Application Environment

UMI-R3-131

Application Environment

Revision		
Number	**History**	**Date**
	Earlier releases as UMI-R3-102 and UMI-R3-130	
001	First release as UMI-R3-131	99-05

Contents

CHAPTER 1

Introduction

This guide describes:

- basic components of robot systems
- peripheral components of robot systems
- robot space and motion
- safe robot use.

CHAPTER 2

Basic Robot Components

The basic system consists of the following robot components:

- robot
- controller
- computer running Robcomm3 software

Robot

The robot transports payloads and performs other motion tasks in space. A robot arm consists of a set of links and joints. Each joint moves under the control of a motor which is controlled by the system controller.

CRS manufactures the following types of robots:

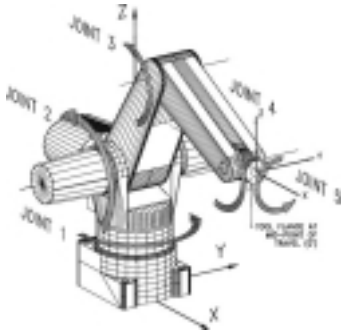
- Articulated Robots (A255, A465, and F3)
- Track Robots (T265, T475, and F3t)

Articulated Robots

An articulated arm has joints for rotary motion. The joints are connected by links. At one end, there is a base attached to a platform, and at the other end, a mechanical interface for a gripper or other tool.

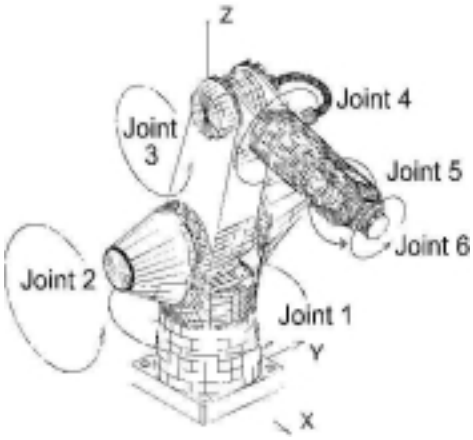
CRS manufactures three articulated robots: A255, A465, and F3.

A255



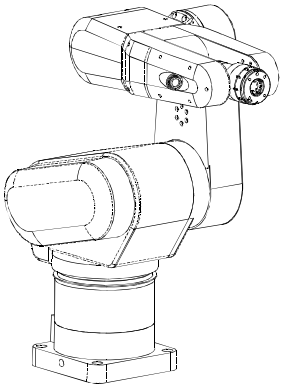
joints/degrees of freedom	5
reach	559 mm.
nominal payload	1.0 kg.

A465



joints/degrees of freedom	6
reach	710 mm.
nominal payload	2.0 kg.

F3



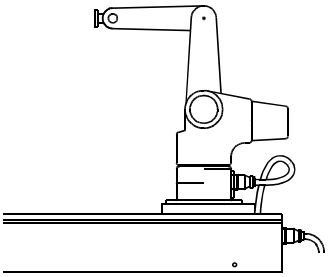
joints/degrees of freedom	6
reach	710 mm.
nominal payload	3.0 kg.

Track Robots

A track robot is an articulated robot mounted on an extra linear axis. The extra linear joint (or degree of freedom) is used to transport the robot over the length of the track. A track robot has an increased workspace compared to a robot without a track.

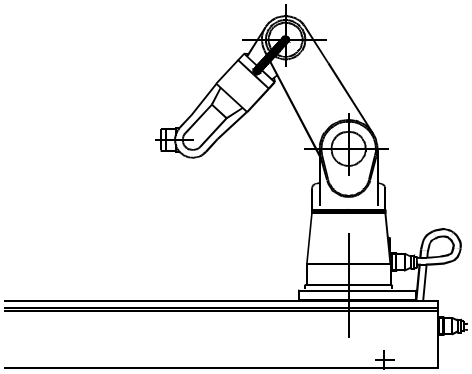
CRS manufactures three track robots: T265, T475, and F3t. Each track robot can be ordered with track lengths of 1, 2, 3, 4, or 5 meters.

T265



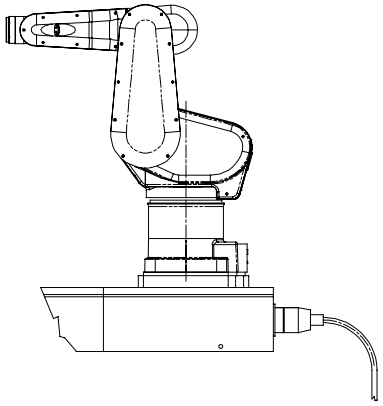
joints/degrees of freedom	6
max reach (5 m track)	6.12 m.
nominal payload	1.0 kg.

T475



joints/degrees of freedom	7
max reach (5 m track)	6.42 m.
nominal payload	2.0 kg.

F3t



joints/degrees of freedom	7
max reach (5 m track)	6.42 m.
nominal payload	3.0 kg.

Controller

The controller is a computerized electronics unit whose primary function is to provide power and control for the robot. It also contains electronics to control other devices. The controller provides control via CRS Robotics Robot Operating System (CROS), which runs RAPL-3 programs.

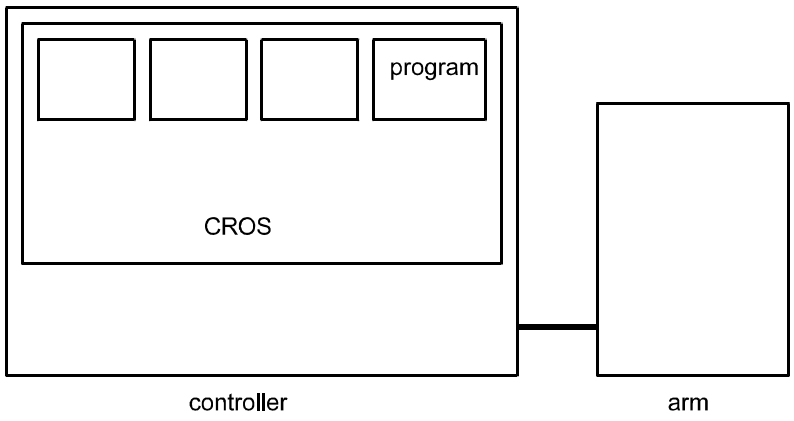
The Operating System

Basic Functions

The controller is the hardware platform for the robot operating system (CROS). CROS manages the resources of the controller, controls the execution of programs, and supports all robot system operations including running the core robot components and interfacing to any peripheral robot components.

RAPL-3 Programs

CROS is the platform for running compiled RAPL-3 programs, which contain step-by-step instructions. You create the program using the Robcomm3 editor window. After you successfully compile the program, you must transfer it to the controller.



To run a RAPL-3 program, access CROS through the terminal, the teach pendant, or the front panel. For details on accessing the terminal and then using the system shell or the application shell, see the *Robcomm3* section and then the sections on *CROS and the System Shell* or *Application Shell* in this *Application Development Guide*. For details on the teach pendant, see the *Teach Pendant* section in this *Application Development Guide*. For details on the front panel and pendant programming commands, see the *Front Panel* section in this *Application Development Guide* and the *RAPL-3 Language Reference Guide*.

Multi-Process Capability

Within CROS, the basic unit of activity is termed a process. A program, when it is running, is a process. CROS can run several processes at the same time. When you run an application, you use one or more processes. Each process can start new processes and other tasks within the operating system. Some processes send signals to the robot arm to initiate motion.

Your application can take advantage of the multi-process capability of CROS. In addition to a main process, other processes are used to perform other functions, such as monitoring inputs and controlling other devices. Different processes can move the arm; however, only one process can have control of the robot at any one time. Control of the robot can be given from one process to another by transferring point of control.

Standard Processes

CROS on a C500 controller has been designed with several standard modules that have specific purposes. These modules include:

- System Shell
- Application Shell

System Shell

The system shell is a command line interpreter (like DOS) for CROS. It interprets your command words into a form that CROS can understand.

You use the system shell to manage processes of the operating system, modify files and directories, configure options, and troubleshoot problems. You can also run robot applications from the system shell.

Application Shell (ash)

The application shell (ash) is an interactive command line interpreter with a database used to develop applications.

With ash you can move the arm, teach locations, modify other variables, and run your applications. The application shell supports a database for locations and other variables.

Front Panel

Although the front panel is physically a part of the controller, it is one of several points where you can control the robot system. The front panel has five switches for operator input. You can write the program for your application to start after an input from the front panel switch. In other words, you can use the front panel switch to start an application.

Serial Port

The controller has a serial port, located at the front of the controller, for connecting to the computer.

Input/Output

The back of the controller has two connectors to control and monitor other devices: General Purpose I/O (GPIO) and System I/O (SYSIO).

GPIO

The GPIO connector has a total of 16 inputs and 16 outputs available to connect to external hardware. The inputs can be monitored by the user application. Likewise, the application can also control devices in the work cell by connecting to the outputs.

CROS can interface with simple digital devices, receiving signals from sensors or sending signals to actuators. The device can be a PLC (programmable logic controller) which controls several devices as a subsystem and communicates with CROS. A PLC receives inputs from sensors, makes decisions according to pre-programmed logic, and sends outputs, usually to simple electrical or mechanical devices such as motors, switches, valves, etc.

SYSIO

The SYSIO port supports remote front panels and e-stop devices. It provides access to the front panel circuits (push-button inputs and the LED indicators) and contains an e-stop contact pair.

