USER MANUAL

Accessory 9WN

PMAC Executive for Windows

3xx-32PWIN-xUxx

October 30, 2003

DELTA TAU
Data Systems, Inc.
NEW IDEAS IN MOTION ...

Single Source Machine Control *Power // Flexibility // Ease of Use*
21314 Lassen Street Chatsworth, CA 91311 // Tel. (818) 998-2095 Fax. (818) 998-7807 // www.deltatau.com

Copyright Information

To report errors or inconsistencies, call or email:

Delta Tau Data Systems, Inc. Technical Support
Phone: (818) 717-5656
Fax: (818) 998-7807
Email: support@deltatau.com
Website: http://www.deltatau.com

Operating Conditions

All Delta Tau Data Systems, Inc. motion controller products, accessories, and amplifiers contain static sensitive components that can be damaged by incorrect handling. When installing or handling Delta Tau Data Systems, Inc. products, avoid contact with highly insulated materials. Only qualified personnel should be allowed to handle this equipment.

In the case of industrial applications, we expect our products to be protected from hazardous or conductive materials and/or environments that could cause harm to the controller by damaging components or causing electrical shorts. When our products are used in an industrial environment, install them into an industrial electrical cabinet or industrial PC to protect them from excessive or corrosive moisture, abnormal ambient temperatures, and conductive materials. If Delta Tau Data Systems, Inc. products are directly exposed to hazardous or conductive materials and/or environments, we cannot guarantee their operation.

Table of Contents

INTRODUCTION

Overview

Welcome to PEWIN32, Delta Tau's PMAC Executive Software for Microsoft Windows. PEWIN32 enables you to configure, control and trouble-shoot your PMAC(s).

PEWIN32 is designed as a development tool for creating and managing PMAC implementations. It provides a terminal interface to the PMAC, and a text editor for writing and editing PMAC motion programs and PLC programs. Additionally, PEWIN32 contains a suite of tools for configuring and working with PMAC and its accessories including interfaces for jogging motors, extensive system utilities, screens for viewing various PMAC variables and status registers.

Features

This Windows version of the PMAC Executive software is based on the previous versions but has been enhanced to take advantage of the Microsoft Windows 32 bit operating systems.

- Multi-threading of PEWIN32 of realtime displays.
- Enhanced graphing in PmacPlot, a new standalone plotting application.
- Better management of memory.
- A thread safe communications driver makes this version is also compatible with the 32 bit version of "NC for Windows" software (and any application using PComm32 or PTalk, Accessory 9PN or 9PT respectively).

PEWIN32 helps you overcome many of the obstacles encountered when developing a PMAC-based application. Typically, the first thing a person wants to do with a PMAC is figure out how to talk to it reliably, and in a Windows environment this basic step can be quite challenging. PEWIN32 enables you to get your card up and running, right out of the box. Then the fun really begins.

You use PEWIN32 in the same manner as our other Executives: setting-up I parameters, configuring and jogging motors, writing and testing motion programs and general trouble-shooting.

PEWIN32 provides basic tools to configure, control and diagnose PMAC, here is a partial list of PEWIN32's features and capabilities:

- A terminal.
- Easy handling of PMAC's thousands of I,P,Q and M-variables, including macro support.
- Watch window for real-time system information and debugging.
- Motor, Coordinate System and Global status windows that display PMAC's status bits in real-time.
- Position window for displaying the position, velocity and following error of all motors on the system.
- Several ways to tune PMAC systems.
- Simplified interface for data gathering and plotting.
- Diagnostic routines for debugging motors and motion programs.
- Real-time status display of all PMAC's connectors.
- The ability to talk to multiple PMACs on a single computer.
- Cloning of one motor's parameters to another.
- and many more.

PEWIN32 is full of tools to help both the novice and the advanced PMAC user get the most out of their PMAC.

Manual Layout

This manual explains how to use PEWIN32 to communicate and control your PMAC motion control card. Knowledge of the basic use of the Windows operating system is assumed.

Conventions Used in this Manual

The following typeface conventions are used throughout the manual:

<ENTER>	Italic text inside arrows is used to represent keyboard
<CTRL+F4>	keys or key combinations.
`OPEN PROGRAM`	Mono-spaced used for code listings.
I VARIABLES	Small-Caps bold, underlined for menu items.
Bus Address	Italic text for dialog-box items.

Special functions or information.

How this Manual is Organized

Chapter 1 - Getting Started

Covers installing the software and establishing initial communications with PMAC.

Chapter 2 - Menu Overview

Gives a brief description of each of PEWIN32's menus.

Chapter 3 - Basic Concepts

Discusses everyday use of the software

Chapter 4 - PMAC Configuration Modification Utilities

Detail coverage of all PEWIN's configuration interfaces. Covers manipulation of variables, coordinate systems, encoder tables, etc.

Chapter 5 - Plotting and Data Gathering

Covers PEWIN's different methods for data gathering, plotting and analysis.

Chapter 6 - Configuration Files

Discusses how to backup and restore all or part of PMAC32's configuration.

Chapter 7 - PEWIN32 Software Configuration

Describes how to modify the appearance of **PEWIN32**.

Chapter 8 - PEWIN32 Help Facility

Details PEWIN's help and diagnostic routines.

Chapter 9 - Motor and System Tuning with PEWIN32

Detail descriptions of PEWIN's Tuning interfaces.

Appendix A - Understanding PMAC's Watchdog Timer

Discussion of the PMAC watch timer and its functions.

Appendix B - Bug Report and Feature Request Procedure

Forms to use to report program inconsistencies and feature requests.

Hardware and Software Requirements

The PMAC Executive for Windows software will run on any computer capable of running Windows 95-98 or NT (TM) (120MHz Pentium and up recommended). Of course, the faster the computer the better. In addition, you will need the following:

Microsoft Windows 95-98 or NT (4.x and up) loaded on your computer.

At least eight MB of free disk space and 16-24 MB of RAM (16MB for Windows 95-98 or 24 MB for Windows NT).

A free serial communications port , or USB port, or PCI-BUS slot, or ISA-BUS slot to talk to PMAC for on-line processing..

Any monitor with VGA resolution (800x600 suggested but 640x480 works fine).

GETTING STARTED

Installing PEWIN32

Before installing PEWIN32, read the license agreement included in this manual (behind title page), and make a backup copy of the installation disks. To install PEWIN32, put the PEWIN32 distribution disk labeled “Disk #1” into a floppy drive and choose **File | Run** from the Program Manager. Enter **A:\SETUP.EXE** or substitute ‘**A**’ for the letter of your floppy drive.

The installation program will suggest a directory path where the program files should be copied. Please use the suggested directory location for the installation for the purposes of uniformity among all PEWIN32 users (and trouble shooting if need be).

Read the “readme.txt” file for last minute additions to this manual.

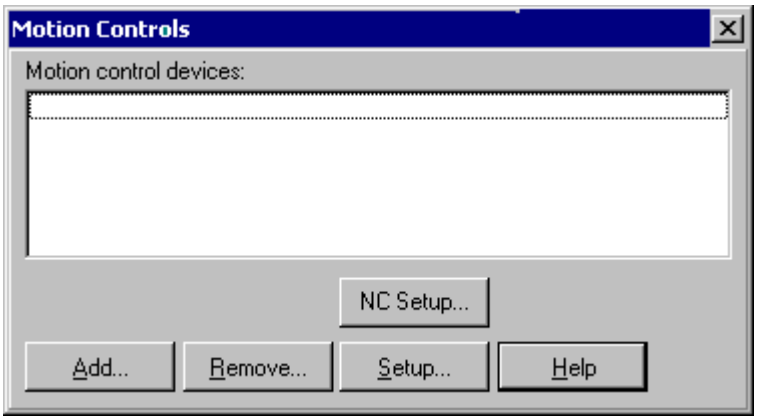
You will want to setup communication before running PEWIN32 for the first time. For details on setting up communications see Setting up Communications with PMAC.

Setting up Communications with PMAC

No applications, including PEWIN32, will be used to add, remove or configure PMAC's in your system. Rather, communication settings have been centralized in your operating system, making the set up of each PMAC much like other devices in your computer (i.e. video card, sound card etc.). All setup is done through the "MOTION CONTROLS" applet (or the "MotionExe.EXE" application) that is accessible through your operating systems CONTROL PANEL. Before running this application, it is important that all applications that use PComm32 (the Delta Tau 32 bit communication driver) be shut down. This includes PEWIN32, NC for Windows, and any applications developed with PComm32 or PTalkDT.

Once you have the control panel open, click on the icon shown below.

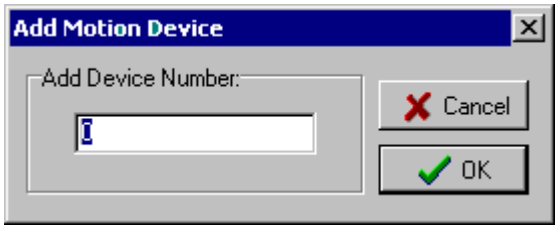
The following dialog box comes up:



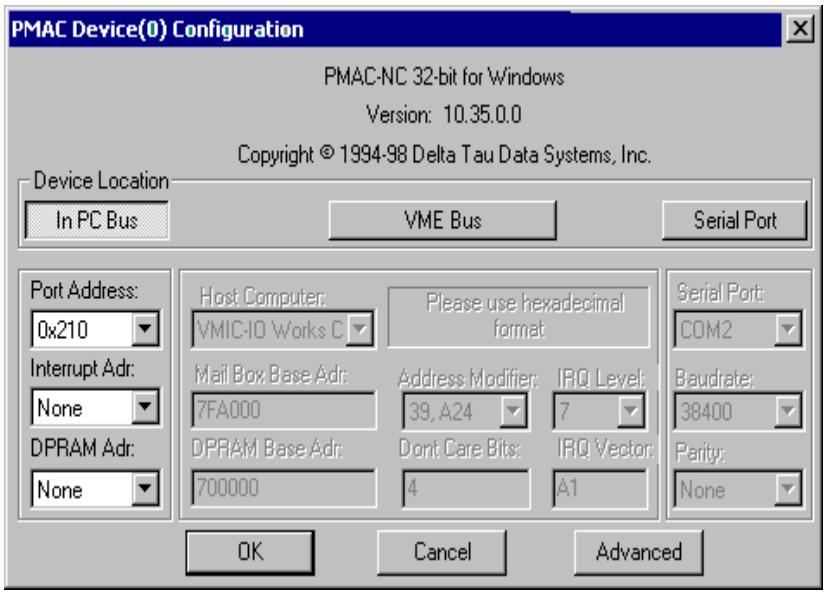
Note

The Unload, Load, and Startup buttons only come up if you are running Windows NT. Typically the Load and Unload are never used but in rare trouble shooting cases. The "Startup" may be used to tell the operating system how to load PMAC communication driver (PComm32).

If this is your first time running the applet there will be no PMAC's listed in the "Motion control devices" list box. This is because none have been "Added" to your operating system yet. To add a PMAC press the "Add" button to get the following dialog box:



This dialog box is prompting you for a device number to associate with the PMAC you are adding. Always start with your first PMAC as Device 0, the second PMAC in your system as Device 1 and so on. The applet will handle the enumerating for you. Press OK, to get the configuration dialog box:



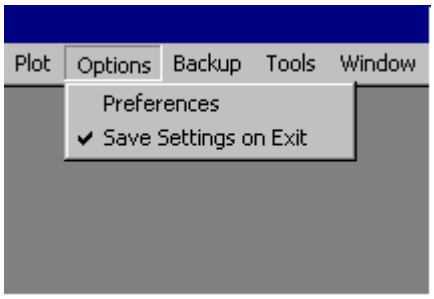
This is where you specify how PMAC is connected to your system, and the communications settings. The key point here is that the configuration you set up must match the hardware jumper settings on the PMAC itself.

The Advanced button is used to configure DPR settings used with the Delta Tau NC for Windows software.

Exiting PEWIN32 and Saving Options

To exit PEWIN32, press the *<ALT+F4>* keys or select **Exit** from the **File** menu. If any open files have not been saved, you will be prompted for saving them.

If the **Save Settings on Exit** options is selected, PEWIN32 will save the current desktop arrangement to a file named PEWIN32.DKT in the working directory.



The next time the program is executed, it will start with the arrangement it had upon exiting.

Help Features

Note

Context sensitive help is currently under construction. The help menu though should function just fine.

You access help in PEWIN32 just like all other Windows programs.

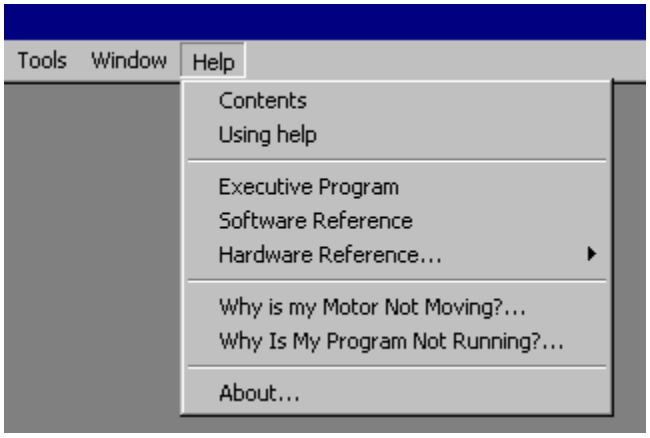
1. Press <*ALT+H*> or select the help option on the main menu to bring up the Help menu.
2. Press <*F1*> to display the Help index directly, without going through the Help menu.

In addition to standard Windows Help, PEWIN32 provides context-sensitive Help. There are two ways to access this feature:

If you want to know more about a menu item you've selected, press <*F1*> to directly access Help for that item.

Press the *Help* button located in the Window or Dialog box to display help on that particular area of the program.

All of the PMAC manuals are available on-line.



WHY AM I NOT MOVING? and **WHY IS MY PROGRAM NOT RUNNING?** are the two diagnostic functions in the Help menu. These are only available if you are communicating with a PMAC. These will probably save you many hours of searching and frustration in tracking down problems associated with running a motor or part program especially if you are new to PMAC. You may run into a condition where a motor simply will not run, despite being sure that all the parameters have been set properly. These functions will look at your configuration and cite possible (warnings) or definite (faults) causes of problems preventing you from running your motor(s) / program(s). "Why am I not moving?" requires

that you specify a motor to analyze while “Why is my program not running?” requires that you specify a coordinate system to analyze.

The **ABOUT...** box will give you detailed information about the version of the software you are running.

Technical Support

Delta Tau is happy to respond to any questions or concerns you have regarding the Windows Executive. By far, you'll get the quickest response if you send your queries to the following e-mail address: support@deltatau.com. Of course, check our Web site at WWW.DELTATAU.COM.

You can call Delta Tau Monday through Friday from 9:00 am to 4:00 PM PST or FAX us your request or problem, and we will deal with it the next business day.

Delta Tau Data Systems, Inc.
21314 Lassen Street
Chatsworth CA, 91311

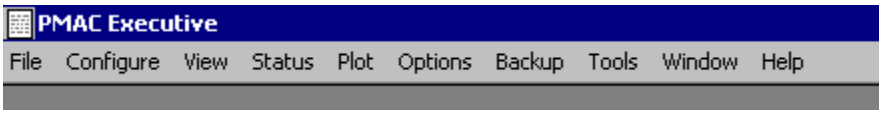
West Coast
Voice : (818) 998-2095
FAX : (818) 998-7807

East Coast
Voice : (804) 795-4288
FAX : (804) 795-4996

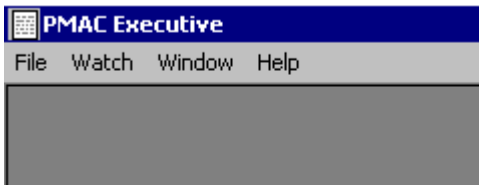
MENU OVERVIEW

How Menus Work

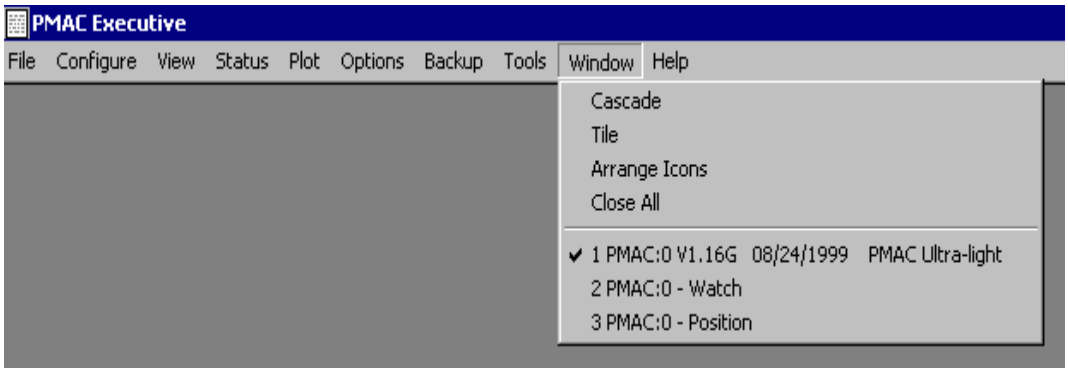
PEWIN32 uses a dynamic menuing system. This means that the menu at the top of the screen changes content depending on what window has the current focus (is highlighted). The standard menu displayed when the terminal has focus looks like this:



but the menu will change when for example the Watch Window is highlighted -



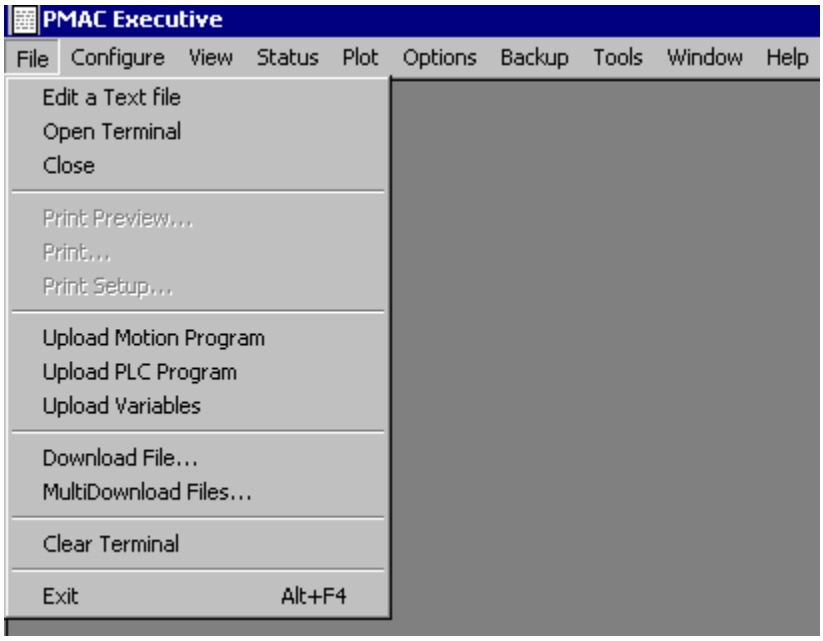
If an option you expect to be available isn't, make sure you have the proper window highlighted. There will always be **WINDOW** and **HELP** menu items available, you can use the **WINDOW** menu to select the window you need to work with.



Also, note that there are very user-friendly "context sensitive" menu's which pop up in many of the PEWIN32 Windows. Use the right mouse click over a window to see what options are available in the context sensitive menu.

File Menu

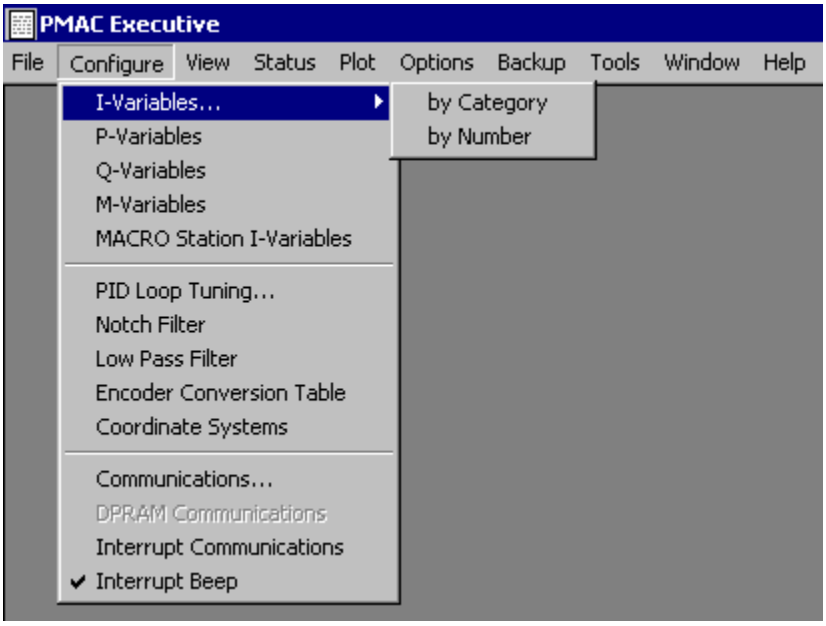
The File menu handles the transfer of files or programs to and from PMAC as well as printing of these files.



EDIT A TEXT FILE - will allow you to create a new PMC, or PLC file.
OPEN TERMINAL - will open a Terminal
CLOSE - will close the highlighted terminal and any windows associated with it.
PRINT PREVIEW - works just like other windows programs and shows you what the terminal would look like if you were to print it.
PRINT - allows you to print the terminals contents.
PRINT SETUP - configures your printer.
UPLOAD MOTION PROGRAM - will upload the specified PMC program into an editor window.
UPLOAD PLC PROGRAM - will upload the specified PLC program into an editor windows.
UPLOAD VARIABLES - will allow you to upload a range of I,P,Q or M variables into an editor window.
DOWNLOAD FILE - allows you to download any file with valid PMAC commands.
MULTIDOWNLOAD FILE - allows the user to manage and download files sets.
CLEAR TERMINAL - this will clear any text in the terminal window
EXIT - closes the program.

Configure Menu

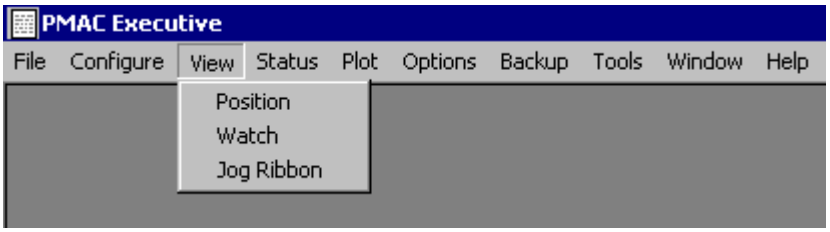
The **CONFIGURE** menu lets you view and change current variable definitions and PMAC feature parameters. In some of the options that allow you to change a value or definition, the change is sent to PMAC immediately after the value is changed. This allows you to automatically verify that the change in the input field has also occurred in PMAC. It also protects against faulty entries since out of range numbers will not be accepted.



I VARIABLES - there are two interfaces for listing and setting I variables; by category, or by numerical order.
P VARIABLES - allows the user to set P variable values.
Q VARIABLES - allows the user to set Q variable values.
M VARIABLES - allows the user to set M variable definitions and values.
MACRO STATION I-VARIABLES – allows the user to set MI variable values.(ultralight P-Mac family only)
PID LOOP TUNING… - allows the user to set the control parameters or gains of PMAC's PID filter.
NOTCH FILTER - allows the user to set up a notch filter.
LOW PASS FILTER - allows the user to set up a low pass filter.
ENCODER CONVERSION TABLE - this menu choice allows you to alter, update, save or retrieve the entries of the encoder conversion table.
COORDINATE SYSTEMS - this menu choice enables you to alter the currently defined coordinate systems, or define new ones.
COMMUNICATIONS - This will give instructions on changing communication settings.
DPRAM COMMUNICATIONS - If your PMAC has the Dual-Ported Ram option, this will toggle the use of DPRAM for ASCII communications.
INTERRUPT COMMUNICATIONS - If you have the PMAC setup for interrupts, setting this option will display any interrupts generated.
INTERRUPT BEEP - If selected, a beep will sound every time your PMAC issues and interrupt.

View Menu

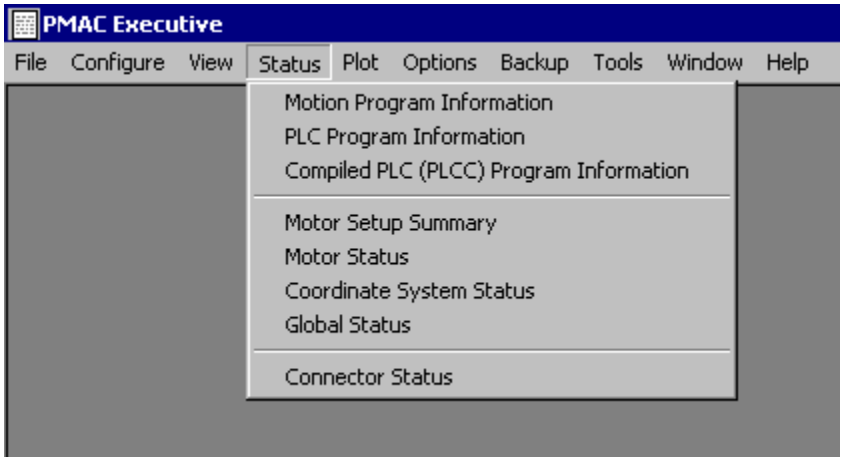
The **VIEW** menu contains interactive status displays and motor movement tools.



