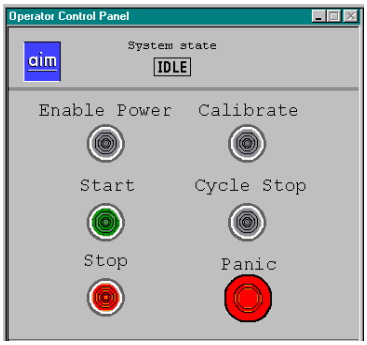
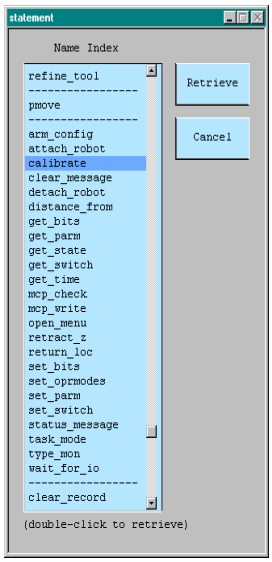
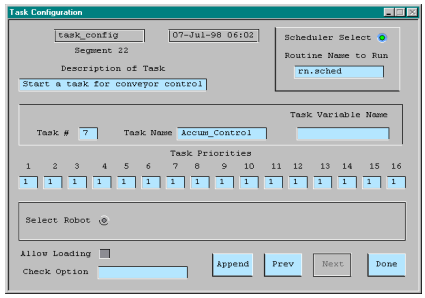
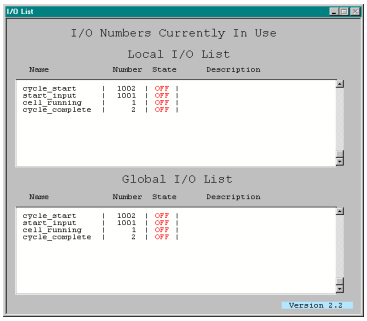
AIM Utilities

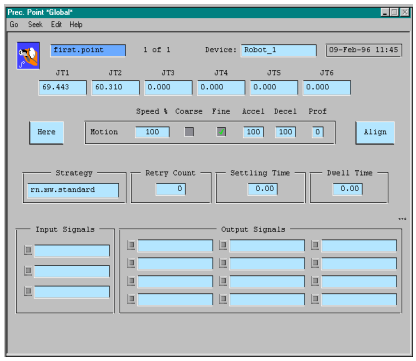
User's & Reference Guide









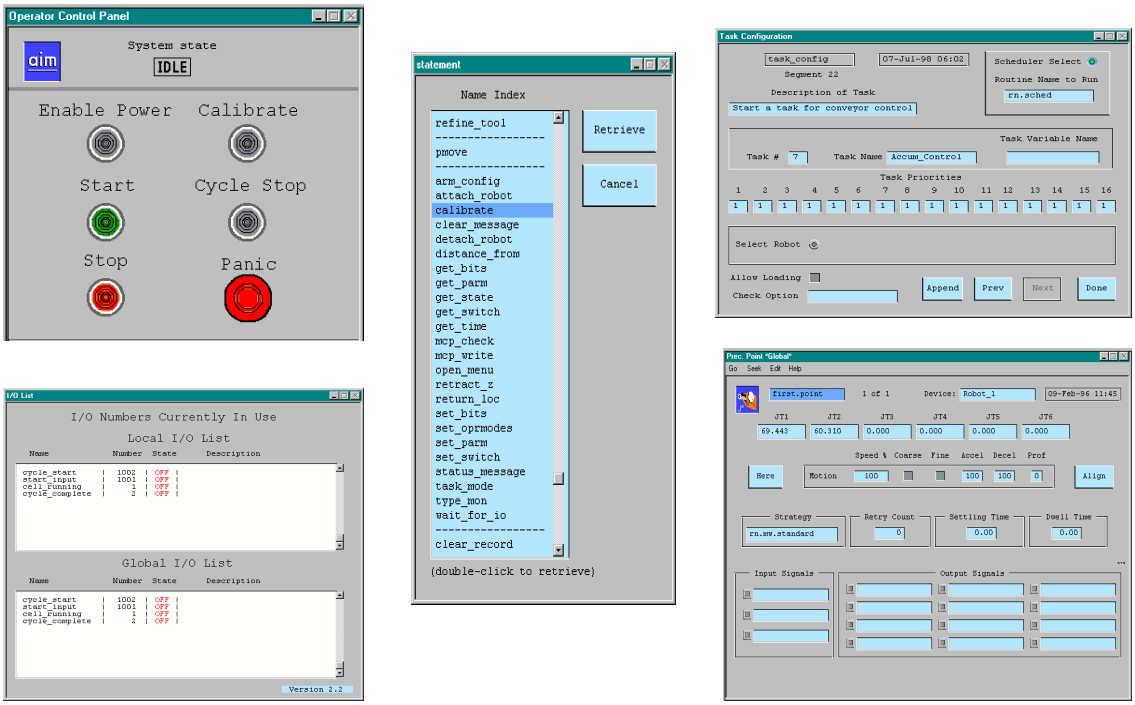


Version 3.2 B

adept®

AIM Utilities

User's & Reference Guide



Part Number RDASG-C0004 Rev. 3.2B

March 1999

adept technology, inc.

150 Rose Orchard Way • San Jose, CA 95134 • USA • Phone (408) 432-0888 • Fax (408) 432-8707

Otto-Hahn-Strasse 23 • 44227 Dortmund • Germany • Phone (49) 231.75.89.40 • Fax(49) 231.75.89.450

41, rue du Saule Trapu • 91300 • Massy • France • Phone (33) 1.69.19.16.16 • Fax (33) 1.69.32.04.62

1-2, Aza Nakahara Mitsuya-Cho • Toyohashi, Aichi-Ken • 441-31 • Japan • (81) 532.65.2391 • Fax (81) 532.65.2390

Printed in the United States of America

Table Of Contents

List of Figures

List of Tables

1 Introduction

1.1 What is The AIM Utilities Module?

The AIM Utilities Module is an AIM application module designed specifically with the implementation of AIM in mind. The module simplifies the process of programming AIM by providing more AIM sequence statements as well as other high end features. The AIM Utilities Module includes Operator Control panels and additional icons.

Unlike other AIM modules, the AIM Utilities Module must be used in conjunction with the MotionWare, PCB, or VisionWare application module.

1.2 Using This Manual

This manual presents a detailed description of the AIM Utilities Module. 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 on the robot and controller. See the *MotionWare User's Guide* for details on using the basic AIM system. If you will be customizing the AIM Utilities Module, we assume you are familiar with the V+ operating system and language. See the AIM reference manuals and the *V+ Language Reference Guide* for information on customizing and writing V+.

Chapter 1 presents an overview of the software features of the AIM Utilities Module. This chapter should be read by all AIM Utilities Module users.

Chapter 2 describes the software installation and startup procedures for the AIM Utilities Module. This chapter is intended for system customizers who will install the software and users interested in the AIM Utilities Module software organization.

Chapter 3 describes the menus and databases used with the AIM Utilities Module. All users should review this chapter and become familiar with these databases.

Chapter 4 details the uses of the AIM Utilities Module. The AIM statements and other features are discussed here. Example applications are reviewed. This chapter should be read by all AIM Utility Module users.

Appendix A contains example sequences from the database examples disk.

Appendix B contains information about the Quick-Change Module. It describes the databases, and operation of the software package.

1.3 Software Overview

AIM Utilities is an application module of the AIM software system. It was developed to speed implementation of Adept's AIM Systems. To achieve these goals, AIM Utilities adds 30+ new task-level statements. These statements allow the user to create calibration sequences, easier use of IO, access to internal AIM settings, statements to create MCP pendant displays from a sequence and many other features.

AIM Utilities provides two software utilities. One allows easy installation and selection of software modules. The other utility allows the ability to add and select executing tasks. These items used to require customization of AIM software. These utilities allow the customizer to create software libraries that can be easier to install by the user.

AIM Utilities also adds the ability to program precision point motions. This allows the user to program robots with locations that have multiple arm configurations.

AIM Utilities provides the ability to program paths relative to frames of references. This is needed when user wants to use the path features for process type applications.

AIM Utilities requires AIM version 3.2+ and the Adept V^+ operating system version 11.4 or higher. Some features of the AIM Utilities are described below. Details are presented in subsequent chapters.

Manual Control Pendant Statements

These statements allow the user to write sequences to display menu's and provide an operator interface using Adept's MCP.

Statements for Auto-Startup and Robot Calibration

Several statements have been added to aid in typical startups of Adept's equipment. An example sequence is provided to better show the uses of these instructions.

Example Sequences and Databases

The AIM Utilities software package includes example databases and sequences that are provided to aid in the development of your applications.

1.4 Hardware Requirements

The PalletWare Module requires the following Adept controller options:

- Adept A-series controller or MV controller with AdeptWindows
- Adept Teach Pendant (MCP)
- Mouse, Monitor, and Keyboard (not required with AdeptWindows)
- 68030 / 68040 / 68060 system processor (with 8 Mb's of RAM) The 68060 processor is recommended for high end applications that demand more processing speed.

1.5 How Can I Get Help?

Service Calls

Adept Technology maintains a fully staffed Customer Service Center at its headquarters in San Jose, CA.

(800) 232-3378

When calling Customer Service, please have the serial number of the robot (if you have one), the serial number of the controller, and the system software version. The system software version is available by issuing the command "ID" at the system prompt.

Training Information

For Adept Training Course information, please call (408) 434-5024.

Application Information

There is also a dedicated phone line for assistance with robot applications. The AIM Utilities Software Package is considered an Adept RDA Services product. Please call the Adept Cincinnati Office with software related questions at 513-792-0266.

International Customer Assistance

For information on training, service, or applications, Adept also has a Customer Service Center in Dortmund, Germany. The phone number is: (49) 231/75 89 40.

Outside of Europe or the USA, call (408) 434-5000.

2 Installation

2.1 Before You Begin

This chapter describes the AIM Utilities 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 3.2 software files, and the MotionWare application module files must already be installed. See the *MotionWare User's Guide*

> **NOTE:** The procedures outlined in this chapter will replace MotionWare files. Any previous AIM customizations may be replaced. If you have customized AIM, check below to assure your files will not be overwritten before installing the AIM Utilities Module.

Table 2-1 lists the files that are included with AIM Utilities. Three diskettes are provided in the AIM Utilities package. They are titled AIM Utilities Software, AIM Utilities Example Databases, and AIM Utilities Software Plug-Ins. The AIM Utilities Software disk contains all of the software for AIM Utilities, this includes the runtime software, the database definition files, and the database files. 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.).

The disk labeled Example Databases contains AIM database modules of examples for different applications. These modules can be loaded by the user to show how to program different applications.

The disk labeled Software Plug-Ins contains additional software utilities that can be loaded into the system via the Software Loader Menu and the Install button option. These plug-ins are generally nice features that not all customers will need to use.

To install the AIM Utilities, follow the steps in section 2.2. 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 AIM Utilities Files

Disk Files		**Contents**
AIM Utilities Software Disk		
AIMUTIL.V2		AIM Utilities Runtime Software routines
INFMOD.OVR	**P**	Plug-In Utilities Menu routines
INFMOD.MNU		Plug-In Utilities AIM Menu
IOL.MNU		IO List Menu file
IOLMOD.OVR		IO List menu routines
IOLMOD.INF		IO List Plug-In Configuration File
IOLRUN.V2		IO List Runtime Routines
LAIM.V2		Loader routines for generalized AIM
MENU_UT.V2	**P**	Menu routines for AIM Routines
MNT.MNU		NFS AIM Menu files
MNTINI.DB		NFS Initialization Database
MNTMOD.INF		NFS Mounting Plug-In file
MNTMOD.OVR		NFS initialization overlay file
MNTRUN.V2		NFS runtime routines
MOW.MNU	✔	Location Record Menu
PATH.SQU	**P** ✔	Modified Path routines
PLGMAIN.ADV		Plug-In software advisor
PLGMAIN.MNU		Plug-In menus
PLGMOD.OVR		Plug-In menu routines
PPOINT.V2		Precision Point runtime software
PPT.MNU		Precision Point AIM Menus

PPT.RFD		Precision Point RFD file
PPT.DB		Precision Point database
PPTICON.DAT		Precision Point Icon data file
PPTMOD.OV2		Precision Point Initialization commented file
PPTMOD.INF		Precision Point Plug-In file
PPTMOD.OVR		Precision Point Initialization file
STATPPT.DB		Precision Point Sequence statement file
STATUT.DB		AIM Utilities Sequence statement file
STS.MNU		Cell Status Menu
STSINI.DB		Cell Status Initialization Database
STSMOD.OVR		Cell Status Initialization file
STSMOD.INF		Cell Status Plug-In file
STSRUN.V2		Cell Status Runtime software
TCHTL.SQU	**P**	Tool Offset teaching software
TOOL.MNU	✔	Tool Offset Menu
UTICON.DAT		AIM Utilities Icon Data file
UTIL.MNU		Station RFD file
UTINI.MNU		AIM Utilities Software Loader Menu File
UTINI.RFD		AIM Utilities Software Loader RFD file
UTINI.DB		AIM Utilities Software Loader Database
UTLINI.DB		AIM Utilities Initialization database
UTMOD.INF		AIM Utilities Plug-In file
UTMOD.OVR		AIM Utilities Initialization Loader File

UTTSK.RFD		Task Configurator RFD file
UTTSK.DB		Task Configurator Database
UTTSK.MNU		Task Configurator Menu
AIM Utilities Plug-Ins		
Quick Change Plug-In		
QCMOD.OVR		Initialization routine for loading Quick Change
QUICKCH.DB		Quick Change database
QUICKCH.MNU		Quick Change menu database
QUICKCH.RFD		Quick Change RFD file
QC_IO.MNU		Quick Change IO display menu
QCINI.DB		Quick Change Initialization database
QCICON.DAT		Icons for Quick Change
QCMOD.INF		Quick Change Plug-in loader file
RUN_CH.SQU		Quick Change squeezed runtime software
RUN_CH.V2		Quick Change commended runtime software
ERRORQC.DB		Quick Change error database
MENU_CH.SQU		Quick Change Squeezed Menu routines
MENU_CH.V2		Quick Change Commented Menu routines
STATQC.DB		Quick change sequence statements
ConnectWare Plug-In		
CNWINI.DB		ConnectWare initialization Database
CNWMOD.INF		ConnectWare Plug-In Loader file
CNWMOD.OVR		ConnectWare Software initialization file

CNWRUN.SQU		ConnectWare runtime software
STATCNW.DB		ConnectWare sequence statements
Event IO Manager		
EVENT.DB		Event Manager database
EVT.MNU		Event Manager menu file
EVT.RFD		Event Manager RFD file
EVTINI.DB		Event Manager Initialization database
EVTMOD.INF		Event Manager Plug-In file
EVTMOD.OVR		Event Manager software initialization file
EVTRUN.V2		Event Manager runtime software
STATEVT.DB		Event Manager Statements
HPGL Graphics Module		
HPGINI.DB		Graphics Module Initialization database
HPGL.MNU		Graphics Module menu database
HPGMOD.INF		Graphics Module Plug-In File
HPGMOD.OVR		Graphics Module Software Initialization file
HPGRUN.V2		Graphics Module runtime software
TCP Server		
TCPMOD.INF		Server Plug-In file
TCPINI.DB		Server Initialization database
TCPMOD.OVR		Server Software initialization programs
TCPRUN.V2		Server Runtime Software
TCP.MNU		Server Menu Database
STATTCP.DB		Server Statements

AIM Utilities Examples		
EX_DISP.MOD		Dispense Module Example Databases
EX_MCP.MOD		MCP Example
EX_PALWR.MOD		PalletWare Example
EX_START.MOD		Startup Example

2.2 Installation Procedure

You can use the V+ utility program DISKCOPY to copy the AIM Utilities floppy disks 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 AIM Utilities files. Copy the AIM Utilities files to the same directory as *AIM MotionWare*.

Note: For AIM versions before AIM 3.2, Do Not install the file *PATH.SQU*. Aim versions before 3.2 will not work with frame relative paths.

1. Turn on your controller.
2. When the start up procedures have completed, place the *Adept Utility Programs* Disk in the A: drive and enter the command. Most newer Adept systems have the utility programs in the \util\ subdirectory on the hard drive. (The hard drive can either be drive C or D depending if an AWC processor is used.)

   LOAD A:DISKCOPY or (C:\UTIL\DISKCOPY) or (D:\UTIL\DISKCOPY)
3. When the program files have completed loading, enter the command:

   EXECUTE a.diskcopy
4. Remove the utility disk, and insert the AIM Utilities Software disk in the A: drive or if your are running an AWC processor another disk drive will have to be mounted using NFS.
5. From the menu that is presented, select:

   4. => Copy multiple files
6. Answer **A** or **NFS>mydrive** (when using an AWC processor) when prompted:

   What is the INPUT disk (e.g., A/B/C/D)?
7. Answer **C** or **D** (if using an AWC processor) 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 sub-directory name may be specified):

   Enter output subdirectory (blank for default):
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 last disk 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.

2.3 Verifying Installation

To verify the correct installation of the AIM Utilities Module, follow the steps below.

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

   LOAD LAIM.V2

   COMMAND LAIM

2. AIM should now load and boot up properly. If you have not installed the Dispense Module or PalletWare software, the system will default to loading the MotionWare application software.

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 Adept Applications Engineering at the number listed in Chapter 1.

2.4 Installing Utilities to existing Applications

The AIM Utilities software can be installed on existing applications provided any customizations followed Adept recommended procedures. Generally, customizers modify the routines provided in the custom.squ file which is the proper place for customizations. The AIM Utilities package provides a different AIM loader, which means any changes to the program *ai.module.init* will be lost.

2.5 Installing AIM Dispense Module or AIM PalletWare

The Dispense 3.2E+ and PalletWare AIM modules are provided with a software plug-in configuration file that will allow you to load these applications either as a plug-in or by the standard way of using the diskcopy utility. If you are installing both AIM Utilities and either the Dispense Module or PalletWare, we recommend using the standard method as shown in installing the AIM utility package. In both cases, you first install AIM Utilities and then install either the Dispense Module or PalletWare. These packages are loaded last because they will overwrite the software module loader database file so that all of the Dispense or PalletWare software is loaded when starting LAIM.

If you are adding either of the two packages above to an existing application, Adept recommends installing them as a plug-in. This is done by starting AIM, and selecting the Software Loader option from the Setup pulldown. Then select the Install button. You need to have the Dispense or PalletWare disk in a disk drive and select this drive. The Install file will either be *DMMOD.INF* or *PWMOD.INF* for Dispense or PalletWare, By selecting the file to install, all of the required files will be loaded.

2.6 Selecting Default AIM Modules

The Software Loader option in AIM Utilities allows the user to select one of the following AIM Applications to load:

PCB Module
MotionWare
VisionWare
Dispense Module
PalletWare

Once selecting one of these options, the software will select the proper software loading and task configurations for this module. The user now only needs to shutdown AIM by either selecting from the Software Module loader or from the "Special" AIM pulldown. Then restart the system by executing the following line from the V+ monitor.

Execute 1 a.aim

2.7 AUTO File Installation

Provided with the AIM Utilities software package is a file on the AIM Utilities Software disk called AUTO.V2. This file can be copied to the root directory so that when the system is booted, AIM will load and start automatically. The Adept controller must be configured so that the Auto file will load and execute automatically. This can be accomplished by setting dip switches on the SIO board if an Adept MV controller is purchased without the AWC controller. If the system has an AWC processor, then the auto startup would be configured in the Config_C program. Please refer to the *Adept MV Controller User's Guide* for more details.

The AUTO.V2 file can be copied by typing in the following V+ Monitor instruction.

FCOPY C:\AUTO.V2 = A:AUTO.V2

This instruction assumes the AIM UTILITIES Software disk is in drive A. If using an AWC processor NFS will have to be used instead of the A drive.

AIM Utilities Menus 3

AIM Utilities Menu Pages

This chapter describes the menu pages used by the AIM Utilities software. Several new menu pages are added to the AIM software package when AIM Utilities software is installed. These include menus for software and task initialization of AIM, Operator control panel, I/O menu, NFS mounting utilities, Tool Offset Teaching and others.

This chapter will provide a general overview of the menus and describe the features by a graphic numeric bubble with text following the figure.

3.1 Software Module Loading

The Software Module Loading database is provided to allow the user to load custom software modules or other software files into the AIM system. This provides the user an easy method to load new software into the AIM system without customization. Please note that the order of which the software is loaded is based on the position in the database. In some cases, software must be loaded in a correct order for proper functionality.

To review or edit the current software being loaded select the Setup pulldown and the Software Module loader option.

Setup ➡ Software Loader

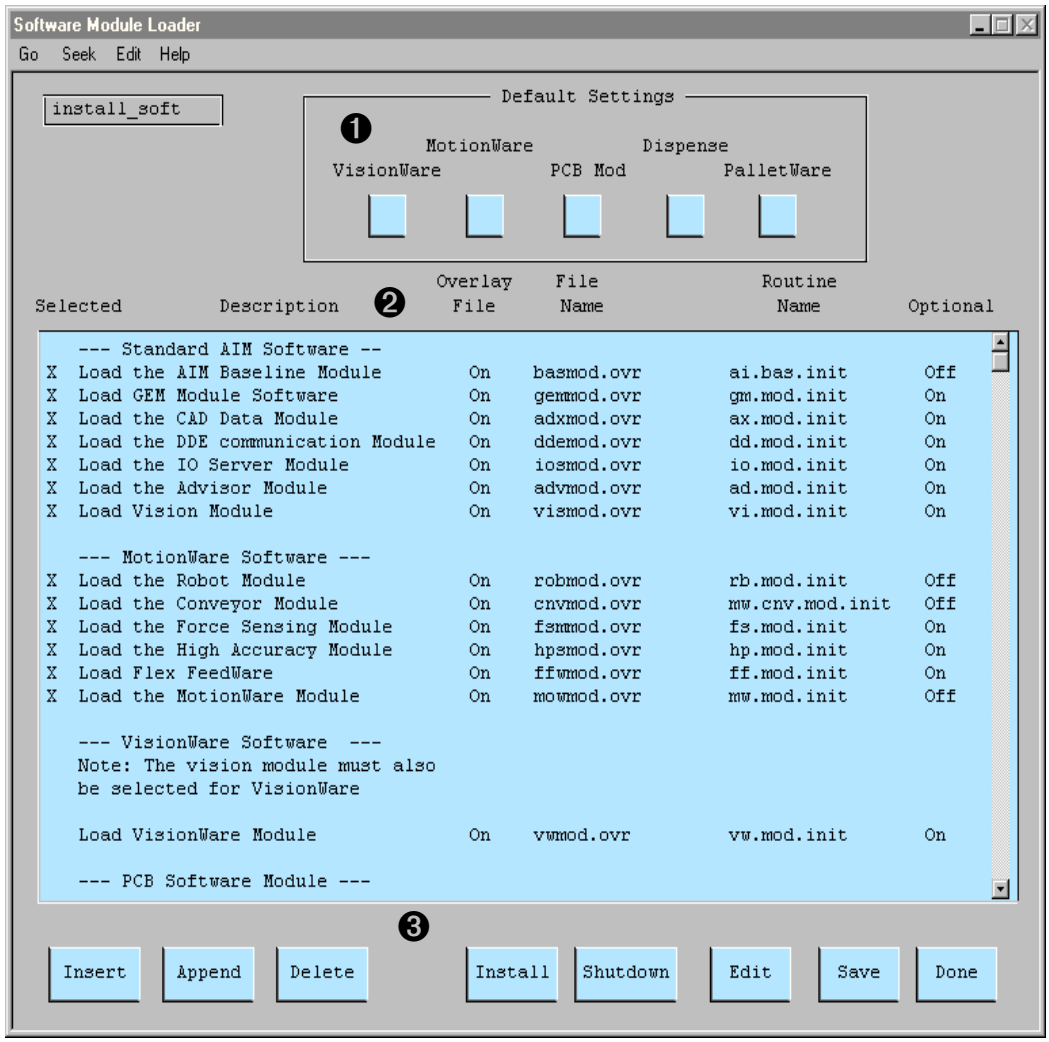


Figure 3-1 Software Module Loader

❶ ***Default AIM Software Setting:***
This allows the user to load any of the current Adept Software modules as long as they have been loaded into the system. The Task Configuration database will be modified after the selection. After selection of the software you must restart AIM for the software to be reconfigured. Be sure to save the databases before restarting AIM.

VisionWare Selection - Pressing on this button will select the standard VisionWare software modules to load as well as the task selections. The Task Configurator database will be changed so that the standard VisionWare tasks will be selected and running the next time AIM is restarted.

MotionWare Selection - Pressing on this button will select the standard MotionWare software modules to load as well as the task selections.

PCB Software Selection - Pressing on this button will select the standard PCB software modules to load as well as the task selections.

Dispense Selection - Pressing on this button will select the standard Dispense module software modules to load as well as the task configurations.

PalletWare Selection - Pressing on this button will select the standard PalletWare software modules to load as well as the task configuration.

❷ ***Software Modules Display:***
This display area shows what is currently selected by the system. The menu displays an ***'X'*** if the software module is selected to load during startup. It displays **On** if the software loaded is an overlay file and displays **Off** under the Optional heading if an error message will occur during loading if the module cannot be loaded.

If you select a row on this display page and double click you will be able to edit the software loader record. Please refer to the Software Loader Record Menu on page 31 for more information on editing records.

❸ Editing Button Area - These buttons allow you to perform functions on the software loader database.

Insert Button - This button allows you to insert a record before the highlighted area on the display page. This is useful because many software modules require other modules to be loaded before they will load properly.

Append Button - This button will create a new record at the end of the database. After selection the record will appear.

Delete Button - This button will delete the currently highlighted record on the display page.

Install Button - This button will pop-up another display page to allow you to install software modules.

Shutdown Button - This button will shutdown the AIM software package. After selection of new software modules it is required to restart the AIM system for the software modules to load.

Edit Button - This button will pop-up the highlighted record on the display page to allow changes to be made. Please refer to the Software Loader Record Menu on page 31 for more information on editing records.

Save Button - This button will save the current changes to the Software Loader database.

Done Button - This button will exit the current menu display page.

3.2 Software Loader Record Menu

The software loader record page allows you to edit the software modules to be loaded.

To review or edit a record, highlight and double click on the record from the display page you wish to edit. Additionally, you can highlight a record and press the edit button.

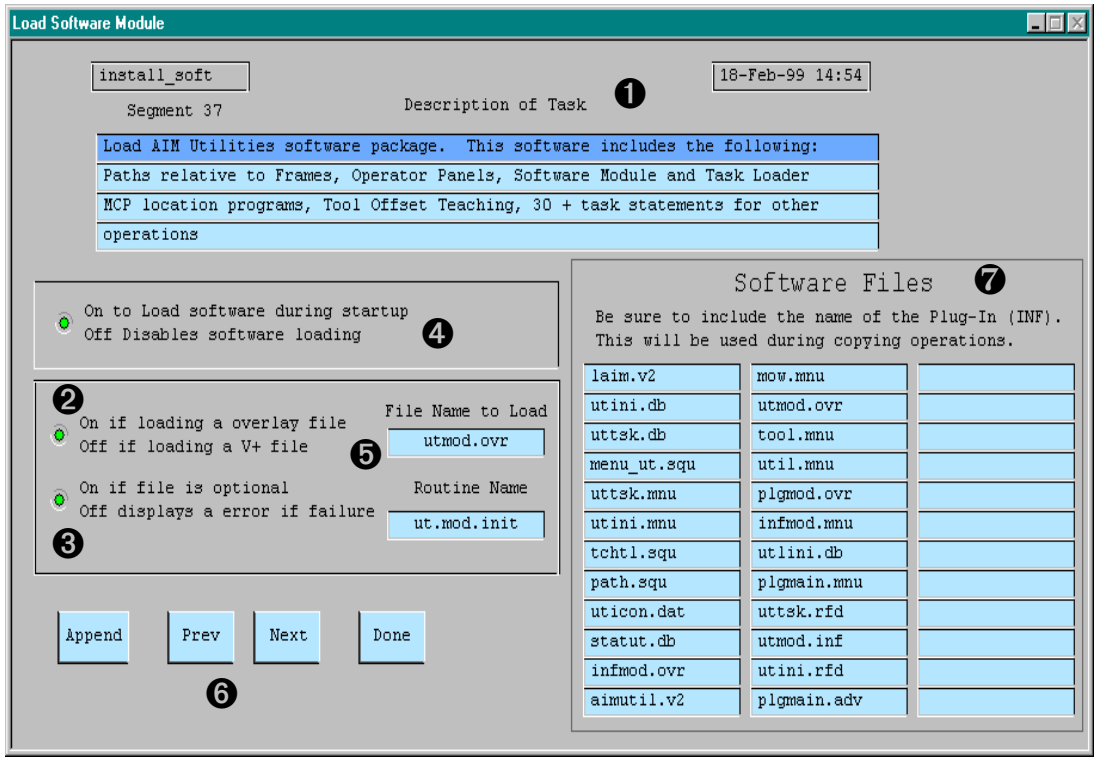


Figure 3-2 Software Loader Record

❶ The description of task is displayed on the main software loader menu page. This is provided so you can add comments or a description of the operation.

❷ The On/Off selection of loading an overlay file performs the following operations.

(ON) - The On state will load an AIM Overlay file and execute the specified program based on the supplied fields. After the program has been executed, the overlay software will be deleted from memory.

(OFF) - The Off state will load a V+ file if the specified routine does not exist in memory. The software will remain in memory after completion.