Robcomm3 Software

Robcomm3 is application software designed for writing and compiling robot application programs in RAPL-3 (Robot Application Programming Language). Robcomm3 also has a terminal emulation feature to facilitate communication between the computer and the controller. To run Robcomm3, your computer should have the following minimum specifications.

CPU	486 at 33 MHz
RAM	8 Mbytes
operating system	Windows 3.1/Windows 95/WindowsNT
screen resolution	800 x 600, SuperVGA, 256 colors
serial port	Standard RS-232

Robcomm3 provides:

- tools to create, edit, and compile RAPL-3 programs
- a utility to transfer files between the computer and the controller
- a terminal to access the controller.

Programs and the Compiler

To write and compile a RAPL-3 program, use an editor window in Robcomm3. You can write and compile a RAPL-3 program using only Robcomm3 on the computer. You do not need to be connected to the controller to write or compile programs.

Use the editor to write a program. Use the compiler to translate that source code to object code. When a program has compiled successfully, you must transfer the object file to the controller in order to run the application.

File Transfer

After you compile the program, send it to the controller. Robcomm3 sends the file from the computer's file system to the controller. Fastacid, a process of CROS, receives the file and places it in the CROS file system.

The process of transferring the program to the controller is automated if you setup an application using the App Setup feature in Robcomm3. When you setup an application, prepare and send the application by using the Robcomm3 Send command.

Terminal Access to the Controller

Robcomm3's terminal window provides access between your computer's keyboard and monitor and the system and application shells on the controller. You can interact with either of the two shells.

Additional Information

Additional information about basic robot components can be found in these places.

Component	Model	Publication
articulated robot	F3	*F3 Robot Arm User Guide*
	A465	*A465 Robot Arm User Guide*
	A255	*A255 Robot Arm User Guide*
track robot	F3t, T475, T265	*Track User Guide*
controller	C500C (with all robots)	*C500C Controller User Guide*
	C500B (with F3/F3t)	*F3 Robot System User's Guide*
	C500 (with A and T series robots)	*C500 Controller User's Guide*
teach pendant		This *Application Development Guide*, reference section titled *Teach Pendant*
Robcomm3		This *Application Development Guide*, reference section titled *Robcomm3*
application shell		This *Application Development Guide*, reference section titled *Application Shell*
system shell		This *Application Development Guide*, reference section titled *CROS and the System Shell*
CROS		This *Application Development Guide*, reference section titled *CROS and the System Shell*
RAPL-3		*RAPL-3 Language Reference Guide*
gripper	servo	*Servo Gripper Option User's Guide*

CHAPTER 3

Peripheral Components

In addition to basic robot components, your system may comprise one or more of the following peripheral components:

- teach pendant
- gripper or other end-of-arm tool
- extra axes
- external devices.

Teach Pendant

The teach pendant is a hand-held terminal for controlling the robot remotely. With the teach pendant, you can move the robot arm, teach locations and other variables, and select and run applications.

During the development of an application, the teach pendant is used primarily to move the arm. It can also be used for teaching locations, modifying variables, and running programs.

The teach pendant provides robot motion capabilities that exceed what is available through the application shell (ash). This includes the teach pendant's capability to move the arm as long as a key is pressed and stop when that key is released. The pendant has an e-stop and a live-man switch. The pendant can be used by the operator near the location. For these reasons, the teach pendant tool is recommended for teaching locations.

It is not necessary to have the teach pendant connected to run an application. Once an application is developed, the teach pendant can be disconnected from the controller and the application run from the system shell, the application shell, or the front panel.

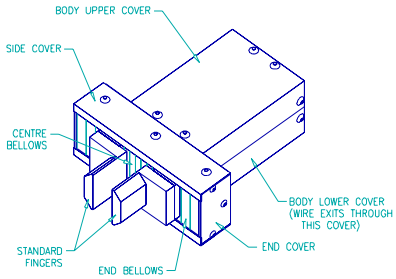
If you have a POLARA-based lab system, your ability to use the teach pendant is limited.

Gripper

The end-of-arm tool is attached to the free end of the robot arm. Often the end-of-arm tool is a gripper, but it can be a dispenser, spray head, buffing wheel, or other tool.

CRS has three standard grippers. The servo grippers are powered by servo controlled electric motors and the pnuematic gripper is powered by pressurized air.

Servo Gripper



Standard Fingers

finger opening range	0 to 50 mm.
finger length	25 mm.

Servo Gripper

Microplate Fingers

finger opening range	82 to 132 mm.
finger length	70 mm.

Pneumatic Gripper

finger opening range	20 degrees
finger length	76 mm.

Extra Axes

An axis, like a joint of an arm, is a place where motion occurs. A system can have additional axes. The motion of an additional axis is controlled by the controller which sends signals to the motor and receives a feedback signal from the encoder on the axis.

A track is the most common extra axis.

External Devices

A system may contain other devices such as a conveyor or elevator in an industrial application, or a pipettor, incubator, fridge, washer, or pump in a laboratory application. These devices can be controlled by the operating system, CROS. Features of the RAPL-3 programming language allow you to write control of the external devices or external applications into your robot application programs.

The system may have additional inputs from sensors monitoring the status of tasks, motion, or external devices within the system. Sensors may also be monitoring the interface between the robot system and the larger industrial or laboratory process, for example, when a part is delivered to the robot work cell or when a machine is ready to accept work from the robot work cell.

CHAPTER 4

Robot Space and Motion

This chapter describes:

- arm space, using coordinate systems
- locations in the arm space
- simple arm motions in space
- arm motions available from different application development tools
- programming for different robot tools
- programming for different positions of the base
- manual positioning of the arm.

Overview

This overview summarizes topics described in detail later in this chapter.

Describing Space

A coordinate system is a way to describe the space around the arm. CRS robots can use any of four coordinate systems: world, joint, cylindrical, and tool. Coordinates of any of these systems can describe any point in robot space.

Coordinate System
World
Joint
Cylindrical
Tool

Identifying Locations

A location is a specific point in space that is stored by the robot for use in an application. There are two types of locations: cloc (cartesian location) that stores coordinates based on the world coordinate system, and ploc (precision location) that stores coordinates based on the joint coordinate system.

Coordinate System	Location Type
World	cloc (**c**artesian **loc**ation)
Joint	ploc (**p**recision **loc**ation)

Robot Motion

The robot arm can move relative to the coordinate system along a particular axis. The robot motion can be continuous or incremental. The types of arm motion available also depends on the robot tool you are using. As an example, you can move the robot using the cylindrical coordinate system using the teach pendant but not ash.

Straight Line Motion

One type of motion is straight line motion, in which several joint's motions are synchronized so that the tool axis moves along a straight line. In straight line motion, the motion of each robot axis is coordinated so that the end effector moves in a straight line. You can move in a straight line in from any position along specified world or tool axis. You can also move in a straight line to a specified variable if the variable is a cloc (cartesian location). You cannot move in straight line mode to a precision location.

Straight line movements are useful when you must maintain the level or orientation of the payload. For example, if you are moving liquids you could use straight line movements to prevent the liquid from spilling.

Moving Along or Around Axes

You can move the tool along most axes of coordinate systems. For straight motion, a number of joints rotate at the same time to move the tool in a straight line along the axis.

Coordinate System	Axis	Motion Availability and Description
World	X	straight motion (positive or negative)
	Y	straight motion (positive or negative)
	Z	straight motion (positive or negative)
Joint	any arm joint	rotation (positive or negative)
Cylindrical	θ	rotation (positive or negative)
	R	straight motion (positive or negative)
	Z	straight motion (positive or negative)
Tool	X	straight motion (positive or negative)
	Y	straight motion (positive or negative)
	Z	straight motion (positive or negative)

Note: In the world and tool coordinate systems, you can move in straight line motion. Straight line motion is not available in the joint and cylindrical coordinates systems.

Moving Continuously or Incrementally

Some motions, along or around an axis, are available in a continuous or incremental type of motion. Some of these are available only using certain robot system tools, such as ash, or the teach pendant. The following table highlights motion types available with specific system tools.

Coordinate System	Motion	Tool		
		Teach Pendant	Application Shell	RAPL-3 Program
World	continuous motion along axis	velocity	[no equivalent]	[no equivalent]
	incremental motion along axis	jog	wx, wy, wz	jog()
	alignment to specified axis	align	align	align()
	straight line motion parallel to an axis	[no equivalent]	wxs, wys, wzs	wxs(), wys(), wzs()
Joint	continuous motion around axis	velocity	[no equivalent]	[no equivalent]
	incremental motion (deg.) around axis	jog	joint	joint()
	incremental motion (pulses) around axis	[not available]	motor	motor()
Cylindrical	continuous motion along axis	velocity	[no equivalent]	[no equivalent]
Tool	continuous motion parallel to tool axis	velocity	[no equivalent]	[no equivalent]
	specified distance motion parallel to a tool axis	[no equivalent]	tx, ty, tz depart	jog_t() tx,(), ty(), tz() depart
	straight line motion parallel to a tool axis	[no equivalent]	txs, tys, tzs departs	jog_ts() txs(), tys(), tzs() departs()

Note: Refer to the teach pendant and application shell sections of this manual for details on the teach pendant and ash motion features. For RAPL-3 motion command details see the *RAPL-3 Language Reference Guide*.

Moving the Arm When Limp

Robot motion is controlled by servo motors for all motion initiated from the robot controller or robot system tools. However, you can disengage servo motor control of the robot joints with the limp command. When limp, the

robot joints can be moved by hand. Individual joints can be limped, or all of the joints can be limped at once.

Limp motion is similar to joint motion; however, in joint motion the servo motor moves the joint, and in limp motion the joint movement is human or other external force. The motion however can be considered to be joint motion.

