must also tell the program to turn off the output. If the output is not turned off, you will not be able to turn it back on to release the next part. To insert the turn output off command, double click on OF.

2. When you have finished programming, compare your program with the following:

```
 1    Remark: **********************************
 2    Remark: ACT7
 3    Remark: Recording More Relative Positions
 4    Remark: **********************************
 5    Open Gripper
 6    Turn on output 2
 7    Turn off output 2
 8    Go to Position 11 fast
 9    Go to Position 1 speed 5
10    Close Gripper
11    Go to Position 11 fast
12    Go to Position 12 fast
13    Go to Position 2 fast
14    Open Gripper
15    Go to Position 12 fast
16    Go to Position 11 fast
17    Go to Position 1 speed 5
18    Turn on output 2
19    Turn off output 2
20    Close Gripper
21    Go to Position 11 fast
22    Go to Position 13 fast
23    Go to Position 3 speed 5
24    Open Gripper
25    Go to Position 13 fast
```

3. Save the program and positions as file **USER7**.
4. Run a single cycle of the program.

Task 7-4: Team Discussion and Review

Q *According to Figure 7-3, the distance between the center of the two cylinders placed on the template is 80 mm. This value was used to calculate the distance from position #1 to position #2.*

Q *In industry, however, it's customary for positions to be recorded relative to given data, rather than recalculating the coordinates.*

Q *How would you record position #2 as a relative position based on the above data?*

Task 7-5: Shut Down

1. Exit RoboCell.
2. Then turn off the computer.

Industrial Applications

Robot Manipulators

The inspiration for the design of a robot manipulator is the human arm, but with some differences. For example, a robot arm can extend by telescoping—that is, by sliding cylindrical sections one over another to lengthen the arm. Robot arms also can be constructed so that they bend like an elephant trunk. Grippers, or end effectors, are designed to mimic the function and structure of the human hand. Many robots are equipped with special purpose grippers to grasp particular devices such as a rack of test tubes or an arc-welder.

The joints of a robotic arm are usually driven by electric motors. In most robots, the gripper is moved from one position to another, changing its orientation. A computer calculates the joint angles needed to move the gripper to the desired position in a process known as inverse kinematics.

Any robot designed to move in an unstructured or unknown environment will require multiple sensors and controls, such as ultrasonic or infrared sensors, to avoid obstacles. Robots, such as the National Aeronautics and Space Administration (NASA) planetary rovers, require a multitude of sensors and powerful onboard computers to process the complex information that allows them mobility. This is particularly true for robots designed to work in close proximity with human beings, such as robots that assist persons with disabilities and robots that deliver meals in a hospital. Safety must be integral to the design of human service robots.

"Robot,"* Microsoft® Encarta® 97 Encyclopedia. © *1993-1996 Microsoft Corporation. All rights reserved.

Activity 8

Recording Positions for Peripheral Devices

OBJECTIVES

In this activity you will accomplish the following:

- Describe the term work envelope.
- Record positions of peripheral devices
- Control a rotary table.

SKILLS

In this activity you will develop the following skills:

- Academic & Employability:
  - Operate lab equipment according to safety regulations.
  - Document inventory and safety procedures for lab set-up and shutdown.
  - Identify industrial application of off-line programming, calibration, and simulation.
- Occupational & Technical:
  - Record positions for peripheral devices.
  - Define work envelope and relate to robotics applications.
  - Use visual commands to view the work envelope.
  - Program a robot arm and rotary table simulation.
  - Modify the design of a program to improve the design process
  - Implement modifications to meet changes in design criteria.

MATERIALS

In this activity you will need the following materials:

- Computer with RoboCell software
- Diskette or personal subdirectory on computer hard drive
- Worksheets for Activity 8

Work Envelope

So far you have only operated the robot in a "stand alone" station. When a robot works as a stand alone, its capabilities are quite limited. The robot can only move a tool within a 3-D environment, reaching a limited range of positions. The positions within the robot's reach are a function of its arm length and structure. This range of accessible positions is known as the robot's **Work Envelope**.

The Work Envelope of a robot is defined as the span of the robot's working range. It is shown as a geometrical shape containing every position within the robot's reach.

In Figure 8-1, the robot arm of the SCORBOT-ER 4U is fully extended.

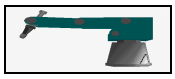


Figure 8-1

The overall length of the arm defines the work envelope. When the robot arm moves up and is fully extended, the TCP moves in an arched trajectory to the position shown in Figure 8-2.

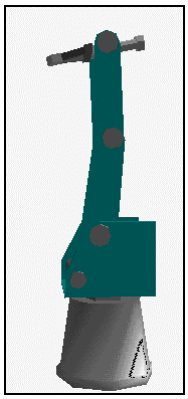


Figure 8-2

Theoretically the TCP of the SCORBOT-ER 4U can reach any position found in a sphere whose radius equals the length of its extended arm. In practice, however, the number of the positions within reach is much smaller due to mechanical limitations. Nevertheless, this type of robot is still defined as a robot with a *spherical work envelope*.

It is important to emphasize that extended arms increase the robot's work envelope but reduce its accuracy and payload.

One of the ways to increase a robot's work envelope without increasing its arm length is by adding accessories to the robot known as **Peripheral Devices**.

Rotary Table

The **Rotary Table** is an excellent example of a peripheral device capable of moving objects in and out of the robot's work envelope. The rotary table is constructed from a round disk that can be rotated by a motor, often controlled by the robot controller (shown in Figure 8-3).

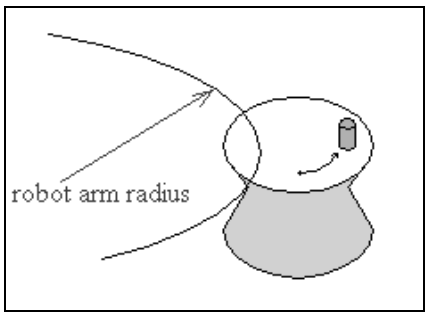


Figure 8-3

Note the cylinder placed on the table in the figure. This cylinder is presently out of the robot's reach; the robot cannot pick and place it. However, if the table were to be rotated, the cylinder would then be within the robot's work envelope, allowing the robot to pick and place it where necessary.

Using a Rotary Table to Stack Cylinders

In this activity, you will program the robot to stack four cylinders (with a height of 35 millimeters each) to form a "tower." To facilitate efficient space utilization, the cylinders are placed on a rotary table. The cylinders will be stacked in the following order: pink base cylinder (already in position) - red - blue - yellow - green, as shown in Figure 8-4.

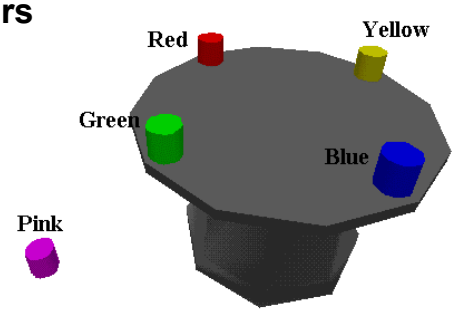


Figure 8-4

RoboCell allows you to record positions both for the robot and the peripheral equipment. In order to perform the task of loading the cylinders and stacking them, you will record eleven different positions -- seven for the cylinders and four for the rotary table:

- Position #1: for picking the green cylinder.
- Position #11: above the green cylinder.
- Position #2: red cylinder final position.
- Position #12: blue cylinder final position.
- Position #22: yellow cylinder final position.
- Position #32: green cylinder final position.
- Position #42: above the green cylinder final position.

On the rotary table, the following four positions will be recorded:

- Position #3: current table position.
- Position #4: red cylinder in the initial position of the green cylinder.
- Position #5: yellow cylinder in the green cylinder's initial position.
- Position #6: blue cylinder in the green cylinder's initial position.

The red cylinder (as well as remaining cylinders) will be stacked by the following procedures:

- Sending the robot to the green cylinder's initial position (position #11).
- Sending the table to position #4 so that the red cylinder will be in the place of the green one.
- Lowering the robot to position #1.
- Closing the gripper to pick the red cylinder.
- Placing the red cylinder on top of the pink base cylinder (position #2).

PROCEDURES

Task 8-1: Running RoboCell and Opening the 3D model File

1. Turn on the computer and run RoboCell.
2. Open the 3D model file **ACT08.3DC**.
3. Adjust the viewing angle of the robotic cell to a comfortable viewing angle.
4. From the RoboCell menu, select **3D Image | Show Robot Work Envelope**.

   Select **Top View** from the Graphic Display menu to view the robotic cell from the top.

   The robot's work envelope is drawn with a blue line, enabling you to see which objects are within the robot's reach.
5. Zoom in to see the work envelope as clearly as possible.

6 To understand the concept of the work envelope, open the robot gripper and then try using the Send options to send the robot to the various cylinders placed on the round rotary table.

Q *Which cylinders does it reach on the rotary table?*

Q *Which doesn't it reach? What message appears in the status bar of the Graphic Display window?*

Q *What does that say about the work envelope of the robot?*

As you can see, only the green and pink cylinders can be reached and are therefore within the robot's work envelope. To enable the robot to pick the other cylinders, the rotary table must be rotated in order to move the other cylinders into the robot's work envelope.

Q *Is it possible to perform this task without a rotary table?*

Q *What is the advantage of using a rotary table?*

7 Disable the Show Robot Work Envelope option.

8 Use the viewing tools to see an unobstructed view of the rotary table with cylinders placed on it, robot and table with base cylinder.

Task 8-2: Recording Positions for the Robot and Peripheral Devices

Q *How will you load the blue cylinder?*

1 Open the robot gripper.

2 Send the robot to the green cylinder and record this as absolute position #1.

In the Include Axes section, make sure that only robot (and not peripherals) is selected!

Position #1 is therefore located next to the green cylinder.

3 Teach position #11 as relative to position #1 with a Z-offset of +40 mm.

Position #11 is located above the green cylinder.

4 Send the robot to the pink base cylinder and record the absolute position as position #2.

Position #2 is 35 mm above the current TCP position.

5 Click Expand, Get Position and then add 35 to the Z coordinate of position #2 and teach (overwrite) the position.

6 Teach position #12 as relative to position #2 with a Z-offset of +35 mm. If necessary, click Clear to clear the contents of the coordinates.

This is the final position for the blue cylinder.

7. Teach position #22 as relative to position #12 with a Z-offset of +35 mm.

   This marks the final position of the yellow cylinder.

8. Teach position #32 as relative to position #22 with a Z-offset of +35 mm.

   Position #32 marks the final position of the green cylinder.

9. Teach position #42 as relative to position #32 with a Z-offset of +35 mm.

   You will now teach the four rotary table positions, using the Peripheral command for the first time.

10. Reset the robotic cell.

11. Then go to position #11.

12. Open the robot gripper.

    The robot is now standing in its position #11. For the table, it will be recorded as position #3 (the initial position of the green cylinder).

13. In the Teach Positions dialog box, do the following:

    - In the Position Number field, enter **3**.
    - Select **Peripheral** to indicate that you are recording the position for peripheral equipment.
    - Cancel the default selection **Robot** to ensure that you will record only the position of the peripheral equipment.
    - Click **Absolute**.
    - Click Record.
    - The rotary table's initial position is now recorded as position #3.

14. Recording the next three positions will take some maneuvering. As defined by the Overview, positions #4, #5 and #6 will be recorded as peripheral positions as they define the position of the rotary table. Before recording each position, you will need to rotate the rotary table counter-clockwise such that a new cylinder will then be located in the green cylinder's initial position (position #1).

    To record position #4 (where the red cylinder is in place of the green), do the following:

    - Send the robot to position #11 again.
    - Open the robot gripper.
    - Use the viewing tools to clearly observe the relationship between the green cylinder and the jaw of the robot gripper.

- Using the buttons in the Manual Movement dialog box, click **8**/**I** to rotate the table until the red cylinder is in the initial position of the green cylinder. You must center the cylinder between the gripper's jaw as much as possible.
- In the RoboCell robotic cell, the rotary table is known as axis 8. As you can see from the Joints mode of the Manual Movement dialog box, axis 8 is controlled by the 8 and I keys (clockwise and counter-clockwise rotation).
- To make sure that the cylinder is centered between the gripper's jaw, move the robot to position #1 and click Close Gripper to make sure that the robot can easily pick the cylinder from the table in that position.
- Record this **peripheral** position as position #4 (the red cylinder). Don't forget to cancel the Robot default in the Teach Positions dialog box.

