

## Fast and powerful robotics simulation

RoboLogix is a robotics simulation software package that is designed to emulate real-world robotics applications. It is, essentially, a software program that converts your computer into a fully-functioning 3D industrial robot that can be programmed to perform various functions and operations. With RoboLogix, you teach, test, run, and de-bug programs that you have written yourself using a five-axis industrial robot in a wide range of practical applications. RoboLogix allows the user to run the simulator to test and visually examine the execution of robot programs and control algorithms, while plotting instantaneous joint accelerations, velocities and positions.

RoboLogix introduces the concepts of multi-axes Robots and demonstrates how they can be used in a plant or manufacturing system utilizing 3D simulation. The primary focus of the software is on automated manufacturing processes and allows for the programming, testing, and debugging of robot programs. Users gain practical, "hands on" programming of an industrial robot through a combination of teach-pendant programming, and 3D simulation environments. It is a powerful learning tool that provides students with access to robotic equipment worth tens of thousands of dollars.

# RoboLogix™

## User Guide



## ***Contents***

|                                   |    |
|-----------------------------------|----|
| 1. Introduction                   | 1  |
| 2. Control Panel                  | 5  |
| 3. RoboLogix Vision System        | 8  |
| 4. Work Envelope                  | 10 |
| 5. RoboLogix Programming Language | 12 |
| 6. Palletizing                    | 29 |
| 7. Tutorials                      | 34 |

## 1. Introduction

The term *robotics simulation* can refer to several different robotics simulator applications. For example, in mobile robotics applications, behavior-based simulators allow users to create simple worlds of rigid objects and light sources and to program robots to interact with these worlds.

One of the most popular industrial applications for robotics simulation is for 3D modeling and rendering of a robot and its environment. This type of robotics software has a simulator that is a “virtual” robot, which is capable of emulating the motion of an actual robot in a real world envelope. Some robotics simulators, such as RoboLogix, use a physics engine for realistic renderings and movements of the robot in 3D space. Physics engines provide precise physics models with variables such as mass, velocity and friction, and can simulate and predict effects under conditions that approximate the real world. The use of 3D robotics simulation for development of robotics programs is highly recommended regardless of whether an actual robot is available or not. The simulator allows for robotics programs to be conveniently written and debugged off-line, with the final version of the program tested on an actual robot.

In order to fully understand the operation of a robot, it is necessary to test, run, and debug programs that you have written yourself. Learning how to program a robot requires practice with the equipment and the capability to learn by trial-and-error. In robotics laboratories or in manufacturing applications, access to equipment is limited and there are few trial-and-error experiences possible due to the risk of injury to personnel and/or damage to equipment. Most Colleges and Universities offering robotics training generally have a small number of robots, since the cost of each robot can be over \$100,000.

Robotics simulation software, such as RoboLogix, is an increasingly popular method of bridging the gap between robot theory and “hands-on” programming. With RoboLogix, the user can run the simulator to test and visually examine the execution of robot programs and control algorithms, while plotting instantaneous joint accelerations, velocities and positions. This type of simulation software allows for verification of the reach-ability, travel ranges and collisions. This provides increased reliability of the planning process and program development as well as reducing the overall completion/commissioning time. RoboLogix enables programmers to write their own robot programs, modify the environment and use the available sensors, which include video cameras to obtain the desired position of the robot and end effector.

RoboLogix is ideal for students as well as robot designers and engineers. Tradespeople such as electricians, millwrights and maintenance mechanics will benefit from the enhanced technical knowledge and skills they will attain from having unlimited access to a fully-functioning robot. It is the only robotics simulation tool that provides engineering-level simulation at such an affordable price. The simulation software allows for verification of the reach-ability, travel ranges and collisions. This allows for increased reliability of the planning process and program development as well as reducing the overall completion/commissioning time.

RoboLogix enables programmers to write their own robot programs, modify the environment and use the available sensors. These sensors include both analog and digital detection devices and are used for functions such as obtaining the desired position of the robot end effector. In addition a teach pendant/control panel is included with the simulator that allows the user to command the robot to pick up a tracked object and return it to a home location through jogged commands or pre-programmed positions.

RoboLogix is a robotics simulation software package that is designed to emulate real-world robotics applications. With RoboLogix, you teach, test, run, and de-bug programs that you have written yourself using robot arms and end effectors in a wide range of practical applications. These applications include pick and place, palletizing, welding, and painting and allow for customized environments so that you can design your own robotics application. With RoboLogix, the user can run the simulator to test and visually examine the execution of robot programs and control algorithms, while plotting instantaneous joint accelerations, velocities and positions.

The benefits of RoboLogix include:

- User-friendly 3D interface allows for “real-world” simulation
- Test and de-bug programs in a safe, non-hazardous environment
- Perform accurate robot simulations to verify reach, cycle time, through-put, etc.
- Design or edit robotic programs without tying up programming time on an actual robot
- Compare robotic programs in order to optimize cycle times
- Enter, modify and retrieve programs using a simulated teach pendant/control panel

RoboLogix introduces the concepts of multi-axes robots and demonstrates how they can be used in a plant or manufacturing system utilizing 3D simulation. The primary focus of the software is on automated manufacturing processes and allows for the programming, testing, and debugging of robot programs. Users gain practical, “hands on” programming of an industrial robot through a combination of teach- pendant programming, and 3D simulation environments. It is a powerful learning tool that provides students with access to robotic equipment worth tens of thousands of dollars.

The RoboLogix programming environment is completely safe, but provides a very realistic simulation of control systems using robotic equipment. The ability to preview the behavior of a robotic system in a “virtual” world allows for a variety of mechanisms, devices, configurations and controllers to be tried and tested before being applied to a “real world” system.

RoboLogix receives control signals, determines if contact or collision between objects in the system has occurred, and returns simulated sensor information as feedback. This system has the capacity of real-time simulation of the motion of an industrial robot through 3D animation. The principles of 3D motion simulation and both geometry modeling and kinematics modeling are presented in the RoboLogix virtual environment.

The simulation functions of RoboLogix include:

- Automatic interference checks and collision detection
- Full trajectory trace of Tool Center Point (TCP)
- Management of up to 12 virtual cameras
- Inclusion of signals from virtual sensors
- Dynamic performance analysis
- Cycle time analysis

## 2. Control Panel

The control panel consists of both robot control instructions as well as environment control functions such as conveyor controls, on-off hardwired control, etc. Various program command keys are also included on the control panel. These keys are used to write, edit, store, recall, and execute robotic programs.