When an axis is limp, the encoders still supply feedback to the controller.

Warning! Caution must be used when limping joints, because a limped joint will fall due to gravity or inertia, resulting in robot collisions which can damage the robot or other equipment.

Moving the Tool Center Point

When moving along an axis or to a specified location, the robot moves the tool center point (TCP). To ensure that the TCP is the point that you want, such as the tip of a dispenser or the point halfway between two gripper fingers, you describe the tool's size and orientation to the software with a tool transform.

Tool Transform

A tool transform informs the controller of the position of the tool (tool center point -TCP). Without a tool transform, the controller moves the arm as if the TCP is the center point of the tool flange surface. A tool transform is the measurements in the tool coordinate system of the mounted tool's TCP. The tool transform also includes the yaw, pitch, and roll coordinates which define the tool's orientation.

Base Offset

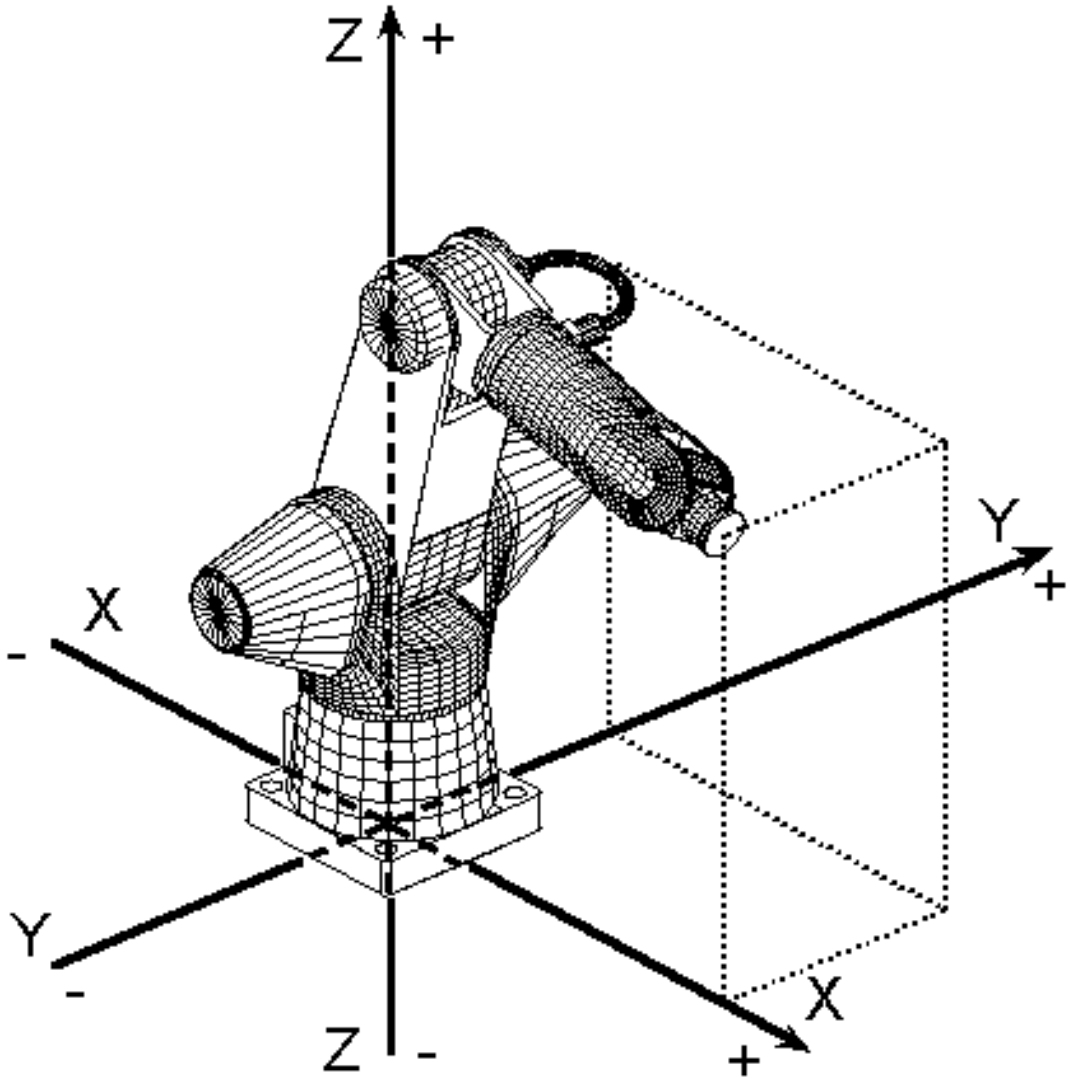
The origin of the world coordinate system is the center of the base of the robot. In some cases, for instance if the robot is to be hung suspended above the workspace, you may need to change this origin. To do so, set a base offset. A base offset is a set of coordinate values which re-defines the new origin. Refer to the *Application Development* section of this *Application Development Guide* for the details on setting a tool transform.

Cartesian Space, Locations, and Motion

One way to describe the arm workspace is the world coordinate system. With this system, points in space are identified by cartesian locations, and motion parallel to any axis is available from the teach pendant, the application shell and RAPL-3 programs.

World Coordinate System

The world coordinate system is based on an axis system with three axes: X, Y, and Z at right angles to each other which intersect at the origin. The origin is the center of the robot mounting flange when a base offset is not set. The following figure shows the axis orientation. The Z axis is vertical with positive Z up. The X and Y axes are horizontal, with positive X forward away from the front of the arm and positive Y to the side as shown. The relationship of X, Y, and Z follows the right-hand rule of thumb, index finger, and middle finger with your palm facing upwards.

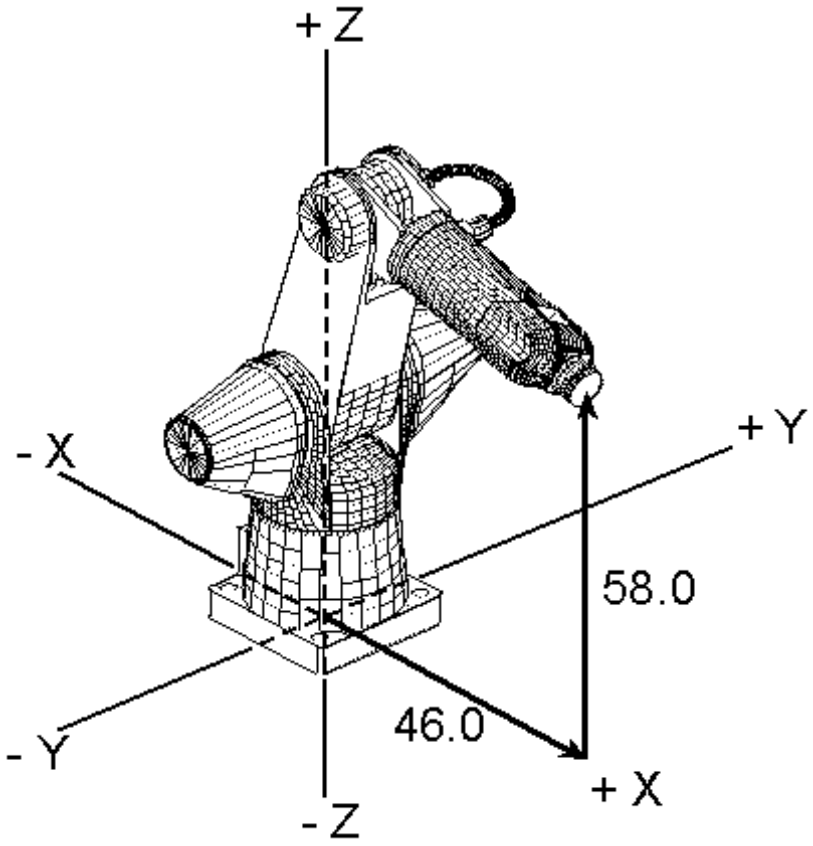


World coordinates can describe any point in the world coordinate system. To store world coordinate data, the software uses cartesian locations. Coordinates contain data about position and orientation.

Position

The position of a point in the workspace is identified by distances (positive or negative) along the X, Y, and Z axes from the origin.

For example, in the diagram with the A465, the center of the mechanical interface can be described as 46.0 cm. in a positive X direction, 0.0 in a Y direction, and 58.0 in a positive Z direction, or (46.0, 0.0, 58.0).



Orientation

When your tool is at a point in space, it could be oriented in different ways. For example, it could be pointing downward, parallel to the negative Z axis, or pointing forward, parallel to the positive X axis.

The orientation of a tool is identified by a rotation (positive or negative) around the X, Y, and Z axes (or around axes that are parallel to the X, Y, and Z axes). Rotation around the X axis is called roll, around the Y axis, pitch, and around the Z axis, yaw. By convention, these are written in the order: yaw, pitch, roll.

For example, in the diagram with the A465, the mechanical interface is oriented with no yaw, a 15.0 degree pitch, and a –35.0 degree roll, or (0.0, 15.0, -35.0).

Motion around an axis follows the right-hand rule. When your thumb is pointing in the positive direction of the axis and your fingers are curled into your palm, your fingers are pointing in the positive direction of rotation.

Full Coordinates

The full coordinates for a tool point are written in the order: X, Y, Z, yaw, pitch, roll. The example is (46.0, 0.0, 58.0, 0.0, 15.0, -35.0).

Note: A tool on an A255 with five degrees of freedom has five coordinates: X, Y, Z, pitch, and roll. It does not have yaw.