❸ The On/Off selection of the optional file performs the following operations. If selected, AIM will try to load the software module, if the file does not load no error condition will result and AIM will continue loading the system. If this button is not selected AIM will try to load the software, if there are problems, AIM will display an error message.

❹ The On/Off selection of loading the software will try to load the software module if selected. If turned off AIM will not try to load the software. This allows a record of the available software, without trying to load the software.

❺ The File Name field allows the user to enter in the name of the software file to load.

If the user should enter in ***None***, the field description will be displayed on the main menu page without the rest of the data on the line. This is useful for adding comments to the main menu page. Be sure that the file is not selected to load.

The Routine Name field is provided for overlay files and for quicker loading of V+ files. If the overlay file is selected, this routine will be executed and deleted when complete. If the overlay file in not selected, this routine name is used to check to see if the software is already loaded.

❻ The Function Button Area

Append Button - This button will allow you to add a new record to the end of the software module loader database. This record will appear after the append button is pressed.

Prev Button - This button allows the user to move between records in the database. This button will allow you to move to the previously displayed record as long as one exists.

Next Button - This button will allow the user to move to the next record in the database.

Done Button - This button will exit the current menu and return to the main menu.

❼ The Software Files selection is a list of all of the files contained in the software module. When creating a software install file this list is placed in the file. During the Install, Update and Uninstall operations this list is used for transporting or deleting files.

3.3 Task Configurator

The task configurator database is provided to allow the user to add more user tasks into the system. This provides the user an easy method to configure tasks into the AIM system without customization.

To review or edit the current software being loaded select the Setup pulldown and the Task Configurator option.

Setup ➡ Task Configurator

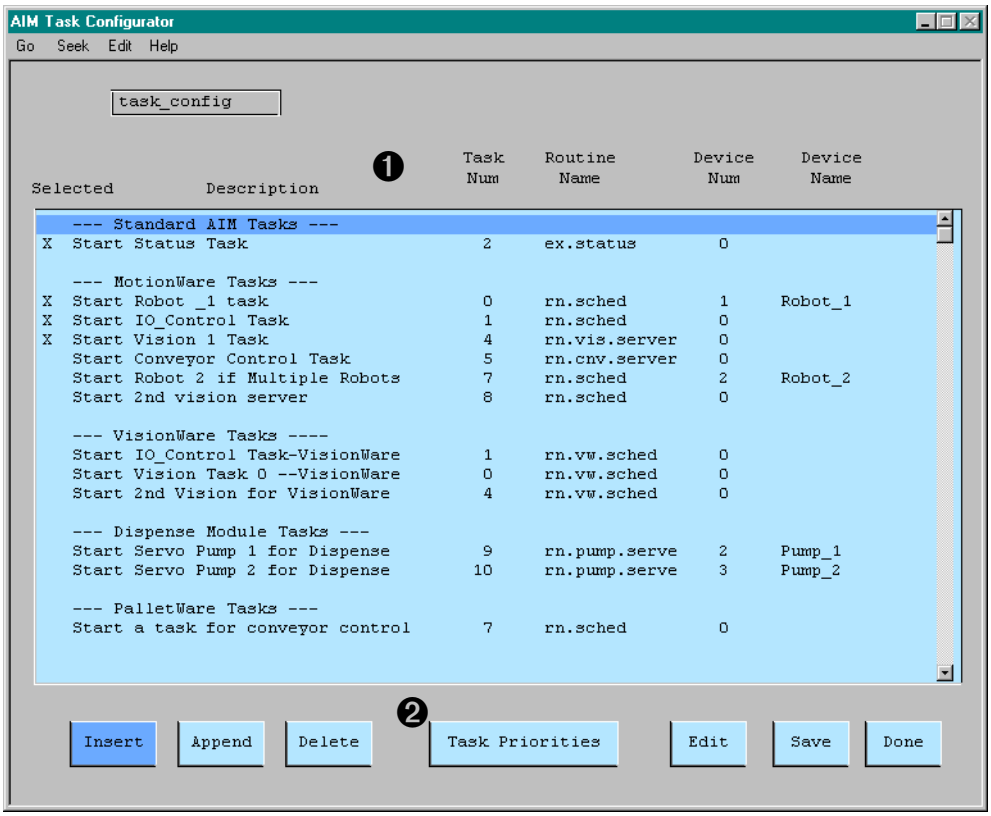


Figure 3-3 Task Configurator Main Menu

❶ Main Menu Display Area

Selected - The selected area indicates if this task is to be started during AIM bootup.

Description - This area displays the text description specified in the task record.

Task Num - This area displays the task number that is selected.

Routine Name - This area displays the routine, server or scheduler running in this task.

Device Number - This area displays the robot device that is running in this task. There will be no display if the task is not running a robot.

Device Name - This area displays the robot device name running in this task.

❷ Button Function Area:

Insert Button - This button allows you to insert a record before the highlighted area on the display page.

Append Button - This button will create a new record at the end of the database. After selection the record will appear.

Delete Button - This button will delete the currently highlighted record on the display page.

Task Priorities Button - This button will pop-up a menu display that shows the selected tasks in a list with all of the task priorities. Refer to *Task Priorities Display Page on page 37* for more details on this menu.

Edit Button - This button will pop-up the highlighted record on the display page to allow changes to be made. Please refer to the Task Configurator Record Menu on page 35 for more information on editing records.

Save Button - This button will save the current changes to the Software Loader database.

Done Button - This button will exit the current menu display page.

Task Configurator Record Menu

The task configurator record page allows you to edit the user tasks to be executed.

To review or edit a record, highlight and double click on the record from the display page you wish to edit. Additionally, you can highlight a record and press the edit button.

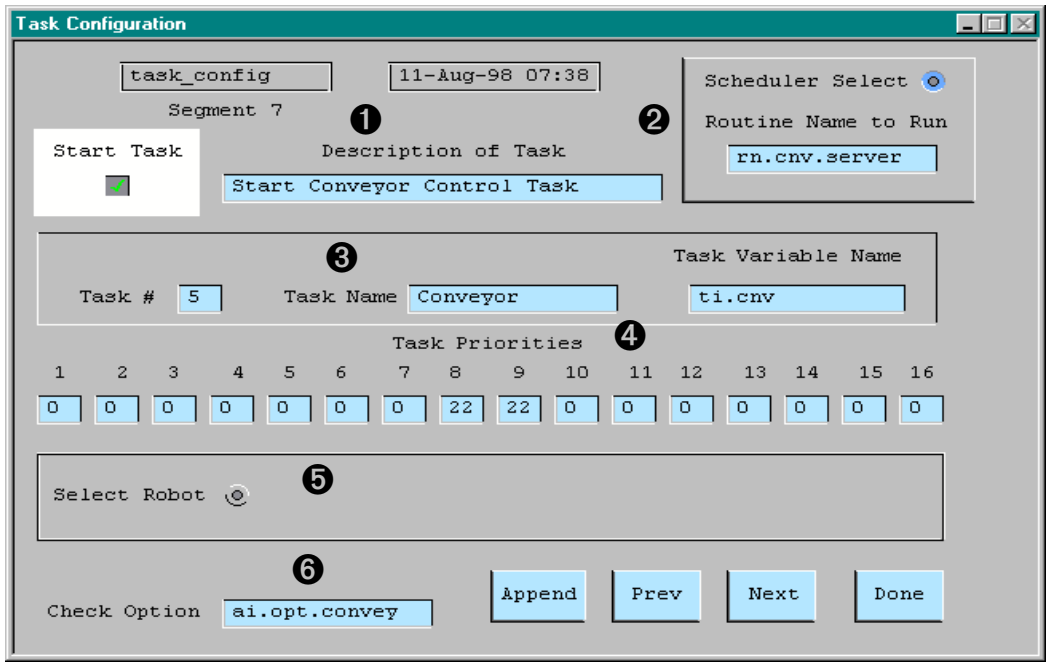


Figure 3-4 Task Configurator Record Menu

❶ The description of the task is displayed on the main software loader menu page. This is provided so you can add comments or a description of the operation.

Start Task - This check box will allow AIM to start this task during boot up of the AIM system. Many tasks a re specified in this database to perform different tasks depending on what AIM modules are selected.

❷ ***Scheduler Selection***

On/ Off button to select whether the task is running a scheduler or a server. The task is run differently based on this selection. If specified as a scheduler a sequence will be executed in the task when it is running. If a server is selected a V+ routine will run in this task and not be linked with a sequence.

The routine name to run specifies the routine that is running in the task. During startup AIM will execute this routine name in the specified task.

❸ ***Task Specification***

Task # - This selection specifies the task number your sequence or scheduler will run in.

Task Name - This field requires the user to enter a name for the task to be displayed in the AIM menus. This field is required.

Task Variable Name - This entry field will define a V+ variable which will be set to the task number if the task is running. This is used with most servers, but also can be used with other V+ routines. This field is optional.

❹ ***Task Priorities*** - This allows the user to specify the priorities used by V+ when running software. V+ has 16 different time slice periods during one V+ tick. For more information please see the *V+ Operating System User's Guide*.

❺ ***Robot Selections***

Select Robot - This radio button will specify that a robot or servo device is going to run in this task. After selection of this button, other menu fields will appear to allow you to specify other needed information.

Robot # - This entry field specifies the robot number to associate with this task. This field is a required field.

Robot Name - This entry field specifies the name of the robot which will be used by AIM. This field is required if a robot is selected.

❻ ***Check Option*** - This entry field allows the user to check if AIM options have been loaded before allowing a task to start during the AIM boot up.

Append - Allows you to add another record to the database without going back to the main menu.

Prev - Allows the user to move to previous records in this database.

Next - Allows the user to move to the following records in this database.

Done - Allows the user to exit this menu.

Task Priorities Display Page

The Task Priorities Display Page allows the user to view all of the tasks and their priorities and edit the priority values. This allows an easy way to compare what has already been setup in other tasks. The menu is placed on the monitor so the *Task Profiler* can be viewed while the menu is being displayed.

To get to this menu page simply select the Task Priorities button on the Task Configurator menu page. See Task Configurator on page 33 for more details.

The user can edit the task priorities by double clicking on the selected task on this menu.

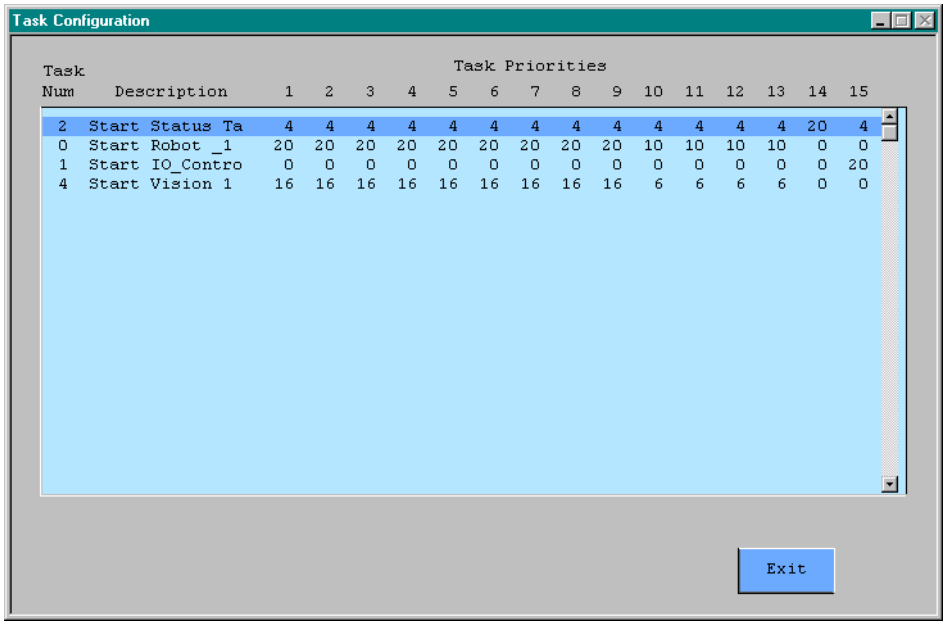


Figure 3-5 Task Priorities Display

3.4 Software Plug-In Utilities

The Aim Utilities package contains software to allow the user to create their own software module install file for easy installation into other systems as well as an easy upgrade path into future revisions of AIM. This utility software allows you to update software module backups, create a software module and uninstall an existing software module.

To get to the software plug-in utilities select the Utilities pulldown and the software install utilities option.

Utilities ➡ Software Plug-In Utilities

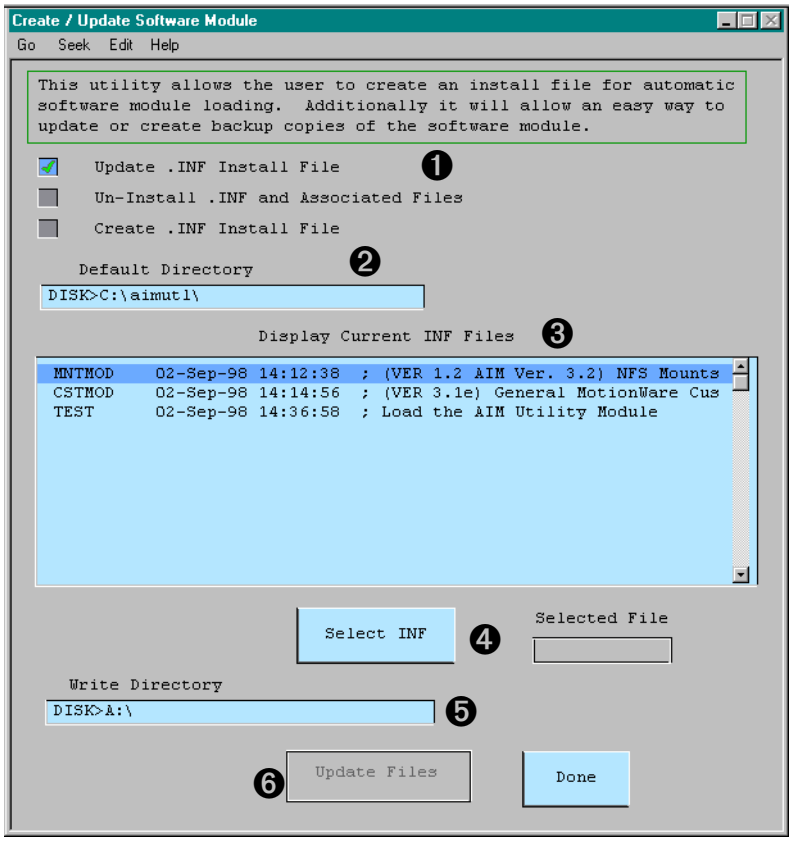


Figure 3-6
Update Software Module

❶ ***Update INF Install File***
This selection will copy the files listed in the install (INF) file to the directory selected as the write directory. This allows the user to either overwrite and update a backup disk or create a copy of the module on a disk.

❷ ***Default Directory***
This is the disk directory to search for the install (INF) files.

❸ ***Display Current INF Files***
This is a pick list for selecting the software module you wish to update.

❹ ***Select INF***
This allows you to select the current highlighted file. This can also be done by double clicking with the pointing device.

❺ ***Write Directory***
This is the directory where the software module will be copied to.

❻ ***Update Button***
This button will begin the update operation.

Uninstall Software Module

The Software Plug-in Utilities allow the user to uninstall a software module that has been loaded previously. This can be done to reduce the number of task statements or the memory used by the software module in the system. This operation will delete the software module on the main directory as well as in the software and task loader databases.

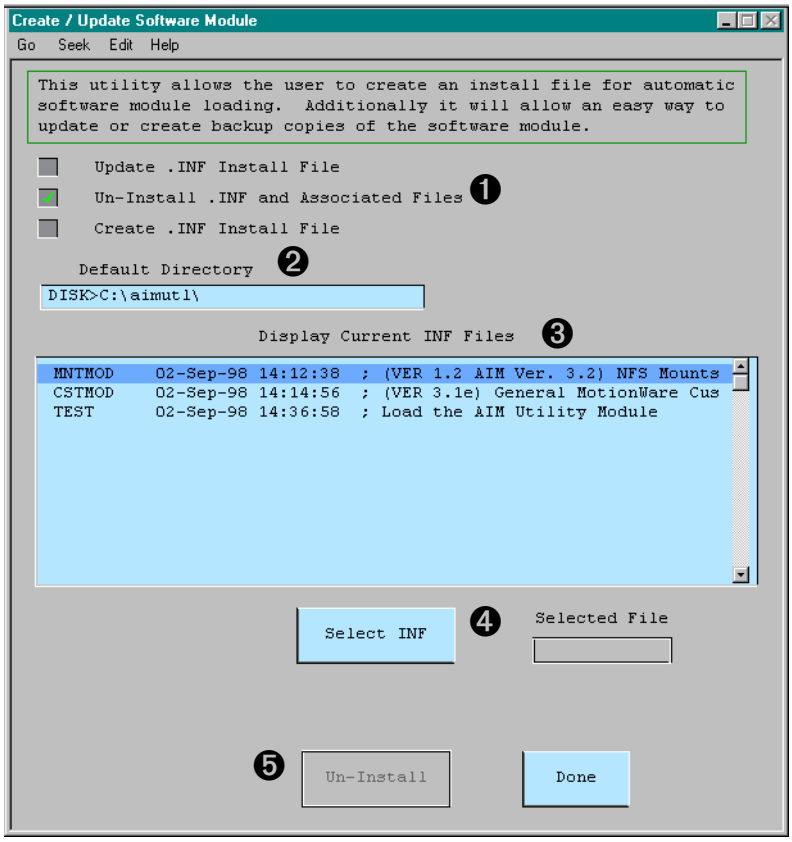


Figure 3-7 Uninstall Software Module

❶ ***Uninstall Software module***
This selection will delete the files listed in the install (INF) file in the directory selected as the default directory. This allows the user to delete all the software files as well as the records defined in the software loader and task configurator databases.

❷ ***Default Directory***
This is the disk directory to search for the install (INF) files and delete from.

❸ ***Display Current INF Files***
This is a pick list for selecting the software plug-in you wish to delete.

❹ ***Select INF***
This allows you to select the current highlighted file. This can also be done by double clicking with the pointing device.

❺ ***Uninstall Button***
Pressing this button will begin the uninstall process.

Create the Software Module Install File

This option allows the user to create an install plug-in file for easy loading of the customizations that the user may have made. This creates a file with the INF extension for easy listing of all software modules. A total of 3 software loader records and 3 task configuration records may be used for the software module. This allows the user more flexibility in the creation of their module. The four lines of description written in the first selected software loader records will be used in the description of the software INF file when displayed during the install process. The 20 fields available for software files are placed in this file for the operations listed in this section of the manual. The menus used for this operation are described below.

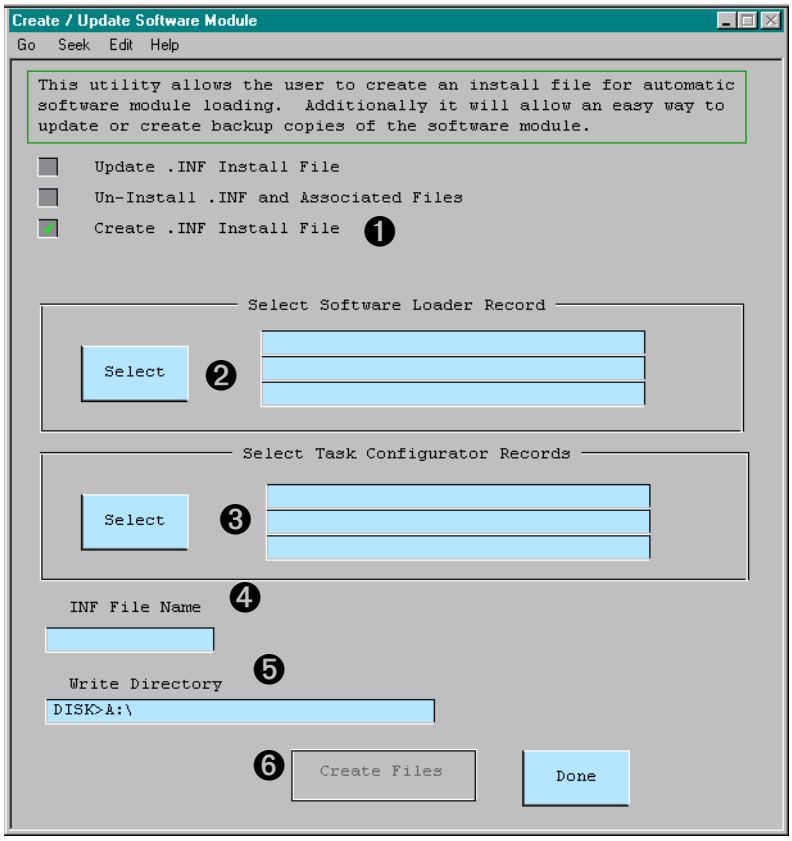


Figure 3-8 Create Software INF File

❶ ***Create Software module***
This selection will create the INF file specified by the entry data on this page.

❷ ***Select Software Loader Record Button***
By pressing this button the Software Loader menu will pop-up to allow the user to select which records will be used with the software module. It also allows the user to make changes to the records to either add more software files or more descriptions which will be added to the INF file.

❸ ***Select Task Configurator Record Button***
By pressing this button the Task Configurator menu will pop-up to allow the user to select which records will be used with the software module. It also allows the user to make changes to the records before the software module is created.

❹ ***Name for the INF File***
This name will be used for the file that will be created. This name should be less than 8 letters and begin with an Alpha character. The software will check the name entered and reduce the size or delete a numeric character if one is entered.

❺ ***Write Directory***
This field allows the user to select the directory the file will be created in.

❻ ***Create Button***
This button will begin the creation process for the file. The process is complete when the create button is not highlighted anymore.

3.5 Operator Control Panels

The AIM Utilities package contains three different operator control panels. The dispense module, palletware and a standard operator control panel are provided. Depending on the software options that are installed, the AIM Utilities loader will select the operator control panel to install.

To display the operator menu select the Execute pulldown and the Operator option.

Execute ➡ Operator

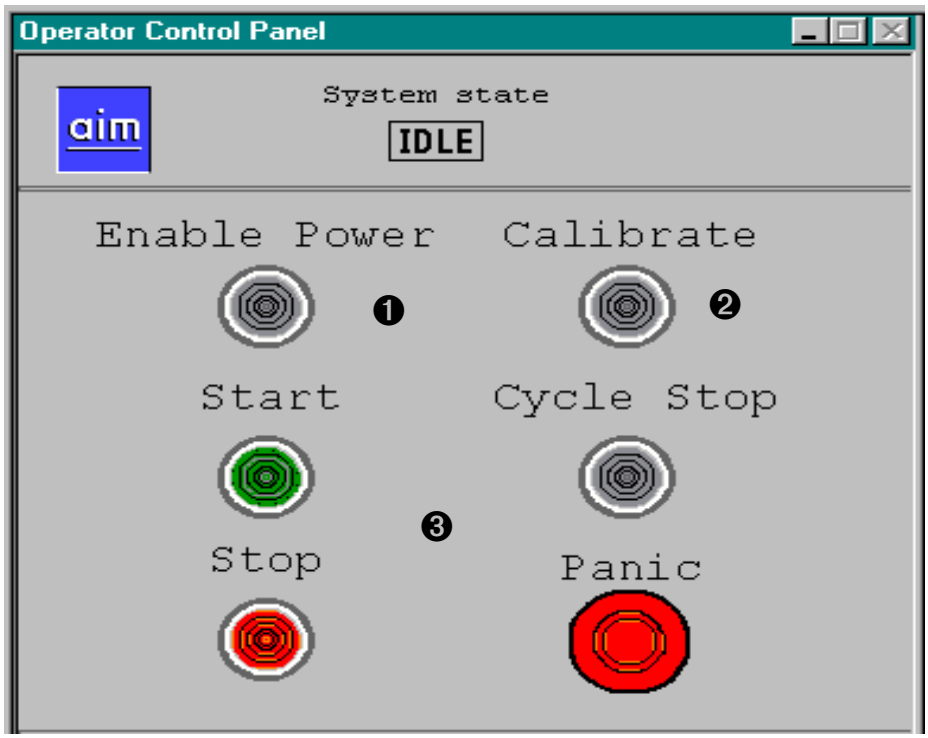


Figure 3-9 Standard Operator Control Panel

❶ The ***Enable Power*** button will try to turn power on to the robot system. Some systems may require the user to press an external button to enable. (This is in the case of a CE certified Cat 3 robot) If an error occurs during this operation an error message will be displayed.

❷ The ***Calibrate*** robot will display the current status of the robot calibration as well as allowing the user to calibrate the robot.

❸ The ***Start*** button will execute the standard control sequence called start. This has the same function as in the Master Control Panel.

The ***Stop*** button executes the stop sequence. This will generally cause a pause state to the sequence.

The ***Cycle Stop*** button executes the cycle stop sequence.

The ***Panic Button*** will halt the robot motion and disable power to the system.

3.6 Precision Point Database

The AIM Utilities package contains software to allow motion to a joint defined position rather than a world coordinate defined position. This allows the user to work better with positions that require a specified arm joint position. This is very useful with jointed arm applications. This will also be useful with applications that require full rotation of Adept arms with Joint 4.

To display the Precision Point menu select the Edit pulldown and the Precision Point option.

Edit ➡ Precision Point

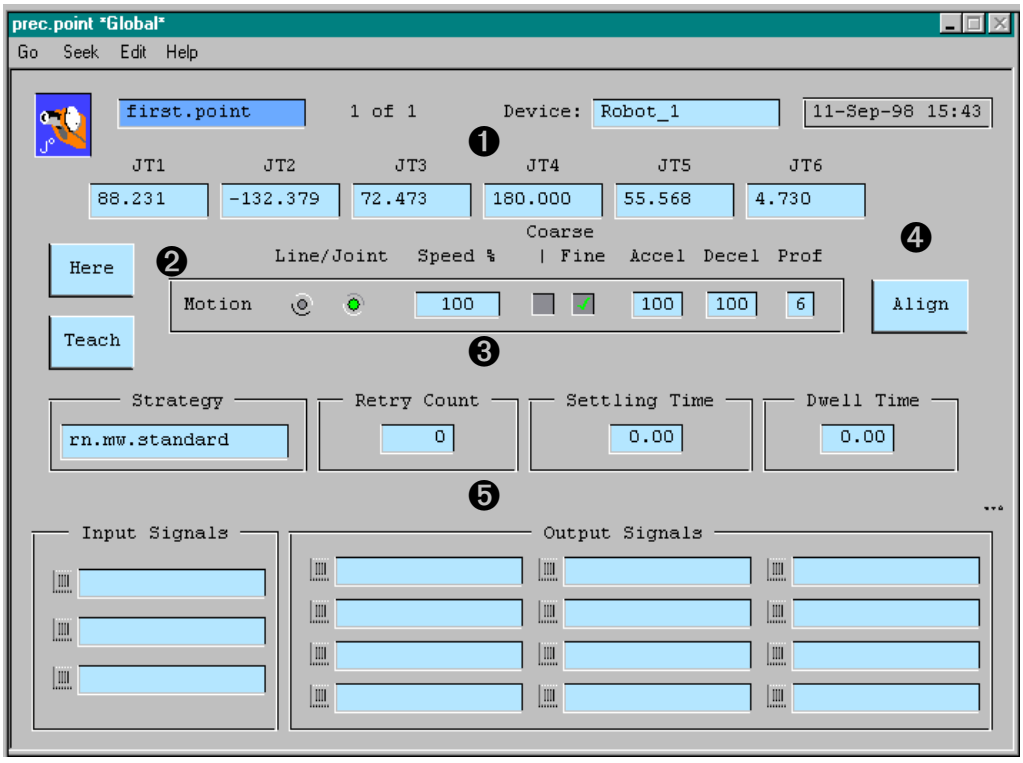


Figure 3-10 Precision Point Menu

❶ Record Name field requires a name for the jointed position.

Device: Requires the proper robot to be selected for the position to be defined.

JT1-JT6 are six fields that define the position of most robot arms supported by the Adept controller. If the robot is only a 4 axis arm then only 4 fields will be used for the position. These fields can be edited by the user or taught by using the Here or Teach buttons.

❷ The ***HERE*** button will teach the current location in the six JT fields provided.

The ***TEACH*** button will allows the user to teach the current position via Adept's manual control pendant. A pendant display will appear with selections and the speed pots can be used to move the robot to the current taught location if needed.

❸ Motion Parameters Section

Line / Joint radio button selection allows the user to define a straight line or joint interpolated motion to the defined position.

Speed selection is a percentage of maximum speed. This parameter is not limited to 100%. This value is dependent on the type of robot being used in the application.

Coarse / Fine nulling tolerance selection: This determines how close the robot must be to the taught location before the next position or process is executed. Fine is typically +_ 0.001 inches with Adept robots.

Accel / Decel specification: These parameters control the acceleration and deceleration during the robot motion to the specified location.

Prof specification: This is the acceleration and deceleration profile during a motion. This is based on the robot that is running. There are several preprogrammed profiles that will allow S curve acceleration.

❹ ***Align*** Feature: This feature will square up the wrist joints on a jointed arm (5 or 6 axis arms). This is a good aid with robot teaching.

❺ Strategy Parameter Section

Strategy Routine: This is a V+ routine that will be called once the robot reaches the specified position. This routine will handle the specified inputs and outputs as well as the settling and dwell delays that can be specified.

Retry Count: This allows the user to specify a number of retries if a failure occurs. In this case the parameter is not used.

Settling Time: This parameter allow a period of time after the robot reaches the position before any outputs occur.

Dwell Time: This parameter allows the user to specify a period of time to dwell after the outputs have been asserted before allowing the process to continue.