RoboLogix Control Panel

The following instructions are accessed from the RoboLogix control panel:

- |               |   |
|---------------|---|
| C1 ON/REV/OFF | For./Rev./Off commands for Conveyor 1   |
| C2 ON/REV/OFF | For./Rev./Off commands for Conveyor 2   |
| SAVE POSN     | Teaches the location by storing positional data in a register. Stores current location into next position register. |
| RESET         | Clears faults, such as workpiece collisions.  |
| SETUP         | Provides access to dialog box containing system settings such as conveyor speed, jogging speed, etc.                |

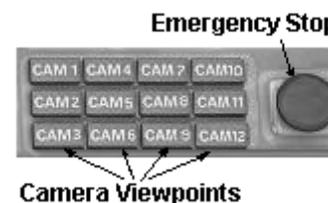
### Control Panel Keys (cont'd)

|                     |  |
|---------------------|--|
| RUN                 | Executes program loaded into memory.   |
| ZERO                | Location of end effector when all rotational position data is zero.  |
| HOME                | Returns end effector to home (start-up) position   |
| CAM 1- 12           | Camera viewpoints located throughout work envelope.  |
| P []                | Access position register screen to create and edit position register information (X, Y, Z, A1–A5)                                      |
| R []                | Variable register screen key, used to access display screen for entering, editing, deleting, and saving variable register information. |
| PROG                | Access programming screen to create, modify and delete programs written as lines of code.  |
| PAL                 | Access palletizing setup screen to create, edit and delete palletizing and de-palletizing routines.                                    |
| GRIPPER O/C         | End effector control buttons. Instruct gripper to either open or close.  |
| X +/-, Y +/-, Z +/- | Linear position register point.  |
| A1 – A5             | Angular position register point.   |
| Up/Down arrows      | Adjust feedrate by either increasing or decreasing feedrate a 5% intervals.  |

The conveyor controls on the control panel are marked as C1 ON/REV/OFF and C2 ON/REV/OFF. These hardwired conveyor controls override any programmed conveyor instructions. The control panel is capable of providing up to five axes of control, with each axis represented by A1, A2, A3, A4, and A5. These keys are often referred to as “jog” instructions and are used to move, or jog, a particular axis a certain number of increments or degrees. In addition, XYZ keys are also provided to allow movement in orthogonal directions. The Gripper Open/Closed instructions are used to pick up and release objects to be held by the end effector. The Emergency Stop button is shown as a large red pushbutton and will disable all programmed instructions and machine operations when pressed.

Motion control of the robot is accomplished by moving the robot from one location (position) to another. This movement is achieved by setting the angular (A1-A5) coordinates and the linear (X, Y, Z) coordinates. Generally, angular position movements are commonly used for large (course) motion and linear position movements are often used for smaller (fine) increments.

The Emergency Stop button is indicated by the large red button on the control panel. When this button is activated power is interrupted to the entire work envelope including the robot, conveyors, and any other equipment operating in the immediate vicinity of the robot.



### 3. RoboLogix Vision System

RoboLogix provides 12 viewpoints, or camera angles, for a given robot work envelope. These viewpoints are accessed by the nine CAM keys and allow for the viewing from a variety of angles and perspectives. By using these camera viewpoints, the user can move around in a 3D world in much the same way they would in the real-world. When programming the robot, the camera views allow you to make fine adjustments to the arm and gripper position, or to view the entire work envelope and surrounding area.

The following 12 camera views are available:

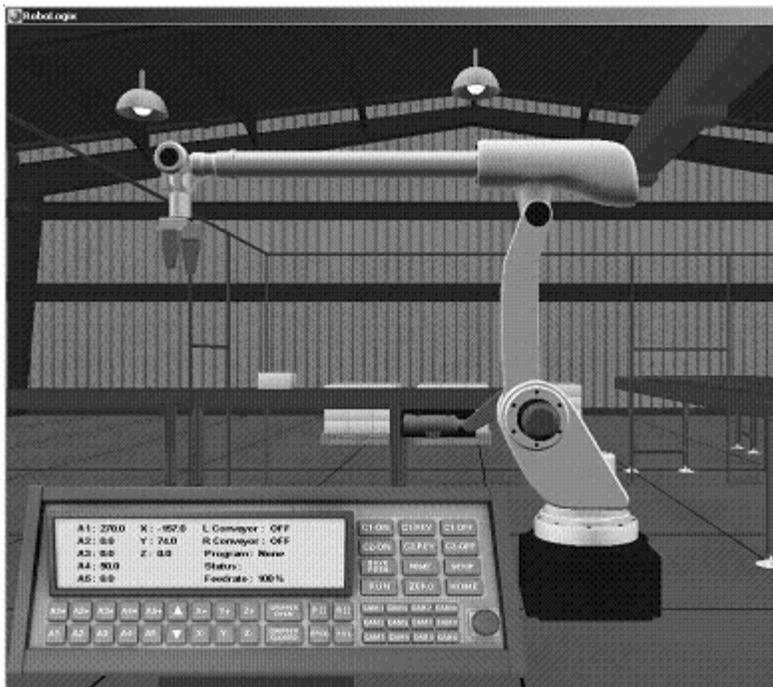
- CAM1** Close-up view from the front of building (facing North). Conveyor 1 is shown on the left side of the screen and Conveyor 2 is located on the right.
- CAM2** Close-up view of Conveyor 1. This view is helpful for workpiece position observation and gripper adjustment. Note blue-colored motor drive on Conveyor 1. This is a useful reference point for other conveyor views.
- CAM3** Close-up top view of both conveyors. Conveyor 1 is shown on left.
- CAM4** Faces toward front of building, with main loading door at far end with an exit door beside the loading door. This view also includes the south loading door for the fenced-in work envelope. The operator's entrance to work envelope is shown to the right (west) of these loading doors. An empty pallet visible in this camera angle, and it is located on the floor between Conveyor 1 and the robot.