15 Record peripheral positions #5 (blue cylinder) and #6 (yellow cylinder) applying the same technique used to record position #4. Don't forget to open the robot gripper before starting.

Task 8-3: Programming

1 Write a program that will command the robot to do the following:

- Move the robot to position #11
- Move the rotary table so that the right cylinder is below the gripper
- Lower the gripper, close it and then place the cylinder.

Note that the same process is repeated four times in the program, differing only in the rotary table position and the final cylinder position. Use the Cut & Paste tools to copy the segment and then modify the necessary two lines.

2 Compare what you just wrote with the following program:

```
 1    Remark: ******************************
 2    ACT8
 3    Remark: Recording Positions for Peripheral Devices
 4    Remark: ******************************
 5    Open Gripper
 6    Remark: STARTING OF RED CYLINDER
 7    Go to Position 11 fast
 8    Remark: robot above rotary table
 9    Go to Position 4 fast
10    Remark: rotary table in position
11    Go to Position 1 speed 5
12    Close Gripper
13    Go to Position 11 fast
```

```
14    Go to Position 42 fast
15    Go to Position 2 speed 5
16    Remark: cylinder in place
17    Open Gripper
18    Go to Position 42 fast
19    Remark: STARTING OF BLUE CYLINDER
20    Go to Position 11 fast
21    Remark: robot above rotary table
22    Go to Position 5 fast
23    Remark: rotary table in position
24    Go to Position 1 speed 5
25    Close Gripper
26    Go to Position 11 fast
27    Go to Position 42 fast
28    Go to Position 12 speed 5
29    Remark: cylinder in place
30    Open Gripper
31    Go to Position 42 fast
32    Remark: STARTING OF YELLOW CYLINDER
33    Go to Position 11 fast
34    Remark: robot above rotary table
35    Go to Position 6 fast
36    Remark: rotary table in position
37    Go to Position 1 speed 5
38    Close Gripper
39    Go to Position 11 fast
40    Go to Position 42 fast
41    Go to Position 22 speed 5
42    Remark: cylinder in place
43    Open Gripper
44    Go to Position 42 fast
45    Remark: STARTING OF GREEN CYLINDER
46    Go to Position 11 fast
47    Remark: robot above rotary table
48    Go to Position 3 fast
49    Remark: rotary table in position
50    Go to Position 1 speed 5
51    Close Gripper
52    Go to Position 11 fast
53    Go to Position 42 fast
54    Go to Position 32 speed 5
55    Remark: cylinder in place
56    Open Gripper
57    Go to Position 42 fast
```

Task 8-4: Running the Program

1. Save the program and positions as file **USER8**.
2. Reset the robotic cell.
3. Run a single cycle of the program.

Task 8-5: Team Discussion and Review

Q *In this activity, the program you wrote can be easily modified to increase the efficiency of the process. By examining the process, you can see that after placing the cylinder, the robot moves to a position above the table (position #11) and only then the table turns to place the next cylinder under it.*

Q *Modify the program to increase productivity by cutting down the time.*

Hint: You can record a single position for the robot and peripheral equipment.

Save the file as USER8A.

Task 8-6: Shut Down

1. Exit RoboCell.
2. Then turn off the computer.

ACADEMICS

Industrial Applications

Off-line Programming and Robot Calibration - The Car Industry

The major users of robots have traditionally been the car companies whose product is unique in that starts as a mass produced sheet of metal whose complexity requires intensive and advanced automation techniques with a short set-up time. In North America, the 'big three' car makers are Chrysler, Ford, and General Motors. Based in Detroit and elsewhere in the continent, these companies have spent the past five years investing heavily in robot simulation and off-line programming. The urgency exerted by an unexpected demand for the most popular models as well as a large upturn in consumer spending on cars has created a pressure-cooker atmosphere in Detroit. Here production lines are designed and built and running in weeks or months and every off-line programming technique is examined closely for potential improvements in set-up time or production rate.

Current practice involves several stages from simulation to off-line programming. First the car is modeled in the company-wide CAD system. Then the fixtures are designed which are to present the parts to the welding robots in a repeatable and predictable manner. The first fixture is then installed on the production line, and from this the position of the first robot is defined. The position of the first fixture then defines the position of the second fixture, and so on, all the way down the production line.

The simulation of the robot programs has usually been completed by this stage using models imported from the CAD system and off-the-shelf robot models purchased from the simulation vendor. The simulation is used to check that the robots can reach all the required positions. The simulation enables the user to plan collision free paths around the clamps holding the part using via points which must be created in such a way that they do not add excessively to the overall cycle time. This is the reason that is so important to derive reasonably accurate cycle times from the simulation.

Downloading the robot programs from the simulation to the robot controller involves taking a disk containing the program, teachpoint, and tool frame files from the air-conditioned offices where the CAD and simulation specialist work down onto the much harsher and unforgiving environment of the factory.

It is here that most advocates of simulation and off-line have a rude awakening. Programs that work fine in the 'virtual reality' environment of the robot simulation seldom work in the uncertain and unpredictable and subtly different real world.

Robots are designed to be extremely repeatable but not all that accurate. To increase the accuracy of the robot towards the same order as its repeatability it must be calibrated. Once the robot has been calibrated it can be used with a pointer of known length as an accurate measuring device to record the position of the fixture datum points. Finally, the correct tool offset must be defined on the robot controller and on the robot simulation.

The robot program created by the simulation is really only ready for downloading to the robot once all the above procedures are complete. If one of these steps is ignored or badly implemented, then the resulting program can be anything from 2 inches to a foot out, depending on the seriousness of the mistakes involved. This is where most of the horror stories involving robot simulation and off-line programming originate: 'we took a program from .... and downloaded it to the robot and the robot drove straight into the fixture' or 'we created 120 off-line programs for the whole production line and not one worked first time - they all needed to be touched up by hand'.

The final source of error in off-line programming is a simple logistic one: by the time the simulation is finished and ready for off-line programming the real workcell may been redesigned, turntables have been introduced or removed, additional robots have been inserted. Only now its too late to go back and simulate because the simulation office is working on simulations for the next production line. The simplest way to avoid this situation is to make certain of the following:

1. One person is responsible for the simulation from start to finish (the finish being when the production line is up and running smoothly and there are no more changes);
2. The simulation goes down onto the shop floor with the off-line program where last minute changes can be immediately introduced into the simulation.

So with all these potential pitfalls and inaccuracies in robot simulation and off-line programming, is it really worth attempting? If you want the answer, come to Detroit. The car companies will not spend money on any technology that doesn't produce savings that can be measured in dollars and cents, and they see robot simulation and off-line programming as the revolutionary technology that will enable them to build more cars, quicker and at a lower cost than ever before.

http://www.rosl.com/offline2.html

Activity 9

Recording Positions Using Encoder Values

OBJECTIVES

In this activity you will accomplish the following:

- Describe the construction and operation of an encoder.
- Describe how a position is recorded using an encoder value.
- Improve the accuracy of the positions recorded for the rotary table.
- Modify the program to increase accuracy and efficiency.

SKILLS

In this activity you will develop the following skills:

- Academic & Employability:
  - Operate lab equipment according to safety regulations.
  - Document inventory and safety procedures for lab set-up and shutdown.
  - Identify educational opportunities in robotics.
  - Use the Internet to research and gather information.
- Occupational & Technical:
  - Record four positions for a rotary table, based upon calculations.
  - Record and store position using encoder values.
  - Improve the accuracy of positions.
  - Modify the design of a program to improve accuracy and efficiency.
  - Monitor and analyze the operation of the system for quality control.

MATERIALS

In this activity you will need the following materials:

- Computer with RoboCell software
- Diskette or personal subdirectory on computer hard drive
- Worksheets for Activity 9

Encoders

In all the previous activities, you recorded the positions of the robot and its peripheral accessories. Recording positions in RoboCell (and with many industrial robots) is done using a device called an **Encoder**, as shown in Figure 9-1.

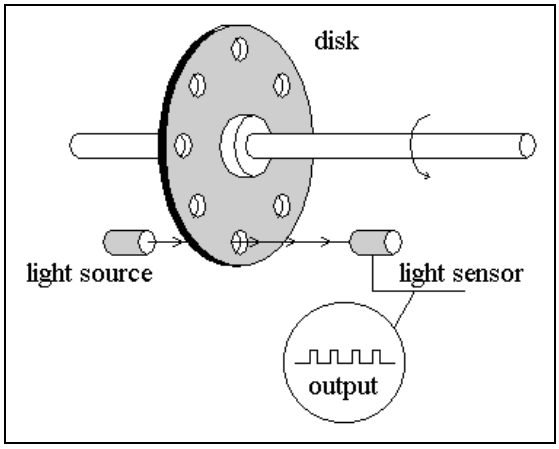


Figure 9-1

An Encoder is composed of a round disk with holes on its periphery. The disk is connected to the motor that drives one of the robot axes (or a peripheral device). One side of the disk is faced by a fixed light source; the other side faced by a light sensor. As shown in Figure 9-1, a beam of light passed through the hole in the disk will reach the sensor.

When the motor and the disk rotate, the path of light from the light source to the sensor is disturbed and the sensor receives light in the form of flashes (pulses). The number of pulses counted enables the encoder to determine how many turns (revolutions) the disk and the motor have completed.

For example, the encoder shown in Figure 9-1 has eight holes. When the sensor counts eight light pulses, it indicates that the axis has rotated a full cycle (provided the disk turned in one direction).

In the SCORBOT-ER 4U, encoders are used to measure and control the robot axes and its peripheral accessories. The number of light pulses counted by each light sensor is stored in the computer and is known as an **Encoder Value**. The encoder value is set to zero when the robot is homed (the cell is reset). When the axis turns in one direction, the encoder value increases in accordance with the number of pulses read. Likewise when the axis turns in the opposite direction, the encoder value decreases.

Recording and Storing Positions Using Encoder Values

In SCORBOT-ER 4U the manipulator movements are controlled by five independent DC motors, each fitted with its own encoder. When a robot position is recorded, five encoder values are stored in a position data table.

Table 9-1 shows an example position data table with three recorded positions. The values in this table represent the encoder values known when the position was recorded.

Pos. #	Encoder 1	Encoder 2	Encoder 3	Encoder 4	Encoder 5
1	1000	2000	3000	4000	0
2	2000	2000	2000	5000	0
3	2000	1000	2000	5000	0

If the robot is in position #1 and is ordered to move to position #2, then the controller will activate the axis motors as follows:

- The motor of axis 1 will be started in the + direction until 1000 pulses are counted. Then the motor will stop.
- The motor of axis 3 will be started in the - direction until 1000 pulses are counted. Then the motor will stop.
- The motor of axis 4 will be started in the + direction until 1000 pulses are counted. Then the motor will stop.

Note that all motors will start and stop simultaneously.

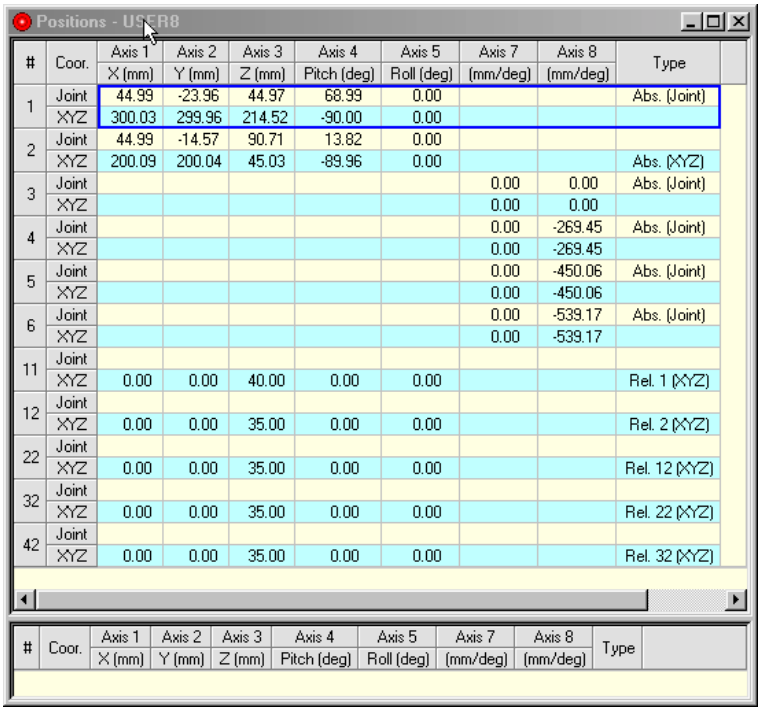
In this activity, you will load and run the program and positions file saved in Activity 8. You will then modify the program to do the following:

- Increase precision by using the encoder counts to record new positions.
- Shorten the cycle duration.