3.7 MCP Emulator

Provided with the AIM Utilities software is a utility to allow the user to move the robot without the use of the MCP. This menu has 2 different modes; Step and Jog mode. The keyboard arrow keys are porvided as hot keys to allow easy stepping of the X/Y axis. Many default parameters are provided in a initialization database. To view the MCP Emulator menu pages, select the Show pulldown and the MCP Emulator option.

Show ➡ MCP Emulator

The figure below is the MCP Emulator in Step Mode.

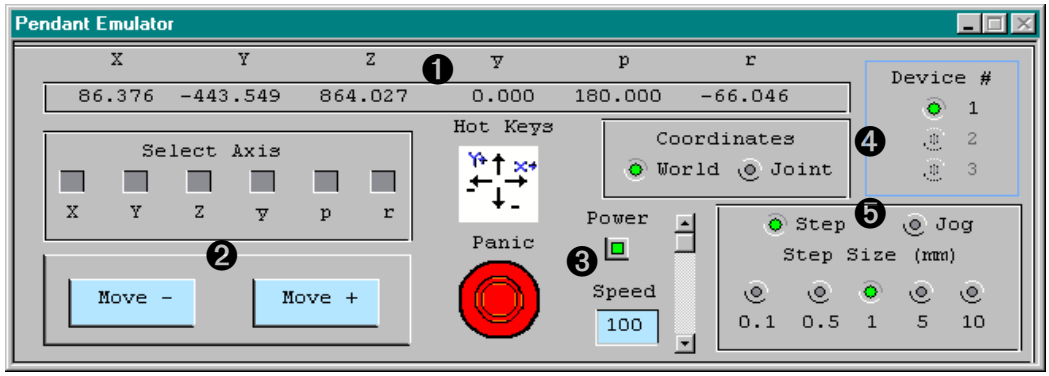


Figure 3-11 MCP Emulator (Step Mode)

❶ ***Robot Position Display:***
This area displays the current location of the robot. This display changes based on the coordinate selection. If Joint Coordinates is selected then joint positions will be displayed.

❷ ***Select Axis:***
The select axis area allows the user to select the robot axis that will move when the move buttons are pressed. The axis selections will change based on the Coordinate selection. If joint is selected then the joint numbers will be selected.

Move Buttons:
The Move +- buttons will move the robot in the positive or negative direction.

❸ ***Hot Keys:***
When the arrow icons appears in a bold state the Hot Keys are active. This will allow the user to step the robot by pressing the keyboard arrow keys. Hot Keys are only active in the step mode function.

Panic Button:
Pressing the Panic button will immediately stop the robot and disable robot power.

Power Button:
The power push button will enable or disable power depending on the current state. Power can only be turned on if all error conditions have been cleared.

Speed:
The speed field and scroll bar allow the user to adjust the speed of the motion. This speed is based on the default values specified in the initialization database.

❹ ***Coordinates:***
The MCP Emulator software allows operation in world or joint coordinates. This allows the user to move the axis along the axis or in a individual joint fashion.

Device:
The MCP Emulator software allows a maximum of 3 robots to be moved by this menu package. The user selects the desired robot by pressing on the radio button.

❺ ***Motion Mode:***
The MCP Emulator software allows motion in either a Step or Jog fashion. The radio button allows the user to select the operation. In Step mode, the robot will move the selected distance every move button press or Hot Key press. In Jog mode, the robot will move continously while the Jog +- button is pressed.

Step Distance:
The desired step distance is selected by pressing the radio button by the distance amount. Units are specified in millimeters. These buttons are only active when the software is in Step Mode.

3.8 NFS Disk Drive Mounting Utilities

Provided with the AIM Utilities software is a utility to allow the user to mount other disk drive devices thru a network. Before this software will function the user must install a NFS server on the other computer system that contains the drives. Additionally, Ethernet must function properly between both systems. To view the mount menu pages, select the Setup pulldown and the NFS Mount option.

Setup ➡ NFS Mount

The figure below is the menu that allows the mounting of other disk drives thru NFS.

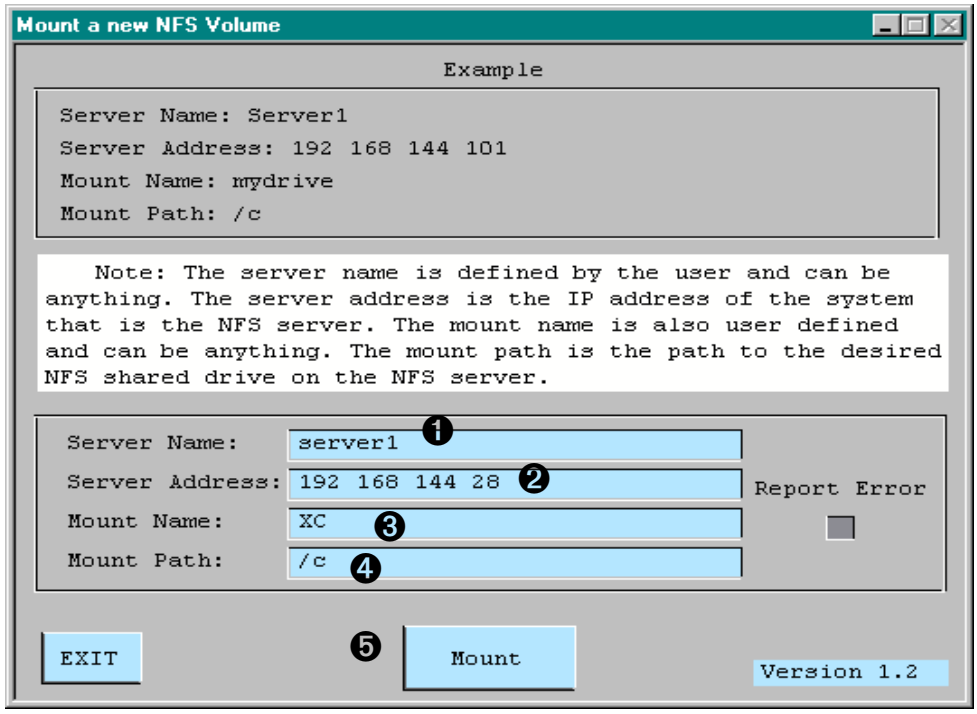


Figure 3-12 NFS Mount Menu Page

❶ ***Server Name:***
This field will label the name of the computer that will be displayed to associate that computer in other menu displays.

❷ ***Server Address:***
This is the IP ethernet address used by the server computer.

❸ ***Mount Name:***
The mount name is used by the user to specify the disk drive on the server computer. This label will be used by the user during any file operations.

❹ ***Mount Path:***
This path is the name of the Server Drive that will be used. Additionally, you can define a path on that drive to a specified directory or folder.

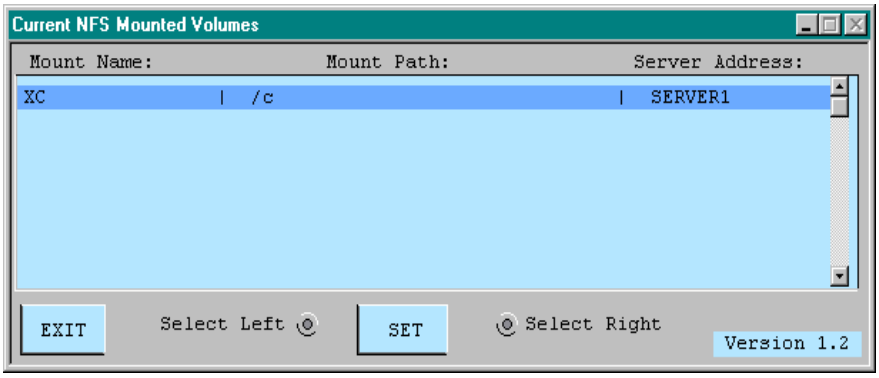
❺ ***Mount Button***
This button will perform the drive attachment based on the data specified.

NFS Mount Status Menu Page

Provided with the AIM Utilities software package is a status display page of the disk drives that are attached to the system via NFS software mounts. To display this menu page look under the SHOW pulldown menu and select the Current NFS Mounts Menu page.

SHOW ➡ Current NFS Mounts

The figure below is the menu page that displays the current drives available.



3.9 Tool Database

The tool offset database menu page has been modified to include a tool teaching utility that runs from the Adept manual control pendant. This utility allows the user to rotate the desired tool center about a common point. This rotation will allow the software to determine what the tool center is.

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*

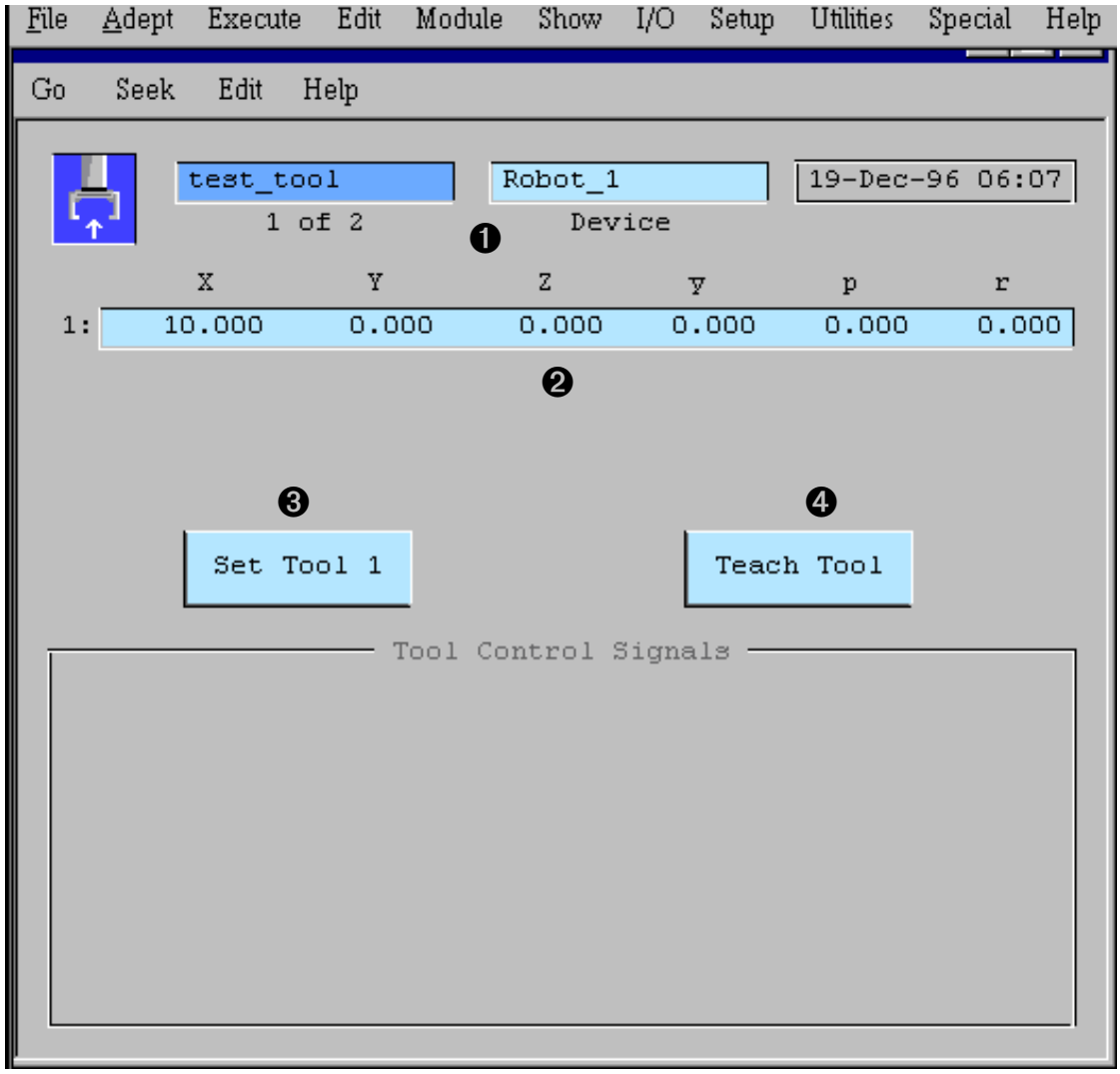


Figure 3-13 Tool Offset Menu

❶ Enter a name for this tool record. The numbers below the name field 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 offset tooling), 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 Section 4.5.

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.

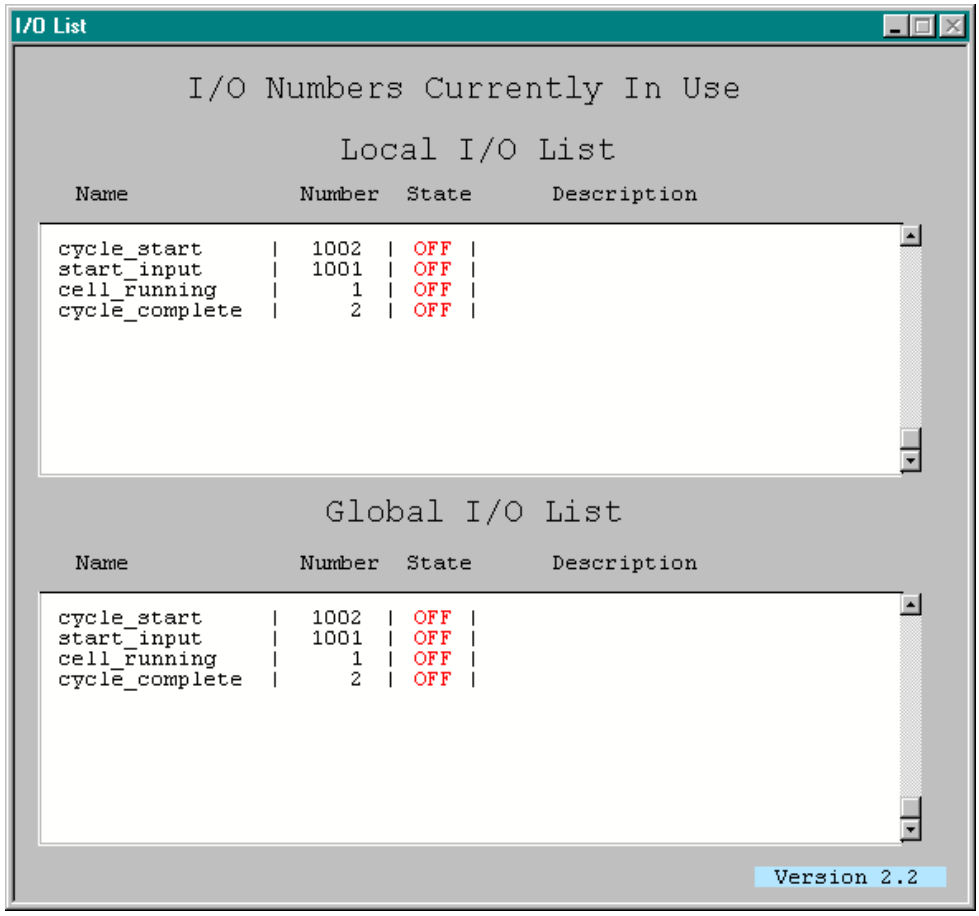
More documentation is available on tool offsets in the *MotionWare Users Guide*.

3.10 IO Status Display Page

Provided with the AIM Utilities software is a utility to display the current IO status based on the names defined in the variable database. This display shows both the global and the local database modules drive devices thru a network.

Show ➡ IO List

The figure below is the menu display of the variable names and input and output values for those IO's.



3.11 Cell Status Display Page

Provided with the AIM Utilities software is a menu screen that shows the current system status. This status menu runs in task 6 where the typical AIM status screen is shown. There is an initialization database provided for the text that appears next to the IO displays on the menu. The initialization database is called stsini.db under the initialization database selection in the pulldown menu called *SETUP*.

This screen shows the IO state and allows the outputs to be set per the display. Additionally, other system information is shown on this screen. This menu can be accessed by using the *Show* pulldown and selecting *Cell Status*.

Show ➡ Cell Status

The figure below is the Cell Status Menu.

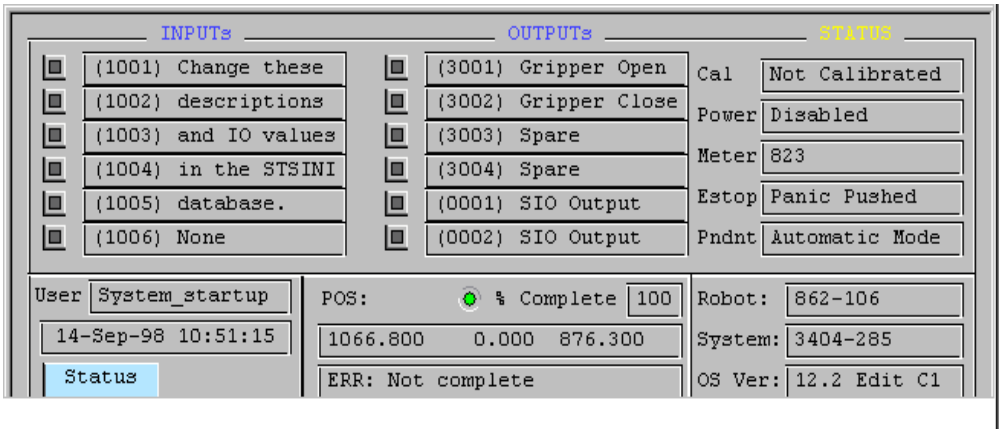


Figure 3-14 Cell Status Menu

3.12 MCP Teach Menu Changes

The AIM Utilities package has changed the MCP menus that are used by both the Location and Precision Point databases. New MCP menu selections are ***Prv Rec*** and ***Nxt Rec***. These buttons allow the user to select other records in the database from the teach pendant. The ***TOG_SEL*** allows the user to toggle thru the Approach, Location and Depart teaching when using the locations database.

This menu appears on the teach pendant when the ***TEACH*** button is pressed when editing a location or precision point record. The other buttons on the pendant remain the same as standard AIM software.

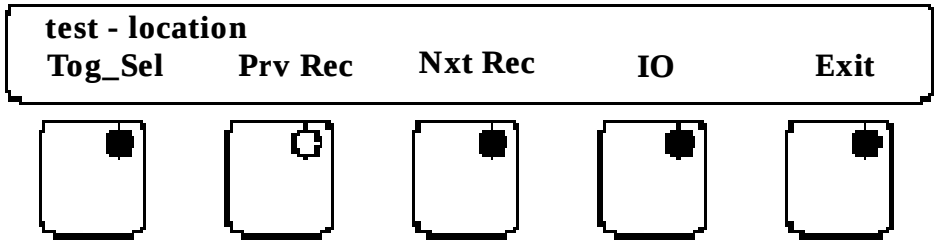


Figure 3-15 MCP Teach Buttons

4 Using AIM Utilities

4.1 AIM Utilities Statements

This chapter describes the sequence statements that are provided with the AIM Utility package. First, the statements and arguments are documented. Finally, example applications are presented to demonstrate the use of some of the statements.

Later sections show examples on how to use these statements as well as information on other available software modules provided with the AIM Utility Package.

AIO

This statement will allow the user to either read or write to an analog board. The statement's syntax is as follows, where the braces ({ . . . }) define optional clauses:

AIO {Input Channel --variable-- {Gain (0 to 3) --variable-- } To (-10 to 10) --variable-- } {Output Channel --variable-- Value (-10 to 10) --variable--}

The statement performs the following steps:

1. If the Input Channel variable is defined the statement will check the gain value and read the specified channel and place the value in the variable argument. The gain value must match the gain set by the jumpers on the AIO board. Values are 0-3
2. If the Output Channel variable is defined the statement will write to the output channel the value specified in the second argument. Please note that the value needs to be between the value of -10 to +10 to represent 10 volts.

ARM_CONFIG

This statement will return the current arm configuration of the robot. The statement's syntax is as follows, where the braces ({ . . . }) define optional clauses:

ARM_CONFIG Return Config --variable--

The statement performs the following steps:

1. This statement will return to the specified variable a value of true if the arm is in the righty configuration. Otherwise the value will be false or zero.

ATTACH_ROBOT

This statement will attach the current selected robot to program control. The statement's syntax is as follows, where the braces ({ . . . }) define optional clauses:

ATTACH_ROBOT

The statement performs the following steps:

1. This statement will perform the V+ attach statement. This is intended to be used with calibration from sequences. See the *V+ Language Reference Guide* for more details on the attach statement.

CALIBRATE

This statement will calibrate the robot from a sequence. The statement's syntax is as follows, where the braces ({ . . . }) define optional clauses:

CALIBRATE

The statement performs the following steps:

1. This statement performs a robot calibration from a sequence. Make sure robot power is on and other robot states are ready before calibration. Refer to the example sequence on calibration.

CLEAR_MESSAGE

This statement will clear a status message displayed by the Status_Message instruction. The statement's syntax is as follows, where the braces ({ . . . }) define optional clauses:

CLEAR_MESSAGE AI.CTL Array --constant--

The statement performs the following steps:

1. The *Array --constant--* argument refers to the specified array number used for the Status_Message instruction. This statement will set the ai.ctl string variable to a null string.

DETACH

This statement will detach the robot from computer control or detach the pendant. The statement's syntax is as follows, where the braces ({ . . . }) define optional clauses:

DETACH {Robot --yes/no--} {Pendant --yes/no--}

The statement performs the following steps:

1. By selecting the Robot option, the current robot running in the task will be detached from computer control.
2. By selecting the Pendant option the MCP will be detached.

DISTANCE_FROM

This statement will return the current robot distance from a specified location. The statement's syntax is as follows, where the braces ({ . . . }) define optional clauses:

DISTANCE_FROM Current POS To --location-- Return To--variable--

The statement performs the following steps:

1. The *--location--* argument is used as a reference to determine the distance in millimeters from the current robot location.
2. The actual distance is output to the *Return To --variable--* in the variable database.

GET_BITS

This statement will return a BCD value based on the current setting of a block of input signals. It also allows the user to wait for the proper input value before continuing the program. The statement's syntax is as follows, where the braces ({. . .}) define optional clauses:

GET_BITS NUM of BITS --constant-- BEGINING SIGNAL --input--

RETURN VALUE TO --variable-- {WAIT UNTIL --opr-- --variable--

TIMEOUT --variable--}

The statement performs the following steps:

1. The *NUM of BITS --constant--* argument specifies the size of the block of inputs to be used to calculate a BCD number.
2. The *Beginning Signal --input--* argument is the first signal of the block of inputs used for the calculation.
3. The *Return Value To --variable--* argument is the output variable where the BCD value is placed.
4. The Optional *Wait Until --opr-- --variable--* argument can be used to wait until a comparison value occurs. The *--opr--* value allows you to specify the type of comparison operation.
5. The *Timeout* argument allows the user to specify a maximum amount of time to wait for the comparison value to occur.

GET_PARM

This statement will return the specified V+ parameter value to a variable. The statement's syntax is as follows, where the braces ({. . .}) define optional clauses:

GET_PARM --parameters-- RETURN TO --variable--

The statement performs the following steps:

1. The *--parameters--* argument provides a pick list of the available parameters that can be selected. The V+ parameters are various system related values that can be returned to a variable.

   Provided below is a listing of the available parameters that can be used with the Get_Parm and Set_Parm statements. Please refer to the *V^+ Operating System Reference Guide* for more information on parameters.

   belt.mode **-** Controls the operation of conveyor tracking.

   belt.zero.count**-** Zero index checking for the conveyor encoder

   display.camera**-** Display the selected camera number on the vision screen

   hand.time**-** Delay timed used by V+ when using OpenI and CloseI Instructions

   kermit.retry **-** Controls the number of Kermit retries with the RS-232 communication.

   kermit.timeout **-** The amount of time before the Kermit connection is failed

   screen.timeout **-** Time after last terminal use for screen blanking

terminal - Returns the current serial port number for the terminal

not.calibrated **-** Calibration status of the robot devices

2. The -*variable*- argument allows the current value of the parameter to be passed to that specified variable.

GET_STATE

This statement will return the current status of the system states. This is based on the V+ *State* instruction. The statement's syntax is as follows, where the braces (`{...}`) define optional clauses:

GET_STATE --states-- RETURN TO --variable--

The statement performs the following steps:

1. The --*states*-- argument provides a pick list of the available system states that can be selected. The V+ states are various system related functions that can be returned to a variable.

   Provided below is a listing of the available state conditions that can be used with the Get_State statement. Please refer to the *V+ Operating System Reference Guide* for more information on parameters.

   panic.button - Returns the status of the system E-Stop Button

   high.power **-** Returns the status of the robot high power

   program.start **-** Returns the status of the Program Start Button

2. The -*variable*- argument allows the current value of the system state to be passed to that specified variable. The variable will be returned with a true or false state.

GET_SWITCH

This statement will return the V+ system switches settings. The statement's syntax is as follows, where the braces (`{...}`) define optional clauses:

GET_SWITCH --switch-- RETURN VALUE TO --variable--

The statement performs the following steps:

1. The --*switch*-- argument provides a list of selectable V+ switches that indicate system settings.

   Below is a list of the supported switches with a description of their purpose.Please refer to *V+ Operating System Reference Guide* for more information on system switches.

   auto.power.off - Stops V+ from disabling High Power when certain motion errors occur. Please see the V+ reference guides for more information.

   belt - Belt Tracking feature is On or Off

   cp - Robot Continuous Path motion is On or Off

   decel.100 **-** Allows the user to set the deceleration in the Accel V+ instruction.

   dry.run - V+ Dry Run Feature -- runs a robot task without motion

interactive - Suppresses interactive messages displayed on the terminal

mcp.message - Suppresses messages displayed on the MCP

mcs.message - Suppresses messages displayed by the MCS instruction

monitors - Allows multiple V+ monitors if multiple processes are used

power - Robot Power is On or Off

profile - Default acceleration profile the robot is running

retry - Controls if a resume will occur if the program start button is pressed

robot[1] - Enables robot 1 to function

robot[2] - Enables robot 2 to function

robot[3] - Enables robot 3 to function

robot[4] - Enables robot 4 to function

scale.accel[1] - allows V+ to automatically scale the acceleration relative to velocity

scale.accel[2] - Same as above except for robot 2

scale.accel[3] - Same as above except for robot 3

scale.accel[4] - Same as above except for robot 4

scale.accel.rot[1] - Allows V+ to scale the acceleration for rotary axis (J4)

scale.accel.rot[2] - Same as above except for robot 2

scale.accel.rot[3] - Same as above except for robot 3

scale.accel.rot[4] - Same as above except for robot 4

set.speed - Enables the changing of the monitor speed from the MCP

trace - Selects the V+ trace options which will display the program statement as the program executes

upper - Determines if string comparisons are case sensitive.

vision - Enables the Adept Vision System to function

2. The -*variable*- argument allows the current value of the system state to be passed to that specified variable. The variable will be returned with a true or false state.

GET_TIME

This statement will return the current timer value based on the V+ timer selected. The statement's syntax is as follows, where the braces ({ . . . }) define optional clauses:

GET_TIME Timer Number --constant-- Push Time --variable--

The statement performs the following steps:

1. The --constant-- argument refers the V+ timer value. V+ has a total of 16 timers as well as special timer modes that are a negative value. Please refer to the *V+ Language Reference Guide* for more details.
2. The --variable-- argument will return a time value to the AIM variable database.

J4_INERTIA

This statement will set the V+ Gain.Set and Payload instructions based on the values specified in the statement. These values will change the tuning for the Joint 4 of an Adept robot to compensate for inertia payloads on the tool mounting flange. Please see the *V+ Language Reference Guide* and the *Robot User's Guide* for more information on the values needing to be specified in this statement.The statement's syntax is as follows, where the braces ({ . . . }) define optional clauses:

J4_INERTIA {Gain.Set (1-4) --variable--} {Payload (1-100) --variable--}

The statement performs the following steps:

1. If the Gain.Set variable is specified, that value will be executed in a V+ gain.set instruction. The values will be different depending on the model of Adept robot that is running. (See the *Robot User's Guide* for more information on gain values.)
2. If the Payload variable is specified, that value will be executed in a V+ payload instruction. The values will be different depending on the model of Adept robot and how much payload is on the tool mounting flange.

MCP_CHECK

This statement will return a value, 1 to 5, to correspond to the button pressed on the MCP. The statement's syntax is as follows, where the braces ({ . . . }) define optional clauses:

MCP_CHECK BUTTON # (1-5) --variable-- {Detach Pendant --yes/no--}

The statement performs the following steps:

1. The --variable-- argument above allows the user to determine which of the 5 function buttons were pressed. A value 1 thru 5 will be returned which can be handled from the AIM sequence.
2. The *Detach Pendant* selection allows the user to detach the pendant after completion of the task. Keeping the pendant attached will prohibit selection of the pendant from other V+ tasks.

MCP_WRITE

This statement will write to the MCP the string data that will come from the variable database. The statement's syntax is as follows, where the braces ({ . . . }) define optional clauses:

MCP_WRITE BUTTON 1 --string-- BUTTON 2 --string-- BUTTON 3 --string--
BUTTON 4 --string-- BUTTON 5 --string-- {Line 1 Text --string_var--
{Line Counter --variable-- }} {Detach Pendant --yes/no--}

The statement performs the following steps:

1. The --string-- argument will write to the MCP function button centered above the button number specified. This allows the user to create a sequence that can use the MCP as an operator interface.
2. The *Line 1 Text* option allows the user to display a text string on the first line on the LCD display on the teach pendant. This text string is read from the variable database.

3. The *Line Counter* argument allows the user to specify a number value that will be appended to the first line string displayed on the pendant.
4. The *Detach Pendant* selection allows the user to detach the pendant after completion of the task. Keeping the pendant attached will prohibit selection of the pendant from other V+ tasks.