#### *Camera Viewpoints (cont'd)*

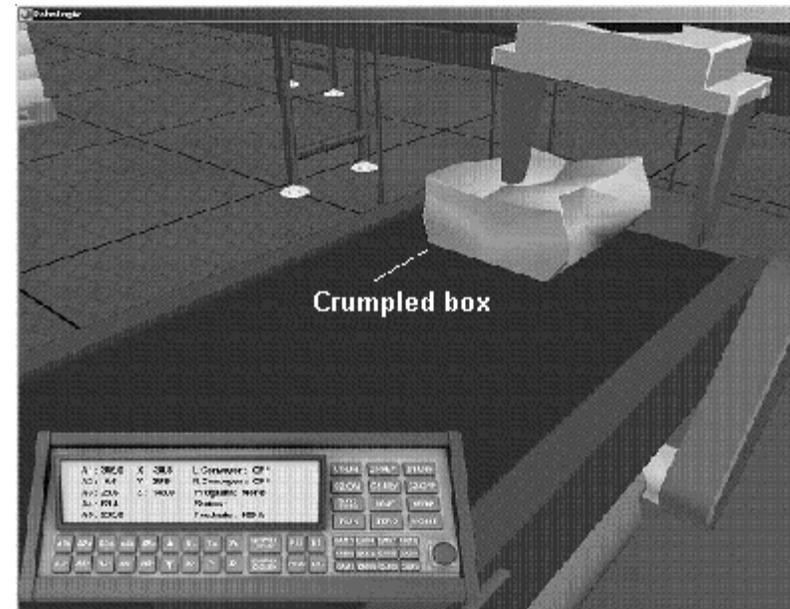
- CAM5** View facing south-west towards the side of the building. The camera is positioned to provide a close-up view of Conveyor2. This view includes the box chute, which is where workpieces (boxes) enter from and exit into.
- CAM6** Shows a close-up view of Conveyor 2. This view also includes the east box chute for workpieces entering and exiting Conveyor 2.
- CAM7** Overhead view of pallet, with Conveyor 1 shown at top of screen and Conveyor 2 on the right side.
- CAM8** Viewpoint from front of building (South end) looking North. Note the three pallets stacked with boxes located outside of work envelope.
- CAM9** Viewpoint looking west, directly into box chute. Provides close-up view of Conveyor 2.
- CAM10** Viewpoint facing east with side loading door in sight. Provides close-up view of pallet for stacking and unstacking boxes.
- CAM11** Top view facing south, towards front of building.
- CAM12** Viewpoint from end-effector (grripper). Can be used to provide panoramic view from anywhere in work envelope.

#### 4. Work Envelope

The default work envelope for RoboLogix is shown in the image below. This scene consists of two conveyors, workpieces (boxes), pallets, and the control panel. The robot arm used in this environment will perform pick-and-place, point-to-point, palletizing and de-palletizing and a range of instructions to demonstrate programming and de-bugging operations.



When a workpiece becomes damaged due to collision, falling, or improper gripper position, the workpiece will turn into a crumpled box. The damaged workpiece can be cleared, or removed, from the work envelope by pressing the *Reset* key on the main menu of control panel.



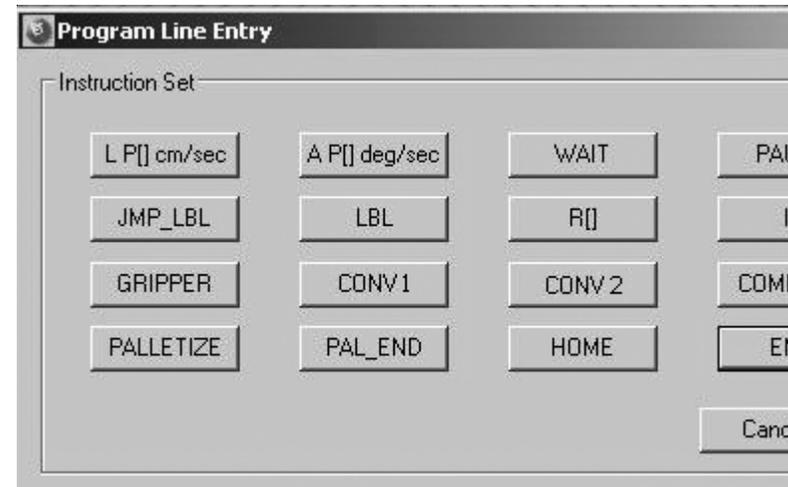
## 5. RoboLogix Programming Language

RoboLogix uses a scripting-type programming language that uses a command, or instruction set, that is common among major robot manufacturers. Like most robot programming languages, RoboLogix programs consist of data objects and program flow. The data objects reside in registers and the program flow represents the list of instructions, or instruction set, that is used to program the robot. RoboLogix program language is a type of scripting language that is used to control the software application.

Programming languages are generally designed for building data structures and algorithms from scratch, while scripting languages are intended more for connecting, or *gluing*, components and instructions together. Consequently, the RoboLogix instruction set is a streamlined list of program commands that are used to simplify the programming process and provide rapid application development.

An important consideration when writing programs in any language is to have good documentation to accompany the lines of code that have been created. This makes the process of manually interpreting the program code much easier for modification, adjustment, and understanding the program operation. With RoboLogix, the documentation feature is provided by *Comment Lines*, which can accompany every line of code that is written. When writing comments for documentation purposes, it is important to be as brief, but concise as possible. Usually only a few words or a short sentence is all that is needed to describe a particular programmed instruction. For example, the comment to accompany C1\_ON could be something as simple as “Energize packaging conveyor”. Comments can also be attached to program names, registers, and any instruction contained in the instruction set.

The RoboLogix instruction set contains 16 commands, which are usually written as a program on a line-by-line basis. These commands are used to instruct the robot to perform tasks such as moving to a specific location, picking up an object, executing a subroutine, waiting, etc.



RoboLogix Instruction Set

One of the more popular commands in the instruction set is the IF instruction, which compares numerical values located in two registers. If a register has a value that is greater than (>), less than (<), greater than/equal to (>=), less than/equal to (<=), equal to (=), or not equal to (<>) another register, it will execute the next line in the program if the condition is true. The IF command is often used with the JMP LBL instruction to control program execution.