POSITION - opens a position display window. You can have multiple position windows open.
WATCH - opens a watch display window. You can have multiple watch windows open.
JOG RIBBON - brings up a simplified motor movement interface.

Status Menu

The **STATUS** menu allows you to view how programs, motors, coordinate systems, global parameters and connectors are organized and functioning in PMAC. Many of the options open windows which allow you to view how the parameters change in real-time.



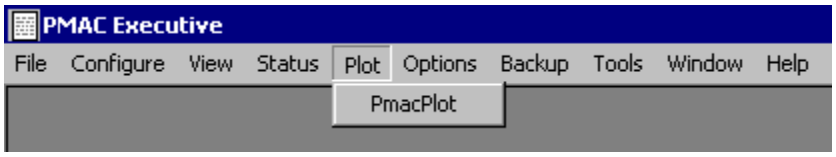
MOTION PROGRAM INFORMATION - displays the program number, starting address and size of all "programs" in PMAC's memory.
PLC PROGRAM INFORMATION - displays the PLC number, starting address and size of all non-compiled plcs in PMAC's memory.
COMPILED PLC (PLCC) PROGRAM INFORMATION - displays the PLCC number, starting address and size of all compiled plcs in PMAC's memory.
MOTOR SETUP SUMMARY - displays the configuration of a specified motor.
MOTOR STATUS - displays the interpretation of status bits associated with a specified motor in real-time.
COORDINATE SYSTEM STATUS - displays the status of the specified coordinate system in real-time.
GLOBAL STATUS - displays the interpretation of the global status bits in real-time.

CONNECTOR STATUS - allows the user to monitor the status of PMAC's connectors. Currently supported connectors are:

J2 (JPAN)	(PMAC1 & PMAC2 series)
J3 (JTHW)	(PMAC1 & PMAC2 series)
J5 (JOPT)	(PMAC1 series only)
J7 (JMACH2)	(PMAC1 series only)
J8 (JMACH1)	(PMAC1 series only)

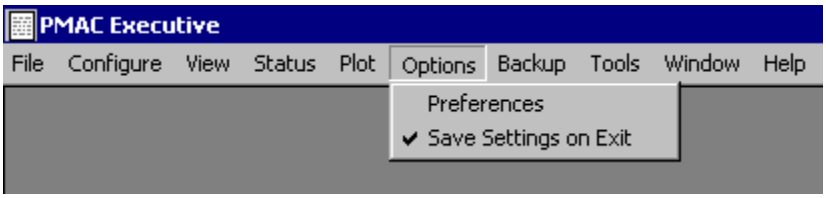
Plot Menu

Use the **PLOT** menu to launch the PmacPlot.EXE application.



Options Menu

This pull down menu allows you to customize PEWIN32. The menu choices provide the ability to set editor, terminal and plot display preferences. You can also determine how PEWIN32 acts on startup and shutdown.



PREFERENCES - allows the customizing of PEWIN32's features.
SAVE SETTINGS ON EXIT - if selected, the next time the Executive is started, it will be configured as it was upon exit.

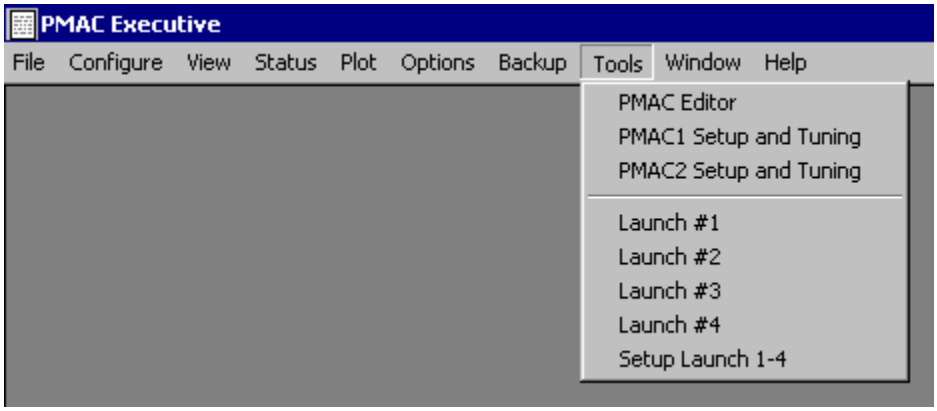
Backup Menu

The options in this menu allow you to save or restore various portions of PMAC configuration.

SAVE CONFIGURATION - allows the user to save all or part of PMAC's configuration to a disk file.
RESTORE CONFIGURATION - allows the user to restore all or part of PMAC's configuration from disk.
VERIFY CONFIGURATION - verifies a specified configuration file.

Tools Menu

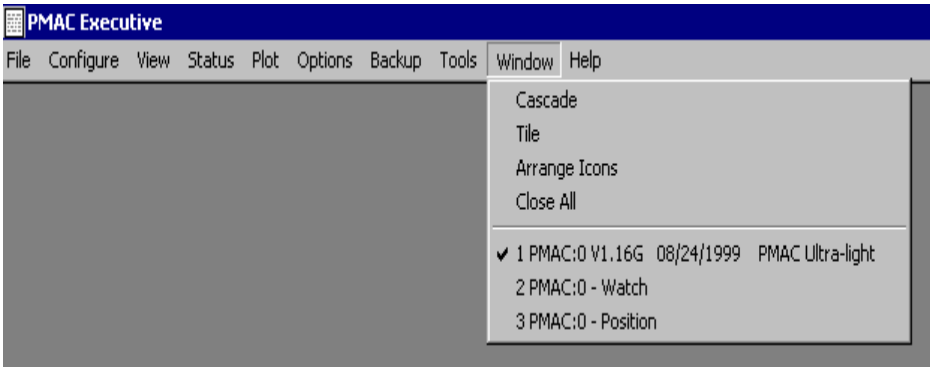
This menu is for managing the position and arrangement of any windows currently displayed.



PMAC EDITOR - will allow you to create a new PMC, or PLC file.
PMAC1 SETUP AND TUNING- starts the PMAC1 setup program.(if it is loaded)
PMAC 2 SETUP AND TUNING - starts the PMAC2 setup program.(if it is loaded)
LAUNCH #(1,2,3,4) – special setup motor # (1,2,3,4).
SETUP LAUNCH 1-4- special setup motor # (1,2,3,4).

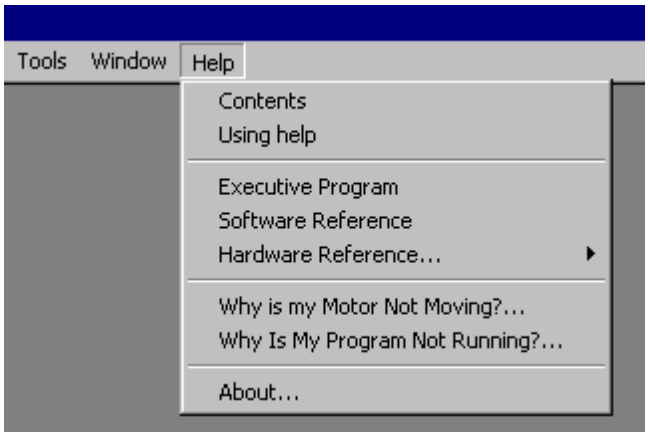
Window Menu

This menu is for managing the position and arrangement of any windows currently displayed.



Help Menu

The **HELP** menu options allow you to retrieve on-line information about PMAC, the Executive program, and the various help functions. You also have access to the two diagnostic routines provided by PEWIN32.



CONTENTS - displays the contents of the PEWIN32 help system.
USING HELP - a brief introduction to using the PEWIN32 help system.
EXECUTIVE PROGRAM - this manual.
SOFTWARE REFERENCE - the PMAC Software Manual and PMAC Users Guide in help format.
HARDWARE REFERENCE - each of the PMAC hardware manuals in help format.
WHY AM I NOT MOVING ? - a diagnostic routine to help determine why a motor is not responding.
WHY IS MY PROGRAM NOT RUNNING ? - a diagnostic routine to help determine why a program is not running.
ABOUT - displays information about your version of PEWIN32, including the version number, the copyright, legal and licensing notices.

BASIC CONCEPTS

Terminal

The text colors described here are the default values. These can be changed by using the "Options" Menu-Terminal Preferences screen.

The Terminal represents a direct connection to a PMAC. This is the basic mode of operation for the Executive Program. Any characters you type at the keyboard are sent to PMAC after pressing <*ENTER*>. Any characters that are sent from PMAC to the PC are displayed on the screen in a color corresponding to the current communications mode. If any command is rejected by PMAC, an error code will be shown as well as its description (and possible remedies) displayed in red text (assuming I6 is set to 1, the default). In addition, as you use the various screens in the Executive, you may notice some green text written to the terminal window. This text comes from the Executive notifying you that a change has been made to PMAC (such as an I-variable change). You should always read this text, as it may affect your application.

Accessing Previously Type Commands

One of the features of the Terminal window is the ability to recall previously typed commands, optionally edit them, and re-send them to PMAC without having to retype the entire command phrase. This is especially useful when, for example, you constantly have to type in a lengthy command phrase such as: `RHY$C000,20`. To access previously typed in commands, simply press the up or down arrow keys and the current line will cycle through the list. Simply make the necessary changes or corrections and press <*ENTER*>. The command phrase will be sent to PMAC.

Terminal Status Bar

Along the bottom of the Terminal window is a status bar containing, the currently addressed motor, the currently addressed coordinate system and its current feedrate override value (NOTE:For speed considerations, this is updated once every 10 seconds). This option can be disabled via the **OPTIONS** | **PREFERENCES** menu item.

Changing the Appearance of the Terminal

Under the **OPTIONS** | **PREFERENCES** menu item is a section for setting the terminal preferences. You can set the font, and display colors, and enable or disable the terminal status bar update.

Editor

PEWIN32 includes a simple Windows text editor. With this interface, you can create and edit program files and then download them to PMAC. This interface includes all the features of a “Notepad” type of program. You can cut, copy, delete and paste highlighted text. You can use the Windows clipboard to copy text from one Windows program to another. Basic search and replace functions are provided. For a more robust editor, we suggest you use your favorite code editor or word processor.

Downloading Files to PMAC

Downloading allows you to transfer the contents of a file on disk to PMAC's memory. This is handy for transferring new or modified PLC or motion programs and/or variable values and definitions to PMAC. It is highly suggested you suspend any plcs or motion programs during a download.

There are three different ways to transfer files from a hard or floppy disk to PMAC:

- Use the "Download File" menu item in the "*File*" menu.
- Download directly from an editor window.

- Use the "Multi-File Download" menu item in the "*File*" menu.

Using Macros in the Editor

While using any text editor window, you have the ability to define your own commands by use of user-defined macros. If you program in Pascal or C, you may be already familiar with the use of macros and appreciate their powerful usefulness. By using macros, you can code PMAC programs to read in English, making it much easier to interpret what a motion or PLC program is doing without having to look up every command in the PMAC user's manual. A macro definition is nothing more than a substitution name (that you invent) which is used in place of any valid PMAC command (like `GATHER` or `DWELL`), command phrase (like `OPEN PROG 1` or

`M1->Y:$FFC2,8,1`), or variable (like `P1`, `M22`, `Q342`, etc.). At the beginning of your program file (or at least before you actually use your macro definitions), you declare your macro definitions using the `#define` command (which can be lower or upper case). Examples of using this command are:

```
#define pressure P1
#define turn_on_pump M1=1
#define collect_data GATHER
#define seconds NULL
```

***Note*: The "File Contains Macros" box in the Download Options dialog must be checked for the program to properly process "#define" and "#include" statements.**

When the PMAC Executive Program is downloading your file from the text editor, if the `#define` command is encountered and if the File Contains Macros or PLCC's box is checked in Download Options dialog box, the definition is stored in PC memory (nothing is actually sent to PMAC). When the macro name is encountered later on the file, the PMAC Executive Program will send the actual PMAC command phrase/variable for the particular macro so that PMAC only sees a valid PMAC command/variable (instead of sending `pressure` to PMAC, `P1` is sent for the above example). The only exception to this when a macro is defined as a `NULL`. In this case, nothing is sent to PMAC-- it's merely available to provide units to numeric values for further macro elaboration (see the sample program listing below for examples on how to use the `NULL` definition). Remember that PMAC never sees the macro names-- when the downloaded program is listed, you will see standard PMAC commands.

A few rules must be followed when declaring and using these macro definitions. The macro name may contain any unique sequence of characters (upper/lower case letters, numbers, symbols, etc.), but must be separated from the `#define` command and the macro's definition by a space (as seen in the above examples). The macro name (like `gas_pressure`, `turn_on_pump`, `collect_data`) may not contain any spaces. You may, however, use the underscore character `_` to separate words for ease of reading. The PMAC Executive Program will differentiate between upper and lower case for the macro names, so take care when mixing upper and lower case letters. If you try to use a macro definition that has not been previously defined properly (because you misspelled it or the upper/lower case letter sequence don't match, i.e. pressure and Pressure), the program will download that line as is and may (and most probably will) result in an error generated by PMAC. Also, remember not to use the same name twice for two different macro definitions.

You may also use macro definitions contained in other files on disk instead or in addition to existing definitions in the file you are downloading. The command to do this is the `#include` command:

```
#include "macro.pmc"
#include "names"
#include "program.def"
```

You may want to create a file which contains nothing but macro definitions, and then have several motion programs "include" these definitions so that they can use the macros. This can help keep the size of program files small when a large number of macro definitions are being used. The rule to follow when using the `#include` command is that the file name (any legal DOS file name) must be enclosed in quotes and must be separated from the `#include` command with a space. The same rules stated above for naming the macros also apply here, including the amount of PC memory used.

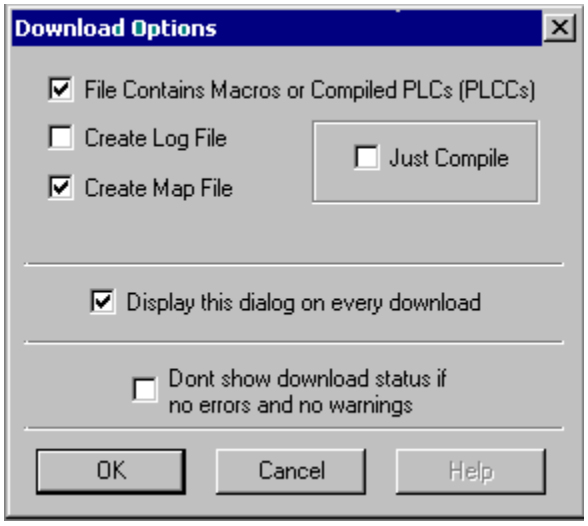
To best illustrate the potential of using macros, here is an example program using cleverly defined macro names:

```
#define msec                        NULL
#define revs/sec                    NULL
#define set_acc_time_of             TA
#define set_S_curve_time_of         TS
#define set_feedrate_of             F
#define set_move_time_of            TM
#define feedrate_time_units         I190
#define setup_gather_pointer        M88->X1,18 M89>X1,19
#define free_up_memory_space        DEL GAT DEL TRACE
#define reserve_memory_for_gather   DEFINE GAT
#define start_gathering_data        M88=1M89=1
#define stop_gathering_data         M89=0
#define motor1_is_axis              #1->
#define select_coordinate_system    &
#define clear_coordinate_system     UNDEFINE
#define move_X_to_position          X
#define move_Y_to_position          Y
#define move_Z_to_position          Z
#define sit_there_for               DWELL
#define repeat_as_long_as           WHILE
#define end_of_loop                 ENDWHILE
#define increment_repetition_count  P1=P1+1
#define repetition_count            P1
#define repetition_limit            3
#define begin_program               OPEN PROG
#define end_program                 CLOSE
```

```
;Now, let's use our new macro definitions!...
free_up_memory_space
setup_gather_pointer

select_coordinate_system1
clear_coordinate_system
motor1_is_axis 2000X

begin_program 2 clear

feedrate_time_units = 1000 msec
set_acc_time_of 200 msec
set_S_curve_time_of 50 msec
repetition_count = 0
set_feedrate_of 1 revs/sec
start_gathering_data

repeat_as_long_as(repetition_count<repetition_limit)
      move_X_to_position 2
      sit_there_for 300 msec
      move_X_to_position 0
      sit_there_for 400 msec
      increment_repetition_count
end_of_loop

stop_gathering_data
end_program

reserve_memory_for_gather
```

Download Options

There are several options associated with the downloading of files. These are accessed through the "*Options - Preferences - Terminal*" screens.



Macros or Compiled PLCs
This needs to be selected if file you are downloading contains PLCC's or uses the "#include" or "#define" statements.

Create Log File
If you wish to generate a log file of the download process, select this box. This is good for debugging a large program.

Create Map File
A map file tells you what symbols are matched to which definitions.

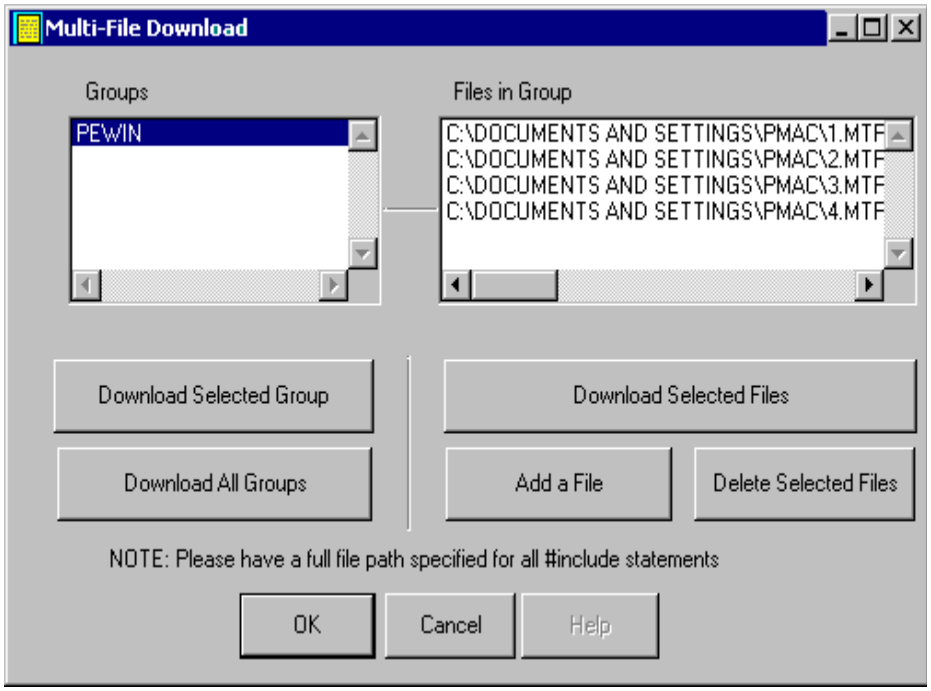
Just Compile
Have PEWIN32 process the files downloaded to a compiled format (output file has a *.56K file extension), but don't download to PMAC.

Display Every Download
If this box is checked, the download options dialog is displayed every time you download a file.

Do not Show if No Errors
If this box is selected, the download status window is removed if there aren't any errors or warnings.

The Multi-File Downloader

The Multi-File Downloader allows the user to specify groups of files to be downloaded at the same time. This is primarily for PLCC's since these all need to be download at once (that is a single file with all PLCC's need's to created for a one time download, the Executive takes care of this for you)..



The user specifies different groups and then adds files to the individual groups.
This shows all registered entries in the Multi-File download interface.

Uploading Files and Variables from PMAC

PEWIN32 gives you the ability to retrieve programs and variable values straight into an editor window. From there, you can edit, save and print the information.

From the "*File*" menu, you can select:

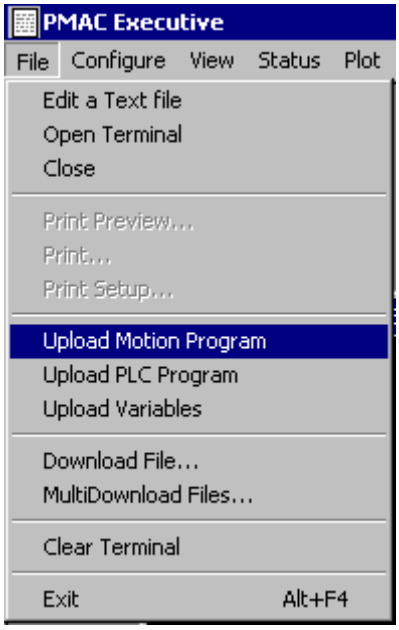
- Upload Motion Program
- Upload PLC Program
- Upload Variables

Note

You can backup PLCC's and other pertinent information from the Backup\Save Config menu item.

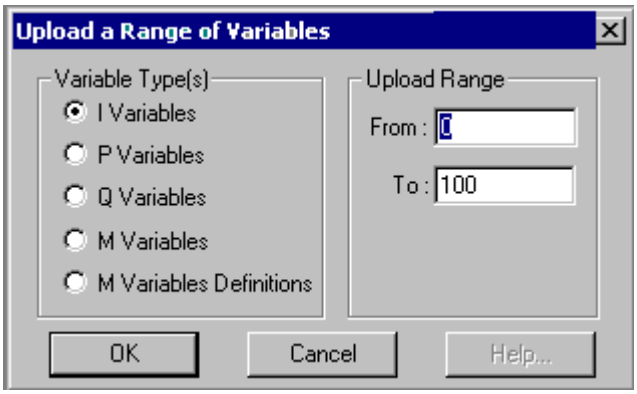
Uploading Motion or PLC Programs

Selecting the upload motion or plc program prompts you for the program number. If you are uploading a motion program, there is no limit to the program number, however, if you are uploading a plc, the program number must be between 0 and 31.



Uploading PMAC Variables

The upload variable menu item allows you to upload a range of variables :



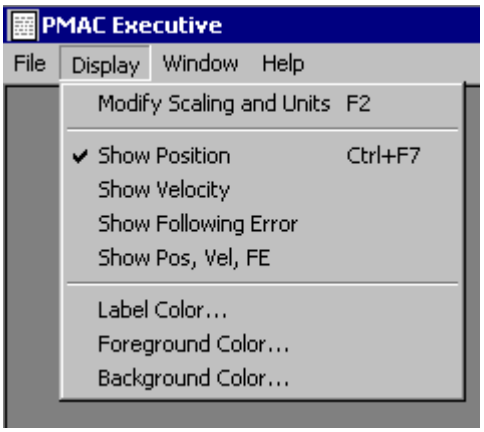
Just specify the type of variables you wish and the range.

Position Window

The position window displays motor position information. You can toggle the window to display position, velocity, or following error for all eight motors simultaneously, or you can display all of these parameters for a specific motor.

DISPLAY menu

When the Position window is highlighted, the menu changes to:



MODIFY SCALING AND UNITS - allows the user to set scaling and unit parameters for the display (see below).
SHOW POSITION - shows the current position of all 8 motors.
SHOW VELOCITY - shows the current velocity of all 8 motors.
SHOW FOLLOWING ERROR - shows the current following error for all 8 motors.
SHOW POS,VEL, FE - show position, velocity and following error for a specific motor. Use the *<PAGEUP>* and *<PAGEDOWN>* keys to change motors.
LABEL COLOR - Color of motor # labels used in position window.
FOREGROUND COLOR - Color of text used to display position.
BACKGROUND COLOR - Color of background of position window dialog box.

By pressing the *<CTRL+F7>* key combination, the display will cycle through the different display modes. Pressing the *<F2>* key will bring up the Scaling and Units dialog interface.

This interface allows the user to set scale factors, unit labels, and other display parameters.

These parameters only affect the way information is displayed in the position window. They have no connection to PMAC itself.

To change the scale factor for displayed information, enter the appropriate number of encoder counts per user units (i.e. if you want to have 10,000 CTS = 1 inch, enter 10000). Next specify a name for your user units (i.e. “inch”, “deg”, “rev”). Select your velocity units (i.e. "per ms", “per second” or “per minute”). Enter in a value for optional rollover (i.e. 360 if your user units are degrees). A value of 0 for rollover indicates no rollover. Lastly, specify how many decimal places to the right of the decimal point you wish to have displayed (for inches, you may want to use a value of 3 so inches are displayed as “2.002 inches”).

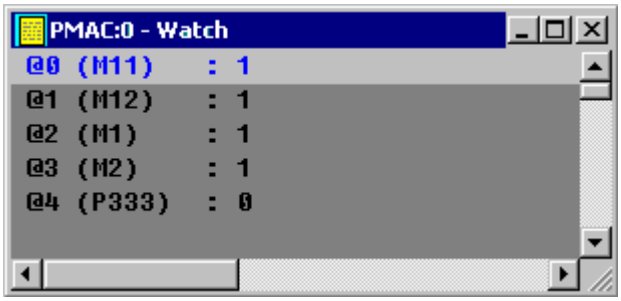
This is nice for systems with less than 8 motors.

Removing or unselecting the *Show* check box associated with a motor will cause the position window to not display information about that motor

This information is saved in your PEWIN.CNF file.

