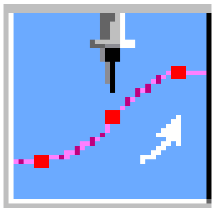
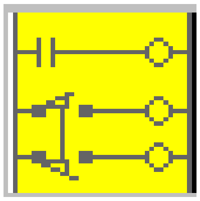
ROBOT™ Doctor

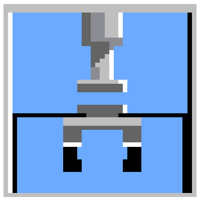
PathWare

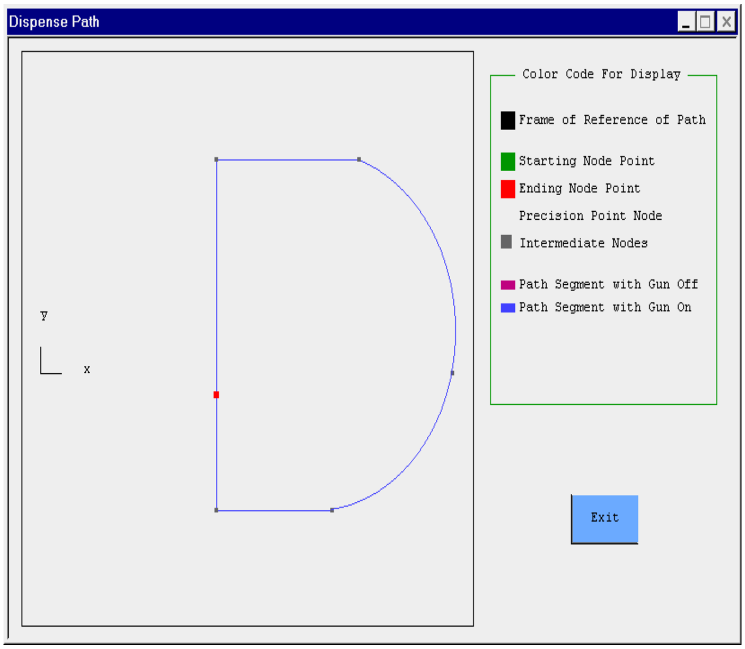
User’s & Reference Guide

Version 4.3





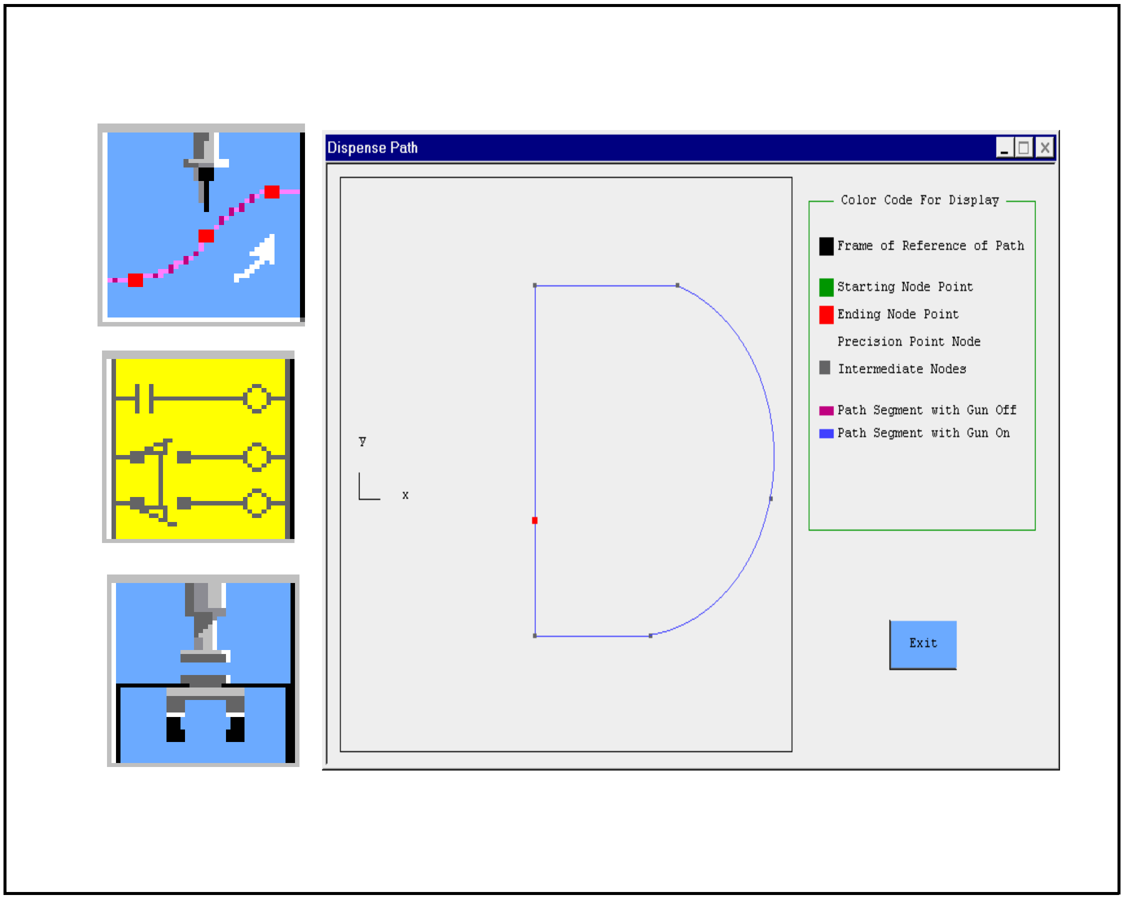




PathWare

User’s & Reference Guide

Version 4.3



00343-PthWar, Rev. A
October 2004

9220 Nottingham Way, Mason, Ohio 45040 , RobotDoctor.Com

The information contained herein is the property of RobotDoctor, LLC, and shall not be reproduced in whole or in part without prior approval of RobotDoctor,LLC. The information herein is subject to change without notice and should not be construed as a commitment by RobotDoctor, LLC. This manual is periodically reviewed and revised.

RobotDoctor, LLC assumes no responsibility for any error or omissions in this document.Critical evaluation of this manual by the user is welcomed.

Table of Contents

List of Figures

List of Tables\

Introduction 1

What Is PathWare?

PathWare is an AIM application module designed specifically for use with Adept controlled motion devices in automated path applications such as material dispensing, grinding, welding, and cutting applications. This software simplifies the process of programming a robot for path control applications where I/O timing, precise tracking, and speed control are critical. PathWare provides a set of utilities that allows you to easily train path patterns. The module also provides motion control routines that ensure precise robot tracking.

Unlike other AIM modules, PathWare must be used in conjunction with the MotionWare application module.

Using This Manual

This manual presents a detailed description of the AIM PathWare Software. We assume you are familiar with AIM, and the basic operation of Adept robot systems. See the ***Adept MV Controller User's Guide*** and *Adept Robot User's Guide.* for details the robot and controller. See the ***MotionWare User's Guide*** for details on using the basic AIM system. If you will be customizing PathWare, we assume you are familiar with the V$^+$ operating system and language. See the AIM reference manuals and the ***V$^+$ Language Reference Guide***

Chapter 1 presents an overview of the software features of the PathWare software. It also reviews the Adept hardware required by this module and the hardware recommended for robotic applications. This chapter should be read by all PathWare users.

Chapter 2 describes the software installation and start-up procedures for a PathWare system. This chapter is intended for system customizers who will install the software and users interested in the AIM PathWare software organization.

Chapter 3 describes the databases used with the PathWare software. All users should review this chapter and become familiar with these databases.

Chapter 4 details creating a PathWare implementation. The AIM statements and the pendant teach routines are discussed in this chapter. An example application is reviewed. This chapter should be read by all PathWare users.

Chapter 5 provides information on customizations that may need to be done to support special tooling or processes. Details on the standard gun control routines, adding custom devices, and servo motor server and other information is included in this chapter.

Chapter 6 summarizes the PathWare runtime routines. System customizers should read this chapter if they intend to modify their AIM system.

Chapter 7 covers the structure of the PathWare databases and the additional system constants created for the databases.

Appendix 8 describes the error codes used with the PathWare Software.

Software Overview

PathWare is an application module of the AIM software system. It was developed to speed implementation of robotic dispensing cells, and provide precise and predictable robot path tracking. To achieve these goals, the PathWare Product adds new task-level statements, additional databases, and new teaching utilities to MotionWare. PathWare software is designed with flexibility in mind for your applications. It supports many different robot devices that work with the Adept Motion Servo product. Additionally, it can be tailored for other process path applications such as water jet cutting and welding.

PathWare requires AIM version 3.2 and the Adept V^+ operating system version 11.3 or higher. Some features of the PathWare Product are described below. Details are presented in subsequent chapters.

The DISPENSE Statement and the Process Path Database

PathWare augments the AIM MotionWare Package. The DISPENSE statement commands the robot to move along a constant velocity path while controlling the dispensing equipment. The result is a consistent, high-quality bead.

The DISPENSE statement retrieves dispensing process information from the Process Path database. This database has fields that specify the desired robot speed, path acceleration and deceleration, the dispensing gun state (on or off), precision point motion, path compensation, path checking, conveyor tracking, frame relative motion, and tool compensation. Path compensation allows the operator to change the robot's path by a distance to the right- or left-hand side of travel (this is sometimes called *cutter compensation*).

Frame relative motion allows vision guidance to be used in conjunction with PathWare. Vision can be used with this product to determine the part's orientation, and inspection of bead quality. When vision is used with conveyor tracking, parts can be found as they move down a conveyor belt. Belt tracking allows a conveyor queue to be established that contains the locations of the parts. As the parts move into the work envelope of the robot, the robot will follow the taught path defined relative to the frame of reference.

Precision point robot locations are also provided with the PathWare software. A precision point will allow you to define the joint angles for a position. This allows the proper arm configuration to be established before the path motion begins. Precision points can be defined for robots with up to six axes of motion. Most of Adept's motion control device modules are supported.

Integral pendant teach routines developed for PathWare simplify path training. Only *primary* path points (i.e., start, corner, and stop points) need to be specified along the path. You use the pendant teach routines to automatically store path points in database records.

The Dispense statement as well as the APPLY_DOT statement are linked to a signal database that defines path relative device actuation. Devices that are supported include Outputs for dispense guns, support for Servo Motors relative to tool tip velocities and support for analog signals relative to tool tip velocities. Gun compensation parameters, which are used as time offsets to compensate for viscous material delays, allow you to adjust dispensed material start and end points without having to reteach paths. Two compensation parameters are provided for On and Off compensation. Additional timing can be added with minor customization for up to five different timing compensation values relative to node points.

PathWare also provides support for tool tip relative control via Servo Motors or Analog signals. To control the device output, the Process Path database includes the *Servo Rate (CC/Vel)/bead %* field which defines the bead size of the dispensed material. The bead size parameter acts as a scale factor for the maximum output of the tool tip velocity relative device.

Support for a velocity relative device requires a either a motion interface board (MI3) for servo motors or an analog I/O board. Please contact the Adept Applications Engineering group for interface connection information for both velocity path control options.

The APPLY_DOT Statement

For simple applications where process paths are not required, this statement can be used to deposit dots of material. This statement moves the robot to a location and then turns on the dispensing device for a specified time. Position data is stored in MotionWare's standard Location database. The dispense signal database is a required input argument. This allows analog and servo pump devices to be controlled with this statement. Other arguments allow time delays, analog voltage output, and volume requirements, depending on the equipment. This statement allows an automated way of purging the system as well as precise metering of material at locations.

The SHIFT_DOT Statement

This statement provides the same functionality as the apply_dot statement except that it allows the user to shift the positions. This can be useful for applications that require a pattern of dots to be applied.

The SET_DM_PATH Statement

This statement allows the user to modify path positions from the sequence. This can allow other processes such as vision to modify the path.

The CHG_DISP_SPEED Statement

This statement allows path speed changes to occur from a sequence. These changes can be in the form of either an offset speed change or an overall percentage change. The offset speed change allows the user to save the database records after the change occurs. This can allow an easy method to deal with viscosity changes in material.

The Z_UP Statement

This statement performs Z-axis motions to allow clearance before motion begins to a position. This is useful at the beginning of the sequence and after errors occur in the cell.

MotionWare Statements

In addition to the statements provided with PathWare, the complete set of MotionWare statements is also provided. See the ***MotionWare User's Guide*** for details.

The TOOL Database

The Tool database is the standard version of the basic AIM MotionWare Tool database. This database menu screen has been modified to include a teach pendant routine to teach a tool transform.

What's New in Revision 4.3

This section is provided to detail the changes in the software for the new release in comparison to the previous release of PathWare. Below are the major changes:

- Support for Device Net Analog I/O has been added.

Hardware Requirements

PathWare requires the following Adept controller options:

- Adept Hardware to support Graphics (VGB or AdeptWindows)
- Adept Teach Pendant (MCP)
- Mouse, Monitor, and Keyboard (not required with AdeptWindows)
- 68040 / 68060 system processor (with at least 8 Mb of RAM) The 68060 processor is recommended for high-end applications that demand more processing speed.

The following hardware is recommended for dispensing:

- Hose hangers and tool-balancing equipment
- Quick-change end-effector tooling

Hose hangers and tool balancers are recommended to minimize the load applied to the robot. For optimum robot performance, dispensing equipment and hoses mounted to the end of the robot must be properly balanced.

Quick-change tooling is recommended to allow dispensing hardware to be easily detached from the robot during robot start-up calibration. Because hoses attached to the robot can hinder the full range of motion for joint 4 (and joint 5 on five-axis robots), the hoses must be removed before start-up calibration. The CHANGE_HAND statement is included in the AIM Utility Software to help with automatic tool changes.

How Can I Get Help

You can obtain help with the RobotDoctor PathWare product by calling directly to (513)-702-3709 or via E-Mail with Steve Roell -- SROELL@RobotDoctor.com

Installation 2

Before You Begin

This chapter describes the PathWare Software installation procedure. To simplify the installation, have the **Adept Utility Disk** handy. The Utility Disk is supplied with all Adept controllers. Complete instructions for using the diskcopy utility are in the ***Instructions for Adept Utility Programs***.

The AIM 4.0 PathWare package is shipped on CD-ROM. This CD is configured with folders that includes all of the software for the complete installation. This manual is written with the assumption that the software needs to be installed. Please be sure that the following software licenses have been purchased and installed.

AIM Software License
AIM PathWare License (Call (513)-702-3709)
(You will need the Controller number or Security ID depending on controller.
Please type the command ID on the Adept monitor)

AIM Software and AIM PathWare are required for the PathWare software package to function. The Aim Extensions License is only required if vision guidance is used.

Some systems may be upgrading and adding the PathWare package. The PathWare software is provided both in a folder that has the software installed and just needs the files to be transferred and also in a folder that just contains the PathWare software packages. The folder labeled AIMPATH contains the complete installation and the folder labeled PathWare contains the files for PathWare. If you are installing PathWare over a customized version of AIM, please review the list below to assure no customized files are common with the file list. PathWare does overwrite some menu files from standard AIM.

Table 2-1 lists the files that are included with the PathWare software. The files with *SQU* filename extensions are *squeezed*, i.e., all comments have been removed from the files. Squeezed files require less memory when they are loaded than their commented non-squeezed *V2* counterparts. The squeezed files and their V2 counterparts are otherwise identical. (See the ***MotionWare User's Guide*** for a description of all AIM file types.)

To install the PathWare software, follow the steps in **Installation Procedure**. The installation procedure copies all the files listed in **Table 2-1** onto the hard disk drive. Files marked with a P are protected files and cannot be read by the user. Files marked with a *replace existing AIM files.

Table 2-1. PathWare Files

Disk Files		Contents
LDM.V2		AIM load file for the dispensing module.
RUN_DM.V2		Commented dispensing runtime routines.
LIB_DM.SQU	P	Library routines for the Dispensing Module.
RUN_DM.SQU		Squeezed dispensing runtime routines.
MENU_DM.SQU	P	Squeezed menu and teach routines.
DMMOD.OVR		Startup overlay file for loading and link databases.
DMMOD.OV2		Commented version of DMMOD.OVR.
DIS.MNU		Dispensing path database menu.
DMFLOW.MNU		Menu file for signal I/O and flow gun.
DMSIGNAL.MNU		Signal database menu file.
DIS.RFD		Dispensing path database .RFD file.
DMSIGNAL.RFD		Signal database .RFD file.
DIS.DB		Dispensing module database file.
DMSIGNAL.DB		Signal database file.
STATDM.DB		Dispensing module statements.
ERRORDM.DB		Dispensing module error database.
DMICON.DAT		Dispensing module menu icons.
CUSTOM.V2	*	Initialization files for custom features.
CUSTOM.SQU	*	Squeezed version of CUSTOM.V2.
PHWINI.DB		PathWare Initialization Database.
MENU_VDP	P	Path Editing Menu Routines.
MVAR.DB		Global Variable Database.
PTH.CGM		PathWare Plug-In application file.

Installation Procedure

You can use the V$^+$ utility program DISKCOPY to copy the PathWare software to your hard disk. Complete instructions for using the utility program are contained in the *Instructions for Adept Utility Programs* supplied with your V$^+$ reference manuals. The following section details the steps necessary to copy the PathWare files. PathWare is now provided on a CD-ROM. This requires the user to either have NFS or an FTP solution to transfer files. The instructions below assume the User is using NFS.

1. Turn on your controller.

2. Make sure the NFS drive has been mounted. The user can transfer the files from the CD over to the PC hard drive or mount the CD drive from NFS.

3. After the NFS drives have been mounted, load the Diskcopy program from the D drive from the UTIL subdirectory.

   ```
   LOAD D:\UTIL\DISKCOPY
   ```

4. When the program files have completed loading, enter the command:

   ```
   EXECUTE 1 a.diskcopy
   ```

5. From the menu that is presented, select:

   ```
   4. => Copy multiple files
   ```

6. Answer **A** when prompted:

   ```
   What is the INPUT disk (i.e., D, device>unit)?
   ```

   If the AWC processor has been purchased, the user may have to use NFS to copy the files into the Adept Controller. In this case the user will have to type in NFS>YOURDRIVE.

7. Answer **D** when prompted:

   ```
   What is the OUTPUT disk (e.g., A/B/C/D)?
   ```

8. Answer ***.*** when prompted:

   ```
   Enter spec of file(s) to copy (blank to exit):
   ```

9. Press the **ENTER** key when prompted (a subdirectory name may be specified):

```
      Enter output subdirectory (blank for default):

      \AIM\
```

10. Answer **Y** when prompted:

```
      Do you want existing files automatically superseded
   (Y/N)?
```

11. Answer **g** when prompted:

```
      Copy this file...
```

12. After the software has been copied, press the ENTER key when prompted:

```
      Enter spec of file(s) to copy (blank to exit):
```

13. Select exit from the main menu.

Verifying Installation

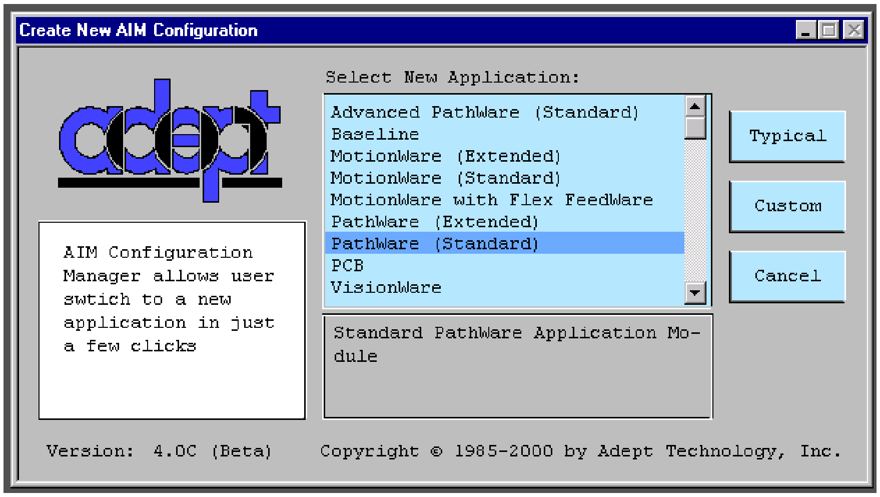
To verify the correct installation of the PathWare Software, follow the steps below.

1. Set the default disk to the AIM subdirectory and enter the following commands to load the Software:

```
LOAD LOAD_AIM.V2

COMMAND LOAD_AIM
```

2. When Running AIM for the first time you will have to configure the system for the software application you wish to run. The menu below should be displayed to allow you to select PathWare. The user should select PathWare or PathWare Extensions (If the user is using a servo pump). PathWare will automatically load several utility plug-ins that are provided for ease-of-use. These plug-ins can be prevented from loading by editing the system configuration.



AIM Configuration Manager allows user swtich to a new application in just a few clicks

Standard PathWare Application Module

Version: 4.0C (Beta)

Copyright © 1985-2000 by Adept Technology, Inc.

Figure 2-1. System Configuration

3. Pull down the **Edit** menu. Options should be displayed for editing the Sequence, Variables, Conveyor, Frame, Path, Tool, Location, PathWare, PathWare Signal, Tool. If additional software options have been installed, there may be more options listed on the menu.

4. Select **Edit** · PathWare Signals. Open a new record (press the *New* key [F2]). Enter a gun signal number and enable the gun signal with the push button (the center should be green). Enter a name for this record.

5. Select **I/O** Disp Controls. Carefully, toggle the output signals to see if the appropriate outputs work properly. Take extreme care when testing the end-effector I/O. Tooling can be damaged if the I/O lines are not hooked up properly.

After completion of the steps above, the installation is complete. If you experience problems, check that the installation procedures were followed exactly. If problems persist, contact RobotDoctor at the number listed in **Chapter 1**.

Please note that Pathware can not be installed as a plug-in. This because a different scheduler is installed for PathWare to function properly. If PathWare is being added to an existing software system please call RobotDoctor for more details.

See the ***MotionWare User's Guide*** for more details on AIM application software configurations.

AIM Utility Plug-Ins

Provided with the PathWare software CD-ROM are several plug-in software packages that are provided for ease-of-use when using the PathWare package. This software is basically the statement software that used to be provided as AIM Utilities. Documentation for these statements is provided on the CD in the AIM Utilities manual.

PathWare Databases and Menu Definitions 3

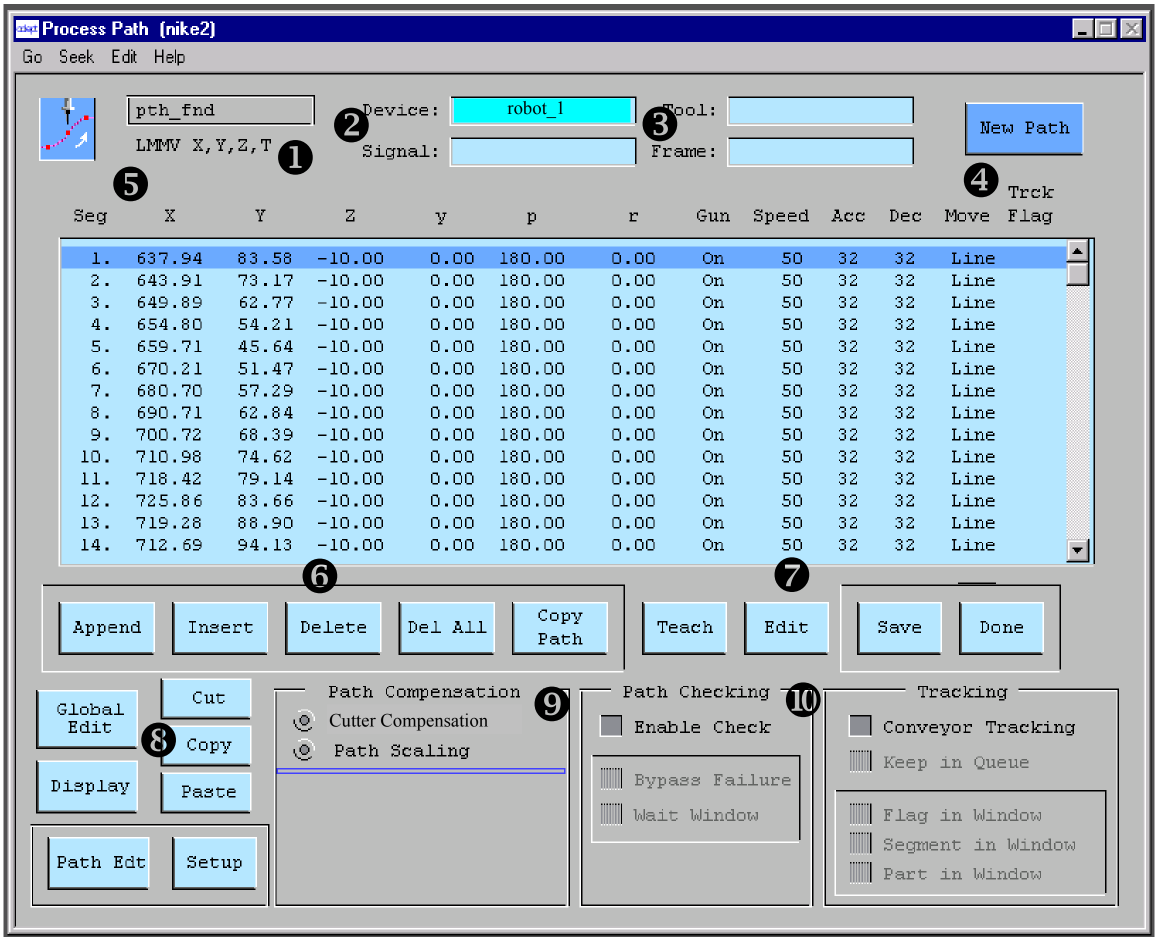
Process Path Database

To create a new path, be sure the proper database module is selected and created before defining the path records. Please refer to the *MotionWare User's Guide* for more information on database modules.

Edit ⇨ Process Path⇨ New Path ⇨ *enter path name* ⇨ Create

To edit an existing path:

Edit ⇨ Process Path⇨ Seek ⇨ Index ⇨ *dbl clk on path name*



pth_fnd

LMMV X,Y,Z,T

Device: robot_1

Signal:

Tool:

Frame:

Seg	X	Y	Z	y	p	r	Gun	Speed	Acc	Dec	Move	Trck Flag
1.	637.94	83.58	-10.00	0.00	180.00	0.00	On	50	32	32	Line	
2.	643.91	73.17	-10.00	0.00	180.00	0.00	On	50	32	32	Line	
3.	649.89	62.77	-10.00	0.00	180.00	0.00	On	50	32	32	Line	
4.	654.80	54.21	-10.00	0.00	180.00	0.00	On	50	32	32	Line	
5.	659.71	45.64	-10.00	0.00	180.00	0.00	On	50	32	32	Line	
6.	670.21	51.47	-10.00	0.00	180.00	0.00	On	50	32	32	Line	
7.	680.70	57.29	-10.00	0.00	180.00	0.00	On	50	32	32	Line	
8.	690.71	62.84	-10.00	0.00	180.00	0.00	On	50	32	32	Line	
9.	700.72	68.39	-10.00	0.00	180.00	0.00	On	50	32	32	Line	
10.	710.98	74.62	-10.00	0.00	180.00	0.00	On	50	32	32	Line	
11.	718.42	79.14	-10.00	0.00	180.00	0.00	On	50	32	32	Line	
12.	725.86	83.66	-10.00	0.00	180.00	0.00	On	50	32	32	Line	
13.	719.28	88.90	-10.00	0.00	180.00	0.00	On	50	32	32	Line	
14.	712.69	94.13	-10.00	0.00	180.00	0.00	On	50	32	32	Line	

Figure 3-1. Process Path Database

(The numbered items are described on the next page.)

Process Path Options

❶ Shows the name of the path being edited.

Indicates the kinematic device module that the current selected robot is running. This has been added for multiple robot support to assure the correct path is running with the proper robot.

❷ AIM supports multiple robots. A different robot can be selected by double clicking on the **Device** field and selecting the proper robot from the pick list.

Specify a **PathWare Signal** record to be associated with this path (double click on the data box to display a pick list of signal records). The signal record is optional. The signals can be turned off from the control panel during debug operations.

❸ In the **Tool** data box, specify a tool record to be associated with this path (double click on the data box to display a pick list of tool records).

In the **Frame** data box, specify a frame record to be associated with this path (double click on the data box the display a pick lists of frame records). See the *MotionWare User's Guide* for details on the frame database. Tool and frame records are optional.

❹ Press **New Path** to create a new path.

❺ This area displays the characteristics of all the path segments defined for the path. See section for details on specifying a path segment. Double click on a segment to edit that segment.

❻ Press **Append** to add a new segment to the end of the path.

Press **Insert** to insert a new path segment above the highlighted segment.

Press **Delete** to delete the highlighted path segment.

Press **Delete All** to delete all segments in the current path.

Press **Copy** to copy the current path to a new name.

❼ Press **Teach** to use the MCP to develop the path segments data rather

than using the graphics menus provided. The teach pendant routine allows the user to accomplish most of the same functions as the menus while teaching points remotely by the robot. See **"Teaching Path Segments With the Teach Pendant" on page 109** for more information.

Press **Edit** to edit the characteristics of the highlighted path segment (see the next section).

Press **Save** to save changes to the path database to the hard drive.

Press **Done** to exit.

❽ Press **Global Edit** to allow parameter changes to many of the path parameters globally though all of the records. Items like Speed, Accelerations, Delays, Bead% and position offsets are allowed from this

pop-up menu. Refer to the following **Figure 3-2** on Global Editing for more information.

Press **Display** to view the current taught path from a window display.

This display is shown in **Figure 3-3**. It indicates all of the node points as well as the gun states.

Press **CUT** to cut a path record from the display scroll screen above. This record will be placed in a buffer that can be pasted in a different location.

Press **COPY** to copy a path record from the display scroll screen above. This record can be pasted in a different location.

Press **PASTE** to paste either a copied or a cut record from the Process Database.

Press **Path Edt** to branch to the Path Editing menu. This menu allows the user to move to positions and test run the path. Additionally, this menu allows the new vision editing features in PathWare (see **Figure 3-7** for more details).

Press **SETUP** to branch to the Path Editing Setup menu. This menu allows the user to set parameters that effect the Path Editing Menu. These parameters allow the user to force the calculation of a reference frame, specify the vision picture record and force tool offsets to be used during path editing (see Figure 3-8 for more details).

❾ This section specifies a path compensation. This compensation can be specified as either Cutter Compensation or Path Scaling.

Cutter Compensation:

Select ☑ **Right** to apply the compensation to the right side of the direction of travel. Select ☑ **Left** to apply the compensation to the left side of the direction of travel.

Select ☑ the plane that the compensation will be applied (X/Y, Z/X, Y/Z)

Enter the amount of compensation to be applied to the path.

Path Scaling:

Enter the X axis scale factor to be applied to the path.
Enter the Y axis scale factor to be applied to the path.

❿ ☑ **Enable Check** allows the user to perform the Path Checking feature
within PathWare. This allows the feature to be turned on with two options if conveyor tracking is used. Path checking compares the joint
velocities relative to the maximum joint velocities specified in the Initialization Database. The maximum joint velocities need to be provided for this feature to function properly.

☑ **Bypass Failure** option will ignore the current part in the queue that
failed the path check when conveyor tracking is used.

☑ **Wait Window** option will continue to check the joint positions as the
part moves down the conveyor belt until it is successful or out of the conveyor window.

This section allows the user to turn on the ☑ **Conveyor Tracking** feature within PathWare. There are several selection boxes that allow the
user to affect the way the software deals with parts just coming into the conveyor window.

The ☑ **Keep in Queue** feature will leave the part in the conveyor queue after the operation is complete.   This is useful if other tracking operations need to be performed on the same part.

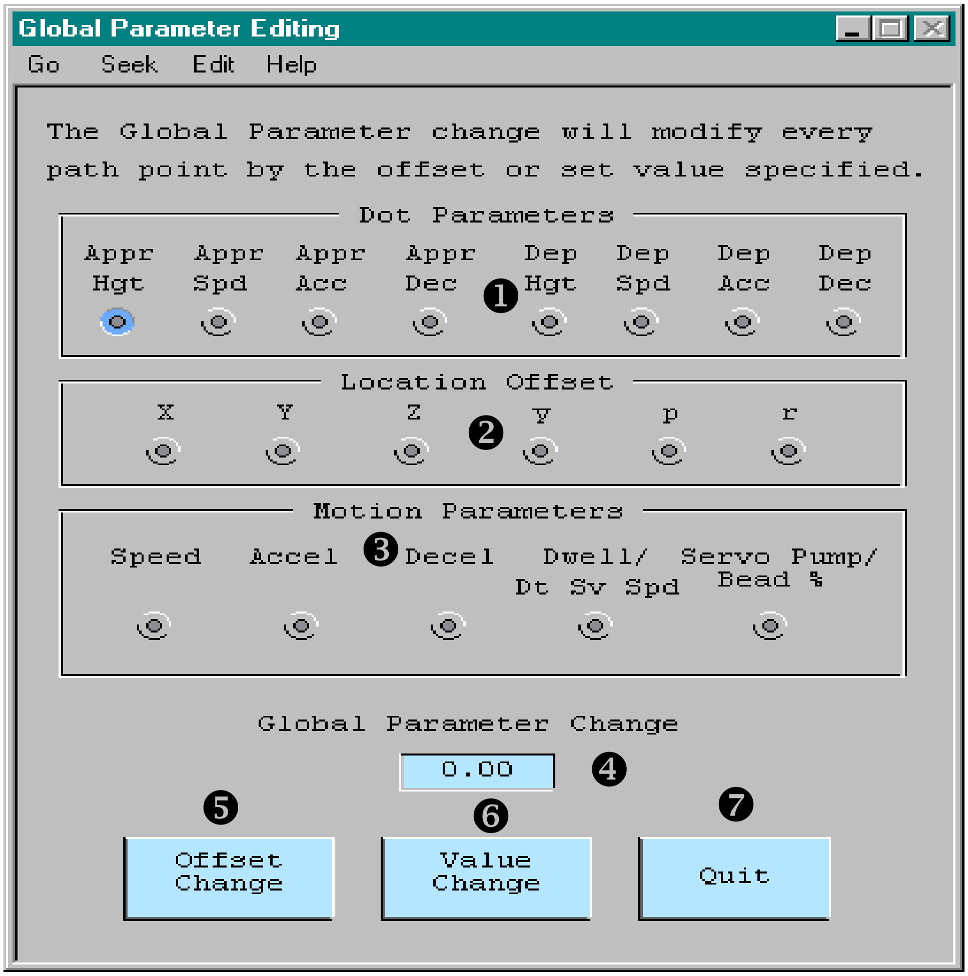
The ☑ **Flag in Window** option allows the user to flag a path segment to force the path operation to wait until the path segment is in the conveyor window. The robot will track the first point and begin the path operation when the flag is in the window.

The ☑ **Segment in Window** feature checks one node ahead of its current position and waits until that node is within the window. This feature is not ideal for dispensing applications because there may be starting and stopping along the path based on the node positions. This is currently used for cutting operations.

The ☑ **Part in Window** feature checks all of the node positions within the path before starting the path operations.

Global Editing Window

The global editing window allows the user to make global offset changes to the entire path family of records under the specified name of the path.



The Global Parameter change will modify every path point by the offset or set value specified.

Figure 3-2. Global Editing of Path

Global Path Editing

❶ Dot parameters are provided for global changes to all dot positions.

❷ Position offsets are selected from these radio buttons. Only one selection can be made per offset change request.

❸ Press the radio button of the Path parameter you wish to change by an offset amount.

❹ Enter the amount, positive or negative, by which you wish to change the selected parameter or positional offset.

❺ Press this button to invoke the offset change specified by the radio button and the change amount.

❻ Press this button to change the selected parameter to this value throughout the entire path selected.

❼ Press this button to exit the menu.

Path Display Window

The path display window will appear when the display button is pressed on the PathWare main menu. This window is useful to view the current taught path before running the robot device along the path. This display indicates starting, stopping, precision points, and standard nodes with different color node points. It additionally will indicate the state of the gun based on the path color (Blue -- ON, Fuchsia -- OFF).

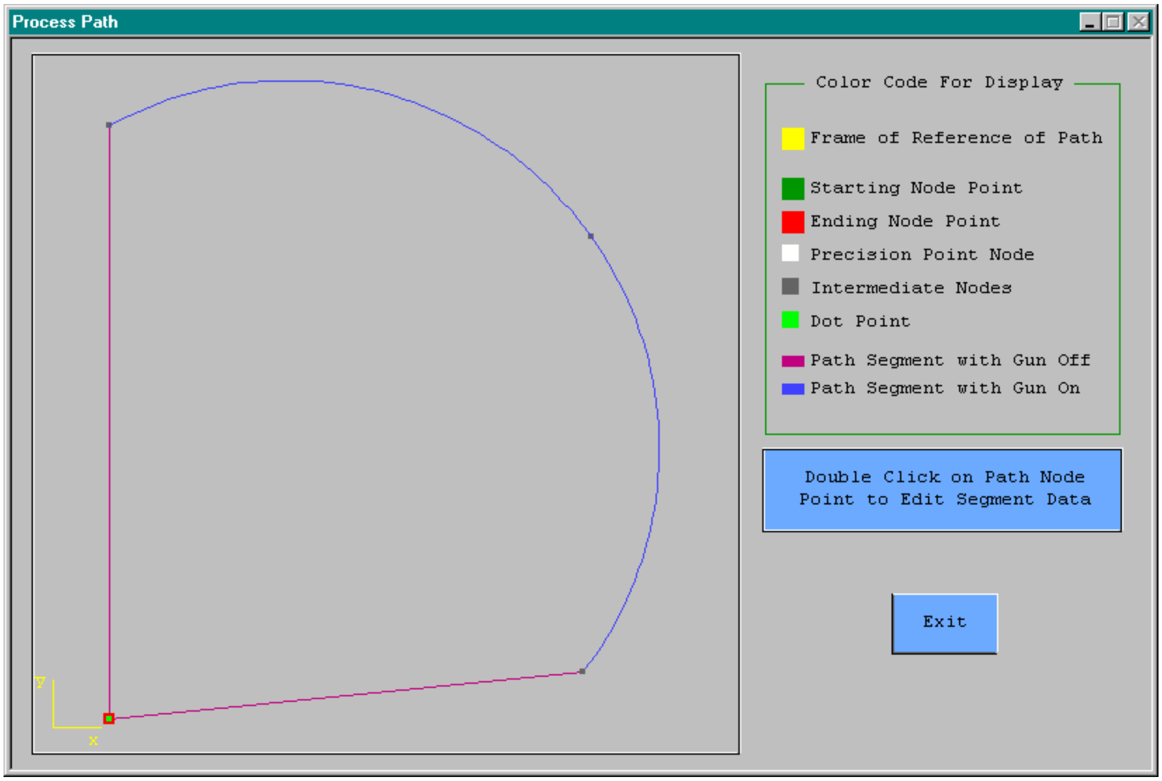

Process Path
Y
X
Color Code For Display
🟨 Frame of Reference of Path
🟩 Starting Node Point
🟥 Ending Node Point
□ Precision Point Node
■ Intermediate Nodes
🟩 Dot Point
▬ Path Segment with Gun Off
▬ Path Segment with Gun On
Double Click on Path Node Point to Edit Segment Data
Exit

Figure 3-3. Displaying Paths

Process Path Segments

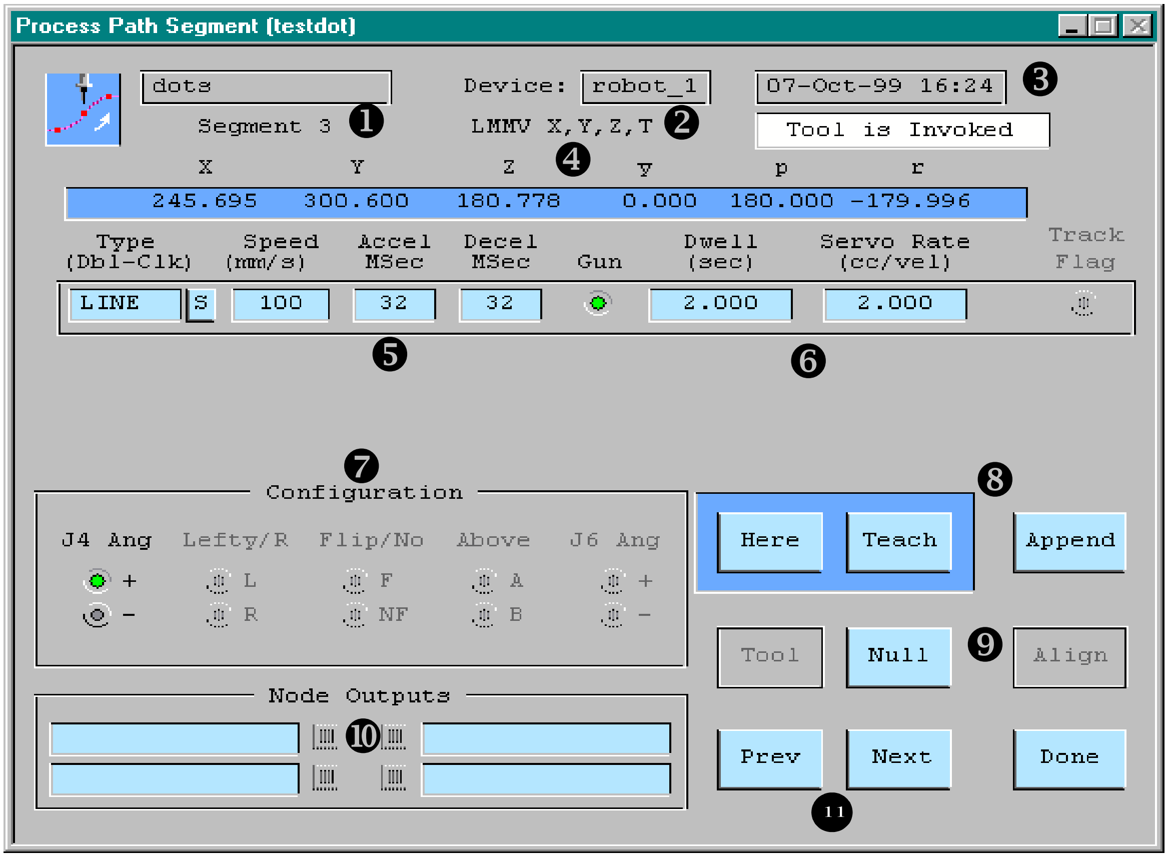
Four different path segment menus will appear depending on the motion type specified. Available for motion types are Line, Arc, Prec, and Dot. In addition, different parameters are available when a servo device or an analog device is used. When the segment is first defined, the Line type motion is selected by default. Three menus will be displayed below. The Line and Arc menus have the same parameters available when selected. The common parameters will be defined online the first window. To create a new path segment:

Edit ⇨ Process Path ⇨ Seek ⇨ Index ⇨ *dbl clk on path name* ⇨

Append or **Insert**

To edit an existing path segment:

Edit ⇨ Process Path ⇨ Seek ⇨ Index ⇨ *dbl clk on path name* ⇨ *dbl clk on path segment*



Process Path Segment (testdot)

dots

Segment 3

Device: robot_1

LMMV X,Y,Z,T

07-Oct-99 16:24

Tool is Invoked

X	Y	Z	y	p	r
245.695	300.600	180.778	0.000	180.000	-179.996

Type (Dbl-Clk)		Speed (mm/s)	Accel MSec	Decel MSec	Gun	Dwell (sec)	Servo Rate (cc/vel)	Track Flag
LINE	S	100	32	32	●	2.000	2.000	○

Configuration

J4 Ang	Lefty/R	Flip/No	Above	J6 Ang
● +	○ L	○ F	○ A	○ +
○ -	○ R	○ NF	○ B	○ -

Node Outputs

Figure 3-4. Line Path Segment

Process Path Line and Arc Segment Options

❶ Shows the name of the path this segment is in, and the current segment number.

❷ Displays the current selected robot number. This number can be changed only from the main menu.

Below the robot number the robot kinematic device type is displayed.

❸ Shows the date this segment was last modified.

If a tool record has been specified for this path, the state of the tool is shown. Please be sure that the tool is invoked if path teaching is to occur. If the tool is not invoked, unexpected positions might occur.

❹ Shows the location values of this segment. These values can be redefined by pressing **Here** or **Teach**. The positional values can also be edited manually, but make sure you do not enter values that will cause the robot to move to inaccessible areas.

❺ The **Line** selection box allows the user to define what type of motion will occur in this path segment. The options are Line, Arc, Prec, and Dot. The motion type can be changed by either double-clicking on the box or pressing the box labeled **S**. The box labeled **S** will step between the available selections. The Line selection will cause the robot to move along a straight line to this location. The **Arc** selection will require two additional path segments to be defined to establish the Arc's radius and start and stop points. These points will have to be defined as line segments for proper system operation.

In the **Speed** data box, enter the robot speed (in millimeters per second) for the move from this location along the segment. In order for this value to be accurate, a speed of 100 must be set at the task control panel. The Master speed must be set to 100% for gun compensation parameters to function properly. It is recommended to change the default speed values in the initialization database for the robot to 100%. Speed is used to control the bead size for standard dispense guns. For precision points, the speed setting is also specified in units of millimeters per second. It is accomplished by calculating the time to travel the required distance and using the V+ duration instruction. Default speeds can be changed in the dispense initialization database

In the **Accel** and **Decel** data boxes, enter values for the robot to use for the acceleration and deceleration. The units for path accelerations are in milliseconds. In order for acceleration and deceleration to occur, the time units must be at least twice the path point spacing. The default path spacing is set to 32 milliseconds as shipped to the customer. For precision point motion the units for acceleration are expressed as a percentage; the maximum is 100%.   Refer to **Path Calculations** for more details on acceleration. Default acceleration and deceleration values can be changed in the dispense initialization database.

Select ◉ **Gun** if the dispensing gun should be turned on for the move following this location. If gun compensation is to be used, the first point cannot be specified to an On gun state.   The point before the first On gun state must be set at a distance far enough and with a speed slow enough that the gun compensation timing value is smaller than the time required to move to the first On position.   The speed and distance can be changed to allow a greater period of time if required. (The gun is always turned off after the last path segment.)

❻ In the **Dwell** data box, enter a time for the robot to rest at this location before moving to the next location. This will allow more material to be applied at node points if required.

The **Servo Rate CC/Vel** & **Bead%** values are intended for use with velocity relative control devices such as Analog Variable Nozzles and Servo Pump control. Enter the percent of maximum bead that should be applied during the move along this segment for analog nozzle control. Enter the pump rate in CC/(mm/sec) for bead control when using the Servo Pump Option. Several parameters for control of these devices are specified in the Dispense Signal database. The text shown above the data entry box is based on what is selected in the signal database. This entry will not appear unless the analog or servo device is selected in the signal database.

❼ This section works in conjunction with the precision point selections.   Its purpose is to specify the proper arm configuration before moving to the precision point specified.   Refer to **"Arm Configuration/Precision Point" on page 37** for more details.

❽ Press **Here** to record the current robot location as the path segment
ment
location.

Press **Teach** to use the manual control pendant teach routine to record a path segment location.