The RoboLogix instruction set consists of the following commands:

|              |   |
|--------------|---|
| LP[] cm/sec  | Linear motion instruction which moves tool center point (TCP) in a straight line to a new position register (X, Y, Z) point.  |
| AP[] deg/sec | Angular motion instruction which moves TCP in an angular motion to a new position register (A1 – A5) point.   |
| R[]          | Performs arithmetic operations on two registers, and stores result in third register. Arithmetic operations performed include +, -, x, /.   |
| GRIPPER      | End effector control. Instructs gripper to either open or close.  |
| IF           | Compares values located in two registers. If a register has a value that is >, <, >=, <=, =, or <> to another register, it will execute the next line in the program if the condition is true. Otherwise, it skips to the next line.  |
| WAIT         | Performs a time-delay function where it will wait for a pre-determined amount of time before activating, or will wait until one register is greater than/less than/equal to/not equal to the other. This instruction will wait for a condition or period of time before continuing. |
| LBL          | A location in the program that the processor jumps to in order to continue program execution. The LBL can be any location in the program which has been identified (labeled) as a destination point to execute from.  |

*Instruction Set (cont'd)*

|           |   |
|-----------|---|
| JMP LBL   | Instructs the processor to jump over certain parts of a program and to execute the program beginning at the point indicated by a label (LBL).   |
| PALLETIZE | Provides access to Palletize instruction dialog box with data pertaining to rows, columns, and layers (RCL) of objects to be palletized, or stacked. It also contains palletizing motion data for approach, stack, and retraction path points. Palletize and de-palletize operations are selected using this instruction. |
| PAL_END   | Palletizing end instruction is a command issued to complete, or finish, the given palletizing operation.  |
| PAUSE     | When initiated, it suspends program operation. Program resumes from where it was halted when PAUSE is de-activated.   |
| HOME      | Returns end effector to home (start-up) position. Default, or starting point of robot.  |
| END       | Instructs the program to end, or terminate.   |
| CONV1     | Conveyor 1 control. Provides forward/reverse/OFF control of conveyor I on left side of screen.  |
| CONV2     | Conveyor 2 control. Provides FWD/REV/OFF  |

### Instruction Set (cont'd)

All instruction set information is stored in registers, which are data locations capable of holding variable numeric values. The difference between the display of Registers and Instructions is that Registers use square-brackets “[ ]” to indicate a Register address. For example, Register number 1 would be written in RoboLogix as R[1]. Instructions are always indicated in capital letters (i.e. IF, WAIT, etc.) and may use an underscore “\_” to indicate an address or identity.

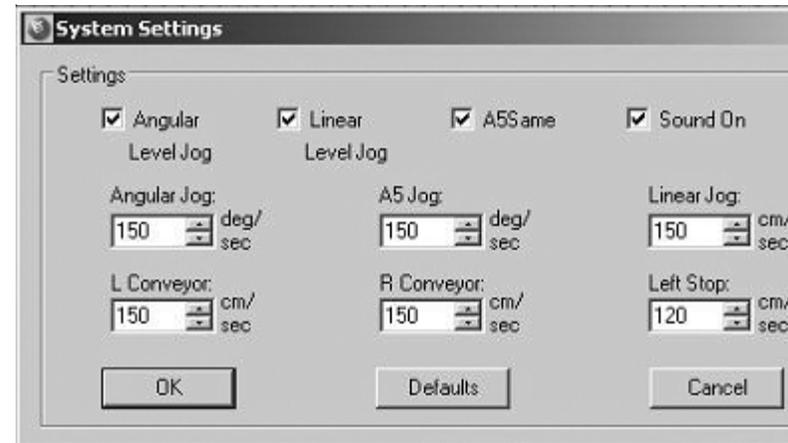
There are two main types of registers used by RoboLogix: position registers and variable registers. Position registers contain both the linear and angular data point coordinates and include axis (joint) information for A1, A2, A3, etc. and for X, Y, Z linear, or Cartesian coordinates. There are also 32 variable registers which can be used for holding instruction set data such as position comparisons and time-delay information.

In addition to position registers and variable registers, some robots also have palletizing registers, which are used to manage the position of the stack point in palletizing applications. RoboLogix uses a position register to function as a palletizing register and, consequently, does not have separate palletizing registers.

Angular position points (AP[n]) and linear position points(LP[n]) can each hold the same type of data, but they are designed to perform different functions when used in a program. For example, angular motion is best represented by an angular position point, while direct motion is best represented by a linear position point.

Any data in any location, including program lines and registers, are selected by left-clicking on the numbered grey buttons on either the left side or at the top of the display screen. Once selected, a red highlight bar appears that allows data to be deleted or loaded for editing. To edit the contents of any register, it must first be highlighted and once the register has been highlighted, its value can be changed by entering a new value in the dialog box.

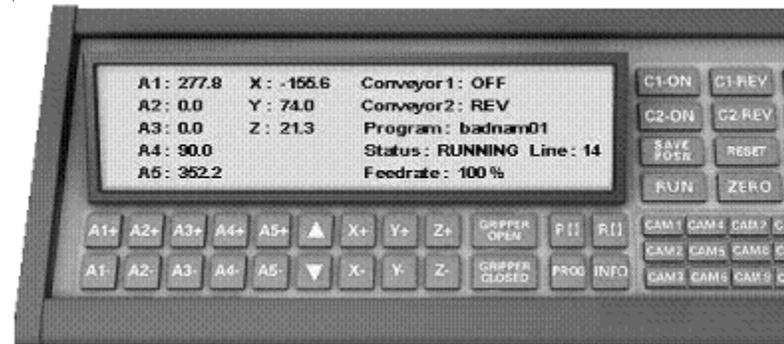
Each command contained in the instruction set is either angular or linear and consists of a feedrate override, which represents a percentage of linear, or circular feedrate. As mentioned earlier, feedrate override is an instruction that controls the programmed feedrate by an adjustable percentage during operation. The Setup key provides access to the system setup dialog box, shown below, with setting adjustments for feedrates for angular (Deg/Sec.) and linear (cm/Sec) jog and conveyor instructions.



The following system control settings are accessed through the system setup dialog box:

|                        |   |
|------------------------|---|
| Angular Level Jog      | Maintains workpiece in a level position while being jogged. X-Y, and Z-Y orientation is held constant.  |
| Linear Level Jog       | Maintains workpiece in a level position while being jogged in a linear motion. X-Y and Z-Y orientation is held constant.                        |
| A5 Same                | Maintains attitude in X-Z plane.  |
| Sound On               | Audio control (sound effects)   |
| Angular Jog            | Changes default maximum feedrate in deg/sec, which can be fractionally adjusted at 5% intervals using feedrate override buttons on main screen. |
| Linear Jog sec.        | Adjusts default maximum feedrate in cm/sec.   |
| A5 Jog                 | Adjusts A5 axis rotation speed in deg/sec.  |
| L Conveyor increments. | Adjusts speed of Conveyor 1 in cm/sec.  |
| R Conveyor             | Adjusts speed of Conveyor 2 in cm/sec.  |
| Left Stop              | Workpiece limit switch position. Indicates location on Conveyor 1 where workpiece will stop.  |

The RoboLogix programming console, or control panel, consists of five separate display screens: main display screen, programming screen, position register screen, variable register screen, and INFO screen. While RoboLogix is a proprietary robot programming language, the various commands, registers, display screens and overall program design and layout is intended to closely replicate the most common features found in modern industrial robots by companies such as Fanuc, Kawasaki, etc.