Watch Window

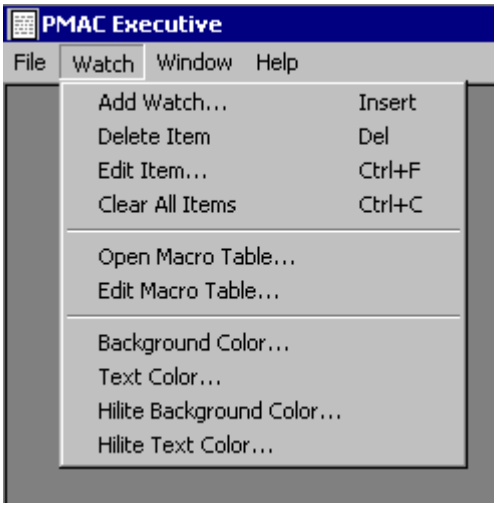
The watch window allows you to watch the response to any valid PMAC command, address or variable you wish to observe in real-time. You may also select defined macro names from the .tbl files created when plcs and pmcs are downloaded to PMAC, or define macros of your own.



Each entry is divided into three sections. The first is the Name or Identifier, the second, in parenthesis, is the actual PMAC command used, and the third is the value returned from PMAC. When visible, the Watch window updates its information as often as your computer and PMAC will allow.

Watch Menu

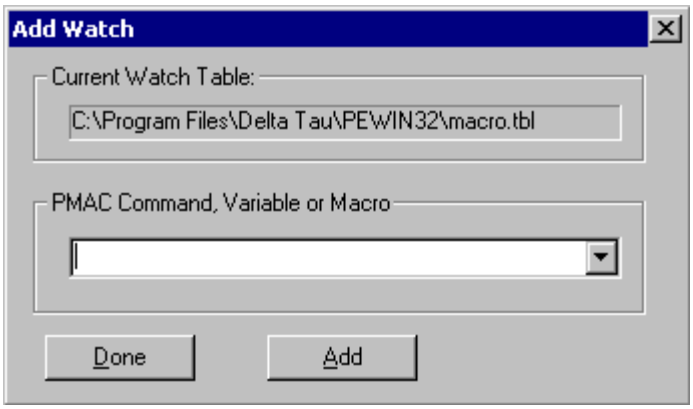
When the Watch window is highlighted, the main menu changes to :



ADD WATCH - use this to add new watch entries to the watch table.
DELETE ITEM - this will delete the currently highlighted item. If there are no entries in the watch table, this item is disabled.
EDIT ITEM - allows the user to change the format in which return values are displayed. The available options are decimal, hex and binary. See the Formatting Watch Entries section below.
CLEAR ALL ITEMS- clears (deletes) all items in the watch table. If there are no entries in the watch table, this item is disabled.
OPEN MACRO TABLE- use this to open a MACRO definition table.
EDIT MACRO TABLE- use this to edit a MACRO definition table.
BACKGROUND COLOR - Color used for the background of the watch window
TEXT COLOR - Color of text used in watch window.
HILITE BACKGROUND COLOR - Selected watch window item's background color.
HILITE TEXT COLOR - Selected watch window item's text color.

Adding Entries to the Watch Window

When the Watch window is highlighted, pressing the *<INSERT>* key or selecting **ADD WATCH...** from the Watch menu brings up the Add Watch dialog interface.



Simply type in the PMAC command you wish to watch the status of and press the *Add* button. The entered command will then be displayed in the watch window. Add as many items as you want, then select the *Done* button to return to the Watch window. You can also select any defined macros from the macro drop down list (click the arrow button on the right).

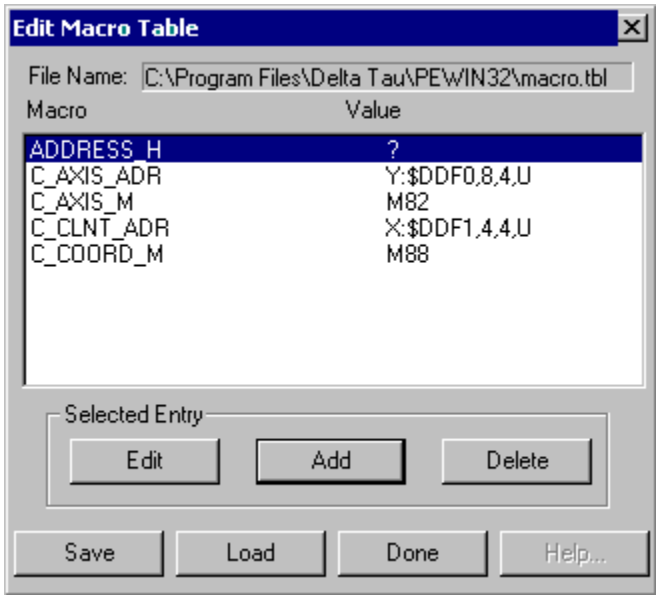
Watch Window Macro Definitions

PEWIN's Watch window has a built in macro management facility. You are able to load and save macro files (like those produced by the downloader) as well as add, edit and delete entries in those files. Once these macros are setup, instead of typing variables into the Add Watch dialog, you can select them from the macro list and view their status in the Watch window.

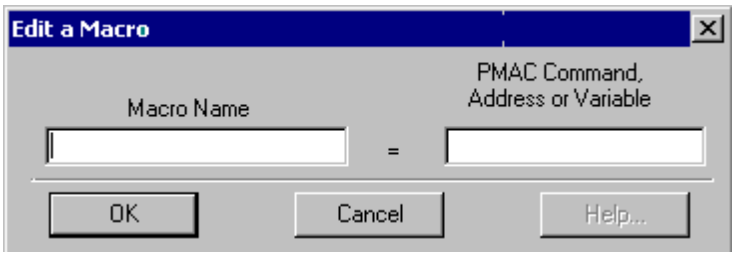
Simply select the entry you wish to watch and select the *Add* button.

Editing Macros

PEWIN's watch table contains a full macro configuration utility. This will allow you to create, edit and delete macros as well as save and load macro tables to disk. These can then be selected from the list in the Add Watch Item dialog box.



Selecting the *Add* or *Edit* buttons will allow you to assign or change a Macro.



Specify a macro name and it's corresponding PMAC command or variable. Pressing the *OK* button will add or change this macro definition in the macro list.

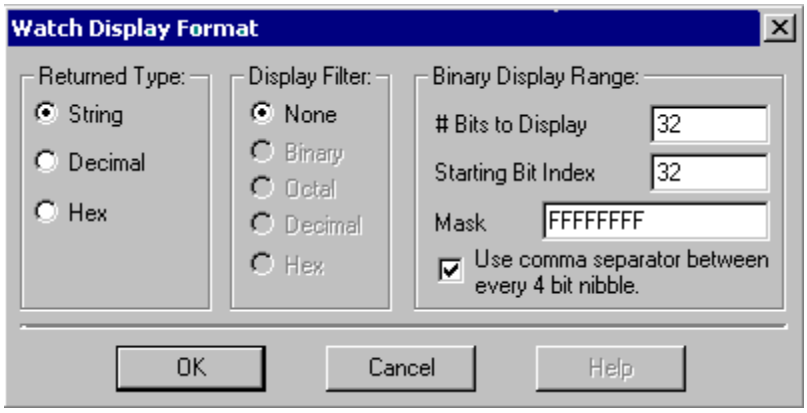
Selecting the *Delete* button will remove the currently highlighted macro definition.

To display plc or pmc macros in the Watch window, load the .tbl file (created when the plc or pmc was downloaded) by selecting the *Load from Disk* button. Select the appropriate file and press <*ENTER*>. Once a macro table has been loaded, it name is displayed in the Add Watch dialog. Open the list box by clicking on the small arrow button to the right of the command line to display a list of loaded macro definitions.

The *Load from Disk* and *Save To Disk* buttons prompt the user for a file and perform the specified operation.

Formatting Watch Entries

Watch entries can be formatted to display their data in a number of different ways.



Select the format the entry will be returned in. If you expect the response as a string (i.e. "TYPE"), Decimal (i.e. RX0) or hex (i.e. "?").

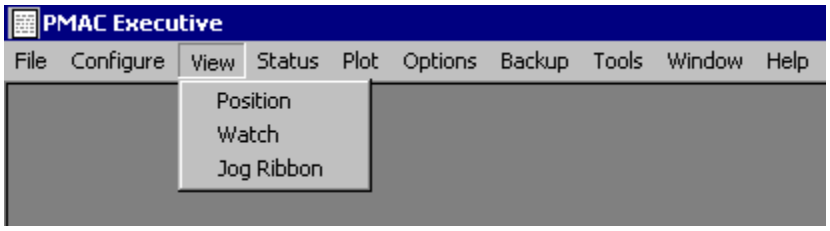
Select the numerical display mode from the selection in the middle, and the number of bits of this entry to use in the display.

To show all bits (the whole number) set the mask to FFFF	The bitmask will be ANDed (&) with the return value and the resulting value displayed in the Watch window.

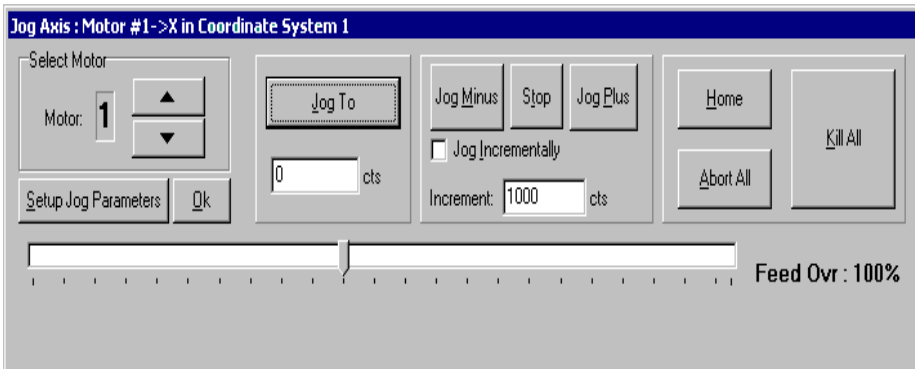
If the *Use Separator* check box is selected, the display will be broken up by separators. Binary numbers will have comas every four bits (nibble), decimal numbers will have commas every 3 decimal places left of the decimal point, hex numbers are separated by spaces every four places.

Jog Interface

PEWIN32 includes an interface for jogging motors. This is called the **Jog Ribbon**, and is accessed through the **JOG RIBBON** menu item in the **VIEW** menu.



Jog Ribbon



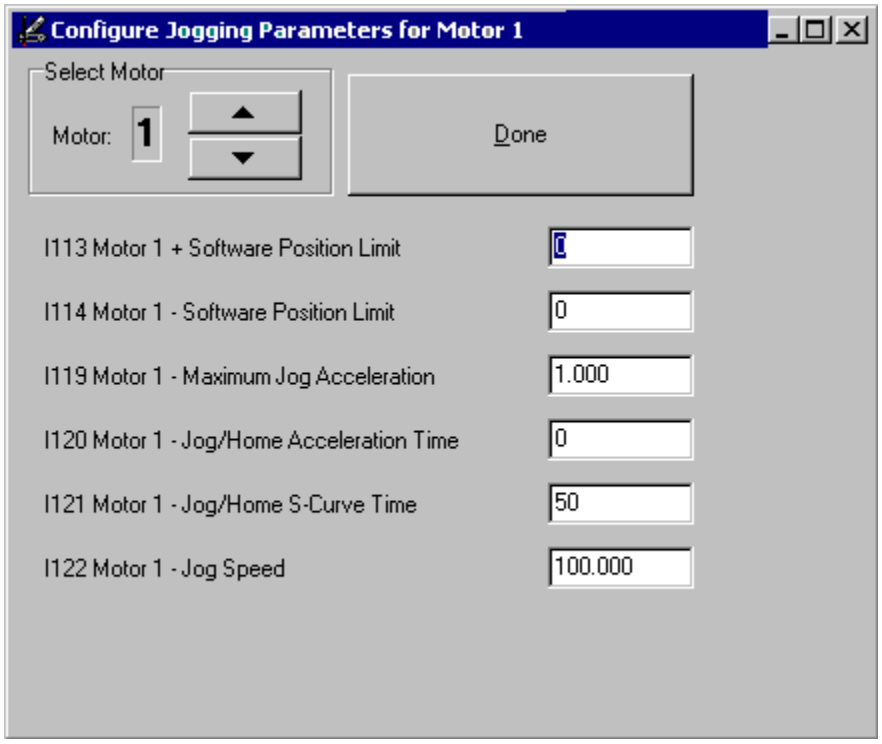
This interface only allows you to jog one motor at a time but gives you push-button access. The motor will start jog as soon as the *J Plus* or *J Minus* button is pressed.

Motor

Selects a motor for jogging. The motor number and its coordinate system definition are displayed in the window's title bar.

Setup Jog Parameters

Pressing this button will bring you to the **Configure Jogging Parameters for Motor #x** dialog box where pertinent I-variables, jog key definitions and relative step sizes may be changed.



Within this dialog box you may customize the jog dialog box and alter jog related I-variables for any motor. Pressing the up or down arrow buttons will scroll you to the next or previous motor number, respectively. Upon entering the Configure Jog dialog box the current motor number will be the same as the last addressed (i.e. last jogged) motor.

Jog Window Related I Variables

Ix13 - Positive Software Position Limit - set this motor's positive software position limit.
Ix14 - Negative Software Position Limit - set this motor's negative software position limit.
Ix19 - Maximum Jog Acceleration -set this motor's maximum jog acceleration rate.
Ix20 - Jog/Home Acceleration - set this motor's jog to home acceleration rate.
Ix21 - Jog/Home S-Curve Time - set this motor's jog to home S-curve time.
Ix22 - Jog Speed - set this motor's normal jog speed.

Jog To

Pressing this button will move the specified motor to the position specified.

Jog Minus or Plus

Pressing one of these buttons will start the jogging of the motor, plus or minus depending on which butoon you press.

Stop

Pressing this button wil stop the motor from jogging.

Home

Pressing this button issues a HM command to the currently selected motor.

Abort All

Pressing this button sends a ^A to PMAC stopping all motors.

Kill All

Pressing this button sends a ^K to PMAC disabling all motors.

Feed Overate

This changes the feed rate (PMAC command %) for the coordinate system associated with the currently selected motor.

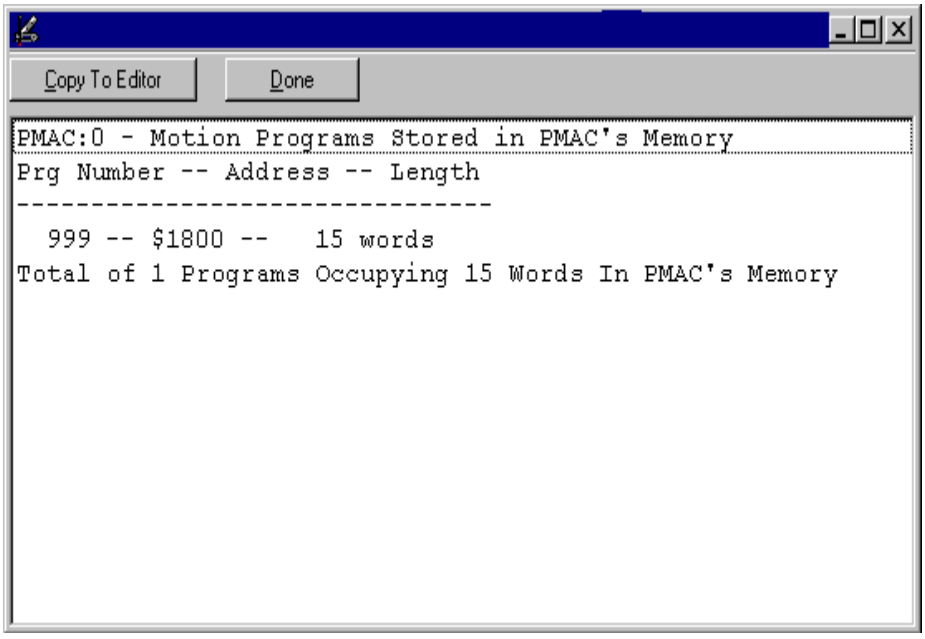
> ***Note:***
>
> This will change the speed of all motors assigned to this coordinate system.
>
> This interface is especially useful when testing new pmc programs. You have quick access to the motor stop and kill functions as well as the ability to change the feedrate of the coordinate systems.

Status Screens

The status menu lets you view how programs, motors, coordinate systems, and global parameters and connectors are organized and functioning in PMAC. Many of the these allow you to monitor parameters in real-time.

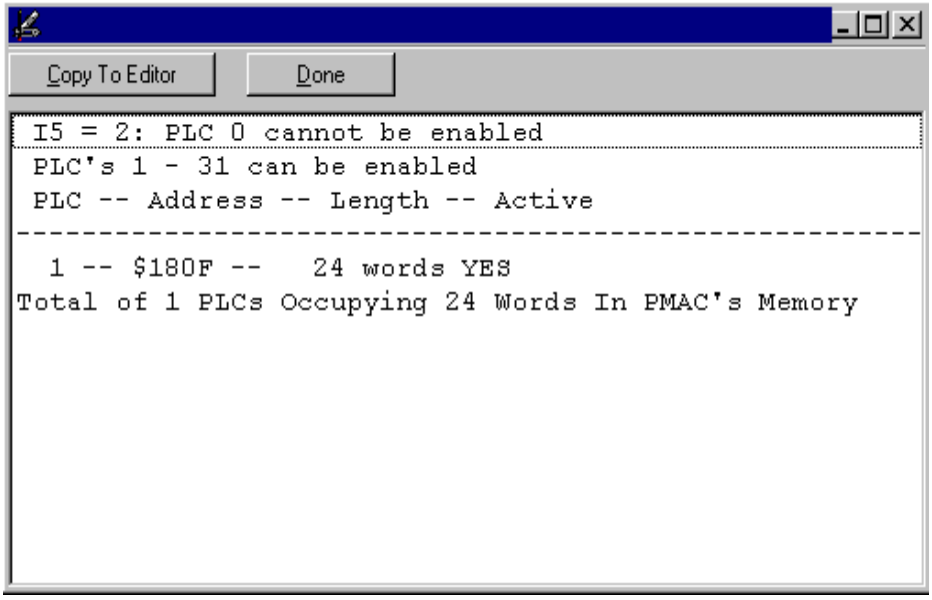
Motion Program Information

This opens a window which displays information about all the programs in PMAC's memory (a program is defined with the statement 'open program *x*' where x is the program number). For each program found, you will see the program number, the program stating address and the total amount of PMAC memory occupied by the program. At the bottom of the screen is the total number of programs and the total amount of memory occupied by all programs. Use the **FILE** menu options to save or print this information.



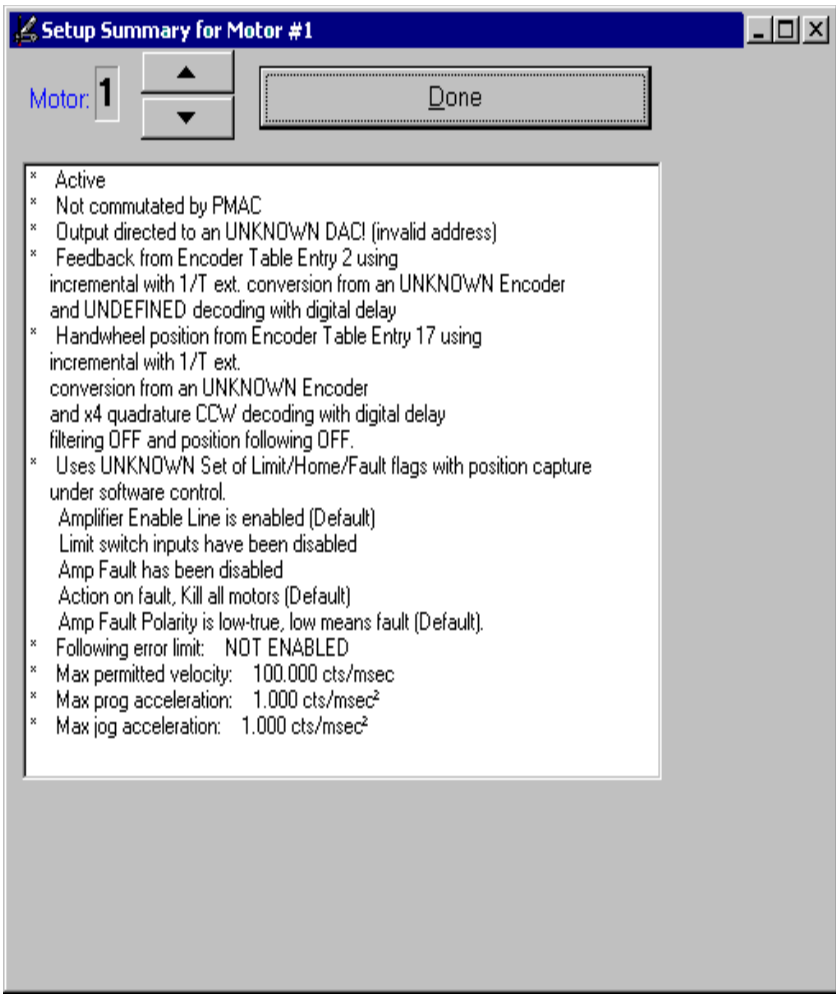
PLC/PLCC Program Information

This option opens a window which displays information about all the PLC programs in PMAC's memory. You are told the value of I5 and what that value means to PMAC. I5 is the I-variable which determines which PLC's can be enabled. For each PLC, the number, starting address and size are displayed as well as whether the PLC is currently active. PLC totals are given at the bottom of the screen. Use the **FILE** menu options to save or print this information.



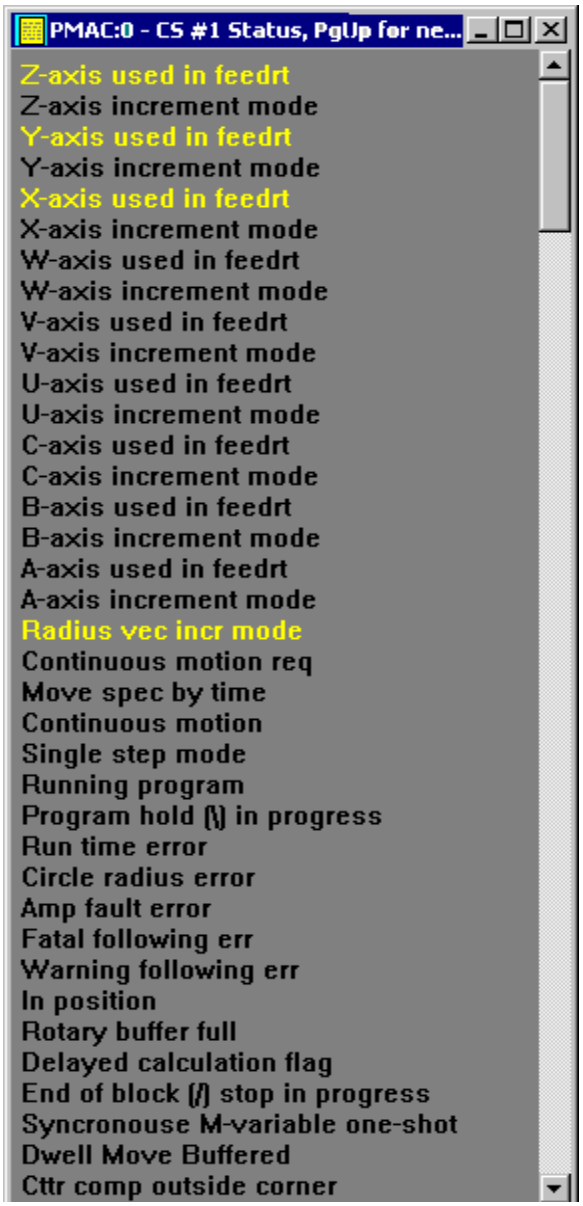
Motor Setup Summary

This will display the configuration of the specified motor. This option reads key PMAC memory locations and then does any necessary intepretations. The main parameter settings for the motor are displayed.. This screen can be helpful in determining problems when trying to servo a motor. To select a particular motor, use the <*PAGEUP*> or ,<*PAGEDOWN*> keys. Use the **FILE** menu options to save or print this information.



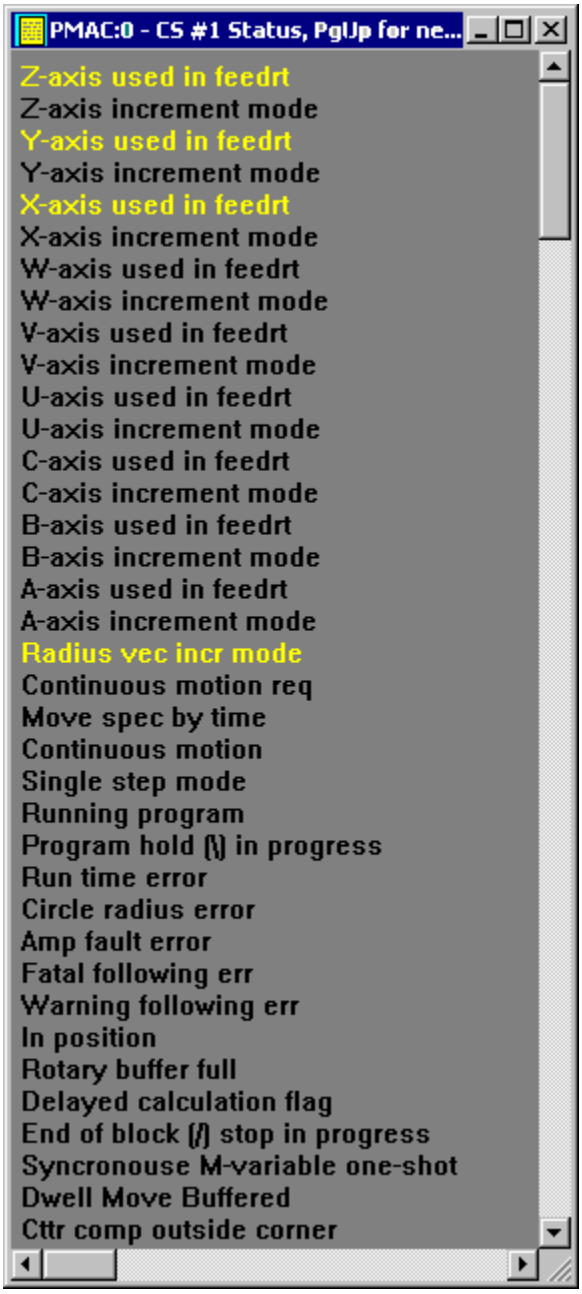
Motor Status

This display shows the interpretation of the status bits of the specified motor in real-time. This is done by continually sending the ? command to PMAC. Those conditions that are true are highlighted. Pressing the <*PAGEUP*> or <*PAGEDOWN*> keys changes the motor being examined. There are 2 modes for this window, condensed and full. Condensed mode only shows the most important status bits while full mode shows all the status bits.