Cartesian Locations

In RAPL-3, a cartesian location, a cloc, represents a point in the arm workspace defined by cartesian-style world coordinates.

The data in a cloc correspond to dimensions in the workspace and are independent of arm type. The location is the same whether accessed by an F3 or an A465.

The data are also independent of robot pose. The location might be accessible by the arm in different poses. In other words, a cloc location variable does not necessarily define unique robot axis positions.

Motion with World Coordinates

You can move the tool center point (TCP) parallel to the world X, Y, and Z. You can also align the axis of the tool to a world axis.

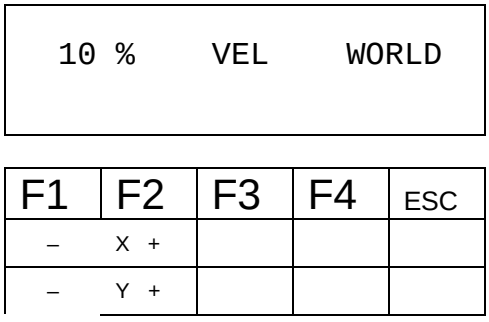
Note: The axis of the tool is defined as the axis perpendicular to the plane of the mechanical interface pointing away from the interface. For the A series robots, the tool axis is by definition the tool X axis. For the F3, the tool axis is defined as the tool Z axis. Refer to the *Tool Space and Motion* section.

Motion	Tool		
	Teach Pendant	Application Shell	RAPL-3 Program
continuous motion along axis	velocity	[no equivalent]	[no equivalent]
incremental motion along axis	jog	wx, wy, wz, xrot, yrot, zrot	jog()
straight line motion	[not available]	wxs, wys, wzs, xrots, yrots, zrots	zrots(), yrots(), xrots() jog_ws()
alignment to axis	align	align	align()

Continuous Motion

You can move the TCP by continuous motion along any axis of the world coordinate system, but only with the teach pendant.

With the teach pendant, you select Vel (velocity) type of motion and World mode. You set the speed with the Speed Up and Speed Down keys. Pressing a coordinate key (X, Y, Z, yaw, pitch, or roll, positive or negative) moves the TCP continuously as long as the key is pressed.



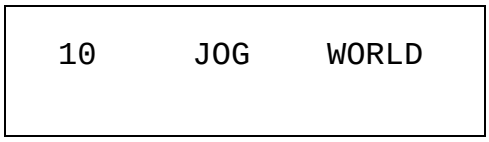
–	Z +	SPEED DOWN	SPEED UP	
–	yaw +			
–	pitch +			
–	roll +			

Incremental Motion

You can move the TCP by specified increments along any axis of the world coordinate system with the teach pendant, ash, or a RAPL-3 program.

Teach Pendant: Jog

With the teach pendant, select Jog type of motion and World mode. You set the incremental distance using the multi-purpose Speed Up and Speed Down keys. One press of a coordinate key moves the TCP one increment.



RAPL-3 Program: jog_w()

In a RAPL-3 program, you use the jog_w() command.

You specify the axis with a string of characters and the direction and incremental distance with a positive or negative number.

Editor

```
jog(JOG_X,-20)
```

You cannot rotate the TCP in yaw, pitch, or roll with the jog() command. Instead you can use the zrot(), yrot(), or xrot() commands. Refer to the *RAPL-3 Language Reference Guide*.

Straight Line Motion

You can move the robot in the world coordinate system with incremental straight line movements, so that the orientation of the end effector is unchanged. The straight line motion options are available only from ash and RAPL-3.

RAPL-3 Program: jog_ws()

In a RAPL-3 program, use the jog_ws() command.

You specify the axis with a string of characters and the direction and incremental distance with a positive or negative number.

Editor

```
jog(JOG_X,-20)
```

You cannot rotate the TCP in yaw, pitch, or roll with the jog() command. Instead you can use the zrots(), yrots(), or xrots() commands. Refer to the *RAPL-3 Language Reference Guide*.

Application Shell: wxs

From the application shell, you can move the robot in straight line movement parallel to a world axis with the wxs, wys, wzs, xrots, yrots, and zrots commands. Each command requires only a distance parameter.

Terminal

```
application > wxs 10
```

For example, wxs 10 moves the end effector in a straight line 10 (in. or mm.) parallel to the world X axis. The units depends on the configuration.

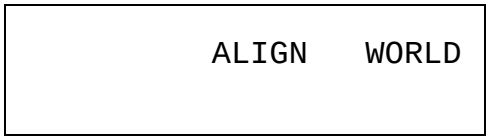
Alignment

You can align the tool axis parallel to any axis of the world coordinate system with the teach pendant, ash, or a RAPL-3 program.

The tool axis is usually straight out from the tool center point, but is described in detail in Tool Coordinate System and Tool Transform.

Teach Pendant: Align

With the teach pendant, you select Align type of motion and World mode.



Pressing a coordinate key, X, Y, or Z, either positive or negative, aligns the tool axis parallel to the positive or negative X, Y, or Z axis.

Application Shell: align

From the application shell, you can use the align command to set the tool axis parallel to one of the world X,Y, or Z axes, either negative or positive direction. To align to the negative Z axis enter:

Terminal

```
application > align -3
```

To align with the closest world axis:

Terminal

```
application >     align n
```

RAPL-3 Program: align()

In a RAPL-3 program, use the align() command.

Specify the axis with a string which begins with a positive or negative sign. You also specify the speed for aligning with a number.

Editor

```
      align(10,-ALIGN_Z)
```

You can align to whichever axis is nearest the current orientation.

Editor

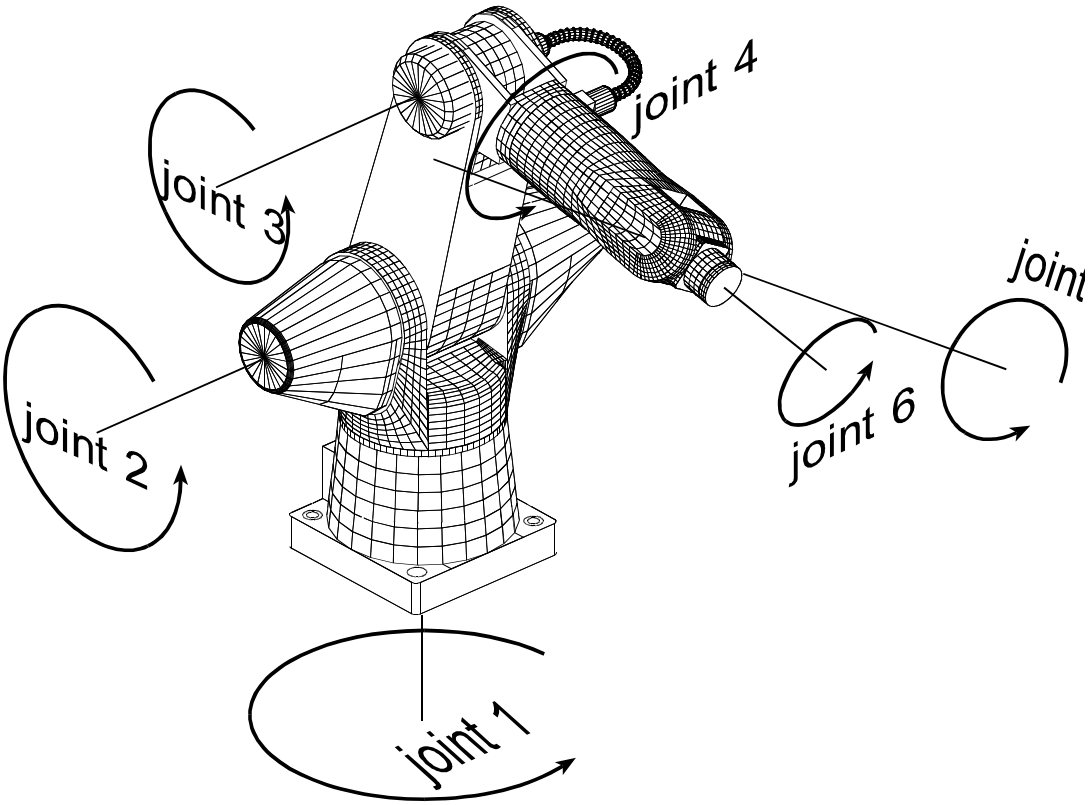
```
align(20,ALIGN_NEAR)
```

Rotational Space, Locations, and Motion

A second way to describe the arm workspace is the joint coordinate system. With this system, points are identified by precision locations and motion around a joint axis available from the teach pendant, the application shell, and RAPL-3 programs.

Joint Coordinate System

The joint coordinate system is a rotational coordinate system. It is based on rotation around each joint axis.



Axes

There are as many axes as joints. Each axis passes through the joint and is the center of rotation of that joint.

Direction of Rotation

For joint 1, positive rotation corresponds to rotation around the positive Z world axis. For joints 2, 3, and 5, positive rotation is rotation that lifts the mechanical interface up and away from the base when the end of the arm is in front of the base. For joints 4 and 6, positive rotation corresponds to rotation around the positive Z world axis if the arm is positioned straight up. This is the same as rotation around the positive X world axis if the outer arm is positioned horizontally in front of the rest of the arm.

Coordinates and Locations

Joint coordinates can describe any point in the arm workspace. To store joint coordinate data, the software uses precision locations.