MOUNT_NFS

This statement will allow the user to mount another drive or memory source to be used with the Adept controller via NFS. This statement allows the user to mount a drive via an AIM sequence. This may be useful in a startup sequence. The statement's syntax is as follows, where the braces (`{...}`) define optional clauses:

MOUNT_NFS Server Name --string_var-- Server Address --string_var--
Mount Name --string_var-- Mount Path --string_var--

The statement performs the following steps:

1. This statement will mount another computer's disk drive or memory source via NFS. It basically performs the equivalent of the V+ monitor instruction FSET to mount the drive. The *Server Name* argument is used to specify the node or name of the computer you are attached to. Please note that you can have as many as nine attachments to other sources, which includes multiple computers.
2. The *Server Address* argument specifies the IP address of the computer that is running on ethernet. Please check you computer for this IP address. Please make sure the address is not duplicated on another computer.
3. The *Mount Name* argument specifies the drive name you will be using within the Adept controller. A example is that the ***D:*** is the flash ram on the Adept AWC. The name, for example, may be *PCA* for pc computer drive A.
4. The Mount Path argument specifies the path to the other computers folders. This path may include other subdirectories or folders within the computer. The following is an example:
   \A --- for the A drive on the other computer
   \C --- for the C drive
   \C:\ADEPT\ --- for the Adept directory on the C drive

MOVE_UT

This statement is similar to the AIM *MOVE* statement, except it allows the user to specify if the Single or Multiple motion instructions are applied to the move. The statement's syntax is as follows, where the braces ({ . . . }) define optional clauses:

MOVE_UT {{APPROACH --path--} FROM --location--} {ALONG --path--}
TO --location-- {DEPART --path--} {USING --tool--}
{{ALONG --path--} REJECT --location-- {RETURN --path--}}
{WAIT_AT --location--} {OK_SIGNAL --o_variable--}
{Single@Appro --yes/no--} {Single@Move --yes/no--}

The statement performs the following steps:

1. The *APPROACH* argument specifies a controlled path motion to the location specified in the *FROM* argument. This is an optional argument.
2. The *FROM* argument allows the user to move to a specified location. This argument is optional.
3. The *ALONG* argument allows the user to specify a controlled path motion to the location specified in the *TO* argument.
4. The *TO* argument specifies the location the robot is moving to. This is the only required argument in this statement.
5. The *Depart* argument allows the user to specify an exit path away form the specified location.
6. The *USING* argument allows the user to specify a tool offset that will be applied during the motions.
7. The *ALONG* path argument allows the user to move along a path to the selected location.
8. The *REJECT* argument allows the user to specify a location for the robot to move should a failure occur. Please refer to the *MotionWare User's Guide* for more information.
9. The *RETURN* argument allows a path to be taught away from the reject location.
10. The WAIT_AT argument is a location where the robot will wait at in the case of a moving frame that is out of reach.
11. The OK_SIGNAL argument is a signal that gets set if the motion is successful.
12. The SINGLE@APPRO if set to yes will force the Joint 4 motion to remain within +_ 180 degrees relative to the zero position of the joint. This will force the joint to "unwind" during the motion to the position. This occurs during the motion to the approach location. This is sometimes desired if the motion is occurring with line tracking to prevent an unwind condition during tracking. If not defined it is set to the multiple selection.
13. The SINGLE@MOVE if set to yes will force the Joint 4 motion to remain within +_ 180 degrees relative to the zero position of the joint. This will force the joint to "unwind" during the motion to the position. This occurs during the motion to the final location. If not defined it is set to the multiple selection.

OPEN_MENU

This statement will allow the user to open an AIM menu from a sequence. A common use would be opening a menu during a startup sequence. The statement's syntax is as follows, where the braces ({. . .}) define optional clauses:

OPEN_MENU Page Name --string_var-- File Name --string_var--

The statement performs the following steps:

1. The Page Name --string_var-- argument specifies the menu page name to be opened by this statement.
2. The File Name --string_var- argument specifies the Menu File to be opened by this statement.

PMOVE

This statement will allow the user to move to a precision point location. The location is defined in joint coordinates rather than world coordinates. This is very useful when using jointed robot arms. The statement's syntax is as follows, where the braces ({. . .}) define optional clauses:

PMOVE {FROM --ppoint-- } TO --ppoint--

The statement performs the following steps:

1. The *FROM* argument allows the user to move to a precision point location and then to a second precision point location from a single statement. This is an optional argument.
2. The *TO* argument will move to a joint defined location based on the specified precision point record in the database.

RETRACT_Z

This statement will move the Z axis of the robot to a specified height. The statement's syntax is as follows, where the braces ({. . .}) define optional clauses:

RETRACT_Z Robot Height --constant-- {Speed --constant--} {Accel --constant--} {Decel --constant--}

The statement performs the following steps:

1. The Robot Height --constant-- argument allows the user to specify the absolute world coordinate Z height for the robot to move.
2. The Speed, Accel and Decel arguments specify the motion parameters for the motion.

RETURN_LOC

This statement will return the current world coordinate location of the robot. The statement's syntax is as follows, where the braces ({ . . . }) define optional clauses:

RETURN_LOC {X Pos --variable--} {Y Pos --variable--} {Z Pos --variable--}
{Yaw Pos --variable--} {Pitch Pos --variable--} {Roll Pos --variable--}

The statement performs the following steps:

1. The --variable-- arguments are all optional and will return the world coordinate value based on the robot's current position. Coordinates are returned in units of millimeters or degrees.

RETURN_JTS

This statement will return the current location of the robot in joint coordinates. The statement's syntax is as follows, where the braces ({ . . . }) define optional clauses:

RETURN_JTS {JT 1 --variable--} {JT 2 --variable--} {JT 3 --variable--}
{JT 4 --variable--} {JT 5 --variable--} {JT 6 --variable--}

The statement performs the following steps:

1. The --variable-- arguments are all optional and will return the Joint coordinate value based on the robot's current position. Coordinates are returned in units of millimeters or degrees.

SET_BITS

This statement will turn on a block of outputs based on the provided BCD number. The statement's syntax is as follows, where the braces ({ . . . }) define optional clauses:

SET_BITS Num of Bits --constant-- Begining Signal --output--
Set Value To --variable--

The statement performs the following steps:

1. The *Num of Bits --variable--* argument will specify the number of output signals used to represent the provided BCD number.
2. The *Beginning Signal --output--* argument specifies the first signal number of the block of outputs to be used.
3. The *Set Value To --variable--* argument is the BCD value the outputs will be set to.

SET_OPRMODES

This statement allows the user to set internal AIM settings that are normally set by the Task Control panel or the initialization database. The statement's syntax is as follows, where the braces ({ . . . }) define optional clauses:

SET_OPRMODES TASK --constant-- TYPE --oprmodes-- SETTING --variable--

The statement performs the following steps:

1. The *TASK --constant--* argument specifies the task to which the settings will apply. This allows the user to configure different values for different sequences.
2. The *TYPE --oprmodes--* argument allows the user to select from a pick list different AIM selections that can occur. Currently the only selections that are available are the settings from the Task Control Panel. The following is a list of the selections.

   SPEED - Refers to the robot speed that is set in the *Task Control Panel*

   REPEAT -Refers to the setting of the repeat selection box in the *Task Control Pane*l.
3. The *SETTING --variable--* argument sets the selected argument to the specified value. The repeat selection will run continuously if set to 0.

SET_PARM

This statement will set the V+ parameters to the specified value in the statement. The statement's syntax is as follows, where the braces ({ . . . }) define optional clauses:

SET_PARM --parameters-- EQUAL TO --variable--

The statement performs the following steps:

1. The --parameters-- argument provides a pick list of the available V+ parameters that this statement will change. Please refer to GET_PARM on page 58 for more details on the available pick list options.
2. The *EQUAL TO --variable--* will set the selected V+ parameter to the specified value.

SET_PATH

This statement will allow the user to modify a path segment position by the coordinates specified. This allows the user to modify a position based on variable data. The statement's syntax is as follows, where the braces ({ . . . }) define optional clauses:

SET_PATH --path-- SEGMENT --variable--
{ X: --variable--} {Y: --variable--} (Z: --variable--} {y: -variable--}
{p: --variable--} {r: --variable--}

The statement performs the following steps:

1. The *--path--* argument specifies the AIM path that the position will be applied toward.
2. The *SEGMENT* argument specifys which path node will be changed.
3. The arguments (X,Y,Z,y,p,r) represent the transformation for the new position.

SET_SWITCH

This statement will enable or disable the specified V+ switch selected. The statement's syntax is as follows, where the braces ({ . . . }) define optional clauses:

SET_SWITCH --switch-- EQUAL TO --on/off--

The statement performs the following steps:

1. The --switch-- argument provides a pick list of the available V+ system switches that can be set. Please refer to GET_SWITCH on page 59 for more details on the available V+ switches.
2. The *EQUAL TO --on/ off*-- argument will enable or disable the selected V+ switch.

SHIFT_MOVE

This statement will allow the user to move to a location offset by X,Y,Z,T dimensional data. This allows the user to modify a position based on variable data. The statement's syntax is as follows, where the braces ({ . . . }) define optional clauses:

SHIFT_MOVE World_Loc --location-- By X: --variable-- Y: --variable
Z: --variable-- T: -variable-- Along Tool --yes/no-- {Appro_Only --yes/no-- }
{OK_SIGNAL --o_variable--}

The statement performs the following steps:

1. The *World_Loc* argument specifies the location that the offset will be applied toward.
2. The arguments (X,Y,Z,T) represent the amount the location will be offset based on the axis or coordinate specified.
3. The *Along Tool* argument allows the motion to be applied to tool coordinates rather than world coordinates.
4. The *Appro_Only* argument if set to yes will only move to the approach location and bypass the other motions in the record specification.
5. The *OK_SIGNAL* argument will turn on the signal if everything is successful in the specified motion.

STATUS_MESSAGE

This statement will post to a specified $ai.ctl array variable the specified string variable. This allows the developer to have a separate status message display on a menu. The message can be set from various sequences. The statement's syntax is as follows, where the braces ({ . . . }) define optional clauses:

STATUS_MESSAGE AI.CTL Array --constant-- Message String --string_var-- {Wait For --constant--}

The statement performs the following steps:

1. The *AI.CTL Array --constant--* argument refers to a user specified array number that can be used by an AIM menu. We recommend using values greater than 200, because AIM uses lower array values for other messages that may be displayed. Several message displays can be used with this statement.
2. The *Message String --string_var--* argument will take text from the variable database based on the variable specified.
3. The *Wait For --constant--* argument will wait for the user specified input signal to go to a true state before continuing. The input signal specified can be a soft signal (2000-2512) if desired.

TASK_MODE

This statement will return the current state of the AIM sequence based on the task specified. The statement's syntax is as follows, where the braces ({ . . . }) define optional clauses:

TASK_MODE TASK --constant-- MODE --variable-- {WAIT UNTIL --variable--}

The statement performs the following steps:

1. The *TASK --constant--* argument specifies the desired AIM task number to determine the current status. You can look at the Task Control Panel to verify the task number for the sequence if needed.
2. The MODE --variable-- argument will return to the specified variable database the following values indicating the state of the AIM task.

   0 - Sequence is currently Running

   1 - Sequence is in Teach mode

   2 - Operator Attention has occurred

   3 - Sequence is Idle

   4 - Group Wait
3. The *WAIT UNTIL --variable--* argument will allow the statement to wait until the specified task state has been reached. The expected state is specified in the variable field provided.

WAIT_FOR_IO

This statement will look for the specified input state and return the amount of time it took to achieve this state as well as allowing a time-out period. The statement's syntax is as follows, where the braces ({ . . . }) define optional clauses:

WAIT_FOR_IO INPUT --constant-- Timer Num --constant--
{Time to State --variable--} {TIME OUT --constant--}

The statement performs the following steps:

1. The *INPUT --constant--* argument specifies a digital I/O number or software output signal.
2. The *Timer Num --constant--* allows the user to specify a V+ timer number to be used during the wait period.
3. The *Time to State --variable--* returns to the variable database the amount of time it took for the specified I/O to achieve a true state.
4. The *TIME OUT --constant--* argument allows the statement to wait a maximum amount of time in this statement. The sequence can monitor the Time to State variable to indicate the maximum time to achieve the desired state.

TYPE_MON

This statement will type text to the V+ monitor based on the argument supplied. The statement's syntax is as follows, where the braces ({ . . . }) define optional clauses:

TYPE_MON String --string-- VAR --variable-- {String --string-- VAR --variable--}

The statement performs the following steps:

1. The String --string-- arguments allow the user to provide a text string to be typed on the V+ monitor.
2. The VAR --variable-- allows the user to type the current variable value from the variable database to the V+ monitor.

4.2 Calibration from Sequences

The following sequence is an example of calibrating from a control sequence. This sequence is provided in the example sequence supplied with this software package.

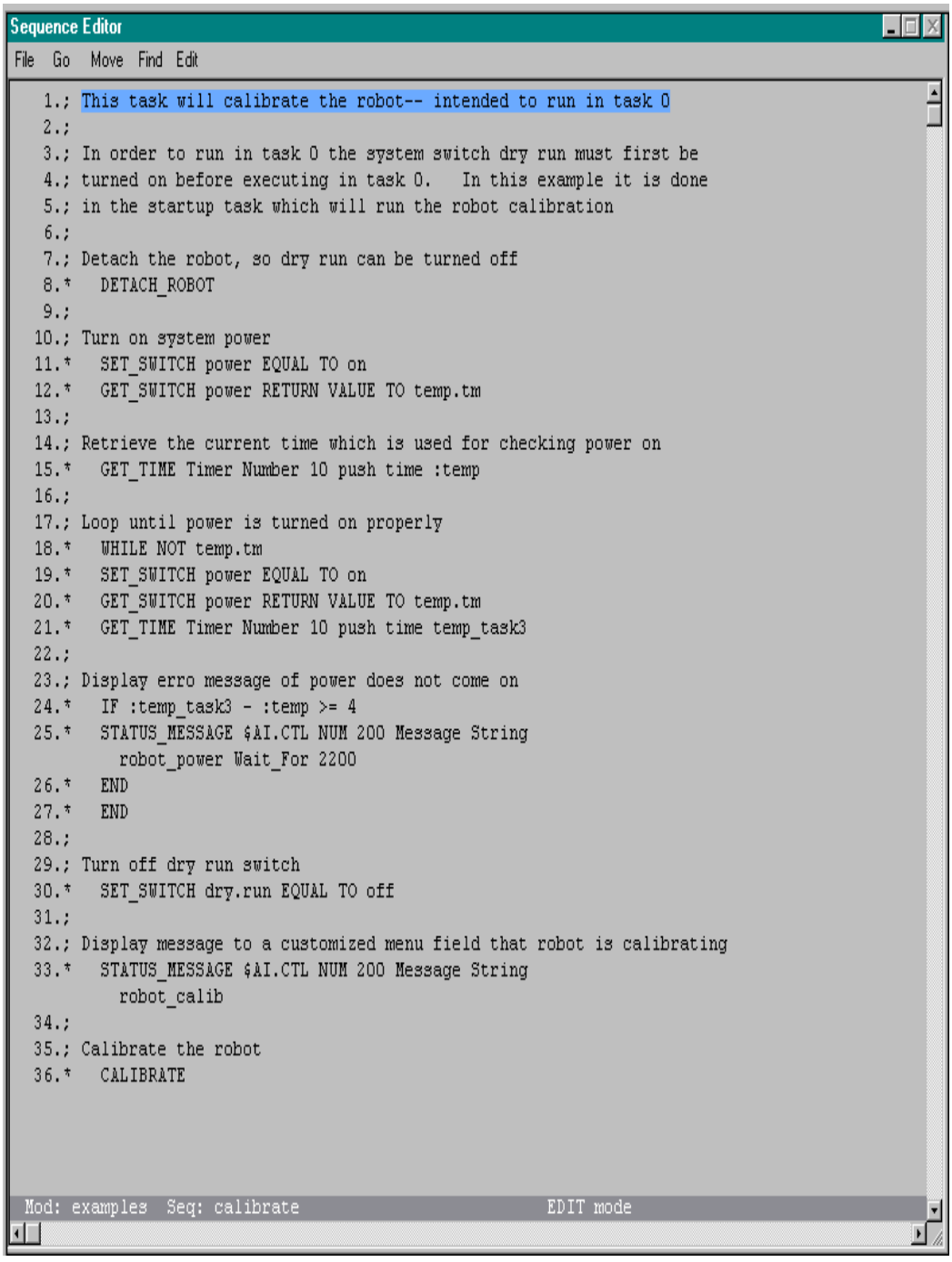


Figure 4-1 Calibration Sequence

4.3 Using the Message Statements

The message statements are provided for a controlled method of displaying messages from a menu display. The statements are STATUS_MESSAGE and CLEAR_MESSAGE. Many AIM installations have error recovery by the use of a sequence and therefore many of the AIM messages that would normally be used by the operators can be ignored. This allows the sequence to display a message to a developed menu via the AIM $ai.ctl array values. This instruction basically transfers a predefined variable string from the variables database to the $ai.ctl variable. This variable can then be displayed on a menu screen. Additionally a software signal can be specified as an acknowledgment that the error occurred.

4.4 MCP example

The MCP sequence statements MCP_WRITE and MCP_CHECK are provided to allow users to develop AIM sequences that can control an interface for use with Adept's MCP. These statements allow the user to write messages to the MCP and check for the function button presses. Below is a sequence that was developed for a customer that wanted to change the pallet rows and columns from the teach pendant. This sequence can be found on the AIM Utilities Database Examples disk.

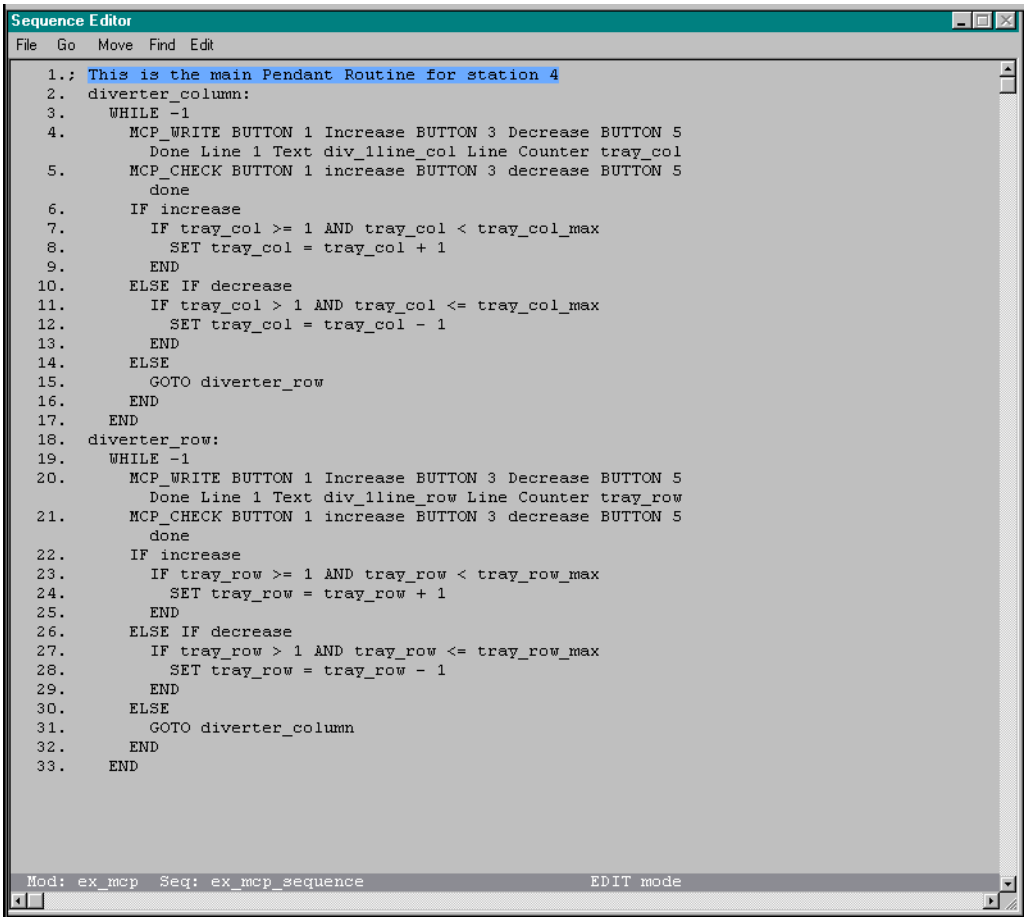


Figure 4-2MCP Sequence

4.5 Tool Offset Teaching

Figure 4-3 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.

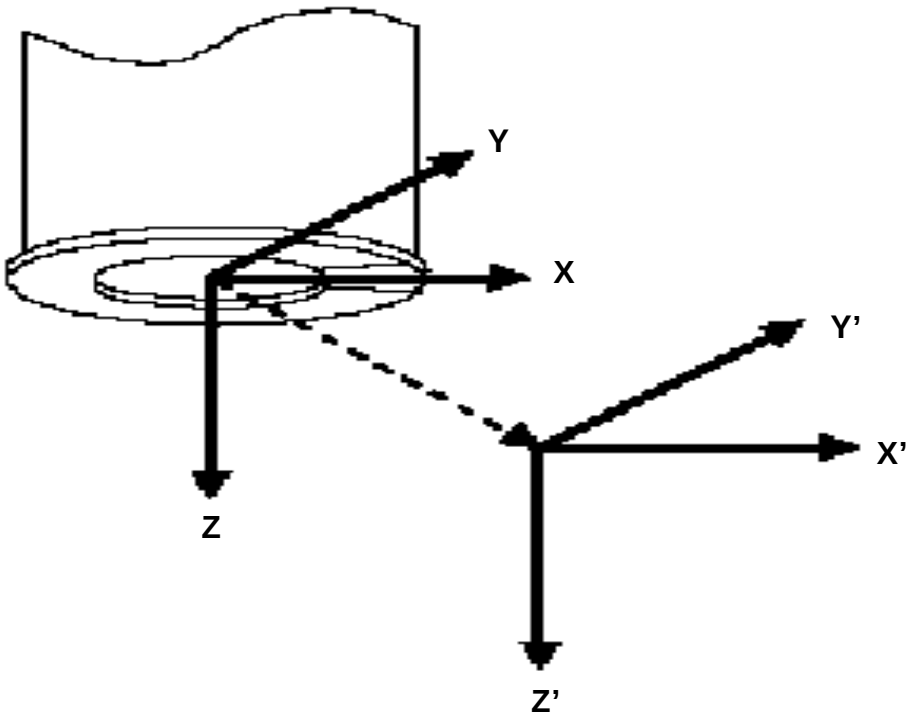


Figure 4-3 Cartesian Tool Offset

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

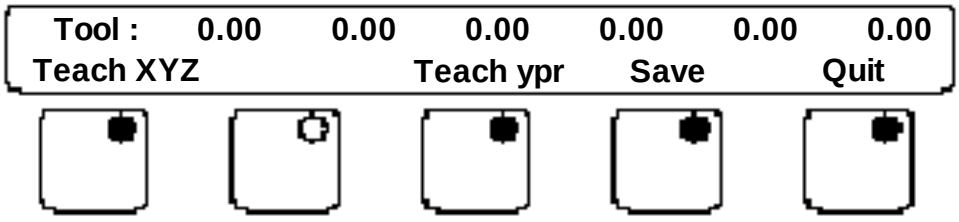


Figure 4-4 Tool Offset Utility Menu Display

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 tooling center tip at a given location and press the pendant REC/DONE key. Rotate the 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 are obtained if more locations are taught. (The robot 'FREE' mode can be used to simplify the teaching process.)
3. Once the locations are taught, press the "CALC TOOL" soft key (Figure 4-5). The locations are "averaged" and used to define the tool offset. The routine will give you the option of not using the location furthest away from the calculated center in the tool transformation calculation. Press "QUIT" to return to the previous menu.

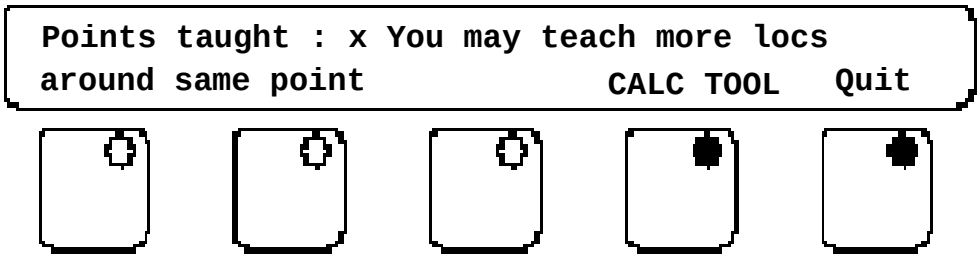


Figure 4-5 Calculating XYZ Offsets

If the tooling is attached at an angle with respect to the robot tool flange, and is not aligned normal to the tool z-axis of the robot and 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. Figure 4-6 shows an example of a tool transformation that contains yaw, pitch, and roll components.

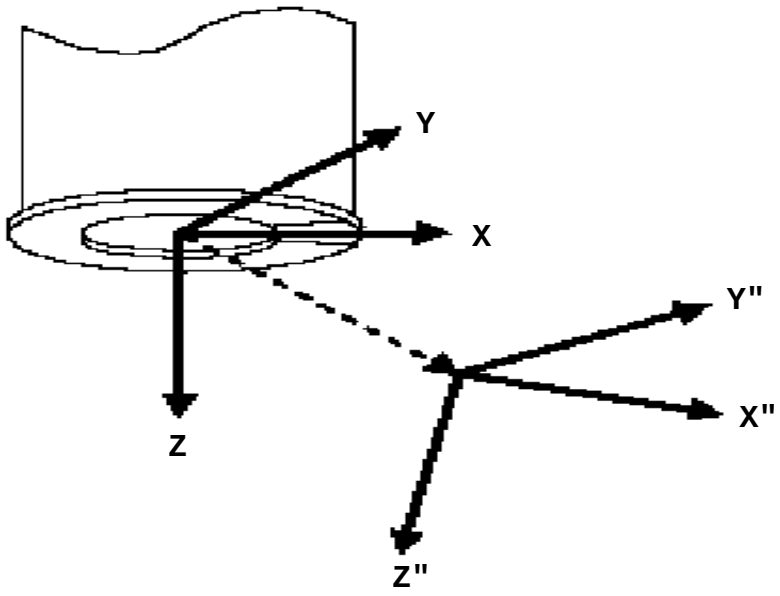


Figure 4-6
General Tool Offset

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

1. Press the "TEACH YPR" soft key (see Figure 4-4).
2. The pendant will prompt you to teach two robot locations along the tool's z-axis (the Z" axis in Figure 4-6.). 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" axes of Figure 4-6 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 the AIM Utilities 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 *SET TOOL* to set the 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-axes

4.6 Paths Relative to Frames

The AIM Utilities package allows the standard AIM Path data to be relative to frames. The file PATH.SQU provided with the AIM Utilities disk is different than the standard file provided with AIM. Paths relative to frames are accomplished by selecting the Named frame selection in the individual path records. This must be selected for every path record along your process using the frame.

4.7 Precision Point Module

The precision point software module is provided for those applications where robot rotary axes must be defined in a joint position rather than a world coordinate position. Examples are robots that have multiple axis definition to define a point or where an axis must unwind a joint before motion can begin. The Precision Point database menus are shown in Chapter 3 of this manual. The sequence statement Pmove is provided for these motions.

4.8 Software Plug-Ins

Several software modules are provided with the easy install file(*.INF). We call these modules Plug-Ins. These modules are additional software functions that are neat features or are totally separate applications. Future software features will be available in a Plug-In form for ease of installation. Plug-Ins can be installed by accessing the *Software Module Loader* under the *SETUP* pulldown. Select the *Install* button to get to the menu shown below. The AIM Utilities Software Plug-In disk contains the following software modules. The display below is the actual Install window that will appear when using this disk.

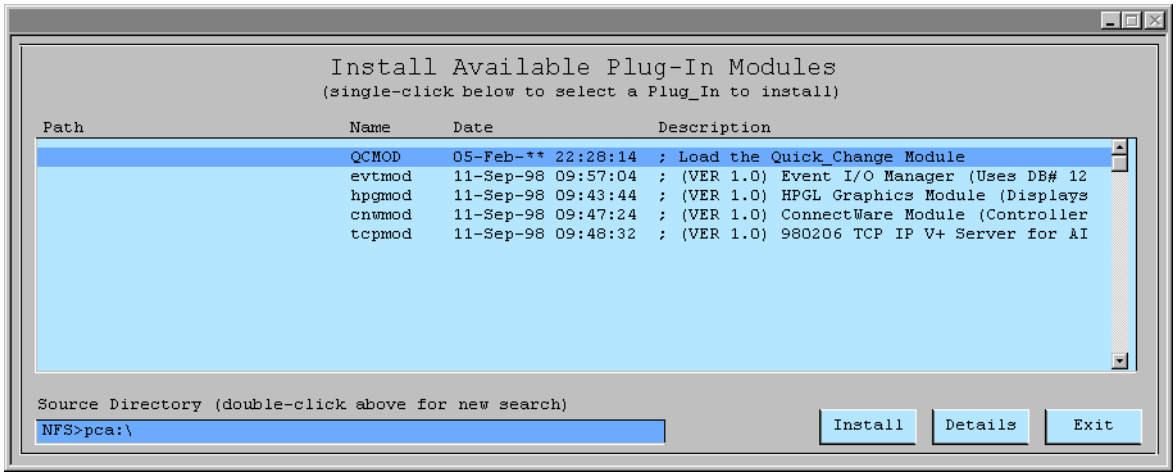


Figure 4-7
Available Software Plug-Ins

The available Plug-Ins are listed below:

1. QCMOD.INF --- Quick Change Module for Changeable Tooling
   This software module provides utilities to deal with tool signatures and placing and retrieving tooling from nest locations. See the appendix chapters provided in this manual for more details.
2. EVTMOD.INF --- Event I/O Manager. This module allows the user to setup I/O events that can be measured in Time and a selected output triggered if the event fails. See the appendix chapters provided in this manual for more details.
3. HPGMOD.INF --- HPGL Graphics Module (Displays Corel HPGL Exports)
   This module allows the user to setup graphic panels on a menu page to display disk resident files onto the menu. HPGL export is currently the only file format supported. See the appendix chapters provided in this manual for more details.
4. CNWMOD.INF --- ConnectWare Module. This module adds new custom statements that allows multiple MV Controllers that are running AdeptNet to communicate through AIM DDE. See the appendix chapters provided in this manual for more details.
5. TCPMOD.INF --- TCP IP V+ Server. This module allows a PC or MV TCP Client to retrieve information from the controller that is running this application. See the appendix chapters provided in this manual for more details.