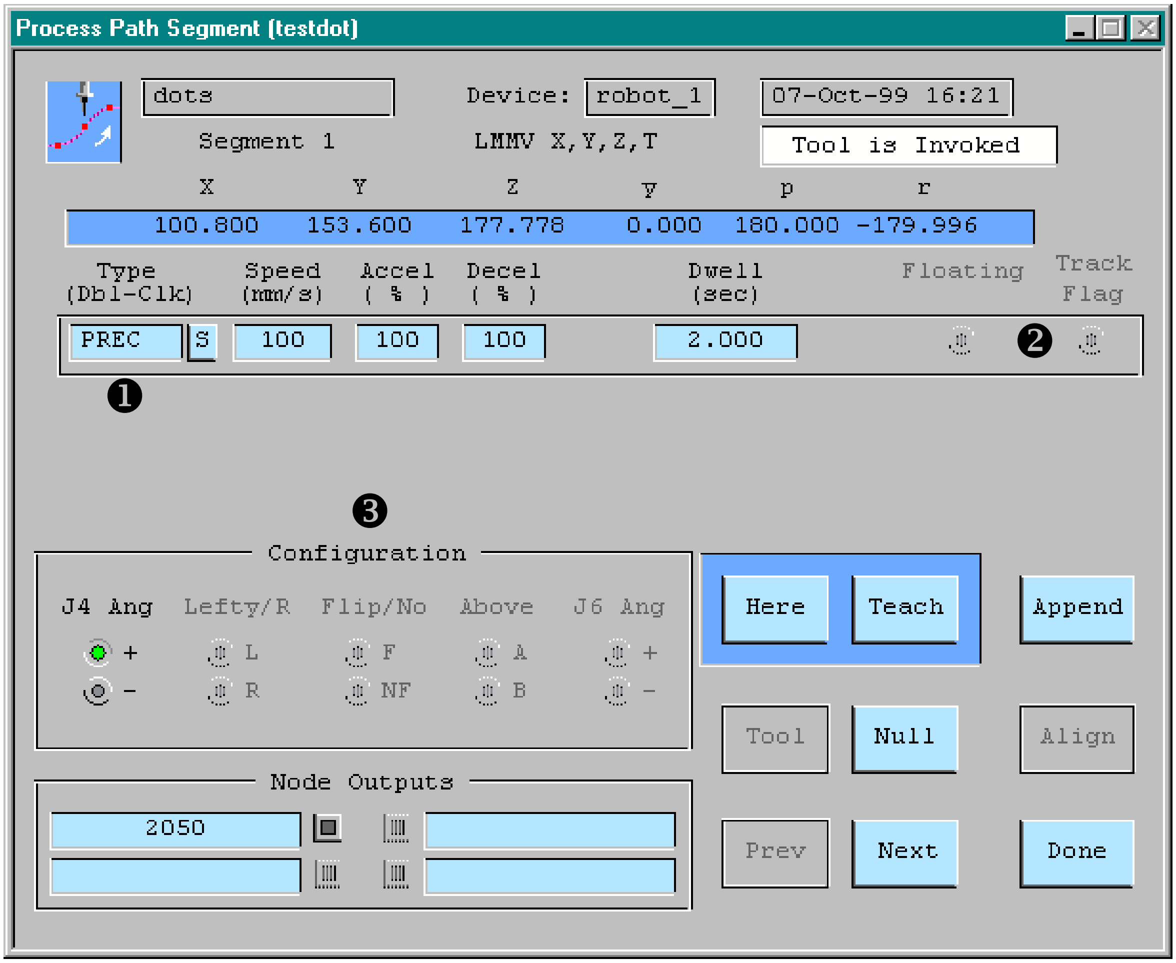
Press **Append** to add another path record to the end of the existing path family. The menu will branch to the new record for editing.

❾ If a tool record has been specified for this path, press **Tool** to invoke the tool offset for use during teaching of this segment of the path. Press **Null** to disable the tool offset.

Press **Align** to force the robot wrist to align itself with the world coordinate system of the robot. This button functions only with robot with 3-axis wrists.

❿ The Node Output area allows the user to specify output signals that get turned on at that position. The user must turn off and keep track of these signals during the cycle, unlide the gun signals which do not require such monitoring.

⓫ Press **Prev** to edit the previous segment in the path. Press **Next** to edit the next segment in the path. Press **Done** to exit path segment editing.



Process Path Segment (testdot)

dots

Segment 1

Device: robot_1

LMMV X,Y,Z,T

07-Oct-99 16:21

Tool is Invoked

X	Y	Z	y	p	r
100.800	153.600	177.778	0.000	180.000	-179.996

Type (Dbl-Clk)		Speed (mm/s)	Accel (%)	Decel (%)	Dwell (sec)	Floating	Track Flag
PREC	S	100	100	100	2.000	○	○

Configuration

J4 Ang	Lefty/R	Flip/No	Above	J6 Ang
● +	○ L	○ F	○ A	○ +
○ -	○ R	○ NF	○ B	○ -

Node Outputs

2050

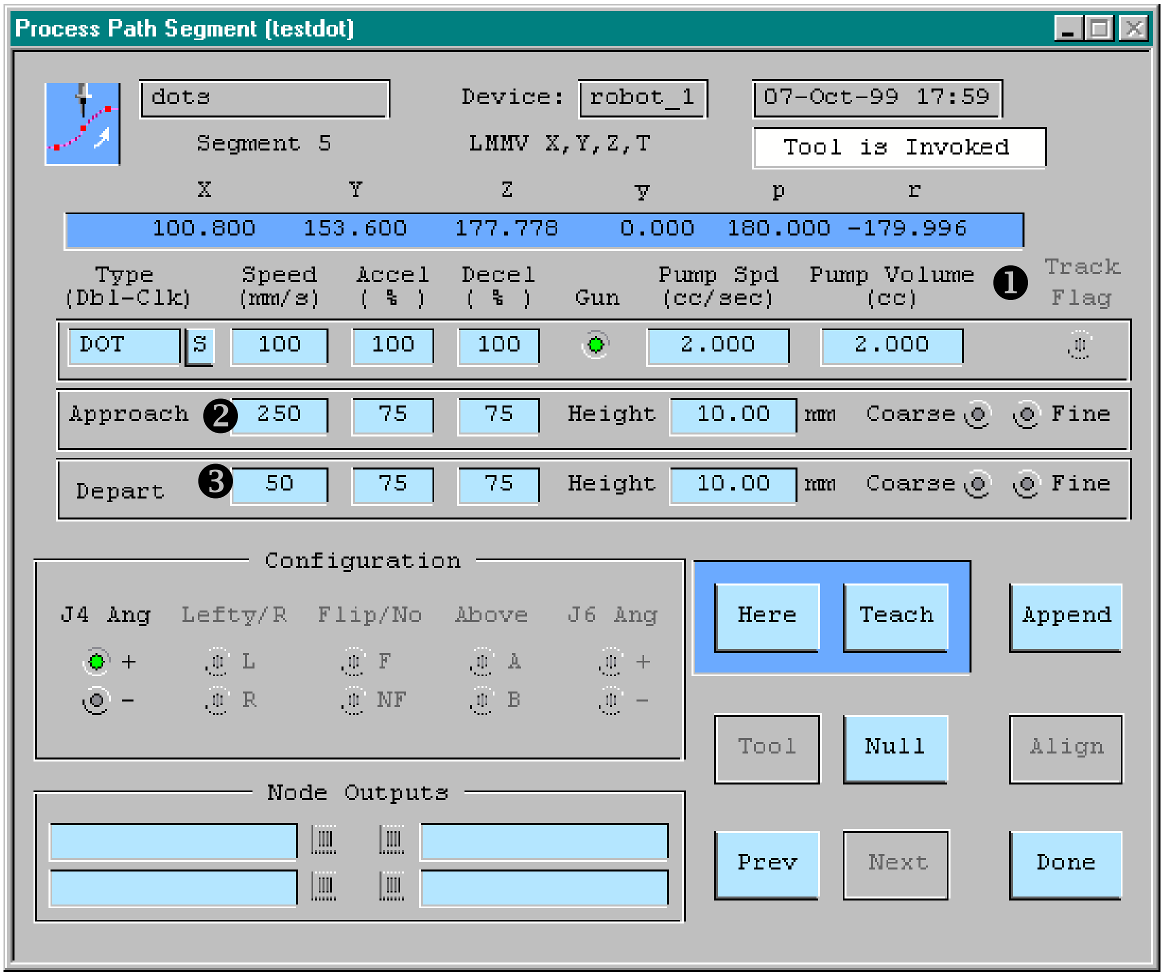
Figure 3-5. Precision Point Path Segment

Process Path Precision Point Segment Options

❶ Select **Prec** if proper arm configuration is required before path movements. This is required for robot devices that have singularities or multiple orientations of the joints. An example would be an AdeptOne with Joint 4 rotation of 540 degrees. If the motion requires Joint 4 to rotate 360 degrees during the path motion, Joint 4 will have to be defined as a precision position before path motion begins. Refer to **"Arm Configuration/Precision Point" on page 37** for more details.

❷ The ◉ **Floating** radio button in the precision point section allows a precision point definition of a tracking location. This allows the joint axis to be defined even if tracking is used with the system.

❸ This section works in conjunction with the precision point selections. Its purpose is to specify the proper arm configuration before moving to the precision point specified. Refer to **"Arm Configuration/Precision Point" on page 37** for more details.



Process Path Segment (testdot)

dots

Segment 5

Device: robot_1

LMMV X,Y,Z,T

07-Oct-99 17:59

Tool is Invoked

X	Y	Z	y	p	r
100.800	153.600	177.778	0.000	180.000	-179.996

Type (Dbl-Clk)	Speed (mm/s)	Accel (%)	Decel (%)	Gun	Pump Spd (cc/sec)	Pump Volume (cc)	Track Flag
DOT S	100	100	100	●	2.000	2.000	○
Approach	250	75	75	Height 10.00 mm	Coarse ○	○ Fine	
Depart	50	75	75	Height 10.00 mm	Coarse ○	○ Fine	

Configuration

J4 Ang	Lefty/R	Flip/No	Above	J6 Ang
● +	○ L	○ F	○ A	○ +
○ -	○ R	○ NF	○ B	○ -

Node Outputs

Figure 3-6. Dot Path Segment

Process Path Dot Segment Options

❶ Select ● **Gun** if the dispensing gun should be turned on for the dot process. If the gun is turned off, no dot material will be dispensed.

In the **Pump Spd / Dwell** data box, enter a time or a pump speed for material to be applied during a dot process. Different text will be displayed based on the type of equipment being used in the application. For all devices but a servo, the dwell data will be required. This display is shown with a servo device selected.

The **Pump Volume / Bead%** values are intended for use with devices such as Analog Variable Nozzles and Servo Pump control. Enter the volume to be dispensed or the % nozzle open during the dot process. This entry will not appear unless the analog or servo device is selected in the signal database.

❷ The approach parameters are provided for situations in which the robot is required to move above the position and then down into position to dispense the dot. In these cases the speed, acceleration, deceleration, height, and position nulling (coarse, fine) are required entries. If a height value of 0 is specified, these values are not required. Please note that in the initialization database the default values for these entries can be changed.

The move to the dot position is a joint-interpolated motion. This allows the fastest motion to that point. Therefore the gun will be turned off during this motion. The move to the next path segment will also be a joint-interpolated motion with the gun off.

❸ The depart parameters are provided for situations in which the robot is required to move above the dot position when exiting the dot process. In these cases the speed, acceleration, deceleration, height and position nulling (coarse, fine) are required entries. If a height, value of 0 is specified, these values are not required. Please note that in the initialization database the default values for these entries can be changed.

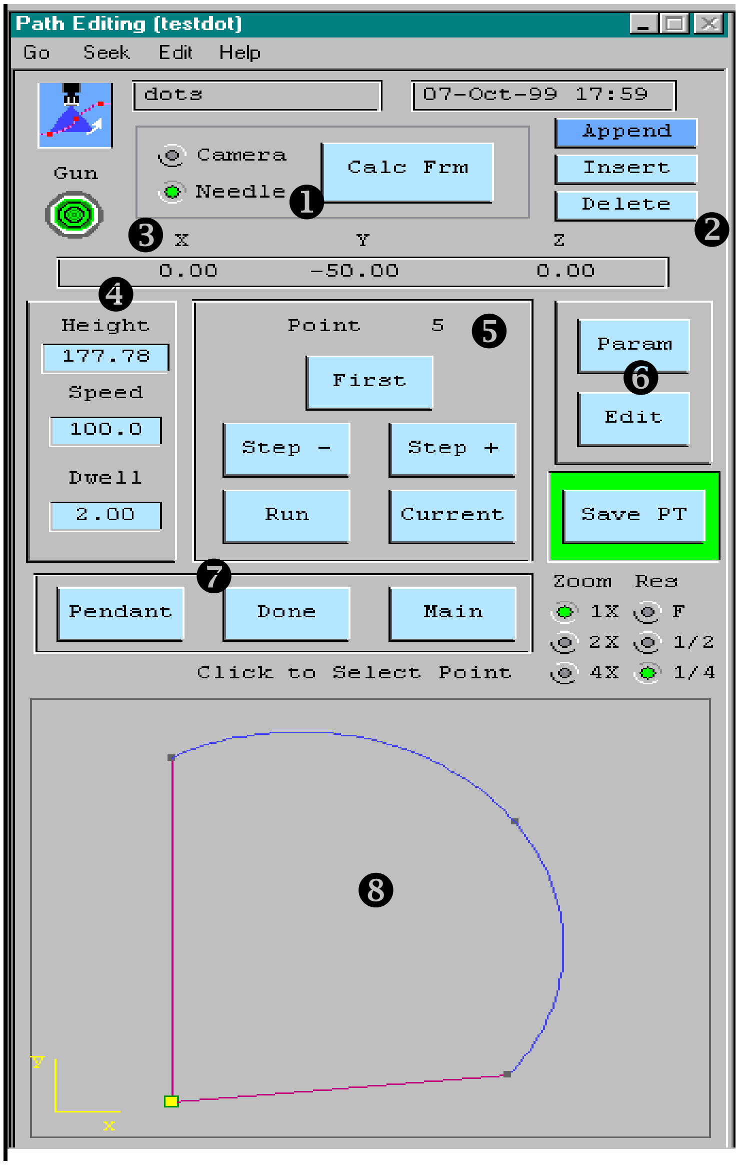
Path Editing / Vision Editing

Provided with PathWare is a new editing feature that allows easier teaching of positions. A new feature is the ability to use an arm-mounted camera to teach positions. Also provided is the ability to step between points and move around the path, all from within one menu. The path editing menu can be found in the edit pulldown or from the Process Path menu.

Edit ⇨ Path Editing

Or from the Process Path Database.

Edit ⇨ Process Path ⇨ **Path Edt**



dots

07-Oct-99 17:59

○ Camera
● Needle

X	Y	Z
0.00	-50.00	0.00

Height 177.78

Speed 100.0

Dwell 2.00

Point 5

● 1X ○ F
○ 2X ○ 1/2
○ 4X ● 1/4

Click to Select Point

Figure 3-7. Path Editing Menu

Path Editing Menu Options

❶ The ◉ **Camera Mode** radio button allows the user to move to positions that are in camera coordinates instead of tool coordinates. This mode allows the user to move to positions that will be viewed by the camera.

The ◉ **Nozzle Mode** radio button allows the user to move to positions that are in tool or world coordinates instead of camera coordinates. This mode allows the user to move to positions that will be executed in the actual cycle.

The **Calc Frm** button will execute a specified control sequence that is set in the Path Editing Parameters menu. (See **"Path Editing Setup Parameters" on page 33** for more details) This allows the user to force the frame operation to occur to allow teaching of positions relative to a frame of reference.

❷ Press **Append** to add another path point to the end of the current path selected. The graphics display will show an error message until this new path point is taught.

Press **Insert** to add another path point before the current path point selected.

Press **Delete** to delete the current selected point. You can determine this point by looking at the graphic display or moving to that position by pressing Current.

❸ Positional Display: This display shows the coordinates of what the actual position will be taught to. This will be relative to reference frames if specified.

The ***GUN*** Icon is provided to show the state of the gun based on the current selected position.

❹ The ***Height*** input field will modify the height of the motion to the current point selected. Please note that after the point is entered, a pop-up window will appear to give you the option to modify all of the points in the path to this height.

The ***Speed*** input field will modify the speed of the motion to the current point selected. This speed is the same as the speed field in the Process Path menu.

The ***Dwell*** input field allows the user to specify a dwell time at the position that is currently being taught.

❺ The ***Point #*** is provided to indicate the currently selected position. In addition, the path graphic provided in the bottom of the menu indicates the point by showing it as a yellow square.

Press **First** to move the robot to the first path position. Care must be taken with this operation. The robot will move from its current position directly to the first path point. If there is an obstruction between these points, a collision may result.

Press **Step –** to move the robot to the previous position. The robot will move directly from its current position to the previous point in the path.

Press **Step +** to move the robot to the next position. The robot will move directly from its current position to the next point in the path.

Press **Run** to move the robot through the taught path based on the speed specified in each of the points as well as the speed setting under the PARAM button. The robot will move directly from its current position to the first point in the path.

Press **Current** to move the robot to the currently selected point. A point can be selected in the graphic screen by clicking on a point. The currently selected point is indicated by a yellow square.

❻ Press **Param** to branch to a parameter entry menu. This menu allows the user to change motion speeds and gun compensation timing values. (**See “Path Editing Parameters” on page 36**.)

Press **Here** to teach the currently selected point. The positions are always saved as World/Tool coordinates even if you are editing in camera mode.

Press **Edit** to branch to the segment editing menu. This is the same menu as in the Process Path Segment menu. It allows you to edit all of the specific path point parameters. (**See “Process Path Segments” on page 23**.)

❼ Press **Pendant** to Pop up the MCP Emulator menu. This menu will function in task 6, which allows you to move the robot and edit the path data in the Path Editing menu at the same time. It replaces the *Status2* menu on the screen.

Press **Done** to branch to the Operator’s Control Menu.

Press **Main** to branch to the Process Path main menu.

The ***ZOOM*** parameters are provided when vision is being used for teaching. These buttons allow the vision image to be zoom to aid in finer teaching positions.

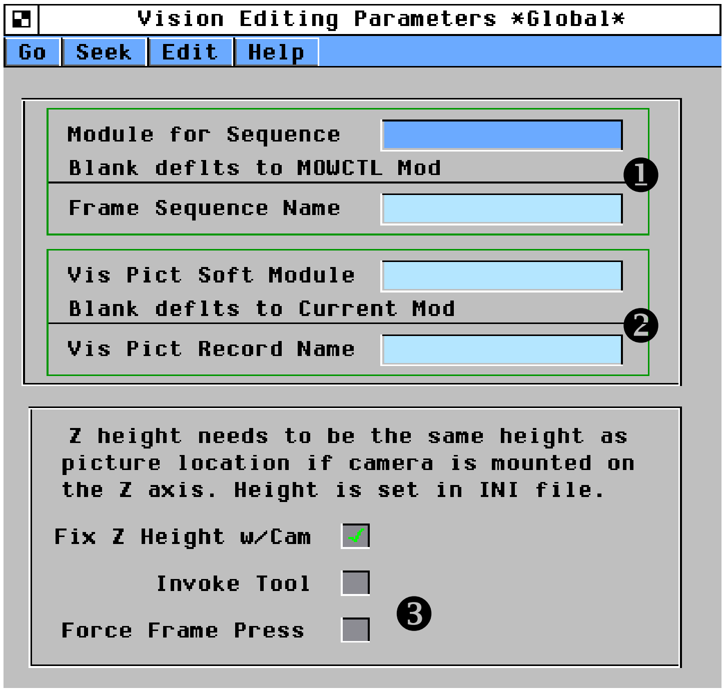
The ***RES*** parameters are provided for use when vision is being used with Adept Windows. The vision image is updated faster at lower resolutions on the PC.

❽ The graphic display area is provided to allow the user to view the current path that is being edited, as well as a means to maneuver easily to the points of interest. Each point is indicated by a square box. The starting point is always shown as a green box. The ending points indicated by the color red. The yellow square indicates the currently selected point. The User can select a different point by clicking near the point of interest. Once the point is selected, the user can move to that point by pressing the ***Current*** button. The parameters for that position will then be displayed. The Initialization database also allows the user to automatically move to the position that is selected by the graphic if enabled on.

Path Editing Setup Parameters

The new Path Editing menu allows several new features to be implemented relative to path teaching from within PathWare. These features include Vision Teaching, Frame of Reference definition, and Forced Tool Offsets. The *Setup* menu allows the user to define which path teaching features will be active when using the Path Editing Menu. The Setup menu can be accessed only in the Process Path Menu. This is done so that the operator access levels will prohibit any viewing of this menu.

Edit ⇨ Process Path ⇨ Setup



Module for Sequence
Blank deflts to MOWCTL Mod
Frame Sequence Name

Vis Pict Soft Module
Blank deflts to Current Mod
Vis Pict Record Name

Z height needs to be the same height as picture location if camera is mounted on the Z axis. Height is set in INI file.

Fix Z Height w/Cam ☑

Invoke Tool □

Force Frame Press □

Figure 3-8. Path Editing Setup Parameters

Path Editing Setup Options

❶ The input field for "***Module for Sequence***" is provided to specify the module in which a control sequence to establish a frame resides. This is required for the ***Calculate Frame*** button to function on the Path Editing Page. The idea is to have a control sequence execute a sequence that runs in the robot task to determine the frame of reference. This can be done either for Vision Frames or Conveyor Frames. If the input field is left blank, the *MOWCTL* module will be use for the control sequence.

The input field for "***Frame Sequence Name***" is the control sequence that will be executed when the ***Calculate Frame*** button is pressed.

❷ The input field for "***Vis Pict Soft Module***" is provided to specify the module in which the vision database resides. This is required for applications that use the vision teaching tools. These tools need to be able to access the vision picture record to establish the image data as well as the camera calibration. If this field is left blank, the current specified module will be used.

The input field for "***Vis Pict Record Name***" is provided to specify the name of the vision picture to be used for the cross hair display. The PathWare vision teaching utility will draw a Cross Hair on the vision image as well as force a live display. Vision teaching allows the user to move the robot while looking at the vision image to define a path position.

❸ The ☑ ***Fixed Z Height w/Cam*** check box will force the Z position of the robot tooling to a specified height that is entered in the initialization database.

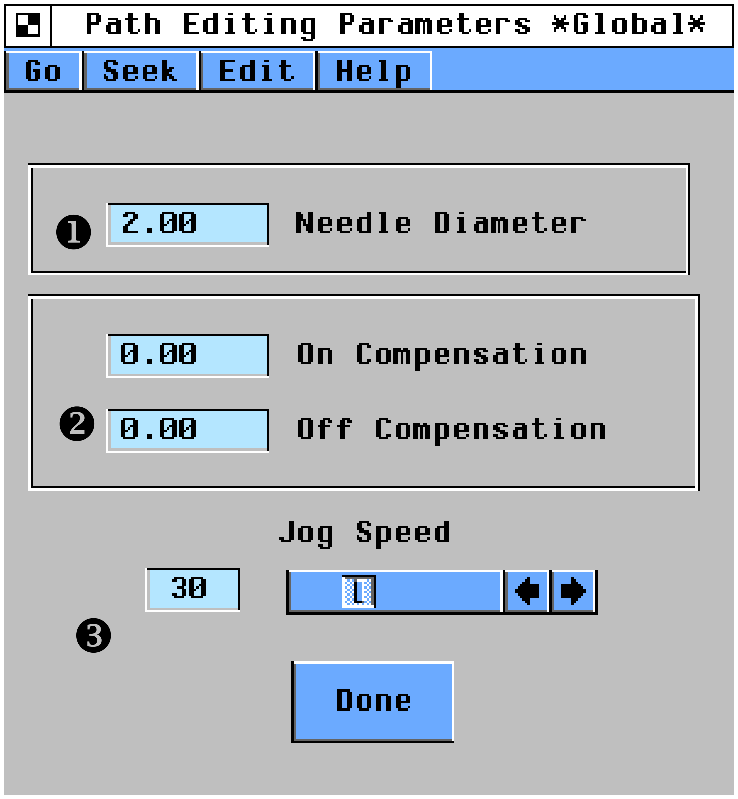
The ☑ ***Invoke Tool*** check box will invoke the robot tool offset when teaching positions in the Nozzle Mode. This minimizes mistakes when teaching points without a tool invoked.

The ☑ ***Force Frame Press*** check box will allow path teaching only after the ***Calculate Frame*** button has been pressed. This minimizes the chances of positions that are not correct due to an invalid frame of reference.

Path Editing Parameters

Provided with PathWare is a new editing feature that allows easier teach of positions. A new feature is the ability to use an arm-mounted camera to teach positions. Also provided is the ability to step between points and move around the path all from within one menu. Please refer to the *MotionWare User's Guide* for more information on database modules.

Edit ⇨ Path Editing ⇨ Param



2.00 Needle Diameter

0.00 On Compensation

0.00 Off Compensation

30

Figure 3-9. Path Editing Parameters

Path Editing Parameters

❶ The input field for "*Needle Diameter*" is provided for use with the vision teaching tool. When this field is specified, the needle diameter is drawn on the vision display with the cross hairs. This is useful when moving the robot in camera mode.

❷ The input field for *"**On Compensation**"* is provided for easy access to the gun compensation timing values that are located in the PathWare Signal Database. (**See "PathWare Signals" on page 39**.)

The input field for *"**Off Compensation**"* is provided for easy access to the gun compensation timing values that are located in the PathWare Signal Database.

❸ The ***Jog Speed*** parameters are provided to control the motion speed of the robot when using the Path Editing Menu. This percentage of speed is applied as a multiplier to the speed setting of each point that is taught. This speed setting is used with the *First, Step, Current,* and *Run* button on the menu.

Arm Configuration/Precision Point

You can control the configuration of the arm during a move to a location. This parameter applies to robot devices that have rotary axis and multiple joint positions to define a location. **Figure 3-10** shows how a SCARA robot can reach a point with the "elbow" pointing in different directions.

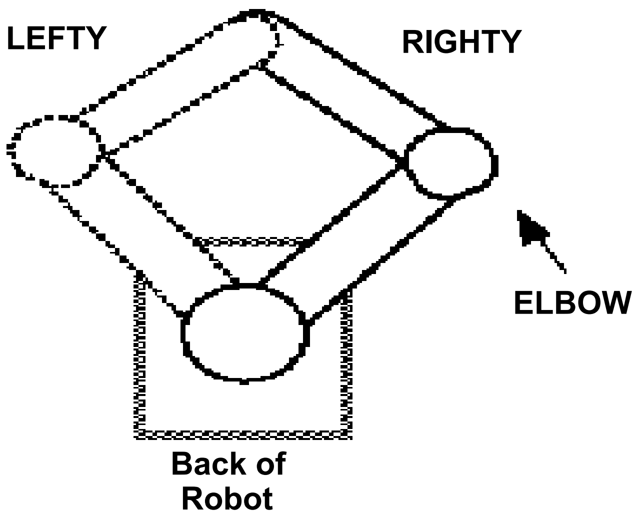


Figure 3-10. Robot Arm Configurations

In many cases, the arm configuration that the robot was in when you taught the location may not be the optimal configuration[1] to use when the robot moves back to that location from another location (arm configuration is recorded when the **Here** button is pressed).

If the arm configuration at a given point is critical (for example, to avoid running into obstructions in the workcell, or when highest precision is required), select ◉ **Precision Move** on the Path Segment menu. (**Figure 3-4**). The configuration table allows the user to select the proper arm configuration for that location. There will be different selections available based on the robot device module being used. Once the configuration is selected the robot will move to that location as a precision point. The selection of the precision move radio button will force a precision motion to that point every time. The robot gun signals cannot be activated during this motion and are not selectable. This option should be selected only when you know that the robot can safely change arm configuration when moving to this location from any other possible location. Path points can be taught before the precision point move. This may be required for a safe motion to that point. This option is needed for any path moves that require 360°of joint 4 motion.

The selections ***J4 Ang, Lefty/R, Flip/No, Above/Below, J6 Ang*** are provided to describe the joint rotations necessary for the device to reach the specified location. Available selections depend on which robot device modules are being used. The six-axis PUMA device module is currently the only robot module that will use all of the selections. In teaching mode, the setting for these selections are determined based on the current robot configuration. For example, the robot configuration may be move in righty rather than lefty. Or the current position of the J4 may be on the high side of the rotation when teaching, which may require an unwind during path motion. (Refer to the *V^{+} Language Reference Guide* for more information regarding the Righty/Lefty, Flip/ Noflip, Above/ Below functions for the arm configuration.)

If none of the options are selected, the arm will move to the location using the current configuration. (If the location cannot be reached in the current configuration, execution will stop and the operator will be alerted.)

[1] The optimal configuration is the one that requires the least amount of joint movement. In most cases, the optimal configuration is achieved by not changing arm configuration during a move.

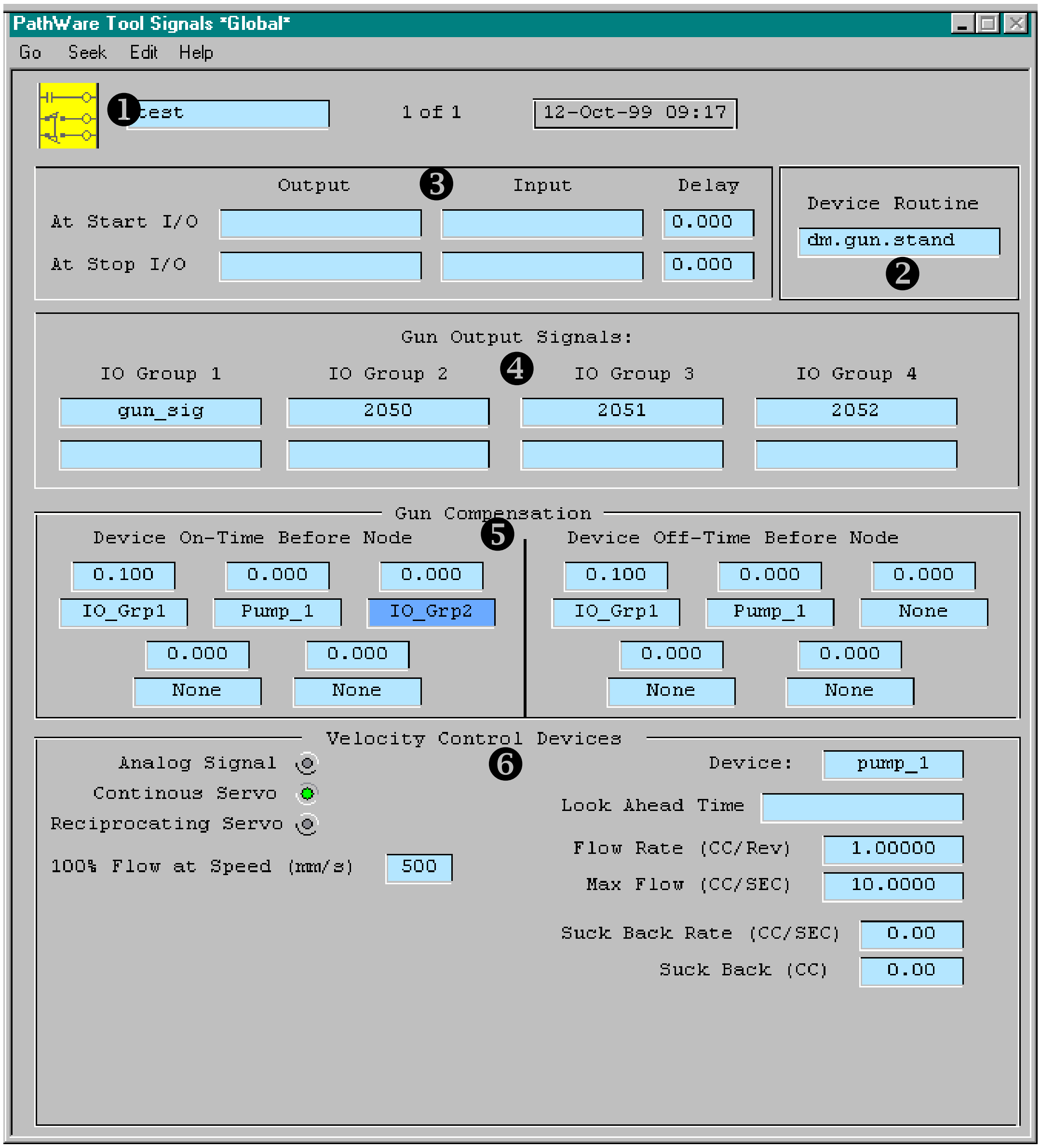
PathWare Signals

To create a new pathware signal record:

Edit ➪ **PathWare Signals** ➪ ***press F2*** ➪ ***enter record name***

To edit an existing signal record:

Edit ➪ **PathWare Signals** ➪ **Seek** ➪ **Index** ➪ ***dbl clk on signal record***



test 1 of 1 12-Oct-99 09:17

	Output	Input	Delay
At Start I/O			0.000
At Stop I/O			0.000

Device Routine: dm.gun.stand

Gun Output Signals:

IO Group 1	IO Group 2	IO Group 3	IO Group 4
gun_sig	2050	2051	2052

Gun Compensation

Device On-Time Before Node:
0.100 IO_Grp1
0.000 Pump_1
0.000 IO_Grp2
0.000 None
0.000 None

Device Off-Time Before Node:
0.100 IO_Grp1
0.000 Pump_1
0.000 None
0.000 None
0.000 None

Velocity Control Devices

Analog Signal ○
Continous Servo ●
Reciprocating Servo ○

100% Flow at Speed (mm/s) 500

Device: pump_1
Look Ahead Time
Flow Rate (CC/Rev) 1.00000
Max Flow (CC/SEC) 10.0000
Suck Back Rate (CC/SEC) 0.00
Suck Back (CC) 0.00

Figure 3-11. PathWare Signal Record

PathWare Signal Options

❶ Enter a name for this record. The numbers directly to the right of the name show the current record and the total number of records in the database.

The last date this record was modified is shown to the right of the numeric field.

❷ The ***Device Routine*** field is to allow custom user routines to handle gun signals, analog control, or servo pump control beyond the standard routine provided. The default is "dm.gun.standard".

❸ Enter digital signals to be used at the beginning or end of a path. These signals are intended for tooling such as cylinders that need to be actuated before the path and after completion of the path. Input signals are provided to check for the tooling states before the path begins. Delay times are provided for a delay period or, if an input signal is specified, the maximum amount of time to wait for the input signal before an error is generated. All input fields can use the variable database for specifying signal numbers. If the actual signal number is to be entered, provide an @ character before the signal number (example @5).

❹ Enter a digital signal number to actuate outputs based on the dispense gun On/Off specifications at node points. The signals are in four groups to allow the user the most flexibility in control of signals. The gun compensation keywords ***IO_Grp1***, ***IO_Grp2***, ***IO_Grp3***, and ***IO_Grp4*** are provided in the gun compensation area for control of these different groups of signals. The Variables Database can be used to specify the signal numbers.

❺ The Parameters in this area are for gun compensation or device timing relative to changes of gun states at node points. Gun compensation allows signals and actions to occur relative to node points. Generally, it is used to turn the gun on or off before a point so that the flow is correct at that point. In version 3.3, the user can enter in negative timing values for operations to occur a time period after the node point.

NOTE: The Device Routine, On-Time, and Off-Time parameters must be set for any gun control to work. If there is no compensation, then the time setting should be 0. The Gun_Sigs must also be defined in the specified areas for proper operation.

The ***Device On-Time Before Node*** area allows Five fields to enter a value of time in seconds for which an action will occur before the node point where the gun is turned on. Below that value a keyword display is shown, which allows you to determine what action you wish to occur. In this case the signals in IO_Grp1 are actuated 0.1 second before the node and the servo pump, and IO_Grp2 is started at the node point. The keywords can be selected by double-clicking on the keyword field.

The ***Device Off-Time Before Node*** area is the same as the previous except for allowing the actions to occur before an off state.

❻ This area is for selection of variable control of analog or servo devices relative to the robot's tool tip velocity. The user can select either ***variable nozzle*** for analog control or ***servo pump control***. The menu display **Figure 3-11** shows the servo pump option selected.

Servo Pump Parameters

Press ◉ **Continuous Servo** to enable the PathWare servo pump options. Note that the parameters will not function unless the servo pump option has been turned on in the initialization database.

Enter the **100% Flow Rate at Speed (mm/sec)** in the table provided. This value is used for scaling of pump speeds with the ratio provided by the bead percentage. This value will affect the linearity of the bead sizes at slower speeds.

The **Look Ahead Time** value will provide future velocities for the analog and servo pump control calculation. The calculated future velocity assumes the robot device is capable of the accelerations specified in the defined path. This value can also be entered in the Variable database.

The **device** selection field is a keyword selection of the robot device number. Double-click on this field and a selection list will appear. If the standard PathWare initialization database values are used for the servo pump, the device name will be **PUMP_1**. Be sure the Servo pump options in the initialization database are turned on before defining the values in the pathware signal database.

Enter the **Flow Rate (CC/REV)** in the provided data field. This value is the scaling factor for determining degrees of rotation to establish the volume and the rate.

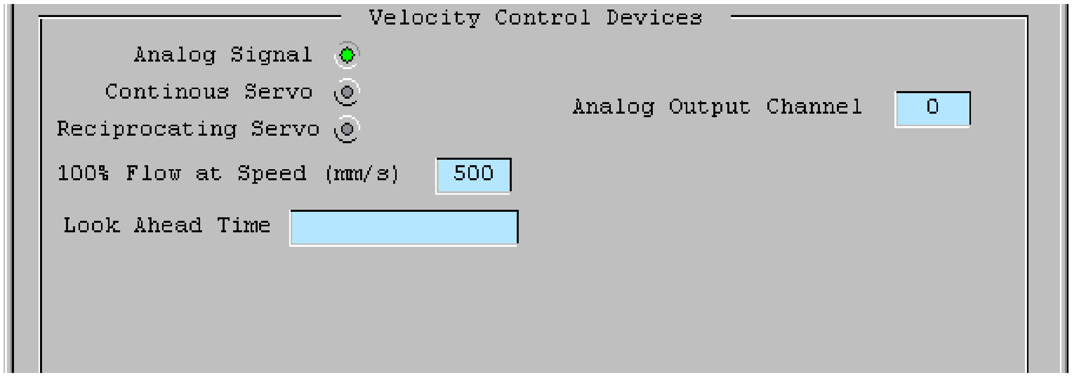
Enter the **Max Flow (CC/SEC)** in the provided data field. This value provides error detection to assure the motor/pump does not exceed these values.

Enter the **Suck Back Rate (CC/SEC)** in the provided field. This data is for applications that require a reverse rotation of the servo to prevent dripping. The value is the rate at which the suck back occurs.

Enter the **Suck Back (CC)** in the provided data field. This value is the volume of material the user wishes to suck back.

Analog Parameters

Press the ◉ **Analog Signal** push button to enable the variable flow nozzle. The following menu parameters appear.



Velocity Control Devices

Analog Signal ●

Continous Servo ○

Reciprocating Servo ○

Analog Output Channel 0

100% Flow at Speed (mm/s) 500

Look Ahead Time

Figure 3-12. Analog Menu Parameters

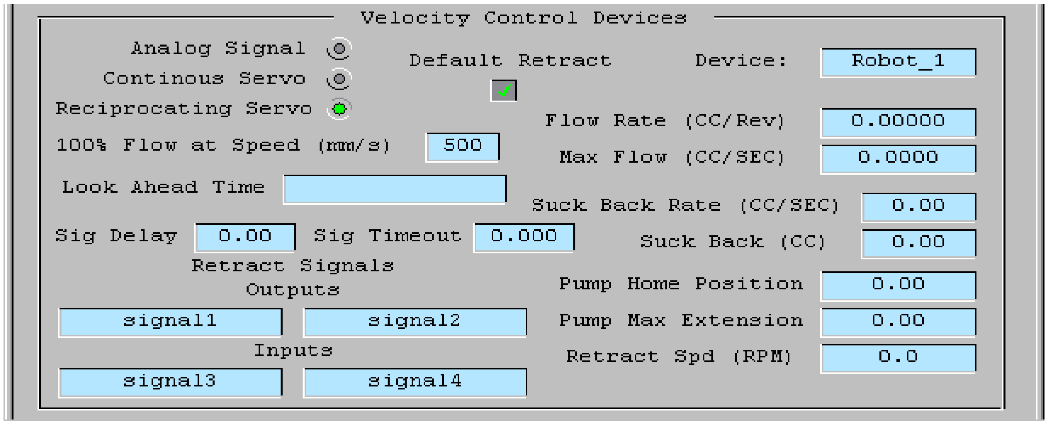
Enter the **Analog Output Channel** in the provided data field. This software assumes the use of the Xycom XVME-540 Analog I/O board. If a different third-party board is to be used, a software driver will have to be written for this interface. The Xycom part number is 70540-001; Xycom can be reached at 800-289-9266.

In the **100% Flow at Speed (mm/s)** data box, enter the robot speed at which the flow gun should be fully open. Robot speed, the 100% flow rate, and the bead size are used to generate digital output to the digital-to-analog converter.

The **Look Ahead Time** value will provide future velocities for the analog and servo pump control calculation. The future calculated velocity assumes the robot device is capable of the accelerations specified in the defined path. This value can be entered in the Variables database.

Reciprocating Servo Pump

Press the ◉ **Reciprocating Servo** push button to enable the reciprocating servo. Please note that the parameters will not function unless the servo pump option has been turned on in the initialization database. The following menu parameters appear.



Velocity Control Devices

Analog Signal ○
Continous Servo ○
Reciprocating Servo ●
Default Retract ☑
Device: Robot_1
Flow Rate (CC/Rev) 0.00000
100% Flow at Speed (mm/s) 500
Max Flow (CC/SEC) 0.0000
Look Ahead Time
Suck Back Rate (CC/SEC) 0.00
Sig Delay 0.00
Sig Timeout 0.000
Suck Back (CC) 0.00
Retract Signals
Outputs
signal1 signal2
Pump Home Position 0.00
Pump Max Extension 0.00
Inputs
signal3 signal4
Retract Spd (RPM) 0.0

Figure 3-13. Reciprocating Servo Selections

Reciprocating Servo Parameters

Enter the **100% Flow Rate at Speed (mm/sec)** in the table provided. This value is used for scaling of pump speeds with the ratio provided by the bead percentage. This value will affect the linearity of the bead sizes at slower speeds.

The **Look Ahead Time** value will provide future velocities for the analog and servo pump control calculation. The calculated future velocity assumes the robot device is capable of the accelerations specified in the defined path. This value can also be entered in the Variables database.

The **Sig Delay** field is a delay that will be applied should no input signals be specified. This delay is applied before the pump starts any motion. This is true both for positive direction for pumping and for retracting the pump.

The **Sig Timeout** field is to specify a maximum amount of time to allow the input signals to get set to their proper state. After the maximum time has elapsed an error message will appear, and the robot motion will stop.

The **Retract Signals** parameter area allows the user to specify a maximum of two output and input signals. The Output fields are specified in the expected state for when the pump is to be retracted. These states are typically set when the pump is not metering flow. A negative output value can be used for a reversed output state. The Input fields are the expected state when the pump is to be retracted. These input can be specified with a negative value to reverse the state.

The **Default Retract** check box is used to force the output states to the retracted position when the pump is not metering flow. If not checked, the outputs will be set only when the pump is retracting.

The **device** selection field is a keyword selection of the robot device number. Double-click on this field and a selection list will appear. If the standard PathWare initialization database values are used for the servo pump, the device name will be **PUMP_1**. Be sure the Servo pump options in the initialization database are turned on before defining the values in the pathware signal database.

Enter the **Flow Rate (CC/REV)** in the provided data field. This value is the scaling factor for determining degrees of rotation to establish the volume and the rate.

Enter the **Max Flow (CC/SEC)** in the provided data field. This value provides error detection to assure the motor/pump does not exceed these values.

Enter the **Suck Back Rate (CC/SEC)** in the provided field. This data is for applications that require a reverse rotation of the servo to prevent dripping. The value is the rate at which the suck back occurs.

Enter the **Suck Back (CC)** in the provided data field. This value is the volume of material that is to be suck back.

The **Pump Home Position** field allows the user to specify the position the pump will retract to. This field is entered typically in degree units depending on the data specified in the Adept SPEC utility.

The **Pump Max Extension** parameter is the distance the pump can travel before a retract will occur after the completion of the cycle. This value should be about a full cycle before the end of stroke to assure that the pump will not reach its maximum extension.

The **Retract Speed** is the speed in RPMs that the motor will retract to the home position.

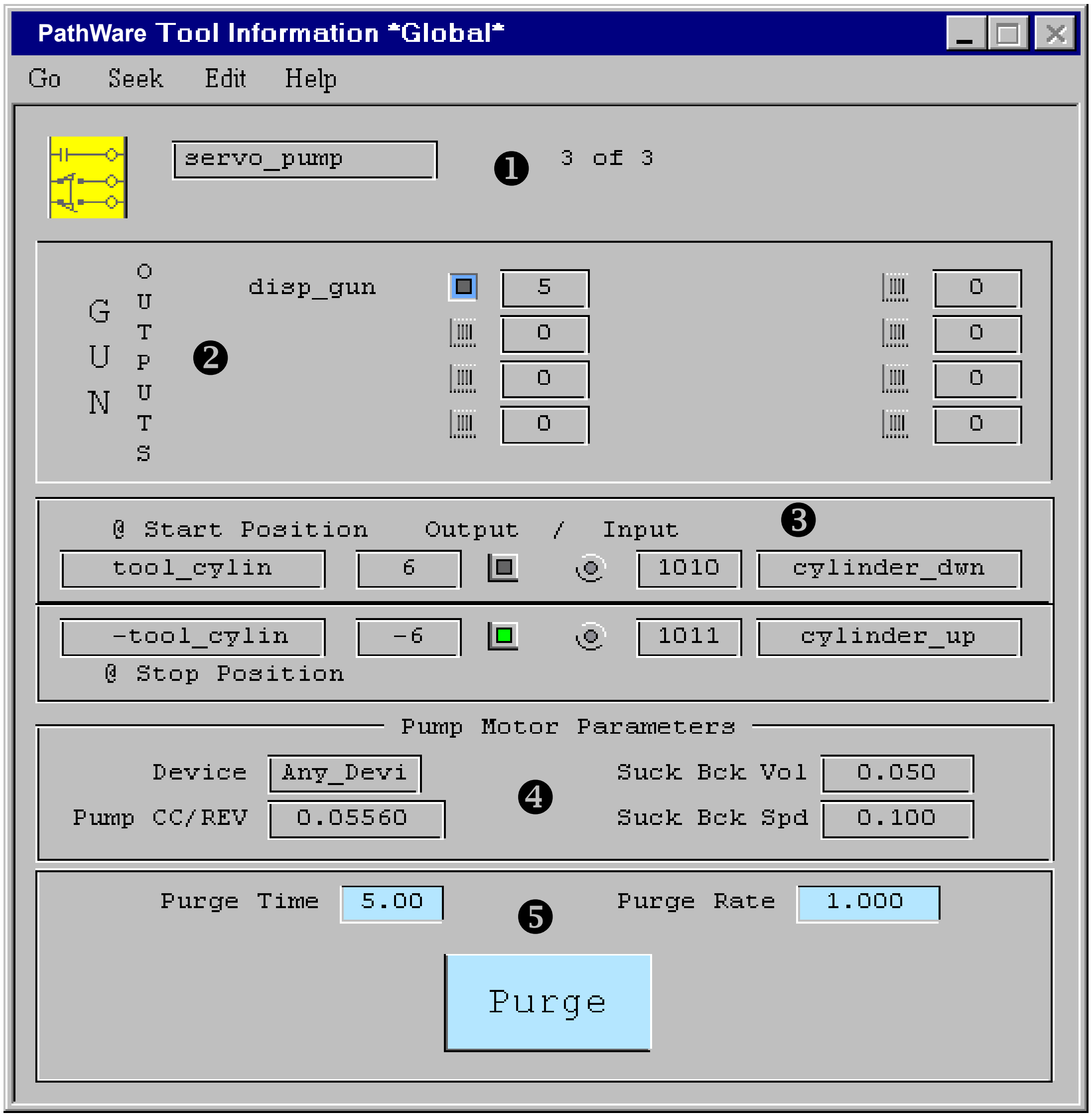
Manual PathWare Signal Control

The following menu page will be displayed showing the state of all of the defined signals from the PathWare signal database. It allows manual operation of those signals as well as a manual purge cycle with full support for analog and servo devices.

IO ·➪ PathWare Controls

To view another signal record:

Edit ➪ PathWare Signals ➪ Seek ➪ Index ➪ ***dbl clk on signal record***



servo_pump 3 of 3

GUN OUTPUTS

disp_gun ■ 5 0 0 0 0 0 0 0

@ Start Position	Output		/	Input	
tool_cylin	6	■	○	1010	cylinder_dwn
-tool_cylin	-6	🟩	○	1011	cylinder_up
@ Stop Position					

Pump Motor Parameters

Device	Any_Devi	Suck Bck Vol	0.050
Pump CC/REV	0.05560	Suck Bck Spd	0.100

Purge Time 5.00 Purge Rate 1.000

Figure 3-14. Manual PathWare Signal Control

Manual PathWare Signal Options

❶ Name of currently selected pathware signal record.

❷ Displays gun signal names and output values. Press the push button to change the signal state for manual operation. Please note there are up to eight values that will be displayed.

❸ Displays I/O at start and stop of path. Allows manual operation of output and displays input signal values.

❹ Displays analog or servo parameters that are set up in the pathware signal record selected. These parameters can be changed only in the PathWare signal main menu.

❺ Allows purging of the dispense equipment. The values displayed above the purge button will be different depending on the PathWare signal record and what equipment selections are made. These values are defaults in the PathWare initialization database. They can be changed in the menu. However, those changes will be lost after a reboot of the system. Permanent purge value changes must occur in the PathWare initialization database.