Precision Pulses

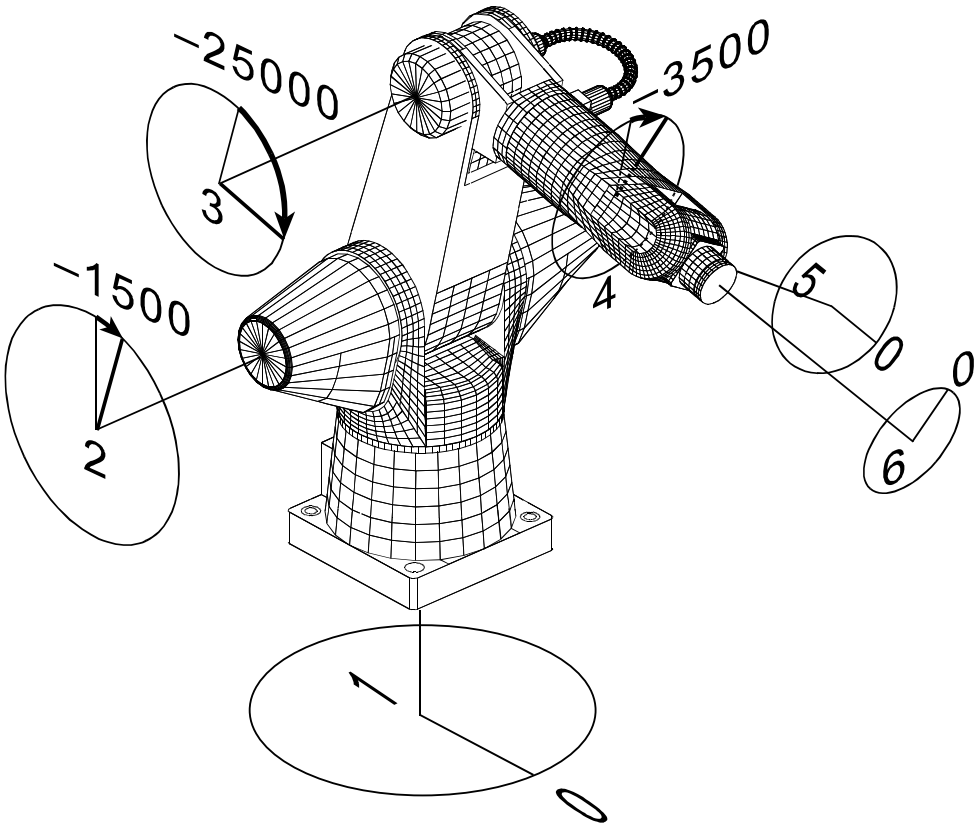
Joint coordinates are based on precision pulses.

Each joint contains an encoder that generates pulses as it rotates (about 200 pulses for each degree of rotation for most non-wrist joints). Any position of the arm can be defined by the number of precision pulses away from zero, for each joint.

Zero is set at the factory with each joint at a certain position. For example, for joint 1, zero is set with the arm facing forward. Pulse counts for joint 1 can range from +48611 to –48611 (all robots).

Joint Coordinates

Any point in the workspace and any tool orientation can be identified by these pulse counts of the joints of the arm.



In the diagram with the A465, the center of the mechanical interface can be described as (0, –1500, –25000, –3500, 0, 0).

Precision Locations

In RAPL-3, a precision location, a ploc, represents a point in the arm workspace defined by precision pulse counts.

The data in a ploc depend on the arm. With different gear ratios in different models of arms, the same ploc places one model of arm at one point in the workspace and another model of arm at another point.

Motion with Joint Coordinates

You can move the tool center point (TCP) according to joint axes. With the tools available, you move one joint at a time. You cannot use straight line motion with motion based on the joint coordinates.

Motion	Tool		
	Teach Pendant	Application Shell	RAPL-3 Program
continuous motion around axis	velocity	[no equivalent]	[no equivalent]
incremental motion (in degrees) around axis	jog	joint	joint()
incremental motion (in motor pulses) around axis	[not available]	motor	motor()
straight line motion	[not available]	[not available]	[not available]

Continuous Motion

You can move the TCP by continuous motion around any axis of the joint coordinate system, but only with the teach pendant.

Teach Pendant: Vel

With the teach pendant, you select Vel (velocity) type of motion and Joint mode. You set the speed with the Speed Up and Speed Down keys. Pressing an axis key (Ax1, Ax2, Ax3, Ax4, Ax5, or Ax6, positive or negative) moves the TCP continuously as long as the key is pressed.

F1	F2	F3	F4	ESC
– Ax1	+			
– Ax2	+			
– Ax3	+	SPEED DOWN	SPEED UP	
– Ax4	+			
– Ax5	+			
– Ax6	+			

Pressing an axis key rotates the TCP around the axis, in a positive or negative direction.

Incremental Motion by Degrees

You can move a joint by specified degrees with the teach pendant, the application shell, and a RAPL-3 program.

Teach Pendant: Jog

With the teach pendant, you select Jog type of motion and Joint mode. You set the rotational increment of degrees using the multi-purpose Speed Up and Speed Down keys. One press of an axis key moves TCP one increment.

```
  0.1 deg  JOG    JOINT
```

Application Shell: joint

With the application shell, use the joint command.

Specify the axis with a number and specify the direction and amount of rotation with a signed number.

Terminal

```
application>joint 2, –45
```

RAPL-3 Program: joint()

In a RAPL-3 program, use the joint() command.

Specify the axis with a number and specify the direction and amount of rotation with a signed number.

Editor

```
      joint(1,-45)
```

Incremental Motion by Pulses

You can move a joint by a specified number of encoder pulses with the application shell and a RAPL-3 program, but not with the teach pendant.

Application Shell: motor

With the application shell, use the motor command.

Specify the axis with a number and specify the direction and number of encoder pulses with a signed number.

Terminal

```
application>motor 3, –2500
```

RAPL-3 Program: motor()

In a RAPL-3 program, use the motor() command.

Specify the axis with a number and specify the direction and number of encoder pulses with a signed number.

Editor

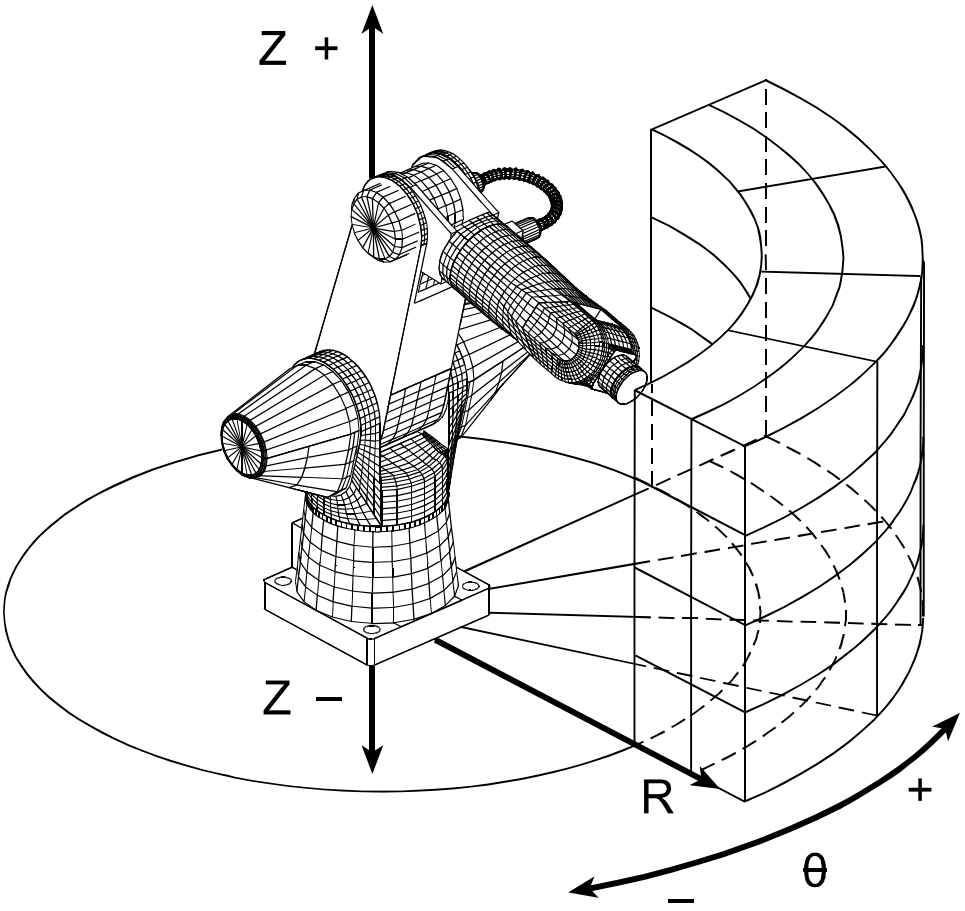
```
      motor(3,-2500)
```

Cylindrical Space and Motion

A third way of describing space is the cylindrical coordinate system. Motion in the cylindrical system is only available using the teach pendant. The application shell and the RAPL-3 language do not support cylindrical locations.

Cylindrical Coordinate System

The cylindrical coordinate system is based on one vertical axis, Z. The system is defined by: 1) a rotation around the Z axis, 2) a distance away from the Z axis, and 3) a distance (height) along the axis.



Rotation: θ

Rotation is around the Z axis. For rotation, the origin (zero degrees) is forward, in front of the arm. Rotation follows the right-hand rule: with your thumb pointing in the positive direction of the Z axis, your curled fingers point in the positive direction of rotation. Rotation is represented by the Greek letter theta, θ.

Radius: R

Radius is the distance away from the Z axis.