PROCEDURES

Task 9-1: Running RoboCell and Opening the 3D model File

1. Turn on the computer and run RoboCell.
2. Open the program and positions file **USER8.WS**. This is the program and positions file you saved in Activity 8.
3. Open the 3D model file **ACT8.3DC**. This file contains the same 3D model used in the previous activity.
4. Click on **View | List Positions** and the Positions dialog box is displayed (Figure 9-2). Note that both the Joint, as well as the XYZ, positions are displayed.



#	Coor.	Axis 1 X (mm)	Axis 2 Y (mm)	Axis 3 Z (mm)	Axis 4 Pitch (deg)	Axis 5 Roll (deg)	Axis 7 (mm/deg)	Axis 8 (mm/deg)	Type
1	Joint	44.99	-23.96	44.97	68.99	0.00			Abs. (Joint)
	XYZ	300.03	299.96	214.52	-90.00	0.00			
2	Joint	44.99	-14.57	90.71	13.82	0.00			
	XYZ	200.09	200.04	45.03	-89.96	0.00			Abs. (XYZ)
3	Joint						0.00	0.00	Abs. (Joint)
	XYZ						0.00	0.00	
4	Joint						0.00	-269.45	Abs. (Joint)
	XYZ						0.00	-269.45	
5	Joint						0.00	-450.06	Abs. (Joint)
	XYZ						0.00	-450.06	
6	Joint						0.00	-539.17	Abs. (Joint)
	XYZ						0.00	-539.17	
11	Joint								
	XYZ	0.00	0.00	40.00	0.00	0.00			Rel. 1 (XYZ)
12	Joint								
	XYZ	0.00	0.00	35.00	0.00	0.00			Rel. 2 (XYZ)
22	Joint								
	XYZ	0.00	0.00	35.00	0.00	0.00			Rel. 12 (XYZ)
32	Joint								
	XYZ	0.00	0.00	35.00	0.00	0.00			Rel. 22 (XYZ)
42	Joint								
	XYZ	0.00	0.00	35.00	0.00	0.00			Rel. 32 (XYZ)

#	Coor.	Axis 1 X (mm)	Axis 2 Y (mm)	Axis 3 Z (mm)	Axis 4 Pitch (deg)	Axis 5 Roll (deg)	Axis 7 (mm/deg)	Axis 8 (mm/deg)	Type

Figure 9-2

Your positions may differ slightly from the figure above. Note the following:

- Positions #1 and #2 were recorded as absolute positions. Each position is stored as the reading of the five independent encoders (axis 1 through 5).
- Positions #3, #4, #5 and #6 are peripheral positions (rotary table). Note that positions 4, 5, and 6 are stored as an encoder reading for axis 8.
- Positions #11, #12, #22, #32 and #42 are relative positions and are listed as such together with their reference positions.

5 Right click on the mouse and click on Show XYZ.

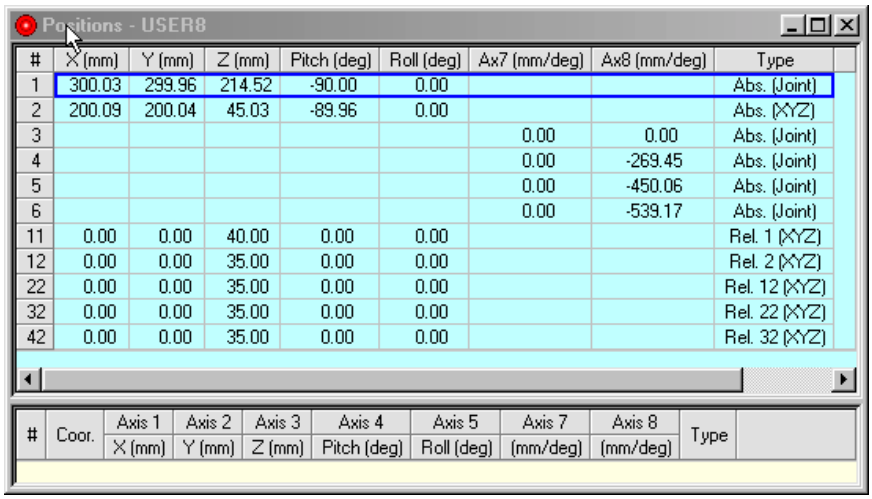


Figure 9-3

- Positions #1 and #2 are now represented only by their XYZ values (with pitch and roll)
- Positions #3, #4, #5 and #6 (the rotary table positions) are still listed as encoder values.
- Positions #11, #12, #22, #32 and #42 are still listed as relative positions and their relative dimension is now listed in the appropriate column (in this case Z).

You can also change the Positions screen to display the Joint positions only by right clicking and selecting Show Joint.

After reviewing the different display options, deselect List Positions.

1 Reset the robotic cell.

2 Run a single cycle of the program.

The robot is executing the task you programmed in Activity 8. Following the completion of its task, zoom in on the tower of cylinders.

Q *Are the cylinders aligned in their stacked tower?*

3 Reset the robotic cell.

4 Click on the first line in the program.

5 Click repeatedly on the Run Single Line icon (F6) until the robot is sent to position #42, just above the base (pink) cylinder (stop after completing line #14).

6 Zoom in and observe how the robot is gripping the cylinder.

7 Continue running the program line-by-line, comparing the relationship between the robot gripper and each of the cylinders.

You probably observed that the stacked cylinders were not perfectly aligned. This is because the rotary table did not rotate precisely a quarter before stopping.

Improving the accuracy of stopping the table will be done using the encoder. The rotary table encoder produces 41,000 pulses for each full rotation. When the green cylinder is in its loading place, the rotary table encoder counts (axis 8) will equal zero.

The red cylinder is located a quarter of a turn away. Therefore, when the encoder value is $\frac{41{,}000}{4} = 10{,}250$ pulses, the red cylinder will be located exactly in the position where the green cylinder was initially located.

When the encoder value is $\frac{41{,}000}{2} = 20{,}500$ pulses, the yellow cylinder will be located exactly in the initial position of the green cylinder, and so forth.

Q *What should the encoder value for the blue cylinder be to be located exactly in the initial position of the green cylinder?*

Task 9-2: Modifying the Positions and Program

1 From the RoboCell menu, do the following:

- Select **Window** | **Simulation & Teach**.
- Select **View | Dialog Bars | Encoders**.

  The Encoder Counts dialog box is displayed along the bottom of the screen, as shown in Figure 9-4.

Figure 9-4

2 Observe the Manual Movement dialog box and note from the following chart how clicking the buttons (or pressing the keys on the keyboard) in Joint Mode (the default) controls the various axes.

Keys	Joint Motion
1 / Q	Control axis 1.
3 / E	Control axis 3.
4 / R	Control axis 4.
5 / T	Control axis 5.
6/ Y	Control axis 6.
8 / I	Control axis 8 (the rotary table).

To increase stacking accuracy, you need to overwrite the rotary table positions. Remember that the rotary table is axis 8.

3 To overwrite the previous position #4, do the following:

- In the Teach Positions dialog box, go to position #4.

  Remember that position #4 was for peripheral equipment and must be so marked.

- For precise stacking, the position #4 encoder counts (the red cylinder) for axis 8 (the rotary table) should be -10,250.
- If they are not, do the following:
  - In the Manual Movement dialog box, select the slowest speed (1) and then click the 8 and I buttons to move the table.
  - When the axis 8 encoder value equals **-10,250** (with a tolerance of error of ±5 counts) in the Encoder Counts dialog box, click record (this will overwrite the previous values for position #4).
- Note that because the number of encoder counts is relatively high, an error of ±5 pulses is acceptable.

4 Record position #5 when the encoder value for axis 8 is **-30,750** (with a tolerance of error of ±5 counts).

5 Record position #6 when the encoder value for axis 8 is **-20,500** (with a tolerance of error of ±5 counts).

6 Save the program and positions as file **USER9**.

7 Run the program and zoom in on the growing tower to observe the improved stacking accuracy.

Task 9-3: Modifying Positions and Program -- Part 2

The program is currently written such that the robot is ordered to move above the table and then the table is rotated. You will now program the robot and the table to perform these tasks simultaneously, thus saving time and increasing efficiency.

For example, in the current program the robot is sent to position #11 and then the rotary table is sent to position #3. However, after sending both the robot and the rotary table to their positions, you can record a new position (#13). Sending the robot system to position #13 will now cause the robot and the table to move together to the position to increase efficiency.

1 Using the Go Position option, send the robot to position #11 and the table to position #3.

2 To record position #13 as both a robot and peripheral position:

- Select **Robot** and **Peripheral**.
- Select **Absolute**.
- Click Record.

3 Resetting the robotic cell first, record an absolute position #14 for the robot and peripheral when the robot is in position #11 and the table is in position #4.

4 Repeat step 3 above to record an absolute position #15 for the robot and peripheral when the robot is in position #11 and the table is in position #5.

5 Repeat again to record an absolute position #16 for the robot and the peripheral when the robot is in position #11 and the table is in position #6.

6 Reprogram the robot by replacing each two lines that first send the robot and then the table to their positions with a single command that will send both to the newly recorded positions (#13, #14, #15 and #16).

Don't forget to change the Remark commands in the beginning of the program.

7 Compare the program you just wrote with the following:

```
 1    Remark: ******************************
 2    Remark: ACT9
 3    Remark: Recording Positions Using Encoder Values
 4    Remark: ******************************
 5    Open Gripper
 6    Remark: STARTING OF RED CYLINDER
 7    Go to Position 14 fast
 8    Remark: robot above rotary table
 9    Remark: rotary table in position
10    Go to Position 1 speed 5
```

```
11    Close Gripper
12    Go to Position 11 fast
13    Go to Position 42 fast
14    Go to Position 2 speed 5
15    Remark: cylinder in place
16    Open Gripper
17    Go to Position 42 fast
18    Remark: STARTING OF BLUE CYLINDER
19    Go to Position 15 fast
20    Remark: robot above rotary table
21    Remark: rotary table in position
22    Go to Position 1 speed 5
23    Close Gripper
24    Go to Position 11 fast
25    Go to Position 42 fast
26    Go to Position 12 speed 5
27    Remark: cylinder in place
28    Open Gripper
29    Go to Position 42 fast
30    Remark: STARTING OF YELLOW CYLINDER
31    Go to Position 16 fast
32    Remark: robot above rotary table
33    Remark: rotary table in position
34    Go to Position 1 speed 5
35    Close Gripper
36    Go to Position 11 fast
37    Go to Position 42 fast
38    Go to Position 22 speed 5
39    Remark: cylinder in place
40    Open Gripper
41    Go to Position 42 fast
42    Remark: STARTING OF GREEN CYLINDER
43    Go to Position 13 fast
44    Remark: robot above rotary table
45    Remark: rotary table in position
46    Go to Position 1 speed 5
47    Close Gripper
48    Go to Position 11 fast
49    Go to Position 42 fast
50    Go to Position 32 speed 5
51    Remark: cylinder in place
52    Open Gripper
53    Go to Position 42 fast
```

Task 9-4: Running the Program

1 Save the program and positions.

2 Reset the robotic cell.

3 Click on the first line in the program.

4 Run a single cycle of the program.

Q *Compare the efficiency of the new program with the previous one.*

Task 9-5: Team Discussion and Review

Q *The following figure shows the encoder values before you performed Task 9-3. What will the encoder values be of the newly recorded positions (#13, #14, #15 and #16)?*

#	Coor.	Axis 1 X (mm)	Axis 2 Y (mm)	Axis 3 Z (mm)	Axis 4 Pitch (deg)	Axis 5 Roll (deg)	Axis 7 mm/deg	Axis 8 mm/deg	Type
1	Joint	44.99	-23.96	44.97	68.99	0.00			Abs. (Joint)
	XYZ	300.03	299.96	214.52	-90.00	0.00			
2	Joint	44.99	-14.57	90.71	13.82	0.00			
	XYZ	200.09	200.04	45.03	-89.96	0.00			Abs. (XYZ)
3	Joint						0.00	0.00	Abs. (Joint)
	XYZ						0.00	0.00	
4	Joint						0.00	90.29	Abs. (Joint)
	XYZ						0.00	90.29	
5	Joint						0.00	270.99	Abs. (Joint)
	XYZ						0.00	270.99	
6	Joint						0.00	180.63	Abs. (Joint)
	XYZ						0.00	180.63	
11	Joint								
	XYZ	0.00	0.00	40.00	0.00	0.00			Rel. 1 (XYZ)
12	Joint								
	XYZ	0.00	0.00	35.00	0.00	0.00			Rel. 2 (XYZ)
22	Joint								
	XYZ	0.00	0.00	35.00	0.00	0.00			Rel. 12 (XYZ)
32	Joint								
	XYZ	0.00	0.00	35.00	0.00	0.00			Rel. 22 (XYZ)
42	Joint								
	XYZ	0.00	0.00	35.00	0.00	0.00			Rel. 32 (XYZ)

Figure 9-5

Task 9-6: Shut Down

1 Exit RoboCell.

2 Turn off the computer.

Education and Employment Opportunities

Benefits of Entering Robotic Contests

Robot contests are fun to watch and fun to enter. Contests are also an excellent way to build one's robot design and construction skills.