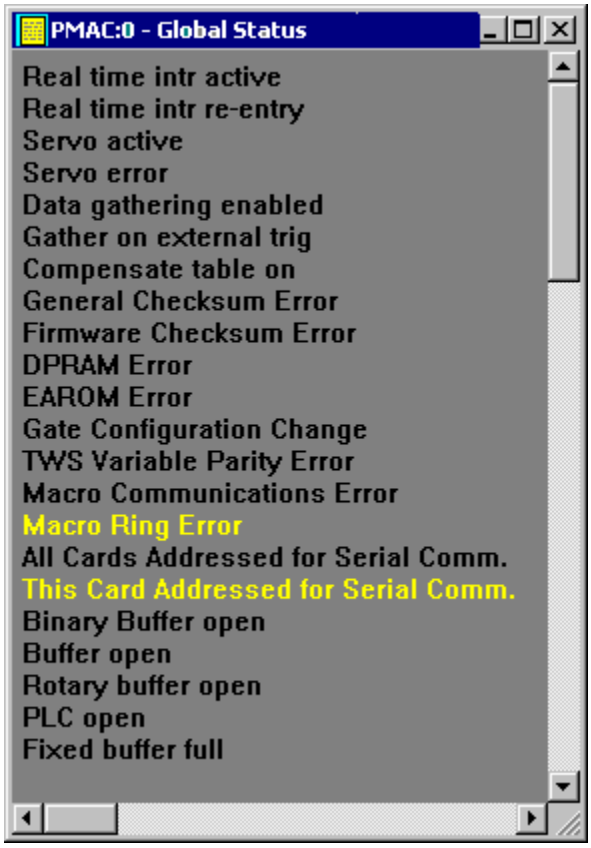
Coordinate Systems Status

Selecting this menu option displays the status of the specified coordinate system. This option actually sends the double question mark command ?? to PMAC and interprets the bits of the hexadecimal number returned by PMAC. Those conditions that are true are highlighted. Pressing the <*PAGEUP*> or <*PAGEDOWN*> keys changes the motor being examined. There are 2 modes for this window, condensed and full. Condensed mode only shows the most important status bits while full mode shows all the status bits.



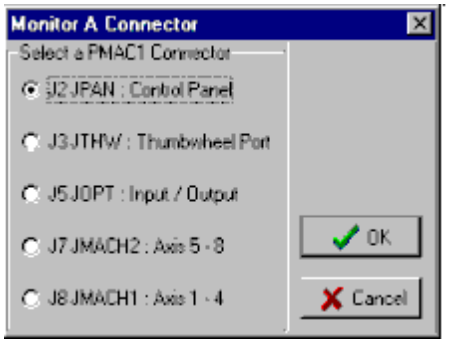
Global Status

This display shows the interpretation of the global status bits in real-time. This is done by continually sending ??? command to PMAC. Those conditions that are true are highlighted. Pressing the <*PAGEUP*> or <*PAGEDOWN*> keys changes the motor being examined. There are 2 modes for this window, condensed and full. Condensed mode only shows the most important status bits while full mode shows all the status bits.

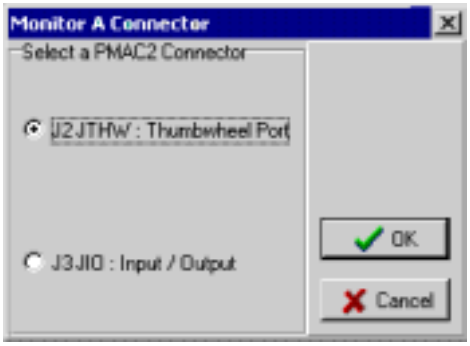


Connector Status

This option allows you to monitor the status of PMAC's connectors. Currently, PEWIN32 supports the J2(JPAN), J3(JTHW), J5(JOPT), J7(JMACH2) and J8(JMACH1) connectors. PEWIN automaticaly selects which connectors may be monitored depending on the PMAC series

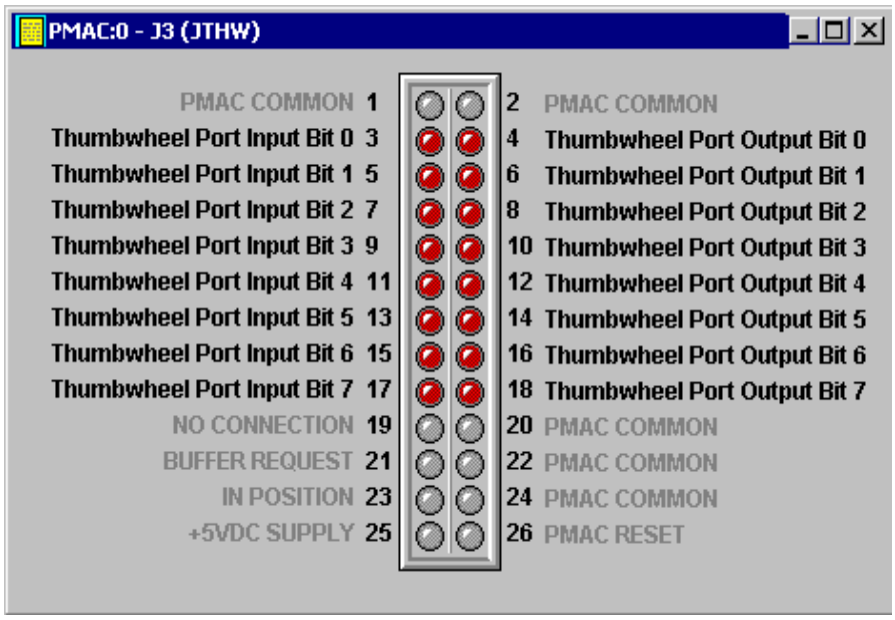


PMAC1 series

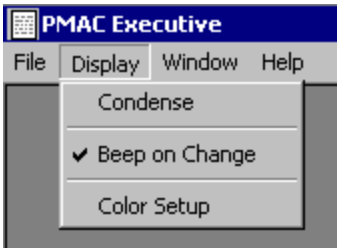


PMAC2 series

After making a selection from this dialog box and pressing the OK button, the status window showing the connector and its respective pins will be displayed. The colors of the pins change depending on the state of the pin, TRUE or FALSE. Pins which are inaccessible through software are marked with an *.

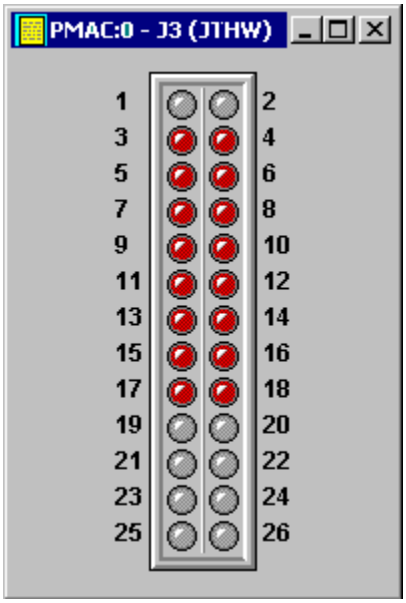


When a connector status window is highlighted, the main menu changes to :



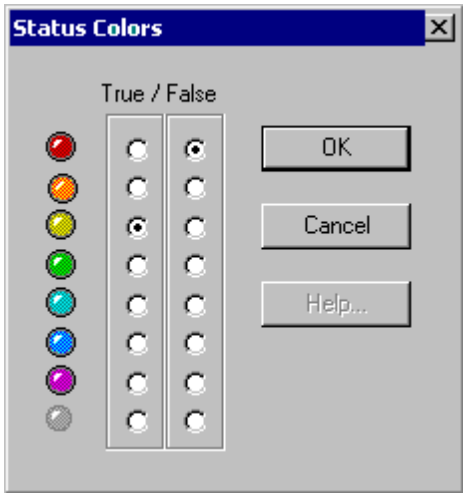
CONDENSE/EXPAND - Places the status window into condensed mode. In this mode only the pins and their respective numbers are displayed.

(a condensed JPAN status window)



BEEP ON CHANGE - When selected, the system will beep each time a parameter changes.

COLOR SETUP - this will bring up the Status Colors dialog box which will allow the user to change the display colors used in the current connector status window.



Select the colors you wish to use for True and False state displays.

This information is saved in your Terminal INI file.

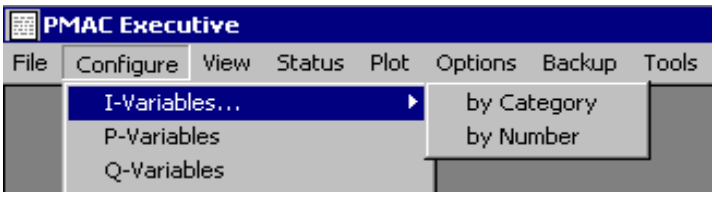
PMAC CONFIGURATION MODIFICATION UTILITIES

I Variables

At the heart of the PMAC's configuration are it's I variables. These determine what makes a PMAC. Because of the enormity of the I variables and their importance, we have developed two interfaces for manipulating them. Note this manual shows the screens that would come up if you have a PMAC. PMAC2 and other PMAC's should be detected and may have different screens.

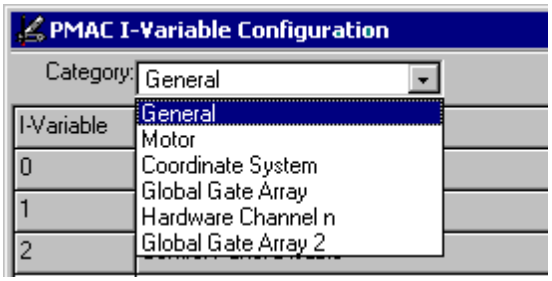
Note:

Changing I variables in these windows or in the terminal does not change the values stored in PMAC's non-volitile memory (EPROM of flash ram). You muse use PMAC's SAVE command to permanently save the changed values.

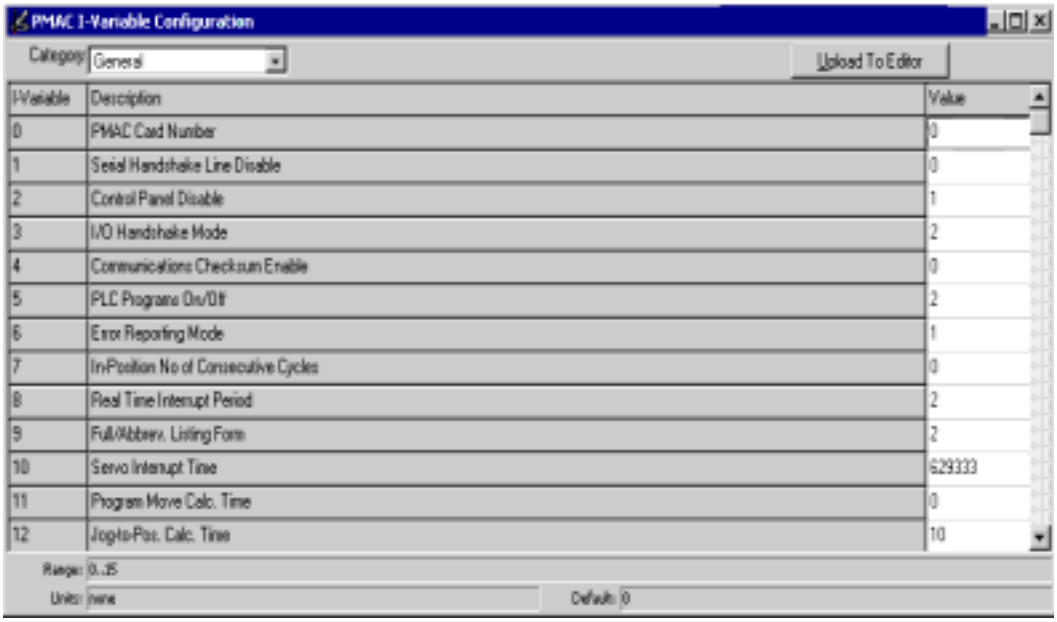


I Variables by Category

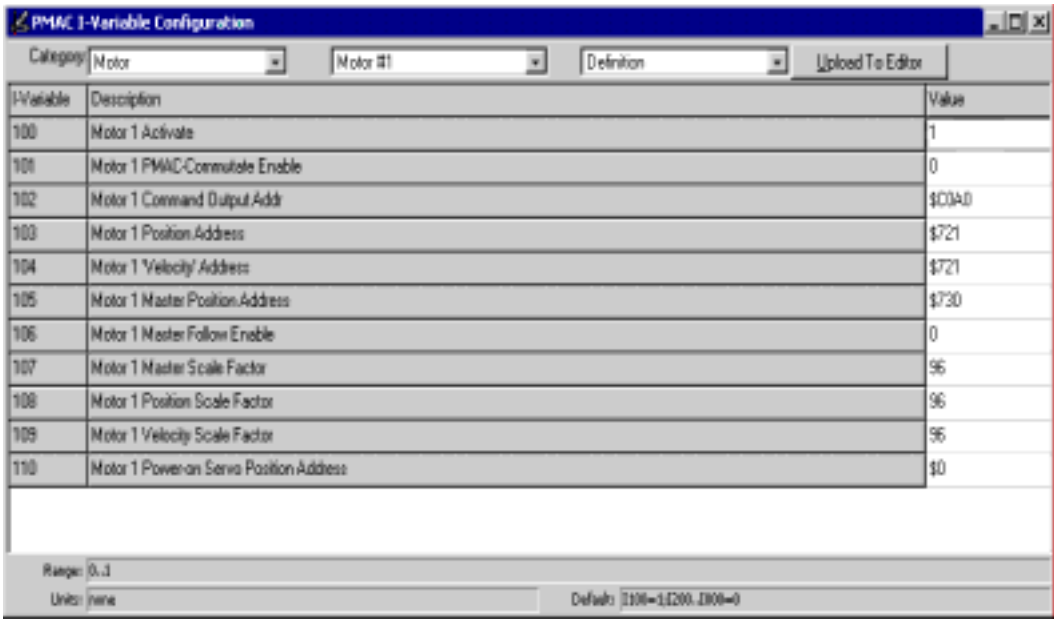
Selecting the by Category list-option will present the I variables broken down into six categories, General, Motor specific variables Coordinate system specific variables, Global Gate Array variables, Hardware channel n variables and Global Gate Array 2 variables.



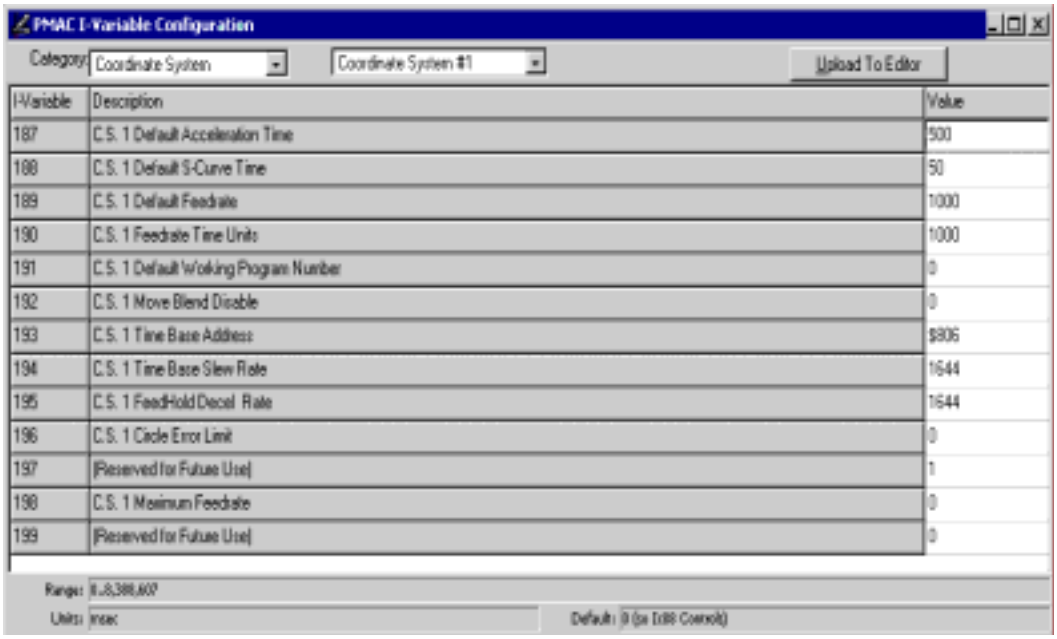
if we open each one indepently, the windows will look like:



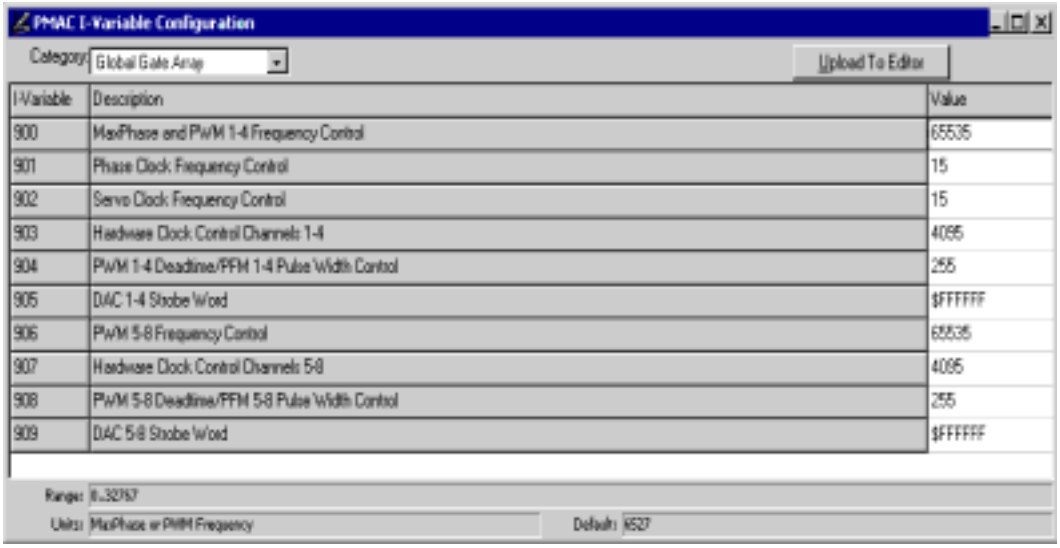
General I variables are all the I variables.



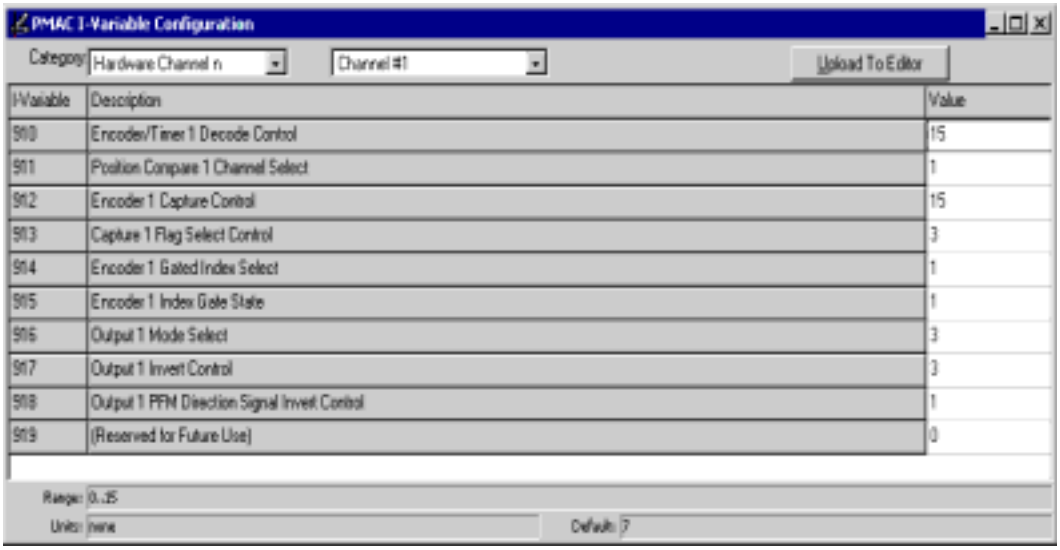
Motor specific I variables determine each motors characteristics.



Coordinate System specific I variables specify how motors assigned to a coordinate system respond.



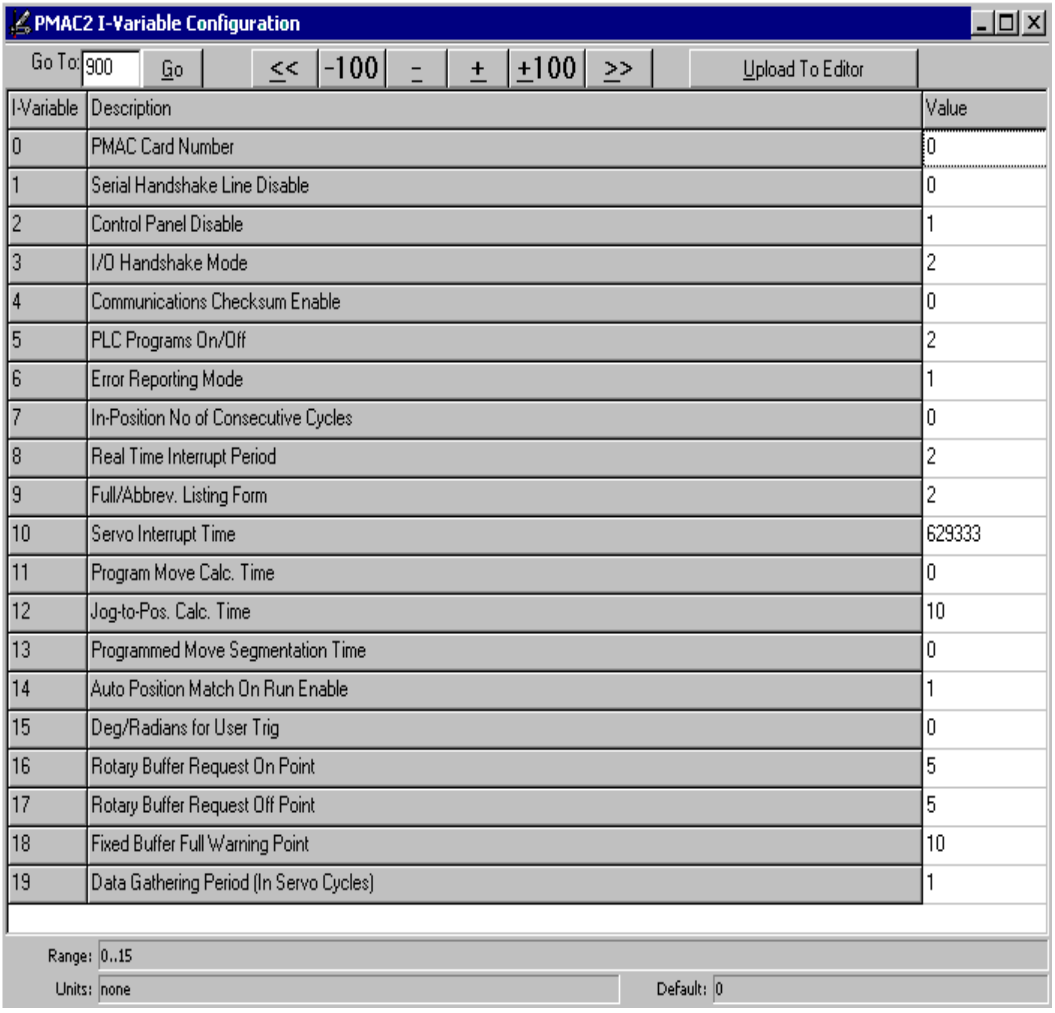
Global Gate Array I variables used for general card setup.



Hardware channel N I variables used for encoder I variables.