The Main Screen is shown below, and indicates angular joint positions (A1 – A5), and the Cartesian positional coordinates (X,Y,Z). The angular and linear joint position adjustments are accomplished by jogging, or inching, the robot towards its desired position using the corresponding A1 –A5 and X, Y, Z buttons on the control panel. Once it is in the desired position, the position coordinates are recorded using the SAVE POSN button.

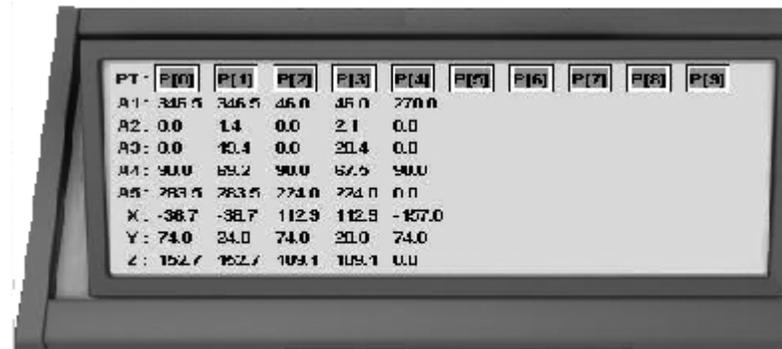


The Main Screen also displays active state information for the conveyors and any other ancillary control devices that may be used in the work envelope. Program information such as program name, status, program line count and feedrate is also displayed on the Main Screen. The feedrate adjusts the operating “speed” of the robot and can be altered by the vertical arrows located between A5 and X on the control panel. The feedrate can be varied in 5 degree intervals.

The Status line displays conditions such as LOADED, WAITING, RUNNING, Ended, ESTOP, and RESET. When the Status line indicates LOADED, it means that the program has been loaded and is ready to run. The WAIT status indicates that the program is waiting a pre-determined amount of time before it continues to execute. When the Status is shown as RUNNING, it means that the program is executing. ESTOP is only displayed when the red emergency stop button has been pressed. When the Status indicates RESET, it means that the RESET button has been pressed on the control panel. The RESET command is used to clear faults, such as a damaged workpiece and to reset the program back to its original starting point.

In addition to the default Main Screen displayed on the Control Panel, there are four additional screens that are displayed when programming with RoboLogix. One of these screens is for displaying position registers, variable registers, text-based messages, and the actual program code. The Control Panel has four buttons to access the position register screen (PRS), the variable register screen (VRS), the programming screen (PRG), and the text information screen (INFO).

The Position Register Screen (PRS) displays all assigned program positions. Each program written with RoboLogix can hold up to 10 position registers, P[1] – P[10], with each position register containing joint position and X, Y, Z coordinate data. Position register data can be altered (edited) by clicking on the position register heading, i.e. P[1], which will launch a dialog box for editing numeric data. The dialog box will also display all assigned labels, or comments. Each position register has a Comment function which allows for the entry of text “comments” regarding a device, program segment, or reference in the program.

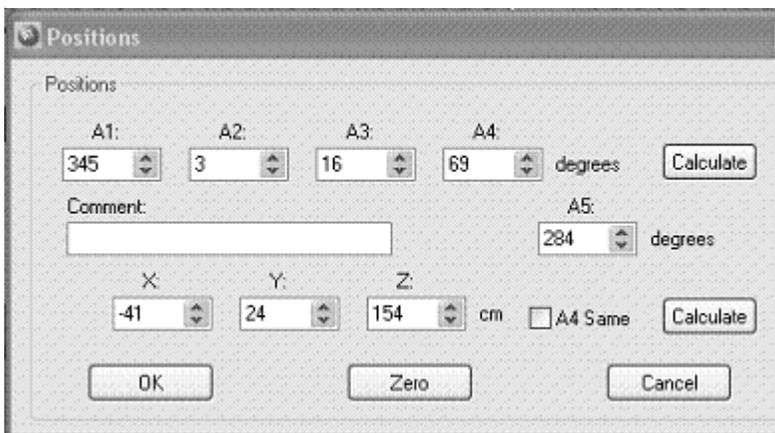


Position register screen

The buttons located on the right side of the PRS are labeled New, Del, Load, and Main. The New instruction assigns a position register to a new position. Specification of position can be done manually or by the SAVE POSN button on the Home Screen. The DEL command will erase contents of a selected position register. The LOAD instruction loads the position register data into the dialog box for editing. The Main buttons returns the control panel to the main, or home screen.

To store a position in a position register, the SAVE POSN button is pressed on the main control panel. This button basically takes a “snapshot” of the robot’s angular and linear coordinates and stores them in the next available position register. For example, if you create a new program, the position registers are initially empty. When you move the robot to the desired position and click the SAVE POSN button, the position data will be automatically stored in register P[0], the next saved position will be in P[1], and so on. It should be noted that for RoboLogix, and most major robot manufacturers, fine position adjustments are made more accurately using X, Y, Z adjustments instead of A1 – A5 angular adjustments.

To edit a stored position select the desired position point to be modified by left-clicking on the position (i.e. P0, P1, P2, etc) so that a red vertical highlight bar appears. Click on the *Load* button to load the position data and a *position display screen* should now appear. Note the two Calculate buttons on this display screen. These buttons are very important since whenever you alter the A1 – A5 position points or the X, Y, Z position points, you must press the corresponding *Calculate* button in order to make the proportional change between the angular and linear position points. For example, if you change the X, Y, Z parameters you must press the *Calculate* button located to the right of the X, Y, Z numeric displays.



A useful feature when editing position points is to be able to move the robot to the position being edited. This allows you to obtain a visual correlation between the numerical values of each axis, and the actual position of the robot. To view a position point in the Position Register Screen, select a register and click on the top arrow key located beside the emergency stop button on the control panel. The upper arrow key will advance the register by one position, and the lower arrow key allows you to move back one register. With these arrow keys, you are able to step the program in both a forward and backward direction to “walk” the program through its various programmed position points.