Anne Wright, speaking at the Fourth Annual Firefighting Home Robot Contest, listed four benefits of contests:

- **Focus On Integration:** Contests force one to use a multi-disciplinary approach to solve a particular problem. Many of us have weak areas that we tend to generally ignore. However, competition goads us to also work at whatever it takes to make our robot perform well.
- **Compare Approaches:** If nothing else, contests are great eye openers in the myriad ways that a problem can be solved. This gives us ideas and techniques that we can use to improve future robots.
- **Real Hardware vs. Simulation:** Forced reality - time to move from hyperbole to metal bending, from inspiration to implementation. The emphasis on hardware vs. simulation has been a particular hallmark of graduates from the MIT mobile robot lab.
- **Assessment Of Current Limits:** When people from different parts of the country, or countries, and different research establishments gather to present their best shot, one gets a good idea of the current state of the art, and where it needs to be pushed.

http://www.robotmag.com/misc/why.html

Student Robotics Automation Contest

The Student Robotics Automation Contest will be held in April 1998 at Ohio Northern University. Contests are divided into four divisions for Middle School, High School, Community College/Technical Institute, and

University Undergraduate/Graduate. An Autonomous and Radio-control class will be held at the contest.

Contest events include

- Robot Construction Contest
- Robotics Automation Work Cell Contest
- Robotics Automation Work Cell Simulation Contest
- Pick & Place Programming (teach pendant)
- Pick & Place Programming (middle school)

- Pick & Place Programming (high school)
- Pick & Place Programming - Block Scramble
- Robot Maze Contest (open)
- Robot Maze Contest (journey)
- Robot Problem Solving (middle school)
- Robot Problem Solving (high school, community college)
- Robot SUMO Contest
- Videotape contest

Find out about the many robot contests and competitions around the world through the listing at: http://www.ncc.com/misc/rcfaq.html .

Activity 10

Programming the Robot to Execute Linear Movements

OBJECTIVES

In this activity you will accomplish the following:

- Program the robot to move along a straight line.
- Program the robot to simulate a welding operation.

SKILLS

In this activity you will develop the following skills:

- Academic & Employability:
  - Operate lab equipment according to safety regulations.
  - Document inventory and safety procedures for lab set-up and shutdown.
  - Identify career opportunities in robotics.
- Occupational & Technical:
  - Program the robot to move along a straight line.
  - Use relative positions to define a line and run a continuous point simulation using the Go Position command.
  - Record two end-point positions and run a point-to-point simulation using the Go Linear command.
  - Monitor and analyze the process for quality control.
  - Simulate a robotic welding application.
  - Modify the design of a program to improve accuracy and efficiency.

MATERIALS

In this activity you will need the following materials:

- Computer with RoboCell software
- Diskette or personal subdirectory on computer hard drive
- Worksheets for Activity 10

OVERVIEW

Controlling the Robot Trajectory (Linear)

In previous activities, you recorded positions that were translated into encoder values and stored in a position table. When the robot was sent to a specific position, the controller activated (independently) the five motors driving the manipulator. Each motor set in motion its particular axis until the encoder value matched the one in the table. As a result, after all motors had stopped, the TCP was in the new, designated position.

Previously, when you wanted the robot to pick an object from the table, you had to make sure it would not touch any other object or the table on its way. You therefore first defined a position above the object and then sent the robot to this position before sending it to the object.

Since these two positions were very close, the robot moved in a relatively linear path between the two defined positions.

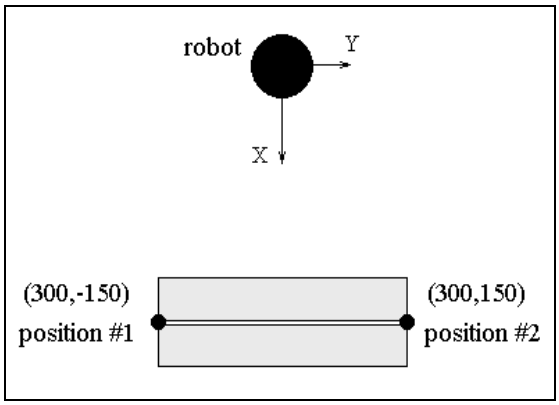


Figure 10-1

While the virtual robot always has a gripper attached to its end effector, you will pretend in this activity that the robot has a welding tool attached. The robot (whose base is shown in Figure 10-1) needs to move the TCP along the edge of the two adjacent blocks to simulate a welding operation. The two end positions, #1 and #2 (whose distance from the origin is equal), were recorded and their encoder values are shown in Figure 10-2.

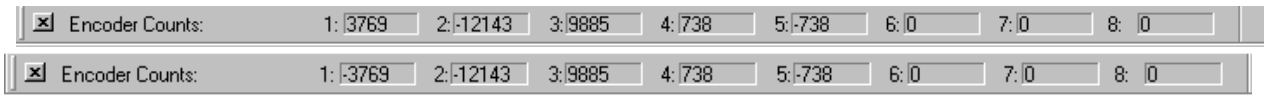


Figure 10-2

Note that in position #1 and position #2, encoders 2, 3, 4, and 5 have the same values. The only difference between the two positions is in encoder 1. When the robot is ordered to move from position #1 to #2, only axis 1 moves and drives the manipulator until the encoder value will equal

the value of position #2. Since axis 1 is a rotary axis (turning the robot base), the robot TCP will move from position #1 to position #2 in a circular path (as shown in Figure 10-3), and not along the straight line.

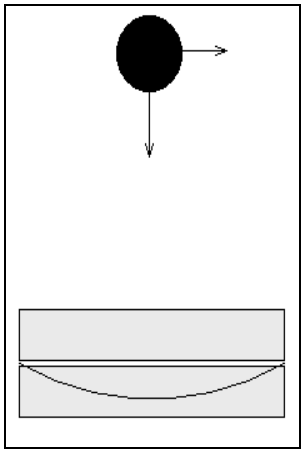


Figure 10-3

A simple way to ensure that the TCP path will be nearly linear is to record another position (#3) along the path, as shown in Figure 10-4. You can then send the robot from position #1 to #2 via position #3, guaranteeing an almost linear path. The more intermediate positions recorded, the more linear the robot trajectory.

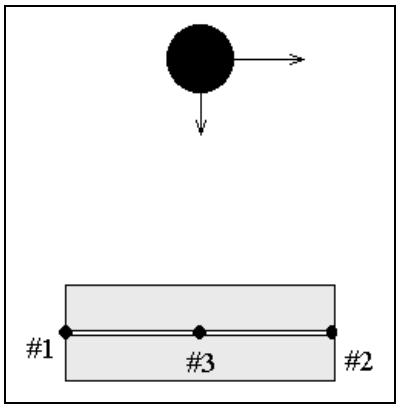


Figure 10-4

An easy way to record intermediate positions would be to use the Relative To command, by continually recording positions *relative to* the current position. The great disadvantage of this method, however, is that the robot will start and stop a number of times between the two end positions, making the process long and tedious.

Linear Movement

The Teach Positions (Simple) dialog box and the Command List offer two options for instructing the robot to move to a position:

- **Go Linear to Position:** This command sends the robot to a recorded position in a straight line. This is termed point-to-point (PTP) movement.
- **Go to Position:** This command sends the robot to a recorded position along a path calculated by the controller, not necessarily a straight line, and usually a curved path. This is termed continuous path (CP) movement.

The Teach Positions (Expanded) dialog box and the Command List also offer a third movement Option, **Go Circular**, which will be discussed later in this book.

To successfully weld, the TCP must move along a straight line at a constant speed. You will compare the following four methods for controlling the TCP trajectory:

- Recording two end positions and then using the Go Position command.
- Recording two end positions and then recording an intermediate one.
- Recording an end position and then sending the robot to this position five times using the Relative To Current command.
- Using the SCORBASE command, **Go Linear**, specially designed for this task.

PROCEDURES

Task 10-1: Running RoboCell and Opening the 3D model File

1. Turn on the computer and run RoboCell.
2. Open the 3D model file **ACT10.3DC**.
3. Redirect the camera to the center of the two blocks.
4. Use the viewing tools to clearly view the blocks and the robot gripper from a comfortable viewing angle.

Task 10-2: Recording Two End Positions and Running the Program

1 Record position #1 and #2, according to Figure 10-1. For both positions, set the Z coordinate to 60 mm and the pitch to -90°.

2 From the RoboCell menu, select Window | Teach & Edit.

3 Write a program that will order the robot to move from position #1 to position #2 fast.

4 Compare with the following program:

```
1       Remark: ****************************
2       Remark: ACT10
3       Remark: Programming the Robot from Point 1 to Point 2
4       Remark: ****************************
5       Go to Position 1 fast
6       Go to Position 2 fast
```

5 Save the program and positions in file **USER10**.

6 Click on the first line of the program.

7 From the RoboCell menu, select **View | Simulation & Teach**.

8 Reset the robotic cell.

9 From the RoboCell menu, select **3D Image | Show Robot Path**.

10 Run a single cycle of the program.

By selecting the Show Robot Path option, the path of the TCP during the cycle has been drawn on the screen. The TCP path is shown drawn at a constant time interval. The distance between the cubes is proportional to the TCP speed.

Q *Describe the TCP path.*

You may wish to use the Top View feature to best observe the TCP path.

Q *Describe the TCP speed during program execution.*

11 Deselect **3D Image | Show Robot Path**.

The TCP path is cleared.

Task 10-3: Recording a Middle Position and Running the Program

1 Select **Window | Simulation & Teach**.

2 In the Teach Positions dialog box, teach position #3 as relative to position #1 with a Y-offset of +150 mm.

Position #3 is located at the center of the blocks.

3 Modify the program so that the robot will pass through position #3 on its way from position #1 to position #2.

4 Compare the program you just wrote with the following:

```
1     Remark: ****************************
2     Remark: ACT10
3     Remark: Programming the Robot from Point 1 to Point 2
4     Remark: ****************************
5     Go to Position 1 fast
6     Go to Position 3 fast
7     Go to Position 2 fast
```

5 Save the program and positions file (to overwrite what you previously saved).

6 Reset the robotic cell.

7 Run a single cycle of the program.

Q *Describe the TCP path.*

You may wish to use the Top View feature to best observe the TCP path.

Q *Is the TCP path in this program closer to a straight line than in the previous program?*

Q *Describe the TCP speed during program execution.*

8 Clear the robot path.

Task 10-4: Recording a Relative Position, Sending the Robot to this Position Repeatedly and Running the Program

1 Record position #4 as relative to the current TCP position with a Y-offset of +50 mm.

2 Modify the program so that the robot is sent to position #4 five times fast on its way from position #1 to position #2.

Note that you can use the Cut & Paste tools to aid in repetitious programming.

3 Compare the program you just wrote with the following:

```
1     Remark: ****************************
2     Remark: ACT10
3     Remark: Programming the Robot to Execute Linear Movements
4     Remark: ****************************
5     Go to Position 1 fast
6     Go to Position 4 fast
7     Go to Position 4 fast
8     Go to Position 4 fast
9     Go to Position 4 fast
10    Go to Position 4 fast
11    Go to Position 2 fast
```

4 Save the file to overwrite the program you previously saved.

5 If the Graphic Display window disappeared, select Window | Simulation & Teach from the RoboCell menu.

6 Click on the first line of the program.

7 Reset the robotic cell.

8 Run a single cycle of the program.

Q *Describe the TCP path.*

You may wish to use the Top View feature to best observe the TCP path.

Q *Is the TCP path closer to a straight line this time?*

Q *Describe the TCP speed during program execution.*

9 Clear the robot path.

Task 10-5: Using the Go Linear Command and Running the Program

You will now use the SCORBASE command **Go Linear** that will move the TCP in a straight line from position #1 to position #2. To execute this command, the controller calculates the coordinates of many positions along the line, and then sends the robot to these intermediate positions without stopping in the middle.