I Variables by Number

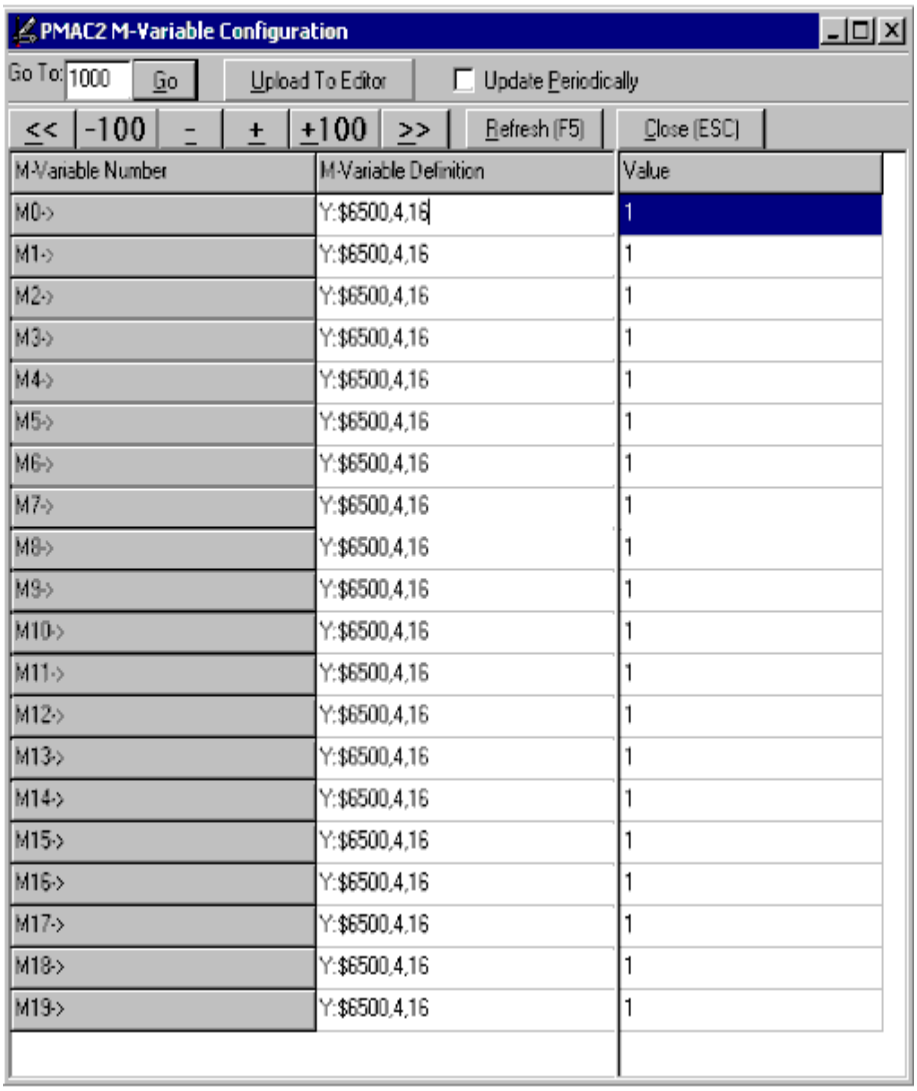
Selecting the by Number option will present the entire list of I variables in order from 0 to 1023.



To move through the list, use the two arrow buttons on either side of the window or use the Go To button to jump to a specific variable.

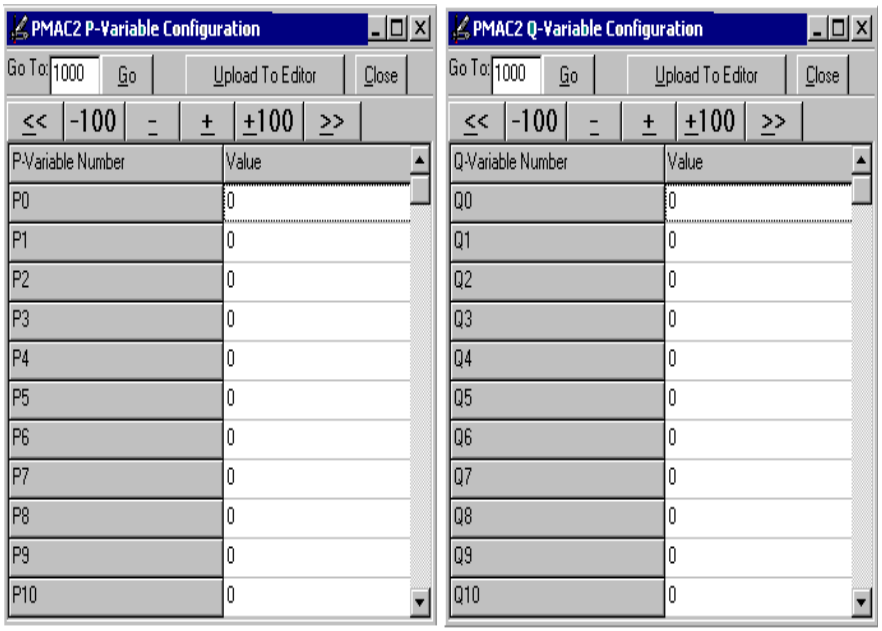
M Variables

This variable window functions the same as the I variable window above. It enables you to view and easily change the value or definition of any of PMAC's 1024 M-variables. It is your responsibility to format the M-variables pointer properly. Refer to PMAC's memory map for addresses to point to. A listing of suggested M-variables pointers is given in the example program SETUP.PMC included on the distribution disks.



P & Q Variables

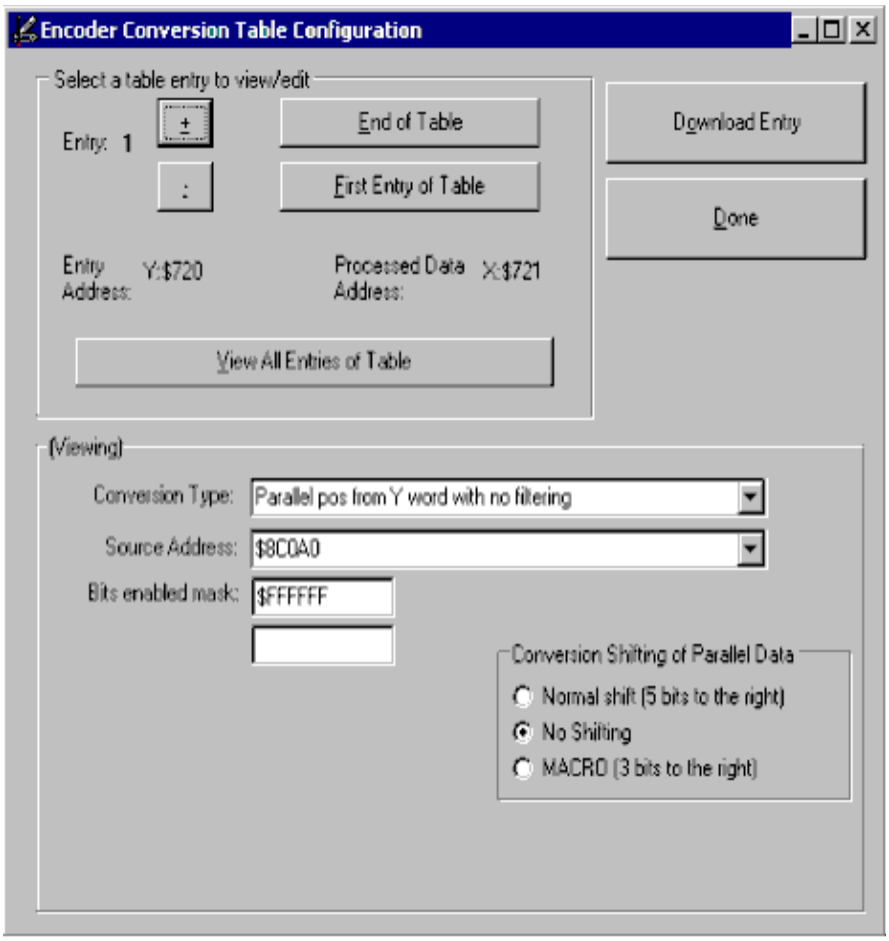
The windows for each of these options is the same as for the I variable window described above with the exception that more variables are displayed at one time.



Encoder Tables

This menu choice allows you to alter, update, save or retrieve the entries of the encoder conversion table. You will need to alter this table in order to use various types of feedback other than the default *1/T Conversion*. Each feedback device will have an encoder configuration. In order for PMAC to interpret a particular feedback device correctly, the raw data must be converted appropriately. The choices within this dialog box enable the user to easily change the conversion process without getting into the necessary details. Please refer to the *PMAC User's Manual* for details concerning how to edit the conversion table manually (i.e. using PMAC memory read and write commands) or press <*F1*> for help.

The table entries, which begin at PMAC address memory location $720 hex, may be scrolled through using <*PgUp*> or <*PgDn*>. Use the mouse or the <*TAB*> and <*SHIFT-TAB*> keys to move to an item in the dialog box. Altering the conversion method may be done by clicking your mouse or using the spacebar and <*ENTER*> on one of the conversion types. To change the encoder source address, simply move to the source address field of the desired entry and select the address from the pull-down list. To leave the conversion table editing without making any changes in PMAC select CLOSE or press <*ESC*>. To change the selected entry in PMAC to the specified format select DOWNLOAD TO PMAC.



Entry Number

This field contains the number of the encoder conversion table entry currently being viewed. Use the up and down buttons or <*PgUp*> and <*PgDn*> to step through the encoder conversion tables.

Entry Address

This field contains the address (in hex) to store converted source data from the Source Address. These begin at memory location $720 hex.

Conversion Type

This field contains the conversion type for the current entry. Select this entry and press <*ENTER*> or click on the down arrow to the right of this field to view a list of common conversion types. You can select a conversion type from this list using a mouse or the keyboard arrow keys. If you need to use a conversion type not shown in this list, you will have to enter the data manually from the terminal. Refer to the section entitled "Encoder Conversion Table" in the Pmac User Manual for help with this process.

Inc with 1/T Ext

Short for "incremental with 1/T extension." For incremental encoders, the source address must be one of the DSP-GATE encoder counters, selected from the following list:

ENC1:	$C000	ENC9:	$C020
ENC2:	$C004	ENC10:	$C024
ENC3:	$C008	ENC11:	$C028
ENC4:	$C00C	ENC12:	$C02C
ENC5:	$C010	ENC13:	$C030

ENC6:	$C014	ENC14:	$C034
ENC7:	$C018	ENC15:	$C038
ENC8:	$C01C	ENC16:	$C03C

Most applications will use the 1/T-extension conversion method which uses timers associated with each counter to estimate fractional resolution.

Those who have set up to use the parallel sub-count interpolation must use a source address of one of the odd-numbered encoders. A typical setup address in this case would be $C010 hex, which provides parallel extension of the encoder 5 counter using encoder 6's flags.

A/D Register

This conversion choice picks up data from the top 16 bits of a 24-bit word. It is intended for use with the A/D converter registers in the DSP-gates, which are fed by Accessory 23 or Accessory 28(A or B). The source address specifies a word in the Y memory space, and should be one of the following:

ADC1:	$C006	ADC9:	$C026
ADC2:	$C007	ADC10:	$C027
ADC3:	$C00E	ADC11:	$C02E
ADC4:	$C00F	ADC12:	$C02F
ADC5:	$C016	ADC13:	$C036
ADC6:	$C017	ADC14:	$C037
ADC7:	$C01E	ADC15:	$C03E
ADC8:	$C01F	ADC16:	$C03F

A typical address for an A/D register would be $C006, which provides the conversion of the ADC2 register. With A/D conversion, there is no rollover (software extension) performed.

Parallel with and without Filter

The four choices Parallel Y without Filter, Parallel Y with Filter, Parallel X without Filter, and Parallel X with Filter differ only in the use of X or Y memory space and whether or not the filter is to be used. If you are providing position information to PMAC as a parallel data word (as from an absolute encoder or processed from a laser interferometer) you will use this conversion method. The parallel data word may come either from the X or Y memory space. Usually this data is brought in on an ACC-14 board, which is in the Y-memory space.

When using ACC-14 to bring in the data, the following source addresses would be used:

1st ACC-14 Port A (J7):	$FFD0
1st ACC-14 Port B (J15):	$FFD1
2nd ACC-14 Port A (J7):	$FFD8
2nd ACC-14 Port B (J15):	$FFD9
3rd ACC-14 Port A (J7):	$FFE0
3rd ACC-14 Port B (J15):	$FFE1
4th ACC-14 Port A (J7):	$FFE8
4th ACC-14 Port B (J15):	$FFE9
5th ACC-14 Port A (J7):	$FFF0
5th ACC-14 Port B (J15):	$FFF1
6th ACC-14 Port A (J7):	$FFF8
6th ACC-14 Port B (J15):	$FFF9

Parallel-feedback conversion requires a double (for non-filtered) or triple (for filtered) entry in the conversion table. The second entry -- filtered or non-filtered --specifies the size of the feedback word

used. The entry is a 24-bit word in which each bit actually used for the parallel feedback is a one; the unused bits above are zeros (parallel feedback should always be connected starting at bit 0 of the data word). For a 12-bit absolute encoder, this entry would be $000FFF hex; for 14 bits, it would be $003FFF hex. The maximum entry permitted is for 19 bits: $07FFFF hex. The count can be software-extended by PMAC, permitting rollover. If more than 19 bits of true absolute position are needed, the power-on position can be read to full range using I*x*09 and I*x*10; from then on, position is kept through rollover using the conversion table.

The converted data from the parallel word is put in the X data word matching the last (2nd or 3rd) setup word for the entry. This is the address that should be used by the motor I-variable that picks up position (I*x*03, I*x*04, or I*x*05). For instance, if the first setup entry (address Y:$720) in the conversion table were $30FFD0 hex (filtered parallel data), the size entry would be in Y:$721, and the maximum change entry would be in Y:$722 hex. The converted data would be placed in X:$722 hex. If this were the position feedback for motor #1, I*x*03 would be set to $722 hex (1826 decimal).

Note:

The parallel data word from a laser interferometer is not true absolute position information, because the interferometer is fundamentally an incremental sensor. For this arrangement, there is no need to wire more than 19 bits of data to the ACC-14 (in fact, 16 bits is sufficient). Position reference is established by a homing procedure, and full range is achieved by using rollover to extend the range of the count in software.Filter/Max change

The parallel data word filter simply sets a maximum amount the data word is permitted to change in a single servo cycle. This value should be entered in the filter box in hex. If PMAC sees a change larger than this in the source data word, the converted data only changes by the maximum amount. There is no permanent loss of position information if the filter "kicks in". This filtering permits protection against spurious changes on high-order data lines, while not delaying legitimate changes at all. This maximum amount is the third setup entry for the encoder in the Y-memory portion of the conversion table. It should be set slightly greater than the maximum actual velocity expected on the sensor.

Time Base

A time-base conversion is a scaled digital differentiation. Every servo cycle, it calculates the difference between the value of the source register for this cycle and the value for the last cycle, and multiplies the difference by the scale factor. The scale factor can be entered under the Bit Enable/Time Base input box. The most common use for the resulting value is for the time-base (feedrate override) control, which makes the speed of PMAC execution proportional to an external frequency (usually the speed of a master device).

Triggered Time Base

This is very similar to the timebase conversion described above, but with the added feature of synchronizing timebase following upon a hardware trigger. Refer to the *PMAC User's Manual* for further reference.

Inc with Parallel Ext

Short for "Incremental Encoder with Parallel Extension". Those who have set up to use the parallel sub-count interpolation for incremental feedback would use this option. In this case, the source address must be one of the odd-numbered encoders. A typical source address would be $C010, which provides parallel extension (i.e. interpolation) of encoder #5 counter using encoder #6's flags.

Inc without Parallel Ext

Short for "Incremental Encoder without Parallel Extension". Those who have set up to use the parallel incremental feedback without interpolation would use this option.

Conversion Type

The result of choosing the add with previous entry option in the conversion will result in the sum of this entry and the previous entry in the table. Setting this option to Additive in entry number 3 for instance, would result in the summation of entry two and entry three at the address of location 3. This permits the servo feedback to sum of two sensors. (If the polarity of the sensors or their counters is opposite, this provides the difference of the sensors. This can be useful for Doppler-type sensors, where the reference wave and the shifted-frequency wave are fed into different counters, one counting up, the other counting down; summing the two counters provides position).

Source Address

This the hex address where the raw data is located.

Bits Enabled Mask

This field contains a scale factor used when the appropriate bit is enabled. The most common use for the resulting value is for the time-base (feedrate override) control, which makes the speed of PMAC execution proportional to an external frequency (usually the speed of a master device).

Max Change

This limits the magnitude that the input value is allowed to change between scans.

View All Entries of Table

This option opens a window containing a list of all the defined encoder conversion table entries in PMAC's memory. This window can then be saved to disk or printed.

Download Entry

This option sends your encoder conversion table configuration to PMAC's memory. Every time you make changes to the table, you need to download it via this button. At this point, the table will be actually be in use by PMAC. An altered encoder conversion table configuration must be stored in PMAC's permanent EPROM memory (using the this button) if it is to survive past a power down or cycle reset.

Done

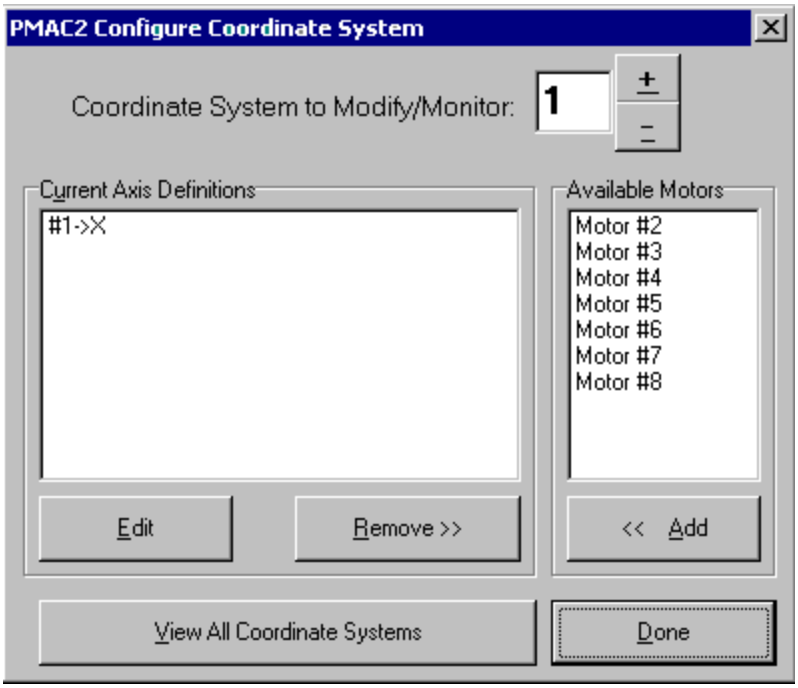
This closes the Configure Encoder Conversion Table window. This will not download the present encoder conversion table entry to PMAC. Use the Download Entry to PMAC button if you want this entry to be sent to PMAC.

Conversion Shifting of Parallel Data

If you want to have fractional parts of the position or you have a MACRO station base system then you have to specify that in the conversion shifting of parallel data.

Coordinate Systems

This menu choice enables you to alter the currently defined coordinate systems, or define new ones. For motors to move within motion programs, they must first be assigned to an axis within one (and only one) of the eight possible coordinate system. Any motor may be assigned to any valid axis (X, Y, Z, A, B, C, U, V, or W) or coordinate system (1, 2, 3, 4, 5, 6, 7 or 8), provided the motor has not been previously assigned to another axis or defined in another coordinate system. To move from one coordinate system to the next, use the <*PgUp*> and <*PgDn*> keys. Assignments are downloaded to PMAC's internal memory as you ADD and REMOVE motors from the current axis definitions window.



Coordinate System to Modify/Monitor

This field displays the number of the coordinate system within PMAC that is currently being viewed. Use the + and - buttons or <*PgUp*> and <*PgDn*> to step through the coordinate systems.

Current Axis Definitions

This window lists all motors and their corresponding axis definitions for the current coordinate system.

Edit

This button opens a dialog box where you can edit the axis definition selected in the current axis definitions window. If you want to undefine a motor definition, use the Remove button.

Remove

This button removes the motor selected in the current axis definitions window from the coordinate system. Once you remove a motor from the current axis definition it will appear in the Available Motors window, and is available again for axis assignment.

Available Motors

This window lists all the PMAC motors which are available to be assigned to a coordinate system. A motor is available if it is not already defined in any coordinate system.

Add

This button adds the motor selected in the Available Motors window to the current axis definitions window. When you press this button a dialog box will open where you can enter the axis definition for the motor before it is added to the coordinate system.

View all Coordinate Systems

This option opens a window containing a list of all PMAC coordinate systems and their axis definitions. You cannot edit the coordinate system definitions in this window but you can print this information.

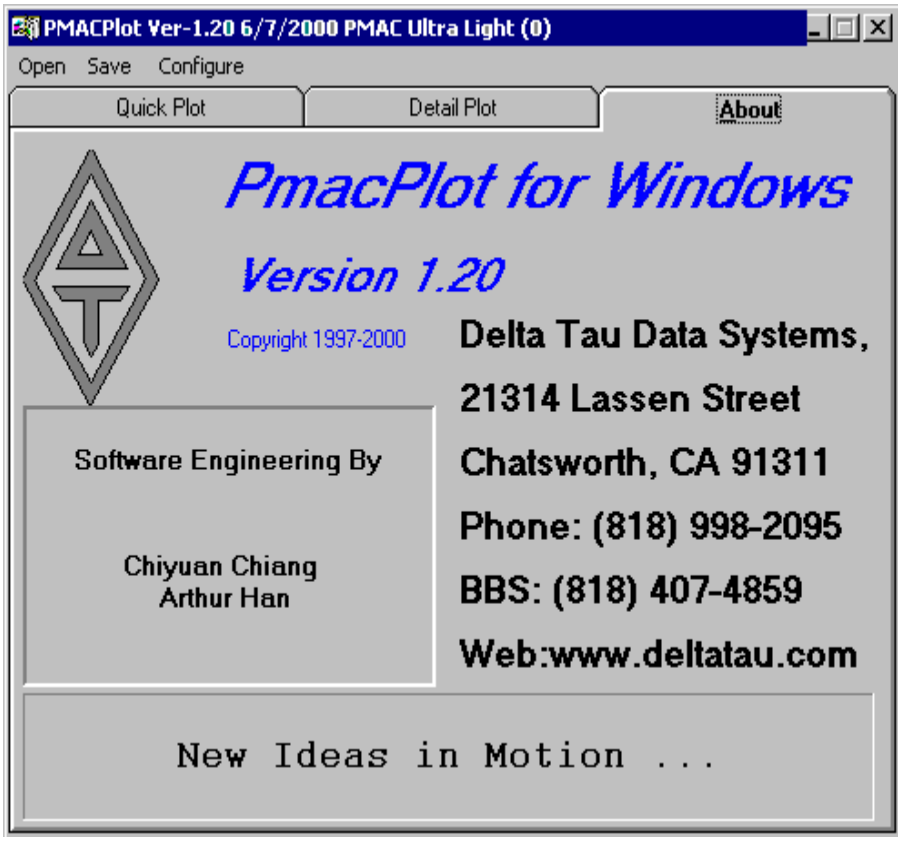
Done

This closes the Configure Coordinate Systems window.

PLOTTING AND DATA GATHERING

PmacPlot

PEWIN32 now uses an independent application for Quick and Detailed plotting features. PmacPlot has its own separate manual.



The Quick and Detailed plotting features allow the user to use PMAC's on board real time data gathering feature for diagnostics purposes. It is highly encouraged that everyone becomes familiar with this feature and its great debugging potential.

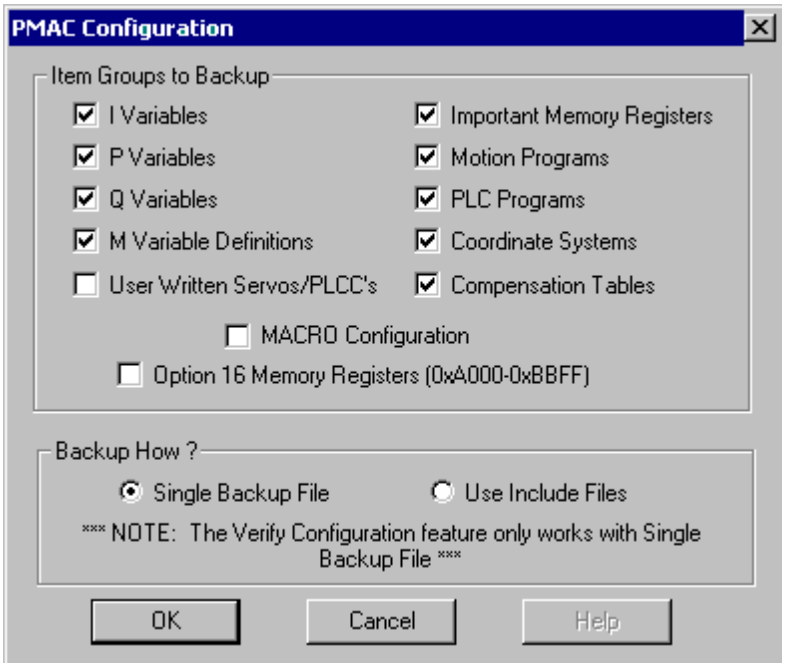
CONFIGURATION FILES

Backup / Restore PMAC Configuration Files

The Backup options let you save, restore, and verify all or part of PMAC's configuration. PMAC's configuration consists of all motion programs, PLC programs, I, P, Q, and M-variable values, M-variable definitions, custom servo algorithms, other important memory locations, leadscrew compensation tables, and coordinate system definitions.