Height: Z

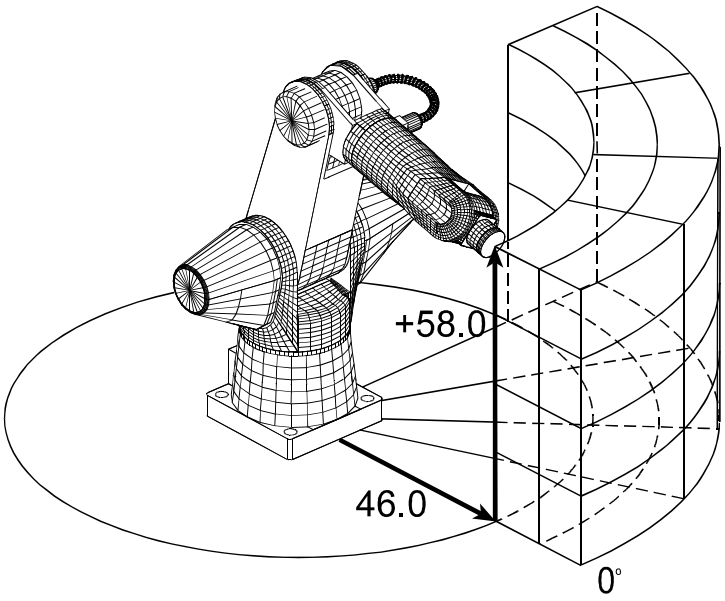
Height is the distance along the Z axis. The origin of the Z axis is the center of the base mounting surface, with positive above and negative below.

Cylindrical Coordinates

Cylindrical coordinates describe the position of a point in the cylindrical coordinate system and the orientation at a point.

Position

The position of a point in the workspace is identified by rotation, radius, and height.

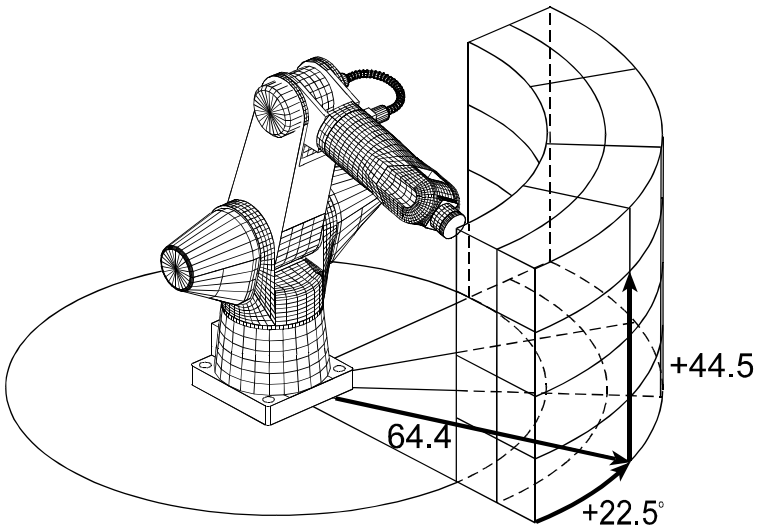


For example, in the diagram with the A465, the centre of the mechanical interface can be described as 0.0 degrees from the origin of rotation, 46.0 in an R direction, and 58.0 in. in a positive Z direction, or (0.0, 46.0, 58.0).

Orientation

At a point in space, your tool could be oriented in different ways. Orientation in the cylindrical coordinate system is described in a way similar to orientation in the world coordinate system.

Orientation is identified by a rotation, positive or negative, around cylindrical axes: R (roll), tangent of arc defined by R and θ (pitch), and Z (yaw), written in the order: yaw, pitch, roll.



For example, in the diagram with the A465, the mechanical interface is oriented with no yaw, a 15.0 degree pitch (positive around the tangent of θ), and a –35.0 degree roll (negative around the R axis), or 0.0, 15.0, -35.0.

Full Coordinates

The full coordinates for a tool point are written in the order: θ, R, Z, yaw, pitch, roll. The example is (0.0, 46.0, 58.0, 0.0, 15.0, -35.0).

Motion with Cylindrical Coordinates

You can move the tool center point (TCP) in cylindrical coordinate system, but only in continuous motion and only with the teach pendant.

Motion	Tool		
	Teach Pendant	Application Shell	RAPL-3 Program
continuous motion	velocity	[no equivalent]	[no equivalent]

With the Teach Pendant: Vel

With the teach pendant, you select Vel (velocity) type of motion and Cyl (cylindrical) mode. You set the speed with the Speed Up and Speed Down keys. Pressing a coordinate key (θ, R, Z, yaw, pitch, or roll, positive or negative) moves the TCP continuously as long as the key is pressed.

```
10 %    VEL    CYL
```

F1	F2	F3	F4	ESC
–	θ +			
–	R +			

–	Z +	SPEED DOWN	SPEED UP	
–	yaw +			
–	pitch +			
–	roll +			

Pressing an axis key moves the TCP in the θ, R, or Z direction, or rotates the TCP around an orientation axis, in a positive or negative direction.

Tool Space and Motion

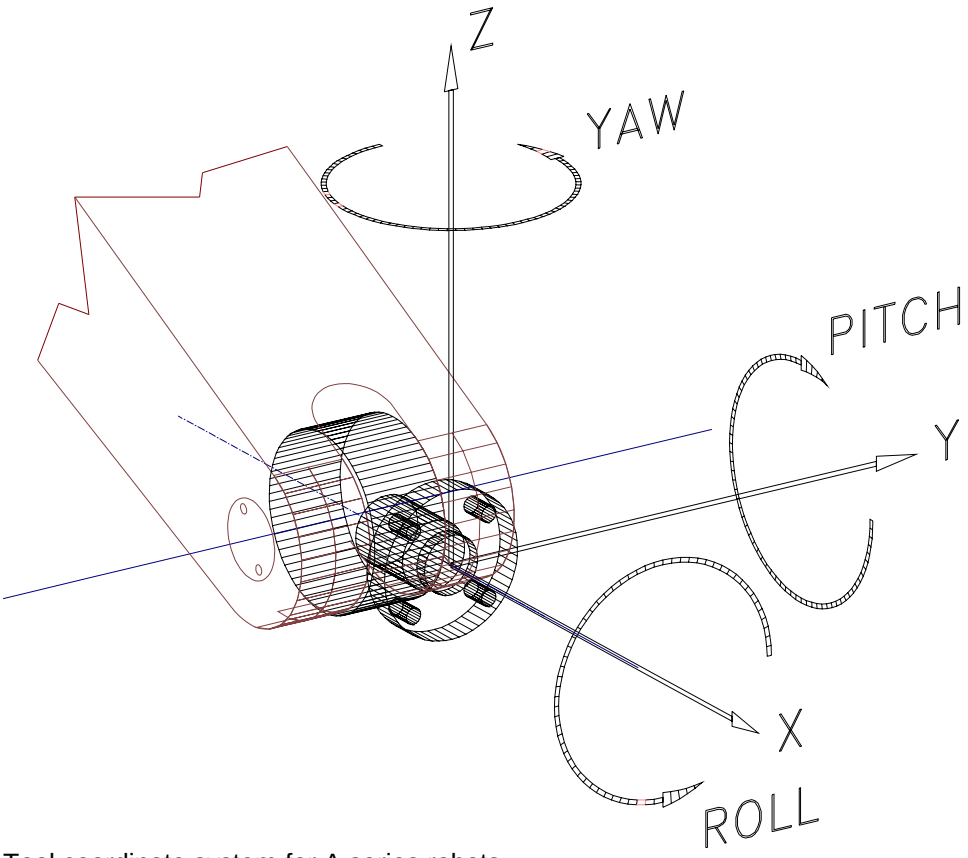
A fourth way of describing space is the tool coordinate system. Motion along one axis is available from the teach pendant, the application shell, and RAPL-3 programs.

Tool Coordinate System

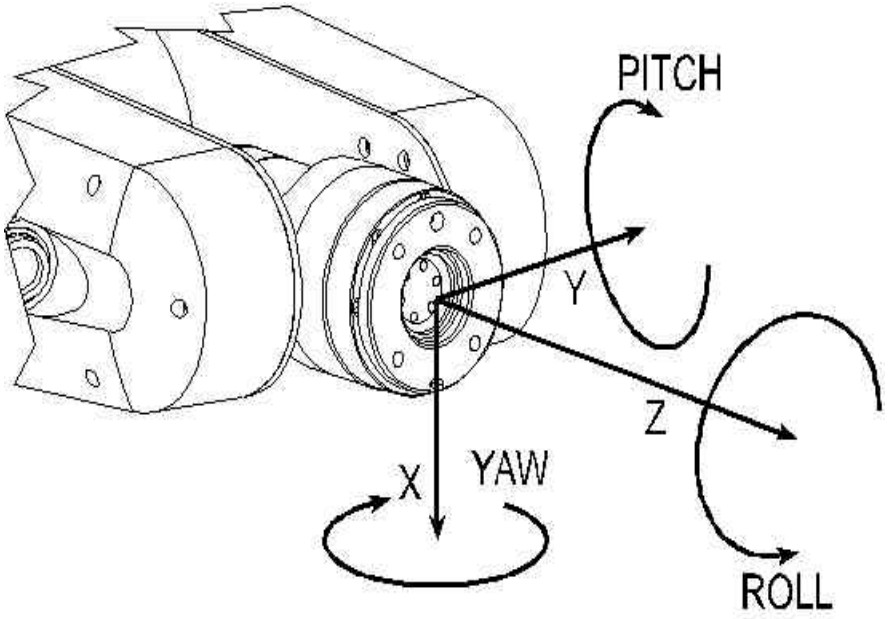
Like the world coordinate system, the tool coordinate system is based on three straight axes (X, Y, and Z) that are at right angles to each other, are related according to the right-hand rule, and intersect at an origin.