1 Leave the first line of the program as is.

2 Cut the remaining lines of the program. You will replace them with the more direct, Go Linear, command.

3 Double click on **GL** (Go Linear to Position) from the Command List.

The Go to Position dialog box opens. In the Target Position field, enter/select **2** and select **Fast**.

Note that **Linear** is selected in the right-hand side of the dialog box.

4 Compare your modified program with the following:

```
1       Remark: ****************************
2       Remark: ACT10
3       Remark: Programming the Robot to Execute Linear Movements
4       Remark: ****************************
5       Go to Position 1 fast
6       Go Linear to position 2 fast
```

5 Save (overwrite) the program and positions.

6 Click on the first line of the program.

7 Reset the robotic cell.

8 Run a single cycle of the program.

Q *Describe the TCP path.*

You may wish to use the Top View feature to best observe the TCP path.

Q *Describe the TCP speed during program execution.*

9 Clear the robot path.

Task 10-6: Team Discussion and Review

Q *Add to your program an instruction to go to position #1. Then run the program with the Encoder Counts dialog box open. Note that when the TCP moves from position #2 to position #1, only the encoder values for encoder 1 changed. When the TCP moves from position #2 to position #1, however, all the encoder values changed.*

Q *How can you explain this?*

Task 10-7: Shut Down

1 Exit RoboCell.

2 Then turn off the computer.

ACADEMICS

Education and Employment Opportunities

Emergence of Skilled High-Tech Workers

Major technological changes are being introduced to improve productivity and quality in the manufacturing industries. As a result of these new technologies, today's production workers need a higher level of skills. Employers need workers with a knowledge of robotics, computers, Computer Numerically Controlled (CNC) machines and Computer-Aided Design (CAD).

The U.S. Department of Labor: "In recent years, employers have reported difficulties in attracting workers to machining and tool programming occupations. Therefore, good employment opportunities should exist for candidates with the necessary mechanical and mathematical aptitudes. Employment of computer numeric control (CNC) machine operators is expected to increase in the future despite the decline in machine operators as a whole."

The personal qualifications for skilled high tech workers include:

- Ability to follow written instructions
- Computer skills
- Interpersonal skills
- Manual dexterity
- Mechanical ability
- Metalwork skills
- Mathematical skills
- Spatial skills

Robot Machine Operators

Robot Machine Operators operate robots in specialized settings. They usually work in a manufacturing plant and set up and operate industrial robots to drill, grip, rivet, spraycoat and perform material handling tasks. Operators who work with the simplest class of robots, called pick-and-place robots, may be responsible for mechanically programming the robot. Operators working with continuous path robots (mostly used for spray painting and other finishing operations) may actually teach the robot by leading the robot arm through motions. Other robots may be programmed using a teach pendant which allows a machine operator to "teach" and then record motions. Machine operators may also change the programming or circuitry and perform routine maintenance.

Job related skills required for robot machine operators include knowledge of robotics, robotic systems and teach pendants. Required education and training include high school, shop training and community college classes. Wage data was not available regarding the annual salary range.

As manufacturers continue to turn to automation and other new technologies they will need workers with high-tech skills. Training will continue to be an important means of assisting workers adjust to these new skill demands. Training programs are offered by vendors, industry trade associations, and industry unions. Training is also available through private and public vocational training agencies and community colleges.

Most machinists work in small machining shops or in manufacturing firms that produce durable goods such as metalworking equipment, industrial machinery, aircraft, motor vehicles, or farm equipment. CNC operators work mainly in fabricated metal products, industrial machinery and equipment, transportation equipment, and primary metals.

The number of openings for machinists and tool programmers is expected to decline slightly, but the U.S. Department of Labor states that employers have reported difficulty in recruiting workers for machining and tool programming occupations and predicts that good opportunities should exist for job seekers with the necessary high-tech skills.

Sources for additional information:

Association for Manufacturing Technology
7901 Westpark Drive
McLean, VA 22102

The Tooling and Manufacturing Association
1177 South Dee Road
Park Ridge, IL 60068

Source: State of California, Employment Development Department, Labor Market Information Division, Information Services Group, (916) 262-2162.

Activity 11

Programming the Robot to Execute Circular Movements

OBJECTIVES

In this activity you will accomplish the following:

- Program the robot to move in an arched path.
- Program the robot to draw a complex figure.

SKILLS

In this activity you will develop the following skills:

- Academic & Employability:
  - Operate lab equipment according to safety regulations.
  - Document inventory and safety procedures for lab set-up and shutdown.
  - Identify industrial application of simulation.
- Occupational & Technical:
  - Program the robot to execute circular motions.
  - Write a robotics program using the Go Circular command and an intermediate point to describe each arc.
  - Monitor and analyze the process for quality control.
  - Modify the design of a program to improve the design process
  - Implement modifications to meet changes in design criteria.

MATERIALS

In this activity you will need the following materials:

- Computer with RoboCell software
- Diskette or personal subdirectory on computer hard drive
- Worksheets for Activity 11

OVERVIEW

Controlling the Robot Trajectory (Go Circular)

You have now learned that positions are recorded using encoder values. When the robot is sent to a position, the motors driving the manipulator axes move independently until the encoder reading matches the recorded value. This control method is known as **Point to Point (PTP)**.

PTP control is very simple but its greatest weakness is that it does not control the TCP path. In order to control the TCP path, intermediate positions should be located and recorded along the trajectory. Then the robot can be ordered to move through these positions on its way to the target position. Increasing the number of intermediate positions will bring the trajectory closer to the operator's demands.

As you learned in the previous activity, however, recording intermediate positions creates an additional problem. The TCP then stops at every intermediate position causing a fragmentary motion not suitable for applications such as painting and welding.

Controlling the TCP path can best be achieved using special SCORBASE commands. In Activity 10, you used the first of these commands, Go Linear, to move the TCP in a straight line. The controller located various intermediate positions, and the TCP was sent to these positions without stopping.

In this activity, you will learn how to move the TCP along an arched path, which too can be problematic. Figure 11-1 points out that defining an arched path between position #1 and position #2 must be carefully done, as an endless number of arcs connecting the two positions exist.

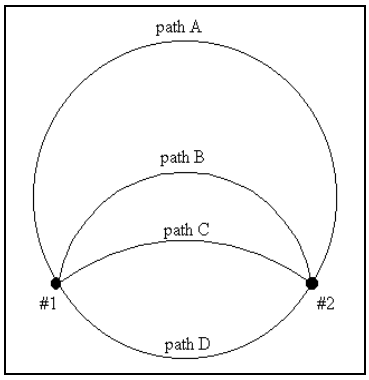


Figure 11-1

To designate a specific arc, a third position along the arc (aside from the endpoints) needs to be recorded. The robot can then be programmed to **Go Circular** to the destination via that intermediate position. Note that the TCP will not stop in the intermediate position.

Using the Go Circular and Go Linear Commands to Draw "B"

In this activity, you will simulate a robot responsible for marking identification codes on products at the end of production. In order to "write," a tool similar to an ink-jet is connected to the robot manipulator. The robot moves the jet, drawing the codes on the product.

As shown in Figure 11-2, you will program the robot to "write" the letter B.

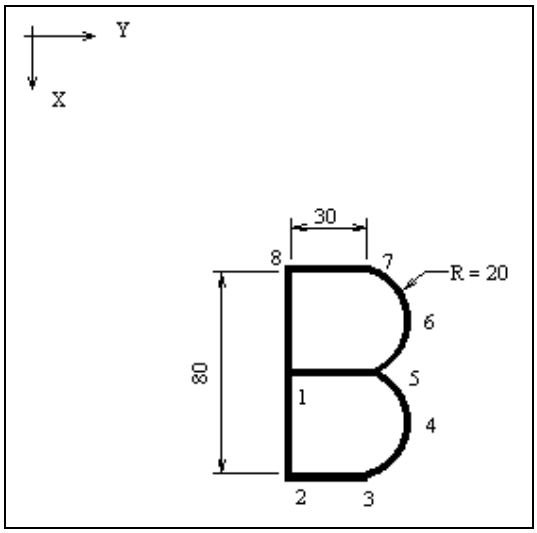


Figure 11-2

The letter B is composed of four lines and two arcs. As shown in the figure, you must record eight positions in order to program the robot to draw the letter.

In your program, position #1's coordinates will be (200,200). You will record the remaining seven positions as relative to position #1 such that changing its coordinates will cause the robot to draw the letter B.

PROCEDURES

Task 11-1: Running RoboCell and Opening the 3D model File

1. Turn on the computer and run RoboCell.
2. Open the 3D model file **ACT11.3DC**.
3. Use the viewing tools to clearly observe the table from a comfortable viewing angle.

Task 11-2: Recording Positions

1. Record position #9 as the robot's initial position.

   When the job is done the robot will be ordered to return to this position.

2. Teach position #1 as Absolute with the following coordinates, roll and pitch:

   - Coordinates: (200,200,20)
   - Pitch = -90°
   - Roll = 0°

3. Record the other seven positions as relative to position #1, using the coordinates shown in the following table.

   Remember that with each new position, you must click Relative To in order to reset the XYZ coordinates.

Position #	X	Y
2	-40	0
3	-40	30
4	-20	50
5	0	30
6	20	50
7	40	30
8	40	0

4. Minimize the Teach Positions dialog box.

Task 11-3: Programming

1. Program the robot to do the following:

   Remember to use the **Go Circular** command from the Command List. As mentioned earlier, in addition to the target position, an intermediate position along the arc must also be used.

   - Send the robot to position #1.
   - From position #1, send it to position #2 in a line.
   - From position #2, send it to position #3 in a line.
   - From position #3, send it in an arc to position #5 via position #4.
   - From position #5, send it in an arc to position #7 via position #6.
   - From position #7, send it to position #8 in a line.

- From position #8, send it to position #1 in a line.
- From position #1, send it to position #5 in a line.
- From position #5, send it to its initial position.

2 Compare what you just wrote with the following program:

```
1     Remark: ****************************
2     Remark: ACT11
3     Remark: Programming the Robot to Execute Circular Movements
4     Remark: ****************************
5     Go to Position 1 fast
6     Go Linear to position 2 speed 5
7     Go Linear to position 3 speed 5
8     Go Circular to position 5 through 4 speed 5
9     Go Circular to position 7 through 6 speed 5
10    Go Linear to position 8 speed 5
11    Go Linear to position 1 speed 5
12    Go Linear to position 5 speed 5
13    Go to Position 9 fast
```

Task 11-4: Running the Program

1 Save the program and positions file as **USER11**.

2 Reset the robotic cell.

3 Show the robot path.

This action is similar to the activation of an ink jet.

4 Run the first 5 lines of the program.

The robot now moves to position #1.

5 Repeatedly click the Run Single Line icon until the robot reaches position #5 for the second time (line #12).

Note that the currently executed command is highlighted in the Program window.

6 Using the viewing tools, observe what the robot has "written."

7 Disable the Show Robot Path option. (Reselect the option to remove the checkmark. Do not clear the robot path.)

This action is similar to the de-activation of an ink jet.

8 Run the last line in the program to return the robot to its initial position (position #9).

Q *Rerun the program line-by-line to determine if the letter B is completed after moving from position #1 to position #8 through all the intermediate positions (up to line #10 of the program)?*

Task 11-5: Team Discussion and Review

Q *Using all the points recorded in this activity, write a program that draws the number "3".*

Save the program as ***USER11A****.*

Task 11-6: Shut Down

1 Exit RoboCell.

2 Then turn off the computer.

ACADEMICS

Industrial Applications

Robot Simulation and FMS: Non-Rigid Objects

The following information was excerpted from the homepage of the Kaemart Group:

In order to improve the cost effectiveness of Flexible Manufacturing System, powerful simulation tools are needed. Off-line programming of robots has proved to be effective in reducing time and cost efforts in manufacturing equipment reprogramming. The use of simulation packages to study and evaluate automated handling systems is becoming more and more important. The objective of this research is to provide the mechanical engineer with a set of tools to design, predict and evaluate via graphic simulation possible configurations also for non-rigid material handling system.

At present, most simulation systems are capable of dealing only with rigid objects. Therefore they are not able to model and simulate the behavior of non-rigid materials during handling operations, e.g. pick and place of a piece of cloth. Our idea is to extend current tools to obtain a software environment to model and simulate systems, products, and all operations involving non-rigid material. This system allows to assembly the working environment and to plan robot task, with emphasis on the analysis of interaction among kinematic devices and objects present in the environment.