Tool Offset

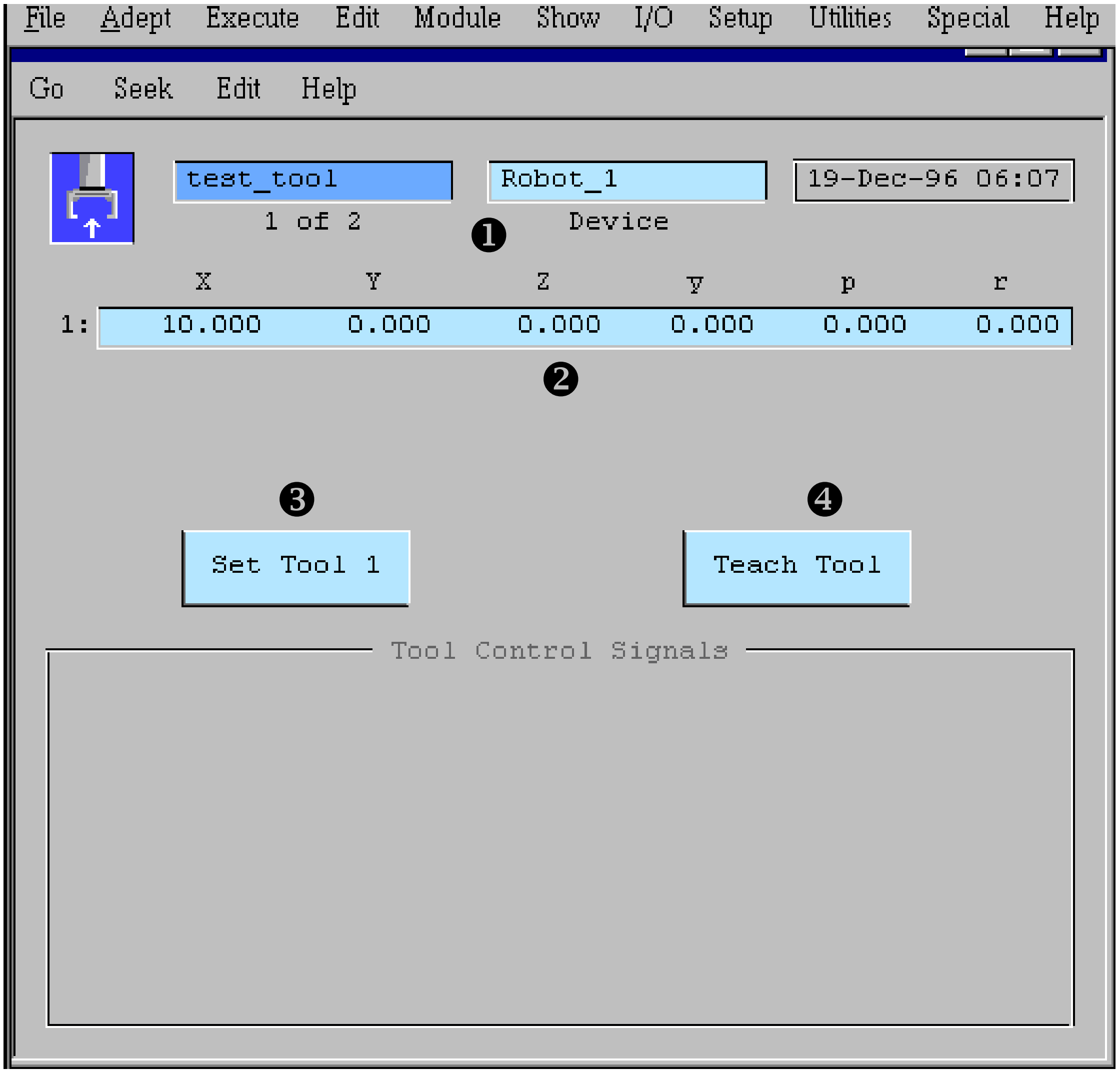
A *tool* is an offset (transformation) that is applied to all robot locations to account for the difference between the robot tool flange center and orientation, and the actual tool tip center and location.

To create a new tool record:

Edit ➪ Tool ➪ *press "F2" ➪ enter record name*

To edit an existing tool record:

Edit ➪ Tool ➪ Seek ➪ Index ➪ *dbl clk on tool record*


File Adept Execute Edit Module Show I/O Setup Utilities Special Help
Go Seek Edit Help
❶
❷
❸
❹
Set Tool 1
Teach Tool
Tool Control Signals

test_tool
1 of 2

Device: Robot_1

19-Dec-96 06:07

	X	Y	Z	y	p	r
1:	10.000	0.000	0.000	0.000	0.000	0.000

Figure 3-15. Tool Offset Menu

❶ Enter a name for this tool record. The numbers indicate the number of this record in the database and the total number of records in the database.

Select the proper robot device number for this tool offset. Double-click on this field to select a different device.

Shows the date this record was last modified.

❷ Shows the offset (transformation) that will be applied to any location the robot moves to while this tool is in effect. These values can be entered directly (based on the engineering data for the dispensing gun), or taught using the teach routine (option ❹).

❸ Press to use the values in the current record as the robot tool transformation.

❹ Press to use the manual control pendant to define a tool transformation. The tool offset teaching utility is described in detail in **Teaching Tool Transformations**.

To test a tool transformation, Press SET TOOL button to invoke the tool offset, then select TOOL state from the manual control pendant and choose "RZ" from the mode control buttons. Move the robot using one of the speed bars. The robot should rotate about the axis of the tool.

More documentation is available on tool offsets in the ***MotionWare User's Guide***.

Operator Control Menu

The operator control panel is available as an option to allow the user to enable power and calibrate the robot arm from the main menu. Additionally, button icons more closely resemble actual push buttons similar to machine interface panels.

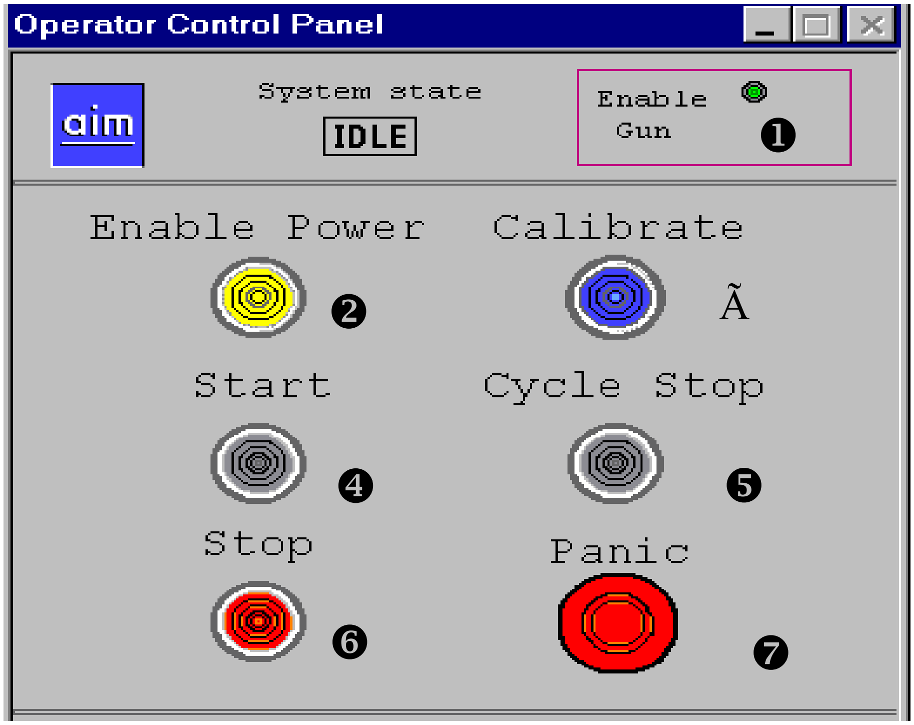


Figure 3-16. Operator Control Panel

Operator Control Panel Options

❶ Enable/Disable dispense gun dry run mode by pressing indicator light. When the indicator light is off, dry run mode is enabled. Gun signals and analog and servo control devices are all disabled during path motion.

❷ Enable/Disable Robot High Power. This button will be yellow when power is enabled and gray when power is off. Pressing the button will change the state of the light.

❸ Calibrate the robot device if the robot is not calibrated with high power on. Calibration will not occur if power is not enabled. Calibration light will be blue when calibration has been completed successfully.

❹ Execute the sequence by the name of *Start* when pressing this button.

❺ Execute the sequence by the name of *Pause* when pressing this button.

❻ Execute the sequence by the name of *Stop* when pressing this button.

❼ Panic will stop robot motion as well as disabling power.

Using PathWare 4

PathWare Statements

This chapter describes the operation of the PathWare Software. First, the DISPENSE, APPLY_DOT, MULTIPLE_DOTS, CHG_DISP_SPEED, CHG_DISP_DWELL, and Z_UP statements are reviewed. Next, details about the path tracking strategy, gun compensation, path compensation, path checking, conveyor tracking, database editing, and pendant teach routines are presented. Finally, example applications are presented to demonstrate the use of AIM PathWare.

PathWare includes the DISPENSE, APPLY_DOT, MULTIPLE_DOTS, CHG_DISP_SPEED, CHG_DISP_SPEED, and Z_UP statements. The statements are discussed below.

DISPENSE

The statement guides the robot along a user-defined path. This path can be used for dispensing, grinding, and many other applications. New with revision 3.3 B, Dot points are now supported. The statement's syntax is as follows, where the braces ({ . . . }) define optional clauses:

```
DISPENSE {APPROACH --path--}
ALONG --pw path-- {DEPART --path--}
{IF SIG ON --variable--} {HNDSHK WAIT --variable-- PATH
STATE --variable--}
```

The statement performs the following steps:

1. If an approach path is specified, move the robot along that path toward the first PathWare location ({APPROACH --path--}).

2. Follow the path defined in the PathWare record and switch the devices or signals on and off at the appropriate path locations. If a tool is specified in the record, set the tool before moving along the dispensing path (ALONG --pw path--) .

3. If the optional depart path is specified, move along that path away from the dispensing path ({DEPART --path--}).

4. If an optional digital signal or variable is defined, execute the statement only if the signal or variable is in the desired state ({IF SIG ON --variable--}).

5. If the HNDSHK WAIT variable is defined, the robot will move to the first path point and wait until the variable state becomes true before beginning the path. This allows sequential motion to occur with another robot or device along the path. The PATH STATE variable will return the current state of the path {HNDSHK WAIT --variable-- PATH STATE --variable--}.

The --`path--` arguments reference Path database records (not Process Path records). The AIM robot module motion routines handle the approach path and depart path motion control. The path specified signals are always off during these motions. See the ***MotionWare User's Guide*** for details.

The --`pw path--` argument references the PathWare database records. All of the records contain the tool name, frame name, signal name, and tool compensation parameters as well as a path location, speed, motion parameters, gun compensation, and dwell time. The *path analyzer*, which is described later in this chapter, uses these parameters and locations to define the robot path and digital I/O control along the path.

The Tool record specified in the Process Path record provides the tool offset to be used when following this path. (The tool offset defines the gun's tip location with respect to the robot tool flange.) If a tool record is not specified, no tool offset is used for the path.

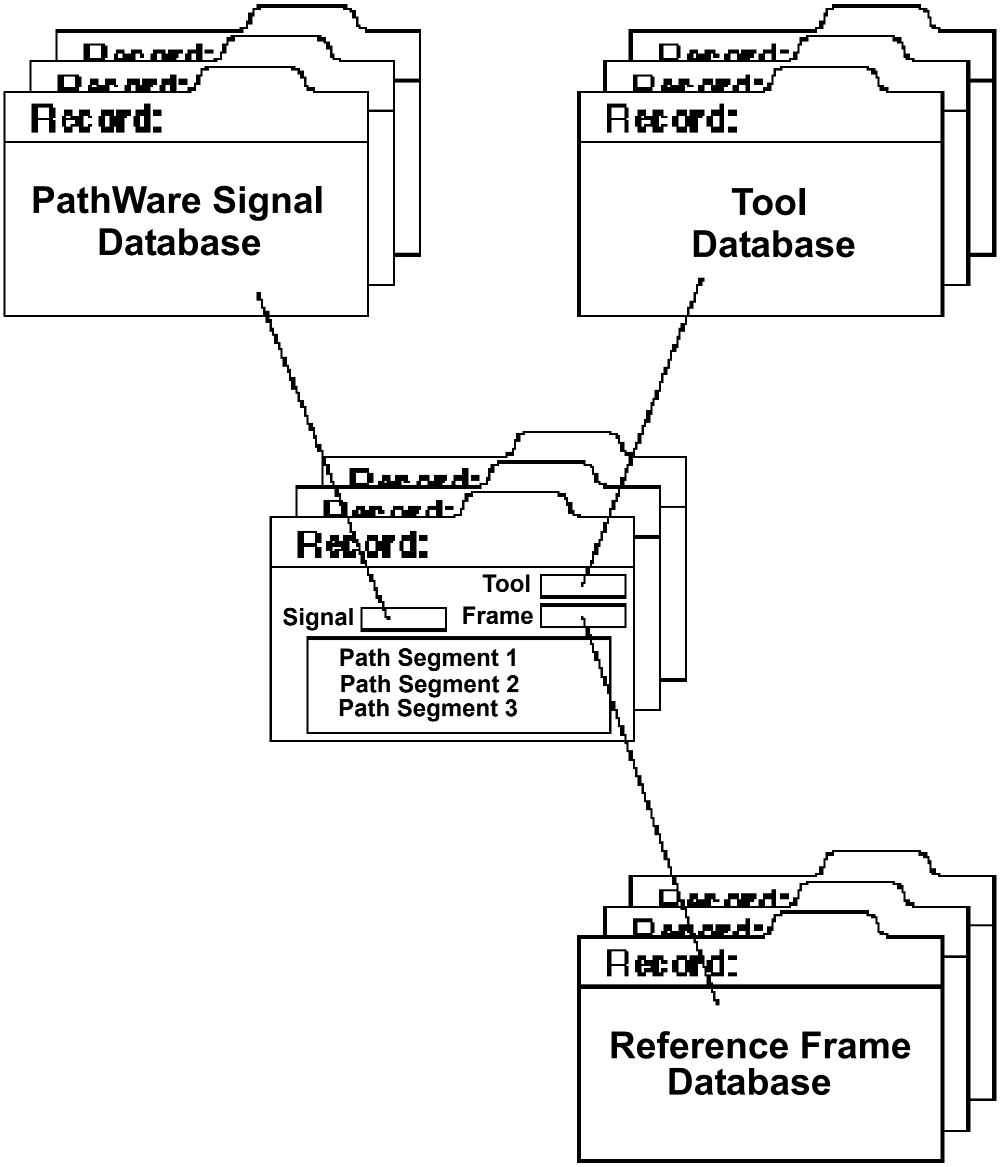


Figure 4-1. PathWare Database Linking Relationship

The **Reference Frame** record provides a reference frame for the Process Path locations. If defined, all of the path's locations are relative to this frame; otherwise, they are relative to the robot's world reference frame. Always define a reference frame before teaching locations that will be relative to that reference frame. Vision Guidance and Conveyor Tracking both use the Frame database which is compatible with PathWare.   Conveyor tracking uses the parameters defined in the Conveyor Database.   In most cases AIM walk-thru training will need to be used for teaching paths relative to vision and moving frames.

Figure 4-1 shows the linking relationship between the Process Path records and the Path, Tool, Reference Frame, and Signal databases. (Not all fields are shown in this illustration.)

APPLY_DOT

The Apply_ Dot statement performs material dispensing tasks that do not require path motion. This statement dispenses material at a single location. This statement supports analog control nozzles as well as servo pumps. Gun operation is specified based on what type of device being used. The syntax of the statement is as follows. The braces (`{...}`) define optional clauses.

```
APPLY_DOT {APPROACH VIA --path--}  AT --location--
{DEPART VIA --path--}  {USING --tool--}  WITH --signal--
{VALVE TIME --variable--}  {NOZZLE%OPEN --variable--
FLOW TIME --variable--}  {PUMP VOLUME(cc) --variable--
FLOWRATE --variable--}   {IF SIG ON --variable--}
{Loop_Thru_Pal --yes/no--}
```

The statement performs the following steps:

1. If a tool is specified, invoke the tool transformation. This transformation should define the offset from the robot tool flange to the gun tip (`{USING --tool--}`).

2. If an approach path is specified, move the robot along that path toward the dispensing location (`{APPROACH VIA --path--}`).

3. Move to the dispensing location, then turn on the dispensing gun for the user-specified dwell time. The location is defined in the MotionWare Locations database (`AT --location--`).

4. If a depart path is specified, move away from the dispensing location along that path (`{DEPART VIA --path--}`).

5. Turn on the valve for a user-specified amount of time. This parameter is used only if the signal record does not contain analog or servo pump parameters. If so, the flowing parameters in the statement are used ({VALVE TIME --variable--}).

6. Open the variable nozzle to a specified percentage and open the valve for a specified amount of time. These parameters are used only for signal records that specify an analog control device ({NOZZLE%OPEN --variable-- FLOW TIME --variable--}).

7. Open the flow valve and run a servo motor until a specified amount of volume occurs at the specified flow rate. The timing for the flow valve and servo pump is established based on the gun compensation values specified in the signal record.

These parameters are used only if the servo pump option has been selected in the signal record ({PUMP VOLUME(CC) --variable-- FLOW RATE --variable-- }).

8. If an optional digital signal is defined, execute the statement only if the signal is in the desired state. This can be used for model selection or gun purging ({IF SIG ON --variable--}).

9. If set to yes, the optional Loop_Thru_Pal argument will go through the entire range of pallet positions performing the gun operations specified.

The Location database stores the apply_dot location and the motion parameters used in moving to the location. In version 3.2E, the pallet parameters in the locations database as well as settle and dwell are functional. The Settle will perform a delay after the robot reaches the position before the gun is activated. The Dwell will perform a delay after the gun has completed. The I/O strategies for the approach and depart positions are now functional. An I/O strategy that is specified for the actual location is not functional in this statement.

MULTIPLE_DOTS

The Multtiple_Dots statement performs material dispensing tasks that do not require path motion. This statement dispenses material at positions that are specified by a process path and motion parameters specified by the location database. This statement supports analog control nozzles as well as servo pumps. Gun operation is specified based on what type of device being used. The syntax of the statement is as follows. The braces ({ . . . }) define optional clauses.

```
MULTIPLE_DOTS {APPROACH VIA --path--} MOTION PARAMS --location--
    USE PATH POINTS -- pw path-- {DEPART VIA --path--}
    {VALVE TIME --variable--}  {NOZZLE%OPEN --variable--
    FLOW TIME --variable--}  {PUMP VOLUME(cc) --variable--
    FLOWRATE --variable--}   {IF SIG ON --variable--} {USE PATH
    DWELL --yes/no--}
```

The statement performs the following steps:

1. If an approach path is specified, move the robot along that path toward the dispensing location ({APPROACH VIA --path--}).

2. Use the motion parameters specified in the location database. The approach, depart, speeds, accelerations, motion configuration, and strategies will be used based on the specified locations record. The transformation and frame of reference will not be used ({MOTION PARAMS --location--}).

3. Move to the dot locations, then turn on the process gun for the user-specified dwell time. The locations are defined in the Process Path database (USE PATH POINTS

`--pw path--`). This statement uses the process path positions to provide the transformation for the dot location. The PathWare Signal database, Tool, and Frame of Reference are used based on the linking to the Process Path database.

4. If a depart path is specified, move away from the dispensing location along that path (`{DEPART VIA --path--}`).

5. Turn on the valve for a user-specified amount of time. This parameter is used only if the signal record does not contain analog or servo pump parameters. If so, the flowing parameters in the statement are used ({VALVE TIME --variable--}).

6. Open the variable nozzle to a specified percentage and open the valve for a specified amount of time. These parameters are used only for signal records that specify an analog control device ({NOZZLE%OPEN --variable-- FLOW TIME --variable--}).

7. Open the flow valve and run a servo motor until a specified amount of volume occurs at the specified flow rate. The timing for the flow valve and servo pump is established based on the gun compensation values specified in the signal record. These parameters are used only if the servo pump option has been selected in the signal record ({PUMP VOLUME(CC) --variable-- FLOW RATE --variable-- }).

8. If an optional digital signal is defined, execute the statement only if the signal is in the desired state. This can be used for model selection or gun purging ({IF SIG ON --variable--}.

9. USE PATH DWELL allows the user to specify the dwell times for the dots from the process database. Specify YES for this instruction to allow the dwell times to be used from the process path.

The Process Path database stores the locations as well as the tool, frame, and signal selections. The motion parameters used in moving to the location are defined in the locations database. The Settle will perform a delay after the robot reaches the position before the gun is activated. The Dwell will perform a delay after the gun has completed. The I/O strategies for the approach and depart positions are now functional. An I/O strategy that is specified for the actual location is not functional in this statement.

SHIFT_DOT

The Shift_ Dot statement performs material dispensing tasks that do not require path motion. This statement dispenses material at a single location. This statement supports analog control nozzles as well as servo pumps. This statement allows the user to shift the coordinates of the dots to be dispensed. This allows the user to define a pattern of dots from the sequence level. The syntax of the statement is as follows. The braces (`{ . . . }`) define optional clauses.

```
SHIFT_DOT World Loc --location-- {By X: --variable--}
{Y: --variable--} {Z: --variable--} {T: --variable--}
{USING --tool--}  {Along Tool --variable--} WITH --signal--
{VALVE TIME --variable--}  {NOZZLE%OPEN --variable--
FLOW TIME --variable--}  {PUMP VOLUME(cc) --variable--
FLOWRATE --variable--}   {IF SIG ON --variable--}
```

The statement performs the following steps:

1. Move to the dispensing location, then turn on the dispensing gun for the user-specified dwell time. The location is defined in the MotionWare Locations database (`World Loc --location--`).

2. Offset position by a specified amount per the arguments (X, Y, Z,T) using the Variables database.

3. If a tool is specified, invoke the tool transformation. This transformation should define the offset from the robot tool flange to the dispense gun tip (`{USING --tool--}`).

4. The Along Tool --variable-- argument allows the user to specify world coordinates or tool coordinates for the offset to be applied.

5. Turn on the valve for a user-specified amount of time.   This parameter is used only if the dispense signal record does not contain analog or servo pump parameters. If it dows contain analog or servo pump parameters, then the flowing parameters in the statement are used ({VALVE TIME --variable--}).

6. Open the variable nozzle to a specified percentage and open the valve for a specified amount of time. These parameters are used only for dispense signal records that specify an analog control device ({NOZZLE%OPEN --variable-- FLOW TIME --variable--}).

7. Open the flow valve and run a servo motor until a specified amount of volume flows at the specified flow rate. The timing for the flow valve and servo pump is established based on the gun compensation values specified in the dispense signal record. These parameters are used only if the servo pump option has been selected in the dispense record ({PUMP VOLUME(CC) --variable-- FLOW RATE --variable-- }).

8. If an optional digital signal is defined, execute the statement only if the signal is in the desired state. This can be used for model selection or gun purging. ({IF SIG ON --variable--})

9. The optional Loop_Thru_Pal argument if set to yes will go through the entire pallet positions performing the gun operations specified.

The Location database stores the dispense location and the motion parameters used in moving to the location. Version 3.2E and above, the pallet parameters in the locations database as well as settle and dwell are functional. The Settle will perform a delay after the robot reaches the position before the gun is activated. The Dwell will perform a delay after the gun has completed. The I/O strategies for the approach and depart positions are functional. An I/O strategy that is specified for the actual location is not functional in this statement.

CHG_DISP_SPEED

The statement allows changes in process path speeds from a sequence statement. This is intended for applications that use automated methods to determine the proper velocity of the path to occur. The velocity changes can occur as a percentage of current speed, which is a temporary adjustment or an offset change which can be saved to the database if selected. The syntax of the statement is as follows. The braces ({...}) define optional clauses.

```
CHG_DISP_SPEED  DISPENSE PATH --pw path--  {PERCENT CHANGE
--variable-- } { OFFSET SPEED --variable--  SAVE DB
--yes/no--}
```

The statement performs the following steps:

1. Define the path records that the velocity is going to change (PROCESS PATH --pw path--).

2. Changes the current velocity by a percentage. This percentage can increase or decrease the speed. The percentage is based on the speed specified in the dispense records. If a the statement is executed again with a different percentage value, the speed will be based on the value defined in the record, not the previous speed ({PERCENT CHANGE --variable--}).

3. Changes the speed defined in the dispense record by a user-specified offset value. This value can be saved to the database after the changes or left unsaved. Note that if the database is not saved every cycle, the path will be reprocessed ({OFFSET SPEED --variable-- SAVE DB --yes / no-- }).

CHG_DISP_DWELL

The statement allows changes in process path dwell times from a sequence statement. This is intended for applications that use automated methods to determine the viscosity of the material being dispensed. The dwell times can change based on a percentage of the current time, which is a temporary adjustment, or an offset change, which can be saved to the database if selected. The syntax of the statement is as follows. The braces ({...}) define optional clauses.

```
CHG_DISP_DWELL DISPENSE PATH --pw path--  {PERCENT CHANGE
--variable-- } { OFFSET DWELL--variable--  SAVE DB
--yes/no--}
{DOT ONLY --yes/no--}
```

The statement performs the following steps:

1. Define the path records that the velocity is going to change (PROCESS PATH --pw path--).

2. ***PERCENT CHANGE:*** Changes the current dwell time by a percentage. This percentage can increase or decrease the speed. The percentage is based on the speed specified in the dispense records. If the statement is executed again with a different percentage value, the speed will be based on the value defined in the record, not the previous speed ({PERCENT CHANGE --variable--}).

3. Changes the dwell time defined in the dispense record by a user-specified offset value. This value can be saved to the database after the changes or left unsaved. Note that if the database is not saved every cycle, the path will be reprocessed ({OFFSET DWELL --variable-- SAVE DB --yes / no-- }).

SET_DM_PATH

This statement allows changes in process path positions from a sequence statement. This statement allows the user to make positional changes to specified path segments. The syntax of the statement is as follows. The braces ({...}) define optional clauses.

```
SET_DM_PATH --pw path-- SEGMENT --variable-- {X:
--variable--}
{Y: --variable--} {Z: --variable--} {y: --variable--} {p:
--variable--} {r: --variable--}
```

The statement performs the following steps:

1. Defines the dispense path record where the position changes are to occur. (--pw path--)

2. Specify the segment number of the path position to change. This variable defines the number of the path node for the changes.

3. Offsets to the path position segment are defined in the variables for X,Y,Z,y,p,r arguments.

Z_UP

This statement will cause the robot to move to a specified Z height without any other axis motion occurring. Speed, acceleration, and deceleration are specified by the user in this statement as well. The intent of this statement is for clearance on a startup of the process or during an error recovery state. The syntax of the statement is as follows. The braces ({...}) define optional clauses.

```
Z_UP   LOC_HEIGHT  --constant--    SPEED --constant--  {
ACCEL --constant--   DECEL --constant-- }
```

1. Define the Z height that is desired for the motion. This is in millimeters in world coordinates (LOC_HEIGHT --constant--).

2. Define the velocity of the motion in unit of percentage of speed (SPEED --constant--).

3. Define the optional acceleration and deceleration for the motion to the Z height ({ACCEL --constant-- DECEL --constant-- }).

MOVE_PATH_SEG

The statement moves to a specified process path position based on the specified segment number. It has the ability to move in either standard robot coordinates or camera coordinates. The syntax of the statement is as follows. The braces ({...}) define optional clauses.

```
MOVE_PATH_SEG {APPROACH --path--} SEGMENT # --variable--
USING PATH POINTS --pw path-- {DEPART VIA --path--}
{IF SIG ON --variable--} {CAMERA COORD --yes/no--}
```

The statement performs the following steps:

1. Specify an approach path if needed (APPROACH VIA --path--).

2. Specify the path segment point transformation for motion (SEGMENT # --variable).

3. Define the process path from which the path segment transformation will be determined (USING PATH POINTS --pw path--).

4. Specify an optional depart path (DEPART VIA --path--).

5. Execute this statement is the defined signal is on (IF SIG ON --variable--).

6. Move to either the position in robot coordinates or camera coordinates (CAMERA COORD --yes/no--).

Path Calculations

PathWare uses the data specified in the PROCESS PATH database records to calculate the path, acceleration, gun compensation, and path compensation for a process robot application. These calculations are discussed in the next three sections.

The Path Analyzer

This section describes PathWare's *path analyzer*. The DISPENSE statement uses the path analyzer to ensure precise robot path tracking, I/O timing, and speed control.

Figure 4-2 shows an example of a simple path. The locations that describe this path are stored in the Process Path database. Given this series of locations, the path analyzer calculates many intermediate locations along straight lines and arc segments. These closely spaced points provide precise motion control and dispensing gun control. (Note that multiple path segments can be linked together to define more complex paths. Also, the segments are not limited to a flat surface.)

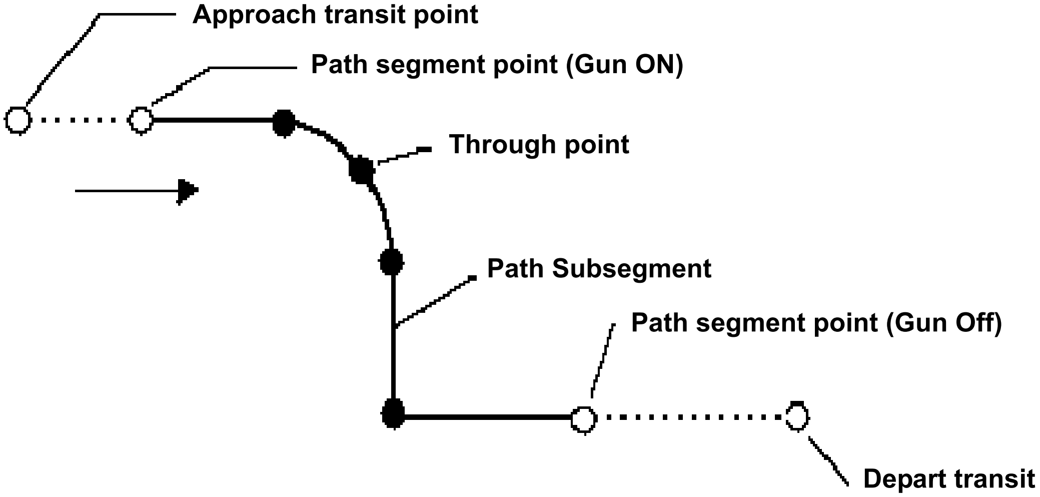


Figure 4-2. Simple Process Path

The spacing between the interpolated points is determined by the value specified in the *speed* field of the Process Path database and the path spacing value specified in the pathware initialization database. The default path spacing is 32 milliseconds apart in the initialization database. This value can be changed to a value no lower than the trajectory generator cycle time.   This value is normally 16 ms. However, it can be change to faster cycle times if the enhanced trajectory generator option is purchased.   Note that smaller spacing will increase the demand for processing of the system. It does enhance the resolution of the path that is performed. Based on the default values, the following calculation can be made. If two taught points are spaced 320 mm apart and the desired robot speed is 200 mm/s and the acceleration parameters are the same as the path spacing, the path analyzer calculates fifty points spaced 6.4 mm apart. The nominal spacing is 6.4 mm since 200 mm/s × .032 seconds = 6.4 mm. There are fifty points because 320 ÷ 6.4 = 50. If the taught locations are spaced 325 mm apart, fifty points would still be calculated (the integer part of 325 ÷ 6.4 is 50). However, since the segment cannot be divided evenly, the spacing is now 6.5 mm (325 mm ÷ 50 = 6.5 mm) and the time between points is now 32.5 ms (6.5 mm ÷ 200 mm/s = .0325 seconds).

Figure 4-3 shows the same path as **Figure 4-2**, except thatthe interpolated locations are displayed. The higher the robot speed, the larger the interval spacing. The DURATION instruction provides the precise speed control along the path

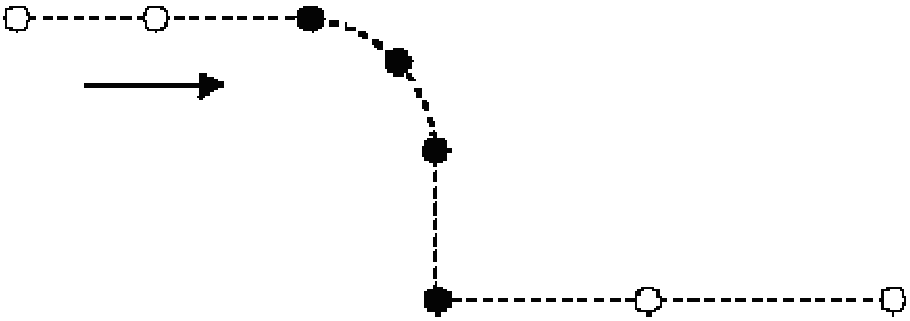


Figure 4-3. Interpolated Process Path

During normal sequence execution, the path analyzer calculates intermediate locations only the first time a DISPENSE statement is executed. The path analyzer reads in the location and motion parameter data from the Process Path database. Next, the path analyzer calculates the intermediate path locations. For subsequent loops through the DISPENSE statement, the path analyzer does not recalculate the process path unless data for the path has been modified. If dispensing parameters have been changed in the Process Path database or the PathWare Signal database, the path analyzer will calculate the process path each time a DISPENSE statement is executed within a sequence until the changes have been saved to disk. Additional details about sequence execution are discussed later in this chapter.

Teaching Arc Segments

Arc segments used with PathWare require three points to determine the radius and center of the arc. The third point will determine what the following segment type will be. A few examples are provided for better understanding of how to use PathWare for defining arc motions.

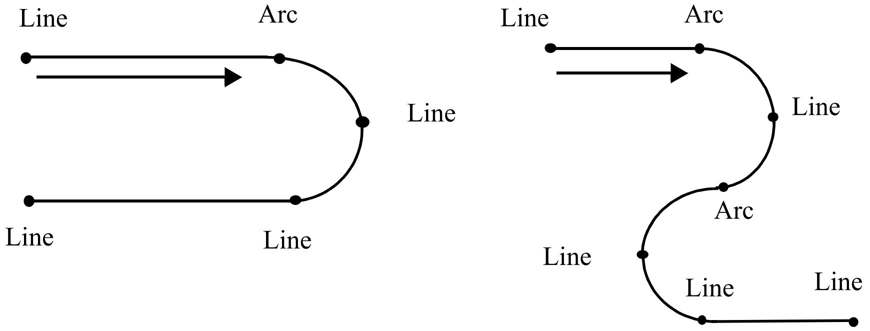


Figure 4-4. Arc Segment Examples

Figure 4-4 shows two examples of arcs being used in dispense paths. The first example shows a line intersecting an arc and finishing with a line segment. The arc needs to be defined only on the first node point where the arc begins. The following two nodes in this case are defined as line nodes. The second example on the right is shows a S-Curve example. This example shows the first node point of the beginning radius defined as an arc. The third point of the first radius is also the first point of the second radius. This point is also defined as an arc point.

When defining a circle, the first and last point of an arc cannot be defined as the same point. The proper way to define a circle is to teach five points where the first and third points would be defined as arcs. Gun changes of state cannot occur on the second point of the arc. Gun states can be changed only at the beginning or end of a segment. The intermediate arc point is in the middle of the segment.

Dot Process Points

A new feature in revision 3.3B allows points to be specified in the path as a dot. This feature allows the user to group all of the possible dispense processes into one group. Path editing and sequence development can be greatly simplified by putting all the path parameters in one group.

The Dot process implies that a motion will occur to that point and a gun process will occur based on the dwell time and gun compensation setup in the signal database. Approach and depart motions can be specified as part of the dot process. No gun operations can occur on the motion to the dot position.

Motions to the dot position and on to the next path position are Joint Interpolated style motions. Speeds, Accelerations, and nulling tolerances can be specified for these motions in the process path database. Note that all speeds are specified in millimeters per second.

Gun operations for the dot position use all of the standard signal database parameters. Gun compensation will be used with the gun operation on the dot. Note that dwell period is considered the time between the Zero On state and the Zero Off state. Positive On compensation values will extend the process time period. Negative timing values that are entered for the Off compensation will occur during the depart motion. When using a servo pump, the dot parameters specified are servo pump speed and volume, instead of the dwell period.

Additional dwell time periods for before the gun process and after the gun process can be specified in the initialization database. Timing for the actual dot process uses a 1 millisecond resolution during the gun operations. Please note that during the line and arc bead process, the gun compensation resolution is based on the path point spacing.

Dot points are shown graphically as a lime green square for each position.

Acceleration

A new feature added to AIM 3.1 version of PathWare is the ability to have accelerations and decelerations occur during a path motion. Previous revisions assumed infinite acceleration during the motion of the path. Adept robots can handle very high accelerations. However, many other devices can not perform without accelerations. Shaky motion during the acceleration areas of the path is a good indicator that acceleration parameters need to be used to reduce the acceleration of the device.

The acceleration and deceleration parameters in the process path records are in milliseconds. Larger values reduce the acceleration rates of the motion. Note that the acceleration values must be greater than twice the path point spacing in order for any acceleration or deceleration to occur during the motion. Linear acceleration and deceleration are accomplished within PathWare by changing the path point spacing during the acceleration and deceleration phase of the motion. This allows gun compensation timing values to still be functional during these motions. **Figure 4-5** shows the path point spacing starting at a small spacing and increasing until reaching full speed and the reverse on the deceleration side.

Acceleration and deceleration curves can intersect along straight segments of the path. However, the desired velocity will not be achieved if this occurs. Arc path segments do not allow the velocity profiles to intersect during the motion. The acceleration or deceleration portion of the motion cannot pass the midpoint segment. If this occurs, an error message will be displayed. The acceleration, deceleration or speed value will have to be changed to avoid this problem. Definition of acceleration and deceleration, is accomplished in the first node of the arc segment; accelerations and decelerations defined in the second node of the arc segment are not used in the calculation. When using the acceleration and deceleration values, the beginning and ending velocities are determined by the previous and the next path segment. If the first or last point is specified with acceleration the starting or ending velocity will be set to zero. Please note that accelerations and decelerations are defined for the entire segment in the first segment node.

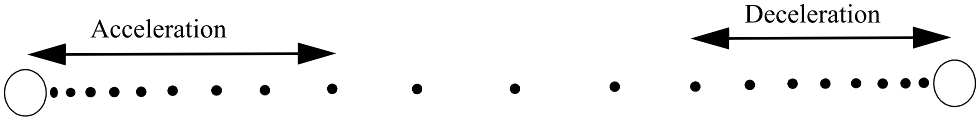


Figure 4-5. Acceleration and Deceleration Along Path

Gun Compensation and Output Control

To adjust for time delays caused by viscous dispensing materials or multiple device timing relative to a node point, the path analyzer uses the *gun compensation* values defined in PathWare Signal records. These values represent the delay between the time the gun is turned on and the material starts to flow, or between the time the gun is turned off and material stops flowing. For example, when using a servo pump and a flow valve, the user may wish to open the flow value before starting the servo pump.

If compensation values are specified, the path analyzer will flag interpolated locations before the taught gun trigger locations so that the dispensed bead starts or stops nearer to the taught locations. Note that the first point in the path cannot use gun compensation because points are required to be established before the gun ON state occurs. Therefore, an entry point before the first gun point must be established a distance and speed away from the ON node point to allow enough time for the gun compensation value selected or an error will occur.

Compensation allows the operator to adjust gun trigger locations without having to reteach path locations. Compensation values are entered in the PathWare Signal records. Multiple values are allowed for both On and Off compensation. The value entered in the record should be the amount of time before the node point at which you want the action to occur. Enter the amount of time, and then select a device below the time field. This is done by double-clicking on the device field and picking a device from the pick list. Normally, six selections are available (IO_Grp1, IO_Grp2, IO_Grp3, IO_Grp4, Nozzle, and Servo Pump). Actual compensation time will vary depending upon the path point spacing selected in the initialization database. By default the spacing is 0.032 seconds; values entered will be rounded as close as possible to the entered value.   Smaller point spacing will allow greater resolution for gun compensation. The path analyzer flags locations farther back along the interpolated process path for larger compensation values.   Gun compensation can occur only after a defined line or arc path point (not after a precision point). Gun compensation cannot occur at the midpoint on an arc segment definition.

The eight Output Signals in the signal database are now available in four different groups. This means the keywords (IO_Grp1, IO_Grp2, IO_Grp3, IO_Grp4) must be specified in the gun compensation area for different output groups to work.

New with revision 3.3, you can now enter negative timing values for process to occur after the node point. The software will check the forward timing value to assure that the process can occur during the path process (you will get a gun compensation error if this cannot occur). Gun compensation cannot occur during the motion following a Dot or Precision Point.

During the Dot process gun compensation is used with the dwell or volume that is specified at that point. Please note that the dwell period is considered the time between the zero ***ON*** compensation and the zero ***OFF*** compensation. Positive values entered in the ***ON*** compensation area will extend the overall process time for a DOT. Negative values that are entered for the ***OFF*** compensation value will occur during the depart motion of the dot process if specified.

Path Compensation

This section describes PathWare's *path compensation* feature. Path compensation allows the operator to modify the manipulator's tool path to the left or right of the direction of travel by an entered value. This feature works in only one plane at a time (X-Y, Y-Z, or Z-X). This feature is very useful for cutting applications where tool wear requires the path to be modified to accommodate for dimensional changes in the tool.

PathWare's path analyzer calculates a new path based on the original positions that are stored in the database. The original positions are not modified: They are still maintained in the database. The path analyzer also takes into consideration whether the positions are arc or linear path segments in the calculation. In **Figure 4-6** the original path is shown as well as the path that is modified with path compensation.

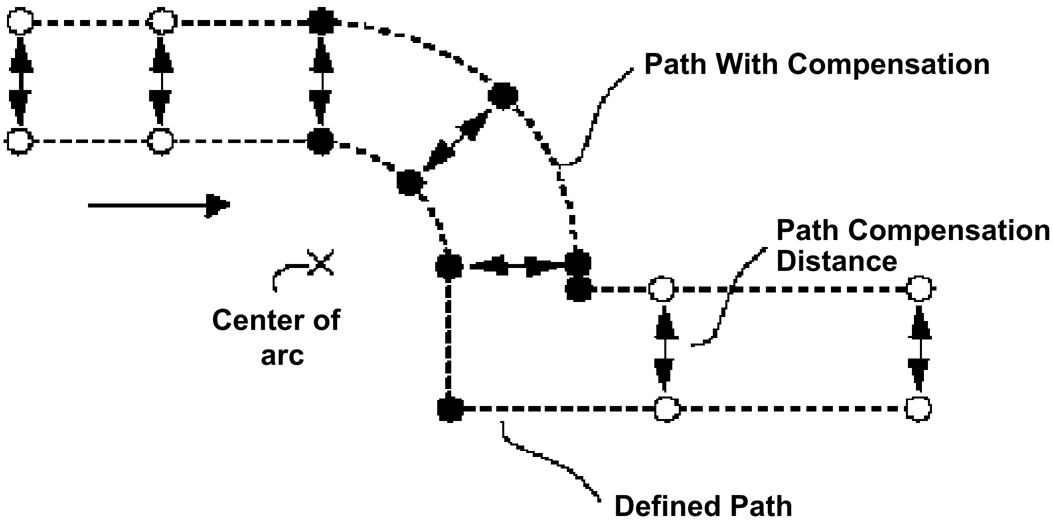


Figure 4-6. Left Side Path Compensation

Path Checking

The Path Checking feature performs a search through the path, inspecting the joint velocities for each motion. This process allows the user to set maximum joint velocities in the PathWare initialization database for this check. This process assures that speed settings for the motion are achievable and that the robot path is not running though a singularity point. This process requires a large amount of CPU time to perform a check. If the path is not using vision guidance or conveyor tracking, it is recommended to run the path check the first time the path is executed.

The path checking feature works differently if vision guidance or conveyor tracking is used. If vision guidance is used without conveyor tracking, the path check occurs after the frame of reference has been established. This check would occur during every cycle. If conveyor tracking is used, two additional selections are available to deal with path errors : 1) ***bypass errors***, 2) ***wait window***. The bypass error feature will let the current part pass on the conveyor if a path error should occur. The wait window option will continue to check the path with different positions on the belt until the part is successful or exits the window.

The path checking process allows a path point spacing parameter that will average the number of positions specified and run the check on a greater distance. This allows a faster check to occur, which can reduce the processing time. The parameter is defined in the PathWare initialization database under the parameter *path check point spacing*.

Conveyor Tracking With PathWare

Conveyor tracking can be used with PathWare for applications with material dispensing as well as cutting applications. Several features within the process path database exist to work with conveyor tracking. These features determine when to begin the path motion, the use of precision points with conveyor tracking, and the path checking features that function with conveyor tracking.

To enable the conveyor tracking option ☑ **conveyor tracking** needs to be selected in the main process path menu. If this is not selected, conveyor tracking does not function even if the frame of reference is defined with the conveyor record.

To use the path tracking features in PathWare several other databases must be defined correctly and linked with the Frame of Reference field in the process path main menu. The Frame of Reference and the Conveyor database are required to be defined and linked together for conveyor tracking to work properly. Please refer to the ***MotionWare User's Guide*** for more information on working with conveyor tracking.

Three options are available for dealing with the conveyor window when starting a path. The selections ☑ **Flag in Window**, ☑ **Segment in Window** and ☑ **Part in Window,** are available from the Process Path Main Menu under tracking. The flag in window option allows the user to select a node point WHICH the robot will wait until the flag has moved past the upstream limit. **Figure 4-7** shows a path with the marked node on a conveyor.

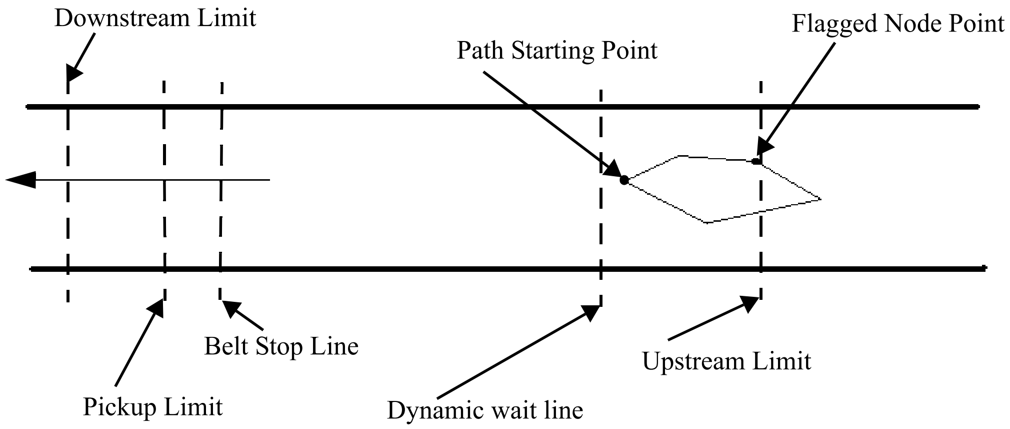


Figure 4-7. Conveyor Tracking, Flag in Window

The robot will wait at the dynamic wait line (if used in the conveyor database) until the starting point is past the Upstream Limit. The robot will continue to track the first point until the flagged node point has passed the upstream limit. The gun signals will not start until the path motion begins. This method will cause problems with window violations if the part can rotate on the conveyor belt.

The ***segment in window*** feature is ideal for applications that require maximum rate but are not concerned with gun activity. Cutting applications where pauses can occur at node points are a good example. The segment in window feature is shown in **Figure 4-8**.

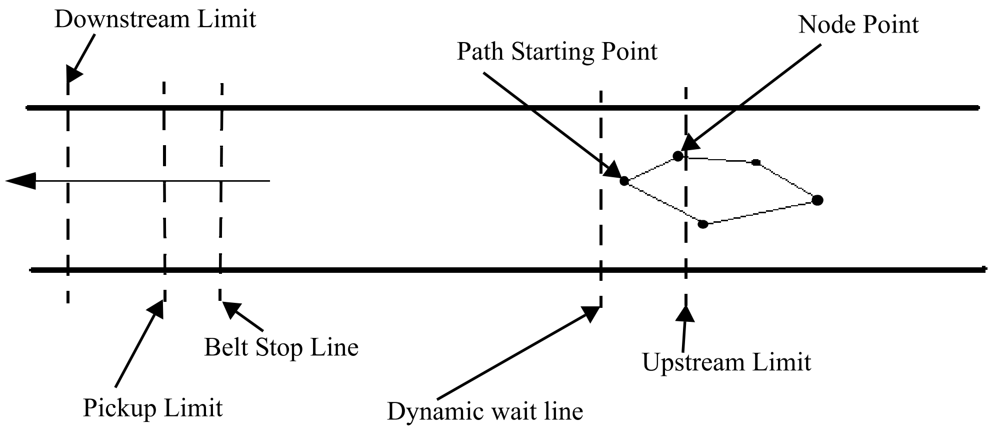


Figure 4-8. Conveyor Tracking, Segment in Window

With the ***segment in window*** feature the robot will track the starting point until the first path segment has passed the upstream window. The robot will move through that segment until it reaches the next node point. At that node point it will wait until the next segment has passed the upstream window. if it has passed the upstream limit, the robot will continue the path without a break. Every node point will be treated the same way. This assures that there will be no window violations.

The ***Part in Window*** selection assures the part has completely passed the upstream limit before path motion begins. This feature basically checks every node point along the path to make sure it has passed the upstream limit. The robot will basically track the starting point after it passes the upstream limit until all node points have passed the upstream limit. This selection is the safest selection when dealing with belt window violations. In addition, it will assure continuous path motion around the part.