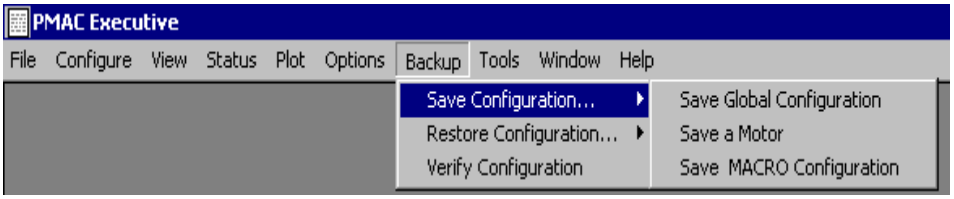
To backup or restore all of PMACs' parameters, choose the *Global Configuration* menu item. This allows the user to selectively choose which of PMACs' parameter to save or restore. To work with motor I variables only, choose the *Motor Configuration* menu item.

Dialog for Global Configuration.



Save Configuration

This dialog box enables you to upload any combination of motion programs, PLC programs, I, P, Q, and M-variable values, M-variable definitions, custom servo algorithms, other important memory locations, leadscrew compensation tables, and coordinate system definitions and save them to disk. The *Setup* button opens a dialog box where you select which parameters to save. These are the items which will be written to an ASCII file when you select the OK button in the Save Full PMAC Configuration window. This file serves as a backup and may be used to duplicate configurations on other PMAC cards.

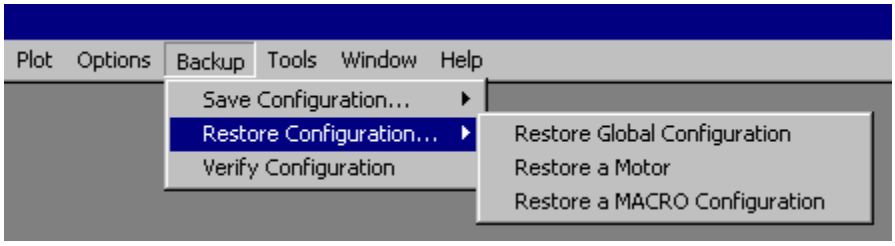


Note:

PLCC's are firmware specific. Therefore, a configuration file which uses PLCCs is only valid for that type and version of PMAC.

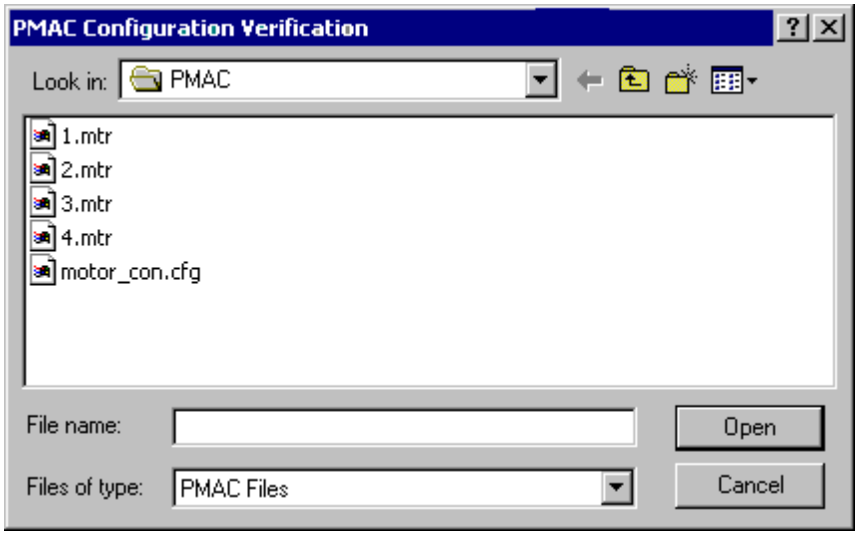
Restore Configuration...

This option is used to restore the contents of a backup file (created with the Save Configuration option) to PMAC. The file is simply downloaded to PMAC.



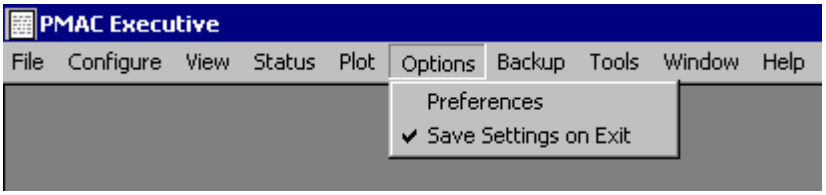
Verify PMAC Configuration

This option allows you to compare the contents of any text file with the contents in PMAC's memory. It provides a good way to validate the backup file created by the Save Configuration option. At present, you may only validate files that have no #include statements, which means files backed up using the "Single Backup File" option.



PEWIN32 SOFTWARE CONFIGURATION

Options Menu



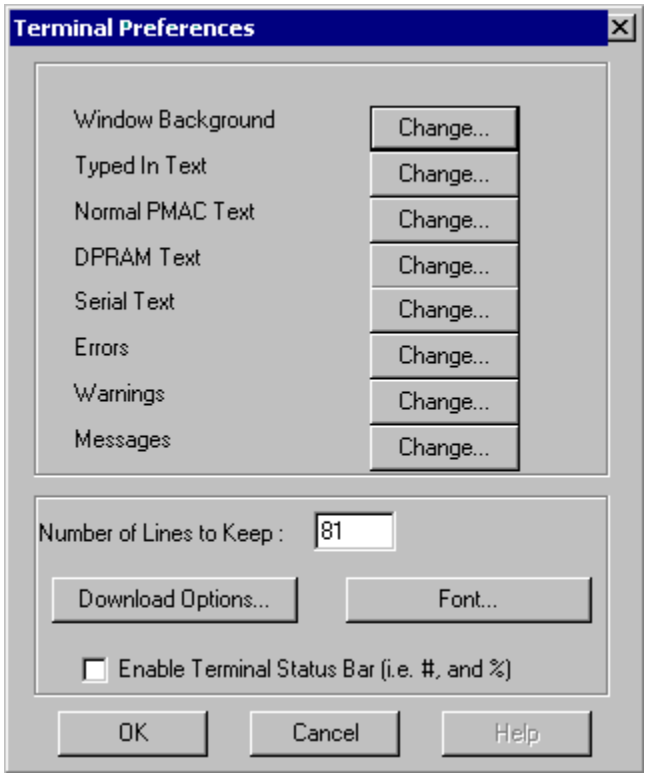
PREFERENCES - set default values for various PEWIN32 displays.
SAVE SETTINGS ON EXIT - when selected, save current window positions and status and restores them the next time the program is run.

Preferences

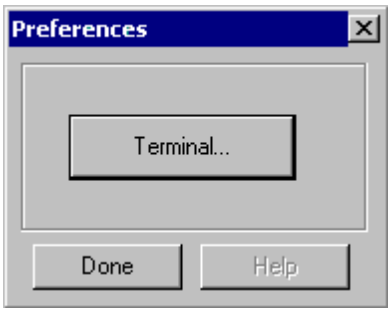
There is currently one preference area:

- The Terminal

Terminal Preferences



Selecting the Terminal Preferences menu item allows the user to change the colors for the various text messages displayed in the terminal as well as the font. So, as to finish press done at the preferences window.


Preferences
Terminal...
Done
Help

PEWIN32 HELP FACILITY

PMAC Users Guide and Software Manual

This manual as well as the PMAC software manual are available on-line. You can retrieve on-line information about PMAC, the Executive program, and the various help functions. This information is available alphabetically or by subject

Hardware Manuals

All four of the main PMAC hardware manuals are accessible on-line. These include the PMAC-PC, PMAC-VME, PMAC-Lite and the PMAC-STD.

Why Am I Not Moving?

This menu choice will probably save you many hours of searching and frustration in tracking down problems associated with running a motor especially if you are new to PMAC. You may run into the condition where a motor simply will not run, despite being sure that all the parameters have been set properly. What this menu option will do is look at your configuration, and cite possible (warnings) or definite (faults) causes of problems preventing you from running your motor(s). To use this self diagnostic feature, you must specify a particular motor

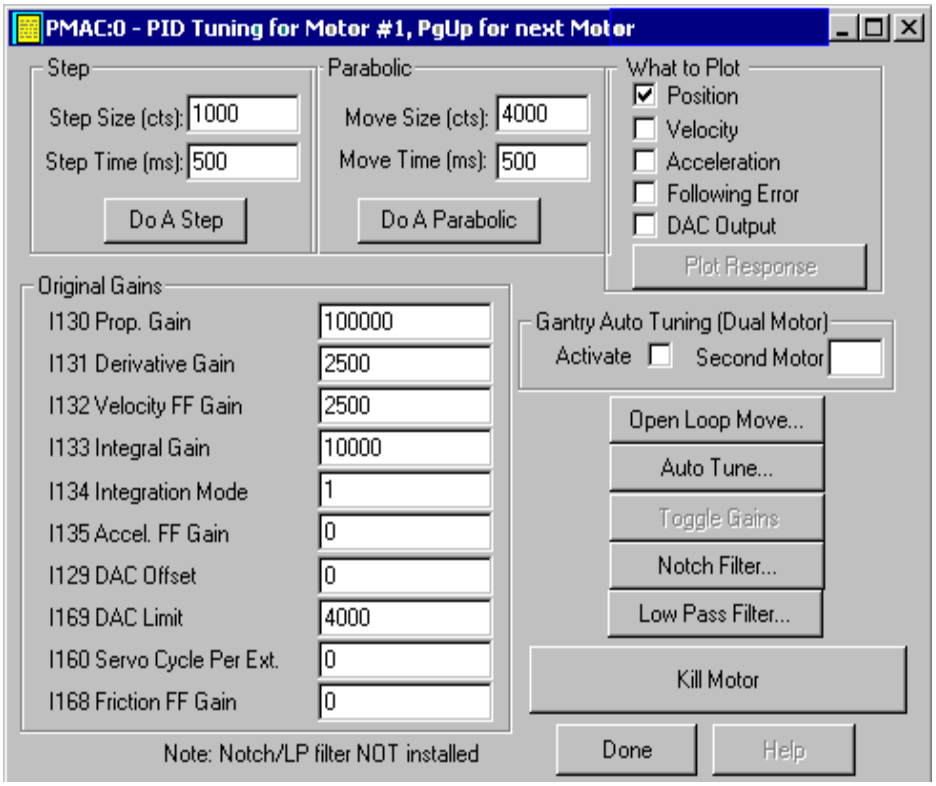
Why is my Program Not Running?

This menu choice will also probably save you many hours of searching and frustration in tracking down problems associated with running a motor especially if you are new to PMAC. You may run into the condition where a program simply will not run, despite being sure that all the parameters have been set properly. What this menu option will do is look at your configuration, and cite possible (warnings) or definite (faults) causes of problems preventing you from running your motion program(s). To use this self diagnostic feature, you must specify a particular coordinate system.

MOTOR AND SYSTEM TUNING WITH PEWIN32

PID Loop Tuning

The Tuning interface is intended to assist you in setting the controls parameters or gains of PMAC's PID filter, a process known as filter tuning, and for adjusting your motor amplifier via amplifier tuning. The goal here is to achieve a stable and well-behaved system. The process is interactive and iterative: you try a setting, see haw it performs, make necessary adjustments, and repeat until satisfactory results are obtained. This procedure is set up so that no special control expertise is required.



When you change one of the I-variables in the lower left fields of this dialog box (I*x*30-I*x*35, I*x*29, I*x*60, and I*x*69), you are actually changing the I-variable value , and consequently the characteristics of the servo loop, in PMAC. (Don't forget to issue a SAVE command when finished tuning to make your changes permanent.)

You have a choice of one of three predefined moves: step, parabolic, and open loop. These moves may be performed on any motor (to select a motor for tuning while in the tuning screen, press <*PgUp*> or <*PgDn*>). The step move, usually done first for *current loop systems* (if your amplifier is *not* using a tachometer it is considered a current loop system), commands the motor to do a closed loop step move, allowing you to analyze the step response of the system and adjust the proportional (I*x*30), derivative (I*x*31) and integral gains (I*x*33) accordingly. The parabolic move, usually performed after a step move, commands the motor to move in a closed loop parabolic profile. This is used to tune the velocity and acceleration Feedforward gains (I*x*32 and I*x*35). For velocity loop systems (if your amplifier is using a tachometer it is considered a velocity loop system), the open loop move is usually done first before a step or parabolic move. This test writes positive and negative values to the motor's DAC output. This can be used to calibrate (i.e. adjust offset in) the amplifier, tachometer, etc. For more information on PMAC's digital filter, see How the PID Works and The PID Algorithm in the PMAC User Manual.

When the PMAC Executive performs these tuning moves, it is necessary for the Executive to download a small motion program (PROG 999) and two PLC programs (PLC 30 and PLC 31) to PMAC. It is also necessary to modify a few I-variable, P-variables and M- variables in PMAC, and the coordinate system definitions. Since these parameters may be used in your application, the Executive will automatically backup this items to a file called TUNE.DAT before downloading the necessary tuning programs and variables. Here is a typical TUNE.DAT file which is automatically generated when you begin tuning:

```
;This file was created by the PMAC Executive during a tuning session.

CLS
UNDEFINE ALL
DELETE GATHER
&8
&7
&6
&5
&4
&3
&2
&1
#1->2000X
#2->2000X
#3->2000X
I19=1
I20=$7
I21=$800028
I22=$80002B
I23=$400045
I24=$0
I25=$800067
I26=$0
I27=$0
I28=$0
I29=$0
I30=$0
I31=$0
I32=$0
I33=$0
I34=$0
I35=$0
I36=$0
I37=$0
I38=$0
I39=$0
I40=$0
I41=$0
I42=$0
I43=$0
I44=$0
M1016->D:$080B
M1017->D:$002B
M1018->X:$3A,4,20
M1019->X:$3,19
M1020->X:$3,18
M1021->X:$3D,13
M1022->X:$7F1,24
M1023->X:$7F0,24
P1021=112.94116633
P1022=50
P1023=12288000
```

```
P1=10
P2=2000
P3=500
P4=1895
P5=5
P6=0
P7=128
P8=2
P9=4
P10=0
P11=0
P12=0
P13=5
P14=0
P15=0
P300=4294967295
P400=-1431655766
M0->X:$0,24
M85->X:$7F0,24
M86->X:$7F1,24
M90->Y:$38,24,S
M95->X:$3E,8,16,S
M164->X:$33,24,S
OPEN PLC 30 CLEAR
CLOSE
OPEN PLC 31 CLEAR
CLOSE
OPEN PROG 999 CLEAR
CLOSE
```

Use this sample file as a reference to indicate which P- variable, M-variable, I-variables are modified when you perform tuning. This file automatically is restored when you exit tuning by either exiting the tuning dialog box, or when you select the EXIT TUNING button which appears at top right of the main menu.

What to Plot

This area in the dialog box lists five check box selections allowing you to select which plots to view during the tuning process. Up to two items can be plotted at the same time. To select or deselect an item for plotting, press the space bar or click the mouse to toggle the selection.

Position

A check mark placed here selects the actual position to be plotted on the screen.

Velocity

A check mark placed here selects the actual velocity to be plotted on the screen.

Acceleration

A check mark placed here selects the actual acceleration to be plotted on the screen.

Following Error

A check mark placed here selects the following error to be plotted on the screen.

DAC Output

A check mark placed here selects the DAC output to be plotted on the screen.

Step Size (cts)

Here you specify the magnitude (maximum travel in one direction) of the step move. The units are in encoder counts.

Step Time (ms)
The move time specifies the time for each half of the step move. The units are in milliseconds.

Move Size (cts)
Here you specify the magnitude (maximum travel in one direction) of the parabolic move. The units are in encoder counts.

Move Time (ms)
The move time specifies the time for each half of the parabolic move. The units are in milliseconds.

Ix30 Proportional Gain
This I-variable provides a control output proportional to the position error (commanded position minus actual position) of motor *x*. It acts effectively as an electronic spring. The higher I*x*30 is, the stiffer the "spring" is.

Ix31 Derivative Gain
This I-variable subtracts an amount from the control output proportional to the measured velocity of motor *x*. It acts effectively as an electronic damper. The higher I*x*31 is, the heavier the damping effect is.

Ix32 Velocity FF Gain
This is the velocity Feedforward gain. This term adds an amount to the control output proportional to the desired velocity of motor *x*. It is intended to reduce tracking error due to I*x*31, analog tachometer feedback and/or frictional effects.

Ix33 Integral Gain
This I-variable adds an amount to the control output proportional to the time integral of the position error for motor *x*. The time integral of the error is limited by I*x*63. With I*x*63 at its default value of 4194304, integration is disabled, regardless of the value of I*x*33. Integration is disabled if the output saturates.

Ix34 Integration Mode
If this parameter is 1, position error integration is performed only when PMAC is not commanding a move. If this parameter is 0, position error integration is performed all the time.

Ix35 Acceleration FF Gain
This is the acceleration Feedforward gain. This term adds an amount to the control output proportional to the desired acceleration for motor *x*. It is intended to reduce tracking error due to inertial lag.

Ix29 DAC Offset
For a motor not commutated by PMAC, this is the value that is added onto the output of the servo algorithm or the open loop output value (including the zero output when the motor is killed) before it is sent to the DAC. If the analog output is unidirectional (bit 16 of I*x*02 is 1), this bias term is added after the absolute value function is performed. For a PMAC-commutated motor, this is the value that is added onto the first-phase output of the commutation algorithm (I*x*79 is added onto the second phase).

Ix69 DAC Limit
This parameter defines the magnitude of the largest output that can be sent from the control loop. If a larger value is calculated, it is clipped to this number. The analog outputs on PMAC are 16-bit DACs, which map a numerical range of -32,768 to +32,767 into a voltage range of -10V to +10V relative to analog ground (AGND). If you are using differential outputs (DAC+ and DAC-), the voltage between the two outputs is twice the voltage between an output and AGND. (If you wish to limit the voltage between DAC+ and DAC- to 10V, I*x*69 should be 16,384.)

This parameter also provides a torque limit in systems with current-loop amplifiers, or a velocity limit with tachometer-based amplifiers. Note that if this limit "kicks in" for any amount of time, the following

error will start increasing. When Ix69 is actually limiting the output, the integrator in the PID loop will turn off for anti-windup protection.

Note:

When using PMAC to do internal open-loop micro stepping (using its own commutation algorithms, not external V/F converters), the servo loop is writing to an internal register, not directly to the DACs. In this case, we can allow more than a +/-32K limit. The value of Ix69 that should be used for this micro stepping is 524,287 219-1).

Ix60 Servo Cycle Period Extension

This parameter permits an extension of the servo update time for motor x beyond the servo interrupt period, which is controlled by hardware (E3-E6, E29-E33, E98, and master clock). The servo loop will be closed every (Ix60 + 1) servo interrupts. With the default value of zero, the loop will be closed every servo interrupt. Other update times, including trajectory update, and phase update are not affected by Ix60. I10 does not need to be changed with Ix60.

Ix68 Friction FF Gain

This parameter add a bias term to the servo loop output of motor x that is proportional to the *sign* of the commanded velocity. That is, if the commanded velocity is positive, Ix68 is added to the output. If the commanded velocity is negative, Ix68 is subtracted from the output. If the commanded velocity is zero, no value is added to or subtracted from the output.

This parameter is intended primarily to help overcome errors due to mechanical friction. Indeed, it can be thought of as a ***Friction Feedforward*** term. Because it is a Feedforward term that does not utilize and feedback information, it has no direct effect on system stability. It can be used to correct the error resulting from friction, especially on turnaround, without the time constant and potential stability problems of integral gain.

If PMAC is commutating this motor, this correction is applied before the commutation algorithm, and so will affect the magnitude of both analog outputs.

Note:

This direction-sensitive bias term is independent of the constant bias introduced by Ix29 and Ix79.

Do a Step

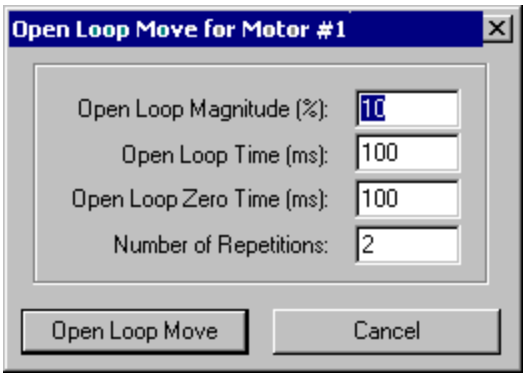
Use this button to perform a step move and view a plot of the response. The move and plot will conform to the parameters set in the tuning window.

Do a Parabolic

Use this button to perform a Parabolic Move and view a plot of the response. The move and plot will conform to the parameters set in the tuning window.

Open Loop Move

This button opens a dialog box which allows you to perform a repetitive sequence of open-loop moves according to the values set in Open Loop Mag, Open Lp Time, Open Lp Zero Time, and Number of Reps. The open loop move will be performed when you press the Open Loop Move button. Select Close or press <*ESC*> to close this screen.



Open Loop Magnitude (%)

Here you specify the magnitude (maximum voltage in one direction) for the open loop move. This is expressed as a percentage (0 - 100%) of the value stored in I*x*69 which holds the maximum allowable output magnitude (in DAC bits) which can range from 0 to 32767 corresponding to 0 to ±10V. The factory default for I*x*69 is 20480 (which is about ±6.3V), but this is usually reset to the full 32767 (±10V).

Open Loop Time (ms)

The open loop time specifies the time duration of the "on time" (when a positive or negative value is written to the DAC output) for the open loop move.

Open Loop Zero Time (ms)

This specifies the time duration of the "off time" (when a value of zero is written to the DAC output) for the open loop move.

Number of Repetitions

This specifies the number of repetitions for the open loop move. A value of zero indicates infinite repetitions until the space bar is pressed.

Plot Response

This option lets you view the last response plot without redoing the move.

Auto Tune

The purpose of this auto-tune option is to provide you with the ability to rapidly select the PID gains and/or calibrate the DAC offset(s) for any motor. The auto-tuner carries out this task by measuring the response of the motor/load combination to a series of test/excitation signals. Based upon the test results and the user's input on the desired closed loop response (bandwidth and damping), the program provides suggestions for the PMAC gains I*x*30, I*x*31, I*x*32, I*x*33, and I*x*35. The number of tests and the magnitude of the excitation signals (as a percentage of DAC output) are selected by the user.

The auto-tuner is intended to be used for tuning systems with motor/load combinations that are structurally stiff (no structural resonance's at low frequencies relative to the selected bandwidth). Systems with low frequency structural resonance's (e.g. motor/load combinations with long and thin shaft couplings or with long chain/belt drives) may still require the manual tuning of the PID gains and the notch filter parameters. For such systems, if high gain control action is not an essential requirement, the auto-tuner may be able to provide reasonable results. To do so the desired bandwidth should be selected at a frequency well below the dominant structural frequency of the mechanics.

Toggle Gains

If you select the Implement Later button in the screen showing the suggested auto-tune gains from an auto tuning session, the new gains will be "remembered" for later comparison between these gains and your

original gains. Pressing this button will allow you to toggle between these two sets of gains so that you may perform step and/or parabolic moves and compare the results.

Notch Filter

The Executive allows you to set up a notch filter very simply, without the need to understand how a notch filter works. Refer to the description for the notch filter under the **CONFIGURE** menu for more details about the notch filter.

Done

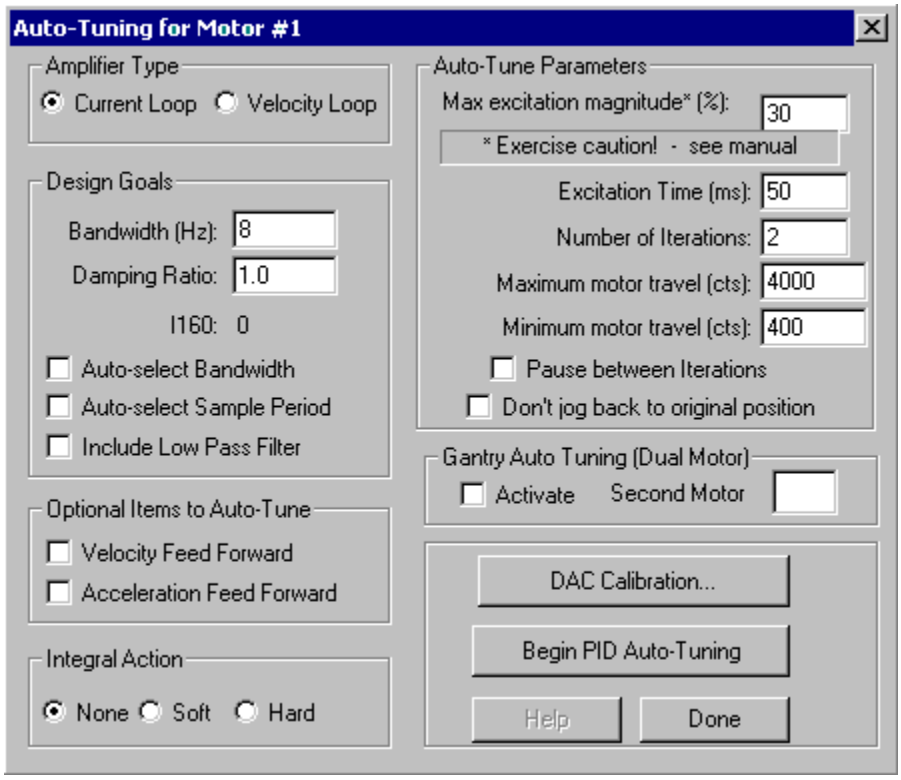
This option exits you from the main tuning dialog box, and downloads the TUNE.DAT file to PMAC, restoring all programs and variables (*except* for I*x*30 through I*x*69) modified by the tuning session.