With a graphical interactive interface users can verify if the target points are reachable, if the path passes through control points and if there are possible interferences among the objects present in the simulation environment.

http://ied.eng.unipr.it/KAEMART/AREAS/OFF_LINE_PROGRAMMING/main.html

Activity 12

Programming with Subroutines

OBJECTIVES

In this activity you will accomplish the following:

- Describe what is a system input and output.
- Use conditional branching in a program.
- Write a subroutine.
- Build a system for sorting objects.

SKILLS

In this activity you will develop the following skills:

- Academic & Employability:
  - Operate lab equipment according to safety regulations.
  - Document inventory and safety procedures for lab set-up and shutdown.
  - Identify industrial application of simulation and off-line programming.
- Occupational & Technical:
  - Simulate a robot system incorporated a robot, peripheral device, and sensor input device.
  - Design a program that will call a subroutine when a specific condition is met.
  - Monitor and analyze the operation of the system for quality control.
  - Modify the design of the p4ogram to improve the design process.
  - Implement modifications to meet changes in design criteria.

MATERIALS

In this activity you will need the following materials:

- Computer with RoboCell software
- Diskette or personal subdirectory on computer hard drive
- Worksheets for Activity 12

Inputs and Outputs

The terms input and output are used to describe the relationship between a system and the devices connected to it. **Input** devices send data *to* the system; while **Output** devices receive control signals *from* the system.

A robot system, like other systems, contains input and output devices. The commands sent to the robot from the keyboard are one of the system inputs, while the motors moving the manipulator represent output.

The SCORBOT-ER 4u controller also gets information from external input devices and then energizes output devices. The relationship between these input and output devices is defined by the specific program being executed.

In many control systems, sensors are used as input devices. The sensors are usually designed to sense the system status or an object's physical dimension. Sensors can be divided into two groups: Analog and Discrete. In this tekLINK, you will work only with discrete sensors.

Discrete sensors are used when there is a need to rule out the existence of a certain condition. For example, discrete sensors could do the following:

- A discrete temperature sensor set for 100 degrees can sense whether the temperature is above/below 100 degrees.
- A discrete color sensor set for detecting red can sense whether the examined object is red (or not).

Conditional Branching

In this activity, you will simulate a robot system (Figure 12-1) that contains a robot, rotary table, chemical treatment tank and a sensor that can detect blue objects (connected to input #1). Placed on the rotary table are red and blue cylinders in random order. The virtual robot's task is to pick *only* the blue cylinders from the table, place them in the tank for two seconds, and then place them back at their original position on the rotary table.

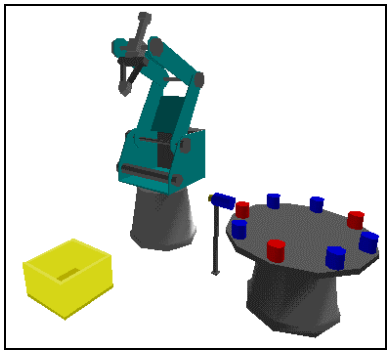


Figure 12-1

The program flow needed to carry out this task differs from the flow of previous programs. This is primarily because the robot is incapable of predicting a cylinder's color and therefore knowing whether or not it should be picked and placed in the tank.

So far you have only worked with programs whose commands were executed once and in a descending order. This type of program flow is only appropriate, however, when all robot actions are easily predicted or known in advance.

In this activity, you will write a program that can "decide," according to the data supplied by the sensor whether to order the robot take a cylinder to the tank or leave it on the rotary table. In such a program, the commands to be executed would differ depending on the color of the cylinder detected. This type of program flow, in which several options are possible, is called **Conditional Branching**.

If Input # On Jump Command

In this activity you will work with a new SCORBASE command used only in conditional branching: **If Input # On Jump** (where # is replaced by an input terminal number). When executed, the controller checks the input terminal status. The status of the sensor connected to the designated input determines whether the program flow should be directed to a subroutine or to a label command.

The If Input #On Jump command is located in the Inputs & Outputs section of the Command List. When selecting this command, the dialog box shown in Figure 12-2 will appear.

Figure 12-2

Subroutines

A subroutine can be used to repeat an action at different places in the program. Instead of writing the same series of commands every time the program requires this action, one subroutine which contains the necessary commands can be written and called (activated) each time it is needed. Subroutines therefore save programming time and storage (file) space.

Figure 12-3 shows a program containing eight commands. When the program is executed, the commands are executed one after the other. However, when the fifth line is executed, the subroutine is called and program execution stops. The subroutine is then executed.

1.	Command	Beginning of program
2.	Command	
3.	Command	
4.	Command	
5.	Call Subroutine	
6.	Command	
7.	Call Subroutine	
8.	Command	End of program
9.	Set Subroutine	Beginning of Subroutine
10.	Command	
11.	Command	
12.	Command	
13.	Return from Subroutine	End of Subroutine

Figure 12-3

After executing the subroutine commands (lines 10, 11, 12), the last command of the subroutine causes program execution to resume (from the sixth line forward). The seventh command will have the same effect as the fifth: program execution will pause and the subroutine commands will be executed again. When the subroutine ends, program execution will resume from the eighth line. Finally, when the program ends at line #9, the subroutine will not be executed again, as it has not been called.

In this activity, you will write a program that includes a subroutine. The program will rotate the table, then stop it when a cylinder faces the sensor and the robot stands above the cylinder. If the sensor determines that the cylinder is blue, the subroutine will be called. The subroutine will order the robot to take the cylinder from the table, place it in the tank, wait two seconds, pick the object from the tank and place it at its original position on the rotary table. If the cylinder is red, the subroutine will not be called. The next cylinder will be examined immediately after examining a red cylinder or, in the case of a blue cylinder, after returning from the subroutine.

Subroutine Commands

A subroutine is created using the following SCORBASE commands:

- **Set Subroutine:** the first command in every subroutine, always followed by the subroutine name.
- **Return Subroutine:** the last command in every subroutine.
- **Call Subroutine:** a command line anywhere within the program which activates the subroutine.
- The **Set Subroutine** and **Return Subroutine** commands are located in the Program Flow section of the Command List. When selecting the Call Subroutine command, the dialog box shown in Figure 12-4 will appear.

Figure 12-4

The program will require the recording of 12 positions, including:

- Eight peripheral device positions in which each of the eight cylinders is facing the sensor (position #3-#10).
- One position at which the robot can pick the cylinder facing the sensor (position #1) -- at this position, were the robot's gripper to close, the cylinder would be in it.
- One position with a Z-offset of 45 mm from position #1 (position #11).
- One absolute position within the tank (position #2).
- A position 100 mm above the tank (position #12).

PROCEDURES

Task 12-1: Running RoboCell and Opening the 3D model File

1. Turn on the computer and run RoboCell.
2. Open the project file **ACT12.3DC**.

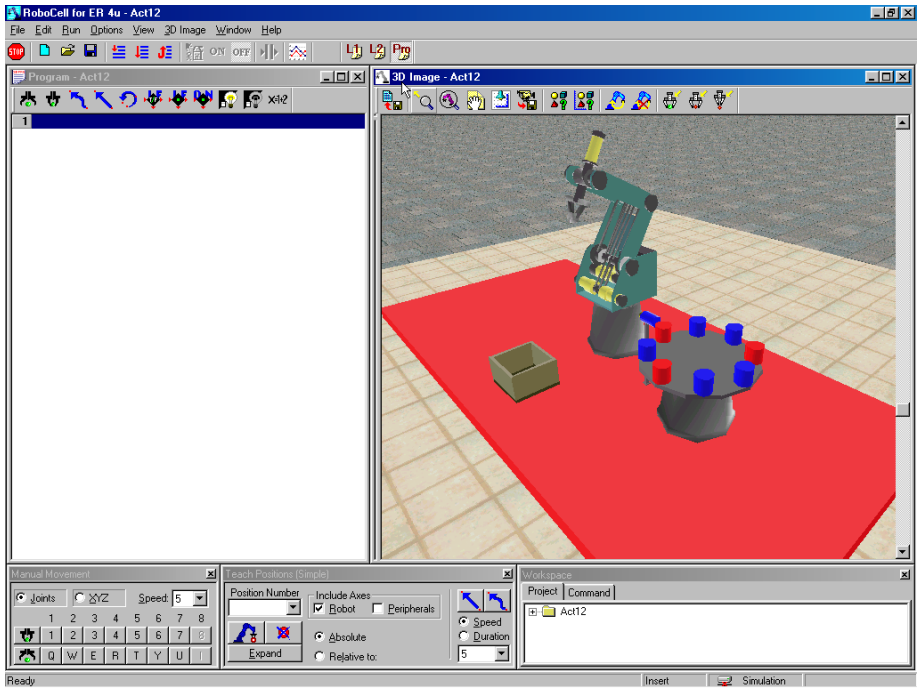


Figure 12-5

3. Use the viewing features to clearly observe the sensor and cylinder facing it.

Task 12-2: Recording Positions

1. Open the robot gripper.
2. To record positions #1 (when the TCP is located at the cylinder facing the sensor) and # 11 (the position 45 mm above position #1) do the following:
   - Send the robot to the red cylinder facing the sensor.
   - Record this as absolute position #1 for the robot.
   - Teach position #11 as relative to position #1 with a Z-offset of +45 mm.
3. Teach position #2 (inside the tank) as the absolute position (x=400, y=0, z=40, p=-90, r=0). This position is located 40 mm above the base of the tank.
4. Teach position #12 as relative to position #2 with a Z-offset of 100 mm.

5. To record position #3 (a peripheral position), do the following:
   - Send the robot to position #11 with its gripper open.
   - Redirect the camera to the red cylinder facing the sensor and zoom in closely.
   - 3D Image | Labels | Object Position.
   - You should see the red cylinder's position label is 100, 350.
   - Record this position as absolute peripheral position #3.
6. To record position #4, do the following:
   - Using the 7/U keys in the Manual Movement dialog box, rotate the table counter-clockwise.
   - Notice that the coordinates of the cylinders change as the table rotates.
   - Remember that you can select a slower speed to increase accuracy.
   - Click 7 until the coordinates of the next cylinder are approximately (100, 350) -- meaning that the next cylinder now faces the sensor.
   - Then record this as absolute peripheral position #4.
7. Record positions #5 to #10 as you did the previous two (cylinder is close to (100, 350)).
8. Select 3D Image | Labels and uncheck Object Positions to hide the labels.
9. Save your program as **USER 12**.

Task 12-3: Programming

1 Program the following program and subroutine.

```
1     Remark: ************************************
2     Remark: ACT12
3     Remark: Programming with Subroutines
4     Remark: ************************************
5     Go to Position 3 fast
6     If Input 1 on call sub. PICK
7     Go to Position 4 fast
8     If Input 1 on call sub. PICK
9     Go to Position 5 fast
10    If Input 1 on call sub. PICK
11    Go to Position 6 fast
12    If Input 1 on call sub. PICK
13    Go to Position 7 fast
14    If Input 1 on call sub. PICK
15    Go to Position 8 fast
16    If Input 1 on call sub. PICK
17    Go to Position 9 fast
18    If Input 1 on call sub. PICK
19    Go to Position 10 fast
20    If Input 1 on call sub. PICK
21    Set Subroutine PICK
22    Open Gripper
23    Go to Position 11 fast
24    Go to Position 1 speed 5
25    Close Gripper
26    Go to Position 11 fast
27    Go to Position 12 fast
28    Go to Position 2 speed 5
29    WaiT 20 ( 10 ths of seconds )
30    Go to Position 2 speed 5
31    Go to Position 12 fast
32    Go to Position 11 fast
33    Go to Position 1 fast
34    Open Gripper
35    Go to Position 11 fast
36    Return from Subroutine
```

Task 12-4: Running the Program

1 Reset the robotic cell.

2 Run a single cycle of the program.

Task 12-5: Team Discussion and Review

Q *Load the 3D model file ACT12_1.3. This robotic cell is similar to the one used in this activity except that the sensor that detects blue was replaced with a sensor that detects red.*

Q *Do the program and/or positions need to be modified in order to now sort the red cylinders? If yes, then save the new program and positions as file USER12A.*

Task 12-6: Shut Down

1 Exit RoboCell.

2 Then turn off the computer.

ACADEMICS

Industrial Applications

Robot Simulation and Off-Line Programming

The following was taken from the home page of Tehdasmallit, a Scandinavian robotics company:

Tehdasmallit uses IGRIP (Interactive Graphics Robot Instruction Program) as a simulation and off-line programming tool. IGRIP is an interactive, 3D graphic simulation tool for designing, evaluating, and off-line programming robotic workcells.