The PathWare path checking option allows two features that work with conveyor tracking to handle path checking errors. These features are labeled ☑ **Bypass Failure** and ☑ **Wait Window**. The Bypass Failure selection will take the part that failed the path checking out of the conveyor queue for the dispensing path after the first path check occurred. The Wait Window feature will allow the path checking process to continue checking the part until the part has passed the pickup limit on the conveyor. The part as it travels along the belt will require different robot joint configurations, which may allow an originally failed operation to succeed later as it moves down the belt.

PathWare allows a precision point feature for conveyor tracking called ◉ **Floating**. This feature will allow the robot to move to a defined joint position that is calculated based on the moving frame of reference along the belt. This will allow the user to unwind or move to a proper joint configuration before beginning a path motion. This feature is simply selected and the arm configuration for the location is defined in that node point. The position will have to be taught the same as the rest of the path locations relative to the frame of reference. This position is recorded with the joint configuration selection bits as it is taught.

Teaching Paths Relative to Variable Frames

All conveyor applications in AIM require a frame of reference that is defined relative to the part. In most cases the path will have to be taught relative to this frame of reference using the position teaching utilities with PathWare. In some cases where the coordinates are know relative to the frame of reference, manual teaching may not be required. In order to teach the paths relative to the frame, the frame of reference must be defined before teaching the positions. If the frame is a taught frame not using vision or conveyor tracking, the frame should be taught and selected in the process path database. In this case, after the fame is defined, path teaching can occur.

The method to teaching paths relative to variable frames of reference is the same as in MotionWare. Greater detail is provided in the ***MotionWare User's Guide***. The Walk-Thru Training option available in the Task Status panel is used to allow AIM to define the frame of reference before teaching locations relative to that frame. The user will start the conveyor server and/or the vision server and the robot task before teaching any path locations. The conveyor, frame, vision, and the process path database must all be linked and records defined. Specifically, all the calibration and conveyor windows and vision tools should be defined and ready to go before beginning this process. The user will start the tasks and present the part to the system. The system will identify the part and try to move to a location relative to the part. Note that in the case of conveyor tracking, after the part has passed the Upstream Limit, the conveyor should be manually turned off. With the robot task running Walk-Thru Training, the system will identify that the first path position is undefined, and ask if you wish to edit the location.

There are basically 2 methods to teach positions relative to the variable frame. The first method requires that the user create all of the process path records needed before beginning the Walk-Thru Training exercise mentioned above. This requires the user to determine beforehand all of the required records needed for the path motion. When working with Walk-Thru Training, all of these locations for the defined records can be taught while the sequence is running. This allows the user to teach the locations wither with the teach pendant or by editing each of the segment records and using the Here button. Once all of the positions are taught, the proceed button can be pressed until the robot moves through the just taught path. Be sure not to move the part while teaching is occurring. Also the Process Gun button should be turned off to assure no material is flowing while this process occurs.

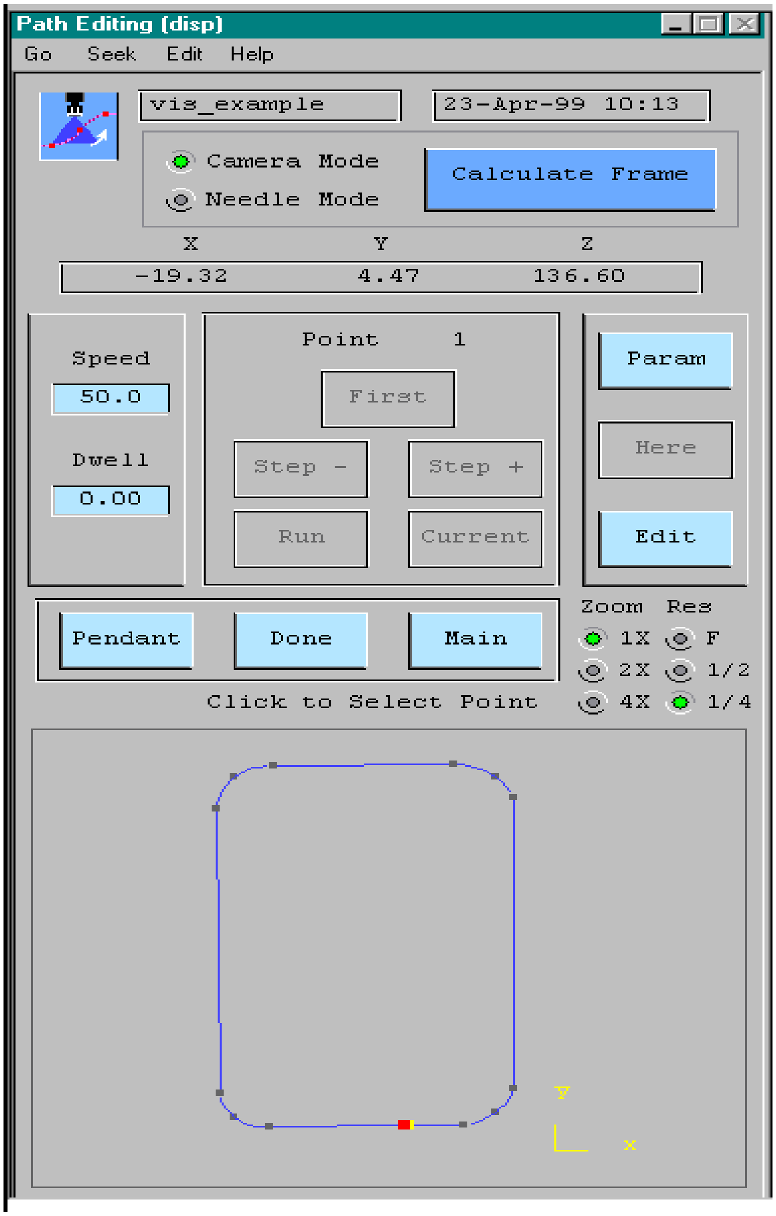
The second method requires that only two process path records be defined before the teaching process begins. All of the databases (conveyor, frame, vision, signal) that need to be linked should be completed before running the walk-thru-training process above. The user will run the process as in the first method until the request to define the first path position is issued. Be sure to turn off the conveyor after the part moves past the Upstream limit. At this point, the frame of reference is defined. The sequences can be aborted, and editing the process path can begin as normal. Note that the part cannot move. The frame of reference is written to the Frame record and is linked to the process path database. As long as the part does not move, editing and teaching of the database can proceed. The sequences must be aborted if additional process path segments are to be added. Be sure to save your databases after teaching positions or editing any records.

Vision Path Editing

Vision Path editing is a new feature provided in the PathWare package. This feature is set up in the Process Path database by selecting the SETUP button. (Please see **See "Path Editing Parameters" on page 48**.) The current implementation of vision editing in PathWare requires the user to use an arm-mounted camera. Other camera configurations are currently not supported. The user must establish a vision database with a picture record for use by PathWare. In addition, this camera must also be calibrated and linked properly to the vision database. If these requirements are not followed properly, vision editing will not function correctly. Inaccurate camera calibrations can cause path teaching to also be inaccurate.

Once the camera requirements have been established, the record names and database names must be entered in the SETUP menu in the Process Database. This menu contains entry fields for these record names. In addition, there are entry points for a forced frame of reference. Many vision guidance jobs use part fiducials or features to determine part position. PathWare allows the user to create paths relative to these frames of reference. The Path Editing menu provides a Reference Frame button to allow the user to have the robot automatically establish a vision frame of reference for use during Path Editing.

The Setup menu also has a check box to force the user to calculate a frame of reference before any path editing can occur. This is needed if reference frames are used in the application. If this check box is enabled, the Path Editing menu will appear as shown below.



vis_example

23-Apr-99 10:13

● Camera Mode
○ Needle Mode

X	Y	Z
-19.32	4.47	136.60

Speed 50.0

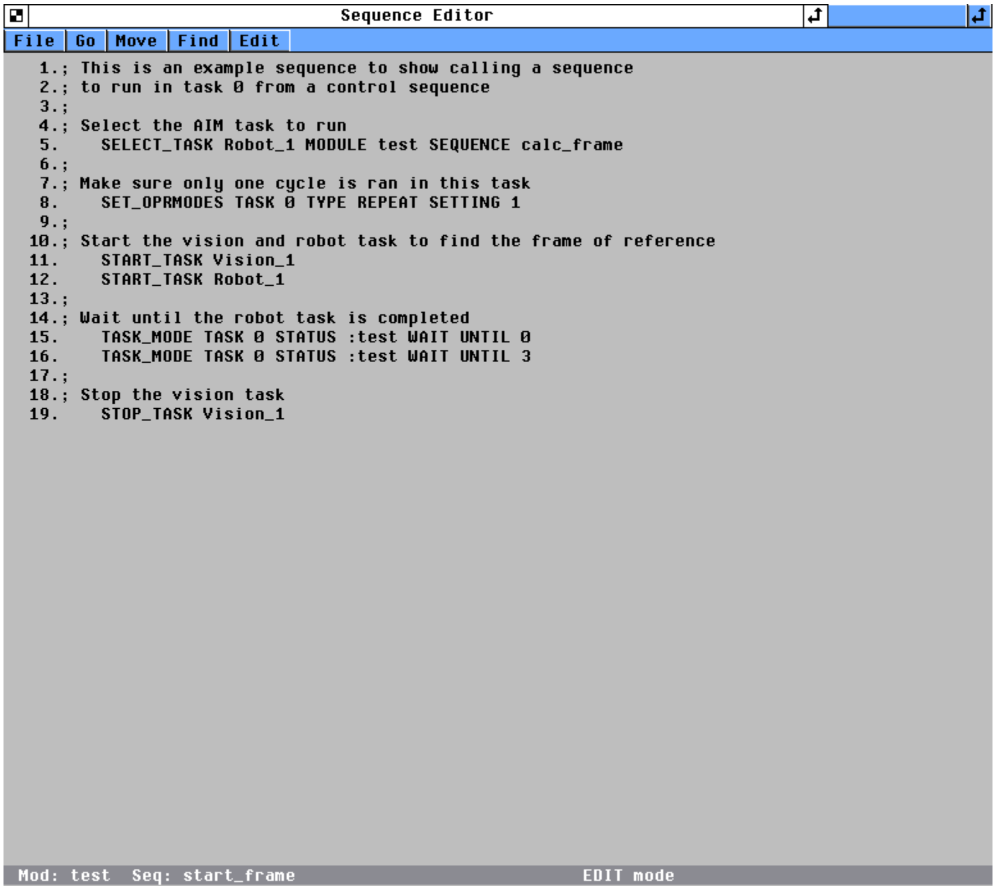
Dwell 0.00

Point 1

Zoom	Res
● 1X	○ F
○ 2X	○ 1/2
○ 4X	● 1/4

Figure 4-9. Vision Editing With Forced Reference Frame

Notice that the motion buttons and the HERE button are not selectable. These buttons will remain in this state until the Calculate Frame button is pressed. When this button is pressed, the sequence specified in the Process Path SETUP menu will be executed. This sequence must be a control sequence. In the standard application, this control sequence will start a robot sequence that will move the robot to calculate the frame of reference. Figure 4-10 provides a sample sequence.



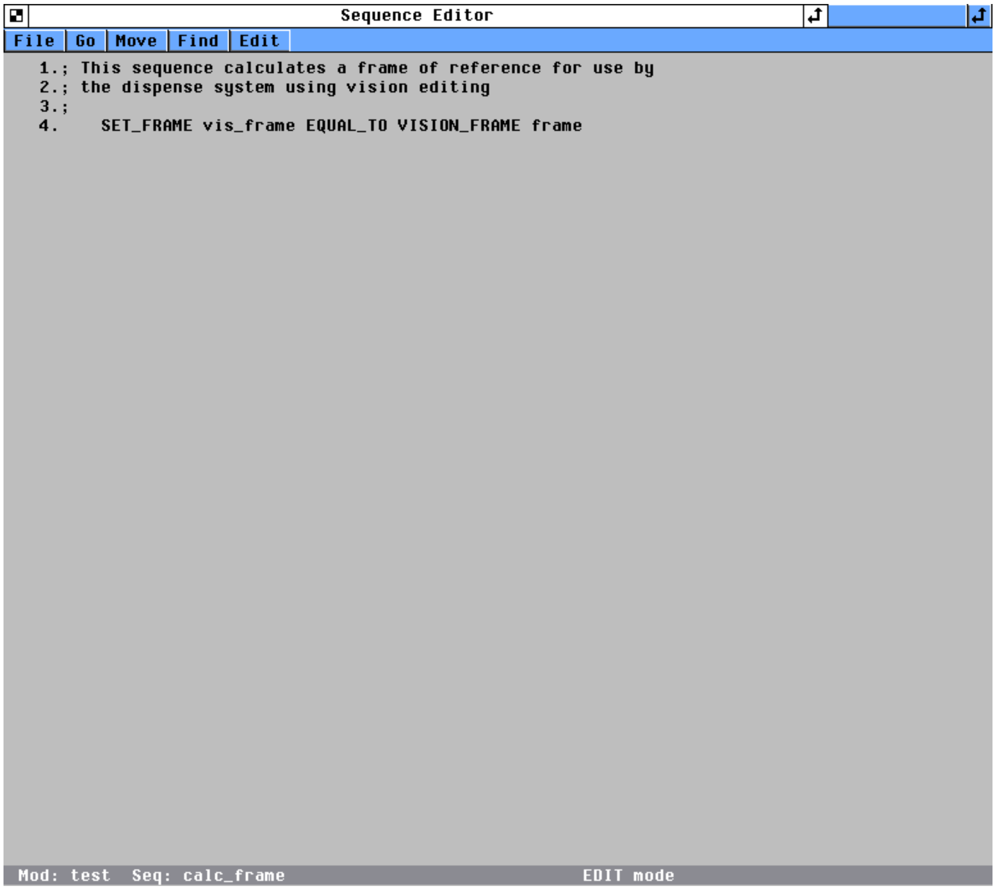
```
; This is an example sequence to show calling a sequence
; to run in task 0 from a control sequence
;
; Select the AIM task to run
   SELECT_TASK Robot_1 MODULE test SEQUENCE calc_frame
;
; Make sure only one cycle is ran in this task
   SET_OPRMODES TASK 0 TYPE REPEAT SETTING 1
;
; Start the vision and robot task to find the frame of reference
   START_TASK Vision_1
   START_TASK Robot_1
;
; Wait until the robot task is completed
   TASK_MODE TASK 0 STATUS :test WAIT UNTIL 0
   TASK_MODE TASK 0 STATUS :test WAIT UNTIL 3
;
; Stop the vision task
   STOP_TASK Vision_1
```

Mod: test Seq: start_frame EDIT mode

Figure 4-10. Vision Editing Example Control Sequence

This sequence would be executed when the Calculate Frame button is pressed on the vision editing menu page. This control sequence will execute a sequence that runs in the robot task to move the camera to find the part orientation. This sequence makes sure only one cycle is executed and also waits until the sequence in the robot control task has been completed. Note that this is an example sequence; more error detection may be required to make it product ready.

Figure 4-11 shows an example sequence that will find the frame of reference.

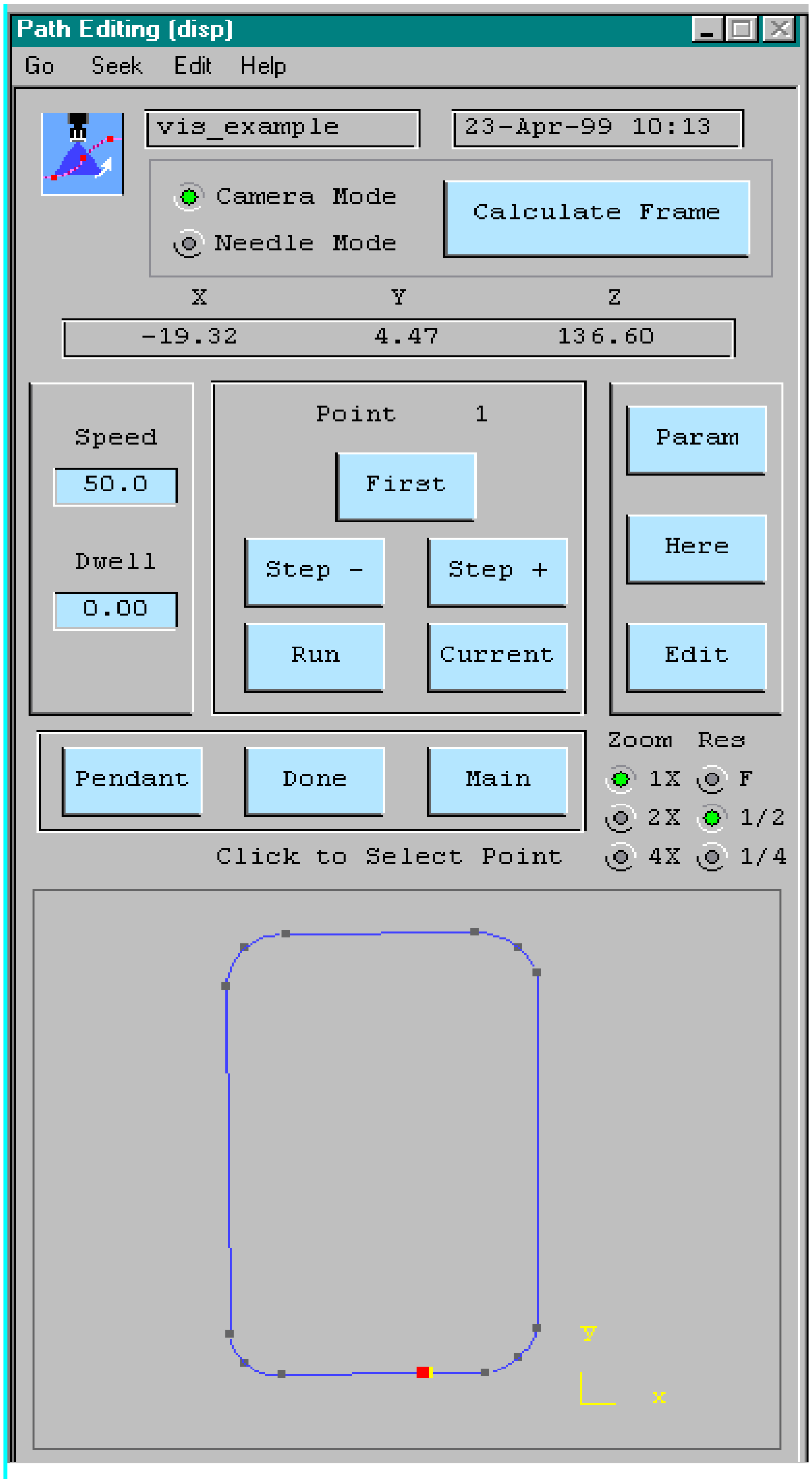


```
1.; This sequence calculates a frame of reference for use by
2.; the dispense system using vision editing
3.;
4.    SET_FRAME vis_frame EQUAL_TO VISION_FRAME frame
```

```
Mod: test  Seq: calc_frame                          EDIT mode
```

Figure 4-11. Vision Editing Frame Sequence

This sequence example uses the SET_FRAME instruction to allow the robot to move to a location and find a reference frame. There are other AIM statements that can be used to define reference frames as well. This sequence will be executed, and when it is completed, the control sequence will finish. The buttons on the vision editing menu will then become active. This menu is displayed in **Figure 4-12**.



vis_example

23-Apr-99 10:13

● Camera Mode

○ Needle Mode

X	Y	Z
-19.32	4.47	136.60

Speed 50.0

Dwell 0.00

Point 1

Zoom	Res
● 1X	○ F
○ 2X	● 1/2
○ 4X	○ 1/4

Click to Select Point

Figure 4-12. Vision Editing Menu With Full Functions

The menu shows all of the editing functions available to the user to modify the paths. Now that the reference frame has been established, pressing the following buttons will move the robot to the relative positions: Step +, Step –, First, Run and Current.

Note that there are two different coordinate modes (Camera & Needle). The camera mode will move to positions that will be shown in the vision monitor. The cross hair on the vision display indicates where the position is taught. The Needle mode will move the robot to the actual position that will be run during normal cycling.

The user can move the robot that will be viewed on the vision monitor. The virtual MCP software can be used to move the robot in a job or step mode. Once the robot is in the correct position, the HERE button can be pressed. The robot heights will not be changed when teaching positions while in Camera Mode. The Camera view of a part is supplied in **Figure 4-13**.

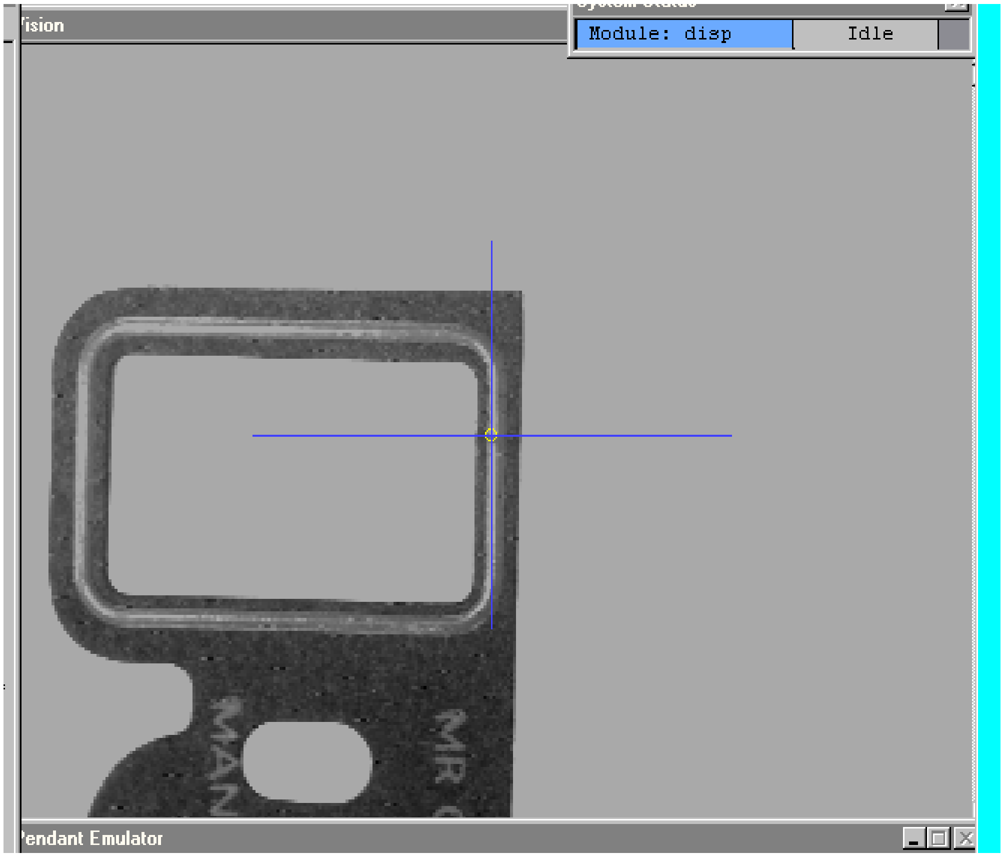


Figure 4-13. Camera Mode Display

Notice on the camera image a cross hair is display with a circle drawn on the center of the cross hair. The circle is provided to show where the actual needle would be during the motion. This circle can be changed in the PathWare initialization database. There are radio buttons supplied to allow the user the zoom the image and change the resolution of the display if needed.

PathWare Initialization Database

The PathWare initialization database allows many features of PathWare to be changed by entering different data. New with Revision 3.3 are many custom menus that group the initialization parameters into logical areas. This reduces the number of different selections in the search index, making it easier to find the appropriate selection. These window are shown after the table below. Many parameters will require the AIM software to be shut down and restarted after changing the parameter. You can edit the PathWare Initialization database by doing the following.

SETUP · **Initialization Database** · ***select from pulldown*** · **phwini.db**

Select the desired parameter for the PathWare changes.

Seek · **Index** · ***dbl clk on desired parameter***

The following table and windows display the parameter selections and a description of what the parameter does. It is a good idea to set up the desired parameters before teaching all of the path points in order to save time. Most all of the parameters specified in the Process Path database have default values that can be specified in the initialization database. Other parameters allow you to select features within PathWare as well as change the operation of the software.

Table 4-1. PathWare Initialization Database

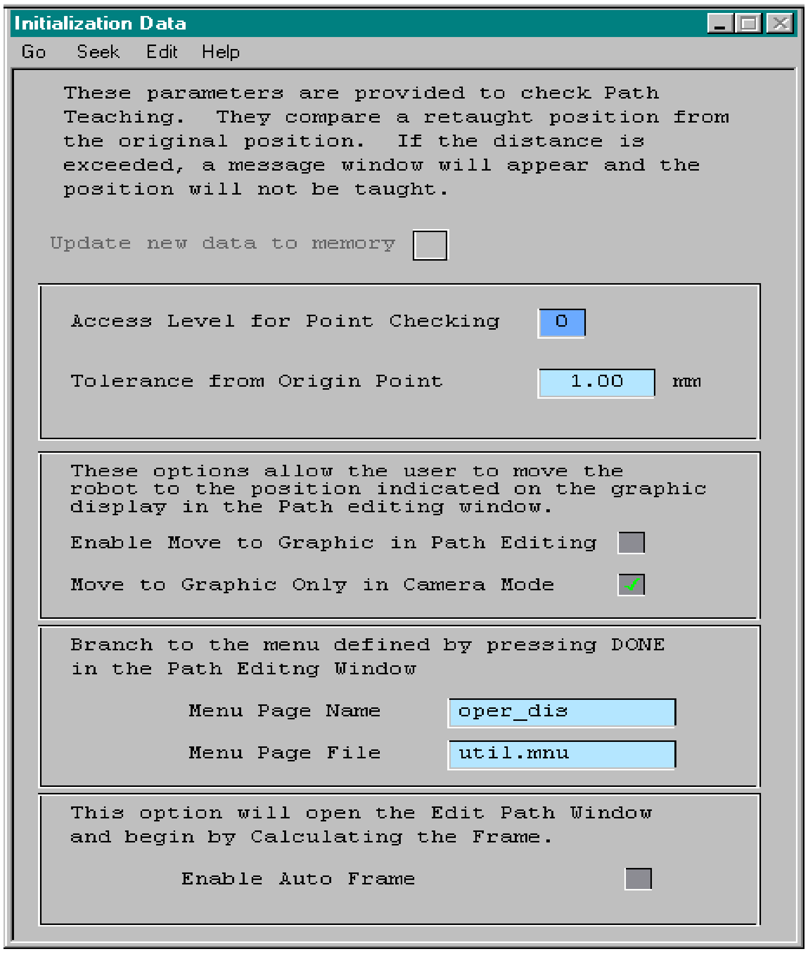
Parameter	Parameter Description
acceleration profile selection	Acceleration profile to be used during the motion along the dispense paths. If S curve profiles are used, the dispense path spacing must be greater than the overall S-Curve time segment.
defaults for operator	See Figure 4-14 for information on the operator parameters for the path editing control.
defaults for path editing	See Figure 4-15 on page 94 for information on default values for the path editing window.

Table 4-1. PathWare Initialization Database (Continued)

Parameter	**Parameter Description**
defaults for process path	See **Figure 4-16 on page 96** for information on default values for the process path database.
defaults for purge	See **Figure 4-17 on page 97** for information on default values for the purge process.
enable operator panel	This parameter enables the operator control menu to be displayed. If you wish not to use the operator control panel, disable this option.
enable servo pump option	This parameter enables support for the servo pump features within the dispense module software. Turning on this option will start an additional server that runs in a specified task to control the servo motor.
error database, additional	This parameter allows AIM to load and merge the dispense module error database with the AIM errors.
gun initial state	This parameter initializes the state of the dry run gun mode displayed on the status control panel to allow operation of a path without the gun I/O.
initialize Dot parameters	See **Figure 4-18 on page 98** for information on default values for the Dot points in the Process Path Database.
motion parameters to first point	See **Figure 4-19 on page 99** for information on the motion parameters to first point.
path checking defaults	See **Figure 4-20 on page 100** for information on the Path Checking default parameters.
path point spacing	This parameter defines the spacing, in seconds, between path points. The minimum setting is .016 seconds unless the user has the enhanced trajectory license. The smaller the number, the greater the resolution.
pump speed during dwell period	This parameter allows the user to specify a pump speed that will be used during a dwell time.

Table 4-1. PathWare Initialization Database (Continued)

Parameter	**Parameter Description**
statements, dispense	This item allows the dispense task statements to be loaded and merged with MotionWare's statements.
tracking break tolerance	This parameter specifies the distance in millimeters from the first path point before allowing the starting outputs to enable. This parameter is used only during conveyor tracking



These parameters are provided to check Path Teaching. They compare a retaught position from the original position. If the distance is exceeded, a message window will appear and the position will not be taught.

Update new data to memory □

Access Level for Point Checking 0

Tolerance from Origin Point 1.00 mm

These options allow the user to move the robot to the position indicated on the graphic display in the Path editing window.

Enable Move to Graphic in Path Editing □

Move to Graphic Only in Camera Mode ☑

Branch to the menu defined by pressing DONE in the Path Editng Window

Menu Page Name oper_dis

Menu Page File util.mnu

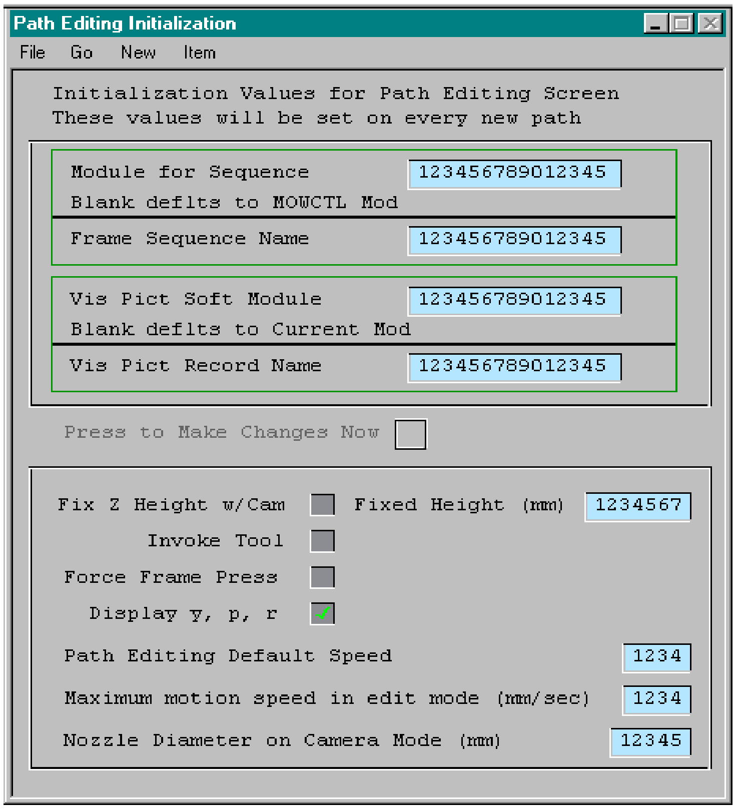
This option will open the Edit Path Window and begin by Calculating the Frame.

Enable Auto Frame □

Figure 4-14. Defaults for Operator

Path Editing Operator Defaults

- The ***Access Level for Point Checking*** allows the user to prevent users from making major changes to path points. This setting is the ***AIM ACCESS Level,*** where path checking occurs for Users with access values equal-to and below.
- The ***Tolerance from Original Point*** is the maximum point change in millimeters that is allowed when the point checking is running.
- The ***Enable Move from Graphic*** selections allow the robot to move to the point selected from the graphic display when the point is clicked on.
- The ***Move to Graphic Only in Camera Mode*** specifies that motion will occur only if camera mode is selected.
- The ***Menu Page*** and ***File Name*** entry fields specify the menu that the path editor will branch to if the ***BACK*** button is pressed.
- The ***Enable Auto Frame*** will allow the Frame to be calculated automatically when the menu page is appended.



Initialization Values for Path Editing Screen
These values will be set on every new path

Module for Sequence 123456789012345
Blank deflts to MOWCTL Mod

Frame Sequence Name 123456789012345

Vis Pict Soft Module 123456789012345
Blank deflts to Current Mod

Vis Pict Record Name 123456789012345

Press to Make Changes Now □

Fix Z Height w/Cam □ Fixed Height (mm) 1234567

Invoke Tool □

Force Frame Press □

Display y, p, r ☑

Path Editing Default Speed 1234

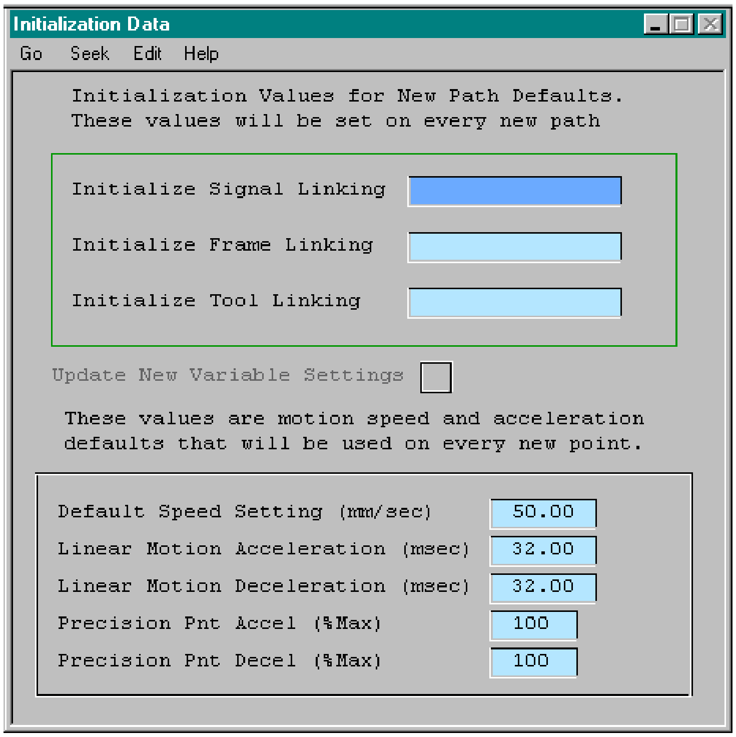
Maximum motion speed in edit mode (mm/sec) 1234

Nozzle Diameter on Camera Mode (mm) 12345

Figure 4-15. Defaults for Path Editing

Additional Path Editing Window Defaults

- The ***Module for Sequence*** entry field allows the user to specify the AIM Database Module that the frame calculation sequence resides in. This data will be set in every new path that is created.
- The ***Frame Sequence Name*** entry field specifies the sequence that is executed when the Calculate Frame button is pressed from the Path Editor menu.
- The ***Vis Pict Soft Mod*** entry field specifies the AIM database module that the vision database resides in. This data will be set in every new path that is created.
- The ***Vis Pict Record Name*** entry field specifies the name of the vision picture record to be used for Path Editing.
- The ***Fix Z Height w/Cam*** selection allows the user to set a default for the height that is used when teaching in Camera Mode.
- The ***Invoke Tool*** selection allows the user to set a default for the invoke tool setting that will automatically invoke the specified Tool Offset when entering the Path Editor Menu.
- The ***Force Frame Press*** selection allows the user to set a default for the Frame Press option. This option prevents teaching and path motion unless the frame has been calculated.
- The ***Display y, p, r*** selection sets the default for the display for the Path Editing menu. This parameter selects whether the yaw, pitch, and roll values of the transformation are displayed in the path editing window.
- The ***Fixed Height (mm)*** entry field specifies the default value for the height in camera mode.
- The ***Path Editing Default Speed*** entry field specifies the default speed for the motion of the robot when moving with the Path Editor menu.
- The ***Maximum Motion Speed*** entry field specifies the maximum speed at which the robot can move while running in the Path Editing menu.
- The ***Nozzle Diameter in Camera Mode*** entry field specifies the default nozzle diameter.



Initialization Values for New Path Defaults.
These values will be set on every new path

Initialize Signal Linking

Initialize Frame Linking

Initialize Tool Linking

Update New Variable Settings □

These values are motion speed and acceleration
defaults that will be used on every new point.

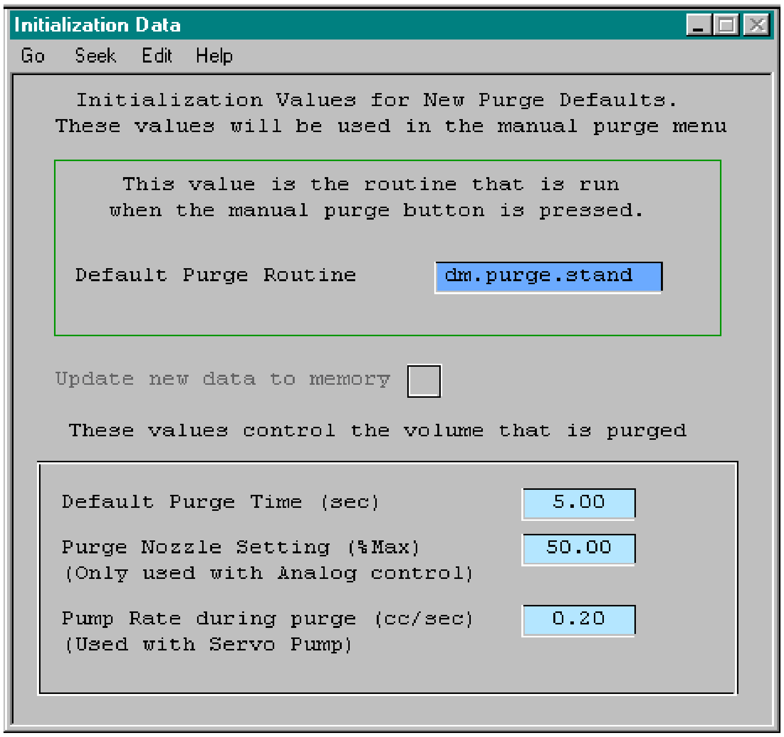
Default Speed Setting (mm/sec)	50.00
Linear Motion Acceleration (msec)	32.00
Linear Motion Deceleration (msec)	32.00
Precision Pnt Accel (%Max)	100
Precision Pnt Decel (%Max)	100

Figure 4-16. Defaults for Process Path

Process Path Database Defaults

- The ***Initialize Signal Linking*** entry field allows the user to specify the signal record that is to be defaulted to when creating a new path.
- The ***Initialize Frame Linking*** entry field allows the user to specify the frame record that is to be defaulted to when creating a new path.
- The ***Initialize Tool Linking*** entry field allows the user to specify the Tool record that is to be defaulted to when creating a new path.
- The ***Default Speed Setting (mm/sec)*** entry field allows the user to specify the default speed when a new path record is created.
- The ***Linear Motion Acceleration (msec)*** entry field allows the user to specify the default acceleration time.
- The ***Linear Motion Deceleration (msec)*** entry field allows the user to specify the default deceleration time.
- The ***Precision Pnt Accel (%max)*** entry field allows the user to specify the default acceleration percentage. This is also used for the Dot style points.

- The ***Precision Pnt Decel (%max)*** entry field allows the user to specify the default deceleration percentage. The is also used for the Dot style points.



Initialization Values for New Purge Defaults.
These values will be used in the manual purge menu

This value is the routine that is run
when the manual purge button is pressed.

Default Purge Routine dm.purge.stand

Update new data to memory □

These values control the volume that is purged

Default Purge Time (sec) 5.00

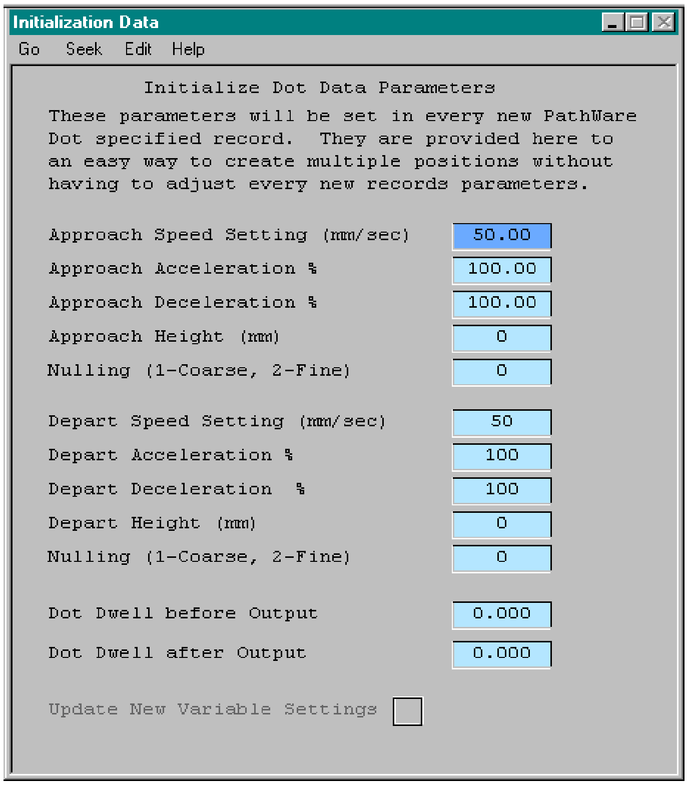
Purge Nozzle Setting (%Max) 50.00
(Only used with Analog control)

Pump Rate during purge (cc/sec) 0.20
(Used with Servo Pump)

Figure 4-17. Defaults for Purge

Purge Default Parameters

- The ***Default Purge Routine*** specifies the routine that is called when the purge button is pressed from this PathWare IO Control menu. This routine is provided in the file RUN_DM.V2 and can be copied and modified for custom applications.
- The ***Default Purge Time (sec)*** entry field allows the user to specify the length of time the purge will last every time the button is pressed.
- The ***Purge Nozzle Setting (%max)*** entry field allows the user to specify the analog setting that will be applied during a purge cycle. This is valid only if the analog feature is selected in the signal database.
- The ***Pump Rate during purge (cc/sec)*** is the pump speed that will occur during a purge cycle if the servo feature is selected.



Initialize Dot Data Parameters

These parameters will be set in every new PathWare Dot specified record. They are provided here to an easy way to create multiple positions without having to adjust every new records parameters.

Approach Speed Setting (mm/sec)	50.00
Approach Acceleration %	100.00
Approach Deceleration %	100.00
Approach Height (mm)	0
Nulling (1-Coarse, 2-Fine)	0
Depart Speed Setting (mm/sec)	50
Depart Acceleration %	100
Depart Deceleration %	100
Depart Height (mm)	0
Nulling (1-Coarse, 2-Fine)	0
Dot Dwell before Output	0.000
Dot Dwell after Output	0.000

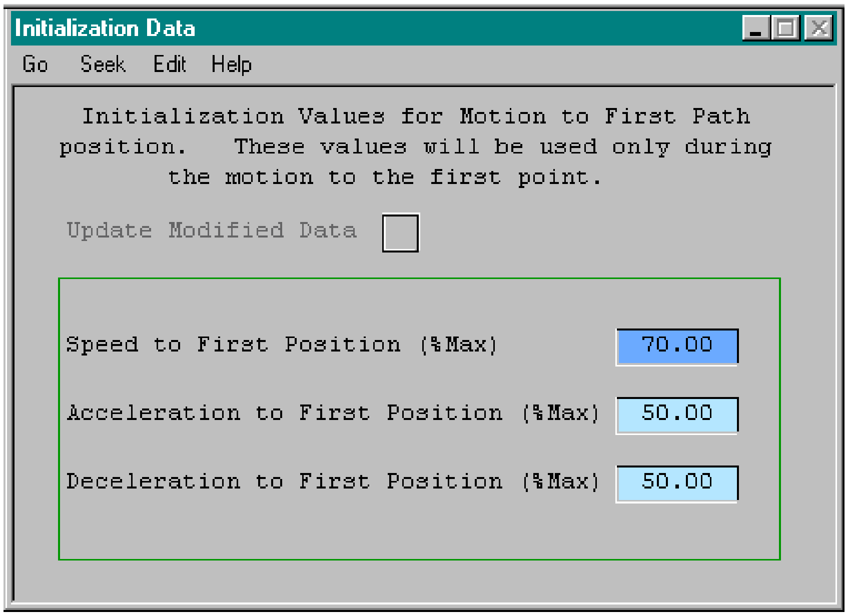
Update New Variable Settings □

Figure 4-18. Initialization for Dot Parameters

Dot Parameter Defaults

- The ***Approach Speed Setting (mm/sec)*** is the default value that will be set during every new path record creation.
- The ***Approach Acceleration %*** is the default value that will be set during every new path record creation.
- The ***Approach Deceleration %*** is the default value that will be set during every new path record creation.
- The ***Approach Height (mm)*** is the default value that will be set during every new path record creation.
- The ***Nulling Tolerance*** is the value for Fine and Coarse that will be set during every new path record creation. If nulling is selected, it will force a break at the approach point.
- The ***Depart Speed Setting (mm/sec)*** is the default value that will be set during every new path record creation.

- The ***Depart Acceleration %*** is the default value that will be set during every new path record creation.
- The ***Depart Deceleration %*** is the default value that will be set during every new path record creation.
- The ***Nulling Tolerance*** is the value for Fine and Coarse that will be set during every new path record creation. If nulling is selected, it will force a break at the depart point.
- The ***Dot Dwell before Output*** is the value used for a timing dwell before the gun process begins during a Dot process.
- The ***Dot Dwell after Output*** is the value used for a timing dwell after the gun process ends during a Dot process.



Initialization Values for Motion to First Path position. These values will be used only during the motion to the first point.

Update Modified Data □

Speed to First Position (%Max) 70.00

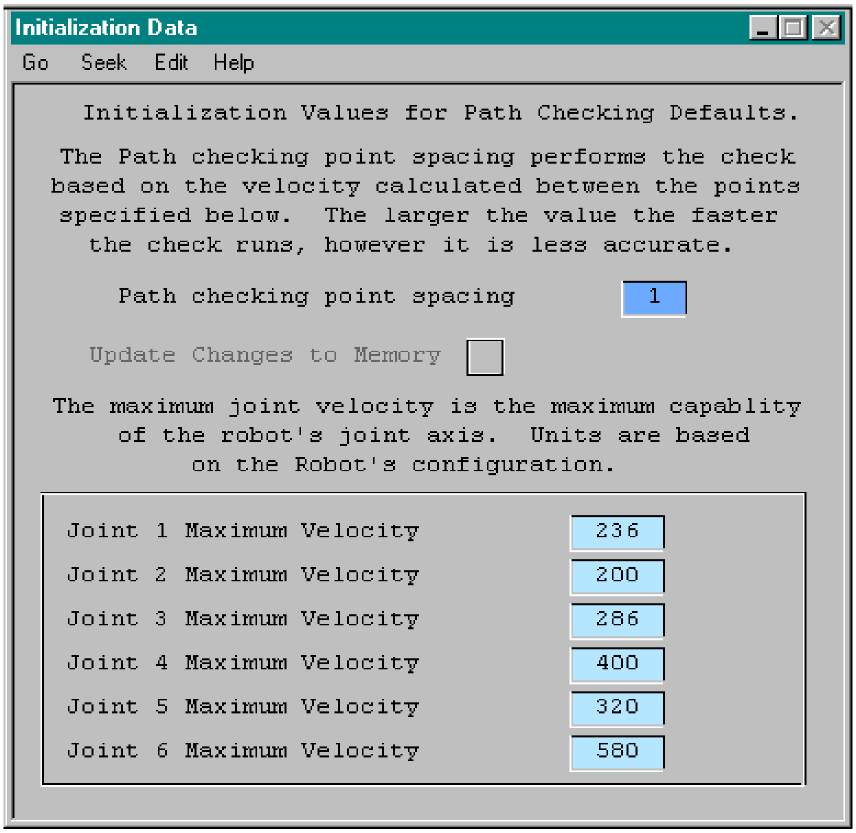
Acceleration to First Position (%Max) 50.00

Deceleration to First Position (%Max) 50.00

Figure 4-19. Motion Parameters to First Point

Motion Parameters to First Point

- The ***Speed to First Position (%Max)*** specifies the speed for the move from the current position to the first path point.
- The ***Acceleration to First Position (%Max)*** specifies the acceleration for the move from the current position to the first path point.
- The ***Deceleration to First Position (%Max)*** specifies the deceleration for the move from the current position to the first path point.



Initialization Values for Path Checking Defaults.

The Path checking point spacing performs the check based on the velocity calculated between the points specified below. The larger the value the faster the check runs, however it is less accurate.

Path checking point spacing 1

Update Changes to Memory □

The maximum joint velocity is the maximum capablity of the robot's joint axis. Units are based on the Robot's configuration.

Joint 1 Maximum Velocity	236
Joint 2 Maximum Velocity	200
Joint 3 Maximum Velocity	286
Joint 4 Maximum Velocity	400
Joint 5 Maximum Velocity	320
Joint 6 Maximum Velocity	580

Figure 4-20. Path Checking Initialization

Path Checking Initialization Parameters

- The ***Path checking point spacing*** specifies the resolution of the number of points to calculate and check the joint velocity to indicate singularites or to high of velocity settings. The best check occurs with a value of 1, it however takes the most time to process which hurts cycle time.
- ***Joint 1 - 6 Maximum Velocity*** specifies the maximum joint velocity that is allowed during the path check.