4.9 Example Sequences

A diskette is provided with AIM utilities that provide examples for different applications. These files are Imported using the AIM Modules utilities. In most cases there are several databases provided with these examples. The examples are provided to give ideas to the user to solve their application problems. They are not intended to work directly in a customer's application. The list below describes the database modules available at this time. Example sequences are documented in appendix A of this manual.

1. EX_DISP.MOD --- This database module contains dispense examples for use with the AIM Dispense Module. Included with in this example is a conveyor dispensing application, a Cell Control or supervisor sequence for error recovery and other dispense examples.
2. EX_MCP.MOD --- This example database module is a sample sequence of how to use the MCP sequence task statements provided in the AIM Utility package. This example shows how to create a sequence to allow a pendant routine to change the palletizing data in the locations database.
3. EX_PALWR.MOD --- This example database module is an example PalletWare application. The AIM PalletWare software must be loaded to view these example databases.
4. EX_START.MOD --- This example shows a startup sequence which will also enable power and calibrate the robot.

Sequence Examples A

A.1 Introduction and Overview

Provided in the appendix are several example sequences which are available in various files on the AIM Utilities Example Database disk. They are listed here for reference.

Start-Up and Calibrate Sequences

These sequences are examples of how to auto-start and calibrate robot systems. This first example resides in the 'EX_START.MOD' modules files. This sequence will startup the production cell and calibrate the robot. The calibrate sequence is the second sequence provided.

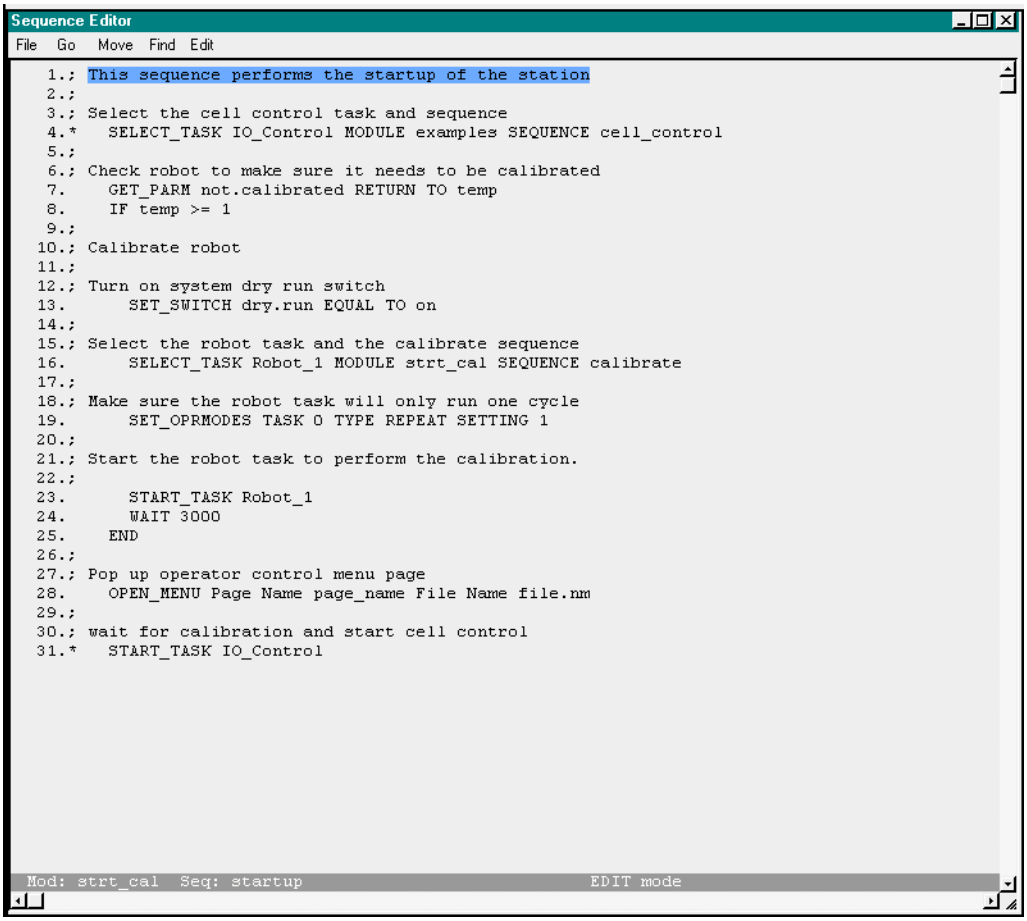


Figure A-1 Auto Start-Up Sequence

Calibrate Sequence

This calibrate sequence is located in the 'EX_START.MOD' modules files provided with AIM Utilities.

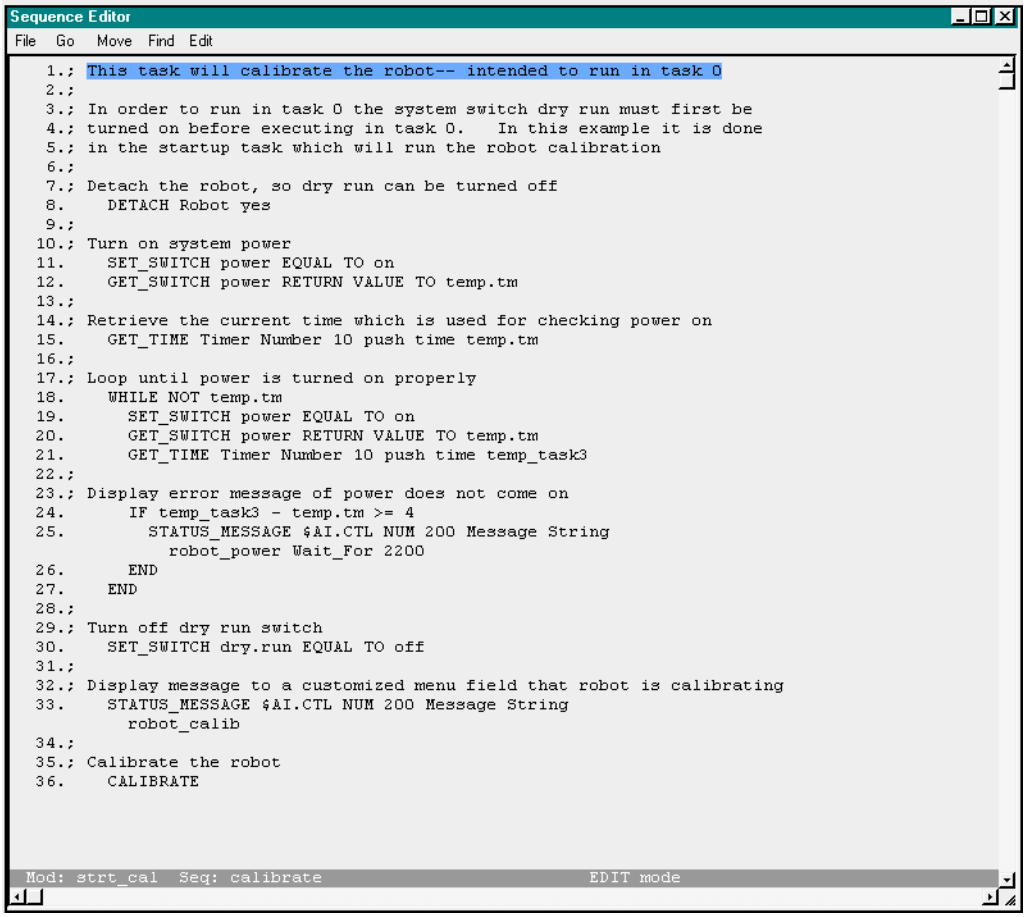


Figure A-2 Calibrate Sequence Example

Start-Up Sequence Example from PalletWare

This sequence is a startup sequence that came from the PalletWare example. With the Adept 1850 Robot, calibration occurs during V+ boot-up because the robot uses absolute encoders. This is a very simple way to start another sequence running in a different task. This example is located in the File 'EX_PALW.MOD

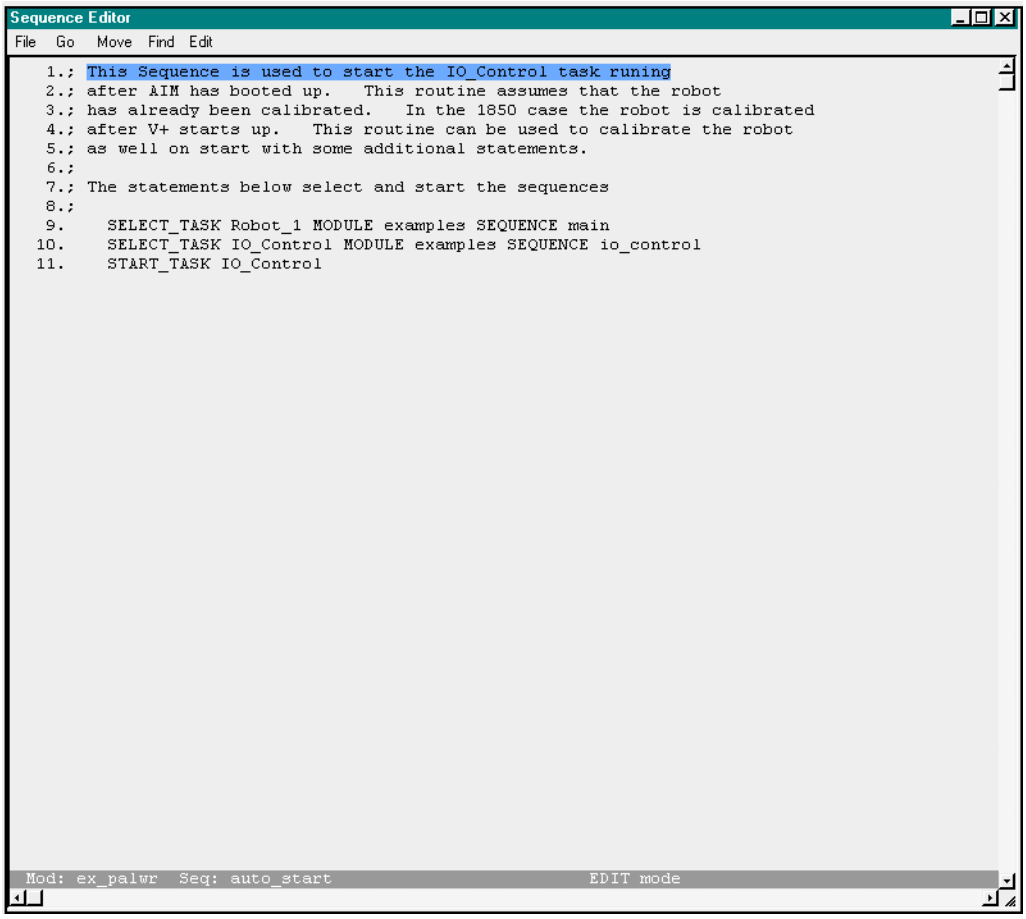


Figure A-3 PalletWare Start-Up

Start Button Control Sequence

This example is a modified version of the provided START sequence. This modification will start the IO_Control task. This example is located in the file 'EX_PALW'.

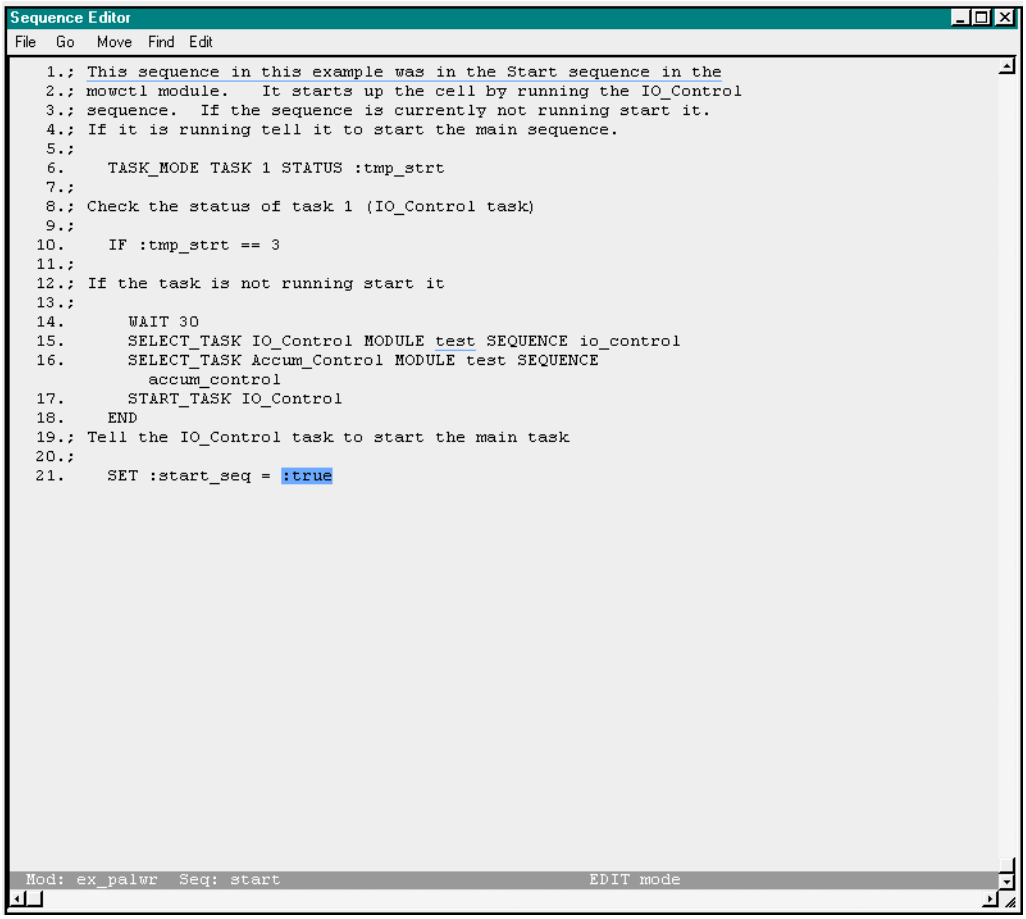


Figure A-4 Start Button Sequence Example

Cell Control Sequence Example

This example comes from an application that was running the dispense module software. In this example a supervisory sequence is running to handle error recovery and startup of the main robot task when needed. The cell control task is always running during normal operation.

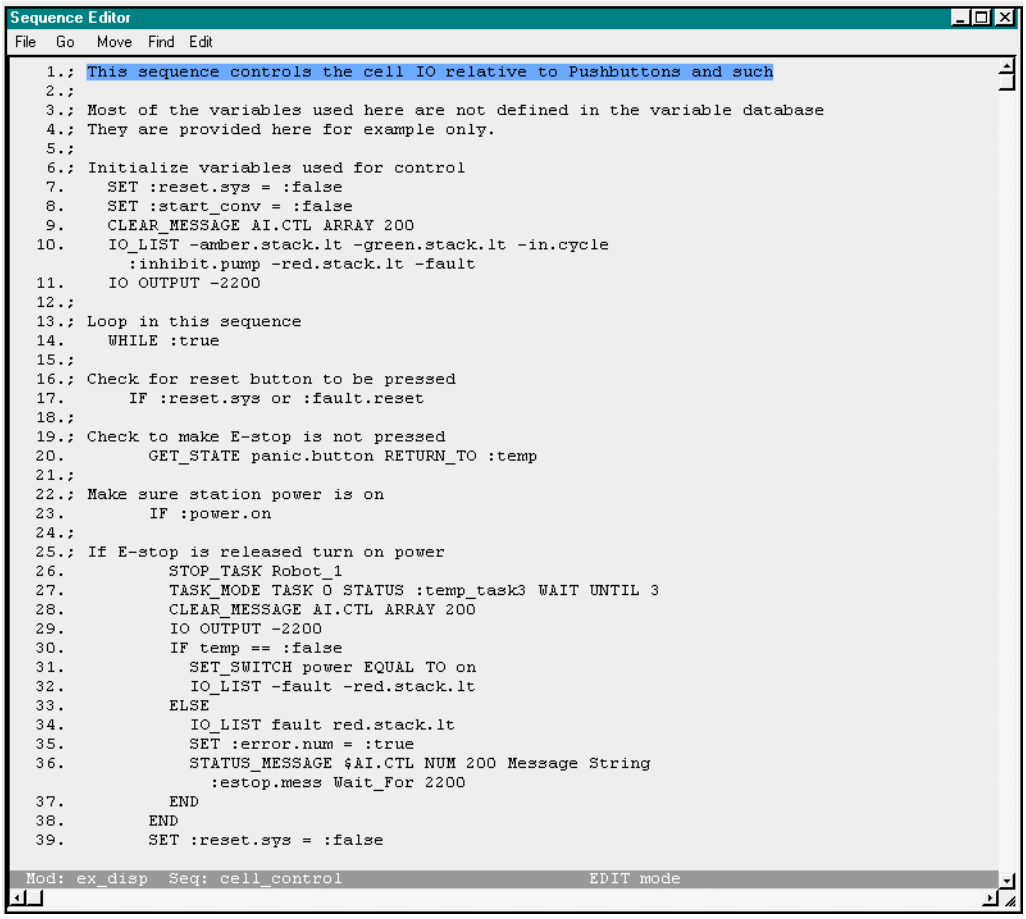


Figure A-5 Cell Control, (First Example Page 1)

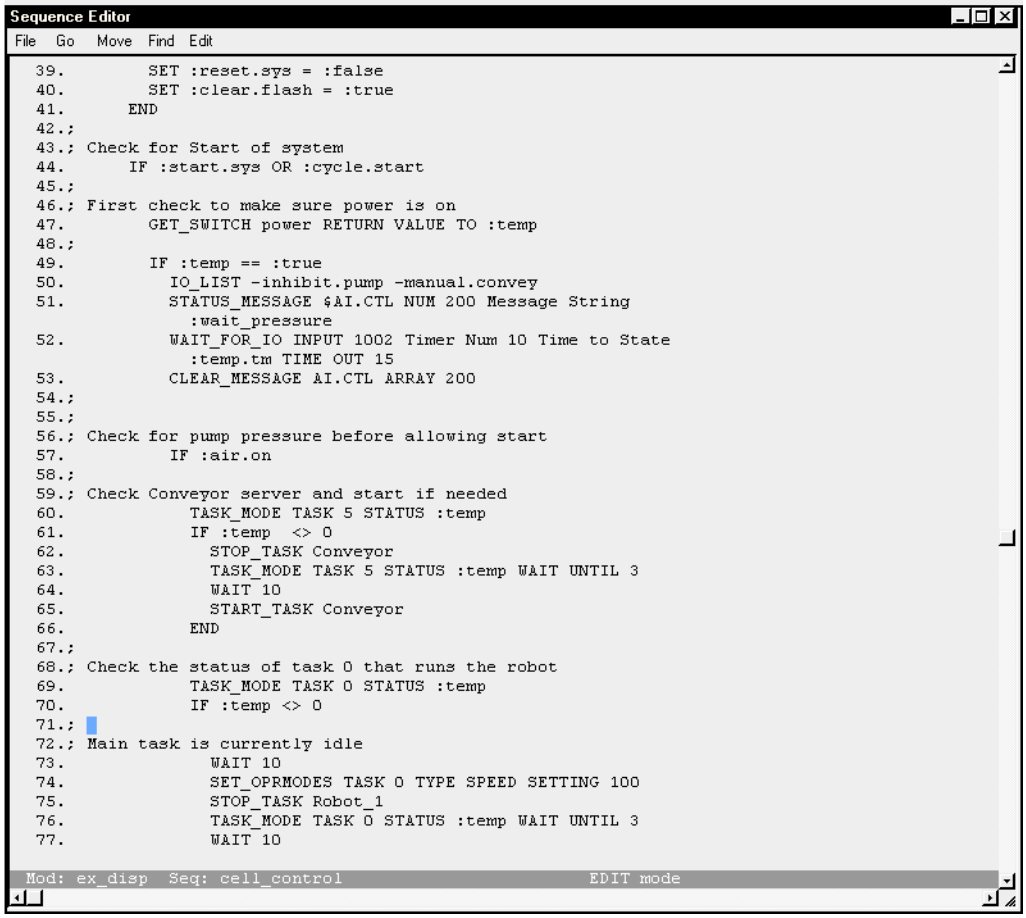


Figure A-6 Cell Control, (First Example Page 2)

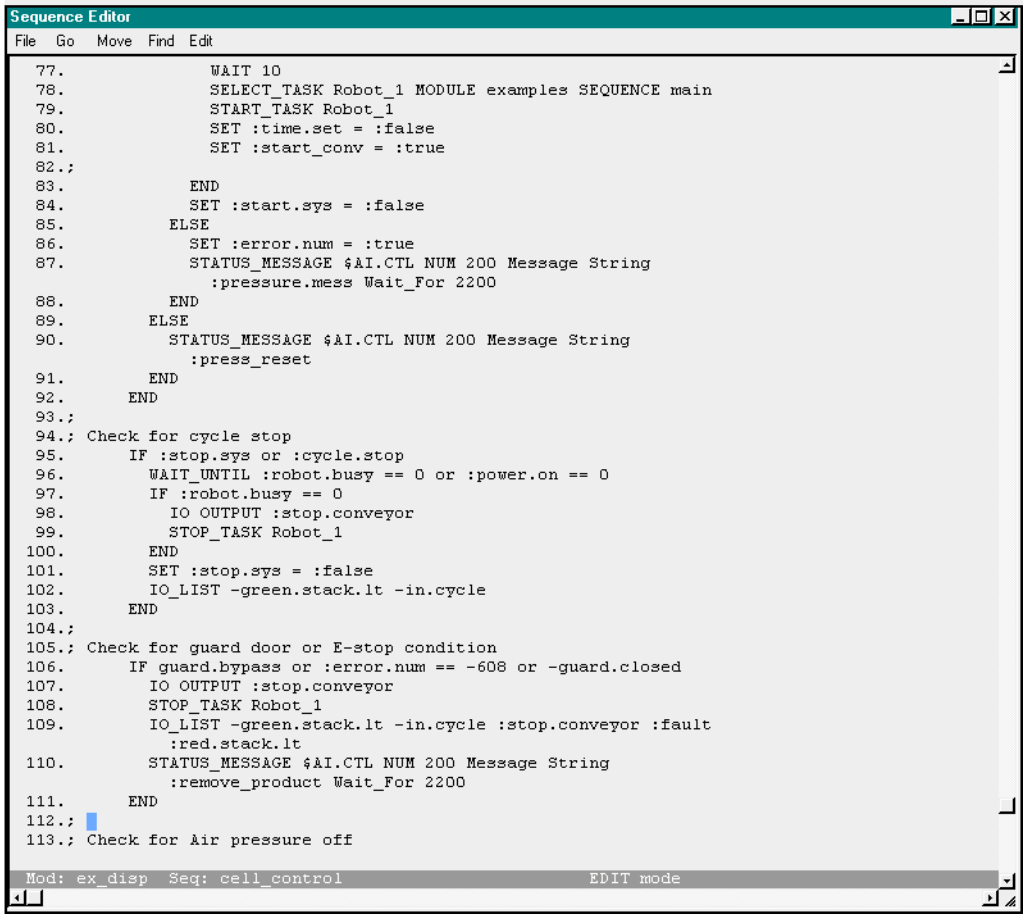


Figure A-7 Cell Control (First Example Page 3)

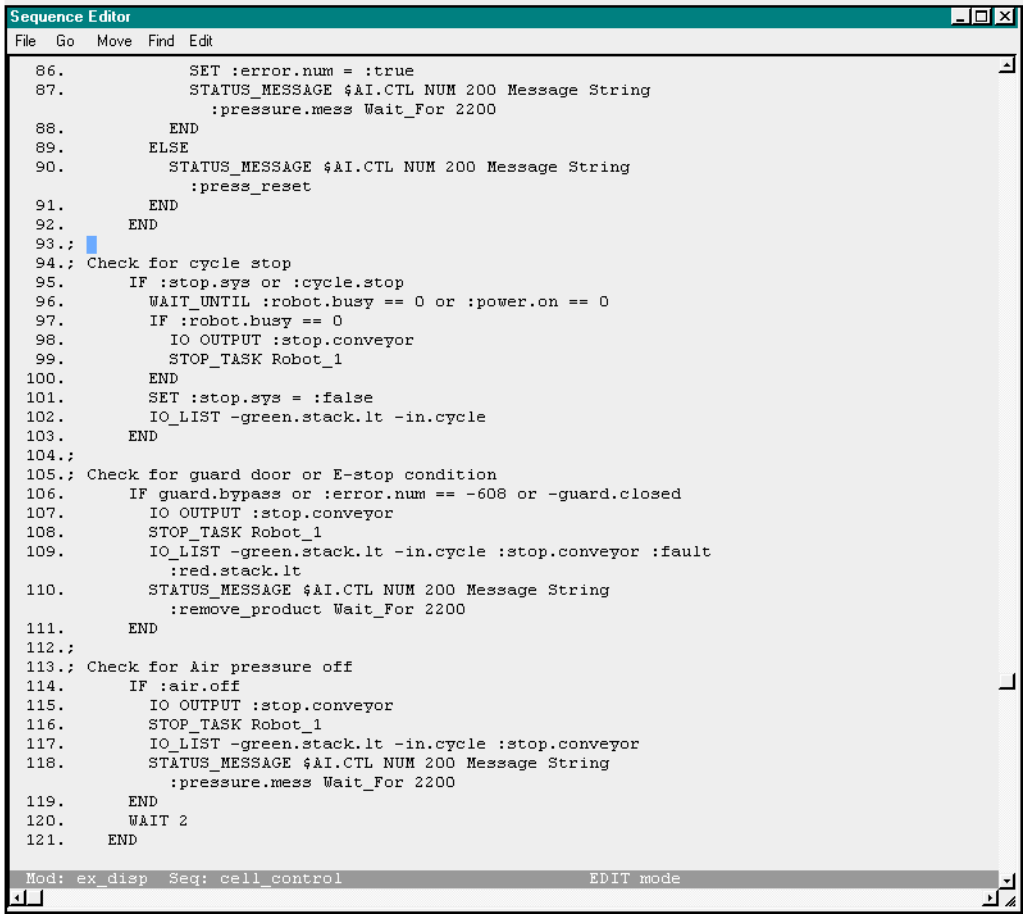


Figure A-8 Cell Control (First Example Page 4)

Second Cell Control Example

This example comes from a PalletWare application. This application is similar to the dispense example in that the cell control task handles error recovery and startup of the main robot task. This example is several pages long. It is located in the file 'EX_PALW.MOD'

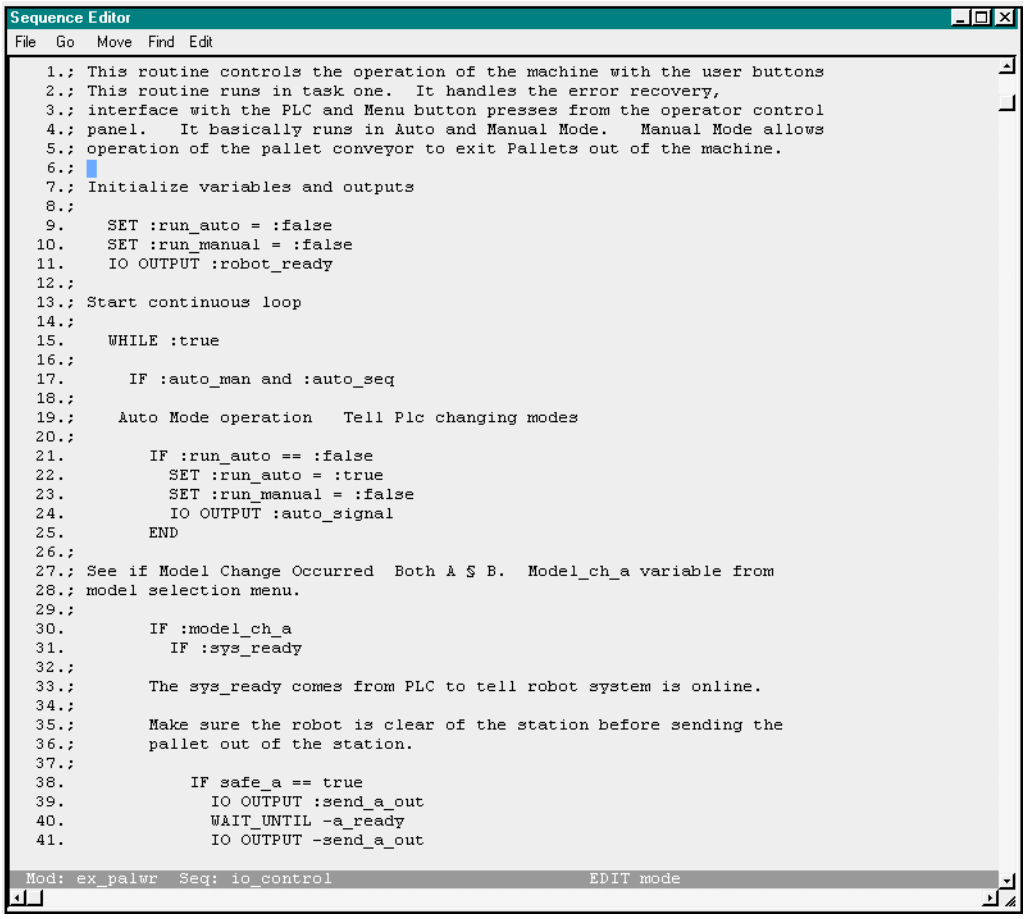


Figure A-9 Cell Control (Second Example Page 1)