Note:

When TUNE.DAT is created because of doing an open loop, step or parabolic move, an "Exit Tuning" button appears in the upper right area of the main menu. Selecting this button causes TUNE.DAT to be downloaded back to PMAC, thus restoring those modified programs and variables contained in this file.

Performance Auto Tuning

Before attempting to auto-tune a motor, make sure the motor is closed loop and that the fatal following error limit (I*x*11) is either set to zero (to disable this feature) or set to a high enough value so that the motor will not exceed this limit while performing the moves during auto-tuning. The default gains with which the PMAC boards are shipped are typically low enough and may be used as the initial gains for the start of a session. Furthermore, a large DC offset (bias) between the amplifier and PMAC's DAC outputs should be reduced to a minimum by the use of the bias I-variables I*x*29 and I*x*79 (refer to the PMAC's User Manual for details and the Offset Calibration menu choice under Auto Tune in this manual). Note that the auto-tuner always leaves the selected motor in the closed loop state with the *desired* motor position unchanged.



The auto-tuning session starts when you select the AUTO TUNE button in the tuning dialog box. The auto-tune dialog box then appears. At this point, you may elect to accept the default setup parameters or provide one or more of the following information:

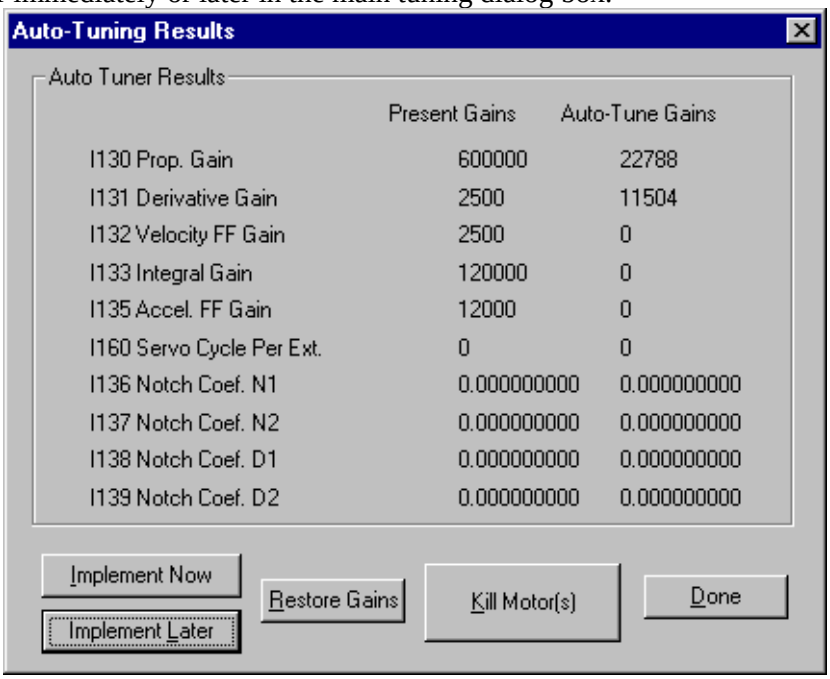
- Type of amplifier used with the selected motor (current mode or velocity mode)
- Maximum excitation magnitude as a percentage of the maximum DAC output (100% represents 32767 (10 volts))
- Excitation time in milliseconds
- Number of test iterations
- Design goals for bandwidth and damping
- The maximum and the minimum travel limits for the tests
- Choice of auto-tuning feed forward gains (either velocity or velocity plus acceleration feed forward--this determines the values of I*x*32 and I*x*35)
- Choice of no integral action, soft integral action or hard integral action (determines the value of I*x*33)
- Choice of reporting to the user the actual distance moved by the motor during each iteration cycle
- Choice of automatic selection of a "safe" bandwidth based upon the auto-tune results
- Choice of automatic selection of a low-pass filter

Once the above selections are properly specified, select the Begin Tuning button. You will observe a series of forward and backward motions on the selected motor.

Note:

You may abort motion on the motors at any time by striking any key.

If the Pause between iterations option is checked, you will be informed of the actual distance moved in each iteration. If this box is not checked the auto-tuner will continue testing the system (non-stop) as many times as selected by the user in the Number of iterations input selection. After the completion of the tests, the auto-tuner computes PMAC's PID gains I*x*30, I*x*31, I*x*32, I*x*33 and I*x*35. It displays them on the screen together with the existing gains for the selected motor. You may elect to either discard them or accept them. If the gains appear acceptable, you have the choice of down loading these gains to the PMAC card either immediately or later in the main tuning dialog box.



To examine the response of the motor under the new (auto-tune) gains, select Implement Now. To compare the motor's behavior with your original gains and the new auto-tune gains, select Implement Later. When you return the tuning dialog box, select Toggle Gains to switch between your original gains and your new auto-tune gains. Use the step and the parabolic moves to test the behavior of the motor under the new set of gains. If the response is unsatisfactory, you may decide to either fine tune the gains manually or perform another auto-tune session with different goal parameters.

Amplifier Type

This selects the type of amplifier used. The default value is for a current loop amplifier. If either an analog tachometer loop is closed within the amplifier or, the amplifier operates without current feedback (in the voltage mode), the Amplifier Type should be changed to velocity loop. Also for stepper motors using PMAC's ACC-8D Opt 2, the choice should be velocity loop.

Maximum Excitation Magnitude

This is the size of the largest DAC signal given as a percentage value of

±10V. For example if only two iterations are chosen for Number of Iterations, and the Maximum Excitation Magnitude is 100%, then in the first iteration the maximum DAC output will be ±5V, in the second iteration it will be ±10V. You must choose this parameter carefully. In general, it should be large enough to overcome friction and other bias torque so that actually some motion does occur. On the other hand, it should not be too large to cause mechanical damage because of excessive torque command. The default value is 30%.

Excitation Time

The period of time (in milliseconds) in which the DAC output is driven by the test signal is entered here. This period should be chosen carefully. It should be long enough for noticeable motion to take place (at least equal to the Minimum Motor Travel). On the other hand, too long a period of time may mean excessive travel and/or excessive motor velocity. In most applications, excitation times between 50 to 100 milliseconds should be sufficient. For motors having very large inertia loads larger excitation times may be appropriate. The default value is 50ms.

Number of Iterations

This refers to the number of backward/forward motions in the testing phase of an auto-tuning session. If a number greater than one is chosen, then the peak DAC output for the first iteration is determined from dividing the Maximum Excitation Magnitude by Number of Iterations. In this way, the last iteration always corresponds to the Maximum Excitation Magnitude input. The default value is 2 iterations

Maximum Motor Travel

For safety reasons, you may specify independently the maximum travel limit during the tests. If this distance is reached, the loop is closed immediately and the motor is commanded to jog back towards it original position. The test will continue however until all the specified number of iterations are completed subject to the specified travel limit. The default value is 4000 counts.

Minimum Motor Travel

This entry allows you to see if in fact a detectable motion took place. If the Max Excitation Magnitude is relatively low and the Number of Iterations is relatively high, the first iteration's peak DAC output may become too small. As a result, no detectable motion may take place during the excitation time. By selecting an acceptable number of counts for the Minimum Motor Travel, you can safeguard against erroneous gain estimations. The default value is 400 counts.

Bandwidth

This entry indicates the desired closed loop system's speed of response and the servo stiffness. In general the higher the value of the bandwidth, the higher the computed proportional gain (I*x*30). Typical closed

loop bandwidths range from 5 Hz, for very large motor/load inertia systems, to 100 Hz for very high gain amplifiers driving extremely light motor/load inertia systems. The default value is 20 Hz. Please note that if the Auto-select bandwidth option is selected, the selected bandwidth after performing an auto-tune procedure will appear in this input box.

Damping Ratio

This indicates the desired closed loop system's rate of transient oscillation decay. In general, low damping ratios (ratios well below one) indicate large overshoots and long periods of transient oscillations leading towards instability. On the other hand, high damping ratios (ratios well above one) indicate sluggish motion. In general, the higher the value of the damping ratio the higher the computed derivative gain (I*x*31). Typical damping ratios should range between 0.6 to 1.2. In most applications, the selection for the value of the damping ratio should be unity. This provides for a theoretical response with critical damping (no overshoot to a step move) for systems controlled by PD controllers (no integral gain). Note that if integral control is used some overshoot will occur even if the damping ratio has the value of unity. The default value is 1. Please note that if the Auto-select bandwidth option is selected, a value of 1.0 after performing an auto-tune procedure will appear in this input box.

Auto-Select Bandwidth

If auto-selection of the closed-loop bandwidth is desired, check this box. This option is extremely useful if you are not sure how large a bandwidth you need. Based upon the results of the auto-tune tests, a "safe" bandwidth is determined, which in turn selects appropriate gains for the motor you are tuning. This "safe" bandwidth is typically lower than the maximum achievable bandwidth, but is guaranteed to give a stable closed-loop system. Perhaps the best method in determining your bandwidth is to first perform auto-tuning with this box checked. After you view the suggested gains, observe the selected value for the bandwidth. Uncheck this box and key in a higher bandwidth and perform the auto-tune test again, repeating until a satisfactory response is achieved.

Auto-Select Sample Period

PMAC allows you to set different servo update rates for each motor using the I*x*60 variable. If auto-selection of the servo update rate for a motor is desired, check this box. This option is extremely useful if you are not sure what type of servo update rate will produce the performance you need. The servo update rate selection is based upon the results of the auto-tune tests and the bandwidth.

Include Low Pass Filter

If auto-selection of a low pass filter is desired, check this box. This option has the auto-tuner configure a notch filter as a low pass filter within your servo loop. This is useful if your servo loop appears noisy (jittery) at the desired gain values. The filter's configuration is based upon the results of the auto-tune test and the bandwidth.

Velocity Feed Forward Gain

This box should be checked if velocity feed forward control action (I*x*32) is required/desired. The auto-tuner will automatically determine its magnitude based upon the test results and the user's input on bandwidth and damping ratio. In general, the velocity feed forward action is needed to overcome following errors proportional motor velocity resulting from back emf, viscous damping, and velocity feedback. If such lags are acceptable then there will be no need for the use of this particular control action (I*x*32 should be set to zero). Alternatively, in applications where such lags are unacceptable the recommended value of I*x*32 may be used.

Acceleration Feed Forward Gain

This box should be checked if acceleration feed forward action (I*x*35) is required/desired. The auto-tuner will automatically determine the magnitude based upon the test results and your input on bandwidth and damping ratio. In general, the acceleration feed forward gain is needed to overcome following errors

proportional to motor acceleration resulting from inertial lags. If such lags are acceptable then there will be no need for the use of this particular control action (I*x*35 should be set to zero). Alternatively, in applications where such lags are unacceptable the recommended value for I*x*35 may be used.

Integral Action

The default value for the integral action option is none. If a considerable amount of friction and/or torque (force) couplings exist, hard integral action should be selected. For less severe cases of disturbances, the soft Integral action should be sufficient.

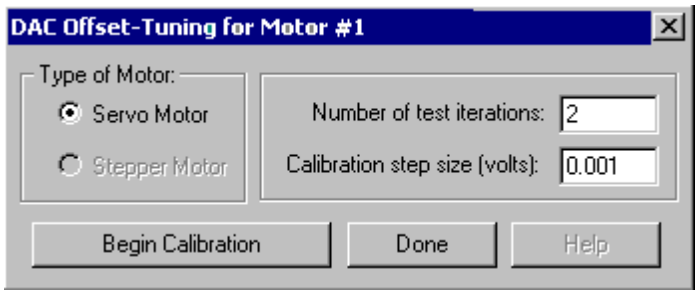
Pause between Iterations

This box should be checked if you are interested to see the distance traveled during each iteration step. Usually the first time in an auto-tuning session this option is activated. Once the optimum values for the Excitation Time and the Minimum Travel Limit are set, there will be no need to pause between iterations.

For more information about using the auto-tuner, refer to Appendix E: Doing An Auto-Tune Session later in this manual.

DAC Calibration

The purpose of this option is to provide for the user the ability to determine the DAC bias value(s).



For a standard DC motor, or a brushless motor which is commutated externally (not by PMAC), this option provides an optimum value for I*x*29. For a stepper motor, driven through PMAC's Accessory 8D Option 2 (the V/F board), this option provides optimum values for I*x*29 and I*x*79*. When you begin calibration, a small PLC program is downloaded and executed in PMAC. After the tests are completed, the optimum values for Ix29 and Ix79 (if you specified a stepper motor) are displayed. Select Implement Now to implement these bias gains in PMAC. Do not use this option if PMAC is commutating your selected motor. See also the special application note included with PMAC ACC-8D Option 2 manual, PMAC Setup for Stepper Motor Control Using ACC-8D Opt. 2, the V/F Converter.

Warning:

Regardless of the state of the selected motor's servo loop (close or open), it will be opened up by the Offset Calibration routines. In a situation where the DAC bias voltage is known to be high, the fact that the servo loop is open may lead to substantial motion during the Offset Calibration procedure. Therefore, make sure that the motor is free to rotate an unknown number of revolutions during the calibration.

Select Motor Type

This radio button box allows you to select whether you are calibrating the offsets for a servo motor or a stepper motor. If you select calibration for a servo motor, only I*x*29 will be calibrated. If you select calibration for a stepper motor, both I*x*29 and I*x*79 will be calibrated.

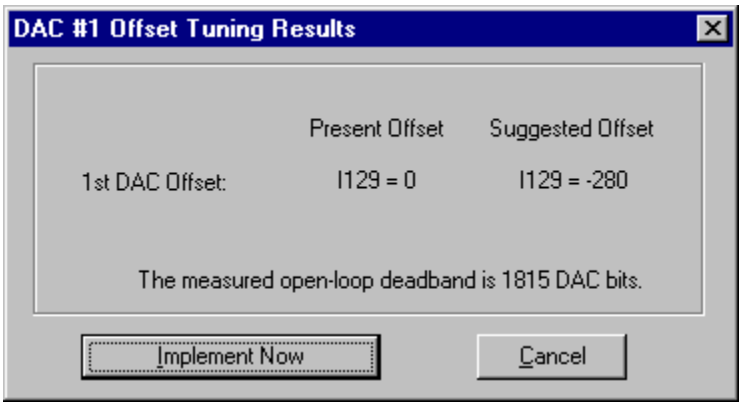
Number of Iterations

This is the number of backward and forward open loop motions in the testing cycle. Value values are from 1 to 10, and the default value is four. The calibration program automatically averages the results.

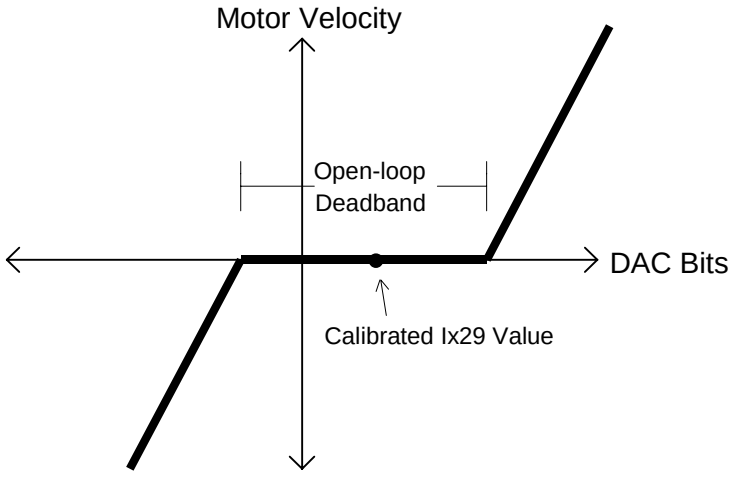
Calibrate

Selecting this button begins the calibrating procedure. Depending on the number of iterations you selected and your motor's friction and open-loop deadband characteristics, the time necessary to complete offset calibration will vary. What the procedure does is increment the DAC in small, positive steps until positive motion is detected. The DAC is then decreased in small, negative steps until negative motion is detected. This completes one iteration.

After the test is complete, the results of the calibration are presented, showing the current and suggested values for I*x*29 and (for stepper motors) I*x*79, and the open-loop deadband size, in encoder counts. This quantity *is not* the programmable PMAC deadband set by I*x*65, but rather the hardware (or open-loop) deadband resulting from the imperfections in the amplifier/motor/load combination. This quantity equates to twice the number of DAC bits which need to be added to or subtracted from I*x*29 to provide a noticeable motion in either direction. Due to amplifier deadband and/or friction in the drive the value of this quantity is often non-zero. However, it should be a small number if a responsive closed loop performance is desired. If a large deadband is reported (say larger than 500 bits) there may be a potential for imperfect servo performance particularly when the gains are low. However, *do not* set I*x*29 equal to the deadband. The value of I*x*29 should be set equal to the offset value.



Graphical Offset Illustration



Begin PID Auto-Tuning

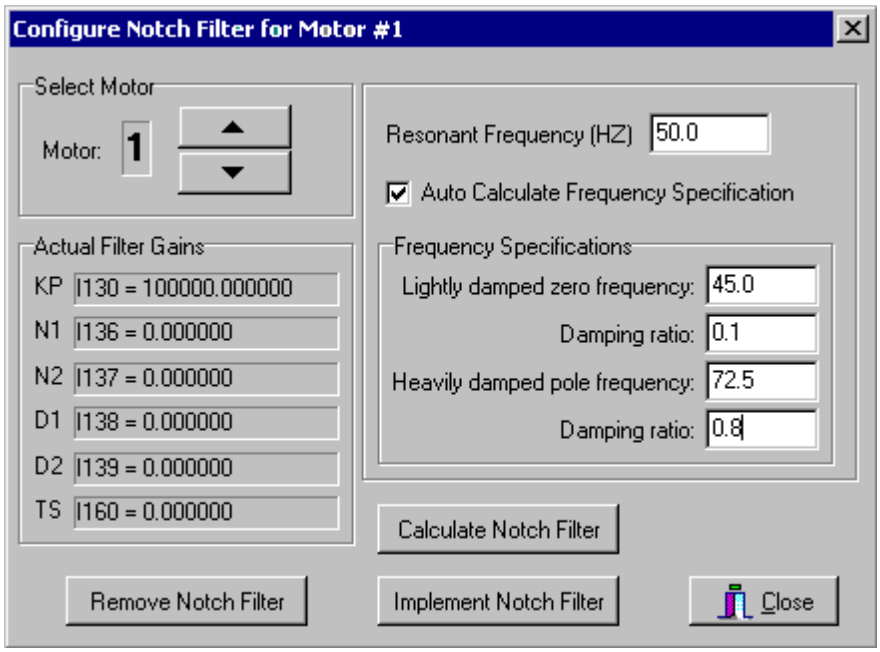
Select the Begin Tuning button and wait for the tuner to give you the suggested gains. If gains appear to be numerically reasonable, you may elect to accept them immediately by selecting Implement Now, or you may want to implement them later for comparison to your original gains by selecting the Implement Later button.

Done

Return to the main tuning dialog box by selecting the Close button.

Notch Filter

PEWIN32 allows you to set up a notch filter very easily, without the need to understand how a notch filter works.



First, select a motor using <*PgUp*> or <*PgDn*>. Next, enter the frequency of the mechanical resonance that you wish to notch out. If you have the Auto Calculate Frequency Specifications checkbox selected, then all you need to do is select Implement Notch Filter. The notch filter coefficients (I*x*36 - I*x*39) will automatically be calculated and displayed on the left, and these new coefficients will be downloaded to PMAC.

Resonant Frequency

This value is the actual frequency (in hertz) you wish to notch out in the servo loop. To determine which frequency this is, you may need to do a step move from the tuning screen and measure this frequency from looking at the plot of the step response.

Auto-Calculate Frequency Specifications

This checkbox allows you to let the Executive determine the necessary other frequencies (and their respective damping ratios) in order to calculate the notch filter coefficients. In most cases, you will want this option selected.

Lightly Damped Zero Frequency

In practice, this frequency is typically a value 90% of the resonant frequency. Remember that this value is calculated for you if you have the Auto Calculate Frequency Specifications checkbox selected

Lightly Damped Zero Frequency Damping Ratio

In practice, this damping ratio is typically a value of 0.2. Remember that this value is calculated for you if you have the Auto Calculate Frequency Specifications checkbox selected

Heavily Damped Pole Frequency

In practice, this frequency is typically a value 145% of the resonant frequency. Remember that this value is calculated for you if you have the Auto Calculate Frequency Specifications checkbox selected

Heavily Damped Pole Frequency Damping Ratio

In practice, this damping ratio is typically a value of 0.8. Remember that this value is calculated for you if you have the Auto Calculate Frequency Specifications checkbox selected

Remove Notch Filter

This button removes the notch filter for the selected motor by zeroing out the coefficients (I*x*36 - I*x*39) and restoring I*x*30 to its near original value (before the notch filter was implemented).

Calculate Notch Filter

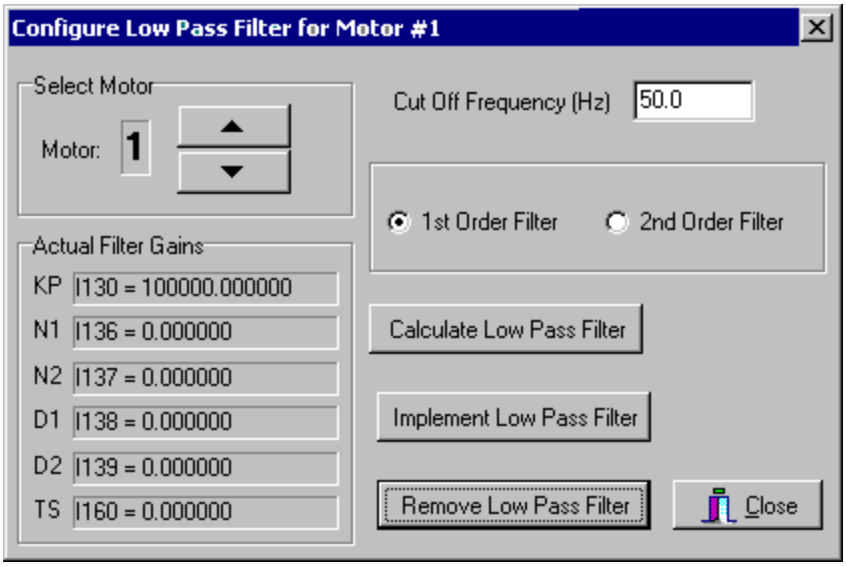
This button calculates the coefficients for the notch filter for the selected motor and displays the values on the left side of the screen. The new values are not downloaded to PMAC. When you calculate a notch filter, you will see the proportional gain (I*x*30) change to a higher value. This is necessary to maintain the same DC gain with the new notch filter implemented. The amount of how much I*x*30 changes will depend on the values of the notch filter coefficients.

Implement Notch Filter

This button calculates the coefficients for the notch filter for the selected motor and displays the values on the left side of the screen. The new values are downloaded to PMAC. When you implement a notch filter, you will see the proportional gain (I*x*30) change to a higher value. This is necessary to maintain the same DC gain with the new notch filter implemented. The amount of how much I*x*30 changes will depend on the values of the notch filter coefficients.

Low Pass Filter

PEWIN32 allows you to set up a low pass filter very easily, without the need to understand how a low pass filter works.



First, select a motor using <*PgUp*> or <*PgDn*>. Next, enter the cutoff frequency and select either a first order or second order filter. Now all you need to do is select Implement Low Pass Filter. The low pass filter coefficients (I*x*38 and I*x*39) will automatically be calculated and displayed on the left, and these new coefficients will be downloaded to PMAC.

Cutoff Frequency

This value is the actual cutoff frequency (in hertz) you wish to use for the low pass filter in the servo loop. To determine which frequency this is, you may need to do a step move from the tuning screen and measure this frequency from looking at the plot of the step response.

1st Order

This checkbox allows you to specify a first order lowpass filter. First order filters typically introduce little lag to your servo but do not exhibit a steep cutoff for the pass band.

2nd Order

This checkbox allows you to specify a second order lowpass filter. Typically, second order filters introduce more lag to your servo but do exhibit a good, steep cutoff for the pass band.

Remove Low Pass Filter

This button removes the low pass filter for the selected motor by zeroing out the coefficients (I*x*38 and I*x*39) and restoring Ix30 to its near original value (before the low pass filter was implemented).

Calculate Low Pass Filter

This button calculates the coefficients for the low pass filter for the selected motor and displays the values on the left side of the screen. The new values are not downloaded to PMAC. When you calculate a low pass filter, you will see the proportional gain (I*x*30) change to a lower value. This is necessary to maintain the same DC gain with the new low pass filter implemented. The amount of how much I*x*30 changes will depend on the values of the low pass filter coefficients.

Implement Low Pass Filter

This button calculates the coefficients for the low pass filter for the selected motor and displays the values on the left side of the screen. The new values *are* downloaded to PMAC. When you implement a low pass filter, you will see the proportional gain (I*x*30) change to a *lower* value. This is necessary to maintain the same DC gain with the new low pass filter implemented. The amount of how much I*x*30 changes will depend on the values of the low pass filter coefficients.

Feedback Tuning with Step Response

Step response is often used as a method of evaluating a feedback filter. Many controls textbooks contain information on interpreting step responses for establishing proper feedback, particularly for second-order systems (current-controlled motors driving inertial loads are second-order systems). In a step response, a sudden change is made to the command position and the feedback filter attempts to bring the system to this new position. In observing how the system gets to the new position, we can deduce a great deal about the properties of the system. It does not matter that you will not ever create such a large instantaneous step in position in the actual operation of your system. The purpose of this "jolt to the system" is to bring out system characteristics that might otherwise not be obvious. This detailed information on the PID filter is not essential to performing the tuning, but is included here for references.