Variable Device Control Features

PathWare allows control of devices relative to the tool tip velocity of the robot. This control is accomplished by either analog voltage or servo motor velocity. The PathWare signal record allows selection of both types of control. Scaling factors are provided for individual segments using the bead percentage, which is relative to the maximum volume at specified tool tip speed. The bead percentage value is located in the segment menu with the process path database. The maximum volume at the specified speed is located in the pathware signal database. If custom features are required for your application, refer to **Chapter 5**.

Analog Velocity Control

The analog control features within PathWare are developed to work in conjunction with Xycom's XVME-540 VME analog board interface. The software written uses the analog instructions within V^+. PathWare assumes the use of the first analog board and allows the user to specify which output channels to use. If a second board is going to be used or a third-party analog board is desired, the pathware software will have to be modified.

The analog output voltage uses the following formula to determine the voltage signal provided to the analog device.

Analog.Output = (bead.perc) * (vel / max.spd)

where:

bead.perc = Bead percentage specified in the segment menu
vel = Calculated tool tip velocity
max.spd = 100% Flow at Speed setting in the Dispense Signal Database

Provided in the pathware signal database is a feature called **Look Ahead Time** which allows the user to specify a period of time ahead of the current location for analog adjustments. A calculated velocity is used based on the preprocessed path. This velocity should be accurate as long as the acceleration specified in the segment can be accomplished by the device. The "look ahead" uses the AIM variable database or a value typed in by the user. This value should be in seconds.

The analog output is adjusted based on the path point spacing, which can be changed in the initialization database. The default spacing is 0.032 seconds. Better analog control can be accomplished with tighter path point spacing.

Servo Velocity Control

The control of a servo motor relative to the tool tip velocity is a feature that is provided in PathWare. This feature requires that an Adept servo board be used to control the servo axis. Adept provides servo boards called the MI3 and MI6 for three- and six-axis control of servo motors. The servo motor that will be controlled by PathWare must be configured with the Joints Kinematic Device Module (JTS) when it is configured in software. Also, the rollover feature must be used and set to a distance smaller than the software stops specified in *SPEC* in the case of a unidirectional motor. If the servo being used is a reciprocating style pump, then the rollover setting should be made higher than the softstop specified. The specifications for *robot joint speed* must allow maximum performance of the motor to allow proper pump control. Generally, the direction of the motor should be positive to the direction for pump flow. The negative direction is used for the unwind of the axis when the application is using the reciprocating servo. Refer to the ***Adept MV Controller User's Guide*** and the ***AdeptMotion VME User's Guide*** for information on installation and selection of the servo equipment.

The PathWare software package assumes that joint one of the Joint Module is used for the servo pump. This should be the case in most applications. If multiple servo motors are to be used with the system, individual Joint Module device configurations are recommended. If special configurations are required, refer to the **Chapter 5**.

PathWare requires that the PathWare initialization database be modified to allow the servo pump to function. The initialization parameter *enable servo pump option task selection* must be set correctly for proper operation. Additionally, the ***Task Configurator*** provided with AIM Utilities must be set to allow the servo task to be installed. Follow the list below to configure the servo motor properly.

1. Select the initialization databases from the pulldown menu under **SETUP**.

2. Select the database called **phwini.db** from the initialization database pulldown.

3. Select the *enable servo pump option* from the scroll pick list of parameters. Turn on this feature. Please note that AIM must be restarted for the software to reconfigure itself for this feature.

4. Select the *Task Configurator* menu from the pulldown menu under **SETUP**.

5. Double-click on the Start Servo Pump 1 selection from the menu.

6. Make sure the following settings are correct for your application:

a. The task selection defaults to task 9 for the first servo pump. Make sure no other software is running in this task. The task numbers can be changed based on the configuration of your system. Please note that it is recommended to try to keep this task the lowest number possible to assure proper performance of the system.

b. The task name and device name are defaulted to Pump_1. The user can change these names as needed.

c. Make sure that the proper device number is selected. In the case of multiple robots and other servo devices running on the system, you need to verify that the proper robot number is used. This value defaults to robot 2.

d. Please keep the other default settings to the predefined values.

7. Make sure the ***Start Task*** selection is checked so that the task will start.

8. Shut down and restart the AIM system.

When AIM starts up, a servo motor server task will be configured to run with the system. This task will provide the motor control for both the DISPENSE and APPLY_DOT statements. PathWare software uses the V^+ SPIN and DRIVE instructions for operation of the motor. During path motion the V^+ SPIN instruction is used for controlling the motor speed. Every calculated path point, the SPIN instruction is updated to modify the servo pump speed if needed. After completion of the path the DRIVE instruction is used for the suck back operation. This operation is specified by a volume amount. The APPLY_DOT statement specifies a volume amount of material to be applied. The server executes a DRIVE instruction for applying the volume specified and then another DRIVE instruction for the suck back feature if specified.

Several items must be specified in the PathWare Signal database for proper pump operation. These fields must be set in the database: *100% Flow at Speed*, *Device*, *Flow Rate (CC/REV)*, *Max Flow (CC/SEC)*, *Suck Back Rate (CC/SEC)*, *Suck Back (CC)*. The *100% Flow at Speed* value is a scaling factor used in determining pump speed based on changes in tool tip velocity. It is used as a scaling factor for changing pump flow rates based on speed. The *Device* field is a keyword pick list that should display the pump name you specified in the initialization database (normally PUMP_1). The *Flow Rate (CC/REV)* field is a calculated value that should be available from the pump manufacturer. The *Max Flow Rate (CC/REV)* is used to make sure the pump does not overrun. This value should also be provided by the pump manufacturer. Please be sure that the pump motor is safe over this range. This value may have to be reduced if the motor cannot rotate at the specified maximum pump rate. The *Suck Back Rate (CC/SEC)* is the pump speed at which you want the suck back to occur. The *Suck Back (CC)* is the volume that should be sucked back during a suck back cycle.

The pump flow rate is calculated based on the flowing formula located in the *dm.gun.stand* routine. It is provided here for better understanding of the values above.

pmp.spd = ((speed.mult *vel* (bead.perc / 100))/ (spd@100))* (max.spd)

where:

speed.mult = Speed specified in the task status panel -- should be 100
bead.perc = Bead percentage specified in the segment menu
vel = Calculated tool tip velocity
spd@100 = 100% Flow at Speed setting in the Dispense Signal Database
max.spd = Maximum RPM of servo pump

The ***Gun Compensation*** area in the PathWare Signal database must also be selected for the Servo Pump to be turned on or off. This area is provided for timing control of the valve and the servo pump relative to the segment node point where the gun is turned on or off. One of the fields under the *Device On-Time Before Node* must be selected to the Pump Name and the timing where you wish the pump to start relative to the node. The same is true for the *Device Off-Time Before Node* area, the pump name and timing relative to the node must be selected.

Provided in the pathware signal database is a feature called ***Look Ahead Time*** which allows the user to look a period of time in the future to make pump adjustments. The velocity for the top tip is based on the preprocessed path.This velocity should be accurate as long as the acceleration specified in the segment can be accomplished by the device. The *look ahead* uses the AIM variable database or a value typed in by the user. This value should be in seconds.

Bead size during the path motion is controlled by the *Pump Rate (CC/Vel)/ Bead %* field located in each path segment node point record. With the servo pump control this value is in CC/ (mm/sec) which in many cases is very small. This is different from analog control with a percentage of max flow.

The servo motor velocity is adjusted based on the path point spacing, which can be changed in the initialization database. The default spacing is 0.032 seconds. More precise control can be accomplished with tighter path point spacing.

The additional Reciprocating Servo parameters are discussed in **"PathWare Signals" on page 51**.

Using Device Net Analog Outputs with PathWare

Pathware supports the use of two (2) 12 bit 0 – 10 volt DC analog outputs to control proportioning dispense valves. In its standard configuration Pathware uses only one (1) of these outputs. The other output, if supplied, is available for connection to a second proportioning dispense valve if desired. However, the user must customize the routine ‘dm.gun.standard’ to access this output. Using analog outputs requires the following user supplied DeviceNet hardware from Beckhoff (www.beckhoff.com):

-BK5200 DeviceNet Bus Coupler
-KL4001 Single Channel or KL4002 Dual Channel (if 2 channels are being used)
  Analog Output Terminal
-KL9010 Bus End Terminal

Note that the above hardware list assumes an independent Beckhoff DeviceNet node. If your system already contains a Beckhoff BK5200 DeviceNet Bus Coupler you need only add the KL4001 or KL4002 Analog Output terminal to that node.

In order to access the DeviceNet analog I/O from the Adept controller you must first configure both the Adept controller and the BK5200 Bus Coupler. The following procedure explains this process. The configuration process assumes that you have no other DeviceNet hardware on the DeviceNet bus and that the Beckhoff block contains only the KL4001 or KL4002 Analog Output terminal or this terminal is the first ‘byte’ oriented terminam to the right of the bus coupler. If you have additional DeviceNet hardware present or you are adding the Beckhoff analog output terminal to an existing node the process remains generally the same. However, the configuration parameters may differ from those in the following example.

Configuring the Beckhoff analog I/O for use by the Adept controller

To configure the Beckhoff DeviceNet analog I/O for use by the Adept controller follow the below listed procedure.

1. Assemble, install and wire the Beckhoff BK5200 Bus Terminal, KL4001 Analog Output Terminal and KL9010 Bus End Terminal in accordance with the manufacturer’s instructions. Set the eight (8) DIP switches on the BK5200 Bus Coupler as follows:

   1, 8 = on
   2, 3, 4, 5, 6, 7 = off

   This corresponds to a node MACID of ‘1’ and a network speed of ‘500K’ baud.

   (see also the ‘Adept MV Controller User’s Guide or the Adept SmartController User’s Guide for information on connecting to Adept DeviceNet)

2. Power up the DeviceNet node and then the Adept controller
3. Load and execute the Adept System Configuration Utility ‘config_c.v2’. (see Instructions for Adept Utility Programs)
4. Enter the ‘V+ System Configuration Data’ menu and then the ‘Edit system Configuration’ menu
5. Select ‘Change DeviceNet configuration’ from the menu selection
6. If you have no DeviceNet nodes already configured the message

```
The DEVICENET section contains no statements

Add a new statement (Y/N)?
```

(Note that if you already have other DeviceNet hardware on the bus you will already have a ‘LOCAL’ statement and one or more ‘MACID’ and ‘MAPPING’ statements. In this case copy down the ‘MACID’ and ‘BAUD’ parameters from the ‘LOCAL’ statement and exit this menu and continue with step 10. Be sure your Beckhoff BK5200 Bus Terminal baud rate matches the baud rate set in the LOCAL statement)

7. Create the following DeviceNet statement:

```
LOCAL = “/MACID 0 /BAUD 500K”
```

8. Exit this menu and select ‘Done Editing’. Answer ‘y’ when prompted to ‘save modified configuration data to disk (Y/N)?
9. Exit the Adept System Configuration Utility and reboot the controller
10. Load and execute the Adept System Configuration Utility ‘config_c.v2’.
11. Select ‘Scan the DeviceNet’ and then ‘Scan the Devicenet for nodes’
12. Enter the local MACID value as ‘0’ and the baud rate as ‘3’ (500K). The scanning process will gather information on all the DeviceNet nodes connected to the Adept network. This will take several minutes.

At the conclusion of the scanning process you will see the following, or similar display if the node was properly recognized:

```
MAC ID 1:    Input size       = 1
             Output size      = 2   (4 if using the KL4002 terminal)
             Vendor ID        = 108
             Device type      = 12
             Product code     = 5200
             Product name     = BECKHOFF BK5200 V03.05
             Major revision   = 3
             Minor revision   = 0
             Serial numbers   = A 79 (hex)
             Status           = 0
```

Copy down this information

13. Return to the main menu and enter the 'V+ System Configuration Data' menu and then the 'Edit system Configuration' menu
14. Select 'Change DeviceNet configuration' from the menu selection. The existing statement(s) will be displayed.
15. Answer 'n' to 'Edit this statement' and 'y' to 'Add a new statement'
16. Create the following DeviceNet statement:

```
MACID 1 = "/INPUT 1 /OUTPUT 2 /VENDOR_ID 108 /DEVICE_TYPE 12
          /PRODUCT_CODE 5200 / DISABLE_POWER NO"
```

(if a KL4002 2 channel analog terminal were used then the 'output' size would be set to 4. If you are adding the analog output terminal to an existing node or your node MACID is other than '1' your MACID statement will differ from the above)

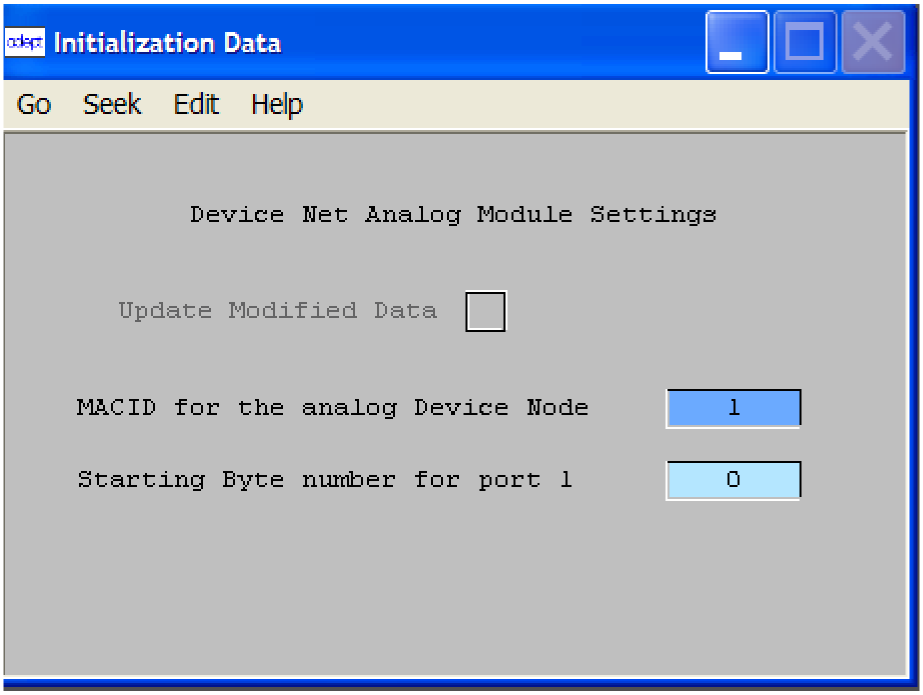
17. Exit this menu and select 'Done Editing'. Answer 'y' when prompted to 'save modified configuration data to disk (Y/N)?
18. Exit the Adept System Configuration Utility and reboot the controller. When the reboot process is completed and V+ is finished initializing the green 'CONNECT' led on the BK5200 coupler will be lit. You can also use the V+ monitor command 'devicenet' at this time to view the DeviceNet activity. If you are correctly configured node 1 will show up in the 'Installed MAC ID' list as 'Device being scanned'

Configuring the Beckhoff DeviceNet analog I/O for use by Pathware

Once the KL4001 or KL4002 analog output terminal is configured for access on Adept DeviceNet you must complete its configuration within Pathware. This requires specifying the node MACID and starting byte number of the analog output terminal within the DeviceNet node image map.

To configure the Beckhoff DeviceNet analog I/O for use by Pathware follow the below listed procedure.

1. Select the ‘Analog Module Settings’ record within the AIM Initialization databases, ‘phwini’ database selection. The record looks as follows:



Device Net Analog Module Settings

Update Modified Data □

MACID for the analog Device Node 1

Starting Byte number for port 1 0

2. If your Adept DeviceNet configuration was per the above procedure then the default values of ‘MACID = 1’ and ‘Starting Byte = 0’ are correct. If you added the analog output terminal to an existing node or your node MACID is other than ‘1’ your values will be different. The ‘Starting Byte = 0’ signifies that the KL4001 / KL4002 analog output terminal is the first ‘byte’ oriented terminal to the right of the bus coupler
3. Shutdown and restart AIM.

Teaching Path Segments With the Teach Pendant

The location data for paths are stored in the Process Path database. Like other AIM databases, there are many ways of defining locations in database records. One approach is to directly enter the locations into the Process Path database. Another option is to download the locations from a CAD system using the ADX Import features.

The most commonly used method of creating path locations is to teach the locations using the manual control pendant. To teach a path location, press the button on the Process Path menu page or the Path Segment menu page. The locations can be taught for an entire path without having to return to the AIM terminal. Additionally, database records can be appended, inserted, and deleted.

To teach path segment locations using the PathWare teach routines, follow these steps:

1. Open the desired dispensing path:

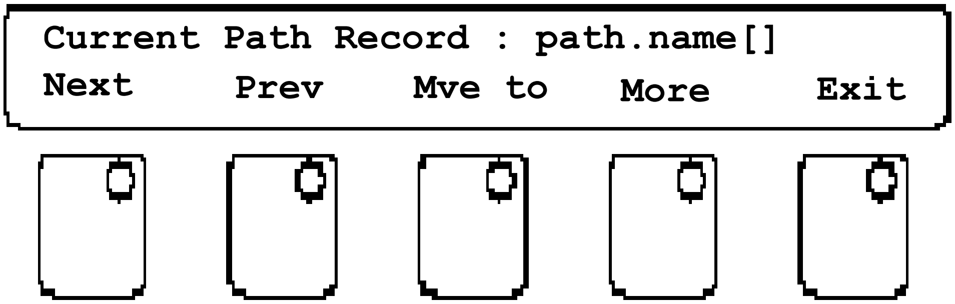
   Edit ➪ Process Path ➪ Seek ➪ Index ➪ *double click on path*

2. Press the Process Path menu page, or open a segment record and press **Teach** .

   If the teach routine is entered during walk-thru training, database path records cannot be created or deleted. However, the location fields can be modified.

   If a tool record has been specified for the path, the first menu selection will allow the setting of the tool offset using the pendant.

3. If a path segment record exists, the pendant will display the menu options shown in **Figure 4-21**. If no records exist, an error will be displayed in the monitor window. To clear an error, press and proceed with the next step.



```
Current Path Record : path.name[]
```

Figure 4-21. PathWare Teaching Utility Menu

4. Press *Next* to go to the next path segment, *Prev* to go to the previous path segment, *Mve To* to move the robot to the current segment location, *More* to display the next pendant menu, or *Exit* to exit the pendant teach routine. (The pendant must be in COMP mode.)

5. If *Mve To* is pressed, the pendant displays the menu options shown in **Figure 4-22**. This option allows you to move to the displayed record position by pressing the "+" speed control pot. If the "–" speed control pot is pressed, the robot moves back to the previous position. If the location in the record has not been defined, the teach pendant will display an error.

6. If *More* is pressed, the pendant displays the following menu options that are shown in **Figure 4-22**.

   Insert — insert a path segment before the segment displayed on the pendant. Once insert is pressed, the pendant will display "Segment Before: PATH.NAME[num]" and the options available are "OK" and "Cancel". Press "OK" and a new record to be inserted and displayed on the pendant.

   Append — add a new record to the end of the path. The pendant will display "Segment Added After: PATH.NAME[num]". Press "OK" and a new record to be added to the end of the path will be displayed on the pendant.

   Teach — move the robot to the desired location using the pendant. Record the current robot position in the path segment record by pressing the "REC/DONE" button. Abort the teaching operation by pressing "Cancel". Refer to **Figure 4-22** for teach pendant display.

   More — (See next step)

7. If "More" is pressed from the second pendant menu, the display shown in **Figure 4-22** will appear. The following options are available:

Gun State	toggle between the On/Off states for the path segment. Press the Gun button and the teach pendant display will reflect the change.
Arc Path	toggle between straight line path segments and arc segments. (Three locations are required to define two consecutive arc segments. Refer to the section on Specifying Arc Segments.) The pendant will display "Yes" or "No" to indicate whether the segment is an arc segment.
Delete	delete the record displayed on the pendant. When the "Delete" button is pressed, "Delete" or "Cancel" options will be displayed.
More	(See next step)
Done	step up to the previous menu.

8. If "More" is pressed from the second pendant menu, the display shown in **Figure 4-22** will appear. The following options are available:

Speed	change the robot speed to this path segment. The pendant menu will display "+", "–" and "Done" options. Press the "+" button to increase speed or the "–" button to decrease speed. The pendant display will show the current speed. Press the "Done" button to return to the previous menu.
Appro	change the approach height. The pendant will display another menu that allow you to enter or teach the approach height.
Depart	change the depart height. The pendant will display another menu that allow you to enter or teach the depart height.

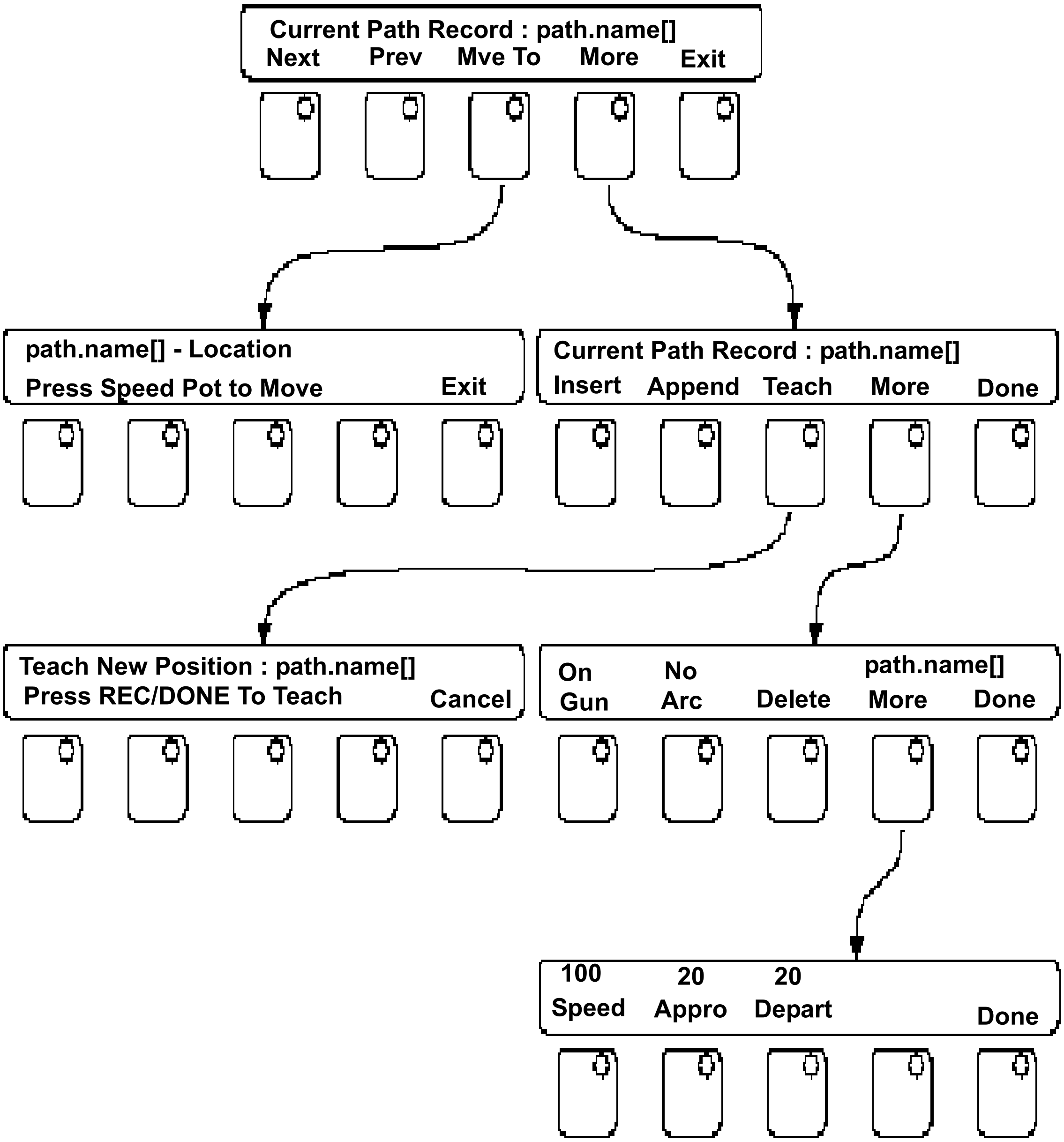


Figure 4-22. Pendant Path Teaching Utility Tree

This teach routine provides most of the functionality that is available from the Process Path menu page. These teach pendant routines operate only with the Process Path database and cannot be used with other databases. If a general path is being taught, the standard AIM pendant routines are used.

Teaching Tool Transformations

The Process Path database *tool* field specifies the tool to be used for moves to locations in the path. The *tool* field is linked to the Tool database. A tool can be specified directly by double-clicking on the Tool data box, and then retrieving a tool record from the displayed pick list. The tool transformation in a Tool record defines the offset from the robot tool flange to the dispensing gun tip. Accurate tool offsets are crucial for precise dispensing. The tool offset should be measured before teaching locations in the Location or Process Path databases.

Tool transformations are required when using downloaded CAD data, since locations downloaded represent the center of the robot tool flange, not the tip of the dispensing nozzle. Tool transformations are required for all applications that are not mounted on the center of the tool mounting flange. Constant velocity paths cannot perform properly without tool transformations with offset tooling. Tool transformations also make it easier to change dispensing guns. If all path segments have been created with a tool transformation in effect that describes the original tool, simply creating and using a tool transformation for the new tool will allow you to use all previously created path points.

Figure 4-23 shows a tool transformation for an Adept Robot. If no tool transformation is in effect, the X-axis of the tool frame points along the tool flange key-way and, the z-axis points down. In this example, the tool transformation (defined by the X′, Y′, and Z′ coordinate system) has offsets in the X, Y, and Z directions, but no change in the y, p, and r components. However, tool transformations can have values for all six transformation components.

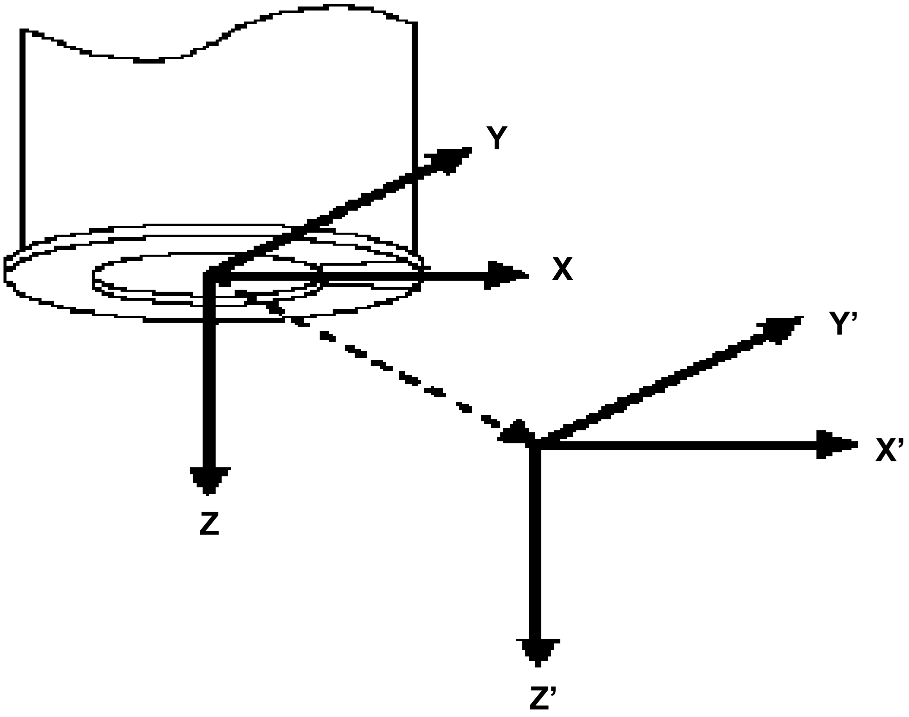
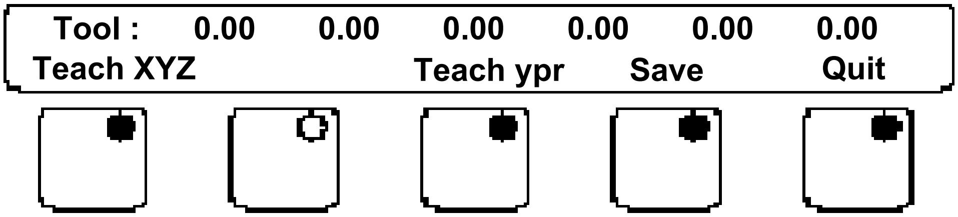


Figure 4-23. Cartesian Tool Offset

To teach a tool transformation, open a tool record and press ***Teach Tool***. The menu shown in **Figure 4-24** will be displayed on the pendant.



```
 Tool :    0.00    0.00    0.00    0.00    0.00    0.00
Teach XYZ                Teach ypr    Save         Quit
```

Figure 4-24. Tool Offset Utility Menu Display

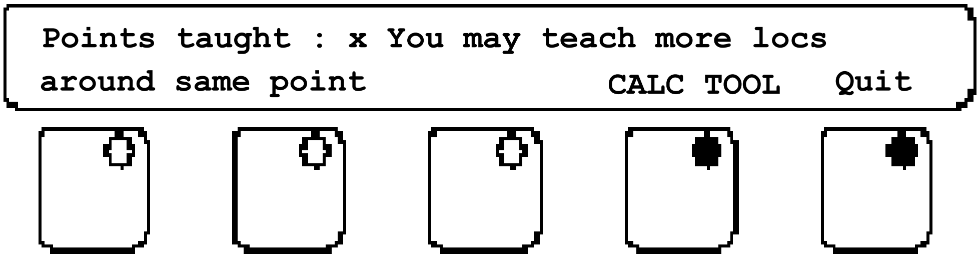
Both Cartesian (X, Y, Z) offsets and orientation (yaw, pitch, roll) offsets can be measured. To measure the Cartesian offsets:

1. Press the TEACH XYZ soft key.

2. The pendant will prompt you to teach a series of locations. Place the gun tip at a given location and press the pendant REC/DONE key. Rotate the gun tip (at least 30^{o}) about this given location and press REC/DONE again. A minimum of two locations must be taught to measure the offset. However, more accurate results result if more locations are taught. (The robot "FREE" mode can be used to simplify the teaching process.)

   Ideally, the tool offset should be relative to the location where the dispensed material hits the part surface. However, it is usually much easier to teach the offset to the dispensing gun tip. Defining the tool transformation at the gun tip is normally precise enough for accurate dispensing results.

3. Once the locations are taught, press the "CALC TOOL" soft key (**Figure 4-25**). The locations are "averaged" and areused to define the tool offset. The routine will give you the option of not using the location farthest away from the calculated center in the tool transformation calculation. Press "QUIT" to return to the previous menu.



```
Points taught : x You may teach more locs
around same point              CALC TOOL   Quit
```

Figure 4-25. Calculating XYZ Offsets

If a dispensing gun is attached at an angle with respect to the robot tool flange, and the gun deposits material along an axis that is not aligned with the normal tool Z-axis of the robot not parallel to the tool coordinate system (i.e., not perpendicular to the robot tool flange surface), the orientation offset might need to be measured as well. Typically, the orientation offset is not required if the dispensing path locations are taught manually. **Figure 4-26** shows an example of a tool transformation that contains yaw, pitch, and roll components.

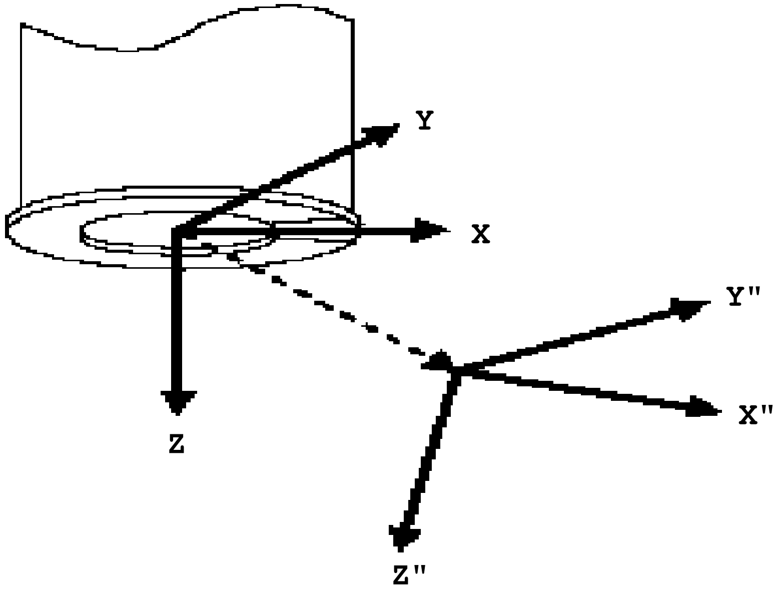


Figure 4-26. General Tool Offset

To teach the yaw, pitch, and roll components of a tool transformation:

1. Press the "TEACH YPR" soft key (see **Figure 4-24**).
2. The pendant will prompt you to teach two robot locations along the tool's Z-axis (the Z-axis in **Figure 4-26**.). Both locations must have the same orientation. Press the "REC/DONE" key to record the locations.
3. After both locations have been taught, you will be prompted to teach a location along the positive X-direction or the positive Y-direction of the tool (the X- or Y-axIs of **Figure 4-26** respectively). Again, the orientation of the tool should not be changed. Once the last location is taught, the yaw, pitch, and roll offsets are automatically calculated. Press the "QUIT" soft key to return to the top-level menu.

Press the "SAVE" soft key to record the measured transformation in the Tool record. Press the "QUIT" soft key to terminate the teaching operation.

Testing the Tool Offset

The method to test the tool transformation with PathWare software is the same as the standard AIM Package. To test the tool transformation:

1. Open a Tool record:
   Edit ⇨ Tool ⇨ Seek ⇨ Index ⇨ *double click on desired record*

2. Press to set the Set Tool specified in the Tool Menu.

3. From the pendant, choose the TOOL manual mode to move the robot around the defined tool location. If the tool offset is correct, the robot should rotate about its tool tip when rotated about X, Y, and Z, and it should move correctly along the tool X-, Y-, or Z-axis.

Other AIM Databases Used by PathWare

The Reference Frame Database

The Process Path database *frame* field specifies the reference frame used for frame-relative motions. Path locations can be taught with respect to the transformation defined in the *frame* field.

See the ***MotionWare User's Guide*** for details on creating Frame database records.

If process path locations are to be relative to a reference frame, the frame record should be assigned to the Location database and the Process Path. Frames should be defined before teaching locations or path segments that are relative to the reference frame.

If your system includes the vision module, the SET_FRAME statement can be used to define a reference frame. Vision reference frames must be determined before teaching the path. Walk-thru training allows you to define a vision reference frame.

The path planner does not recalculate the path if the reference frame data changes. Using the SET_FRAME statement, you can apply the same path to many different reference frames.

The Location Database

Robot locations are taken from the location database for using the Quick-Change module and for the Apply_Dot Statement. To teach a location, open a Location record and press the **Here** or **Teach** button. Approach and depart values must also be specified.

Sequence Execution

Before a sequence can be executed, it must first be created and then edited to include the appropriate statements to perform the desired dispense task. The sequences associated databases such as Location, Frame, and Conveyor must also be defined. See the ***MotionWare User's Guide*** for details on creating, copying, and deleting sequences and location databases.

Depending upon the statements used in a sequence, the following information must be defined before the sequence can be successfully executed (many of these items can be defined during walk-thru training:

- If a DISPENSE statement is included in a sequence, its Process Path database record must exist.

  Double-clicking on any record name in the sequence editor will take you to the editing menu page for that record type.

  Dispense path data points can be taught from within or outside of walk-thru training.

  If a tool or reference frame is specified in the Process Path database record, the corresponding records must be defined in the Tool and Reference Frame databases. Typically, tool offsets and reference frames are taught using an AIM utility when a sequence is not executing.

  The PathWare Signal Database must exist if the APPLY_DOT or DISPENSE statement is being used. If the sequence is run without the signal database, an error will occur.

- APPLY_DOT statement locations are defined in the Location database associated with the sequence.

The above statements can also use safe paths from the path database. These paths are used to approach or depart from the dispensing area. See the ***MotionWare User's Guide*** for details on defining safe paths.

DISPENSE Statement Execution

There are some differences between AIM Base Package and PathWare sequence execution. Most of the differences pertain to the DISPENSE statement. Here is the list of differences:

- During walk-thru-training, if **Edit Data** and **Walk Thru** are selected, the DISPENSE statement will calculate the path for each loop through the sequence. If changes are made to the Process Path database, the changes are incorporated into the path planning automatically.
- If a record in the Process Path or PathWare Signal database is changed, the DISPENSE statement will always compute the path during sequence execution until the "Save All Dbs" option is selected, saving database modifications to disk. If a modification is made during debug execution, make sure a DISPENSE statement is executed once and a new path is calculated before selecting the "Save All DBs" option. This ensures that the changes are incorporated into a newly interpolated path.
- If the pause after **Action** is selected, the robot will stop at the taught dispense path locations.
- If the pause after **Operation** is selected, the robot will stop at all of the interpolated path locations defined by the path analyzer.
- If the robot speed is set from the control panel to a value less than 100, robot speed will be reduced proportionally. However, the interpolated path points calculated by the DISPENSE statement will not change. If the value is set greater than 100, the robot will not traverse the dispensing path any faster than the speeds defined in the Process Path database. For example, if the Dispense Path specifies a path speed of 100 mm/s, and the robot speed is set to 50, the robot will move along the path at 50 mm/s (50% x 100 mm/s = 50 mm/s). If the robot speed is set to 150, the robot will move along the path at 100 mm/s. It is highly recommended to change path speeds from the Process Path database rather than the robot speed field in the Task control panel.
- The PathWare Gun push-button on the Debug Control Panel allows you to turn off the dispensing gun while debugging sequences.

Application Example 1

To review the topics discussed in this chapter, consider the example part shown in Figure 4-27. The part requires that two beads be dispensed along complex paths, and that two dots of material be applied in the corners. For this example, assume that the dispensing gun is not attached to the robot before sequence execution and a single DISPENSE statement is used to deposit the two beads. Also assume that the dispensing gun tip is offset from the robot tool flange when the gun is attached to the robot.

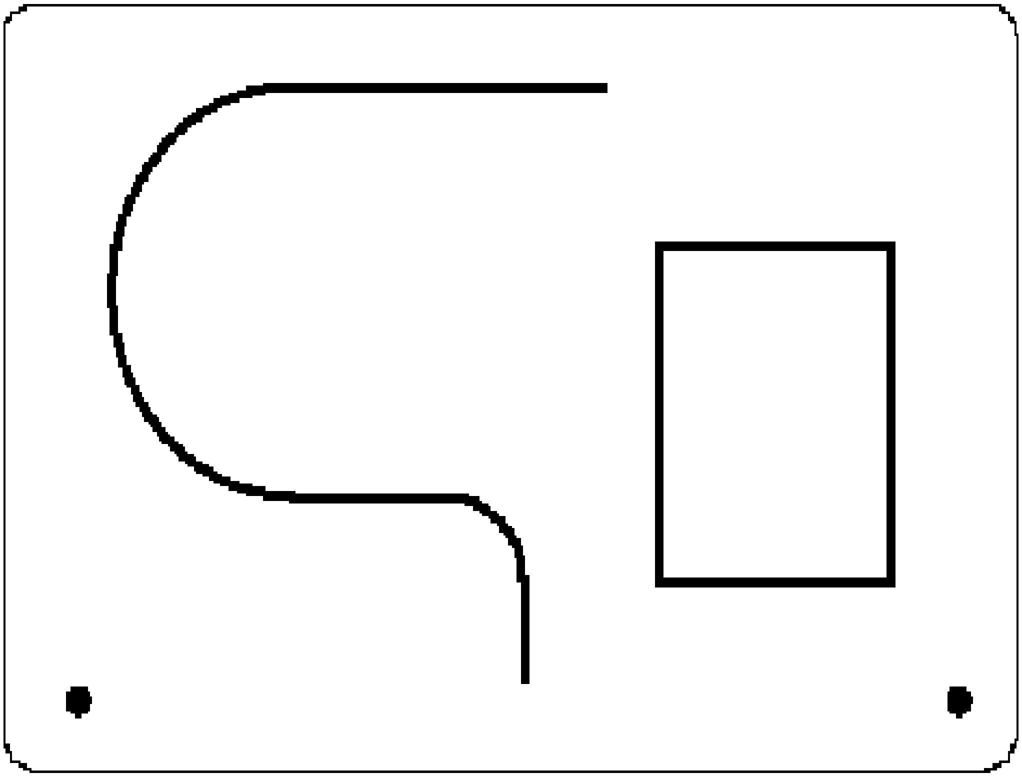


Figure 4-27. Example Part

Before editing a new sequence, a *map* of the application should be defined to determine the data requirements. The map will help determine the number and types of statements to use, and how to divide the application into path segments. Figure 4-28 is an example map of the locations needed to perform this dispensing task. Two APPLY_DOT statements are used to dispense the dots of material in the two corners and a single DISPENSE statement (using a process path with 18 locations) is used to apply the two beads.

In Figure 4-28, the arrows define the direction of robot motion along the dispensing path. The locations outside of the path are defined so that the robot is in motion when the dispensing gun is triggered, preventing material from accumulating at path end locations.

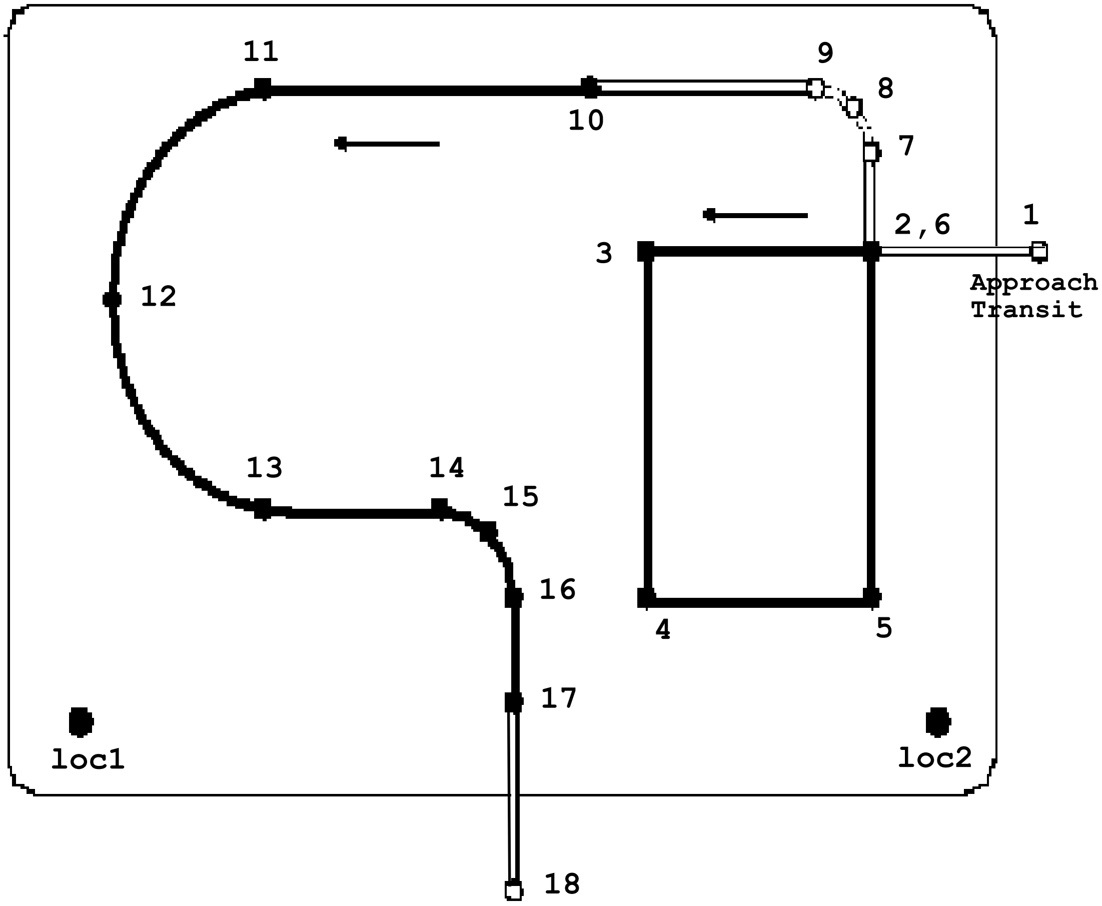
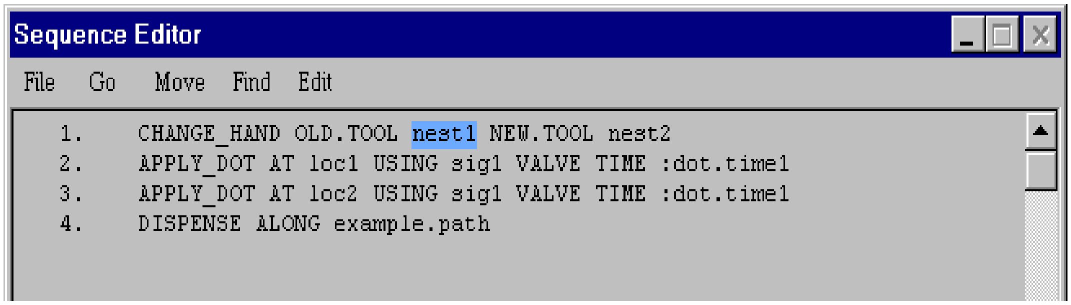


Figure 4-28. Map of the Example Part

Example Sequence for First Part

The following AIM task statements will be used to perform this dispensing application. Provided below is the actual sequence used for this example.



```
1.    CHANGE_HAND OLD.TOOL nest1 NEW.TOOL nest2
2.    APPLY_DOT AT loc1 USING sig1 VALVE TIME :dot.time1
3.    APPLY_DOT AT loc2 USING sig1 VALVE TIME :dot.time1
4.    DISPENSE ALONG example.path
```

Figure 4-29. Example Sequence 1st Part

The CHANGE_HAND statement is included for showing the optional Quick-Change module, which is discussed later in this manual.

Sample Database Records

This section details creating records for the above sequence. The data defined in the records can be taught using the methods described in the *Database Editing* section of this chapter.

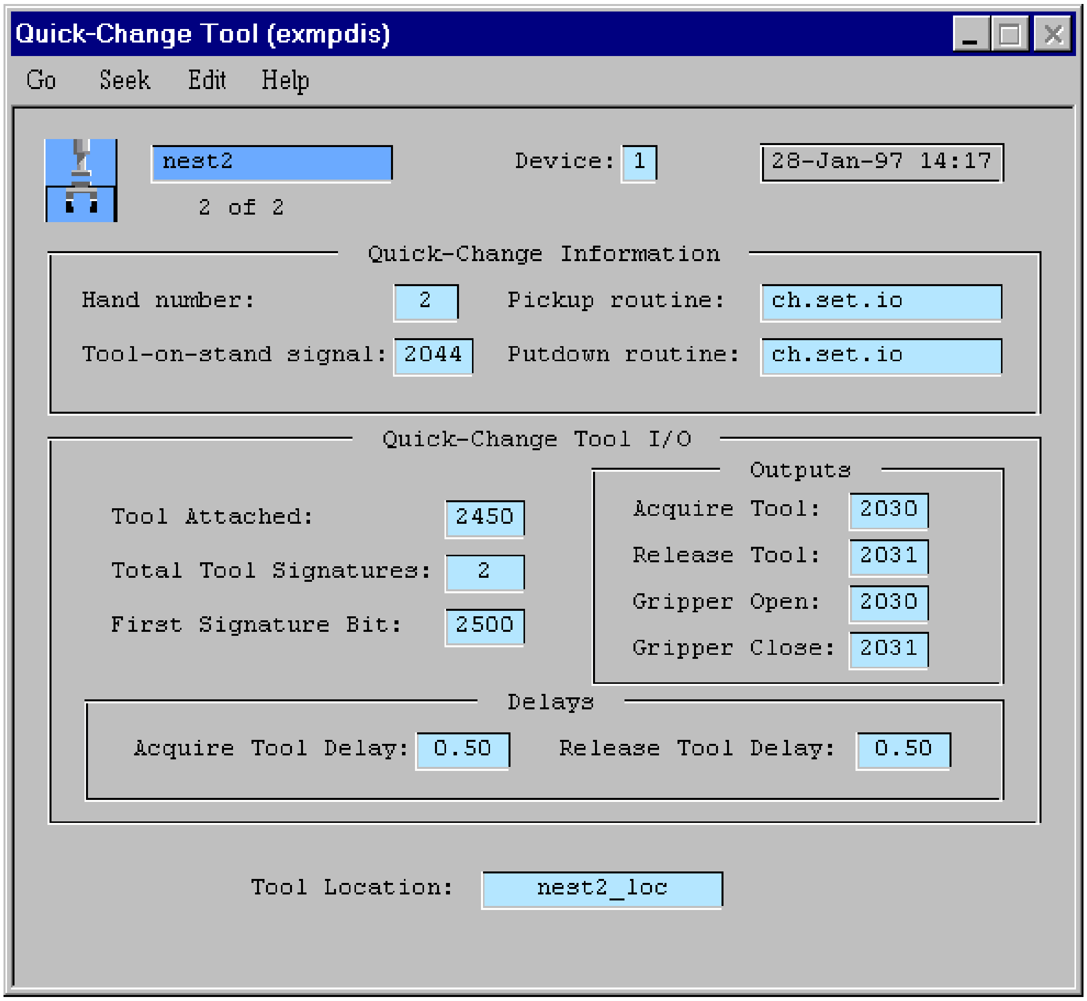
Quick-Change Database

In our example application, the CHANGE_HAND statement is used to pick up the dispensing gun (see **Chapter 6**). Source fields MUST be defined before linking can be successful if the tool is specified in the Dispense Path database. Please refer to **Figure 4-30** for the example record menu. Important fields used by the CHANGE_HAND statement include:

Tool Location This is the nesting location for the tooling when not being used by the robot. The Quick-Change database is linked to the location database to use positions taught there. In this example the location record is called *nest2_loc*. You can branch or select the location record by double-clicking on this field.