The Variable Register screen (VRS) displays 32 registers (R0 – R31) which contain numeric variables. These variables can be assigned for holding data such as instruction set command information. For example, if a palletizing instruction (PAL) is used, variable registers would be used to store information such as the number of workpieces (boxes) to be stacked.



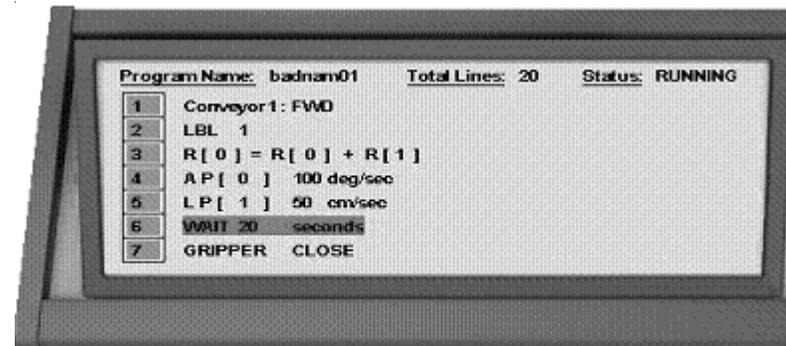
Variable register screen

Variable registers are used to store numeric information such as data for counters. The most common application for counters in robot programs is for counting loops. A loop is a repeated execution of a section of a program. By counting loops, you can control the number of times that a program is executed. For example, if you wish to stack five boxes, you would write a program to count five loops of a box-stacking routine and store the count data (5) in a variable register (i.e. R[1]).

Loops can be created by inserting a LBL instruction at the beginning of the program you wish to loop, and a JMP LBL command at the end. An LBL instruction is an unconditional branch which allows you to loop part, or all, of a program. The LBL and JMP LBL must have the same reference address in order to function. The following program code is an example of a loop routine:

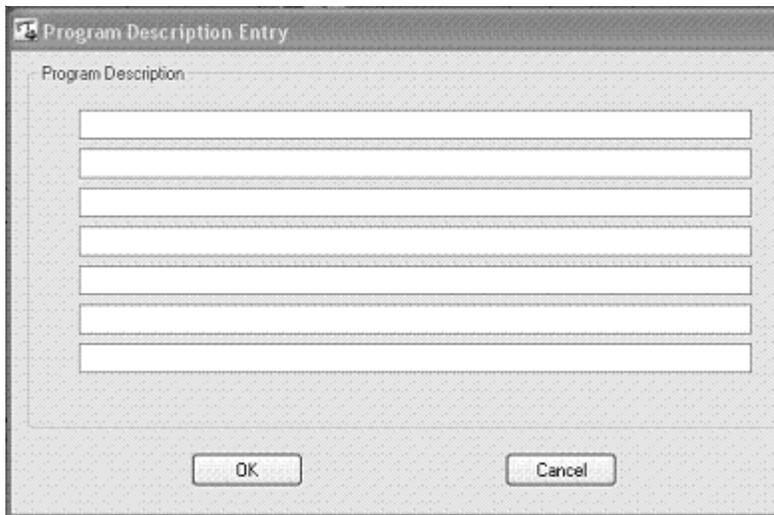
1. LBL [1]
2. AP[0]
3. WAIT 1
4. LP[1]
5. AP[2]
6. GRIPPER Close
7. AP[3]
8. GRIPPER Open
9. JMP LBL [1]

The Programming Screen (PRG) is the display screen where RoboLogix programs are entered, edited, saved, and retrieved. Each line of program code can be edited, by clicking on the line to launch a dialog box with various edit functions. A new line of code can be started by using the *New* instruction, which will add a new line immediately below the last line. The Programming Screen displays up to 7 lines of code at a time. When a program is longer than 7 lines, subsequent lines of code would be displayed on the next program screen which is accessed by the right arrow key located beside the emergency stop button.



Programming screen

The INFO button accesses a program description display screen, which can contain a short descriptive summary up to 400 characters in length. The program description display screen is shown below, and is the default display screen for any of the pre-built lab projects that are included with RoboLogix.



**Program description display screen**

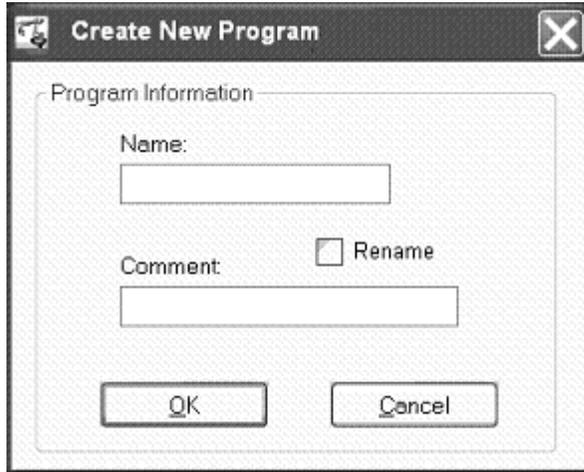
Program descriptions are useful when designing your own programs since it allows you to write and store additional documentation to accompany your program. The *New* and *Load* keys on the program description display screen allow you to write a new description or edit an existing description.

Programs that have been previously written and stored can be retrieved from system memory by pressing the *Load* button on the Programming Screen. This instruction will launch a dialog box which will allow you to search for a program on the system's hard-disk or ancillary memory device. When there are large numbers of programs stored in memory, they can be browsed by using the arrow keys located beside the red emergency stop button on the Programming Screen. The right arrow key allows you to view the next 7 program names stored in memory and the left arrow key provides access to the previous 7 program names.

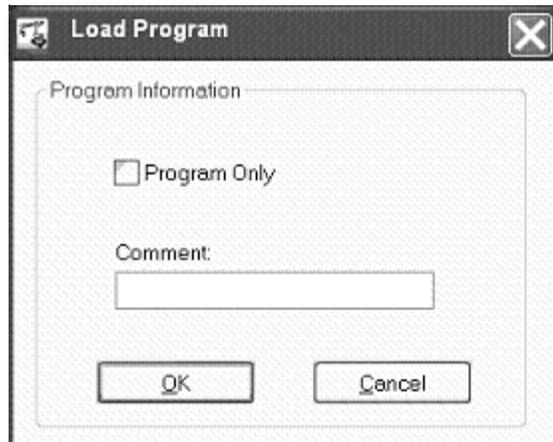
Once a program has been loaded, it can be launched, or executed, by using the *Run* function. When the *Load* button is pressed in the programming mode, it loads the selected program, and a dialog box appears called "Program Name Entry" which allows you to change the program name or comment. When a program is running, the Program Display Screen will show a red highlight bar. This highlight bar indicates which program line is being executed, and moves sequentially from one program line to the next.