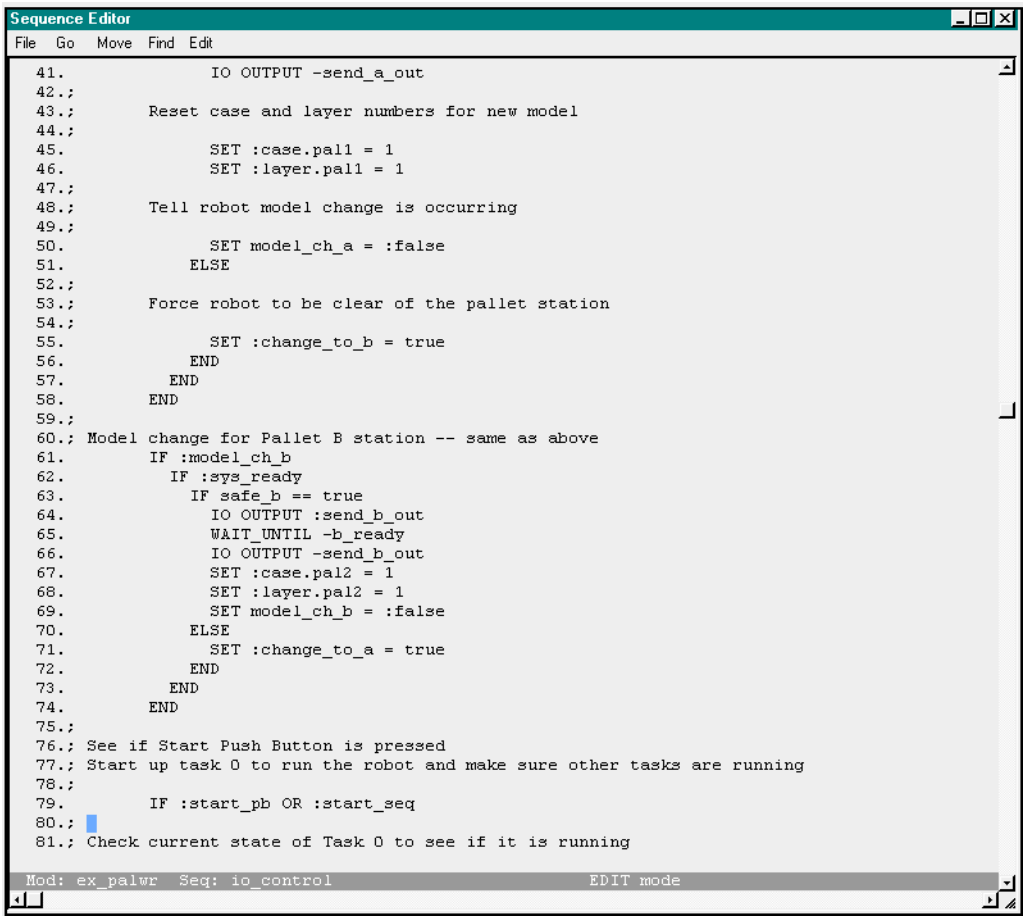


Figure A-10 Cell Control (Second Example Page 2)

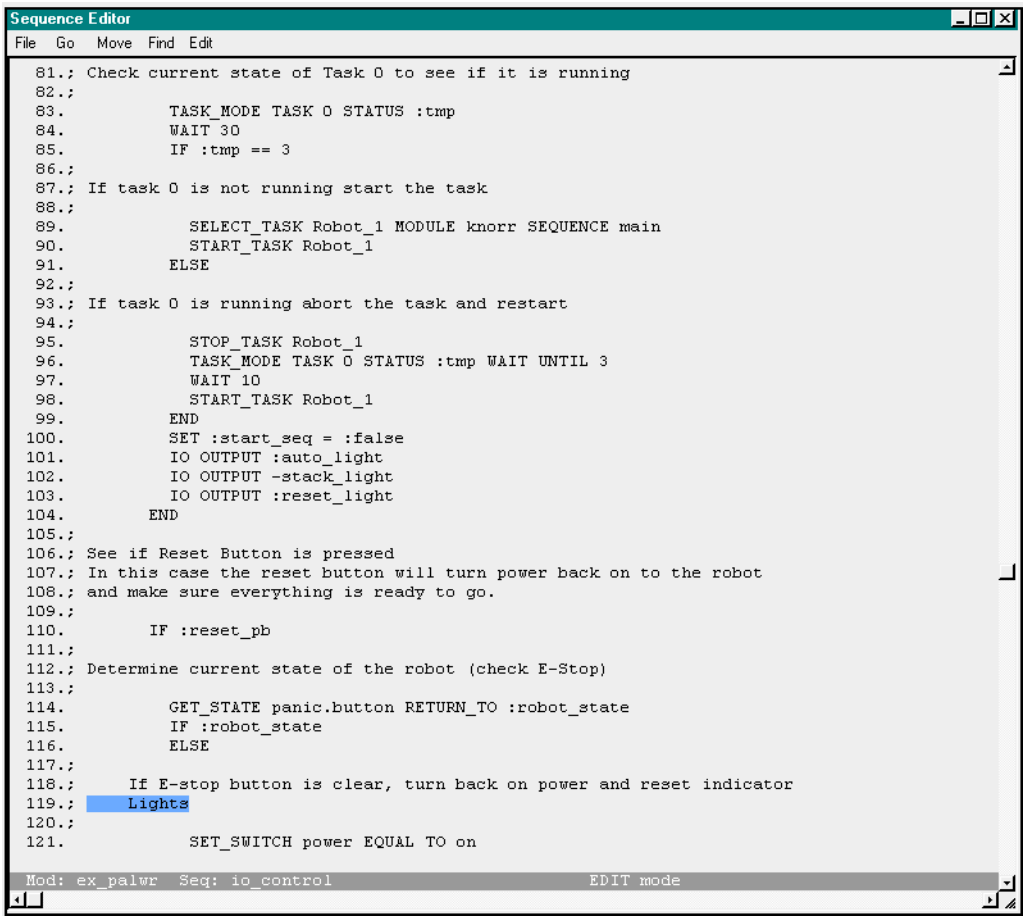


Figure A-11 Cell Control (Second Example Page 3)

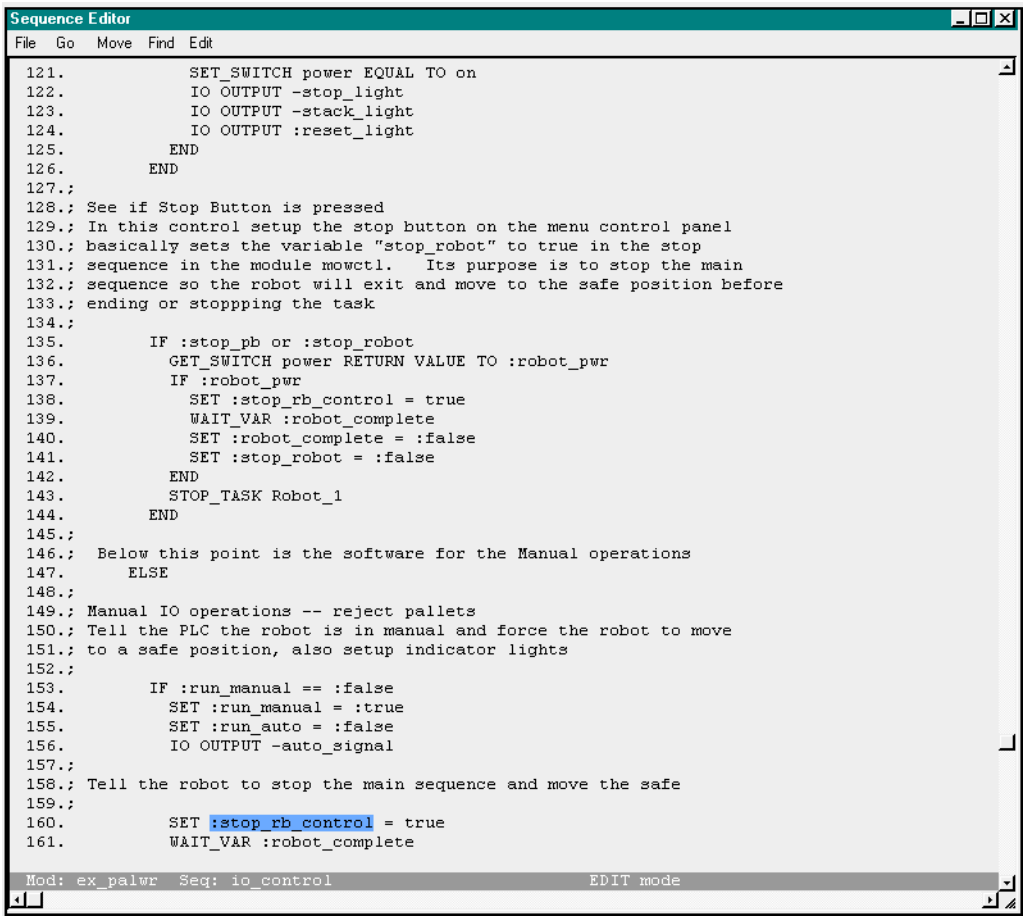


Figure A-12 Cell Control (Second Example Page 4)

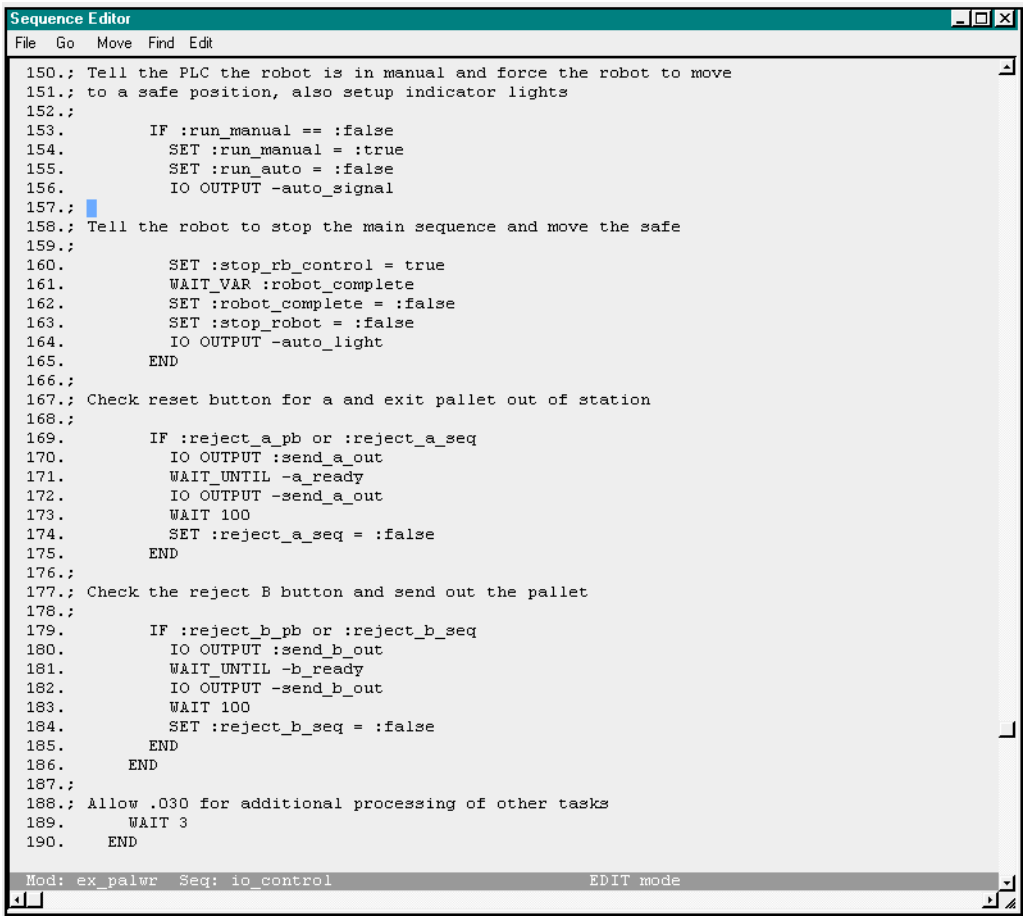


Figure A-13 Cell Control (Second Example Page 5)

Robot Main Sequence

This sequence controls the motions of the robot system. This example comes from a dispense module application and is used for conveyor tracking. This is located in the 'EX_DISP.MOD' modules file.

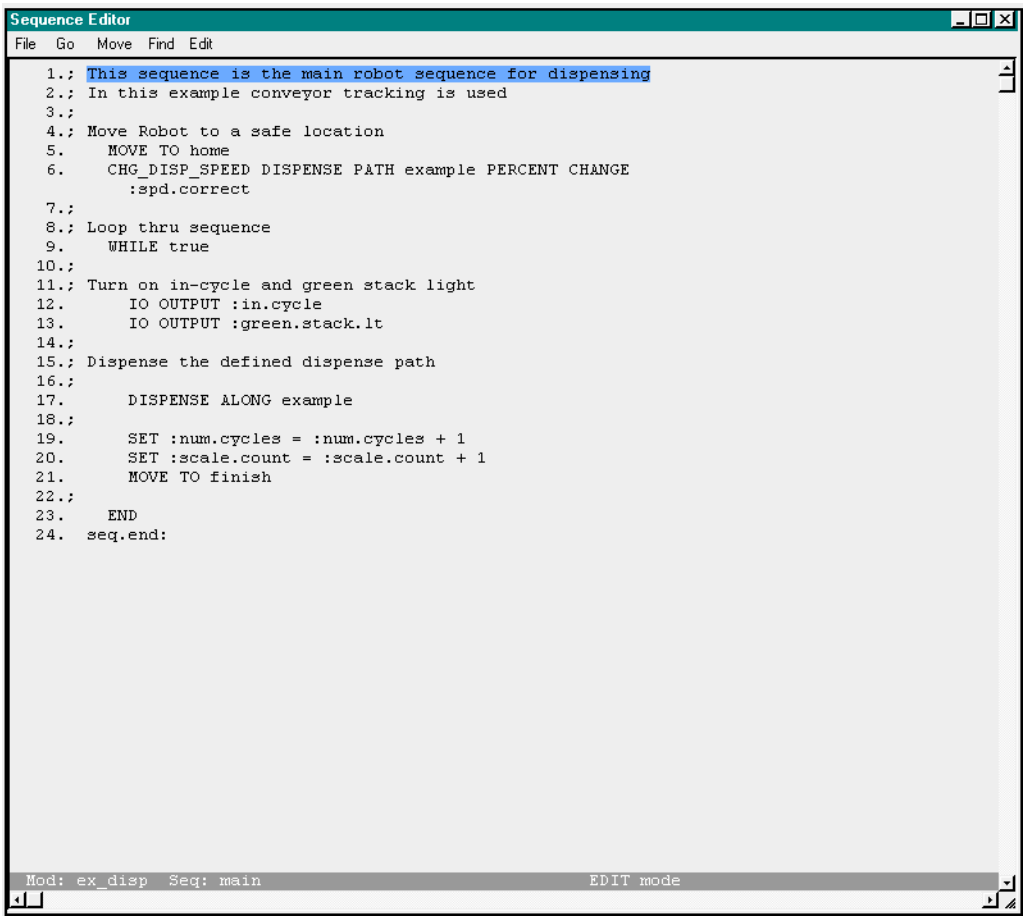


Figure A-14 Dispense Robot Main Sequence

Main Sequence Example from PalletWare

This example comes from a PalletWare example. It is very useful in showing how to handle Righty and Lefty arm configurations as well as a general guide in PalletWare applications.

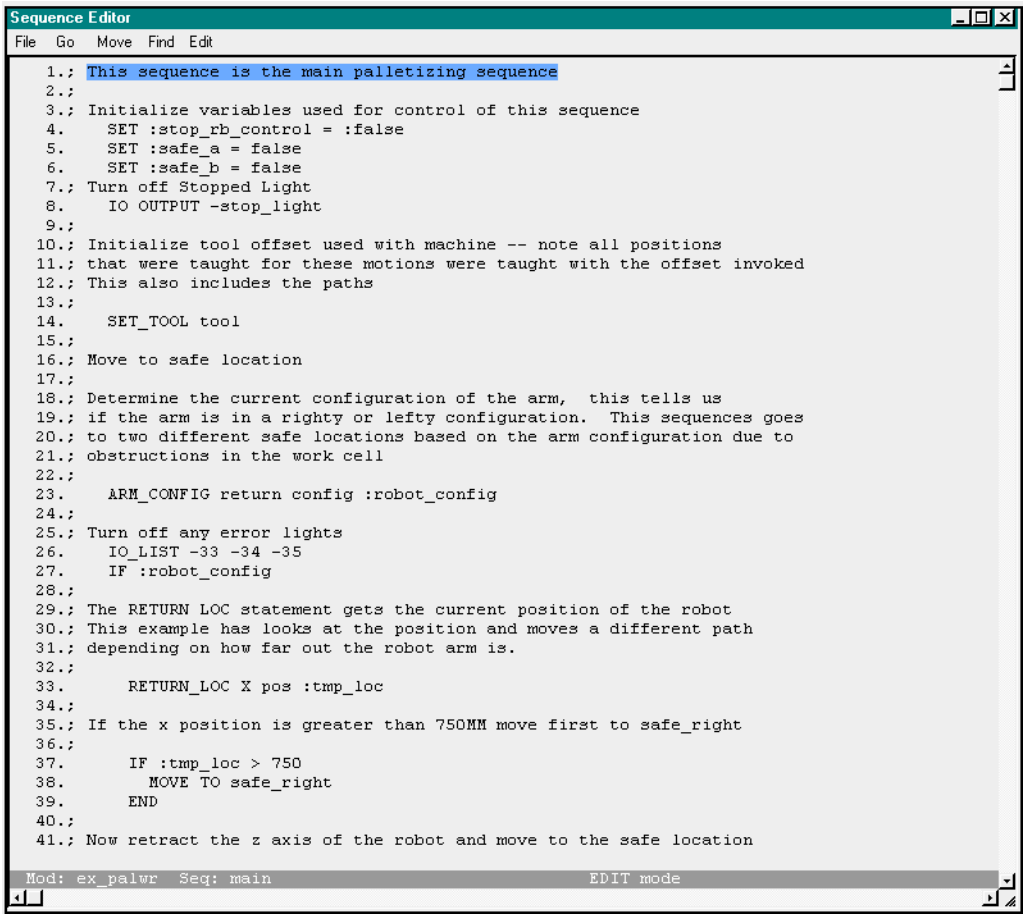


Figure A-15 PalletWare Main Sequence (Page 1)

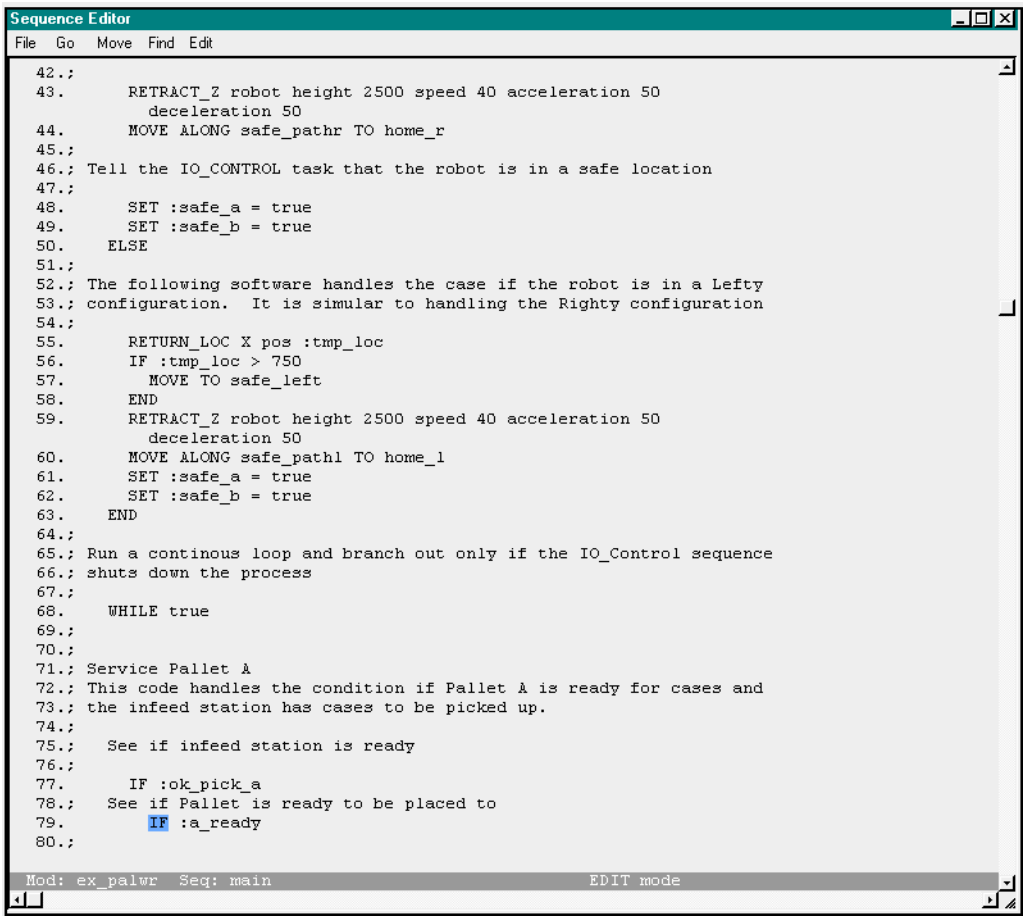


Figure A-16 PalletWare Main Sequence (Page 2)

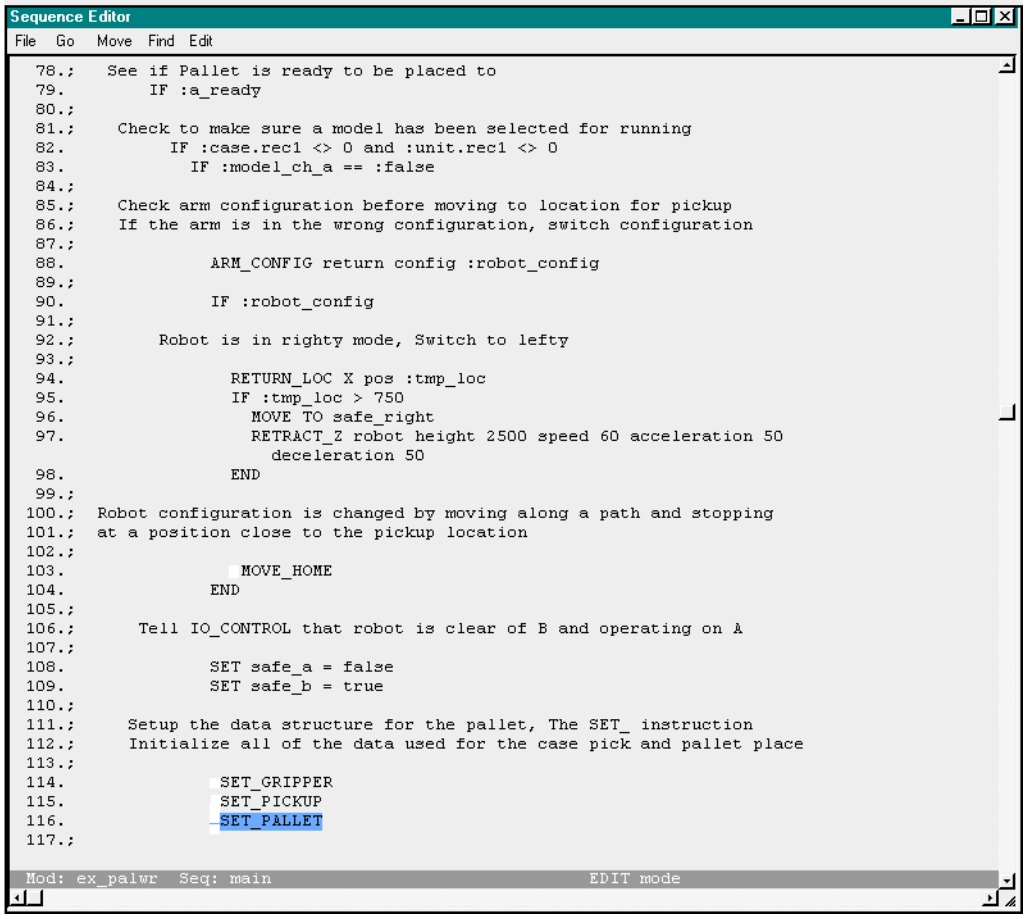


Figure A-17 PalletWare Main Sequence (Page 3)

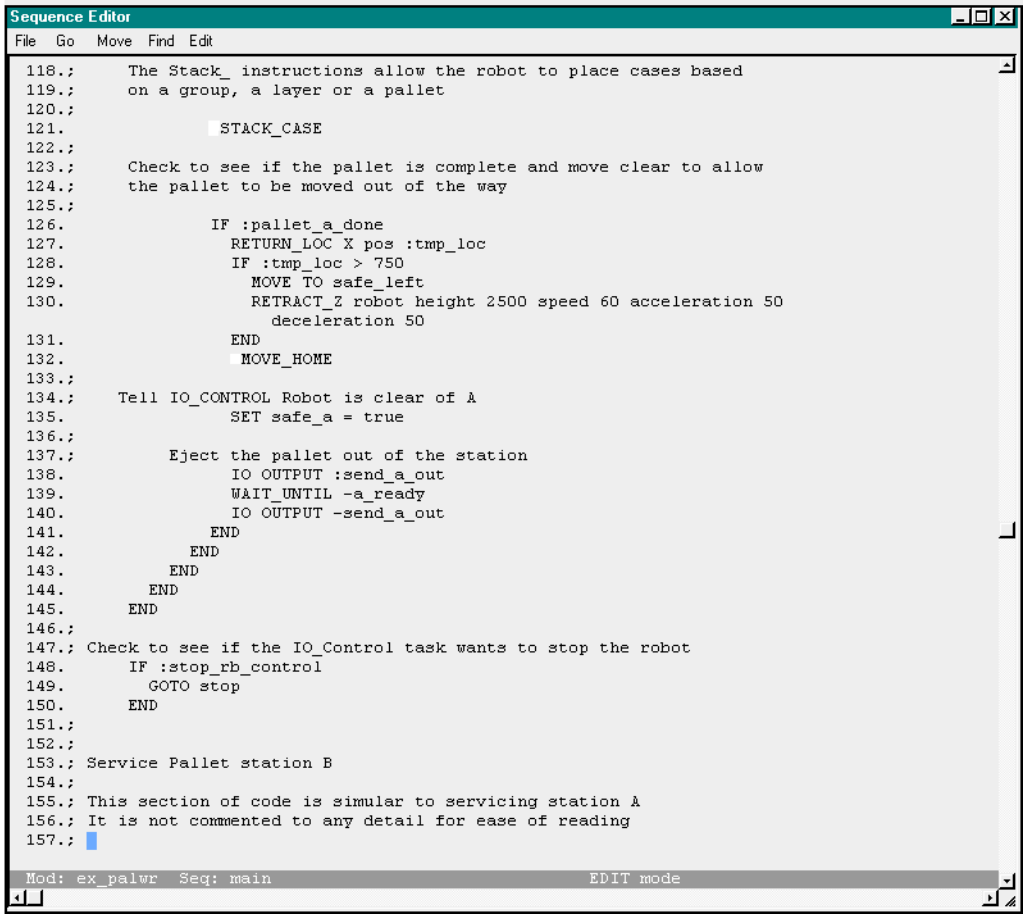


Figure A-18 PalletWare Main Sequence (Page 4)