Actual robotic/device geometry, motion attributes, kinematics, clamps, fixtures, and I/O logic are incorporated to produce extremely accurate simulations. The IGRIP provides an interactive 3D graphics based environment, in which to verify production concepts, workcell designs and manufacturing processes before implementing them on the shop floor. After verification is completed, automated factory floor devices, such as robots and turntables, can be programmed off-line based on the CAD data for the part being processed.

Simulation and analysis functions include automatic collision and near-miss detection, and automatic adjustment of a robot work envelope for tool offsets and joint limits.

Off-line programming case can be divided as follows:

- Creating 3D model of the robotic workcell

- Calibrating the simulation model:
- Creating the off-line programs by using IGRIP's Graphical Simulation Language (GSL)
- Verification of the created programs collision and near-miss detection, checking the joint limits and speeds, cycle time analysis
- Downloading the created programs to the robot specific language

As a result of off-line programming:

- The quality of the programs is better compared to programs created on-line.
- Production stops due to the new program creation are minimized.
- Optimized programs due to the powerful analysis tools always correct orientation of the used tool.
- No unexpected collisions.
- Easy program editing and modifying much faster program creation compared to on-line programming.

http://www.tdm.fi/products/rosiolp.html

Activity 13

More Programming with Conditional Branching

OBJECTIVES

In this activity you will accomplish the following:

- Learn about conditional branching in robot programs.
- Use labels in a program.
- Program the robot to simulate a sterilization process using conditional jumps.

SKILLS

In this activity you will develop the following skills:

- Academic & Employability:
  - Operate lab equipment according to safety regulations.
  - Document inventory and safety procedures for lab set-up and shutdown.
  - Identify career opportunities for electronics test technicians.
- Occupational & Technical:
  - Simulate a robot system incorporating a robot, conveyor belt, and sensory input device.
  - Use flow control commands to move to different portions of the program.
  - Program an interrupt that will call a subroutine.
  - Monitor and analyze the operation of the system for quality control.

MATERIALS

In this activity you will need the following materials:

- Computer with RoboCell software
- Worksheets for Activity 13

OVERVIEW

Review of Conditional Branching

In the previous activity, you encountered conditional branching for the first time. The program you wrote ordered the rotary table to place a cylinder in front of a color sensor and then check the sensor response. Based on the sensor response, the program determined whether to call a subroutine.

Timing was crucial to the program's success. The If Input command had to be programmed such that it would be executed only after the cylinder stopped in front of the sensor. Were the command to be executed before the object faced the sensor, the sensor would signal the controller that no blue cylinders are detected. Therefore, the controller would act as if it had detected a red cylinder. The subroutine would not be called and the table would continue to rotate.

Sterilizing Medical Equipment Using the If Input Command

In this activity, you will simulate a robot responsible for sterilizing medical equipment and then packing the equipment (shown in Figure 13-1).

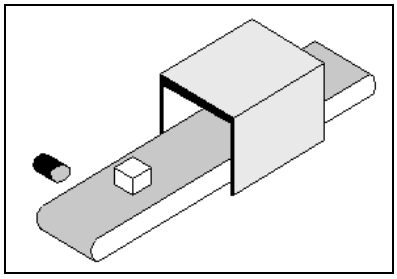


Figure 13-1

First, the robot will place medical equipment needing sterilization onto a conveyor. The conveyor will be started, transporting the equipment into a machine responsible for sterilizing it. After being sterilized, the equipment is moved out by the conveyor from the sterilizing machine. When it reaches a sensor the conveyor is stopped, and the robot takes the sterilized object away.

Writing a program for this task, using the If Input command, may seem easy. However, further examination shows that the synchronization of the program flow with the conveyor speed may cause problems. For example, the equipment could pass the sensor while another command (other than the If Input command) is being executed. The equipment would therefore not be detected by the sensor, and it would therefore continue on its path along the conveyor until it falls off.

On Input Interrupt #_Call Subroutine...

In this activity, you will use a new SCORBASE command called **On Input Interrupt #....** (**OI**) to pause the conveyor and thus solve this likely hazard. This command, located in the Inputs & Outputs section of the Command List, opens the dialog box shown in Figure 13-2.

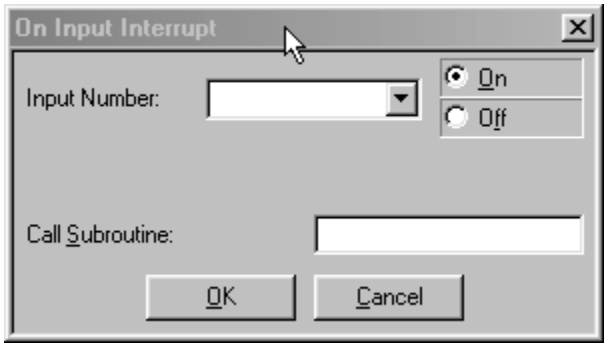


Figure 13-2

It sets the condition for an input interrupt service. The service (Call Subroutine) will be performed whenever the condition (input status) is satisfied, regardless of the current program flow. This means that the system will continuously check the sensor (input terminal) until a signal is received. Once the signal is received, the program will abandon the command currently being executed and immediately execute the command specified for this interrupt instead. The program will resume from the point where it was interrupted as soon as the subroutine completes its execution.

When working with the On Input Interrupt#_On command, you must also add the following pair of commands to your program:

- **Disable Interrupt #_**: This command causes the specified input interrupt to become inactive.
- **Enable Interrupt#_**: When an interrupt is disabled, it is disregarded until this command reactivates it.

These two commands open the dialog box shown in Figure 13-3.

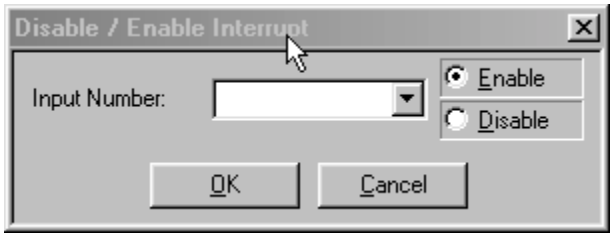


Figure 13-3

In the Input Number field, you should enter an input number, variable or the word ALL.

PROCEDURES

Task 13-1: Running RoboCell, Opening the 3D model File and Recording Positions

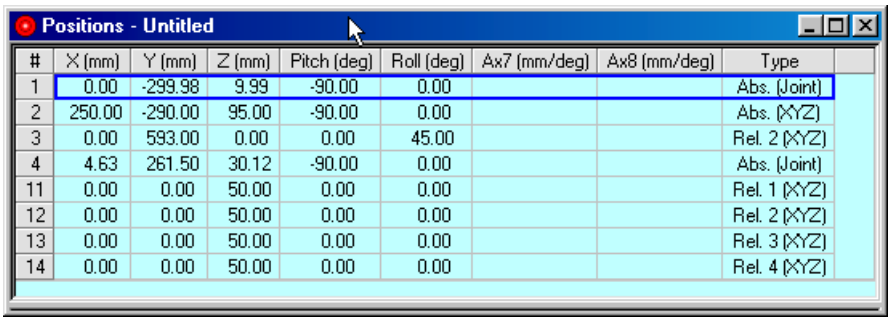
The program will require the recording of 8 positions.

1. Turn on the computer and run RoboCell.
2. Open the 3D model file **ACT13.3DC**.
3. Use the viewing tools to clearly view the cell from a comfortable viewing angle.
4. Open the robot gripper.
5. Using the Send to Object option, record position #1 as near the green cylinder's initial position such that closing the gripper would grip the cylinder in the center of the its jaws.
6. Record position #11 as relative to position #1 with a Z-offset of +50 mm.
7. Teach position #2 as absolute [x = 250; y = -290; z = 95; p = -90; r=0].

   This is the position on the conveyor at which the robot will initially place the cylinder.
8. Teach position #12 as relative to position #2 with a Z-offset of +50 mm.
9. Teach position #3 as relative to position #2 with a Y-offset of +593 mm and a roll of +45°.

   Position #3 is the point where the cylinder reaches the opposite end of the conveyor and faces the sensor.
10. Teach position #13 as relative to position #3 with a Z-offset of +50 mm.
11. Record position #4 as the position for placing the cylinder on the template.

    Hint: The easiest way to record the position would be to grip the cylinder in the robot's grippers and using the Send to Point option, place the cylinder on the template.
12. Teach position #14 as relative to position #4 with a Z-offset of +50 mm.
13. Compare the XYZ positions you just recorded with those shown below.



#	X (mm)	Y (mm)	Z (mm)	Pitch (deg)	Roll (deg)	Ax7 (mm/deg)	Ax8 (mm/deg)	Type
1	0.00	-299.98	9.99	-90.00	0.00			Abs. (Joint)
2	250.00	-290.00	95.00	-90.00	0.00			Abs. (XYZ)
3	0.00	593.00	0.00	0.00	45.00			Rel. 2 (XYZ)
4	4.63	261.50	30.12	-90.00	0.00			Abs. (Joint)
11	0.00	0.00	50.00	0.00	0.00			Rel. 1 (XYZ)
12	0.00	0.00	50.00	0.00	0.00			Rel. 2 (XYZ)
13	0.00	0.00	50.00	0.00	0.00			Rel. 3 (XYZ)
14	0.00	0.00	50.00	0.00	0.00			Rel. 4 (XYZ)

Figure 13-4

Task 13-2: Programming

1 Try to do the programming yourself. The program should consist of the following steps:

- Define an interrupt. Whenever the sensor detects an object, the program execution pointer will call the subroutine STOP.
- Pick the cylinder (from position #1 and place it in position #2 while moving through positions #11 and #12).
- Start the conveyor (in plus direction) and move the TCP to position #13 (the gripper should be above the conveyor, next to the sensor).
- Wait for the interrupt 30 seconds.
- If the interrupt does not happen after 30 seconds, stop the conveyor and end the program.
- In case of an interrupt, disable the interrupt, stop the conveyor, order the robot to pick the object and place it on the template, enable the interrupt and stop the program execution.

The program is as follows:

```
1     Remark: ********************************
2     Remark: ACT13
3     Remark: More Programming with Conditional Branching
4     Remark: ********************************
5     On Input Interrupt 1 on call sub. STOP
6     Open Gripper
7     Go to Position 11 fast
8     Go to Position 1 speed 5
9     Close Gripper
10    Go to Position 11 fast
11    Go to Position 12 fast
12    Go to Position 2 speed 5
13    Open Gripper
14    Go to Position 12 fast
15    Start Conveyor axis 7 at speed 3 in Plus direction
16    Go Linear to position 13 fast
17    WaiT 300 ( 10 ths of seconds )
18    Set Subroutine STOP
```

```
19      Stop conveyor axis 7
20      Disable input Interrupt 1
21      STop conveyor axis 7
22      Go to Position 3 speed 5
23      Close Gripper
24      Go to Position 13 fast
25      Go to Position 14 fast
26      Go to Position 4 speed 5
27      Open Gripper
28      Go to Position 14 fast
29      Enable input Interrupt 1
30      Return from Subroutine
```

Task 13-3: Running the Program

1 Save the file as **USER13**.

2 Reset the robotic cell.

3 Click on the first line of the program.

4 Simulate and run a single cycle of the program.

Task 13-4: Team Discussion and Review

Q *Remove the Disable input Interrupt 1 command from the program and run it again.*

Q *Print the program (or copy it) and then draw lines that describe the program flow.*

Q *Describe the robot response.*

Q *Give reasons for the robot response.*

Q *Describe the task of the Disable input Interrupt 1 command.*

Task 13-5: Shut Down

1 Exit RoboCell.

2 Then turn off the computer.

Education and Employment Opportunities

Electronic Assembly Testing Technician

The following career opportunity was excerpted from the homepage of GEC Alsthom Schilling Robotic Systems (http://www.schilling.com).

This position is responsible for the assembly, testing, and troubleshooting of complex electronic SRS products.

ABOUT GEC ALSTHOM SCHILLING ROBOTIC SYSTEMS (SRS): Schilling Robotic Systems (SRS) is a world leader in telerobotic technology for use in extreme environments, and has earned an impeccable reputation for equipment design and customer service. SRS is seeking the best and brightest people to enhance an exciting product line of telerobotic manipulators and associated hardware for use in subsea applications. SRS is the place where people and machines work together to achieve practical engineering solutions through quality, reliability, and exceptional customer service. SRS offers the security of a company that was founded in 1985 in Davis, CA and joined the

GEC Alsthom group of companies in 1992. GEC Alsthom is one of the largest industrial organizations in Europe and provides SRS access to development laboratories and manufacturing facilities worldwide.