Pickup routineThis program is executed after the end-effector is picked up from its nest. The program might set the dispensing gun signal low so that the material is not accidentally dispensed when the tool is first attached to the robot. System customizers typically will write this program; the required calling sequence is described in **Chapter 6**.

Hand number If there are multiple quick-change end-effectors in the workcell and tool signature signals are used, this parameter uniquely identifies this tool. The hand number is used to verify that the robot picks up the correct tool.



Quick-Change Tool (exmpdis)

nest2 Device: 1 28-Jan-97 14:17

2 of 2

Quick-Change Information

Hand number: 2 Pickup routine: ch.set.io

Tool-on-stand signal: 2044 Putdown routine: ch.set.io

Quick-Change Tool I/O

Tool Attached: 2450

Total Tool Signatures: 2

First Signature Bit: 2500

Outputs

Acquire Tool: 2030

Release Tool: 2031

Gripper Open: 2030

Gripper Close: 2031

Delays

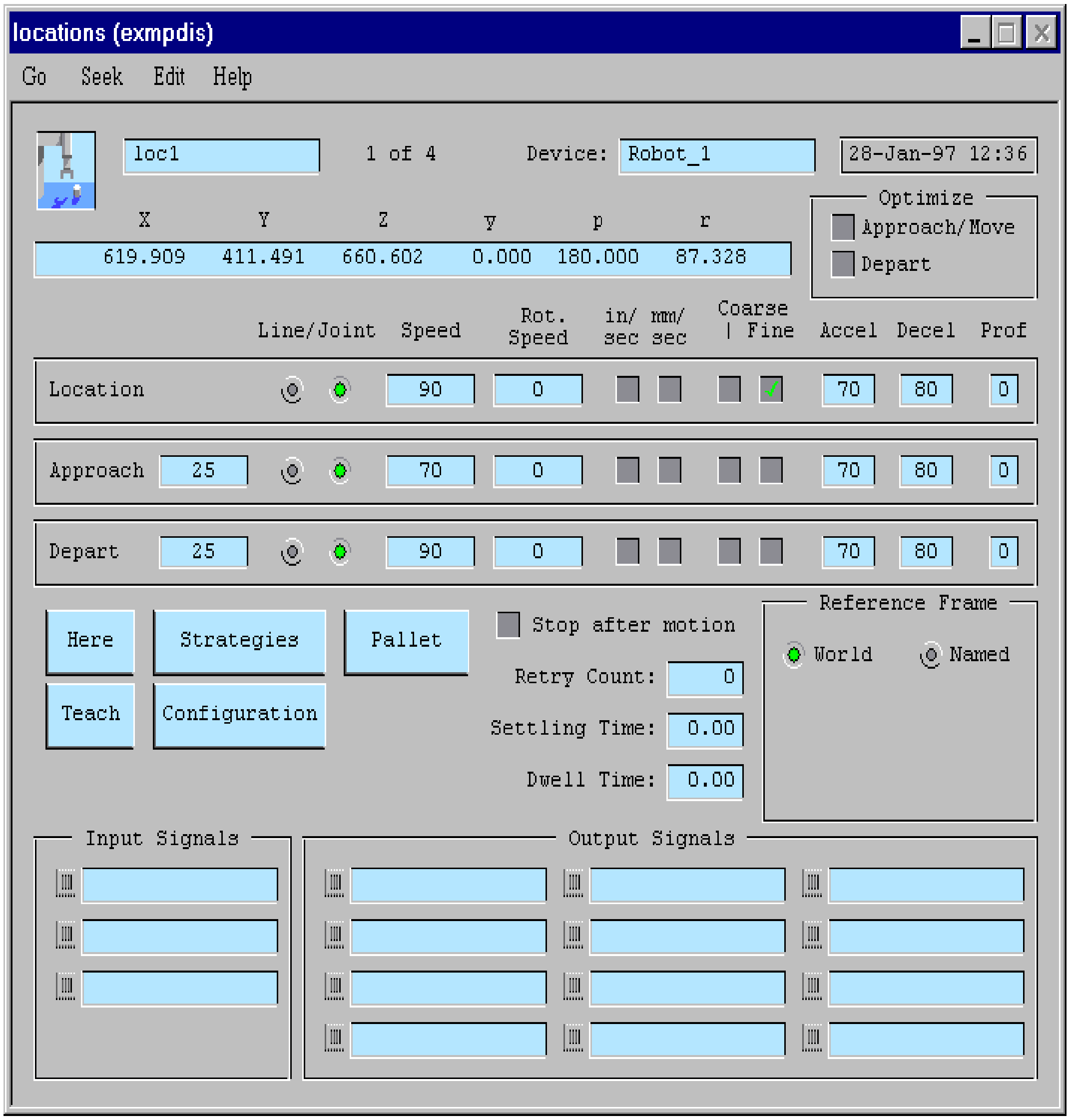
Acquire Tool Delay: 0.50 Release Tool Delay: 0.50

Tool Location: nest2_loc

Figure 4-30. Quick-Change Example 1

Location Database

The APPLY_DOT statement dispenses the dots of material at the corners of the example part. Figure 4-31 shows the Location database record for *loc1*. The *location* field specifies the material dispensing location. The amount of volume dispensed is based on the options used in the Dispense Signal database. In this case the *valve time* parameter used in the sequence statement is used for the valve on-time. Please note that this value is defined in the Variable database using the name:*Dot.Time1.*



loc1 1 of 4 Device: Robot_1 28-Jan-97 12:36

X	Y	Z	y	p	r
619.909	411.491	660.602	0.000	180.000	87.328

Optimize
□ Approach/Move
□ Depart

		Line/Joint		Speed	Rot. Speed	in/ sec	mm/ sec	Coarse	Fine	Accel	Decel	Prof
Location		○	●	90	0	□	□	□	☑	70	80	0
Approach	25	○	●	70	0	□	□	□	□	70	80	0
Depart	25	○	●	90	0	□	□	□	□	70	80	0

□ Stop after motion
Retry Count: 0
Settling Time: 0.00
Dwell Time: 0.00

Reference Frame
● World ○ Named

Figure 4-31. Location Record Example 1

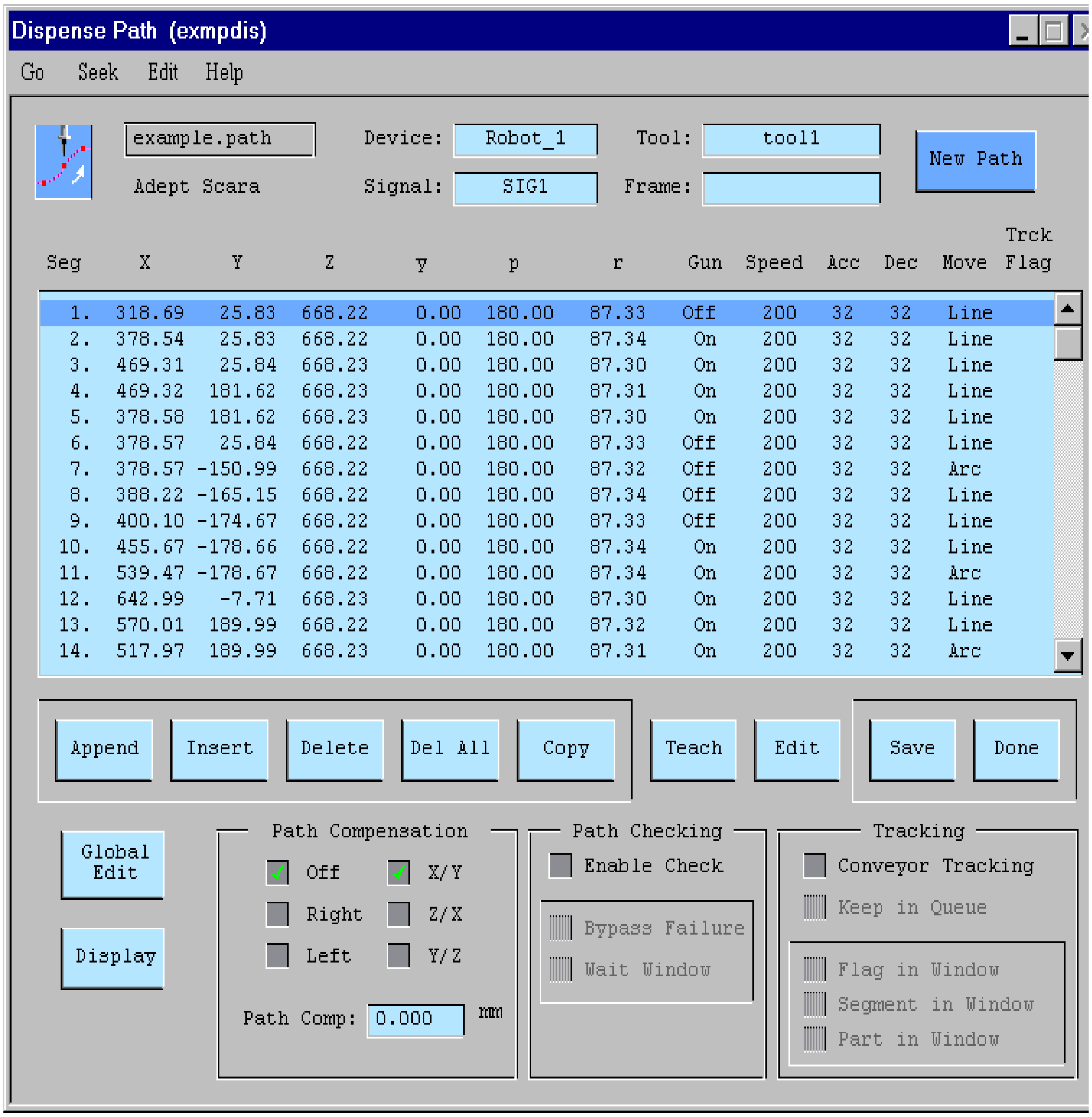
Process Path Database

The example part requires 18 path positions to deposit the two beads of material. Figure 4-32 shows the Process Path database record for the *example.path* DISPENSE process.

Locations 1, 7, 8, 9, and 18 are referred to as Transit locations. The Transit locations are specified to ensure the robot is moving while turning the dispensing gun on and off.

The other key fields are on the main menu:

Tool field used because the dispensing gun tool tip is offset from the robot tool flange.



example.path

Adept Scara

Device: Robot_1

Signal: SIG1

Tool: tool1

Frame:

Seg	X	Y	Z	y	p	r	Gun	Speed	Acc	Dec	Move	Trck Flag
1.	318.69	25.83	668.22	0.00	180.00	87.33	Off	200	32	32	Line	
2.	378.54	25.83	668.22	0.00	180.00	87.34	On	200	32	32	Line	
3.	469.31	25.84	668.23	0.00	180.00	87.30	On	200	32	32	Line	
4.	469.32	181.62	668.23	0.00	180.00	87.31	On	200	32	32	Line	
5.	378.58	181.62	668.23	0.00	180.00	87.30	On	200	32	32	Line	
6.	378.57	25.84	668.22	0.00	180.00	87.33	Off	200	32	32	Line	
7.	378.57	-150.99	668.22	0.00	180.00	87.32	Off	200	32	32	Arc	
8.	388.22	-165.15	668.22	0.00	180.00	87.34	Off	200	32	32	Line	
9.	400.10	-174.67	668.22	0.00	180.00	87.33	Off	200	32	32	Line	
10.	455.67	-178.66	668.22	0.00	180.00	87.34	On	200	32	32	Line	
11.	539.47	-178.67	668.22	0.00	180.00	87.34	On	200	32	32	Arc	
12.	642.99	-7.71	668.23	0.00	180.00	87.30	On	200	32	32	Line	
13.	570.01	189.99	668.22	0.00	180.00	87.32	On	200	32	32	Line	
14.	517.97	189.99	668.23	0.00	180.00	87.31	On	200	32	32	Arc	

Path Compensation

☑ Off ☑ X/Y

□ Right □ Z/X

□ Left □ Y/Z

Path Comp: 0.000 mm

Path Checking

□ Enable Check

▨ Bypass Failure

▨ Wait Window

Tracking

□ Conveyor Tracking

▨ Keep in Queue

▨ Flag in Window

▨ Segment in Window

▨ Part in Window

Figure 4-32. Process Path Record Example 1

Path segment The gun field associated with each path

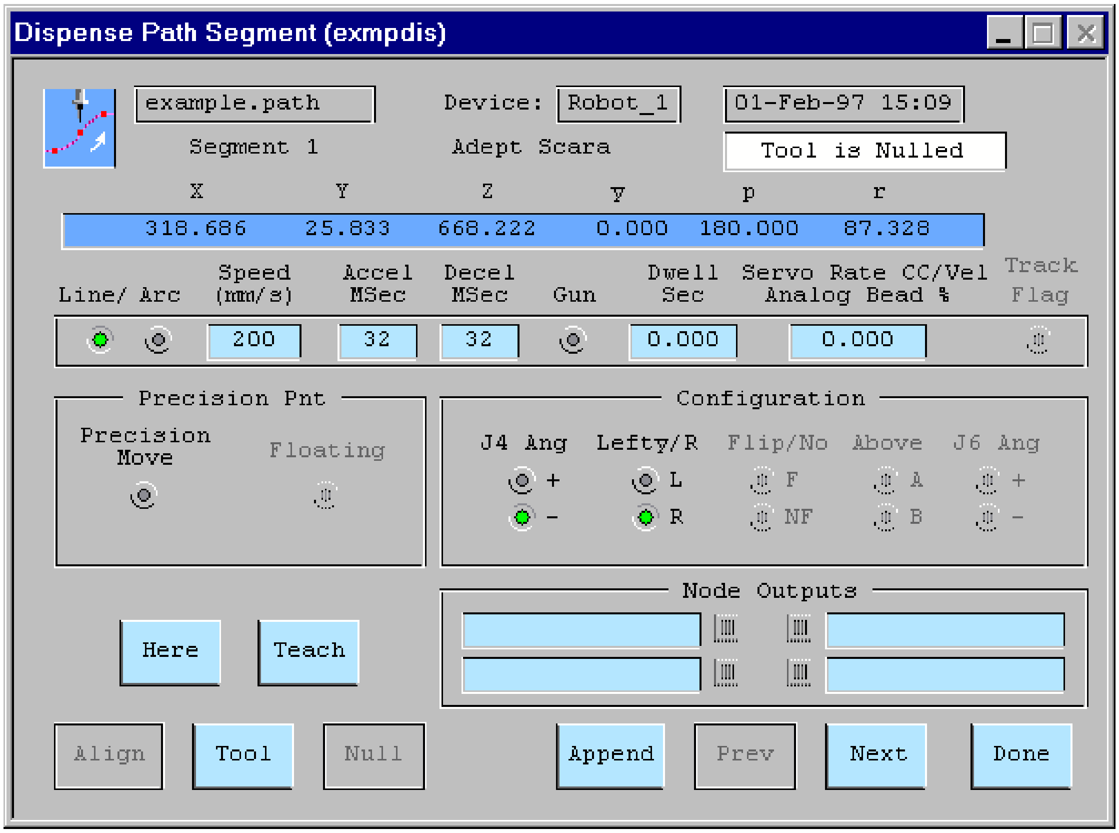
Motion paramssegment defines where the dispensing material is applied. The gun fields for locations 1, 6, 7, 8, 9, 17, and 18 are set to OFF. The gun state is displayed in the main menu and set in each individual path record.

Path Comp parameters are turned off unless the path needs modification.

Arc Points arc points are defined as three consecutive path segments in a path. Define the first segment as an arc point and the next two points as linear points. A circle is defined by five points, with points 1 and 3 being arc points.

Process Path Segment Record

Figure 4-33 shows a sample path segment record with gun compensation. Notice the settings for the gun state, line/arc motion, dwell time, speed, acceleration, deceleration, and bead percent.



Dispense Path Segment (exmpdis)

example.path Device: Robot_1 01-Feb-97 15:09

Segment 1 Adept Scara Tool is Nulled

X	Y	Z	y	p	r
318.686	25.833	668.222	0.000	180.000	87.328

Line/ Arc		Speed (mm/s)	Accel MSec	Decel MSec	Gun	Dwell Sec	Servo Rate CC/Vel Analog Bead %	Track Flag
●	○	200	32	32	○	0.000	0.000	○

Precision Pnt

Precision Move ○

Floating ○

Configuration

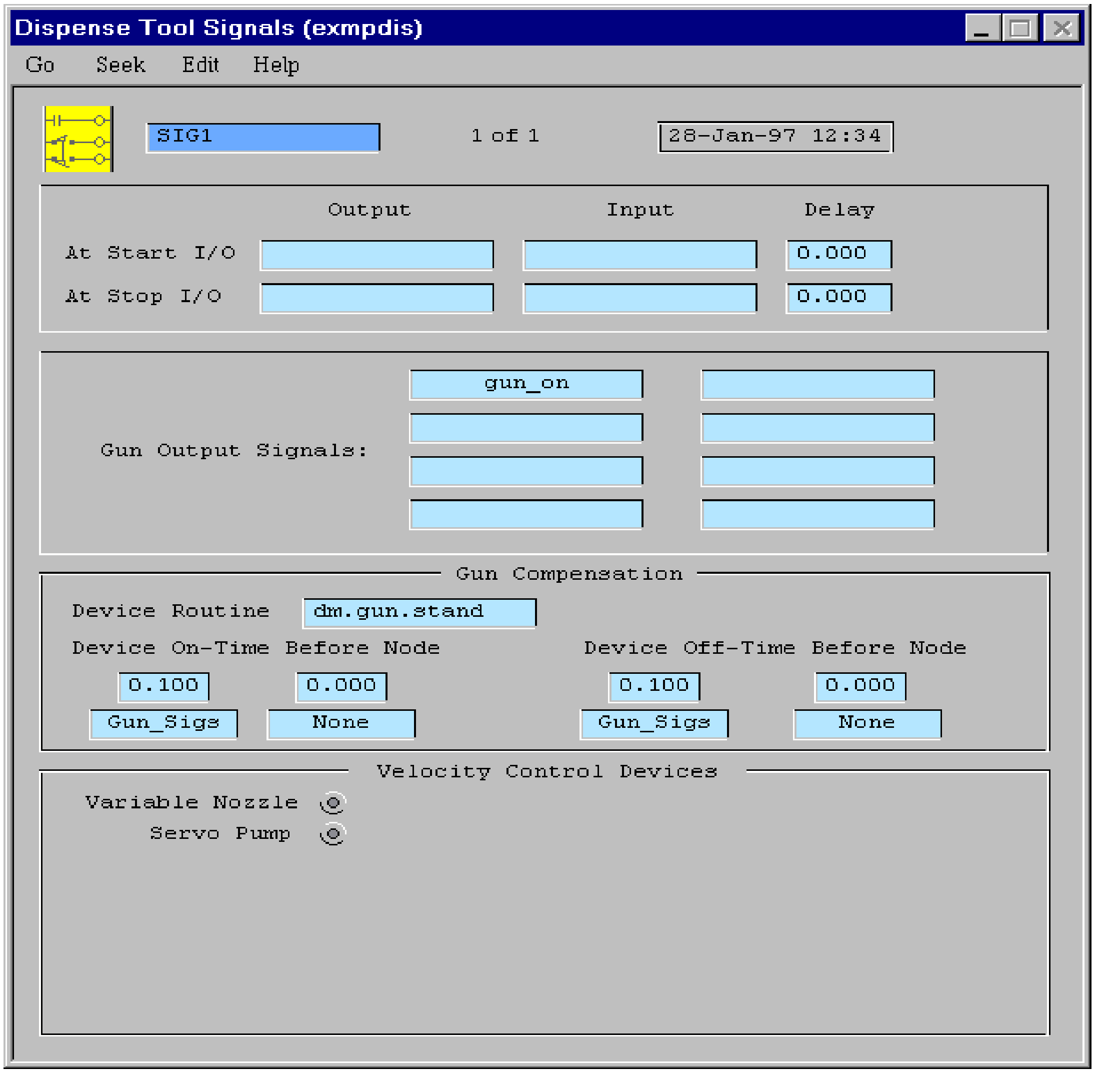
J4 Ang	Lefty/R	Flip/No	Above	J6 Ang
○ +	○ L	○ F	○ A	○ +
● -	● R	○ NF	○ B	○ -

Node Outputs

Figure 4-33. Process Path Segment Record Example 1

PathWare Signal Database

The APPLY_DOT statement and the DISPENSE statement use the signal database. Figure 4-34 shows a sample signal database record that uses the Variable database record name *gun_on* as the output signal for the valve. Figure 4-35 shows the Variable database record definition for the gun signal. Please note that the signal is set to the software signal 2400. Gun Compensation is used with this example and is defined in the Dispense Signal database under the *Gun Compensation* section. In this example the gun output signals turn on .1 second before the gun on node and off .1 second before the gun off locations.



SIG1 1 of 1 28-Jan-97 12:34

	Output	Input	Delay
At Start I/O			0.000
At Stop I/O			0.000

Gun Output Signals: gun_on

Gun Compensation

Device Routine dm.gun.stand

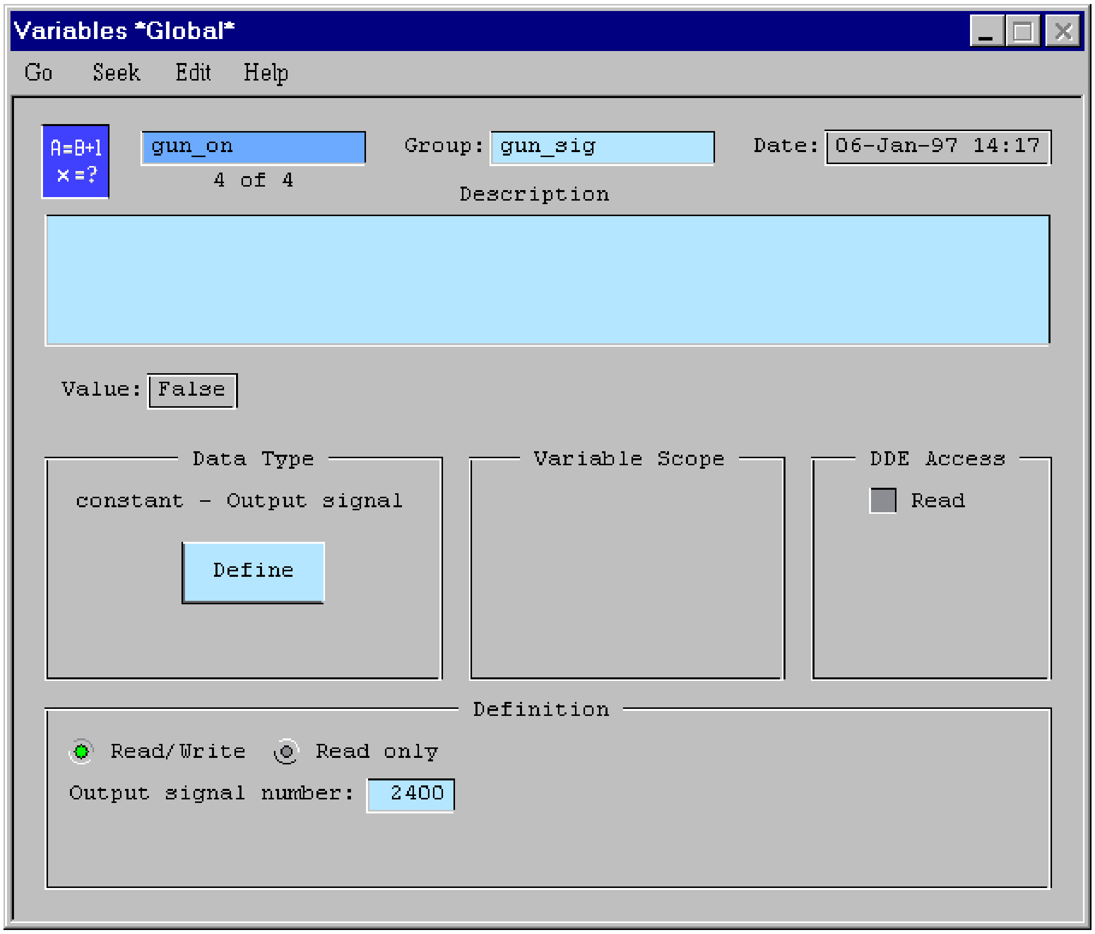
Device On-Time Before Node		Device Off-Time Before Node	
0.100	0.000	0.100	0.000
Gun_Sigs	None	Gun_Sigs	None

Velocity Control Devices

Variable Nozzle ○

Servo Pump ○

Figure 4-34. Example 1 PathWare Signal Record



gun_on

4 of 4

Group: gun_sig

Date: 06-Jan-97 14:17

Description

Value: False

Data Type

constant - Output signal

Variable Scope

DDE Access

▨ Read

Definition

● Read/Write ○ Read only

Output signal number: 2400

Figure 4-35. Variables Database for Valve Output

Conveyor Tracking Example 2

This example is provided for showing the required record for a conveyor tracking application. This example uses conveyor tracking with vision guidance. Several items are not shown with this example, such as Belt and Camera calibrations, which must be defined with such an application. Please refer to the *MotionWare User's Guide* for greater detail on conveyor tracking and vision guidance with conveyor tracking.

The dispense pattern is shown in **Figure 4-36**. Notice that the frame of reference is located near the center of the part. This is because the frame of reference is used with this application. The vision system is locating the part with the blob finder tool which returns the center of mass of the part as well as the rotational orientation.

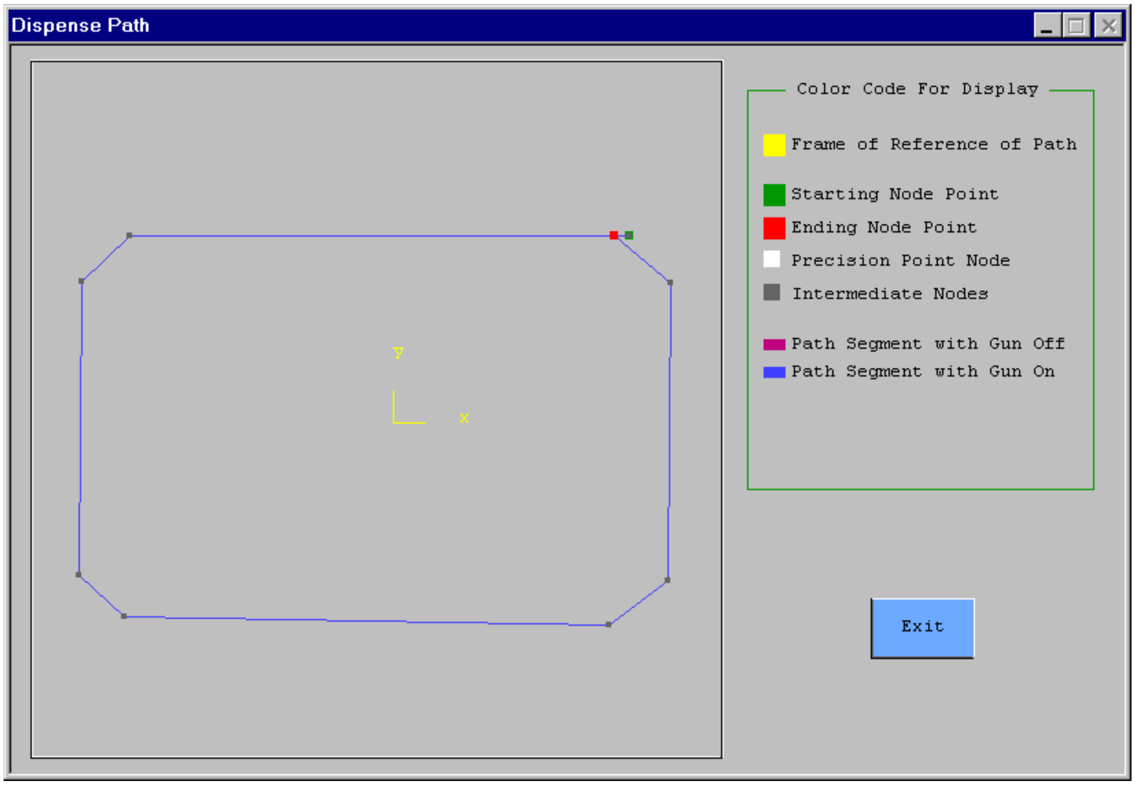
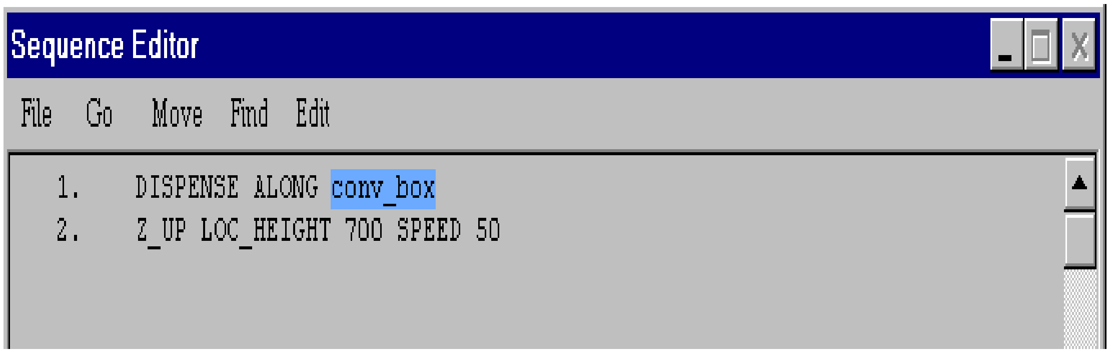


Figure 4-36. Path Pattern Example 2

Conveyor Tracking Sequence

The sequence used is shown in Figure 4-37. Because the databases are linked together the sequence is quite simple for this example. The Dispense Statement is all that is needed to handle the conveyor queuing and vision operation. With applications that use vision guidance without a conveyor, the sequence requires the SET_FRAME statement to acquire the image and compute the frame of reference.

With conveyor tracking a nontracking motion is required to detach the robot from the tracking frame. In this case the Z_UP statement was used to move directly above the last position. The standard MOVE statement could have been used in place of the Z_UP statement if desired.



```
1.    DISPENSE ALONG conv_box
2.    Z_UP LOC_HEIGHT 700 SPEED 50
```

Figure 4-37. Conveyor Tracking Sequence Example 2

The conv_box argument in the first line is the record name of the dispense path. The arguments 700 and 50 in the second line are constants corresponding to world coordinate z position and speed of the motion.

Process Path Example 2

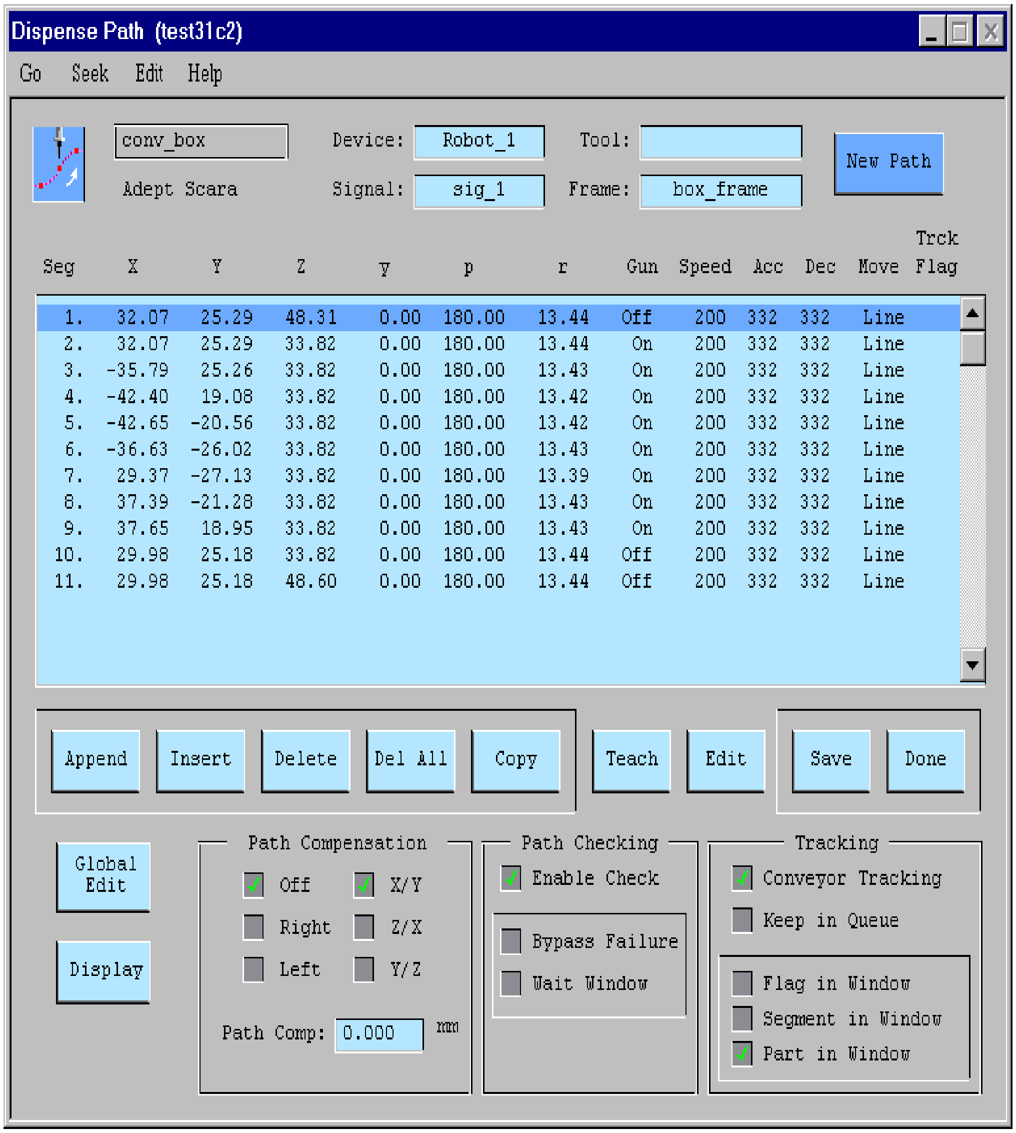
The dispense path shown in Figure 4-38 shows several of the items that are required for conveyor tracking with PathWare. The following list describes items that are required for conveyor tracking applications.

1. The Frame field must be linked with a Frame of Reference record that is set up for conveyor tracking. Notice the Frame field contains *box_frame,* which is the name of the Frame record menu shown in Figure 4-39.

2. The *Conveyor Tracking* check box is turned on. This tells the dispense software that the frame of reference is a *Moving Frame*. Conveyor tracking will not work with the dispense module if this is not turned on. The selection of this box also allows several other features to be selectable.

3. The *Part in Window* check box is also enabled. This forces the dispense module to check all of the part node positions to assure the part is within the upstream window before allowing the path motion to begin. The *Segment in Window* and *Flag in Window* are also available for this purpose. Refer to the **"Conveyor Tracking With PathWare" on page 82** for more information.

Notice that the coordinates of the taught locations are no longer in robot world coordinates. Rather, they are relative to the frame of reference defined in the *Frame* field. Before running the application, be sure that the coordinates appear to make sense. If the frame of reference is in the center of the part, then the coordinates should be roughly half the size of the part.



conv_box

Adept Scara

Device: Robot_1

Tool:

Signal: sig_1

Frame: box_frame

Seg	X	Y	Z	y	p	r	Gun	Speed	Acc	Dec	Move	Trck Flag
1.	32.07	25.29	48.31	0.00	180.00	13.44	Off	200	332	332	Line	
2.	32.07	25.29	33.82	0.00	180.00	13.44	On	200	332	332	Line	
3.	-35.79	25.26	33.82	0.00	180.00	13.43	On	200	332	332	Line	
4.	-42.40	19.08	33.82	0.00	180.00	13.42	On	200	332	332	Line	
5.	-42.65	-20.56	33.82	0.00	180.00	13.42	On	200	332	332	Line	
6.	-36.63	-26.02	33.82	0.00	180.00	13.43	On	200	332	332	Line	
7.	29.37	-27.13	33.82	0.00	180.00	13.39	On	200	332	332	Line	
8.	37.39	-21.28	33.82	0.00	180.00	13.43	On	200	332	332	Line	
9.	37.65	18.95	33.82	0.00	180.00	13.43	On	200	332	332	Line	
10.	29.98	25.18	33.82	0.00	180.00	13.44	Off	200	332	332	Line	
11.	29.98	25.18	48.60	0.00	180.00	13.44	Off	200	332	332	Line	

Path Compensation
☑ Off ☑ X/Y
□ Right □ Z/X
□ Left □ Y/Z
Path Comp: 0.000 mm

Path Checking
☑ Enable Check
□ Bypass Failure
□ Wait Window

Tracking
☑ Conveyor Tracking
□ Keep in Queue
□ Flag in Window
□ Segment in Window
☑ Part in Window

Figure 4-38. Dispense Path Example 2

Frame of Reference Example 2

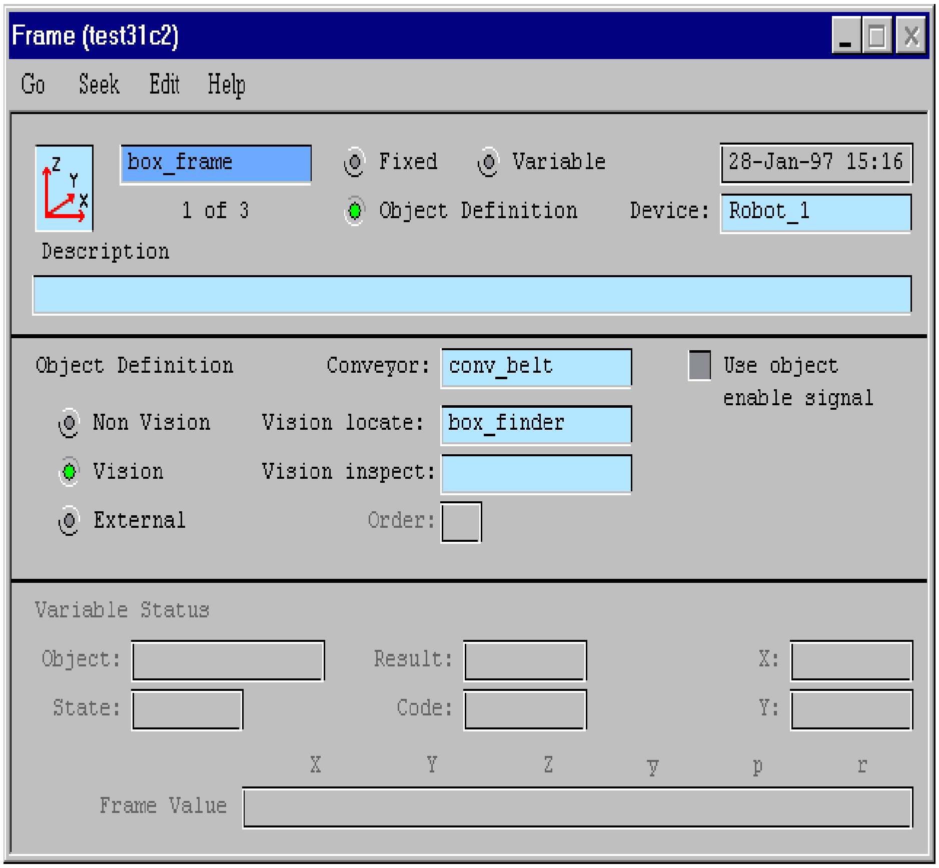
The Frame of Reference record used with this conveyor tracking example is shown in **Figure 4-39**. There are several items of interest that must be defined in this record for conveyor tracking. The following list describes these requirements.

1. The *Object Definition* radio button must be selected to allow the vision and conveyor options to appear for selection.

2. The *Vision* radio button must be selected to allow the vision pictures to function. Conveyor tracking without vision does not require this selection.

3. The *Conveyor:* field must be linked to a conveyor database. Shown in this example is the conveyor record conv_belt. This record is shown in **Figure 4-40**.

4. The *Vision:* field must be linked to a Top Level vision frame record. This record is shown in **Figure 4-41**. Additional information is available in the ***MotionWare User's Guide*** under Frames of Reference.



Frame (test31c2)

box_frame

1 of 3

○ Fixed ○ Variable

28-Jan-97 15:16

● Object Definition

Device: Robot_1

Description

Object Definition

Conveyor: conv_belt

□ Use object enable signal

○ Non Vision

Vision locate: box_finder

● Vision

Vision inspect:

○ External

Order:

Variable Status

Object: Result: X:

State: Code: Y:

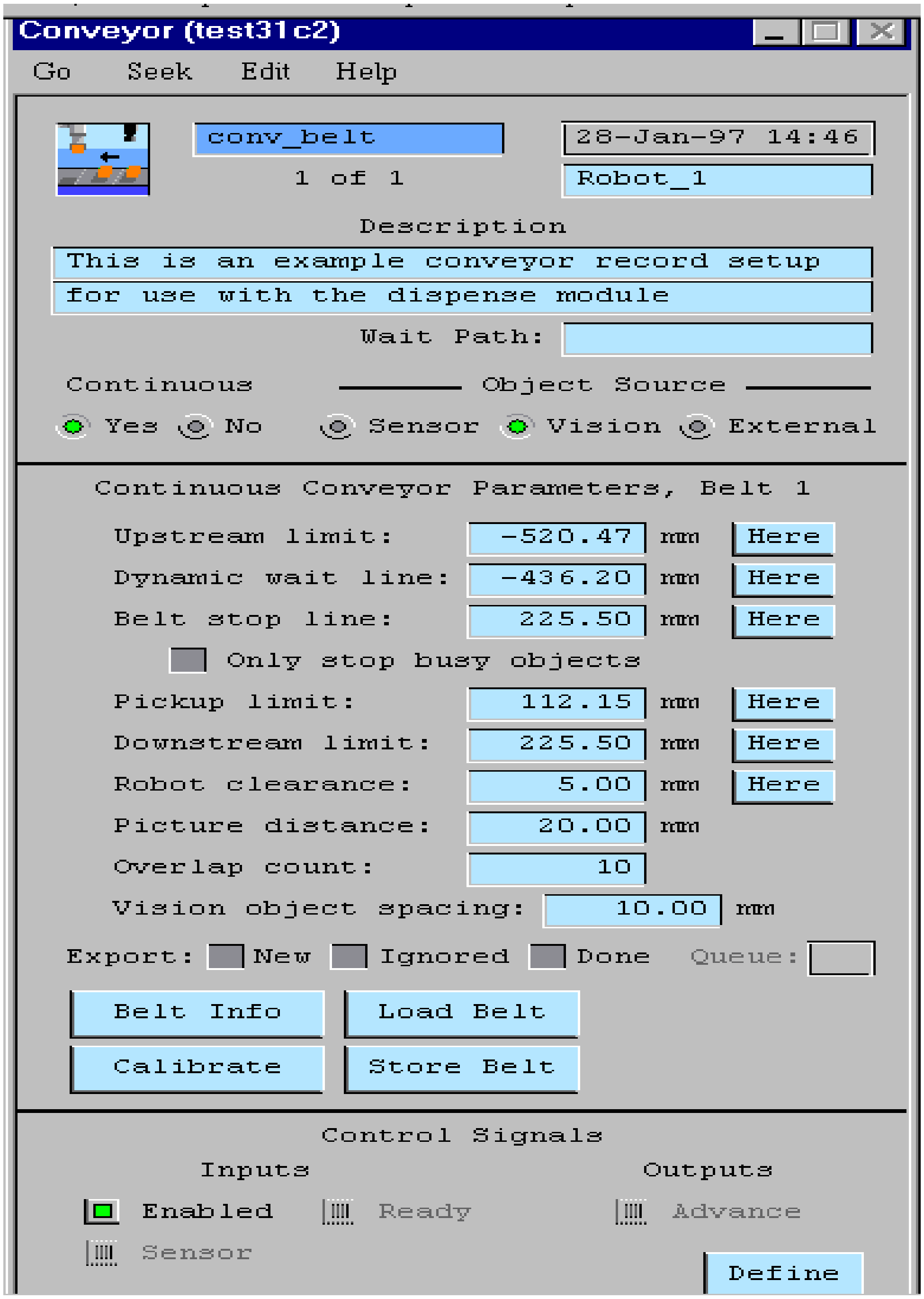
X Y Z y p r

Frame Value

Figure 4-39. Frame of Reference Example 2

Conveyor Record Example 2

The conveyor database is provided here for reference only. Refer to the ***MotionWare User's Guide*** for information on the Conveyor Database and all of the required field definitions for conveyor tracking. The Dispense Module supports all of the conveyor features allowed in the conveyor database.



conv_belt 1 of 1

28-Jan-97 14:46

Robot_1

Description

This is an example conveyor record setup for use with the dispense module

Wait Path:

Continuous: ● Yes ○ No

Object Source: ○ Sensor ● Vision ○ External

Continuous Conveyor Parameters, Belt 1

Upstream limit:	-520.47	mm
Dynamic wait line:	-436.20	mm
Belt stop line:	225.50	mm
Pickup limit:	112.15	mm
Downstream limit:	225.50	mm
Robot clearance:	5.00	mm
Picture distance:	20.00	mm
Overlap count:	10	
Vision object spacing:	10.00	mm

□ Only stop busy objects

Export: □ New □ Ignored □ Done Queue:

Control Signals

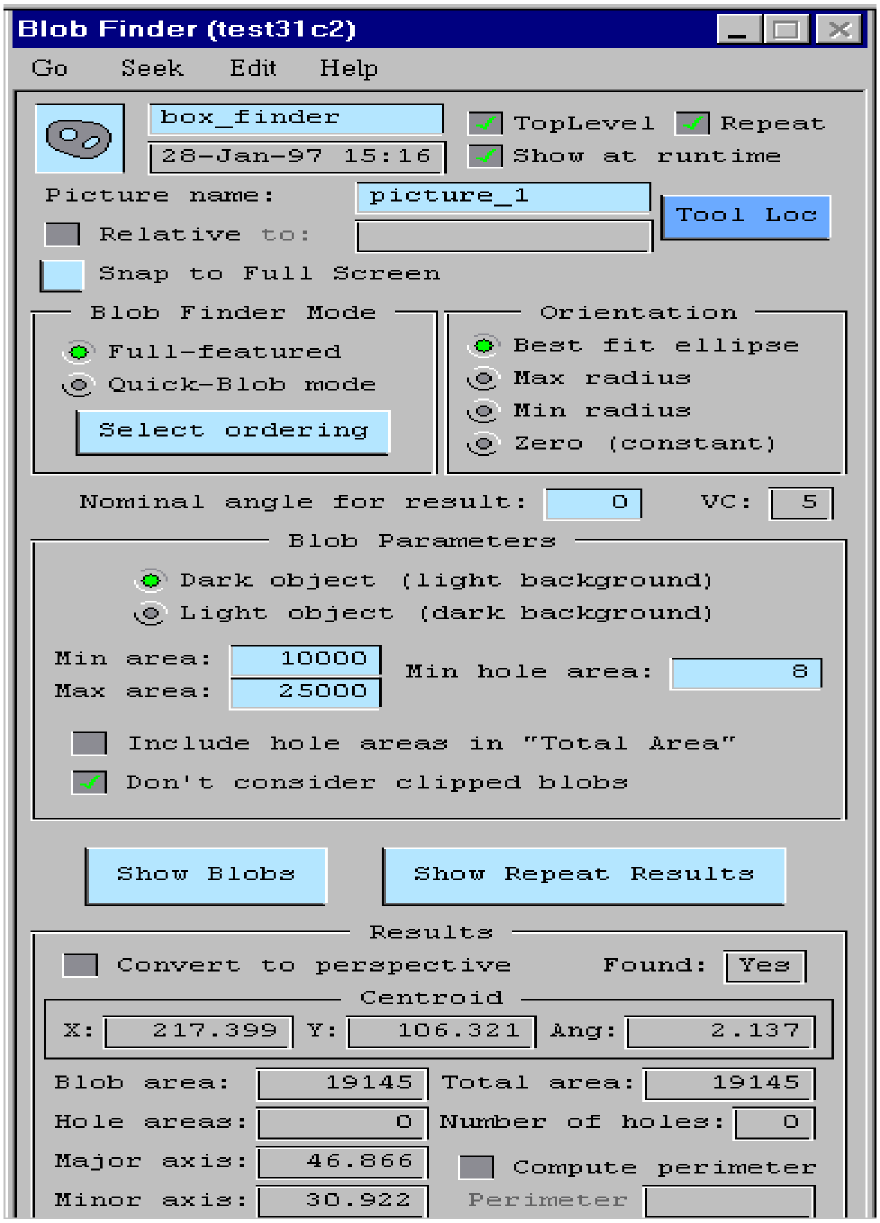
Inputs: ☑ Enabled □ Ready □ Sensor

Outputs: □ Advance

Figure 4-40. Conveyor Database for Example 2

Vision Blob Finder Record for Example 2

The Blob Finder vision record is provided here for reference only. Refer to the *MotionWare User's Guide* and the **VisionWare User's Guide** for more information on vision guidance and vision tool definition. Notice that the *TopLevel* and the *Repeat* check boxes are selected. The *TopLevel* box allows the record to be selected by Sequence statements and by the Frame record. The *Repeat* selection allows multiple parts to be found in the vision image. The vision image is shown in **Figure 4-42**.



box_finder
28-Jan-97 15:16
☑ TopLevel ☑ Repeat
☑ Show at runtime
Picture name: picture_1
□ Relative to:
□ Snap to Full Screen

Blob Finder Mode
● Full-featured
○ Quick-Blob mode

Orientation
● Best fit ellipse
○ Max radius
○ Min radius
○ Zero (constant)

Nominal angle for result: 0 VC: 5

Blob Parameters
● Dark object (light background)
○ Light object (dark background)
Min area: 10000
Max area: 25000
Min hole area: 8
□ Include hole areas in "Total Area"
☑ Don't consider clipped blobs

Results
□ Convert to perspective Found: Yes
Centroid
X: 217.399 Y: 106.321 Ang: 2.137
Blob area: 19145 Total area: 19145
Hole areas: 0 Number of holes: 0
Major axis: 46.866 □ Compute perimeter
Minor axis: 30.922 Perimeter

Figure 4-41. VisionWare Blob Finder Tool for Example 2

Vision Image for Example 2

The vision image is provided here for reference. Please note the frame of reference of the part is shown in the center of the part with the orientation of the X-axis relative to the long side of the part.