The tool coordinate system is dependent on the mechanical interface and is independent of the rest of the arm or work cell. This coordinate system moves with the mechanical interface. The definition of the tool coordinate system differs between the A series and F series robots. The difference is simply in the definition of the axes for the different robots. In A series robots the tool axis is the X axis and for the F3 robot the tool axis is the Z axis.

Note: The tool axis is the axis pointing perpendicular away from the plane of the mechanical interface. The other two axis are mutually perpendicular and parallel to the mechanical interface plane.



Tool coordinate system for A series robots



Tool coordinate system for F3 robot

Origin

The origin is the center of the mechanical interface.

Axes

With an F3, robot the positive Z axis points away from the mechanical interface towards the tool. It is called the tool axis. The X and Y axes are in the plane of the mechanical interface.

With an A, series robot the positive X axis points away from the mechanical interface towards the tool. It is called the tool axis. The X and Y axes are in the plane of the mechanical interface.

Coordinates

Like world coordinates, tool coordinates contain data about position and orientation.

Position

The position of a point is identified by distances (positive or negative) along the X, Y, and Z axes from the origin.

Orientation

The orientation of a tool is identified by a rotation (positive or negative) around the X, Y, and Z axes (or around axes that are parallel to the X, Y, and Z axes), written in the order: yaw (around Z), pitch (around Y), and roll (around X). Rotation around an axis follows the right-hand rule of your fingers pointing in the positive direction of rotation around the axis when your thumb is pointing in the positive direction of the axis and your fingers are curled into your palm.

Coordinate Use

You use tool coordinates to describe the tool to the software. You cannot save a location to a variable name using the tool frame of reference. Location variables can only be saved as cloc (cartesian locations - world frame of reference) or ploc (precision locations - joint frame of reference).

The tool coordinates are used to specify a tool transform, so that the software controlling the robot motion knows the position and orientation of the end effector.

Tool Center Point

The tool center point (TCP) is the important point of the tool, where work occurs. Examples include: the tip of a dispenser, the end of a test probe, or the point halfway between two gripper fingers.

The TCP is the part of the robot that you want to move to your locations when you run your application.

You describe the TCP with coordinates in the tool coordinate system.

Tool Transform

To describe the tool to the software, you set a tool transform. After you set a tool transform, the software modifies the movement of the arm to place the TCP, not the center of the mechanical interface, at the location.

Moving Without a Transform

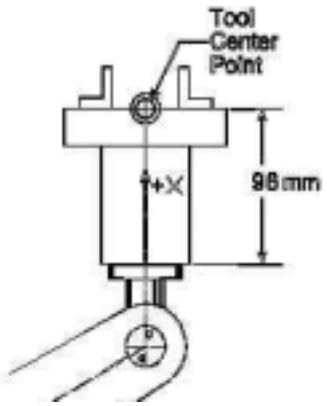
If you do not specify a transform, the TCP is set at its default which is the origin of the tool coordinate system. When the arm moves, the origin of the tool coordinate system (the center of the mechanical interface) moves to the location. If you have any tool attached to the interface, that tool extends past the location and collides with your equipment. As well, any motion along the tool axis, uses the default tool axis, the Z axis of the tool coordinate system for the F3 robot or the tool X axis for the a series robots.

Measuring for a Tool Transform

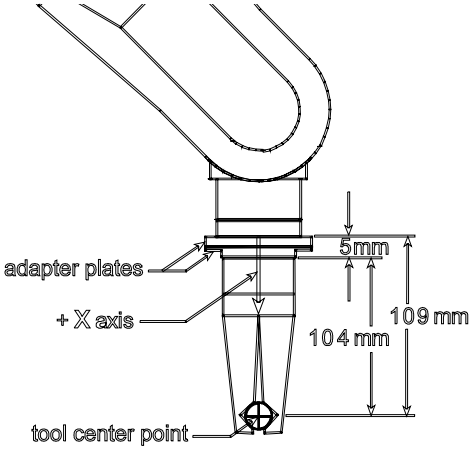
When you measure the distance to the TCP from the mechanical interface, make sure you measure the total distance including the thickness of any adapter plate between the mechanical interface and the tool.

The three examples show:

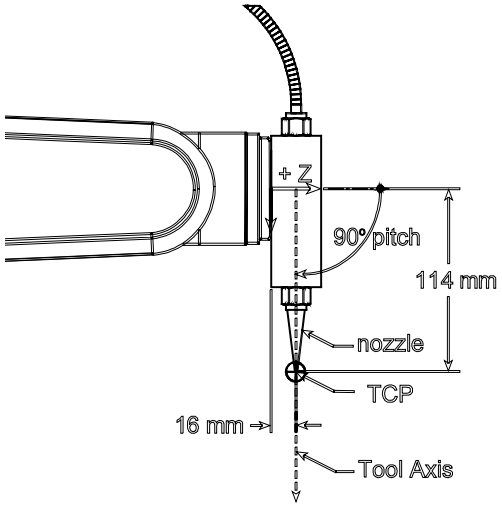
- three types of tools: 1) servo-gripper, 2) pneumatic gripper, 3) dispenser
- on three different arms: 1) A255, 2) A465 requiring 465 adapter, 3) F3 requiring F3 adapter
- with the tool coordinate system's Z axis pointing: 1) upward, 2) downward, 3) horizontally, with respect to the world coordinate system. Remember: the tool coordinate system moves with the mechanical interface and the world coordinate system can be ignored.



arm	A255
tool	CRS servo-gripper
TCP	96 mm. on the X axis, no change in orientation
transform coordinates	(96.0, 0.0, 0.0, 0.0, 0.0, 0.0)



arm	A465
tool	pneumatic gripper with custom-machined fingers (adapter plate between tool and mechanical interface)
TCP	109 mm. on the X axis, no change in orientation
transform coordinates	(109.0, 0.0, 0.0, 0.0, 0.0, 0.0)



arm	F3
tool	dispenser
TCP	114 mm. on the X axis, 16 mm. on the Z axis, 90 degree pitch
transform coordinates	(114.0, 0.0, 16.0, 0.0, 90.0, 0.0)

Setting a Tool Transform

You can set a tool transform with the application shell, a RAPL-3 program, and the teach pendant.

With the Application Shell: tool

With the application shell, you use the tool command.

You specify the position and orientation of the TCP with coordinates of the tool coordinate system.

Terminal

```
application>tool 2.0, 0.0, 3.0, 0.0,
90.0, 0.0
```

With a RAPL-3 Program: tool_set()

In a RAPL-3 program, you use the tool() command.

You specify the position and orientation of the TCP with coordinates of the tool coordinate system.

Editor

```
tool_set(2.0, 0.0, 3.0, 0.0, 90.0, 0.0)
```

Moving in the Tool System

You can move the tool center point (TCP) along tool axes in continuous motion, in jog motion and also in straight line motion. Robot motion in the tool coordinates system is tool dependent. The following table lists the available options as a function of the tool.

Motion	Tool		
	Teach Pendant	Application Shell	RAPL-3 Program
continuous motion along any axis	velocity	[no equivalent]	[no equivalent]
incremental motion along any axis, not straight line	[no equivalent]	tx, ty, tz, yaw, pitch, roll	tx(), ty(), tz(), yaw(), pitch(), roll() jog_t()
incremental motion along any axis straight line	jog	txs, tys, tzs, yaws, pitchs, rolls	txs(), tys(), tzs(), yaws(), ptichs() rolls() jog_ts()
incremental motion along the tool axis	[no equivalent]	depart departs	depart() departs()

Continuous Motion

You can move the TCP by continuous motion along any axis of the tool coordinate system, but only with the teach pendant.

Teach Pendant: Vel

With the teach pendant, you select Tool mode and Vel type of motion. You set the speed with the Speed Up and Speed Down keys. Pressing an axis key (X, Y, Z, yaw, pitch, or roll, positive or negative) moves the TCP continuously while the key is pressed.

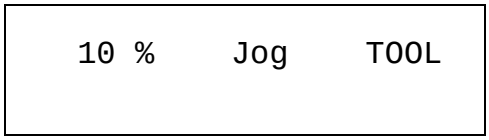
F1	F2	F3	F4	ESC
–	X +			
–	Y +			
–	Z +	SPEED DOWN	SPEED UP	
–	yaw +			
–	pitch +			
–	roll +			

Incremental Motion Along Any Axis

You can move incrementally along any tool axis using the teach pendant, the application shell, or a RAPL-3 command in a program.

Teach Pendant

With the teach pendant, you select Tool mode and Jog type of motion. You set the speed with the Speed Up and Speed Down keys. Pressing an axis key (X, Y, Z, yaw, pitch, or roll, positive or negative) moves the TCP in increments of the specified distance along the axis corresponding to the key pressed.



F1	F2	F3	F4	ESC
–	X +			
–	Y +			
–	Z +	SPEED DOWN	SPEED UP	
–	yaw +			
–	pitch +			
–	roll +			

Application Shell: tx, ty, tz, yaw, pitch, roll

With the application shell, you use the tx, ty, tz, yaw, pitch and roll commands. The tx, ty, and tz commands require a distance parameter . The yaw, pitch and roll commands require an angle parameter. Both the distance and angle parameters can be positive or negative.

The tx, ty, and tz commands require a distance parameter . The yaw, pitch and roll commands require an angle parameter. Both the distance and angle parameters can be positive or negative.