START DATE: Immediately.

KEY RESPONSIBILITIES:

- Performs electronic assembly, setup, operation, testing, troubleshooting, and reworking of product/prototype.
- Works from complex schematics and assembly blueprints.
- May prepare recommendations for improvements in methods or processes to enhance productivity and quality.

QUALIFICATIONS:

- Minimum of 2 years electronic assembly, testing, and troubleshooting experience (5 years preferred).
- Prefer experience with or certified in mil spec soldering.
- Prefer leadership/supervisory experience.
- Prefer test equipment development and implementation experience.
- Prefer control system setup and operation experience.

KNOWLEDGE AND SKILLS REQUIRED:

- Ability to read and understand blueprints and schematics.

- Proficiency in soldering.
- Prefer PC skills.

COMPENSATION PACKAGE: SRS offers a competitive salary/benefit package which includes:

- Health, dental, and life insurance.
- Flexible spending plan for child care and unreimbursed medical expenses.
- Employer 401K.

FOR MORE INFORMATION ABOUT SRS: Visit us on the World Wide Web at http://www.schilling.com .

http://www.schilling.com/jobs/eatt.html

Activity 14

Advanced Use of Subroutines

OBJECTIVES

In this activity you will accomplish the following:

- Use the SCORBASE command Interrupt in a program.
- Use Labels and Variables in a program.
- Use the If Jump command in a program.
- Program the robot to simulate a sterilization process using a Subroutine.

SKILLS

In this activity you will develop the following skills:

- Academic & Employability:
  - Operate lab equipment according to safety regulations.
  - Document inventory and safety procedures for lab set-up and shutdown.
- Occupational & Technical:
  - Write a program that includes loops, variables, and truth conditions
  - Program a variable whose value will change when a subroutine is called by the input
  - Describe quality issues resulting from improper procedures.
  - Utilize troubleshooting skills to improve the production process.
  - Modify the design of the program to improve the design process.
  - Implement modifications to meet changes in design criteria.

MATERIALS

In this activity you will need the following materials:

- Computer with RoboCell software
- Diskette or personal subdirectory on computer hard drive
- Worksheets for Activity 14

OVERVIEW

Review of Subroutines

In Activity 12, you used a subroutine to control the robot actions. The relationship between a program and its subroutine is shown in Figure 14-1.

1.	Command	Beginning of program
2.	Command	
3.	Command	
4.	Command	
5.	Call Subroutine	
6.	Command	
7.	Call Subroutine	
8.	Command	End of program
9.	Set Subroutine	Beginning of Subroutine
10.	Command	
11.	Command	
12.	Command	
13.	Return from Subroutine	End of Subroutine

Figure 14-1

Program execution starts from the first line of the program (the first command). During program execution, commands are executed one after the other in the order in which they appear. When the Call Subroutine command is executed, the main program execution is halted and the commands of the subroutine are then executed one after the other. At the end of the subroutine, the Return command returns execution back to the line following the Call command in the main program.

One of the great advantages of using a subroutine is that you can order the robot to repeatedly perform the same set of commands without repetitious programming. In the above example, the main program flow is unidirectional (starting from the beginning of the program to its end).

Advanced Use of Subroutines

In Activity 13, you used a Subroutine command to program a robotic cell that sterilized products. The working procedure of the program was as follows:

1. The robot placed the object requiring sterilization on a conveyor.
2. The conveyor was started, moving the object into the sterilizing machine.
3. When the sterilization process was finished, the conveyor moved the object out of the sterilizing machine. The object was then detected by a sensor.
4. Following detection, the conveyor was stopped and the robot picked the object and placed it on a template.

In the program from the previous activity, when the robot reached position #13, the program execution paused until the sensor detected the object. Upon detection, the program performed the subroutine. However, if the object was not detected by the sensor, the program was terminated after a 30 second pause.

To prevent the termination of the program (if the object was not detected), you will modify the program from Activity 13 by adding another subroutine. Adding another subroutine is not without its problems, however. After executing the subroutine commands, the program will return to the Wait command and pause program execution for another 30 seconds. Reexecuting the Wait command is unnecessary and program execution should terminate after the object has been placed on the template.

To solve this problem, you will use a flag variable named STOP_LOOP. When the robot reaches position #13, the program will set the value of this variable to zero (flag is down) and wait three seconds (using the Wait command). If after three seconds the value of STOP_LOOP is still zero, the program will wait another three seconds and then recheck the value of STOP_LOOP. This procedure will continue in an endless loop while the value of STOP_LOOP is zero. In the case of an interrupt, the subroutine will be called and it will set the value of STOP_LOOP to one.

The If Jump Command

The **If Jump** command is a conditional jump command that is used to check the value of variables. If the condition is true, program execution will jump to the line that contains the specified Label. The command, located in the Program Flow section of the Command List, opens the dialog box shown in Figure 14-2.

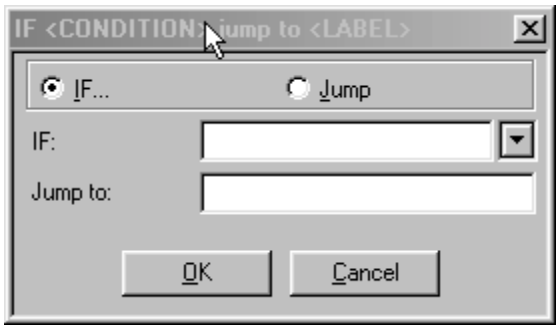


Figure 14-2

The Set Variable Command

The **Set Variable** command allows you to assign a value or an expression to a variable. In this activity, you will assign the variable "STOP_LOOP" to the values "0" and "1."

The Set Variable command, located in the Program Flow section of the Command List, opens the dialog box shown in Figure 14-3.

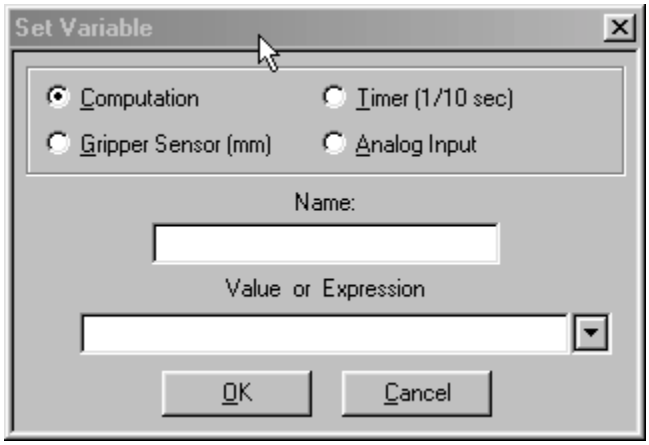


Figure 14-3

PROCEDURES

Task 14-1: Running RoboCell, Opening the 3D model File and Loading the Positions and Program

This activity requires the same positions recorded in Activity 13, and much of the program is similar, as well. To avoid the tedious rerecording of the positions, you can simply load the file saved at the end of Activity 13 (containing the program and positions) and save it under a new name. Then use the cut and paste tools to make the necessary changes in the program.

1 Turn on the computer and run RoboCell software.

2 Open the 3D model file **ACT13.3DC**.

3 Open the program and positions file **USER13**, which you saved in the previous activity.

4 Now save the file under the new name **USER14**.

5 The program you will write is almost the same as in the previous activity:

- Enable interrupt.
- Pick the cylinder and place it on the conveyor.
- Start the conveyor and move to position #13.
- Wait for the interrupt signal.
- When the interrupt is executed, call a subroutine that will stop the conveyor.
- Immediately return to the main program.
- Pick the object from its position on the conveyor where it faces the sensor.
- Place it on the template.

6 When you finish (or give up), compare your program with the following:

```
1     Remark: ************************************
2     Remark: ACT14
3     Remark: Advanced Use of Subroutines
4     Remark: ************************************
5     On Input Interrupt 1 on call sub. STOP
6     Open Gripper
7     Go to Position 11 fast
8     Go to Position 1 speed 5
9     Close Gripper
10    Go to Position 11 fast
11    Go to Position 12 fast
12    Go to Position 2 speed 5
```

```
13      Open Gripper
14      Go to Position 12 fast
15      Start Conveyor axis 7 at speed 3 in Plus direction
16      Go Linear to position 13 fast
17      Set Variable STOP_LOOP = 0
18      WAIT:
19      WaiT 30 ( 10 ths of seconds )
20      IF STOP_LOOP==1 jump to CONTINUE
21      Jump to WAIT
22      CONTINUE:
23      Disable input Interrupt 1
24      Go to Position 3 speed 5
25      Close Gripper
26      Go to Position 13 fast
27      Go to Position 14 fast
28      Go to Position 4 speed 5
29      Open Gripper
30      Go to Position 14 fast
31      Enable input Interrupt 1
32      Set Subroutine STOP
33      Disable input Interrupt 1
34      STop conveyor axis 7
35      Set Variable STOP_LOOP = 1
36      Enable input Interrupt 1
37      Return from Subroutine
```

Task 14-2: Running the Program

1 Save the program and positions file as **USER14**.

2 Reset the robotic cell.

3 Click on the first line of the program.

4 Run a single cycle of the program.

Task 14-3: Team Discussion and Review

Q *Modify the program you wrote in this activity as follows.*

- *Change the conveyor speed to fast (speed #10)*
- *Set the speed of the robot movement to position #13 to slow (speed 1).*

Q *Run the modified program.*

Q *What happened?*

Q *Give reasons why you think this happened and suggest a way to correct this bug.*

Task 14-4: Shut Down

1 Exit RoboCell.

2 Then turn off the computer.

Activity 15

Conclusion

OBJECTIVES

In this activity you will accomplish the following:

- Measure your knowledge of robotics.
- Design, program and simulate a robotic application.

SKILLS

In this activity you will develop the following skills:

- Academic & Employability:
  - Operate lab equipment according to safety regulations.
  - Document inventory and safety procedures for lab set-up and shutdown.
- Occupational & Technical:
  - Program the robot to stack two cylinders on the table at a given location.
  - Program the robot to stack two cubes on the table at a given location.
  - Program the robot to stack two prisms on the table at a given location.
  - Implement modifications to meet changes in design criteria.
  - Modify the design of the program to improve the design process.
  - Use problem-solving skills to solve design challenges.

MATERIALS

In this activity you will need the following materials:

- Computer with RoboCell software
- Diskette or personal subdirectory on computer hard drive
- Post-Test and Post-Test Answer Sheet
- Worksheets for Activity 15

OVERVIEW

Post-Test

This activity concludes the Robotics and Materials Handling 2 tekLINK. The test you will now take will measure your knowledge and skills in the field of robotics.

Take the **Post-Test** according to your teacher's instructions. Allow 30 minutes for the test.

When you have finished the test, hand it in to your teacher.

Final Projects

The final projects in this activity are designed to demonstrate various properties of 3-dimensional space. The first project will not be difficult but the second and third will be more challenging. Do as many of the projects as time allows.

PROCEDURES

Task 15-1: Final Projects

Do as many of the following projects as you can in the time that remains:

1. Load the 3D model file **ACT15A.3DC** in which two cylinders (height = 35 mm) are placed on a table.

   Program the robot to stack the cylinders on the table at (220,0). Note that the cylinders' contact areas should overlap.

   Save the program and positions in file **USER15A**.

2. Load 3D model file **ACT15B.3DC**. This cell is similar to the first cell except that the cylinders were replaced with cubes (height = 35 mm). By using the Show Object Positions options, you can see the cubes are placed exactly in the same positions where the cylinders were.

   Program the robot to stack the cubes on the table at (220,0). Note that the cubes' contact area should overlap.

   Hint: Modify the program you just wrote.

   Save the program and positions in file **USER15B**.

3. Load the 3D model file **ACT15C.3DC**. This cell is also similar to the two previous cells except that the cylinders/cubes were replaced with two prisms (height = 35 mm; length = 35 mm; width = 50 mm). The prisms are placed exactly where the cylinders/cubes were.

Program the robot to stack the prisms on the table at (220,0). Note that the prisms' contact area should overlap.

Hint: Modify the program that you just wrote.

Save the program and positions in file **USER15C**.

Task 15-2: Shut Down

1. Exit RoboCell.
2. Then turn off the computer.