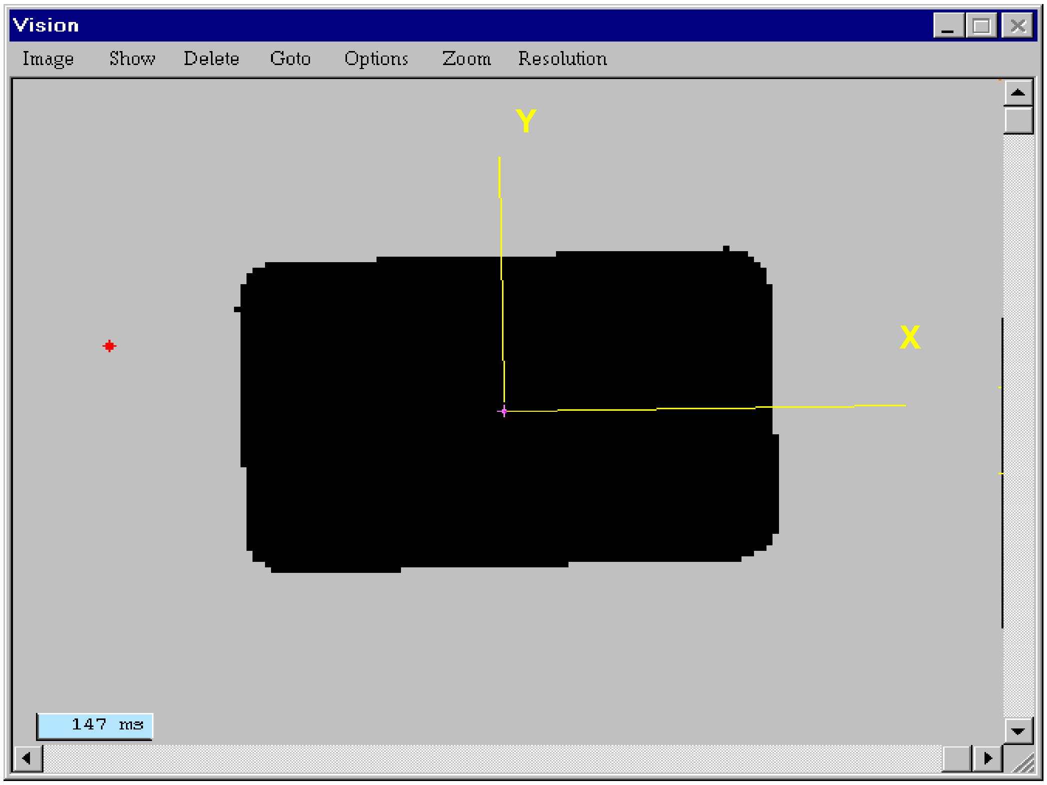


Figure 4-42. Vision Image of the Example 2

Servo Pump Example 3

This example is provided to show the acceleration and servo pump features within the dispense module. The following dispense pattern is shown for this example.

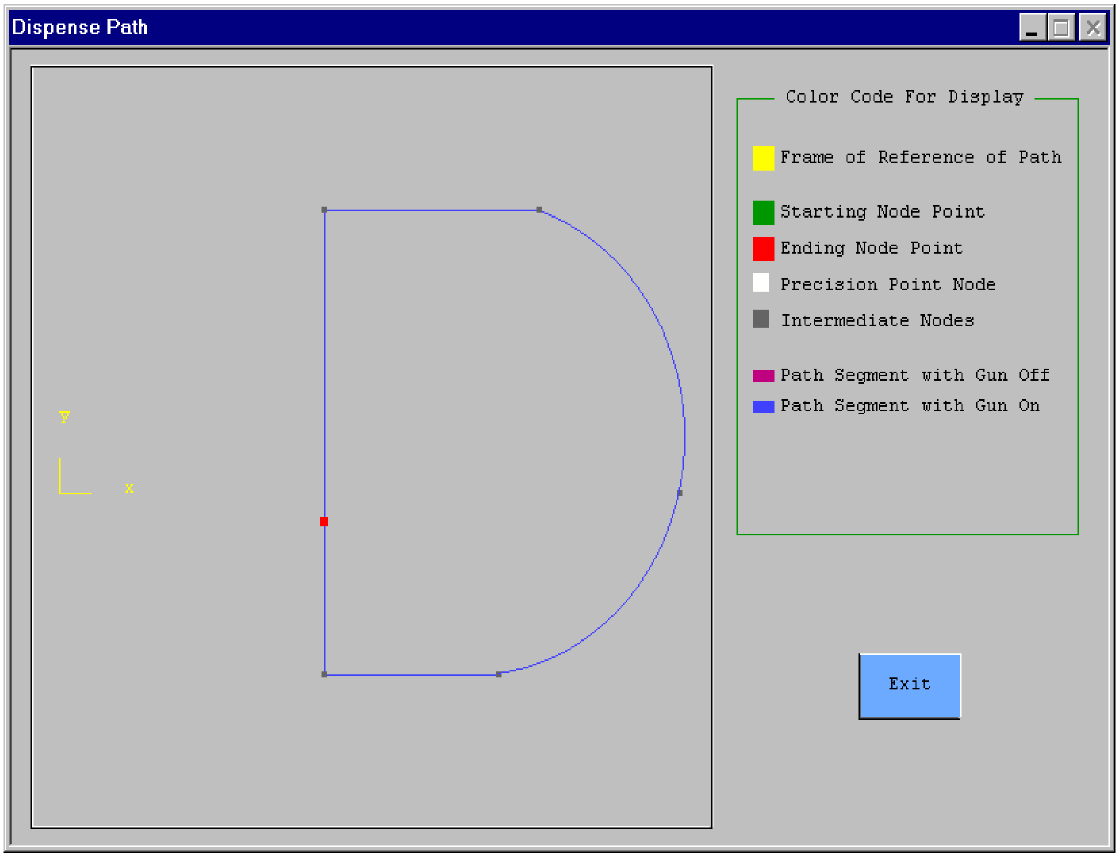
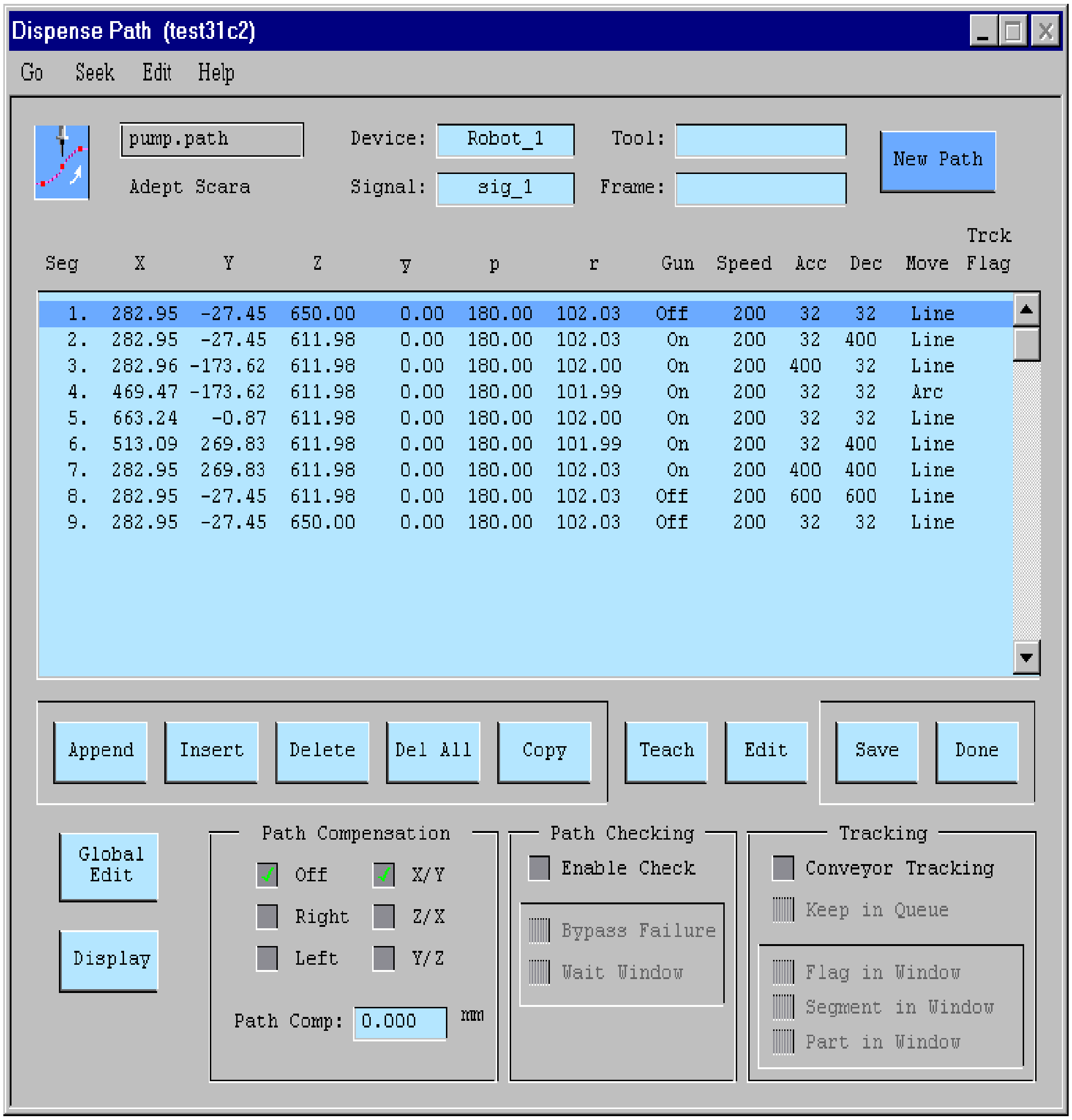


Figure 4-43. Process Path Display for Example 3

Process Path Records Example 3

This example will use the acceleration and deceleration features into and out of the corners in this pattern. This will reduce overshoot and makes the motion smooth through the pattern. The servo motor will change its speed based on the tool tip velocity that is needed during the acceleration and deceleration areas in the path. This path has the starting and stopping points located above the part. These points are required if gun compensation is to be used.

Review the listing of the locations in the dispense path menu provided in **Figure 4-44**. Notice the acceleration and deceleration values listed for points 2,3 & 6-9. These points are allowing accelerations and deceleration at corner points. Also note that the definition of the acceleration and deceleration are for the first point in the path segment for the motion.



pump.path

Adept Scara

Device: Robot_1

Signal: sig_1

Tool:

Frame:

Seg	X	Y	Z	y	p	r	Gun	Speed	Acc	Dec	Move	Trck Flag
1.	282.95	-27.45	650.00	0.00	180.00	102.03	Off	200	32	32	Line	
2.	282.95	-27.45	611.98	0.00	180.00	102.03	On	200	32	400	Line	
3.	282.96	-173.62	611.98	0.00	180.00	102.00	On	200	400	32	Line	
4.	469.47	-173.62	611.98	0.00	180.00	101.99	On	200	32	32	Arc	
5.	663.24	-0.87	611.98	0.00	180.00	102.00	On	200	32	32	Line	
6.	513.09	269.83	611.98	0.00	180.00	101.99	On	200	32	400	Line	
7.	282.95	269.83	611.98	0.00	180.00	102.03	On	200	400	400	Line	
8.	282.95	-27.45	611.98	0.00	180.00	102.03	Off	200	600	600	Line	
9.	282.95	-27.45	650.00	0.00	180.00	102.03	Off	200	32	32	Line	

Path Compensation

☑ Off ☑ X/Y

□ Right □ Z/X

□ Left □ Y/Z

Path Comp: 0.000 mm

Path Checking

□ Enable Check

▥ Bypass Failure

▥ Wait Window

Tracking

□ Conveyor Tracking

▥ Keep in Queue

▥ Flag in Window

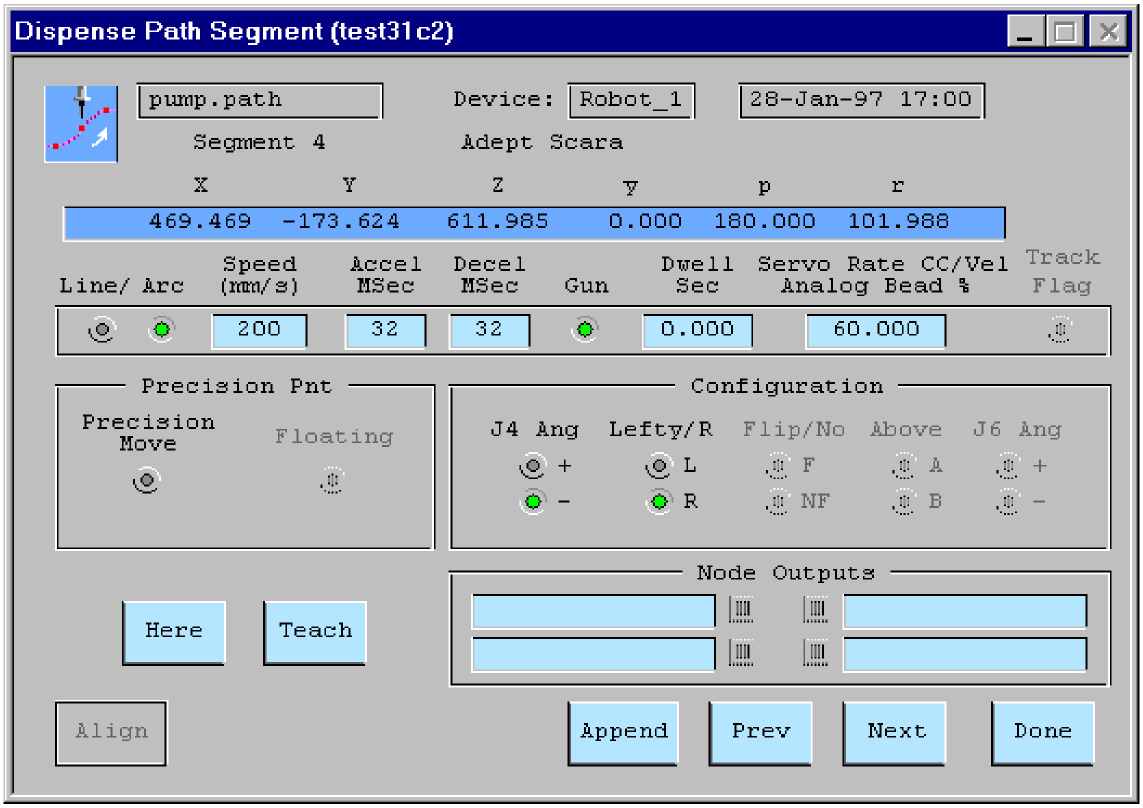
▥ Segment in Window

▥ Part in Window

Figure 4-44. Process Path Record for Example 3

Process Path Segment Example 3

The dispense segment menu in Figure 4-45 shows the Pump Rate / Bead % value as set to 60. This value is set very high and will normally be a small number. The number represents CC/(mm/sec). In many cases the values entered will be in decimal. Note that the analog control is based on a percentage of maximum, which is different from the servo pump interface.



Dispense Path Segment (test31c2)

pump.path Device: Robot_1 28-Jan-97 17:00

Segment 4 Adept Scara

X	Y	Z	y	p	r
469.469	-173.624	611.985	0.000	180.000	101.988

Line/ Arc		Speed (mm/s)	Accel MSec	Decel MSec	Gun	Dwell Sec	Servo Rate CC/Vel Analog Bead %	Track Flag
○	●	200	32	32	●	0.000	60.000	○

Precision Pnt: Precision Move ○ Floating ○

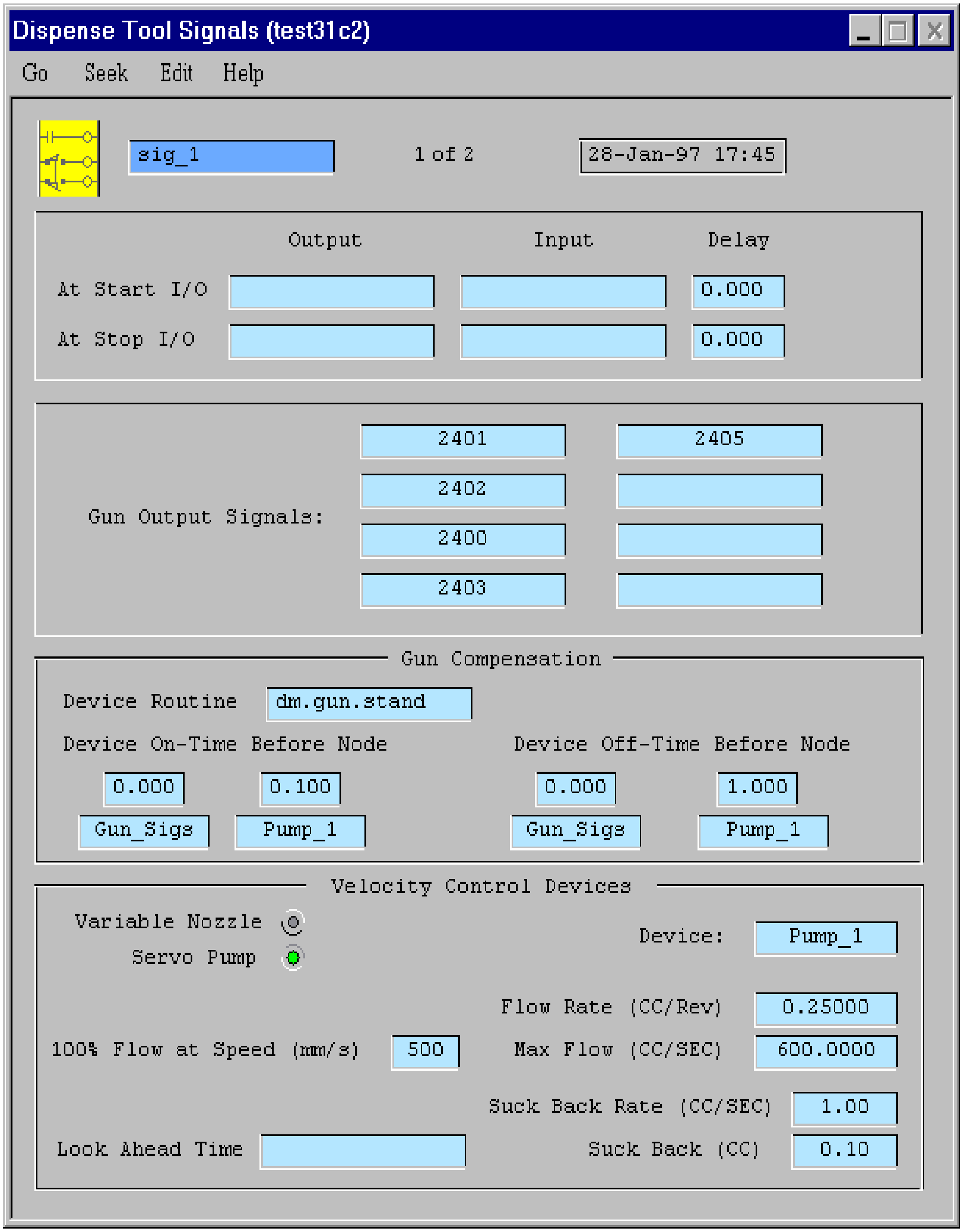
Configuration:

J4 Ang	Lefty/R	Flip/No	Above	J6 Ang
○ +	○ L	○ F	○ A	○ +
● -	● R	○ NF	○ B	○ -

Figure 4-45. Process Path Segment Servo Example 3

PathWare Signal Pump Example 3

The pathware signal menu shown in Figure 4-46 is provided to show an example servo pump setup as well as the gun compensation setup for a servo pump. The servo pump parameters must be defined before running any dispense path with a servo pump. Values entered as zero for *100% Flow Rate at Speed*, *Flow Rate*, and *Maximum Flow Rate* could cause program problems and could be nonzero. The *Suck Back* values can stay at zero if no suck back is desired. The Gun Compensation values for using a servo pump generally allow the value to be open before starting and stopping the value to reduce any back pressure. This example shows the value opening and closing at the node points and the pump starting .1 second before the gun on and stopping 1 second before the valve off condition.



sig_1 1 of 2 28-Jan-97 17:45

	Output	Input	Delay
At Start I/O			0.000
At Stop I/O			0.000

Gun Output Signals: 2401, 2402, 2400, 2403, 2405

Gun Compensation

Device Routine: dm.gun.stand

Device On-Time Before Node: 0.000 Gun_Sigs; 0.100 Pump_1

Device Off-Time Before Node: 0.000 Gun_Sigs; 1.000 Pump_1

Velocity Control Devices

Variable Nozzle ○
Servo Pump ●

Device: Pump_1
Flow Rate (CC/Rev) 0.25000
100% Flow at Speed (mm/s) 500
Max Flow (CC/SEC) 600.0000
Suck Back Rate (CC/SEC) 1.00
Look Ahead Time
Suck Back (CC) 0.10

Figure 4-46. PathWare Signal Pump Example 3

PathWare Customizing 5

Modifing I/O Operations

This section of the manual is provided for the user that requires custom I/O operations during the motion of the path. PathWare has been designed to allow custom operations to occur during path motion. Several unprotected routines are provided in the file RUN_DM.V2. These routines will be described in the following sections.

Dispense Signal Specified Routine

In the dispense signal menu page the user is allowed to specify a custom routine to control I/O during the path motion. The provided routine *DM.GUN.STAND* is designed to handle most dispensing operations such as valve control, analog velocity control, and servo pump velocity control. This routine is commented in the RUN_DM.V2 file provided on the dispense disk and is documented in **Chapter 6**.

This routine is called every path point based on the path point spacing in the dispense initialization database when using the *DISPENSE* sequence statement. As shipped to the customer the spacing is 0.032 seconds. This routine is also used by the *APPLY_DOT* sequence statement. All operations that use the dispense signal database, except for the *At Start* / *At Stop* outputs, use this routine for control. This allows a single routine which the user can customize for both the *Dispense* and *Apply_dot* statements.

The *GCOMP* argument in the routine call to *dm.gun.standard* controls on/off states of the devices used. This variable is an array variable that is tagged relative to every path point. The variable uses a bit structure to identify what operations are to occur. A positive or negative value indicates whether the operation is On or Off; negative indicates the operation is to turn off. The first bit indicates a change of state. The remaining bits are based on the selections made in the dispense signal database in the gun compensation selection fields.

The second bit, which is for gun compensation, has a decimal value of 2 controls the gun outputs for the IO_Grp1 in the keyword list. The third bit, with a decimal value of 4, controls the analog on/off state. (Currently, the dm.gun.stand routine does not set the analog voltage to zero if the off state is specified.) The fourth bit, with a decimal value of 8, controls the servo pump On/Off state. The section below describes how to add new devices to the gun compensation table.

Other calling arguments provide information to the routine, such as signal numbers, servo/analog scaling factors and tool tip velocities. For more information on this routine see **Chapter 6**.

Adding New Gun Compensation Devices

Additional gun compensation keywords may need to be added for special applications. Currently, the dispense signal menu X allows V two *On Compensation* and two *Off Compensation* fields. The runtime software provided with the system allows up to five On and Off fields without customization. This allows the customizer to add a lot more timing control relative to the On or Off node point. The dispense signal database allows eight signals which can be used with different keywords.

To modify the software to allow more gun compensation selections, the file *Dmmod.ov2* will have to be loaded. This file contains the commented version of all of the dispense initialization routines. The program *DM.INIT.KEY* contains the definitions for the gun compensation keywords. To add a new selection for the gun compensation fields a new line like the example below will have to be added.

```
CALL ai.key.add( $dm.arg[1, ], "New Sel", 16)
```

This line will add a new selection called *New Sel* to the pick list provided. The last argument in the calling field will be adding a new bit to the list. Currently, the first 4 bits are used; bit 5 would be next which has a decimal value of 16.

In order for the new selection to function, the routine *DM.GUN.STAND* will have to be modified to support this new feature. It is recommended to copy this routine to a new name and add your new features to this software.

Please note that your changes will have to be saved. If modifications have been made to the file *DMMOD.OV2,* it will have to be saved and then squeezed after completion to the file name *DMMOD.OVR*.

To modify the dispense signal menu to add more selection fields for gun compensation, the current fields can be copied. The database array field used will have to be increased based on the last selection field added. Please note that the first new selection field would have the array value of 2 for both *On* and *Off* compensation. More information about menu editing can be found in the ***MotionWare User's Guide***.

Start and Stop I/O Routine

A unprotected routine is provided for the I/O *At Start* and *Stop* control options in the dispense signal records. The routine *dm.strstp.out*, found in the file RUN_DM.V2, controls the use of the I/O. This routine can be modified for different use of the outputs.

Currently, the routine turns on the output specified before begining path motion. If an input is specified, the routine will wait for the input until the delay timer is exceeded. If this should occur, an error message will be displayed. If no input is specified, the routine will turn on the output and wait the amount of time specified in the delay field.

IOSIG Array

The IOSIG array contains all of the information from the dispense signal database. This array is used throughout all the dispense routines for both the Dispense and Apply_dot statements. It is stored in the *rn.dispense* routine as a Local two-dimensional array where the first array index is a pointer value based on task, sequence step number, and sequence calling number. The second array index uses the array structure documented below. Global variables are provided for ease of use and can be found in the files *DMMOD.OVR* and *DMMOD.OV2.* The routine dm.init.gbl initializes these variables. The table below lists and describes the array data.

Table 5-1. IOSIG Global Array Structure

Array Variable	IOSIG Array Description
ds.start.out	Output signal number for the *At Start* field in the dispense signal database.
ds.stop.out	Output signal number for the *At Stop* field in the dispense signal database.
ds.start.in	Input signal number for the *At Start* field in the dispense signal database
ds.stop.in	Input signal number for the *At Stop* field in the dispense signal database
ds.start.tmr	Timer value for the delay / max time for input signal field in the dispense signal database.
ds.stop.tmr	Timer value for the delay / max time for input signal field in the dispense signal database.
ds.first.sig	First valve signal number of an array of 8 that is standard.
ds.num.sig	Number of valve signal fields used in the dispense signal database.
ds.ana.ena	Variable defines if analog or servo velocity control features are used. Bit 1 is used for analog and bit 2 is used for servo control.
ds.ana.channel	Defines the analog channel used for analog velocity control.

Table 5-1. IOSIG Global Array Structure (Continued)

Array Variable	IOSIG Array Description
ds.ana.mspd	Defines the speed at which max analog voltage is applied. Also used for scaling with the servo control.
ds.look.ahead	Look Ahead time field in the dispense signal database.
ds.pump.flow	Defines the CC/REV constant used with the servo control feature in the dispense module.
ds.max.flow	Defines the maximum flow rate of the servo pump.
ds.suck.vol	Defines the suck back volume specified in the dispense signal database.
ds.suck.rate	Defines the suck back flow rate used in the dispense signal database.
ds.device	Defines the robot number used for the servo pump.

The routine *dm.get.signals,* which is in the file *RUN_DM.V2,* retrieves all of the data that is stored in the IOSIG array. If additional data is needed, this routine can be changed as well as the *iosig* array. This program is used by both the *Dispense* and *Apply_Dot* statements used in the dispense module.

DM.OUT Array Used in CU.REACTE Routine

The *dm.out* variable is a global variable that contains the dispense signal valve outputs that are specified. When a fatal error occurs, the routine *cu.reacte* is called. This routine will turn off the valves when this condition occurs. The *dm.out* array is a two-dimensional array. The first element is the task number that the dispense routines are running in. The array requires the task number so multiple robots can be run with the dispense system. The second element contains all eight signal numbers.

The routine *CU.REACTE* is a standard AIM routine which the dispense module overwrites so the guns can be turned off if an error occurs. This routine is contained in the file *CUSTOM.V2* and *CUSTOM.SQU.*

Servo Pump Motor Server

The dispense module software, as shipped to the customer, is configured to handle one servo pump device . The software configures a dedicated task or server to handle the servo motion. There are basically 3 routines that make up the pump server; *rn.pump.server*, *dm.pump.spn*, and *dm.pump.drv*. These routines all reside in the files RUN_DM.V2 and RUN_DM.SQU. These routines are unprotected and are documented in **Chapter 6**.

Adding a Second Motor Server

Should a second servo pump be required, it is recommended to establish a second server to accomplish this task. To do this you will have to modify several routines and add items to the intialization database. The following paragraphs describe the changes must occur.

The first step in adding a second motor server is to copy the original motor server and make some minor modifications to differentiate it from the other server. The server routines are located in the files RUN_DM.v2 and RUN_DM.SQU. Changes will be shown in Bold text.

```
Program rn.pump2.server


dv.pump2.id[TASK()] = 0                 ;Add 2nd servo pump
server

Several lines between statements

; Dispense Pump server   Connects if pump is present

   CALL rn.cli.connect(ti.pump2, 0, TRUE,
dv.pump2.id[TASK()], error)
   IF error == rn.opr.abort GOTO 100       ;Exit if aborted
```

The second pump server program is very similar to the orginal program. It uses the same motion routines as the other server. The motion routines are structured using auto variables and global variables are defined on a task basis.

The next change will be made to the main initialization routine *ai.module.init* which is located in the LDM.V2 file. This routine needs to be able to initialize the robot devices as well as starting the server routines The changes will initialize the routines. Please note that some of the variables that will be used need to be added to other locations in the software.

```
Program ai.module.init

;Several lines of instructions before initializing device variables


ti.pump1 = 0                            ;Assume no servo pump
ti.pump2 = 0                             ; Adding 2 pump device

;Several more lines of instructions until the next couple of changes

IF (SELECT(ROBOT, -1) >= 2) AND ((dm.servo.pump)) THEN  ;If servo pump
    IF SWITCH(ROBOT[2]) OR SWITCH(DRY.RUN) THEN
      CALL ai.dev.define(dm.pmp.device, $dm.pump.dev, 0, dm.pmp.device,
halt)
      IF halt GOTO 100
      CALL ai.task.define(0, dm.server.task, $dm.pump.dev, $dm.pump.dev,
FALSE, ti.pump1, halt) ;L510
      IF halt GOTO 100
    END
END                                                              ; A02-
IF (SELECT(ROBOT, -1) >= 2) AND ((dm.servo.pump2)) THEN  ;If servo pump
    IF SWITCH(ROBOT[2]) OR SWITCH(DRY.RUN) THEN
      CALL ai.dev.define(dm.pmp2.device, $dm.pump2.dev, 0,
dm.pmp2.device, halt)
      IF halt GOTO 100
      CALL ai.task.define(0, dm.server.task2, $dm.pump2.dev,
$dm.pump2.dev, FALSE, ti.pump2, halt) ;L510
      IF halt GOTO 100
    END
END                                                              ; A02-

;New variables added -dm.servo.pump2, dm.pmp2.device, $dm.pump2.dev
;which will be defined in the dispense module initialization database

; Several more lines of instruction before starting tasks


CALL ai.task.prior(19, 21, 19, 21, 19, 21, 19, 21, 19, 9, 11, 11, 9, 0,
0, 0, p[])
CALL ai.task.start(ti.pump1, , "rn.pump.server", p[], halt, "")
IF halt GOTO 100

CALL ai.task.prior(21, 19, 21, 19, 21, 19, 21, 19, 21, 9, 11, 11, 9, 0,
0, 0, p[])
CALL ai.task.start(ti.pump2, , "rn.pump2.server", p[], halt, "")
IF halt GOTO 100
```

Several program lines have been added to initialize and start the second pump server. Almost everything can be copied and changed with new variable names.

The next changes will be to the dispense initialization database. You will be adding the new variable names that were added to the ai.module.init routine. These changes can be made by selecting *Edit Init Databases* under the *Special* pulldown. Select the database *dmini.db*. The following records will have to be copied and modified with new variable names and new default values. These records are *pump device name, pump device number, pump server task selection*. They can be found by using the SEEK pulldown and selecting *index*. The record names will have to be changed after copying to indicate the second pump variables. The following variable names will have to be changed based on the record selected.

```
pump 2 device number  dm.pmp2.device
pump 2 device name    $dm.pump2.dev
pump 2 server task selectiondm.server.task2
```

Next, changes must be made to the dispense runtime code to support the second pump server. The first changes will have to be made to the routine *rn.dispense* located in the files RUN_DM.V2 and RUN_DM.SQU. The changes below will allow the routine to determine which pump server to comumicate with. The change will look at the device that was selected in the dispense signal database and compare it to the defined devices for the servers.

PROGRAM rn.dispense

Skip down to near the end of the routine and the following lines will be found.
The text in bold is what changes will have to be made.

```
; If pump is selected, set drive mode of pump and prepare
pump for start

67  IF iosig[ctsk,ds.ana.ena] == 2 THEN ;Servo Pump control
selected

   IF iosig[ ctsk , ds.device ] == dm.pmp.device THEN
     $server = "rn.pump.server"
     pump.id = dv.pump.id[TASK()]
   ELSE
     $server = "rn.pump2.server"
     pump.id = dv.pump2.id[TASK()]
   END

   IF (dm.dry.gun[task]) AND (STATUS( $server ) <> -1) THEN
; Check server and dry run modes

     $tmp = " "

     CALL rn.cli.send( pump.id , dm.spn.drv, TASK(), $tmp, 1,
msg.id, $reply, error)
   IF error <> 0 GOTO 90
   ELSE
     IF STATUS( $server ) == -1 THEN
       st = ec.dm.no.server
       CALL rn.error(error, er.all, st, "", dp.db, 0, -1, 0)
   END
END
```

The following changes allow the user to have a second pump selection in the dispense signal menus for the gun compensation. This change is not really required if the pump_1 selection for gun compensation is not confusing. Basically, the runtime changes above allow the proper server to be connected to the software. The changes below allow a second pump selection for the gun compensation pick list. If the changes below are made, the final listing of changes in this section will also be required. If you choose not to add a second pump selection, no further changes are required.

The changes below will have to be made to the *dm.init.key* program located in the files DMMOD.OVR and DMMOD.OV2.

PROGRAM dm.init.key

```
; Keyword for Dispense gun compensation selection of tag
bits

       CALL ai.key.new($dm.arg[1,])
   CALL ai.key.add($dm.arg[1,], "None", 0)
   CALL ai.key.add($dm.arg[1,], "Gun_Sigs", 2)
   CALL ai.key.add($dm.arg[1,], "Nozzle", 4)
   CALL ai.key.add($dm.arg[1,], $dm.pump.dev, 8)

   CALL ai.key.add($dm.arg[ 1, ], $dm.pump2.dev, 16)
```

Finally, the changes that are shown below will allow the runtime code to support the new gun compensation device. The changes are made to the routine *dm.gun.stand* which is located in the files RUN_DM.V2 and RUN_DM.SQU.

PROGRAM dm.gun.stand

```
IF (ABS(gcomp) BAND ^B1000) OR (ABS(gcomp) BAND ^B10000 ) THEN  ;
Turn pump on or off
   IF gcomp > 0 THEN ; Pump on
     dm.pmp.clnt[task,dm.suck.bck] = ((iosigs[ds.suck.vol]
/iosigs[ds.pump.flow])) *360
     dm.pmp.clnt[task,dm.suck.spd] = (iosigs[ds.suck.rate]
/iosigs[ds.pump.flow])*360
   IF dm.pmp.clnt[task,dm.suck.spd] > max.flow THEN
     dm.pmp.clnt[task,dm.stop] = TRUE
     SIGNAL -iosigs[3], -iosigs[4], -iosigs[5], -iosigs[6],
-iosigs[7], -iosigs[8], -iosigs[9], -iosigs[10] ;A01
     CALL rn.error(error, er.abo, ec.dm.max.flow, "", 0, 0, 0, 0)
     GOTO 90
   END
     dm.pmp.clnt[task,dm.start] = TRUE
   ELSE
     dm.pmp.clnt[task,dm.stop] = TRUE
   END
END
```

Purge Routine

The dispense module allows the user to create a custom purge cycle if one is required for the application. The routine *dm.purge.stand* is provided as an example of the calling arguments required for a custom routine. When adding a custom routine, the dispense initialization database allows the user to specify a new routine name in the place of *dm.purge.stand*. This routine must be loaded before any purge button presses are allowed from the manual Dispense I/O control purge button can be pressed. The routine *dm.purge.stand* is fully documented in **Chapter 6** of this manual.

Adding New Kinematic Device Modules

The dispense module comes with most of the AdeptMotion Kinematic devices already defined for the system. If Unknown Device is displayed in the upper left-hand corner of the dispense menus when a robot device is selected to be used with the dispense path, the device has not been configured in the dispense software. It is important to have the correct configuration to allow proper precision point selections to be allowed in the path segment menus. The new kinematic device can be easily added to the system by customizing the routines *dm.init.key* and *dm.init.glb,* which are located in the files DMMOD.OV2 and DMMOD.OVR.

The routine dm.init.key initializes all of the keyword lists that are used in the dispense module. In order for the menu displays to show the new kinematic module, it must be entered in this routine. The following lines of software reside in dm.init.key with a example line added for the new kinematic module.

```
CALL ai.key.new($dm.arg[2,])
CALL ai.key.add($dm.arg[2,], "Unknown Device", 0)
CALL ai.key.add($dm.arg[2,], "Adept Scara", 1)
CALL ai.key.add($dm.arg[2,], "Gantry", 3)
CALL ai.key.add($dm.arg[2,], "Gen Scara", 6)
CALL ai.key.add($dm.arg[2,], "X,Y,Z,T", 8)
CALL ai.key.add($dm.arg[2,], "Scara", 11)
CALL ai.key.add($dm.arg[2,], "New Device", 55)
```

The new name NEW DEVICE will be added to the display list for the dispense menus. The number 55 in the call is the Kinematic Device number used by V^+. It is displayed in the V^+ monitor when the ID monitor instruction is entered. This number must be correct for the proper name to be displayed.

The next step will configure the selection bits that are used for the precision point configuration definition. Basically, a global array variable with the Kinematic device number used for the array index defines what selections are available for a particular device. Basically, the first bit sets whether or not Joint 4 rotation above 360 degrees is possible. The second bit selects whether or not joint 6 rotation can be above 360 degrees. The third bit sets whether or not righty or lefty is possible. The fourth bit sets whether or not above or below is possible. The fifth bit sets whether or not flip or noflip is possible. For more information on the terms above refer to the ***V^+ Language Reference Guide***. The program where the changes are to be made is *dm.init.gl*b, which is found in the files DMMOD.OV2 and DMMOD.OVR.

PROGRAM dm.init.glb

```
; Module selection bit settings

; Bit combinations for K module selections for Precision
Point Config
```

```
;        MSB                                   LSB
;       Flip/No  Above/B   Righty/L   Jnt 6   Jnt 4


dm.mod[0] = ^B11111     ; Unknown Device
dm.mod[1] = ^B101       ; Adept Scara
dm.mod[3] = ^B1011      ; Gantry
dm.mod[8] = ^B1         ; X,Y.Z,Y
dm.mod[55] = ^B11111;New device wil all selections
```

Procedural Path Spacing

The procedural path spacing for the dispense module is set to 0.032 second spacing. This spacing should allow the dispense module to run very efficiently. However, applications that require very accurate paths or very demanding control of outputs may require tighter point spacing. The procedural path spacing can be lowered to 0.016 second spacing without any configuration changes to the Adept Controller. Lower spacing can be achieved down to 0.002 second spacing if the Enhanced Trajectory Generator license is purchased through Adept.

By decreasing the spacing, arcs and rounding around corners will be improved. Also servo and analog velocity control will be improved. The biggest limitation are program memory and processor speed. Advanced applications will need more processing power.

To change the procedural path spacing, select the dispense initialization database (dmini) from the SETUP pulldown in AIM. Select Path Point Spacing when the index list is displayed.

D.Path and Path.Mve Local Arrays

The program rn.dispense contains two local arrays of interest, d.path[,] and path.mve[,,]. These arrays contain all of the information used for the robot motion that occurs during a path. The array *d.path* contains all of the transformations for all of the paths that have been processed. The path.mve array contains all of the parameter information for each node point taught.

The first index of both variables uses a pointer which is described in **Path Pointer**. This pointer allows the dispense module to retain the path information even if multiple devices and sequence call statements are used in the process.

The array *d.path* is structured with the begining point starting at 1 and continuing through all of the calculated points along the path.

The *path.mve* array is a three-dimensional array where the second index element references the node or segment points along the path. The third index is based on the parameter defined. The parameter indexes are defined as global variables for every desired parameter. They are all listed in **Global Variables** in this chapter.

Path Pointer

The dispense module uses a pointer method to keep track of the processed dispense paths to minimize the frequency of processing the paths. This method uses two variables to accomplish the path pointer. They are *dm.pnt*[,,] and *ctsk* and are used only in the routine *rn.dispense*. The dm.pnt array is defined as below:

```
dm.pnt [ task , rn.ctl[ task ,rn.ctl [rn.ctl.seqnum ] ,rn.ctl
[rn.ctl.c.step ] ] = counter
```

where

```
task            current task the dispense statement is running in
rn.ctl [ rn.ctl.seqnum]called sequence number (CALL statement)
rn.ctl [ rn.ctl.c.step ]current sequence statement number running
```

With this array structure, there could be memory contraints based on large sequences and many sequence call statements. Additionally, it is a good idea to try to run the dispense tasks on lower number tasks to reduce memory useage. This method was required to allow multiple robot and sequence call support.

The number retained in the dm.pnt is the counter value to which *ctsk* is set to. Several dispense module variables use this counter as the first array index field within the routine *rn.dispense*.

Global Variables

The following sections list all of the global variables used by the AIM PathWare software. They are presented in the groups in which they are logically used to allow easy finding of specific variables.

The following variables are for the dispense record field definitions. These are also documented in **Chapter 7**.

dp.index	3	Field variable for number of path records
dp.loc	4	Dispense location transformation field
dp.loc.data	5	Dispense location data field
dp.loc.mot	6	Dispense location motion bits
dp.loc.speed	7	Dispense location speed --mm/sec
dp.loc.accel	8	Dispense location acceleration
dp.loc.decel	9	Dispense location deceleration
dp.loc.config	10	Dispense position arm configuration
dp.loc.type	11	Dispense location type
dp.singularity	12	Dispense singularity checking
dp.gun.state	13	Dispense gun on or off
dp.dwell.time	14	Dispense dwell time at corner position
dp.bead.size	15	Dispense flow gun nozzle opening
dp.tracking	16	Dispense setting of the optional tracking
dp.dev.mod	17	Dispense device module model number
dp.tool.name	18	Dispense tool name string
dp.tool	19	Dispense tool record number
dp.frame.name	20	Dispense frame name
dp.frame	21	Dispense frame record number
dp.signal.name	22	Dispense signal name
dp.signal	23	Dispense signal record number
dp.max.index	24	Dispense number of path points
dp.jnt.config	25	Determines if a precision move is to occur
dp.plane	26	Path compensation plane
dp.comp	27	Dispense path compensation amount in mm
dp.direct	28	Dispense path compensation direction left or right
dp.out.name	29	Dispense path node output names
dp.out.number	30	Dispense path node output numbers

The following variable definitions are for the dispense signal record fields.

ds.max.speed	3	Signal database flow gun open 100% speed
ds.flow.gun	4	Signal database select either nozzle or pump control
ds.analog	5	Signal database analog channel or joint number

ds.out.name	6	Signal database name for output signals
ds.out.number	7	Signal database numbers for outputs
ds.in.name	8	Signal database name for variable db
ds.in.number	9	Signal database number for input signals
ds.delay	10	Signal field number for delay data
ds.routine	11	Signal db field number for user routine
ds.sig.on	12	Signal db field number for gun comp on states
ds.sig.type	13	Signal db with binary number for device selection
ds.sig.off	14	Signal db field number for gun comp off states
ds.type.off	15	Signal db field number that contains device bits
ds.pump.parms	16	Signal db field number for pump parameters

The following definitions are for the path parameter array indexes "p.mve[]" .

dm.mve.straight	0	array index for line or arc, values are 0 or 1
dm.mve.speed	1	array index for speed, values in p.mve are in mm/sec
dm.mve.gun	2	array index for gun state
dm.mve.dwell	3	array index for dwell time
dm.mve.bead	4	array index for Servo Rate / Bead %
dm.mve.sing	5	array index for enabling singularity check
dm.mve.nmoves	6	array index for the number of intermediate points
dm.mve.duration	7	array index for duration time between points in segment
dm.mve.accel	8	array index for acceleration values in segment
dm.mve.decel	9	array index for deceleration
dm.mve.arm.c	10	array index to determine arm config for prec move
dm.mve.jnt	11	array index to enable a precision move
dm.mve.track	12	array index that contains the tracking option bits
dm.mve.bnum	13	array index that contains the belt number for tracking
dm.mve.trckflg	14	array index for segment window tracking flags
dm.mve.mfrec	15	array index for record number of conveyor db
dm.mve.output	16	array index for the four node output

		signals
dm.mve.time	21	array index for path checking time estimate
dm.mve.convel	22	array index for specifing procedural path motions
dm.mve.app	23	array index for specifying DOT parameters -uses 10 indexes
dm.mve.last	33	variable contains number of indexes for the array

The following definitions are for the iosig array indexes.

ds.start.out	1	array index for At Start output signal
ds.stop.out	2	array index for At Stop output signal
ds.start.in	16	array index for At Start input signal
ds.stop.in	17	array index for At Stop input signal
ds.start.tmr	14	array index for At Start timer
ds.stop.tmr	15	array index for At Stop timer
ds.ana.ena	11	array index for enabling analog and servo options
ds.ana.channel	12	array index for defining the analog output channel
ds.ana.mspd	13	array index for speed at which max analog voltage is applied
ds.look.ahead	24	array index for look ahead value
ds.device	23	array index for pump device number
ds.pump.flow	18	array value for pump flow ratio
ds.max.flow	19	array value for max flow rate
ds.suck.vol	20	array value for suck back volume
ds.suck.rate	21	array value for suck back rate
ds.num.signal	34	array value for number of values in iosig array
ds.pmp.home	25	array value for home position in io.sigs
ds.pmp.max.ext	26	array value for pump max position before retract
ds.sigdelay	27	array value for delay time if signals not used
ds.pmp.retspd	28	array value for retract speed in io.sigs
ds.rsig1	29	array value for retract output signal 1
ds.rsig2	30	array value for #2 retract ouptut
ds.risig1	31	array value for retract input signal 1
ds.risig2	32	array value for #2 retract input
ds.timeout	33	array value for timeout period
ds.deflt	34	array value for defaulting to retract signal setting

Pump global variables.

dm.drv.drv	1	selection variable for server communication
dm.spn.drv	2	selects spin-driver pump operation
dm.retract	3	selects retraction of pump
pmp1.volume	0	global variable that contains last volume of pump 1
pmp2.volume	0	global variable for pump 2 volume
dm.cycle.tm	0	global variable of robot cycle time
dm.rob.busy	0	global variable that indicates if robot is running a path

Array index for dm.pmp.clnt[task,] variable that controls server pump.

dm.start	1	Start pump operation
dm.stop	2	Stop pump
dm.revs	3	Rotate pump X degrees
dm.speed	4	Rotate pump X speed
dm.suck.bck	5	Suck degrees
dm.suck.spd	6	Suck back speed
dm.end	7	End pump operation wait for next request
dm.loop	21	Loop the server to wait for a function change
dm.switch	22	Switch to different function based on value
dm.drv.sel		Set to true if server is running the drive function
dm.spn.sel		Set to true if server is running the spin function

The following variables define what precision point selections are available based on the Kinematic module selection.