Whenever a program is created or modified, it is automatically saved. Any change that is made to any program variable is saved automatically. The autosave feature is activated when you change from one display screen to another or when you exit the program.

When creating, or writing, a new program, the *Create New Program* dialog box appears when you press the *NEW* key on the program display. The Create New Program dialog box allows you to enter the name of the new program as well as a descriptive comment. In addition, this dialog box also has a setting called *Rename* which allows you to make copies of the program. For example, if you wish to make a copy of program "PAL1", select the program by pressing the *PROG* key and loading the program. Once the program is loaded, press the *NEW* key, select *Rename* in the dialog box and change the name of the program (i.e. PAL2). By clicking "OK" you have now created a new program (PAL2) which is identical to the original program (PAL1).



When you load an existing program, a *Load Program* Dialog box appears which has a Program Only setting to allow you to load only the program lines of code, but not the program's variable registers, position points, conveyor speeds, etc. The Program Only feature allows you to load a new program while retaining all of the control data and position information from another program. It is useful in applications such as palletizing, where you can write one program to palletize workpieces in a certain configuration, and write a second program to de-palletize without changing the configuration.



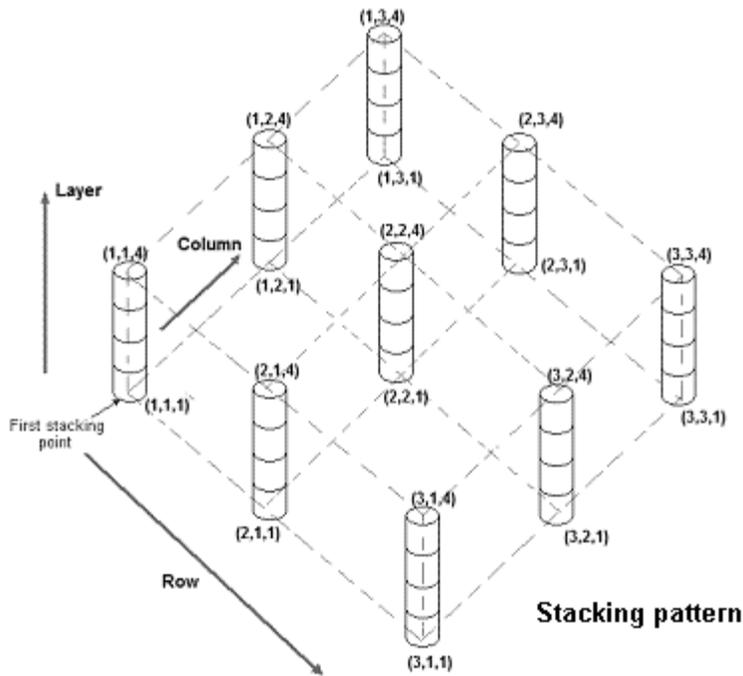
As with any programming language, there are many “rules of thumb” that are useful when writing programs using RoboLogix. Some of these recommendations for writing lines of code include:

- Lines of code should be less than 50 characters in length and be limited to one statement per line.
- Restrict each program to a maximum of 20 lines of code. Programs larger than 20 lines should be broken into sub-programs.
- When naming procedures in a program, indicate action by using verbs, and use names for functions that return values.
- Use comments to explain *why* and *what* is being performed in the program, rather than *how* it is being performed.

One of the most important rules of thumb for robot programs using a gripper is to be as accurate as possible with *grripper position points*. In order to pick up a workpiece (box) using RoboLogix, the gripper must be precisely positioned for the gripper to effectively close and pick up the box. Precise alignment requires that the gripper's fingers are located at the center of the box and that the fingers are extended almost to the bottom of the box.

## 6. Palletizing

One of the more popular applications for robots is palletizing, which is accomplished by combining a series of commands into a palletizing routine. A palletizing routine consists of five program instructions, or lines. The first instruction is the Palletize instruction, which is followed by a linear motion instruction to move to the stack point. The stack point is the top-center of the workpiece, and a group of stack points forms a stacking pattern. The first stack point in a stacking pattern is located at (1,1,1) and is incremented each time the palletizing routine is executed.



The number of workpieces to be stacked in a palletizing routine is a product of the number of rows, columns, and layers. In equation form:

$$\# \text{ of workpieces palletized} = RCL$$

where R = # of rows  
 C = # of columns  
 L = # of layers

The first command in a palletizing routine determines the approach/retract point. It does this for each workpiece to be stacked every time the routine executes. As part of this first command, the robot moves the workpiece to the approach/retract point. The second command moves the workpiece in a linear (downward) direction to the stack point. The third command opens the gripper. The fourth command is a WAIT instruction, and the fifth is the PAL\_END instruction. The PAL\_END instruction moves the end effector directly upward to the approach/retract point.

In order to perform a palletizing operation, the RCL values are set using the PAL instruction, and a counting routine is required to count the number of boxes to be stacked and to stop the robot when the count is complete. The count is generally accomplished in palletizing operations by using an IF instruction to increment the count and an IF instruction to terminate the count when the preset value equals the actual value.

The following IF instruction can be used to automatically increment a register by a value of 1:

$$R[0] = R[0] + R[1] \quad \text{where, } R[1] = 1$$

With this instruction, R[0] is initially zero and each time the instruction is executed the value of R[0] increases by a value of 1.

The following IF instruction can be used to compare the value of the incremented register with the preset value to be counted:

```
IF R[0] <= R[2] THEN
  JMP_LBL 1
```

With this instruction, if R[0] is less than or equal to R[2] then the program is instructed to jump to the beginning of the program routine. If R[0] is greater than R[2], then the program will stop.