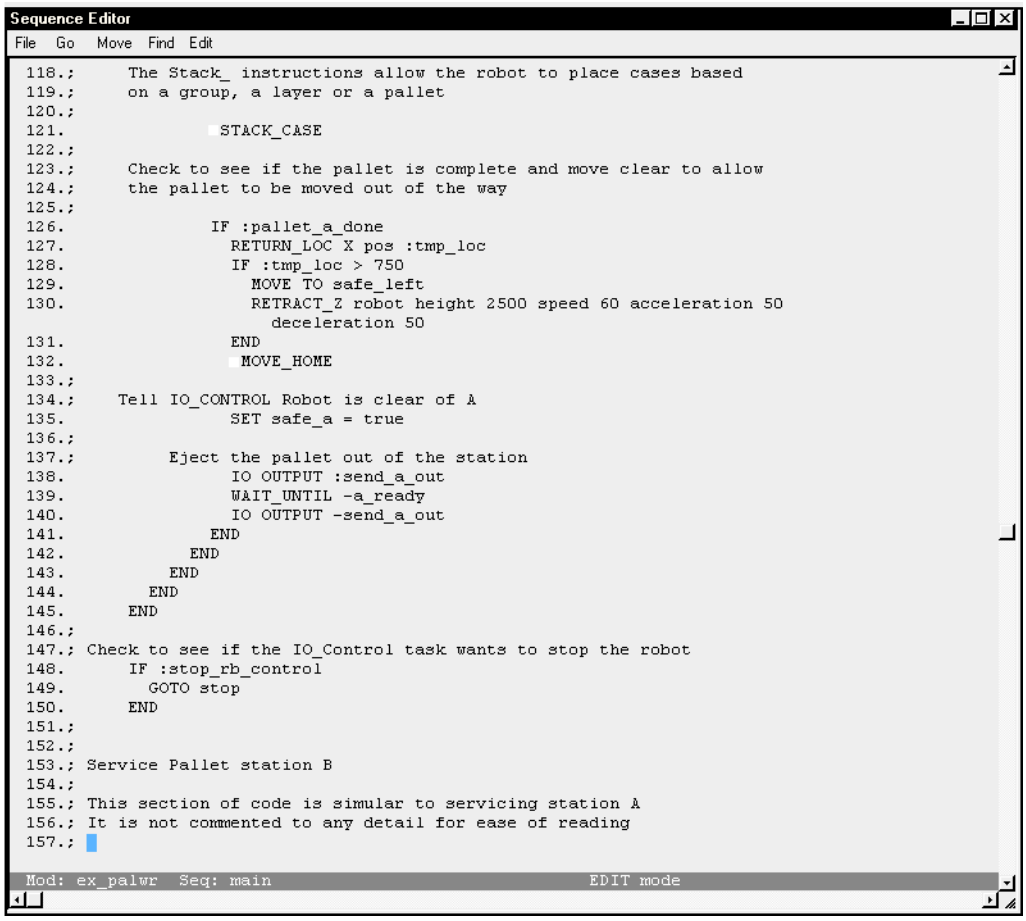


Figure A-19 PalletWare Main Sequence (Page 5)

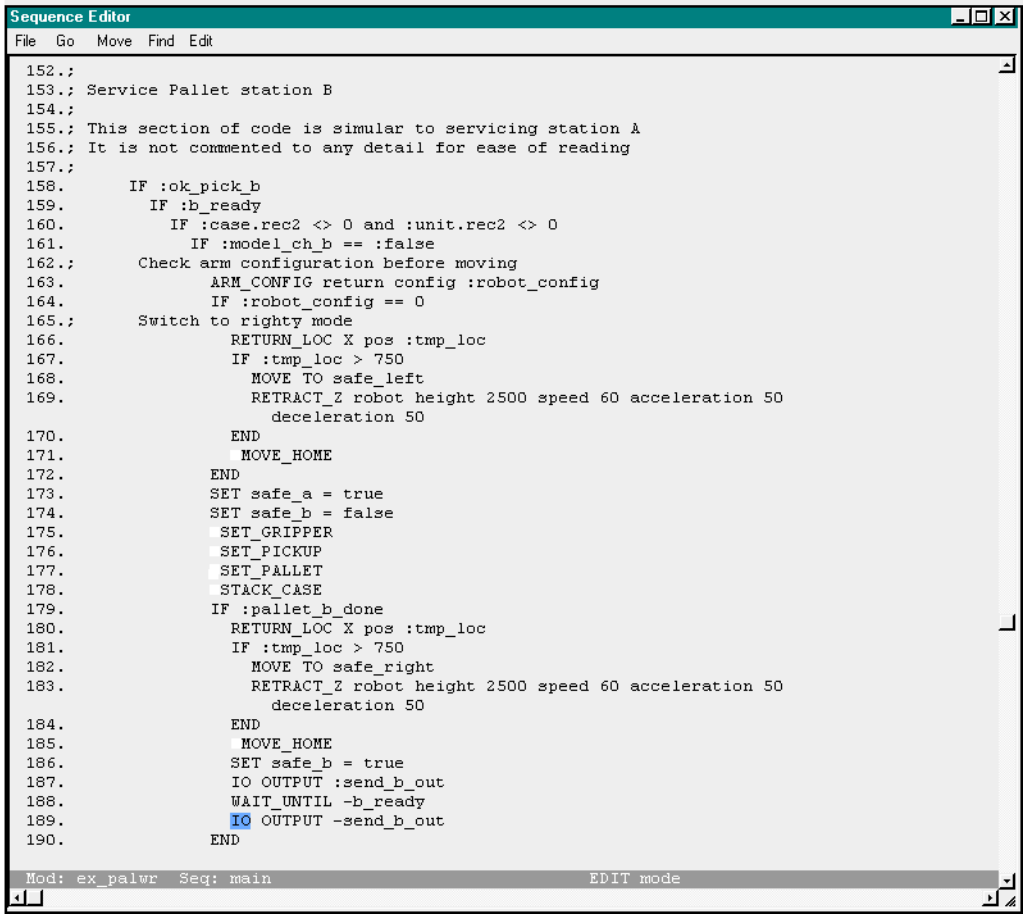


Figure A-20 PalletWare Main Sequence (Page 6)

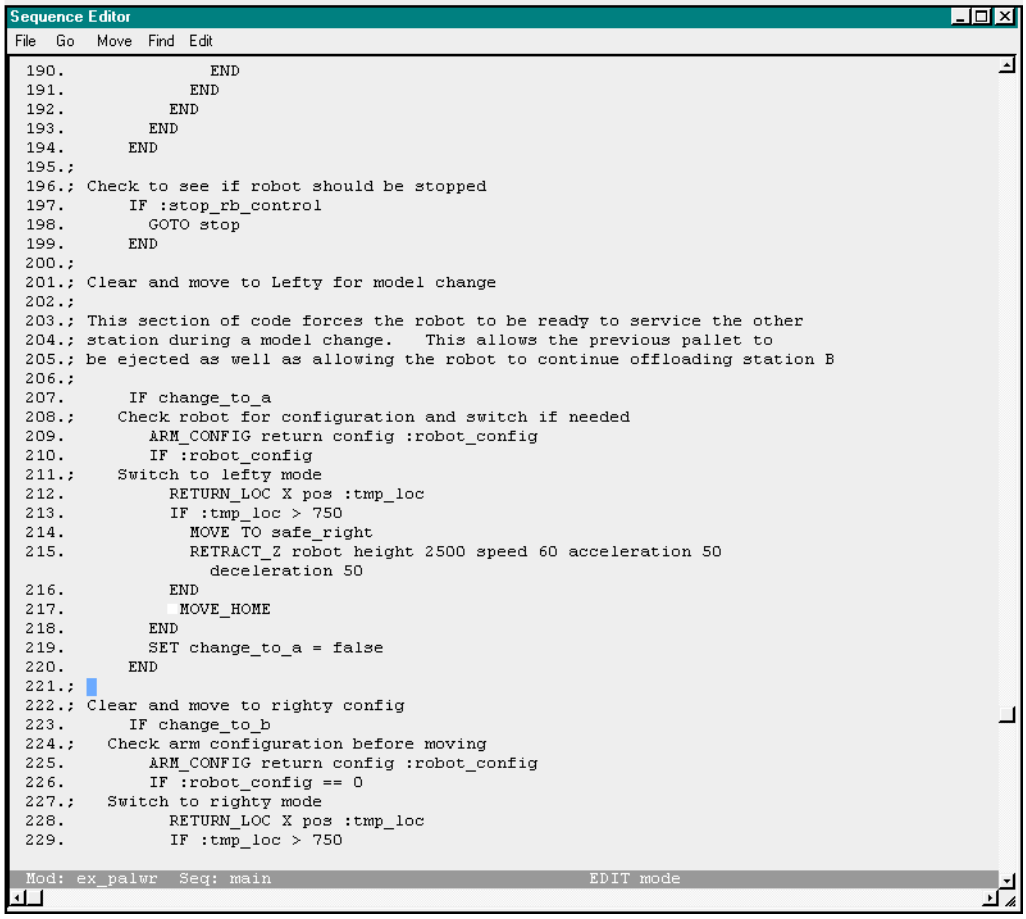


Figure A-21 PalletWare Main Sequence (Page 7)

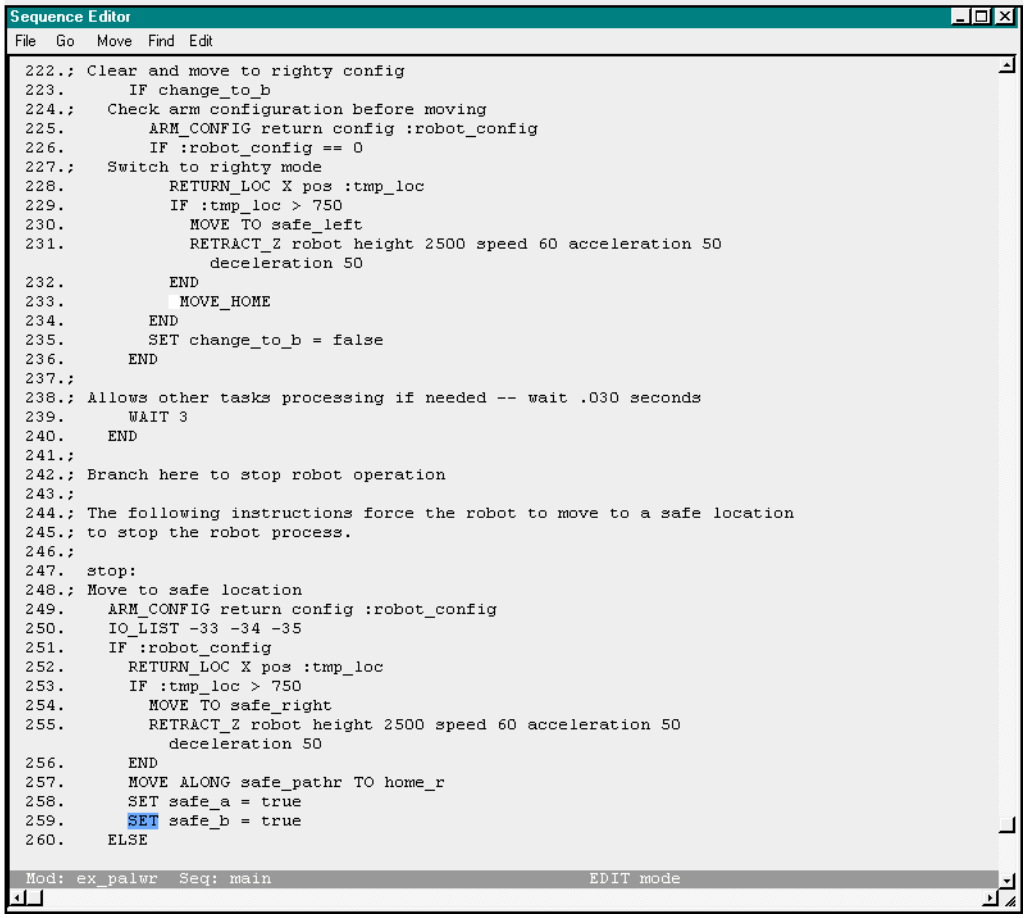


Figure A-22 PalletWare Main Sequence (Page 8)

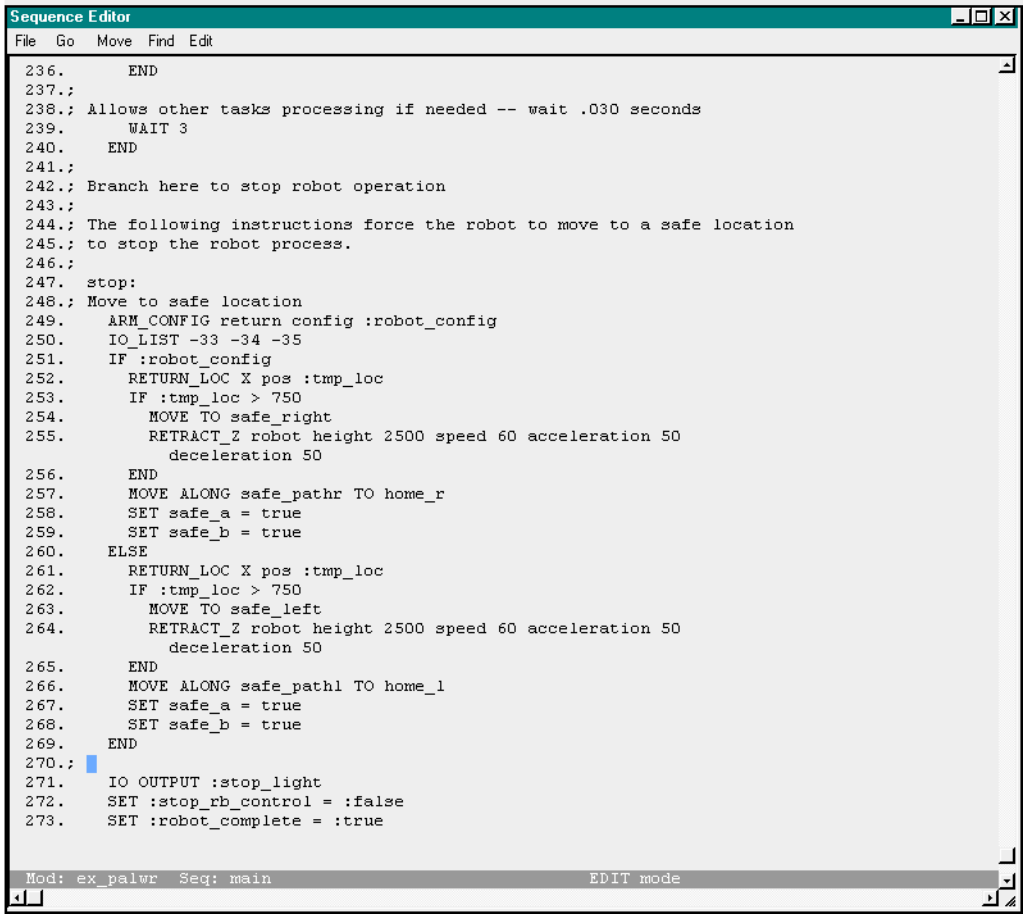


Figure A-23 PalletWare Main Sequence (Page 9)

MCP Sequence Example

This example demonstrates running a sequence to use the MCP as the interface. This particular example uses the MCP to allow the operator to change the pallet parameters for a standard motionware depalletizing pick and place application. This sequence is available in the 'EX_MCP.MOD' modules file.

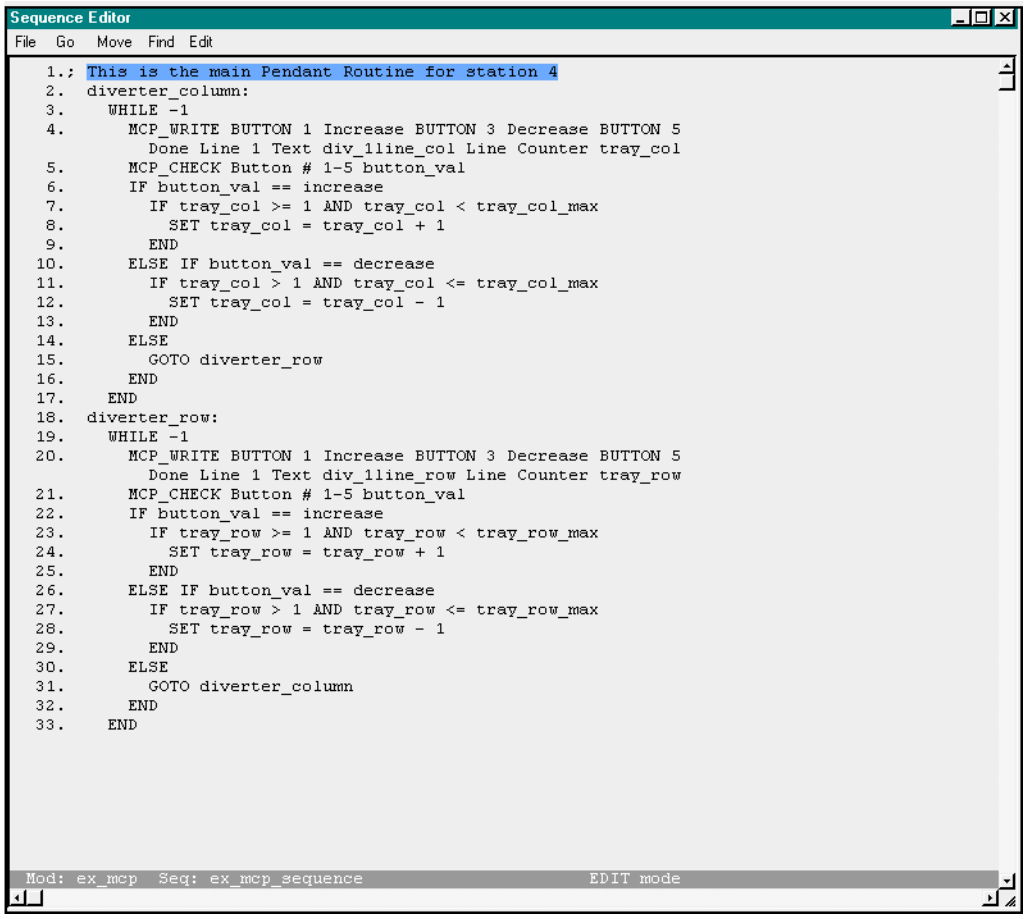


Figure A-24MCP Example

Quick-Change Documentation B

B.1 Introduction and Overview

The Quick-Change Module is a fully integrated software package designed specifically for use with Adept products. This software module simplifies the programming of changing multiple robot tooling with Adept Aim software. This software is for use with Aim revision 3.1 and higher. This software module is supplied with the AIM Utility software package.

This document includes all the information to implement this software with other Aim Modules. The document is divided into 4 sections: Introduction and Overview, Quick-Change Database, Software Operation, and Routine Dictionary.

The Introduction and Overview section provides information on sequence task statements, software requirement, and hardware requirements.

The section on the Quick-Change Database provides information about the requirements and uses of the database.

The Software Operation section provides information on how to edit and use the Quick-Change database, and provides information on application requirements.

The Routine Dictionary section provides information about the V+ routines that run the Quick-Change Module.

The CHANGE_HAND Statement

The statement performs tool-change operations if quick-change end-effectors are used. With the CHANGE_HAND statement, the robot can be programmed to automatically return a tool to its nest, exchanging it for a new one. A new Aim database called for Quick-Change has been added to allow the positions and I/O signals to be entered.

Installation Procedure

This section describes the Quick-Change Module installation procedure. To simplify the installation, the user should have the ***Adept Utility Disk***. The Utility Disk is supplied with all Adept controllers. Because this software is strutured in AIM, the system must be equipped with "A" series controller.

Table B-1 lists the files that are included with the Quick-Change Module. The files reside on the ***AIM Utilities*** diskette; this diskette contains the runtime routines, and the database definition files. 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. In Table B-1, files marked with a "P" are protected files and cannot be read by the user.

To install the Quick-Change Module, perform the steps described below. These steps assume that AIM version 3.1 or later is already copied onto the system's hard drive. The installation procedure copies all the files listed in Table B-1 onto the hard disk drive.

1. Load and execute the DISKCOPY utility from **Adept Utility Disk**. To execute the program, place the Utility Disk in drive A and issue the following monitor commands.

```
LOAD A:DISKCOPY
EX A.DISKCOPY
```

2. Remove the Utility Disk and insert the ***AIM Utilities*** disk into drive A. Choose the "Copy multiple files" option from the DISKCOPY menu and copy all of the ".V2" and ".SQU" files to the AIM default subdirectory on the hard disk drive. These files must be reside on the AIM default subdirectory during AIM execution or an error will occur. Specifying the wildcards "*.V2" and "*.SQU" for the copy command will copy all of the ".V2" and ".SQU" files to the hard disk. Because some of the files replace standard AIM files, answer "Y" (for yes) to DISKCOPY's *supersede* prompt.

Table B-1
Quick-Change Module Files

Files	Contents
RUN_CH.V2	Commented Quick Change runtime routines.
RUN_CH.SQU	Squeezed Quick Change runtime routines
MENU_CH.SQU	Squeezed menu routines.
QCMOD.OVR	Startup and initializing Quick Change Software.
QUICKCH.MNU	Quick Change menu file.
QC_IO.MNU	Quick Change IO menu file
QUICKCH.RFD	Quick Change database rfd file.
QUICKCH.DB	Quick Change database.
QCICON.DAT	Quick Change menu icons.
ERRORQC.DB	Quick Change error database.
STATQC.DB	Quick Change statement database

The ".DB" , ".RFD", and ".MNU" files of Disk #1 must reside in the same subdirectory as the AIM Base Package ".DB" files. Copy those files to the appropriate subdirectory.

Specify a "*.*" diskcopy if the program, database, and database definitions files were placed in the same subdirectory when the AIM Base Package was installed. Again,

supersede existing files with DISKCOPY since some ***AIM Utilities*** files replace AIM Base Package files.

After successful completion of the steps above, the installation is complete.

B.2 The Quick Change Database Menu

The Quick Change database supports all the needed data for the Change_Hand sequence statement. The Quick Change database includes a field that defines the tool nest location (where a quick-change end-effector is acquired and returned), fields for tool signature and nest presence signals, and fields that specify I/O reset routines. The I/O reset programs, executed before returning a tool to its nest or after the tool has been acquired, sets end-effector I/O to default settings. These routines can be used to prevent air lines from blowing when a tool is detached from the robot.

The Quick Change database record form is shown in Figure B-1 and the field descriptions are in Table B-3. Data from each field can be accessed in an applications program by using the V+ variable names listed in Table B-4. The record fields are described below.

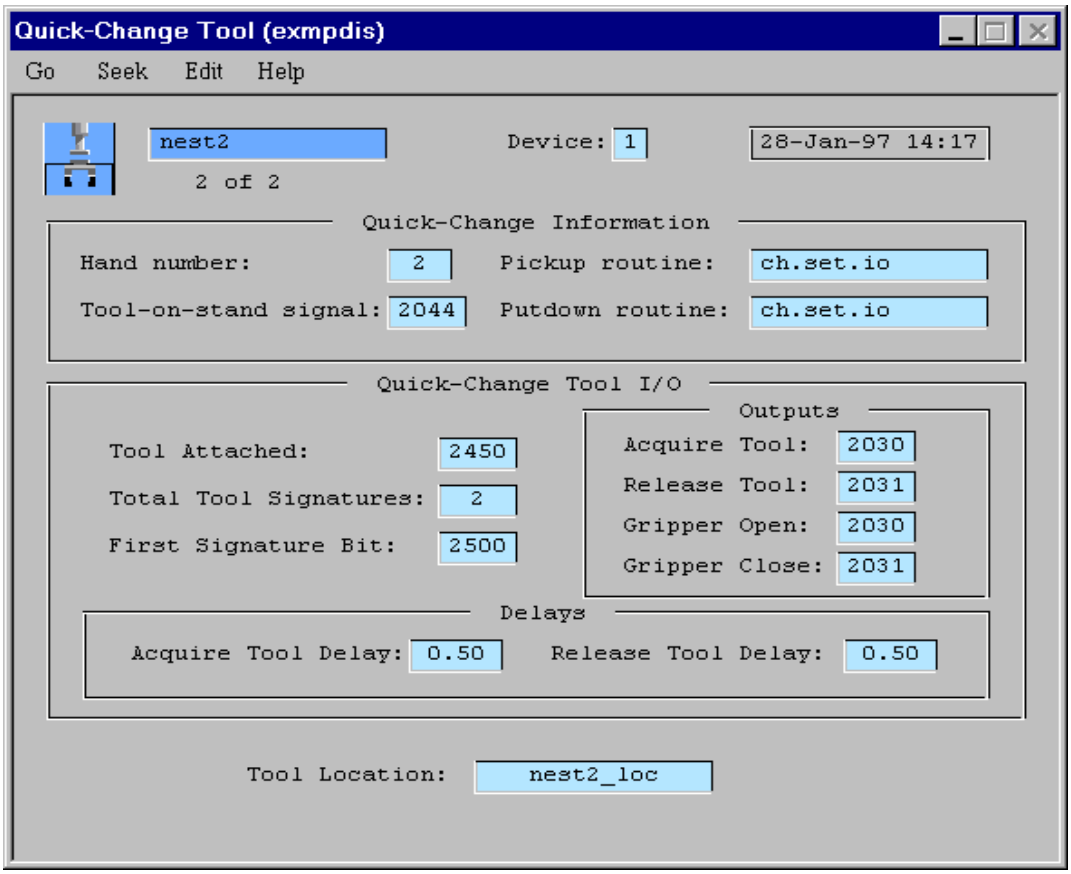


Figure B-1
Quick-Change Database Record Form

Quick-Change Database

Data for the ***Change_Hand*** statement is stored in the Quick-Change database. Important fields in the Quick-Change database include the tool nest location (where the end-effectors are acquired and replaced), I/O and Delay parameters, and the I/O reset routines.

The tool nest locations are specified with the standard AIM location database. Double click on the location field and the software will branch to the location record. Like other

location fields, the location is taught by moving the mouse pointer to the Here or Teach Button and clicking the mouse. If Teach was selected the standard Aim teach pendant routines are used to define the location. Approach and depart fields must also be defined for this location.

The Quick-Change database has several fields that are used for Input Signals to operate the change hand mechanism. The list of possible inputs include: *Tool-On-Stand*, *Tool Attached*, *Total Tool Signatures*, *First Signature Bit*. The *Tool-On-Stand* signal determines if the end-effector is in the nest, this input signal is required for operation of the software. The optional *Tool Attached* signal allows the software to check for the presence of the end-effector on the robot arm. The optional *Total Tool Signatures* and *First Signature Bit* allow the system to be configured to check which Hand Number is attached to the robot. The *Total Tool Signatures* field defines the number of bits required to sense the largest Hand Number available. The *First Signature Bit* defines the first input signal of the consecutive signals to define the Hand Number. The optional *Hand Number* field allows the user to define hand numbers to each end-effector. This field is compared to the sensed number when the tool is picked up to assure the proper tool is available.

The available Quick-Change outputs fields are *Acquire Tool*, *Release Tool*, *Gripper Open*, *Gripper Closed*. The *Acquire Tool* and *Release Tool* output signals operate the change hand mechanism to pick-up and release the end-effector. The *Gripper Open* and *Gripper Closed* signals are both turned off when the end-effector is picked-up or released to assure no open air lines.

The *Acquire* and *Release Delays* are set for the end-effector pick-up mechanism actuation time during the pick-up and release cycles. These values are to be entered in seconds.

The I/O reset routines are used to set default I/O settings for the quick-change end-effectors. Specify the optional programs in the *pickup routine* and *putdown routine* fields in the Quick-Change database. The pickup routine is called after acquiring a new end-effector and the putdown routine is called before returning an end-effector that is attached to the robot. For dispensing applications, the putdown routine might be used to make sure that the dispensing gun is turned off before placing the end-effector back into its nest. The I/O routines must conform to the calling sequence described below:

Quick-Change IO Debug

The Quick-Change software includes an additional menu for debugging I/O and to allow manual operation of the change hand mechanism. This menu can be found in the IO pull-down menu under the name QC CONTROLS. The QC CONTROLS menu also shows the binary inputs for the Hand Number in a LED type format with the least significant bit to the right side.

Table B-2 Quick-Change Database Fields

name String (15 characters)

A standard AIM name that uniquely identifies this tool for the tool changing utility. This name is referenced in the change_hand routine.

update date Date

The date and time when this record was last modified. This field is automatically set to the current date whenever information is changed by an operator.

tool-on-stand signal Integer

The number of the input signals indicating whether or not the corresponding quick-change end-effector is in its nest. If the value is negative, the logic is inverted. If the signal number is set to zero, no switch is assumed.

hand number Integer

Optional tool signature value that uniquely identifies an end-effector. This enables the tool acquisition and return routines to check that the proper tool is attached to the arm before continuing the Change_Hand sequence.

tool attach Integer

Optional tool attach database field specifies an input signal number that senses whether the tool is attached to the manipulator's wrist.

pickup routine String (15 characters)

A standard name that specifies the name of the subroutine that is called to set default end-effector I/O settings after acquiring an end-effector from its nest.

putdown routine String (15 characters)

A standard name that specifies the name of the subroutine that is called to set default end-effector I/O settings before returning an end-effector to its nest.

total signatures Integer

Optional total signatures database field specifies the number of bits required corresponding to the number of tools to be used in the system.

first signature Integer

Optional first signature database field specifies the first input signal number relative to the number of signatures specified in the total signatures field.

acquire tool Integer

The acquire tool database field specifies the output signal that is activated to lock the tool to the quick-change wrist.

release tool Integer

The release tool database field specifies the output signal that is to be activated to release the tool from the manipulator's wrist.

acquire delay Real

The acquire delay database field specifies the amount of time to be delayed after the acquire signal has been activated before departing from that position.

release delay Real

The release delay database field specifies the amount of time after the release signal has been activated before departing from the nest position.

gripper open Integer

Optional gripper open database field specifies a gripper signal to be turned off before releasing or acquiring the tool.

gripper closed Integer

Optional gripper closed database field specifies a gripper signal to be turned off before releasing or acquiring a tool.