Bit combinations for K module selections for Precision Point Config:

MSB LSB
Flip/No Above/B Righty/L Jnt 6 Jnt 4

dm.mod[0]	^B11111	Unknown Device
dm.mod[1]	^B101	Adept Scara
dm.mod[3]	^B1011	Gantry
dm.mod[8]	^B1	X,Y.Z,Y
dm.mod[6]	^B101	Gen Scara
dm.mod[11]	^B101	Scara
dm.mod[14]	^B11011	Puma
dm.mod[18]	^B1	Enh X,Y,Z,T

```
dm.mod[19]   ^B1011   Enh Gantry
dm.mod[22]   ^B1      Cart
dm.mod[32]   ^B1      LMMV X,Y,Z,T
dm.mod[33]   ^B1      Enh LMMV
```

The following global variables are used for various functions within the dispense module.

```
dm.sched.start 0  Flag to indicate that scheduler was just
                  started
dm.dry.gun     -1 Dispense Dry Run status at startup
dm.timer       0  Timer used for io checking  (master timer)
dm.cv.pause    8  rn.ctl pause status
dm.cv.edit     7  rn.ctl edit status for walk-thru-training
dm.cv.speed    9  rn.ctl task robot speed
dm.path.state  2  Global argument for path state
dm.hndshk      1  Global argument for handshake
dm.look.ahead  0  Defines a time segment ahead for velocity
dm.sp.ptr[tsk]    This variable is pointer used to keep
                  track of variable speed changes that are
                  set by the statement chg.disp.speed
dm.sp.pc[tsk,p]   This variable contains a percentage speed
                  change used by the statement
                  chg.disp.speed
dm.sp.rec[tsk,p]  This variable contains the dispense path
                  record number to use to determine which
                  path gets the speed change
dm.dw.ptr[tsk]    Pointer used to keep track of variable
                  dwell changes from the runtime statement
                  chg_disp_dwell
dm.dw..pc[tsk,p]  This variable contains a percentage dwell
                  change used by the statement
                  chg.disp.dwell
dm.dw.rec[tsk,p]  This variable contains the dispense path
                  record number to use to determine which
                  path gets the dwell change
dm.out[ , ]       This variable contains the dispense gun
                  signal numbers used in the routine
                  cu.reacte to turn off the gun signals
dm.sched.start[]  This variable is set to true every time
                  the scheduler is restarted.  This forces
                  the dispense path to be recalculated
```

dm.last.cnt[]	This variable is used to work with the local variable array for the calculated path points.
$dm.menu.name	Dispense menu file name
$ai.mu.dm.str[]	Defines teach pendant menus used in the dispense module

The following variables are defined in the dispense initialization database and used in the runtime software.

dm.start.spd	This variable is the speed used to approach the first path point and if belt tracking, the speed to the wait line.
dm.start.acl	This is the acceleration to move to the first path point
dm.start.dcl	This is the deceleration to move to the first path point
dm.def.pacl	Precision Point acceleration defaults
dm.def.pdcl	Precision Point Deceleration defaults
dm.def.cacl	Linear Motion Acceleration defaults
dm.def.cdcl	Linear Motion Deceleration defaults
dm.def.spd	Default Speed for dispense path
dm.break.tolerance	Prohibits forward processing when tracking is used until the position of the robot is within the specified tolerance
dm.servo.pump	This variable will enable a pump servo task
dm.server.task	This is the task that the pump server will run in.
$dm.pump.dev	Allows selection of the pump device name
dm.pmp.device	Robot device number for dispense servo pump.
dm.purge.tm	Time in seconds for purge cycle to execute
dm.purge.rt	Rate for purge material to be dispensed
$dm.purge	Variable that contains routine to execute cycle.
dm.purge.nz	Defines percentage open during manual purge cycle.
dm.sing.spc	Spacing between joint path singularity check
dm.sing.chck[1-6]	Comparison values to check for singularity
dm.dot.data[]	Global array that contains dot default data.

The following variables are used with the reciproating pump (dm.pmp.clnt[]).

dm.ret.pmp	8	Array value for specifying reciprocating pump
dm.pmp.home	9	Array value for the pump home position
dm.pmp.maxext	10	Array value for specifying maximum pump extension
dm.pmp.retspd	11	Array value for specifying the speed for retracting
dm.pmp.delay	12	Array value for pump delays
dm.pmp.sig1	13	Array value for Output signal #1
dm.pmp.sig2	14	Array value for Output signal #2
dm.pmp.isig1	15	Array value for Input signal #1
dm.pmp.isig2	16	Array value for Input signal #2
dm.pmp.tmout	17	Array value for pump signal time out
dm.pmp.deflt		
dm.pmp.scale	20	Array value for Pump Scale factor
pmp1.volume		Variable contains current volume from pump #1
pmp2.volume		Variable contains current volume for pump #2

The following variables are used by the path editor menu.

```
vdp.speed
vdp.fix.hgt
vdp.to.cam
vdp.x.center
vdp.y.center
vdp.cam.mode
vdp.frame.,set
vdp.oldtl
vdp.tool
vdp.force.tool
vdp.frame.sz
```

The following variables define the error codes used in the dispense module.

ec.dm.calc.path	4575	Calculating dispensing path
ec.dm.calc.comp	4576	Calculating compensation path
ec.dm.set.out	4577	Setting outputs at start or end
ec.dm.no.path.s	-6575	No path Segment defined
ec.dm.no.points	-6576	Path Positions are not defined
ec.dm.no.circle	-6577	Circular Interpolation Failure
ec.dm.no.signal	-6578	Signal Database Does not Exist
ec.dm.no.licens	-6579	Applications License does not exist
ec.dm.no.input	-6580	Input signal did not change state
ec.dm.no.speed	-6581	Speed set to low for division by 0
ec.dm.acdc.ovr	-6582	Accel/Decel overlap on circular path

```
ec.dm.gun.comp -6583  Gun compensation cannot be
                      completed on first path point
ec.dm.gun.index -6584 Compensation cannot be performed
                      because not enough path points are
                      available for time specified
ec.dm.zero.flow -6585 Servo pump rate is set to zero
ec.dm.max.flow -6586  Flow rate exceeds pump rate
ec.dm.sing.chk -6587  Singularity check failed
ec.dm.no.server -6588 Pump Server is not executing
ec.dm.cir.gun  -6589  Gun changed state at mid circle point
ec.dm.cir.def  -6590  Circle defition failure -First and
                      last pt same
```

Routine Dictionary 6

Documented Routines

This chapter describes PathWare routines that a customizer can use in custom AIM software.

The routines listed in Table 6-1 are documented in this chapter and can be used by AIM system customizers. Programs marked with a **P** are protected programs which cannot be edited or modified. Edits to unprotected files should be made to the commented **.V2** versions of the files. The files should then be squeezed to their **.SQU** versions for loading with AIM.

Table 6-1. Documented Routines

Program Name		Autoloaded File	Commented File	Purpose
ch.reset.io		RUN_CH.SQU	RUN_CH.V2	Sample tool change I/O program
dm.mu.teach		MENU_DM.SQU	MENU_DM.V 2	Teach Pendant path teaching routine
rn.apply.dot		RUN_DM.SQU	RUN_DM.V2	APPLY.DOT runtime primitive
rn.dispense		RUN_DM.SQU	RUN_DM.V2	DISPENSE runtime primitive
rn.dm.cutter.cp	P	LIB_DM.SQU	None	Path Comp. interpolation routine
rn.dm.calc.path	P	LIB_DM.SQU	None	Path interpolation routine
rn.dm.move	P	LIB_DM.SQU	None	Low-level path motion routine
rn.dm.move.pat h	P	LIB_DM.SQU	None	Path dispensing routine
rn.pmove	P	LIB_DM.SQU	None	Precision Pnt. motion routine
dm.gun.stand		RUN_DM.SQU	RUN_DM.V2	Controls Dispense Signals

Table 6-1. Documented Routines (Continued)

dm.purge.stand		RUN_DM.SQU	RUN_DM.V2	Controls manual purge cycle
rn.pump.server		RUN_DM.SQU	RUN_DM.V2	Runs pump task server
dm.pump.spn		RUN_DM.SQU	RUN_DM.V2	Controls Servo spin operation
dm.pump.drv		RUN_DM.SQU	RUN_DM.V2	Controls servo dot operation
dm.strstp.out		RUN_DM.SQU	RUN_DM.V2	Control start / stop I/O operations
dm.get.signals		RUN_DM.SQU	RUN_DM.v2	Retrieves dispense signal data

PathWare Routines

Each routine documented in this section is presented on a separate page, in alphabetical order. The dictionary page for each routine contains the sections shown in the sample on **page 166**:

> **NOTE:** The variable names used for the routine parameters are for explanation purposes only. Your application program can use any variable names you want when calling the routine.

Calling Sequence

The format of the V CALL instruction for the routine is shown.

Function

This is a brief statement of the routine's function.

Usage Considerations

This section points out any special considerations associated with the routine.

Parameters

Each of the input parameters in the calling sequence is described in detail. For parameters that have a restricted range of values, the allowable range is specified.

Each of the output parameters in the calling sequence is described in detail.

Global Variables

Global variables accessed by the routine are described.

Details

A complete description of the routine and its use is given.

Related Keyword

Other AIM routines that are related to the routine's function are listed.

Calling Sequence

```
    CH.SET.IO (error)
```

Function

Sample I/O reset routine specified in the Quick Change database for use with the CHANGE.HAND statement. The program sets output signals to default values before replacing or after acquiring an end-effector.

Parameters

error	Real variable that receives the standard AIM operator response code

Details

Used in conjunction with the CHANGE.HAND statement, this routine is called after acquiring a tool (if specified as a pick-up routine) or before replacing a tool (if defined as a put-down routine) to set output signals to their default values. This can be used, for example, to prevent air lines from blowing when a tool is detached or to prevent dispensing guns from turning on after acquiring a tool.

Calling Sequence

```
DM.MU.TEACH (arg, db.p, $cmd)
```

Function

Teach pendant routine that is called from the dispensing path database menu. This routine allows basically the same functionality from the teach pendant as from the menu screen.

Usage Considerations

This program conforms to the standard AIM format for a menu spawn routine. More information can be found regarding the menu spawn routines in Chapter 3 of the ***AIM Customizer's Reference Guide***.

Parameters

arg Real value, variable, or expression that specifies the value of the "routine argument" defined in the button menu record. This value is not used in the routine.

db.p Real value, variable, or expression that defines the number of the primary database in effect for the current menu page.

$cmd String value that contains the *ky.spawn* or *ky.m.spawn* command that initiated the menu spawn routine.

When used as an output, this string variable contains an error message or a new screen command to be processed by the menu driver.

Details

This program is written for the Dispensing Module and can function only with the dispense path database. Dispense Path Record fields are accessed and written to during the operation of this program. This program allows the operator to add, insert, and delete records from the database. Additional features include moving the robot to positions taught in different records. This program cannot allow additions or deletions of records during sequence operations. However, it does allow position teaching during walk-thru training.

Calling Sequence

```
RN.APPLY.DOT (app.path, dep.path, ns.tool, signal,
              disp.data[], error)
```

Function

Runtime primitive routine to apply a dot of material at a location. The APPLY.DOT statement uses this primitive.

Usage Considerations

This program assumes the appropriate Location database record is open.

Parameters

app.path — Real value, variable, or expression that defines the Path database record number for the optional path to follow while approaching the assembly location.

dep.path — Real value, variable, or expression that defines the Path database record number for the optional path to follow while departing from the assembly location.

ns.tool — Real value, variable, or expression that defines the record number of the signal database to be used. This database is required for the APPLY.DOT statement to function.

signal — Real value, variable, or expression that defines the record number of the signal database to be used. This database is required for the APPLY.DOT statement to function.

disp.data[ ] — Real value that contains the timing and gun information from the task statement call. Depending on the gun signal record specified, different data is required. Data comes from Variables database.

error — Real variable that receives the standard AIM operator error response code

Details

This routine performs the following steps:

1. If a tool offset transformation is specified, set the tool to this transformation.

2. Move along the optional approach towards the location.

3. Go to the dispensing location and turn on the dispense gun for the user-specified time.

4. Move along the optional departure path away from the location.

Related Keyword

rn.dispense

Calling Sequence

```
RN.DISPENSE (app.path, dep.path, dis.rec, hndsk, path.stat,
                 error)
```

Function

Runtime primitive to dispense material along a dispensing path. The DISPENSE statement calls this primitive.

Usage Considerations

This program assumes that the appropriate Dispensing Path database record is open.

Parameters

app.path — Real value, variable, or expression that defines the Path database record number for the optional path to follow while approaching the dispensing work area.

dep.path — Real value, variable, or expression that defines the Path database record number for the optional path to follow while moving away from the dispensing work area.

dis.rec — Real value that contains the record number of the dispense path database record use with the selected path.

hndsk — Real value that contains a logical value true or false of a variable record that will allow the path to start motion.

path.stat — Real value passed back to AIM Variables database indicating the current status of the path.

error — Real value that receives the standard AIM operator error response code.

Details

This program performs the following steps:

1. If the robot motions require a tool transform, set the tool to the transformation.
2. Fetch the path locations, then calculate the path.
3. If an optional approach path is specified, move along it.
4. Move along the defined dispensing path, turning the gun on and off at the appropriate path locations.
5. If specified, depart along the optional depart path.

Related Keyword

rn.apply.dot
rm.dm.calc.path

Calling Sequence

```
RN.DM.CUTTER.CP (comp, direct, plane, p.type[], path[], error)
```

Function

Runtime primitive to calculate a different path based on the compensation value entered in the Dispensing Path database. The compensation is applied with respect to the plane and direction of travel defined in the Dispensing Path database.

Parameters

comp	Real value, variable, or expression that defines the amount of compensation in millimeters to be applied to the path.
direct	Real value, variable, or expression that defines on which side of the path the compensation will occur relative to the direction of travel. The selection will determine whether the compensation will occur to the left or right relative to the direction of travel.
plane	Real value, variable, or expression that defines the plane the compensation will occur in. The available selections are 0 for XY plane, 1 for XZ plane, and 2 for the YZ plane.
p.type[]	Real array value that determines whether the path segment is linear or along an arc. A value of 0 defines the path segment as linear; any nonzero value defines the path segment as curved.
path[]	Location array variable that defines the positions of the path segments. These values are changed based on the compensation. Precision positions are handled as linear points. When used as an output, the location array variable contains the modified path segment positions based on the compensation values.
error	Real value that receives the standard AIM operator error response code.

Details

This program performs the following steps:

1. Determine the compensation value, direction, and plane for the values to be applied.

2. Determine the direction of travel based on the next position point. Also determine if the path segment is linear or is along an arc.

3. Decompose the path positions and select coordinates to be compensated (X, Y, or Z) based on selected plane.

4. Calculate intersections based on the next location and whether the path segment is linear or an arc.

5. Redefine the new path positions.

The returned path end points are then run through the path analyzer, which creates intermediate path points approximately 0.032 seconds apart. This program does not change the original path positions that are stored in the database. Therefore, the original positions are always available.

Calling Sequence

```
RN.DM.CALC.PATH (frame, dis.rec, segments, dispense.path[],
                 path.mve[,],  iosigs[], $ds.routine, perc,
                 gcomp[,], error, st)
```

Function

Calculates the intermediate path points along a dispensing path. Program *rn.dispense* calls this routine.

Usage Considerations

This program assumes that the appropriate Dispensing Path database record is open.

Parameters

frame — Transformation variable, function, or compound transformation specifying a reference frame. If the dispensing path locations are defined frame relative, they are defined with respect to *in.ref.*

dis.rec — Real value that defines the current open dispensing path record number. This number is needed for keeping track of the record number because of multiple records.

perc — Real value that allows the to be changed on a percentage basis.

segments — Real value that receives the number of segments that define the dispensing path.

dispense.path — An array of locations that contains the interpolated path that describes the dispensing pattern. Nominally, the locations are spaced 32 ms apart.

path.mve[,] — The dispensing path motion parameter list. The list describes the motion strategy for path segments. The symbolic names for the *path.mve[segment,]* array elements are described below.

dm.mve.straight	straight line flag
dm.mve.speed	speed (mm/s)
dm.mve.gun	gun flag (ON or OFF)
dm.mve.dwell	dwell time (seconds)
dm.mve.bead	bead size (percent)
dm.mve.nmoves	number of moves in path segment

dm.mve.duration	duration of robot moves
dm.mve.accel	acceleration for that path segment
dm.mve.sing	specifies if path check is required
dm.mve.arm.c	contains arm configuration bits
dm.mve.bnum	contains the belt number for tracking
dm.mve.trckflg	contains the belt window flag setting
dm.mve.mfrec	contains the conveyor record number
dm.mve.output	contains the node signal outputs
dm.mve.time	contains the estimated time for path checking

iosigs[] Real array values that define the signals to be used for the dispense gun and variable nozzle option. The iosigs array is documented in **Chapter 5**.

gcomp[] Real array containing gun compensation bit settings for every path point.

error Real variable that receives AIM's standard operator error response code.

st Real variable that contains a status error that has not been reported through the rn.error routine yet. These status errors are handled in higher order routines.

Details

This routine performs the following steps:

1. Check for the optional approach and depart transit locations in the Dispense Path database.

2. Get the data from each Dispensing Path record. This includes path positions and motion parameters

3. Define the parameters for the *path.mve[,]* array.

4. Redefine path positions if path compensation is selected.

5. Calculate the intermediate dispensing path locations.

6. Perform path checking if variable reference frame is not used.

Each Dispense Path Record defines one segment of the path. Motion parameters stored in the *path.mve[,]* array are defined for each path segment.

Related Keyword

rn.dispense
rn.dm.move
rn.dm.move.path
rn.pmove

Calling Sequence

```
RN.DM.MOVE (first.pt, last.pt, d.path[], p.mve[], iosigs[],
              frame, $ds.routine, bnum, seg, gcomp[,], error)
```

Function

Motion primitive to move the robot along a path segment.

Usage Considerations

The program *rn.dm.calc.path* is typically used to set up the data values for this program. Program *rn.dm.move.path* calls this routine.

Parameters

first.pt Real value, variable, or expression that specifies the first location in the input path array to move to.

last.pt Real value, variable, or expression that specifies the last location in the input path array to move to.

d.path[ ] The interpolated array of dispensing path points.

p.mve[ ] The interpolated path's associated motion parameter array. The array elements are detailed on the *rn.dm.calc.path* dictionary page.

iosigs[ ] Real array values that determine the output signals to be used for the dispense gun and proportional flow gun support.

frame Reference frame transformation used for determine next motion.

$ds.routine String variable that contains the specified gun routines to run during robot motion along path.

bnum Real variable that specifies the belt number used if conveyor tracking is specified.

seg Real variable that returns the current segment number used for the path motion.

gcomp[ ] Real variable that contains the current gun compensation bits used for each path point established.

error Real variable that receives the AIM standard operator error response code.

Details

This is the low-level motion routine used to move the robot along a dispensing path segment. The robot is moved using straight-line motions, and the DURATION instruction is used to handle the speed control. The values contained in the motion parameter list define the appropriate control parameters for the motions.

This routine calls the gun output routine for the dispense signals. After every motion instruction some calculation occurs that determines the current velocities and a look-ahead velocity if required.

Belt Tracking and HPS are also supported in this software if the options are specified.

Related Keyword

rn.dispense
rn.dm.move.path

Calling Sequence

```
RN.DM.MOVE.PATH (n.segments, d.path[], path.mve[,], iosigs[],
                 frame, $ds.routine, f.rec, hnd[], gcomp[],
                 error)
```

Function

Dispenses material along the input path. This handles communication with the pump server and analog control, and the path calculated motions from the Dispense Statement.

Usage Considerations

The program *rn.dm.calc.path* calculates the necessary input parameters for this routine.

Parameters

n.segments — Real value, variable, or expression that specifies the number of segments along the dispensing path.

d.path[ ] — Transformation array containing the interpolated path locations calculated for the dispensing path.

p.mve[,] — The dispensing path's associated motion parameter list defined for each path segment. The dictionary page for *rn.dm.calc.path* describes the array elements in this list.

iosigs[ ] — Real value array that defines the output signals to be used for the dispense gun and proportional flow gun support.

frame — Reference frame value for the path. A NULL frame value is used if no frame is defined.

$ds.routine — Routine name for gun signal actuation. This value is defined in the Dispense Signal database.

f.rec — Real variable that defines the first dispense record used for this path.

hnd[ ] — Real array variable that contains the handshake data which will determine when the path motion starts.

gcomp[ ] — Real array variable that contains the gun compensation data, which is used in the gun control program.

error — A real variable that receives AIM standard operator error response code.

Details

This routine moves the robot along a dispensing path. The dispensing gun I/O is handled within this program. The *p.mve[,]* motion parameter list defines the gun trigger locations along the dispensing path. This program calls *rn.dm.move* to perform the robot motions.

Related Keyword

rn.dispense
rn.dm.calc.path
rn.dm.move
rn.pmove

Calling Sequence

```
RN.PMOVE (#loc, sl, error)
```

Function

Motion primitive to move the robot to a precision point. This program can move the robot only in joint-interpolated motions.

Parameters

#loc Location value defined as a precision position rather than a world coordinate.

sl Real value, variable, or expression that specifies whether a linear or joint interpolated move is to occur. This value is not used and is here to keep the calling parameters the same as the *rn.move* AIM program.

error Real variable that receives AIM's standard operator error response code.

Details

This is the low-level motion routine to allow the robot to move to a precision point.

Calling Sequence

```
DM.GUN.STAND (iosigs[], p.mve[], vel, actvel, oncomp, offcomp,
               dot, data[], error)
```

Function

Dispense module routine to handle control of gun signals, analog control, and servo motor control. The Dispense module's *Dispense*, and *Apply_dot* statements, use this routine to handle gun control.

Parameters

iosigs[ ] Real array variable containing gun signal and analog / servo parameters that are used in calculation of process data for control.

p.mve[ ] Real value, variable, or expression that contains all of the motion parameters based on the current segment that is running.

vel Real value that defines the current velocity or look-ahead velocity based on the dispense signal parameters. These values are calculated based on path distances and specified time for the motion. If the robot cannot achieve the specified setting, these values will be inaccurate.

actvel Real value that defines the current actual velocity based on current position between last two points.

oncomp Real value that specifies the on gun control operations that will occur during execution of this routine.

offcomp Real value that specifies the off gun control operations that will occur during execution of this routine.

dot Real value that if logically true, then the apply_dot statement is using this routine.

data[ ] Real array value that defines the control data for the analog or servo pump when the apply_dot statement is run.

error Real variable that receives AIM standard operator error response code.

Details

This routine is executed every path point spacing. Faster path point spacing would force more execution on this routine which, allows faster control loops of all of the controlled devices. This routine can be copied and used under different names within the dispense module. The Dispense Signal database allows the user to specify other routine names on the menu page. Any custom routine must follow the same calling argument format.

Gun compensation flags are used in this routine to determine what items are executed during a cycle. The gcomp parameter uses bit comparisons to determine the processes to occur. These flags are described in **Chapter 5**.

The data[ ] array values used are described here. Data[1] specifies the analog voltage to be set by the routine. Data[2] specifies the amount of time the valves are open when running with analog. Data[3] specifies the volume to be dispensed when using the servo pump. Data[4] specifies the rate that the servo pump runs to dispense the volume.

Calling Sequence

```
DM.PURGE.STAND (arg, db.p, $cmd)
```

Function

This routine will perform the purging of the dispense system from a manual mode. It handles the outputs, analog nozzle, and servo pumps. This routine is executed from the manual dispense I/O menu.

Parameters

arg	Real variable that is returned from an AIM menu spawn. Not used here.
db.p	Real value, variable, or expression that specifies the current open database the menu is using.
$cmd	String array that contains the menu command string for future menu operations.

Details

This routine is specified in the Dispense Initialization database. The user can specify custom routines to perform purge operations from a menu. Dm.purge.stand is located in the *RUN_DM.V2* file and is fully commented so it can be customized by the user. It basically determines what options are selected in the *Dispense Signal* record and performs a purge cycle for the amount of time specified in the menu window. The default values for the time of the purge and the rate values are also located in the *Dispense Initialization* database.

Calling Sequence

```
    RN.PUMP.SERVER   ( )
```

Function

This program runs a special task that controls servo motor operations relative to the robot speeds or a commanded volume and rate if the Apply_dot statement is used. It uses the AIM client-server relationship to determine what operations are to occur.

Details

This routine uses the client-server relationship to select whether a spin-drive or a drive-drive operation is to occur. It uses the variables *func* and *qual* for these tasks. The func will return a value equal to *dm.spn.drv* or *dm.drv.drv,* which are global variables. The *qual* variable returns the task in which the client is running in. A global variable structure using an array with the name *dm.pmp.clnt* contains the information for the motions that are called in the routines dm.pump.drv and dm.pump.spn. The array values are documented in **Chapter 5** for further information.

Related Keyword

dm.pump.spn
dm.pump.drv
dm.gun.stand

Calling Sequence

```
DM.PUMP.SPN (tsk.index, error)
```

Function

Motion primitive to move a continuous unidirectional device, typically a servo pump, a specified speed with an optional reverse motion after completion of the first move. The speed of the motor is based on the parameters set in the Dispense signal database and the tool tip velocity of the robot as it moves along a path.

Parameters

tsk.index	Real variable that returns the task of the client that is using this routine. Global data from the client is provided based on the client's task number.
error	Real variable that receives AIM's standard operator error response code.

Details

This is the low-level motion routine to allow unidirectional motion to occur. The device that will be running is assumed to be defined with the JOINTS kinematic device module. This routine uses the V^+ Spin instruction to control the motor's speed. The speed is calculated in the routine *dm.gun.stand* and uses a variable *dm.pmp.cln*t to pass this information to the pump server and this routine. This routine is called from the pump server routine *rn.pump.server*. The variable dm.pmp.clnt uses the following global variables to specify the array indexes: dm.start, dm.stop, dm.revs, dm.speed, dm.suck.bck, dm.suck.spd, dm.end.

Related Keyword

rn.pump.server
dm.pump.drv
dm.gun.stand

Calling Sequence

```
DM.PUMP.DRV (tsk.index, purge, purge.dev, error)
```

Function

Motion primitive to move a continuous unidirectional device, typically a servo pump, a specified speed with an optional reverse motion after completion of the first move. The speed of the motor is based on the parameters set in the Dispense signal database. This routine is used with the Apply_dot statement as well as with the manual purge software.

Parameters

tsk.index Real variable that returns the task of the client that is using this routine. Global data from the client is provided based on the client’s task number.

purge Real variable that defines whether this routine is being executed by the manual purge software.

purge.dev Real variable that defines the robot device number that the motion will run with.

error Real variable that receives AIM’s standard operator error response code.

Details

This is the low-level motion routine to allow unidirectional motion to occur. The device that will be running is assumed to be defined with the JOINTS kinematic device module. This routine uses the V^+ Drive instruction to control the motor’s speed.   The speed is calculated in the routine *dm.gun.stand* or *dm.purge.stand* and uses the array variable *dm.pmp.cln*t to pass this information to the pump server or directly to this routine.   This routine is called from the pump server routine *rn.pump.server*. The variable *dm.pmp.clnt* uses the following global variables to specify the array indexes: dm.start, dm.stop, dm.revs, dm.speed, dm.suck.bck, dm.suck.spd, dm.end.

Related Keyword

rn.pump.server
dm.pump.spn
dm.gun.stand

Calling Sequence

```
DM.STRSTP.OUT (outsig, insig, tmr, error)
```

Function

This program actuates an output signal and waits for an input signal or a timer delay to continue processing. The timer delay is also used for a time-out if the input signal never turns on.

Parameters

outsig	Real value, variable, or expression that specifies an output signal to invoke.
insig	Real value, variable, or expression that specifies an input signal to wait for if specified.
tmr	Real value, variable, or expression that specifies a timing value for working with the input and output signals.
error	Real variable that receives AIM's standard operator error response code.

Details

This routine is used for handling the I/O at the start and at the stop of the path. The signals are specified in the Dispense Signal database. This routine is unprotected and is located in the file *RUN_DM.V2*. The user can modify the operation of the routine for special applications.

Related Keyword

rn.get.signals

Calling Sequence

```
DM.GET.SIGNALS (sig.rec, iosigs[], $ds.rtn, dm.on[],
                dm.ontg[], dm.of[], dm.oftg[], st)
```

Function

This routine retrieves the data from the Dispense Signal database. It establishes the *IOSIG* array variable for use in both the *Dispense* and *Apply_dot* sequence statements.

Parameters

sig.rec — Real value, variable, or expression that specifies the Dispense Signal record to use.

osigs[ ] — Real array variable that contains the I/O, analog, and pump parameters that are specified in the Dispense Signal database.

$ds.rtn — String variable that specifies the name of the gun signal routines to use during a path motion.

dm.on[ ] — Real array variable that contains the amount of time entered before a node point to be used for ON gun compensation.

dm.ontg[ ] — Real array variable that contains the flag value for the device that was selected with the On time specified.

dm.of [ ] — Real array variable that contains the amount of time entered before a node point to be used for OFF gun compensation.

dm.oftg[ ] — Real array variable that contains the flag value for the device that was selected with the On time specified.

st — Real variable that receives AIM's standard operator error response code.

Details

This routine is used by both the Dispense and Apply_dot statements to retrieve the data used for gun control. This is provided in one unprotected routine to allow custom modifications to be made to the Dispense Signal database and easy implementations to support changes. The routines dm.gun.stand, dm.purge.stand, and dm.strstp.out use the data that is retrieved by this routine.

Related Keyword

dm.gun.stand
dm.purge.stand
dm.strstp.out

PathWare Database Reference 7

The Process Path Database

name String (15 characters)

A standard name that uniquely identifies a Dispensing Path database record. This name is referenced by the DISPENSE task statement.

update date Date

The date and time when this record was last modified. This field is automatically set to the current date and time when information in this record is changed.

device Integer

The device that is to be used for dispensing.

index Integer

The index field is used in the dispensing database to keep track of the records that are stored under the same name. The physical record number is stored here.

location Transformation

This field stores the dispensing path locations. The following seven fields are used in conjunction with this location to describe several parameters about the location. Please refer to the ***AIM MotionWare Reference Guide*** for more information about these fields.

[location] Byte

This field is used for standard AIM location fields for motion.

location motion bits Byte

The location motion bit field describes the following motion parameters: coarse, fine, linear, joint, mm/sec, in/sec.

location speed Real

This field specifies the desired robot speed. The dispensing program requires all speed entries to be in millimeters per second.

location accel, decel Byte

This field specifies the accel and decel parameters for this location.

location configuration Byte

This database field defines the Adept robot arm configuration during the move (Righty or Lefty). Some manipulators may not need to use this field.

location type bits Integer

This is not used in the dispensing path programs, but is required for location fields.

[location frame]Integer

Source field for standard location records. It is not used in the dispensing path programs.

output state Binary

Binary value (displayed ON/OFF) signifying whether or not the dispensing gun should be activated for the corresponding path segment.

dwell time Real

The time (in seconds) the robot will pause at the last location of the specified path segment.

bead size Integer

Specifies the dispensed bead size for systems that include a proportional flow gun. The bead size is specified as a percentage of the maximum bead size for the gun.

tool name String (15 characters)

A standard name that specifies the tool offset used for the dispensing application. A record with this name must be present in the Tool database.

[tool] Integer

The number of the record in the Tool database which corresponds to the tool name field. The AIM linker automatically defines this value.

reference frame name String (15 characters)

A standard name that specifies the reference frame used for frame-relative motions. A record with this name must be present in the Reference Frame database.

[frame] Integer

The number of the record in the Reference Frame that corresponds to the reference frame name field. The AIM linker automatically defines this value.

signal name String (15 characters)

A standard name that specifies the signal database to used for the dispensing gun control. A record with this name must be present in the Signal database.

[signal] Integer

The number of the record in the Signal database that corresponds to the signal name field. The AIM linker automatically defines this value.

max index Integer

The max index field in the dispensing database stores the current number of records with the same name in the database.

path comp Real

This field stores the distance to compensate the path while using the tool compensation mode of the dispensing module.

direction Integer

This field determines whether to offset to the right or left of the direction of motion. Right and left are defined by the orientation of the plane. See the next field.

plane Integer

This field determines in which plane the tool compensation is to occur. The compensation can occur in the X-Y, Y-Z, and Z-X planes.

app_depart Real array

This field contains data used for the approach and depart parameters with the Dot features.

singularity checking Integer

This field in the database determines whether path checking occurs during operation of the path. The tracking options with path checking use the higher order bits.

tracking bits Integer

This field allows the tracking window bits to be set. This allows all of the window options to be selected from dispense menu.

device module Integer

This field specifies the kinematic device module for the robot being used with the dispense path.

output names String Array

This field contains the linked variable database record name for a signal output specification.

node outputs Integer Array

This field contains the output signal numbers that were specified in the Variable database.

menu names String Array

This field contains the linking names for the sequence and picture records used with the path editor.

menu selections Real Array

This field contains various menu selections used by the path editor.

Table 7-1. Process Path Database Record Definition

#	**Field Name**	**Data Type**	**Size**	**Sort**	**Array**	**User**
0	name	string	15	1		
1	update date	date	4			
2	device	integer	2			
3	index	integer	2	2		
4	location	transform	48			
5	[location]	byte	1			
6	location motion bits	byte	1			
7	location speed	real	4			

Table 7-1. Process Path Database Record Definition (Continued)

8	location acceleration	integer	1			
9	location deceleration	integer	1			
10	location configuration	byte	1			
11	location type bits	integer	2			
12	singularity checking	integer	2			
13	output state	binary	1			
14	dwell time	real	4			
15	bead size	integer	2			
16	tracking bits	integer	2			
17	device module	integer	2			
18	tool name	string	15			
19	[disp tool]	integer	2			
20	reference frame	string	15			
21	[reference frame]	integer	2			
22	signal name	string	15			
23	[signal]	integer	2			
24	max index	integer	2			
25	app_dep	real	4		10	
26	plane	integer	2			
27	path comp	real	4			
28	direction	integer	2			
29	output names	string	15		4	
30	node outputs	integer	2		4	

Table 7-1. Process Path Database Record Definition (Continued)

31	menu names	string	15		5	
32	menu selections	real	4		5	

Table 7-2. Process Path Database Variable Names

Variable Name	**Interpretation**
dp.db[ ]	Dispensing Path database number
cc.name	Field number for *name*
cc.date	Field number for *update date*
cc.device	Field number for *device*
dp.index	Field number for *index*
dp.loc	Field number for *location*
dp.loc.data	Field number for *[location]* record
dp.loc.mot	Field number for *location motion bits*
dp.loc.speed	Field number for *location speed* record
dp.loc.accel	Field number for *location acceleration*
dp.loc.decel	Field number for *location deceleration*
dp.loc.config	Field number for *location configuration*
dp.loc.type	Field number for *location type bits*
dp.gun.state	Field number for *output state*
dp.dwell.time	Field number for *dwell time*
dp.bead.size	Field number for *bead size*
dp.singularity	Field number for *path checking*
dp.dev.mod	Field number for *device module model number*
dp.tool.name	Field number for *tool name*
dp.tool	Field number for *[tool] record number*

Table 7-2. Process Path Database Variable Names (Continued)

dp.frame.name	Field number for *reference frame name*
dp.frame	Field number for *[frame] record* number
dp.signal.name	Field number for *signal name*
dp.signal	Field number for *[signal] record number*
dp.max.index	Field number for *max index* counter
dp.jnt.config	Field number for *joint configuration*
dp.plane	Field number for path comp *plane*
dp.comp	Field number for *path compensation*
dp.direct	Field number for path direction
dp.out.name	Field number for node *output names*
dp.out.number	Field number for node *output numbers*
dp.str.block	Field number for menu string links
dp.real.block	Field number used for menu block selections

The PathWare Signal Database

The database configures the dispensing gun and flow nozzle signals. This database is required for both the dispensing path database field and APPLY.DOT statement for the manipulator to execute a path. This database has an enable signal field which allows the dispense gun to be turned off while executing a cycle in dry run mode.

The Dispense Signal field descriptions are in Table 7-3. Use the variable names listed in Table 7-4 in application programs to access data from the Dispense Signal database. A description of each field in the Dispense Signal database appears below.

name String (15 characters)

A standard name that must uniquely identify the Dispense Signal database. It is linked to the Dispensing Path database with this name and is used in the APPLY_DOT sequence statement as well.

update date Date

The date and time when this record was last modified. This field is automatically set to the current date and time whenever any information is changed by the operator.

max speed Integer

The max speed data field specifies at what speed (mm/sec) the variable flow nozzle of the dispense gun is open 100%. This field is used to determine the percentage open the nozzle is while it moves about the programmed path.

gun signal Integer

This field specifies the output signal number that is used to turn the dispense gun on and off.

select velocity control Integer

This field specifies whether a analog or servo velocity control features are used.

device channel Integer

This field specifies the analog channel number for the analog board.

output name String Array

This field contains the output signal names that link with the Variable database.

output number Integer Array

This field contains the signal output number for the devices to actuate.

input name String Array

This field contains the input signal names that link with the Variable database.

input number Integer Array

This field contains the signal input number for the Start/Stop and Look Ahead menu fields.

delay Real Array

This field contains the timer real data for the start/stop signal setup.

device routine String

This field contains the name of the gun routine to run during path motion.

device on time Real Array

This field contains the ON gun compensation times before the node for operation.

device on type Integer Array

This field contains the number of the selected device keyword that relates to the ON time specified.

device off time Real Array

This field contains the OFF gun compensation times before the node selected.

device off time Integer Array

This field contain the number of the selected device keyword that relates to the OFF time specified.

pump parameters Real Array

This field contains the servo pump parameters specified in the Dispense Signal menu. This includes flow rates, flow scale, suck back volume, suck back rate, and device number.

Table 7-3. PathWare Signal Database Record Definition

#	**Field Name**	**Data Type**	**Size**	**Sort**	**Array**	**User**

Table 7-3. PathWare Signal Database Record Definition (Continued)

0	name	string	15	–1		
1	update date	date	4			
2	device number	integer	2			
3	max speed	integer	2			
4	select velocity control	byte	1			
5	device channel	integer	2			
6	output name	string	15		10	
7	output number	integer	2		10	
8	input name	string	15		5	
9	input number	integer	2		5	
10	delay	real	4		2	
11	device routine	string	15			
12	device on time	real	4		5	
13	device on type	integer	2		5	
14	device off time	real	4		5	
15	device off type	integer	2		5	
16	pump parameters	real	4		5	

Table 7-4. PathWare Signal Database Variable Names

Variable Name	**Interpretation**
ds.db[ ]	Signal database number
cc.name	Field number for *name*
cc.date	Field number for *update date*
ds.max.speed	Field number for *max speed*
ds.flow.gun	Field number for selecting *flow guns*

Table 7-4. PathWare Signal Database Variable Names (Continued)

ds.analog	Field number for *analog channel number*
ds.out.name	Field number for *output signal names*
ds.out.number	Field number for output signal numbers
ds.in.name	Field number for *input signal names*
ds.in.number	Field number for *input signal numbers*
ds.delay	Field number for *time delays*
ds.routine	Field number for *signal user routine*
ds.sig.on	Field number for *On gun compensation*
ds.sig.type	Field number for *On device type*
ds.sig.off	Field number for *Off gun compensation*
ds.type.off	Field number for *Off device type*
ds.pump.parms	Field number for *pump parameters*

Error Codes 8

PathWare Error Messages

This appendix describes the Dispensing Module error codes. Each description includes the message text, its error code, an explanation of the probable cause of the message, and the suggested user action.

Table 8-1 lists the error codes in numerical order. The variable names used to access the error codes are also listed.

Table 8-1. PathWare Error Codes

Code	Variable Name	Text of Message
4575	ec.dm.calc.path	Calculating Dispense Path
4576	ec.dm.calc.comp	Calculating: Path Compensation
4577	ec.dm.set.out	Setting output signals
4600	ec.ch.put.hand	Putting hand in nest
4601	ec.ch.get.hand	Getting hand from nest
–6575	ec.dm.no.path.s	*No path segment defined*
–6576	ec.dm.no.points	*No points defined in the path segment*
–6577	ec.dm.no.circle	*Circular interpolation failure*
–6578	ec.dm.no.signal	*No Signal Database Defined*
–6579	ec.dm.no.licens	*AIM Applications License is not installed*
–6580	ec.dm.no.input	*Input signal did not change state*
–6581	ec.dm.no.speed	*Path speed is set to near zero value*
–6582	ec.dm.acdc.ovr	*Accel/Decel is overlapping on ARC segment*
–6583	ec.dm.gun.comp	*Gun Compensation is specified on first point*
–6584	ec.dm.gun.index	*Too short a distance for Gun Compensation*

Table 8-1. PathWare Error Codes (Continued)

–6585	ec.dm.zero.flow	*Servo pump rate is set to zero*
–6586	ec.dm.max.flow	*Flow rate exceeds max flow specified*
–6587	ec.dm.sing.chk	*Path Check failed*
–6588	ec.dm.no.server	*Servo pump server is not executing*
–6589	ec.dm.cir,gun	*Attempt to change gun state at Arc Midpoint*
–6590	ec.dm.cir.def	*First and Last point of Arc are the same*
–6600	ec.ch.h.robot	*Hand attached to the robot*
–6601	ec.ch.no.stand	*No hand on stand*
–6602	ec.ch.no.h.rbot	*No hand attached to the robot*
–6603	ec.ch.h.stand	*Hand on stand*
–6604	ec.ch.bad.hand	*Wrong hand attached to the robot*

Calculating Process Path (4575)

Explanation Trace message displayed during walk-through training indicating that the runtime system is interpreting some data.

User action No action required.

Calculating: Path Compensation (4576)

Explanation Trace message displayed during walk-through training indicating that the runtime system is interpreting path compensation data. If many path points are taught, path compensation can take several seconds.

User action No action required

Setting outputs at start or end of path (4577)

Explanation Trace message displayed during walk-through training indicating that the runtime system is invoking output signals at the begining or end of a path.

User action No action required

***No path segment defined* (-6575)**

Explanation No path segment was specified for the current Dispensing Path database record. At least one path segment is needed to define a dispensing path.

User action Verify that the appropriate path segments are specified in the Dispensing Path database record.

***No points defined in the path segment* (-6576)**

Explanation Not enough path records defined for a dispensing path segment.

User action Verify the path segment data in the Path database, checking that the appropriate flags are set and the correct locations are specified as points. Note that additional path records cannot be added during walk-through training.

***Circular interpolation failure* (-6577)**

Explanation An arc could not be fit to points defined for a dispensing path segment.

User action Verify the path segment data in the Dispensing Path database, checking that the appropriate flags are set and the correct locations are specified as arc points.

***No signal database defined* (-6578)**

Explanation No signal database record name has been defined in the Dispensing Path database.

User action Select the correct signal database name to be entered in the Dispensing Path database field or in the APPLY.DOT task statement.

***AIM PathWare License is not installed* (-6579)**

Explanation The software license to run the PathWare is not installed on your system.

User action Check the controller's NVRAM by using the config_c utility program. The AIM PathWare license must be installed for operation. Contact Robot-Doctor for License.

***Input Signal did not change state*(-6580)**

Explanation The specified input signal did not come on during the operation within the time limits specified.

User action Check operation of input signal and time delay in Dispense Signal database for proper operation.

Precision Point speed set to slow for floating move (-6581)

Explanation When using conveyor tracking with a floating precision point, the robot speed must be great enough to allow the robot to work with conveyor tracking. Also used if values of zero are entered for speed.

User action Raise the robot speed.

Circular Interpolation - Accel Decel profiles overlap (-6582)

Explanation During Arc motions the acceleration and decleration profiles cannot overlap or pass the taught midpoint of the arc.

User action Lower the time for acceleration or deceleration or the speed.

Gun Compensation can not work on first path point (-6583)

Explanation Gun compensation requires path points before the compensation can occur.

User action Teach entry points before the first location where the gun is required to be turned on.

Gun Compensation Index limit (-6584)

Explanation Not enough path points before a gun operation with gun compensation is to occur.

User action Change the speed of the motion before the gun compensation point or lower the timing of the gun compensation.

Pump Flow rate is set to zero (-6585)

Explanation The pump speed if set to zero.

User action Servo Pump setting must allow operation of the pump.

Required flow rate exceeds maximum pump flow rate (-6586)

Explanation The parameter setting of the path with the servo pump exceeds the maximum flow rate set in the dispense signal database.

User action Reduce path speeds or change the pump parameters for proper operation.

Path Check Failed (-6587)

Explanation The dispense path checking utility found joint velocities along the path that exceed the maximum joint velocities set in the Initialization database.

User action Change the joint configuration or location of the path in the robot's work envelope or lower the speed.

*Dispense pump server is not executing * (-6588)

Explanation The server for the dispense pump is not running.

User action Make sure the servo pump has been turned on in the Initialization database and AIM has been restarted. Make sure the pump task has been started before the robot task is started.

Gun state changed at Arc midpoint (-6589)

Explanation The gun state can be changed only at the begining or end of a segement. The midpoint of the Arc cannot be different from the starting point.

User action If the gun needs to change state at the midpoint, then teach break the Arc into two segments.

Circle definition error (-6590)

Explanation The first and last record of an arc cannot be the same point. Therefore circle need to be defined as two arcs.

User action Break the circle into two arc than can be defined as five points.

Putting hand in nest (4600)

Explanation Trace message displayed during walk-through training indicating that the robot is returning its end-effector to a tool nest.

User action No action required.

Getting hand from nest (4601)

Explanation Trace message displayed during walk-through training indicating that the robot is getting a new end-effector from a tool nest.

User action No action required.

Hand attached to the robot (-6600)

Explanation During a CHANGE.HAND acquire operation, the desired end-effector is already attached to the robot. The CHANGE.HAND statement is skipped. If the tool is being returned, the hand was not released from robot.

User action Make sure the tool release and acquire signals are working. If no end-effector is attached to the robot, make sure the tool signature and tool attached signals are working properly.

No hand on stand (-6601)

Explanation During a CHANGE.HAND operation, the desired end-effector is not in its stand. If a tool is being acquired from its tool nest, the tool-on-stand signal is not set. If an end-effector has just been returned to the stand, the signal is not set.

User action Verify that no tool is on the stand. If there is a tool, ensure that the tool-on-stand switch is being activated properly and the I/O lines are working.

No hand attached to the robot (-6602)

Explanation After a CHANGE.HAND acquire sequence, the desired end-effector is not attached to the robot.

User action Make sure the tool release and acquire signals are working. If the correct end-effector is attached to the robot, make sure the tool signature and tool attached signals are working properly.

Hand on stand (-6603

Explanation Before returning a tool to its stand in a CHANGE.HAND sequence, the tool-on-stand signal is set; an end-effector is already in the nest. If a tool is in the nest after a tool acquire, the end-effector was not acquired by the robot.

User action Verify that there is a tool on the stand. If there is no tool, ensure that the tool-on-stand switch is being activated properly and the I/O lines are working.

Wrong hand attached to the robot (-6604)

Explanation The wrong end-effector is attached to the robot. In a CHANGE.HAND acquire sequence, the robot picked up the wrong hand. If the robot is returning a tool, the tool specified for return is not the hand attached to the robot.

User action If the correct end-effector is attached to the robot, make sure the tool signature and tool attached signals are working properly.

Index

RobotDoctor User's Manual Comment Form

We have provided this form to allow you to make comments about this manual, to point out any mistakes you may find, or to offer suggestions about information you want to see added to the manual. We review and revise user's manuals on a regular basis, and any comments or feedback you send us will be given serious consideration.

If you prefer to submit your comments by e-mail, you can send them to:

sroell@RobotDoctor.com

Thank you for your input.

NAME ______________________________ DATE ____________________

COMPANY ____

ADDRESS ____

PHONE ____

MANUAL TITLE ____

PART NUMBER and REV level ________________________________

COMMENTS:

____

____

____

____

____

____

____

MAIL TO:

RobotDoctor, LLC
Technical Publications Dept.
9220 Nottingham Way
Mason, OH 45040

FAX: (513) 398-9619