The following program is an example of a palletizing routine to stack 18 workpieces on a pallet:

| <b>Program code</b>                 | <b>Comment</b>                 |
|-------------------------------------|--------------------------------|
| Conveyor 1 FWD                      | Start Conveyor 1               |
| WAIT 20 seconds                     | Load workpiece                 |
| LBL 1                               | Start palletizing              |
| routine                             |                                |
| R[0] = R[0] + R[1]                  | Increment                      |
| register instruction                |                                |
| Conveyor 1 FWD                      | Restart Con-                   |
| veyor 1                             |                                |
| AP[0] 100 deg/sec                   | Move to 1 <sup>st</sup>        |
| position point                      |                                |
| LP [1] 100 cm/sec                   | Move to 2 <sup>nd</sup> (grip) |
| position point                      |                                |
| WAIT 4 seconds                      | Wait for                       |
| workpiece to settle                 |                                |
| GRIPPER Close                       | Pick up                        |
| workpiece                           |                                |
| LP [0] 100 cm/sec                   | Retract to 1 <sup>st</sup>     |
| position point                      |                                |
| Conveyor 1 FWD                      | Restart Con-                   |
| veyor 1                             |                                |
| AP[2] 100 deg/sec                   | Move towards                   |
| pallet (3 <sup>rd</sup> pos. point) |                                |
| PALLETIZE ( 3 3 2)                  | Set palletize parameters       |
| LP [3] 100 cm/sec                   | Move closer to                 |
| pallet (4 <sup>th</sup> pos. point) |                                |
| GRIPPER Open                        | Release workpiece              |
| WAIT 4 seconds                      | Wait for                       |
| workpiece to settle                 |                                |
| PAL_END                             | Close palletize                |
| parameters                          |                                |
| IF R[0] <= R[2] THEN                | Compare contents of two        |
| registers                           |                                |
| JMP_LBL 1                           | Jump to LBL 1                  |

The PALLETIZE command sets the palletizing routine parameters, which in this case are 3, 3, 2. Position Register P[0] has been pre-loaded with the first stacking point (1, 1, 1). Register R[0] has been pre-loaded with the number 0, and register R[1] has been loaded with the number 1. R[0] is the reference, or starting number for the palletize count operation, while register R[1] is the amount that the count increments each time the command is executed. Register R[2] contains the number of workpieces to be stacked by the palletizing operation. In this example, R[2] has been preloaded with the number 18.

Registers R[0] and R[1] are used by the IF instruction  $R[0] = R[0] + R[1]$ . This instruction adds the contents of register R[0] to the contents of register R[1] and stores the result in register R[0]. Since register R[1] contains the number 1, each time the loop executes register R[0] increases by 1.

The second IF command,  $IF R[0] \leq R[2] THEN$ , is used to compare the contents of the incremented register R[0] with the preset value in register R[2]. If the preset value is greater than or equal to the current count, the JMP\_LBL 1 command is executed and the program jumps to the beginning of the programmed routine. If the incremented value should try to exceed the preset value, the program will not loop but will instead come to an end.

## 7. Tutorial

### i) Load and run an existing program

1. Press  key, which displays all programs stored in memory.
2. Select a program on the display screen by clicking on the number button to the left of the program name. A red highlight bar will display the selected item.
3. Press the  key to load the program.
4. Return to main screen , press  key to run program.

### ii) Edit an existing program

1. Press  key, which displays all programs stored in memory.
2. Select a program on the display screen by clicking on the number button to the left of the program name. A red highlight bar will display the selected item.
3. Press the  key to load the program.
4. Select a line of the program to be edited by clicking on the line number box located to the left of the program line.
5. Click on the load  button to load the program line for editing
6. An edit display box should now appear.
7. Change the numeric values in the display box by +20%, and click OK in the display box.
8. Return to main screen , press  key to run program.

## Tutorial (cont'd)

### iii) Write a program.

Objective: Write a program that will pick up a box from Conveyor 1 and move it to Conveyor 2.

1. Press  key, and  key to create new program. Type the word "test" in the dialog box to name the new program. Click "OK".
2. Enter the first line of the program by pressing the  key and selecting the  instruction. This first line of the program will instruct the robot to begin from its Home position.

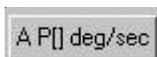
3. Add a program line to start Conveyor 1:  

Note that the default setting for Conveyor 1 is in the forward (fwd) position. If it is not in this position, use more mouse to select it.

4. Add a program line to instruct the robot to wait 18 seconds before moving towards box  

Note. This Wait time is required to allow the box to travel from the chute to the stop point on the conveyor.

5. Move gripper towards box in angular direction until it is above the box (1st pos. point)

  Enter the number "0" *Hint: Use A1+ key to swing the arm in an angular direction towards Conveyor 1.*

## Tutorial (cont'd)

6. Move gripper down to box:   Enter the number “1” (2nd pos. point)

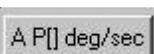
7. Close the gripper to grasp box:   Select “Close”

8. Move gripper to point directly above box

  Enter the number “0”

Note. This “retract” point for the box is the same as the “entry” point.

9. Move gripper towards Conveyor 2 until it is at a point above the conveyor (3rd position point):

  2

10. Move gripper down towards Conveyor 2 (4th pos. point)

  3

11. Open gripper to release box:   Select “Open”

12. Move gripper up from Conveyor 2:   2

13. Return to Home position:  

Now that the program code has been written. The next steps involve setting the position points by moving the robot using the “jog” keys.

## Tutorial (cont'd)

Now that the program code has been written. The next steps involve setting the position points by moving the robot using the “jog” keys.

Click on the Main key  to return to the Main Screen. Switch on Conveyor 1  to send a box into the work envelope. Verify that the robot is in the Home position by clicking on the Home button .

Wait until the box comes to a stop on Conveyor 1. Press and hold the A1+ key to rotate the robot arm towards Conveyor 1. Use the various camera angles to verify that the end effector is directly above the box on Conveyor 1. Once you have determined that the gripper is above the box, press the Save Position key , to store the 1st position point.

Move the gripper directly down towards the box using the Y-axis key . Use the various camera angles to verify that the gripper fingers are properly positioned to pick up the box. *Hint: The exact location where the box can be picked up has an X coordinate of -40 and a Z coordinate of 155.* Click on the Save Position key  to store the 2nd position point.

Close the gripper to pick up the box using the  instruction.

Press the  key to lift the box up from the conveyor. Move the box to the 3rd position point by pressing and holding the A1+ key. Once the box is positioned directly over the conveyor, press the  key to save the 3rd point.

## ***Tutorial (cont'd)***

Lower the gripper towards Conveyor 2 by pressing the  key.

Press the  key to drop the box onto Conveyor 2. Press the

 key to move the box out of the work envelope. Press the

 key to return to the home position.

Verify that the four position points have been entered by clicking on the  key to display the position registers. Position Registers P0 - P3 should all have data.

Click on the MAIN key to return to the Main Screen. Verify that the program named "Test" is shown in the Program Title, and press the Run key  to start the program.