Table B-3 QUICK-CHANGE 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	location name	string	15			
4	[location]	integer	2			
5	tool-on-stand signal	integer	2			
6	hand number	integer	2			
7	pickup routine	string	15			
8	put-down routine	string	15			
9	tool attach	integer	2			
10	total signatures	integer	2			
11	first signature	integer	2			
12	acquire tool	integer	2			
13	release tool	integer	2			
14	acquire delay	real	4			
15	release delay	real	4			
16	gripper open	integer	2			
17	gripper closed	integer	2			

Table B-4 Quick-Change Database Variable Name

Variable Name	Interpretation
qc.db	Quick-Change database number
cc.name	Field number for *name*
cc.update	Field number for *update date*
cc.device	Field number for *device*
qc.on.stand	Field number for *tool-on-stand signal*
qc.hand.num	Field number for *hand number*
qc.get.rtn	Field number for *pickup routine*
qc.put.rtn	Field number for *put down routine*
qc.tool.attch	Field number for *tool attach*
qc.total.sig	Field number for *total signatures*
qc.first.sig	Field number for *first signature*
qc.acquire.tool	Field number for *acquire tool*
qc.release.tool	Field number for *release tool*
qc.acquire.del	Field number for *acquire delay*
qc.release.del	Field number for *release delay*
qc.grip.open	Field number for *gripper open*
qc.grip.close	Field number for *gripper closed*
qc.loc.name	Field number for *location record name*
qc.location	Field number for *location record number*

B.3 Quick-Change Operation

This section describes the operation of the Quick-Change Module. First, the ***Change_Hand*** statement is reviewed. Next, details about the database menu and field inputs are discussed. Finally, an example using the Quick-Change Software is reviewed.

The CHANGE_HAND Statement

The statement performs automatic tool exchanges. The statement's syntax is:

```
CHANGE_HAND{{APPROACH -path-} OLD.TOOL -tool- {DEPART -path-}}
{{APPROACH -path-} NEW.TOOL -tool- {{DEPART -path-}}
{IF SIG ON -constant-}
```

The statement performs the following series of steps:

1. If an **old** tool is specified and is attached to the robot, place the tool in its nest. First, approach the old tool nest using the optional approach path. Next, move to the nest location and detach the hand. Finally, move away from the nest along the depart path if a depart path is defined.
2. If a **new** tool is specified and the hand is not already attached, pick up a hand at its nest location. First, approach the nest following the optional approach path. Next, move to the nest location and attach the hand. After picking up the tool, depart along the transit path if a depart path is specified.
3. If the optional IF SIG ON is specified, the statement will be executed if the specified signal is on. If the signal is off the statement will be by-passed.
4. The `-path-` arguments reference records in the Path database, specifying standard AIM paths. The `Quick-Change` argument is linked to the Quick-Change database records which contains the tool nest locations.

Note that the ***CHANGE_HAND*** statement can be programmed to simply return an end-effector, acquire an end-effector, or perform an entire quick-change task. If tool signature and nest presence signals are incorporated with the hardware, the ***CHANGE_HAND*** statement uses these signals to verify that the correct end-effector is attached to the robot and that the correct tool is in a tool nest before the robot exchanges tools. If an error condition in the workcell exists (e.g. an end-effector is not released into its nest), the robot stops and the operator is notified of the error. This prevents potential crashes.

B.4 Quick-Change Example

To review the topics discussed in this chapter, consider an application that requires 3 different end-effectors. This example will show the database requirements for 2 of the 3 tools, and show the sequence statements to pick-up tool 1, release tool 2, and finally release Tool #1 and pickup Tool #2. For this example, we are assuming all the optional tool inputs are being used, also the provided sample pickup and putdown routines are used.

Before editing the Quick-Change database, determine the required signal numbers to be used. For this example, we are using the following signals:

Table B-5 Change_Hand Example Tool I/O

Inputs	**Signals**	**Comments**
Tool-On-Stand signal (gripper #1)	1001	
Tool-On-Stand signal (gripper #2)	1002	
Tool Attached signal	1003	
First Signature Bit Signal	1004	Two bits are required for 3 tools
Outputs		
Aquire Tool signal	-2031	Internal signal for Adept arm valves
Release Tool signal	2031	
Gripper Open signal	9	
Gripper Closed signal	10	

Quick-Change Database

The Quick-Change database records required for 2 of the 3 end-effectors are shown in Figure B-2 and Figure B-3. This record is created during database editing, or automatically by the AIM linker during sequence start-up if the tool is specified in the sequence database. Important fields used by the ***CHANGE_HAND*** statement include:

- Location field. This is where the tool is stored when not attached to the robot. The robot executes a complete approach and depart sequence to acquire and replace the t.ool

- Pickup routine. This program is executed after the end-effector is picked up from its nest. The program might set the tool signal low so that the tool is not actuated when the tool is first attached to the robot. System customizers typically will write this program; the required calling sequence is described in Appendix B.
- 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.
- I/O signal definition. Both end-effector Quick-Change records show the signals that were described the the previous section.

Sequence Statements

The following ***CHANGE_HAND*** statements are used for the application described above. The first sequence statement will pick-up tool 1 from the tool stand.

CHANGE_HAND NEW.TOOL end_effector_1

The second sequence statement will place tool 2 back into the tool stand.

CHANGE_HAND OLD.TOOL end_effector_2

The third sequence statement will place tool 1 back and pick-up tool 2 from the tool stand.

CHANGE_HAND OLD.TOOL end_effector_1
NEW_TOOL end_effector_2

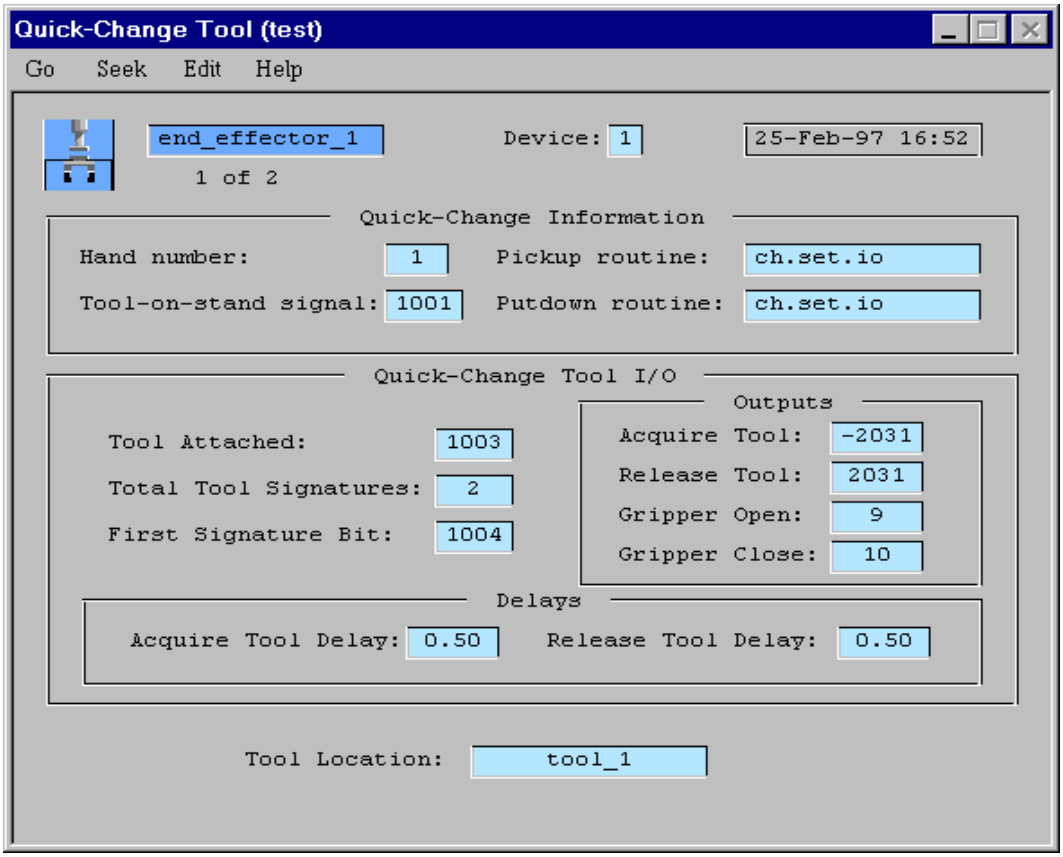


Figure B-2
Quick-Change Database Record End-Effector 1

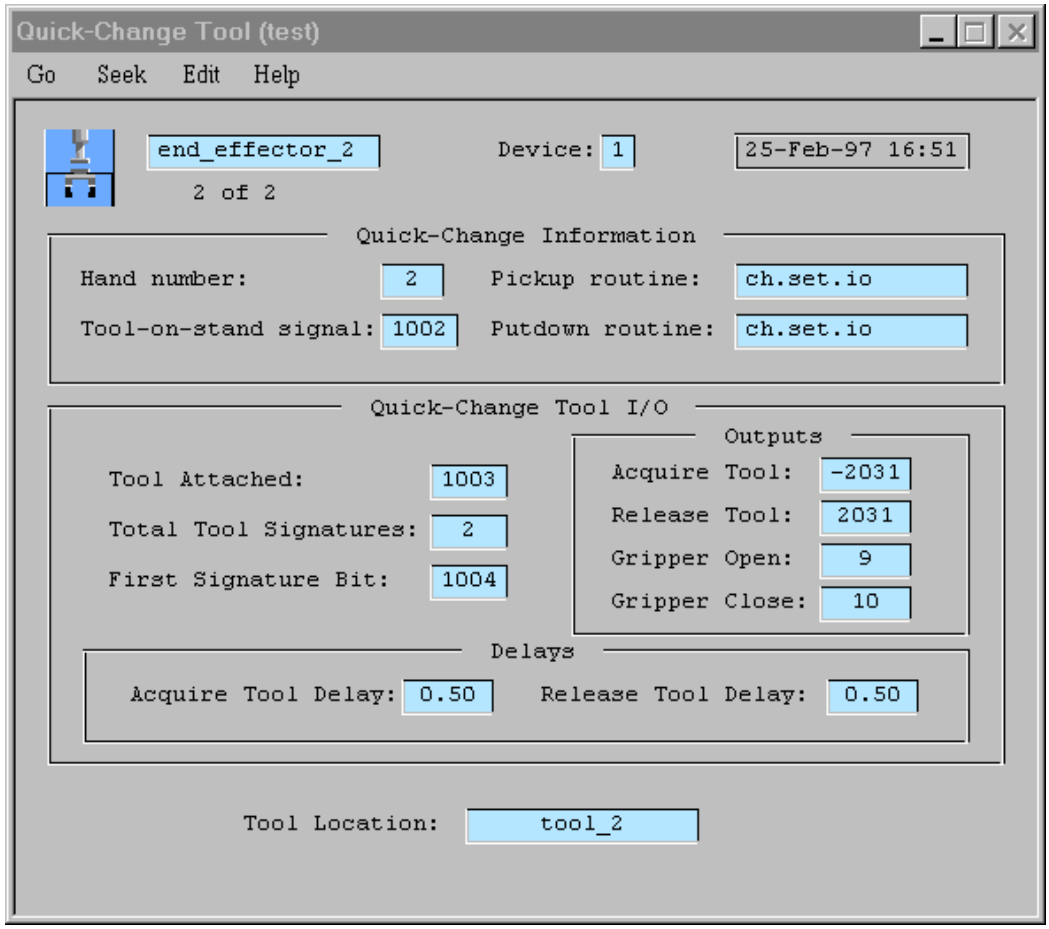


Figure B-3
Quick-Change Database Record End-Effector 2

B.5 Quick-Change Routines

The routines listed in Table B-6 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 B-6
Documented Routines

Program Name	Autoloaded File	Commented File	Purpose
ch.reset.io	RUN_CH.SQU	RUN_CH.V2	Sample tool.change I/O program
rn.ch.get	RUN_CH.SQU	RUN_CH.V2	Tool acquire runtime primitive
rn.ch.put	RUN_CH.SQU	RUN_CH.V2	Tool replace runtime primitive

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

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.

Input 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.

Output Parameters

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 Routines

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 modified Tool database for use with the CHANGE_HAND statement. The program sets output signals to default values before replacing or after acquiring an end-effector.

Input Parameters

None.

Output 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 "pickup routine") or before replacing a tool (if defined as a "putdown 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.

Calling Sequence

rn.ch.get(app.path.rec, ref, dep.path.rec, error)

Function

Runtime primitive routine for acquiring a hand from its stand. The CHANGE_HAND statement uses this primitive.

Usage Considerations

Assumes the appropriate Tool database record is open.

Input Parameters

app.path.rec	Real value, variable, or expression that defines the Path database number for the optional path to follow while approaching the tool nest location.
ref	Transformation variable, function, or compound transformation specifying a reference frame. If the tool nest locations are defined frame relative, they are defined with respect to *ref*.
dep.path.rec	Real value, variable, or expression that defines the Path database record number for the optional path to follow while departing from the assembly location.

Output Parameters

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

Details

This routine acquires a tool from its tool nest. The program performs the following steps:

1. If there is an approach path specified, begin moving along it.
2. Move to the approach location.
3. If tool signature bits and nest presence signals are used, check to make sure the end-effector is in its nest and verify that there is no hand attached to the robot.
4. Move to the tool acquire location and activate the hand acquire signals.
5. Depart from the pickup location.
6. Make sure the the correct tool is attached to the robot and the tool nest is empty.
7. Execute the "pickup routine" to initialize gripper I/O.
8. If a departure path is specified, move along it.

Related Routines

rn.ch.put

Calling Sequence

rn.ch.put(app.path.rec, ref, dep.path.rec, error)

Function

Runtime primitive to return a tool to its nest. The ***CHANGE_HAND*** statement uses this primitive.

Usage Considerations

Assumes the appropriate Tool database record is open.

Input Parameters

app.path.rec	Real value, variable, or expression that defines the Path database record number for the optional path to follow while approaching the tool nest location.
ref	Transformation variable, function, or compound transformation specifying a reference frame. If the tool nest locations are defined frame relative, they are defined with respect to *ref*.
dep.path.rec	Real value, variable, or expression that defines the Path database record number for the optional path to follow while departing from the assembly location.

Output Parameters

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

Details

This program returns an end-effector to its tool nest. The approach and depart heights used by this routine are the depart and approach heights, respectively, specified in Tool database. This prevents the robot crashes in the workcell.

The program performs the following steps:

1. If no hand is attached to the arm, or the hand is already in its nest, exit.
2. If an approach path is specified, move along it.
3. Move to the approach location.
4. If tool signature and nest presence signals are used, make sure the correct tool is attached to the arm and that no tool is sitting in the nest.
5. Execute the user specified "putdown routine." This sets the tool's default I/O signals.
6. Move to the tool nest location and activate the detach signals.
7. Depart and verify that there is no tool attached to the arm and that the hand is in its tool nest.
8. If a departure path is specified, begin moving along it.

Related Routines

rn.ch.get

Event I/O Manager C

C.1 Introduction and Overview

Provided with the AIM Utilities software is a utility to allow the user to monitor I/O events that occur. This software will allow you to look for the event to occur and monitor the time during the occurance. Time limits can be set and a alarm will be generated when the event exceeds the limits. This software has a logging feature which will the user to specify a description and a file name.

C.2 Event Monitor Editing

This section describes the menus used for adding events to the event monitor software package. You can reach this menu by selecting the ***EDIT*** pulldown and the ***Event*** option on the pulldown list.

Edit ➡ Event

The figure below is the Event monitor database page.

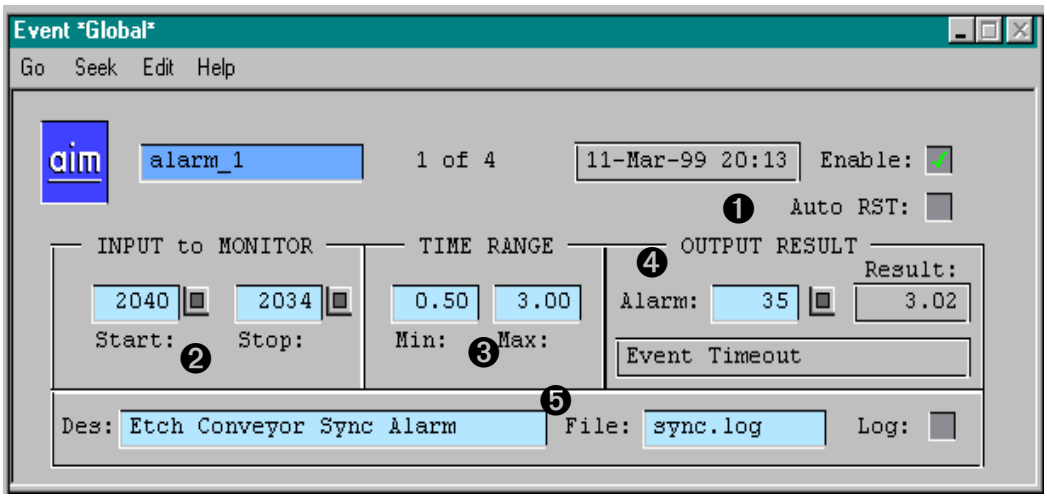


Figure C-1 Event Monitor Database

❶ ***Enable Selection:***
This check box will allow the event to be monitored when the Event Task is running in AIM.

Auto RST:
This check box will reset the alarm after the stop input has been set. This can reduce operator responses.

❷ **Start Input:**
The start input field is the input signal number to monitor for the event process to start. This will begin the timer to check for the proper event to occur.

Stop Input:
The stop input field will monitor the event until the input signal specified has turned into a true state. When this occurs the event monitoring will stop until the next start process has begun.

❸ ***Min Time Range:***
This field is used to set the minimum time the event process can occur. Events that occur in a shorter time will be flagged with an alarm.

Max Time Range:
This field is used to set the maximum time the event can occur. Events that occur in a longer period of time will be flagged with an alarm.

❹ ***Alarm Output:***
This output signal number will be turned on when a event exceeds the timing parameters.

Result Displays:
There are two displays that are also shown based on the last event occurance. The first display shows the actual event time that occurred. The second display shows the state message based on the last event.

❺ ***Event Alarm Logging:***
Alarm data can be logged to a specified file with a description statement provided in this database. The check box needs to be turned on to enable this feature.

C.3 Event Initialization Database

The event software provides an initialization database for setting default parameters. Currently only 2 items exist in this database. You can reach this menu by selecting the selecting the ***SETUP*** pulldown and the ***Initialization Data*** option. Then you must select the ***EVTINI.DB*** option from the resulting selection window.

Setup ➡ Initialization Data ➡ evtini.db

The following 2 records will appear:

1. Log Delimiter - This determines what charaters get placed between event text data in the log file that is specified. Currently the default is 9 which places a TAB between text data.
2. Statement_EVT - This record loads the statement database called ***statevt.db***. Currently no additional statements are provided with this software package.

C.4 Running the Event Monitor

The event monitor is a server that runs in task 18 as provided with this software package. This server needs to be executed by the user of the event monitor to be operational.

HPGL Graphics Module D

D.1 Introduction and Overview

The HPGL Plug-In allows the user to import Corel graphics files into a AIM menu static graphic area. Below is a set of instructions provided with this utility to load a HPGL file.

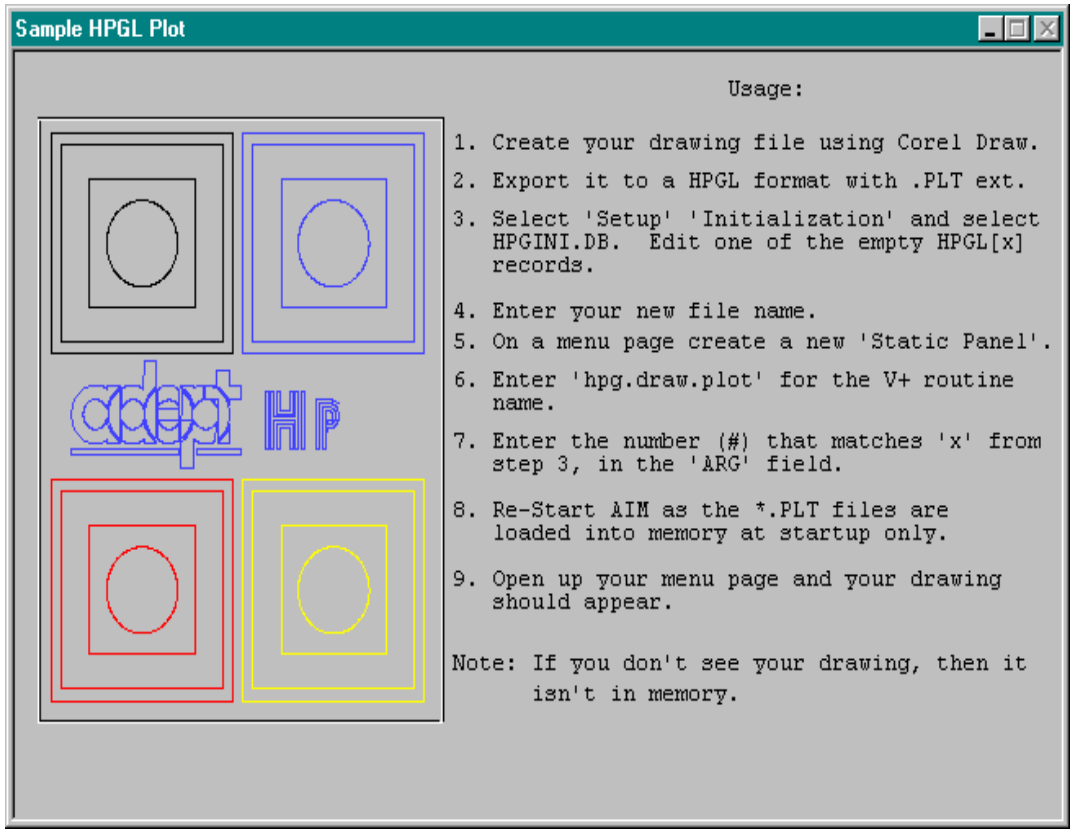


Figure D-1HPGL Instructions

ConnectWare Module E

E.1 Introduction and Overview

The ConnectWare is a fully integrated software package designed specifically for use with Adept Controllers. This software module allows network communications between multiple Adept Controllers. This software uses the DDE interface that is supplied with the AIM product. Generally, the PC or other device has a DDE server for the communication. This package transfers AIM Variable data via sequence instructions for the communication. This software is for use with Aim revision 3.1 and higher.

E.2 Sequence Statements

This section describes the sequence statements that are provided with the ConnectWare Plug-In software package. First, the statements and arguments are documented.

DDE_CONNECT

This statement will connect the controller to another Adept controller based on its IP address. The statement's syntax is as follows, where the braces ({ . . . }) define optional clauses:

DDE_CONNECTIP --string_var--HANDLE--variable--STATUS--variable--

The statement performs the following steps:

1. The ***IP*** argument specifies the IP address of the other controller.
2. The ***Handle*** argument returns back a logical unit number that can be used by the sequence for future communication.
3. The ***Status*** argument returns back the status of the network connection

DDE_DISCONNECT

This statement will disconnect the communication between the two controllers. The statement's syntax is as follows, where the braces ({ . . . }) define optional clauses:

DDE_DISCONNECTHANDLE--variable--

The statement performs the following steps:

1. The ***Handle*** argument allows the user to specify the Logical Unit Number of the device you wish to disconnect from.

DDE_READ

This statement will read the specified data from the other controller that has been attached via the logical unit number returned by the connection statement. The statement's syntax is as follows, where the braces ({ . . .}) define optional clauses:

DDE_READHANDLE--variable--FROM_VARIABLE--string_var--
IN_MODULE --string_var--PUT_IN_VAR--string_var--STATUS --variable--

The statement performs the following steps:

1. The ***Handle*** argument allows the user to specify the Logical Unit Number of the device you wish to read from.
2. The ***From_Variable*** argument specifies the variable name in the other Adept controller to read from.
3. The ***In_Module*** argument specifies the database module in the other Adept Controller to read from.
4. The ***Put_In_Var*** argument specifies where the data is written to in the variables database.
5. The ***Status*** variable returns success if the communication worked.

DDE_WRITE

This statement will write the specified data to the variable database in the other other Adept controller. The statement's syntax is as follows, where the braces ({ . . .}) define optional clauses:

DDE_WRITEHANDLE--variable--SEND --string_var--
TO_VARIABLE--string_var--IN_MOD--string_var--STATUS --variable--

The statement performs the following steps:

1. The ***Handle*** argument allows the user to specify the Logical Unit Number of the device you wish to write to.
2. The ***Send*** argument specifies the variable that will be used to extract the data for transfer.
3. The ***To_Variable*** argument specifies the variable in the other Adept controller the data will be written to.
4. The ***In_Mod*** argument specifies the database module in the other Adept controller where the other variable database resides.
5. The ***Status*** variable returns success if the communication worked.

TCP-IP V+ Server F

F.1 Introduction and Overview

This software module allows a PC or an Adept MV TCP CLIENT to retrieve information from the controller that is running this application. This allow some communication to occur when following the format shown below.

TCP Message Format

The TCp message must be constructed as shown in the format

{x | string}

Where 'x' is the type of message that is to occur based on the list below.

1. x = 's' - evaluate a string expression

   Example: '**{s | $ai.ctl[1]}**'

2. x = 'r' - evaluate a real variable

   Example: '{r | ai.ctl[1]}'

3. x = 'f' - evaluate a function

   Example: '{f | switch(power)}'

   Example: '{f | bits(1001,8)}'

4. x = 'o' - perform a function

   Example: '{f | x=100}'

5. x = 'aim' - read an AIM variable

   Example: '{aim | test \ var_1}

   module<---^ ^----->variable

The return message is the evaluated value in 'ASCII' format without any formatting characters. The server will echo the original message following the results.

Example: Request '{f | switch(power)}'

Result '{f | switch(power)}{-1}'

F.2 TCP Debug Menu

Provided with this software package is a bebug panel to allow the user to type in messages and veiw the response back from the other controller or device.

To veiw the TCP Debug panel, select the Setup pulldown and the TCP Debug option.

Setup ➡ TCP Debug

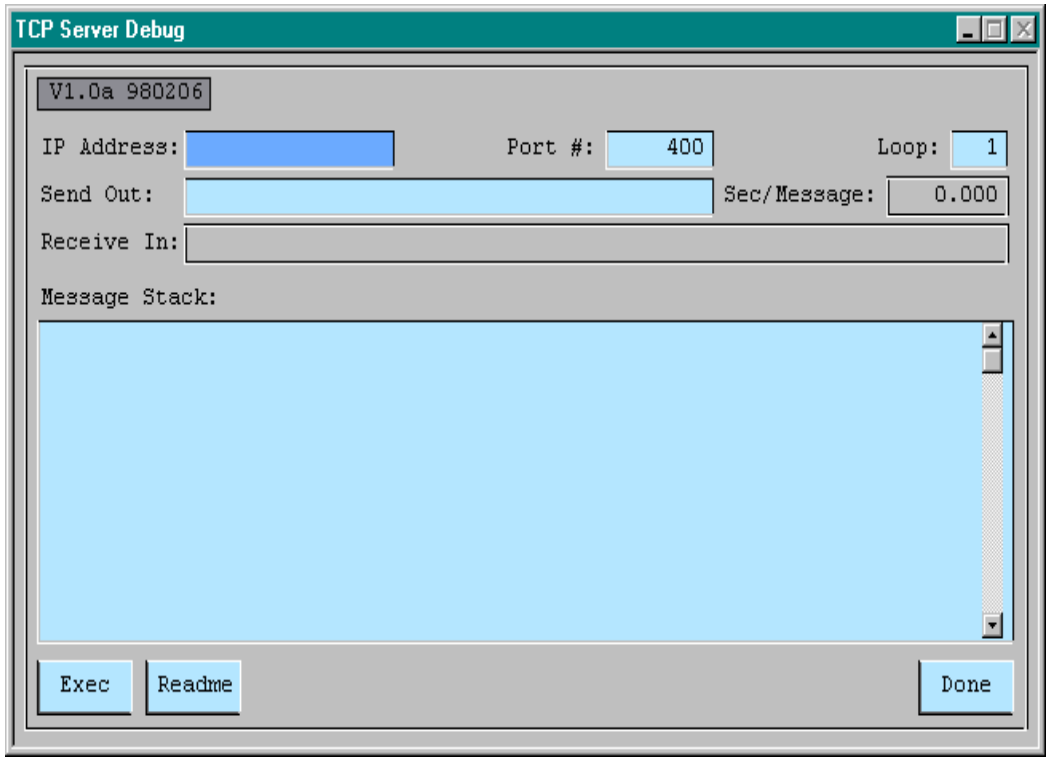


Figure F-1TCP Debug Window

F.3 TCP Initialization Database

Provided with this software package is a initialization database to allow the user to configure to operation of the TCP software package. Listed below are the items available in the database provided with this software.

1. ***Statements_TCP***- This record in the database loads a statement database for customized statements for TCP. Currently no statements are available.
2. ***TCP Clients Allowed*** - This is the maximum number of simultanious connections to the TCP Server. Maximum is 15 -- Default is 5
3. ***TCP Port Number*** - This the TCP port number used for the communication between the devices. The current default is 400.
4. ***TCP Task*** - This the task number the server will run in. Currently defaulted to 18.

Index

Adept 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. Thank you for your input.

NAME _____DATE_____

COMPANY _____

ADDRESS_____

PHONE _____

MANUAL TITLE: _____

PART NUMBER and REV level: _____

COMMENTS:

_____

_____

_____

_____

_____

_____

_____

_____

_____

MAIL TO: Adept Technology, Inc.
Technical Publications Dept.
11133 Kenwood Rd
Cincinnati OH 45242

FAX: (513) 792-0274

adept
technology, inc.

150 Rose Orchard Way
San Jose, CA 95134
408•432•0888

RDASG-C0002, Rev. B.