Terminal

```
application>tz 5
application>yaw 3
```

RAPL-3 Program: tx()

With the application shell, you use the tx, ty, tz, yaw, pitch, and roll commands. The tx, ty, and tz commands require a distance parameter . The yaw, pitch and roll commands require an angle parameter. Both the distance and angle parameters can be positive or negative.

Editor

```
tx(5)
yaw(3)
```

Incremental Straight Line Motion along the tool axis

You can move the TCP in a straight line motion by a specified distance along any of the tool axis of the robot. This feature can be accessed using the, application shell or a RAPL-3 program.

Application Shell: txs, tys, tzs, yaws, pitchs, rolls

Each command requires that you specify the direction and distance from the current position with a signed number.

Terminal

```
application> txs 5
```

Turn online mode on when doing pure rotations.

Terminal

```
application> online on
```

RAPL-3 Program: txs(), tys(), tzs(), yaws(), pitchs(), rolls().

For each of the commands you specify the direction and distance from the current position with a signed number.

Editor

```
    tzs(5)
    yaws(3)
```

Turn online mode on when doing pure rotations.

Editor

```
online(ON)
```

Incremental Motion Along The Tool Axis

You can move the TCP by a specified distance along the tool axis. The tool axis is the “approach/depart” axis.

Application Shell: depart

With the application shell, you use the depart command.

You specify the direction and distance from the current position with a signed number. Positive is away from the location.

Terminal

```
application>depart 5
```

RAPL-3 Program: depart()

In a RAPL-3 program, you use the depart() command.

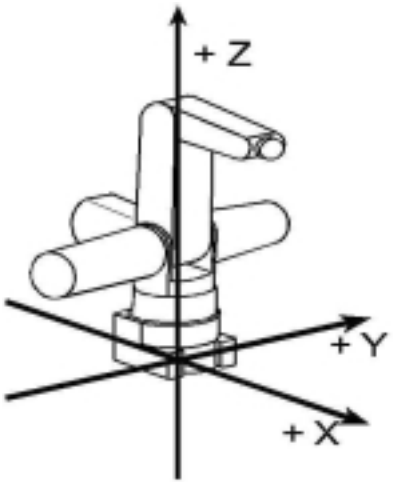
You specify the direction and distance from the current position with a signed number. Positive is away from the location.

Editor

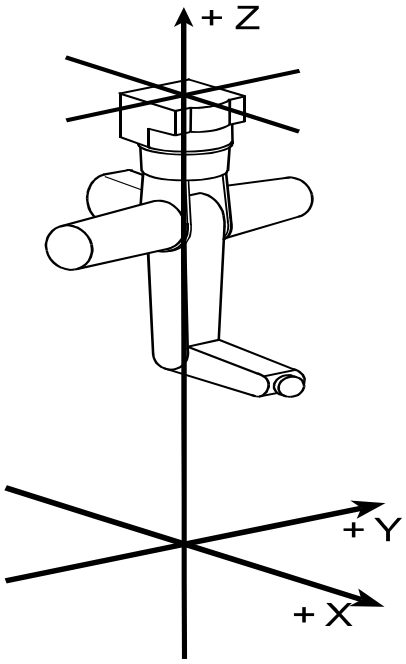
```
depart(5)
```

Base Offset

In a basic robot installation, you install the arm upright at the center of the work space. The center of the base mounting surface is the origin of the world coordinate system.



As you design your work cell, you might want to install the arm at a different position or orientation. For example, to keep the table surface clear, you might install the arm upside-down over the center of the work space.



To keep the world origin at the center of the table surface, you need to describe the difference, or “offset”, between the base of the arm and the origin of the coordinate system. To do this, you set a base offset.

A base offset is given in cartesian coordinates. It states the translation (distances along the axes) and orientation (rotation around the axes) of the

offset. For example consider the inverted installation as shown above. The coordinates of the base offset are:

Type of Coordinate	Specific Coordinate	Description	Value for Above Example
Translation	X	distance along X axis	0.0
	Y	distance along Y axis	0.0
	Z	distance along Z axis	920.0 [mm. for an A255]
Orientation	Z-rot	rotation around Z axis	0.0
	Y-rot	rotation around Y axis	180.0
	X-rot	rotation around X axis	180.0

The resulting base offset is {0.0, 0.0, 920.0, 0.0, 180.0, 180.0}. To set the base offset use the base set command.

Application Shell: base

With the application shell, you use the base command.

You specify the coordinates. If you use an integer, the system converts it to the float it needs. This offset is in effect until a new offset is set from ash or a new offset is set from a program. This setting is lost when the controller is powered down.

Terminal

```
application>base 0, 0, 920, 0,
180, 180
```

RAPL-3 Program: base_set()

In a RAPL-3 program, you use base_set() or its alias base().

Editor

```
      base_set( 0, 0, 920, 0, 180,
180)
```

Without an Offset

If you do not specify a base offset, the origin of the world coordinate system is set at its default which is the base of the arm.

Limp Motion

When a joint is limp, the servo motors and brakes are disengaged. This allows a limped joint to be moved manually. All, or selected, axes can be limped using ash, the teach pendant, or RAPL-3.

Even though the motors and brakes are disengaged, the encoders are sending signals from the arm back to the controller, informing the controller of the position of the arm. Since the encoders maintain position data, you can use limp motion to teach locations.

Warning: Caution must be used when limping joints. A limped joint will fall due to gravity or inertia. This can result in a collision which can damage the robot or other equipment.

Application Shell: limp, nolimp

With the application shell, you limp with the limp command and unlimp with the nolimp command.

Terminal

```
application> limp
```

If you do not specify any axis or axes, all the axes are limped. If you are limping all the axes, be prepared, the robot may fall under gravity. When the arm is limp, move the arm manually to the position that you want. When you are finished moving, enter the nolimp command.

Terminal

```
application> nolimp
```

You can limp and unlimp one or more specific axes.

Terminal

```
application> limp   3
application> nolimp 3
```

Terminal

```
application> limp   2, 3
application> nolimp 2, 3
```

RAPL-3 Program: limp(), nolimp()

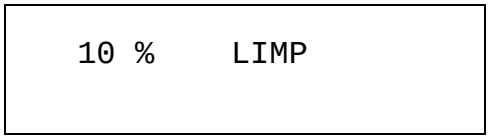
Although you can use the RAPL-3 commands the limp() and nolimp(), this is not usually done from within a program. When the program executes the limp command, the arm falls due to gravity.

Editor

```
      limp( )
      nolimp( )
```

Teach Pendant

To limp one joint, ensure that Motion Type is set to Limp.



F1	F2	F3	F4	ESC
– Ax1	+			
– Ax2	+	NO LIMP ALL	LIMP ALL	
– Ax3	+	SPEED DOWN	SPEED UP	
– Ax4	+			
– Ax5	+			
– Ax6	+			

To limp an individual joint, press the positive (+) axis key for that joint. For example **+ Ax1** for joint 1, **+ Ax2** for joint 2, etc.

To unlimp an individual joint, press the negative (-) axis key for that joint. For example **- Ax1** for joint 1, **- Ax2** for joint 2, etc.

To limp all joints, press the LIMP ALL key. To unlimp all joints, press the NO LIMP ALL key. The Limp motion type does not need to be selected. The LIMP ALL key and the NO LIMP ALL key can be pressed any time the Manual Menu screen is displayed on the pendant screen.

CHAPTER 5

Safe Robot Operation

Perform the following safety checks and procedures before beginning your application development. For additional details, refer to the safety chapter in your robot user's guide.

Safety Checks

BEFORE applying power to the arm, verify that:

- The robot is properly installed, mounted, and is stable (refer to your robot system user's guide for details).
- The electrical connections are correct and the power supplies (voltage, frequency and interference levels) are within the specified ranges (refer to your robot system user's guide for specified ranges).
- If you have modified your system, added hardware, software, or serviced your robot, recheck all the changes or additions.
- User memory is intact. Errors should not appear in your programs, location, or variable files.
- Safeguards are in place.
- The physical environment (humidity, atmospheric conditions, and temperature) is as specified. For more information refer to your robot system user's guide.

AFTER applying arm power, verify that:

- The start, stop, and function keys on the teach pendant and controller front panel function as intended.
- E-stops, safety stops, safeguards, and interlocks are functional.
- At reduced speed the robot operates properly and has the ability to handle the workpiece.
- Under normal operation, the robot functions properly and has the capability to perform its intended task at the rated speed and load.

Working Within the Robot's Workspace

Before entering within the robot's workspace, perform the following checks and safety precautions.

- Visually inspect the robot to determine if any conditions exist that can cause malfunctions or injury to persons.
- If the teach pendant controls are used, test them to ensure that they function correctly. If any damage or malfunction is found in the teach pendant, complete the required repairs before allowing personnel to enter within the robot workspace.
- While programming or teaching locations, ensure that the robot system is under your sole control.
  a) When possible, program the robot with all personnel outside the safeguarded area.
  b) When programming the robot and teaching locations within the safeguarded area, ensure that robot motion is reduced to at least 25% speed.
- While servicing the robot arm, ensure that the robot system is under your sole control.
  c) Ensure that the robot is off-line. The arm must not be stopped in, or running, a program.
  d) Ensure that the robot does not respond to any remote signals.
  e) Ensure that all safeguards and e-stops are functional.
  f) Always remove power to the arm and controller before connecting or disconnecting cables.
  g) Ensure that suspended safeguards are returned to their original effectiveness prior to initiating robot operation.
- When power to the robot arm is not required, it should be turned off.