PMAC has three feedback parameters to be adjusted in this process:

Kp: Proportional gain (I*x*30)
Kd: Derivative gain (I*x*31)
Ki: Integral gain (I*x*33)

We will be looking at three key step response parameters to set the feedback:

Rise Time: the time it takes the system to go from 10% to 90% of the commanded step (natural frequency is directly related to this).
Overshoot: the percentage past the commanded step that the system travels (damping ratio is directly related to this).

Settling time: the time it takes the system to get and stay within 5% of the commanded step

Typically, what is desired is a quick rise time with little or no overshoot and quick settling time. The case of critical damping, which is the fastest possible rise time that creates no overshoot, is often the goal. There are usually tradeoffs between these parameters, particularly between fast response and low overshoot. If your amplifier has a tachometer, the tachometer is providing derivative gain (and therefore damping) within the amplifier itself. If the amplifier has been well tuned, you should not have to add any more derivative gain in the digital filter, but you are free to do so if you wish. On PMAC, it is possible to have the error integration active at all times by setting I*x*34 to 0, or to have it active only when the motion is stopped by setting I*x*34 to 1. While the step response for these two cases will look essentially identical, the behavior on real moves will be very different. Error integration that is active at all times can reduce following error on an extended profiled move, but at the cost of reduced system stability and of overshoot at the end of the move (which makes up for the lag at the beginning of the move). In a system without feedforward, the close following may be worth these costs. But the velocity and acceleration feedforward terms in PMAC can virtually eliminate following error without these drawbacks. For this reason, most PMAC customers use error integration only when motion is stopped -- where it can eliminate steady state errors due to static friction or net torque loads.

Because the proportional gain term Kp (I*x*30) is outside the brackets in the filter equation (see previous page), it also affects derivative and integral gains, and is not strictly speaking a true proportional gain. For this reason, if you modify Kp when Ki (I*x*33) or Kd (I*x*31) is not equal to zero, you are also changing the effective integral or differential gain. The shape of the response curve will not change much, although its timing will. You will want to change Ki and/or Kd in the opposite direction from Kp if you want to keep their effective gains constant.

Feedforward will affect step response even though it has no effect on the system stability we are really evaluating. Be sure that both acceleration and velocity feedforward are set to zero as you are doing the step responses.

The default step size of 100 encoder counts may or may be adequate. The guidelines are to make the step large enough so that the granularity of the position measurement is not a nuisance, but small enough so

that the filter does not saturate on the step (the step size times proportional gain should be less than 178,950,000 with some margin; for instance a step size of 3000 with a proportional gain of 60000 will saturate, giving a misleading response).

Some systems will have mechanical resonance's in the coupling of the motor to the load. The PID filter cannot compensate for these resonances; if their presence is not tolerable, you must keep the gains low enough not to stimulate them, or (preferably) stiffen the coupling to reduce the resonance's. (See the examples of a system with resonance, below, near the end.)

Doing the Step Response

1. Set a safe starting filter with a little proportional gain, with no (or almost no) derivative or integral gain, and no feedforward. The current values for Kp, Ki, and Kd are displayed on the screen. See the figures on the following pages for sample values.
2. Do a step move and observe the plotted response displayed on the screen, along with the calculated statistics.
3. Adjust (probably increase) Kp (Proportional Gn) to get the fastest rise time possible without a huge amount of overshoot. Allow more overshoot here than you will in your final response.
4. Once you have a fast response, increase Kd (Derivative Gn) to bring down the overshoot to the desired value. Note that this will also increase the rise time.
5. You may need to do further tradeoffs between Kp and Kd to get the desired response.

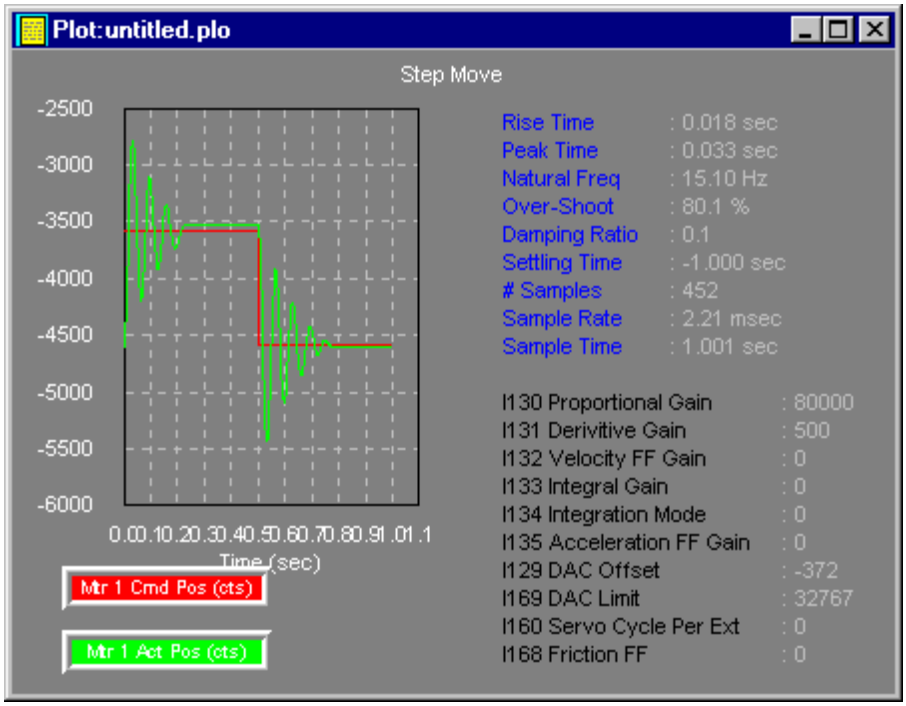
Note:

You may wish to change the size and/or the duration of the step to be able to observe the response better. The default values are a 100-count step with a 500 millisecond dwell time.

6. Once you have set Kp and Kd, you have taken care of your dynamic step response (provided you are using error integration only in position). Now you will want to add integration to improve the static holding properties of the system. As you increase Ki (Integral Gn) and observe the step response, you will notice that it increases overshoot but comes back to the command position more quickly. A good value for Ki is one that brings the response back down to the command position as quickly as possible without going back past it.

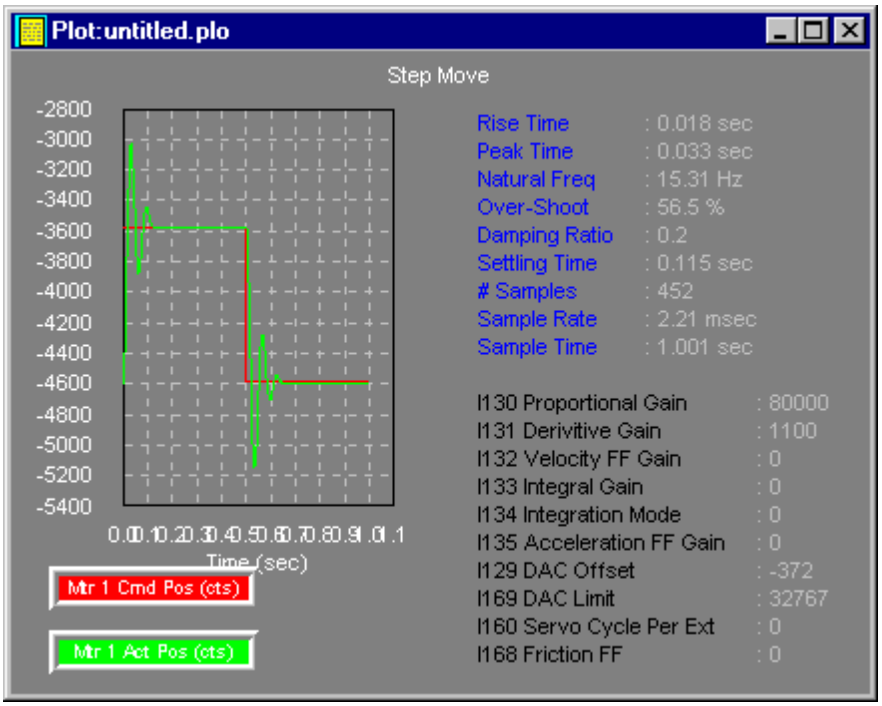
The following figures are taken from the actual tuning procedure for a motor using this program. They were done on a small DC laboratory motor with virtually no load, so the response is faster than it would be in almost any real-world application. The actual times are not important, however, the shapes of the response curves are. This system has a current-controlled amplifier with no tachometer. The goals of this system are critically damped step response, the quick elimination of steady-state errors at rest, and the minimization of following errors.

Initial step move response.



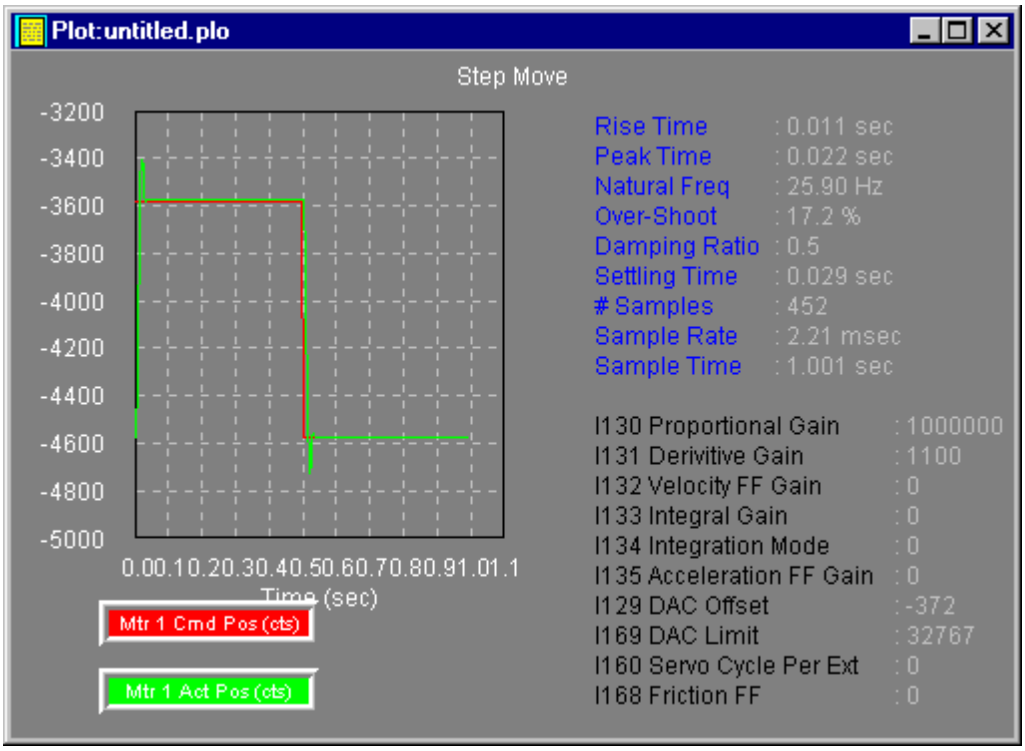
The screen on the previous page shows the step response with very little damping added (derivative gain Kd = 500). The large amount of ringing is obviously unacceptable; therefore some more Kd is needed. The next figure shows the response with an increased Kd of 1100. The ringing is largely eliminated; there is an overshoot of 60% and a rise time of 24 msec. Let us see if we can make the response quicker (which means a stiffer system).

Modified step response with higher Kd (Ix31).



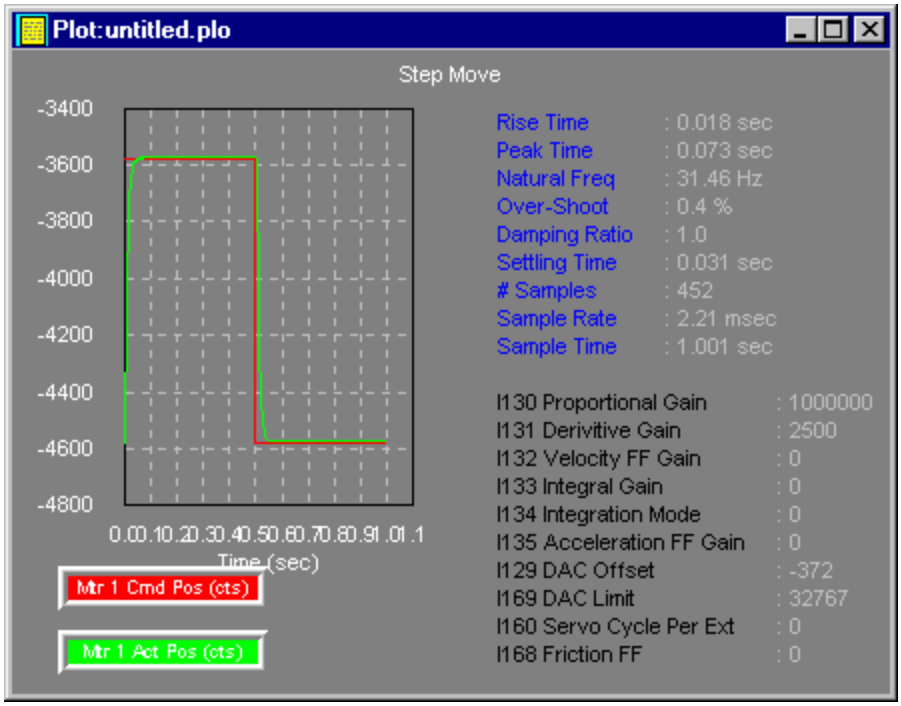
The next figure below shows the response with Kp increased from 80000 to 1000000. Note that the shape of the curve has not changed much (this is because the effective derivative gain is increasing with Kp), but the rise time has improved slightly (to 18 msec). We turn now to Kd.

Modified step response with higher Kp (Ix30).



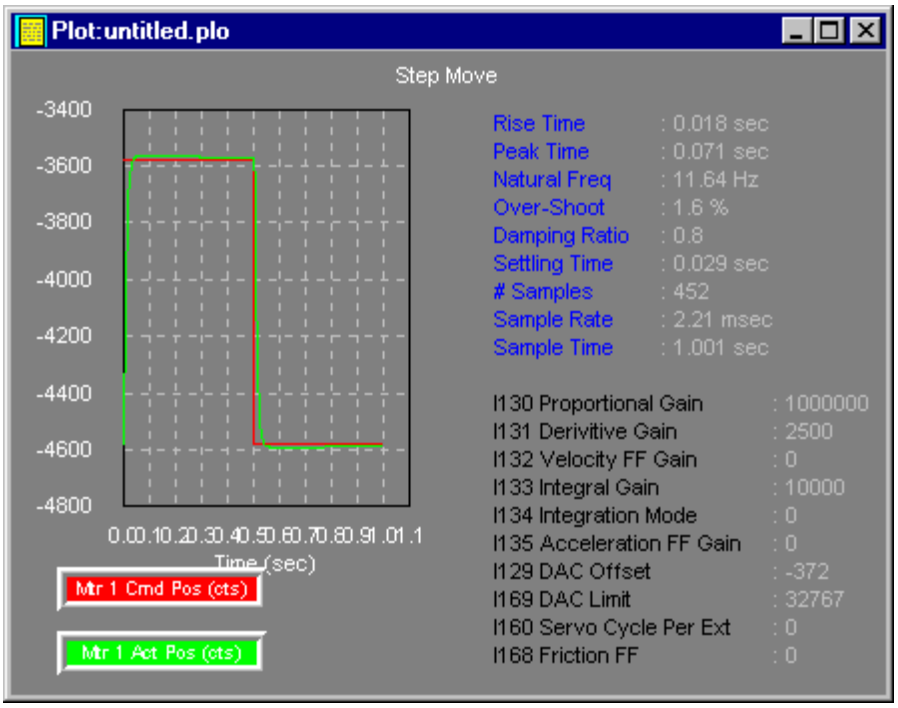
The figure below shows the step response with a Kd of 2500. This is the critically damped case; i.e. fast response with no overshoot. With Kd any smaller, we get some overshoot. In addition, with it any larger, we just slow down the response. The tendency of the system to settle slightly off from the target position is due to a net torque or static friction. We will eliminate this with integral gain.

Modified step response with increased Kd again.



The figure below shows what happens with a little bit (relatively speaking) of integral gain (Ki = 10000): the steady- state error is gone, but the nature of the curve has not really changed.

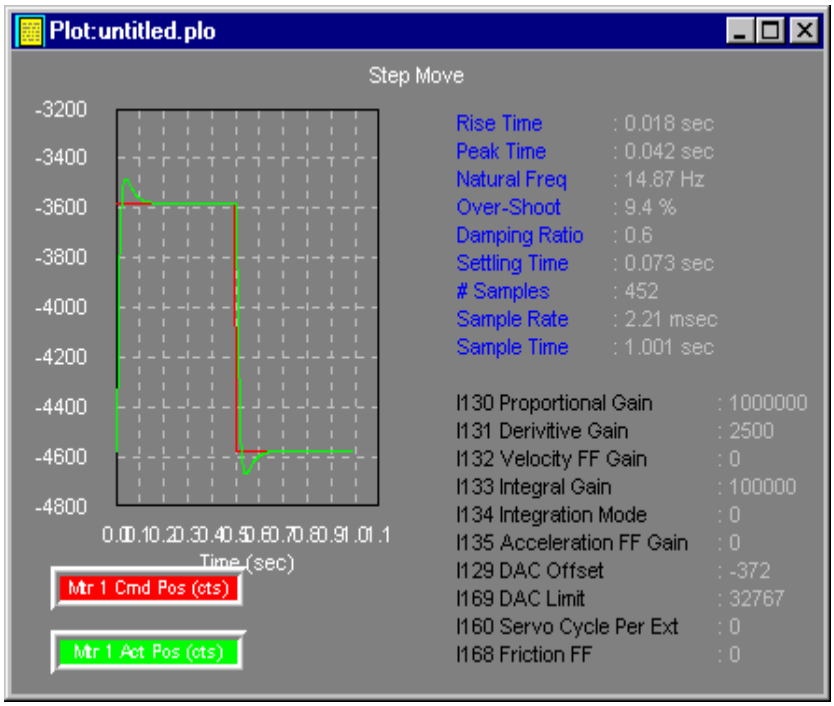
Modified step response with increased Ki (Ix33).



The figure below shows the response curve for a substantially increased Ki (100000). This curve demonstrates how quickly the system would respond to a disturbance while in position. (Remember that

we are using integral gain only when in position, so changing integral gain does not affect our actual dynamic response, although it will change the shape of the step response here. We still have the stability characteristics of critical damping while on the move.) Even the higher value of Ki does not result in oscillations, so we will use that. We have achieved what we wanted with feedback.

Modified step response with very high Ki.



Now we tackle feedforward.

Feedforward Tuning with a Profiled Parabolic Response

In a position servo system without feedforward or dynamic error integration, there must be a continual error between the commanded position and the actual position in a profiled move (known as following error) to produce a motor command. This means, however, that you are never really where you want to be when you want to be there, which is often the point of a profiled move in the first place. These following errors will usually be proportional (well correlated) to the velocity and/or the acceleration. The velocity and acceleration feedforward terms can be used to reduce these following errors virtually to zero. These parameters add terms to the torque command that are proportional to the commanded velocity and acceleration, respectively, in each cycle of the profiled move.

Mathematically speaking, if two sets of data, such as velocity and following error, vary in complete proportion to each other, they have a correlation of 1.0 (perfect correlation). If they vary completely independently of each other, they have a correlation of 0.0 (no correlation). The more they vary in proportion to each other, the closer their correlation will be to 1.0. In graphical terms, the more two curves are shaped like each other, the better they will be correlated. Another important figure is the constant of proportionality between the two sets of data, which is the average ratio between matching points in the sets. Even if two sets of data are very well correlated, the ratio may be so low that one of the sets is negligible in practical terms.

For each move done, the program will calculate the correlation between velocity and following error, and between acceleration and following error. It will also calculate the average ratio between following error and both velocity and acceleration. As a feedforward gain is increased, the ratio of following error to that quantity will decrease linearly (e.g. if a gain of 0 produces a ratio of 12.0, and a gain of 5000 produces a

ratio of 6.0, then a gain of 10,000 can be expected to drive the ratio to zero). The ratio will decline even as the correlation stays high. When the ratio gets small enough, the correlation should decrease as some other factor becomes the dominant cause of following error (e.g. noise or the other feedforward gain). Ideally, the correlation will be brought near zero as well as the ratio.

PMAC has two feedforward terms:

1. Kvff: Velocity Feedforward gain (I*x*32)
2. Kaff: Acceleration Feedforward gain (I*x*35)

The strategy in this section is to do a series of "parabolic" moves (cubic in position, parabolic in velocity, linearly varying in acceleration), while adjusting the feedforward terms to reduce the following error and its correlation to velocity and acceleration. After each move, the program will automatically calculate the correlation's and ratios, and the maximum following error. These will be displayed on a plot of the position and the following error. The feedforward terms are increased from zero (working first with velocity feedforward) until the ratios (and hopefully the correlation's) are as close to zero as possible, without going strongly negative. If either correlation goes very far negative, you will be likely to get overshoot at the end of a move.

To get meaningful correlation information, particularly about acceleration, you must push the system hard. By increasing the length and/or decreasing the time of the move, you can get higher velocities and accelerations. Decreasing time is appropriate if increasing the length would cause problems with maximum travel or top velocity.

In a system without a tachometer, you will probably want to set the velocity feedforward equal to the derivative gain or slightly greater. In a system using a tachometer, the velocity feedforward should be greater than the derivative gain.

When you do the parabolic move without any feedforward, you will probably see a very high correlation to velocity (≈1.0) and almost no correlation to acceleration (≈0.0). There is most likely a real correlation to acceleration, but it is swamped out by the velocity correlation. As you reduce the velocity correlation, you should see increased acceleration correlation. When you then reduce the acceleration correlation, the correlation to velocity may increase again, but the actual ratio and magnitude of the following error should be very small.

Parabolic moves were chosen for this procedure because their acceleration and velocity vary continually and are uncorrelated to each other, making them ideal for this type of analysis. For further examination of the move, you may plot the velocity curve, the acceleration curve, or the following error curve. These are automatically scaled to fill up the plot window for maximum resolution. The move statistics are re-displayed with these plots.

Doing the Parabolic Move

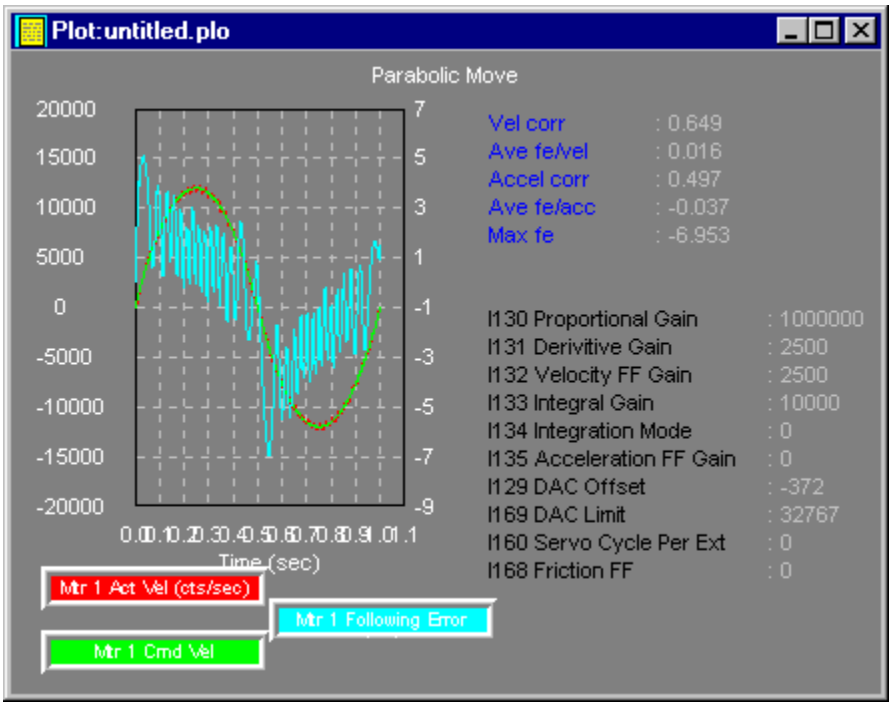
1. Do a parabolic move and observe the plotted response and the calculated statistics for the move.
2. Presuming there is a high correlation between following error and velocity (the velocity and following error curves have the same shape) and a relatively high maximum following error (as shown on the plot), increase Kvff (Velocity FF Gn I*x*31).
3. Do another parabolic move. If there is still inadequate Kvff, there will still be a high correlation, but the FE-to-Vel ratio and the maximum FE will be reduced. Repeat steps 2 and 3 until you have the maximum Kvff that produces a square wave looking shape for the following error.
4. At this point, there should be noticeable correlation between acceleration and following error. You can see this by the numerical correlation value, or by plotting the acceleration and the following error, and noticing the similarity in shape. If you do not see any correlation, try increasing the length or decreasing the time of the move to get higher accelerations.

5. Increase Kaff (Accel FF Gn I*x*35). Do another move, and evaluate the correlation and FE-to-Accel ratio again. Repeat until you have the maximum Kaff that retains any positive correlation. At this point, you should have minimal following error, and most of what remains should be caused by noise or mechanical friction. If mechanical friction is the cause, you will see a strong velocity correlation because the friction causes the following error related to the sign of the velocity. However, each half of the following error curve should be quite flat if friction is the primary cause.

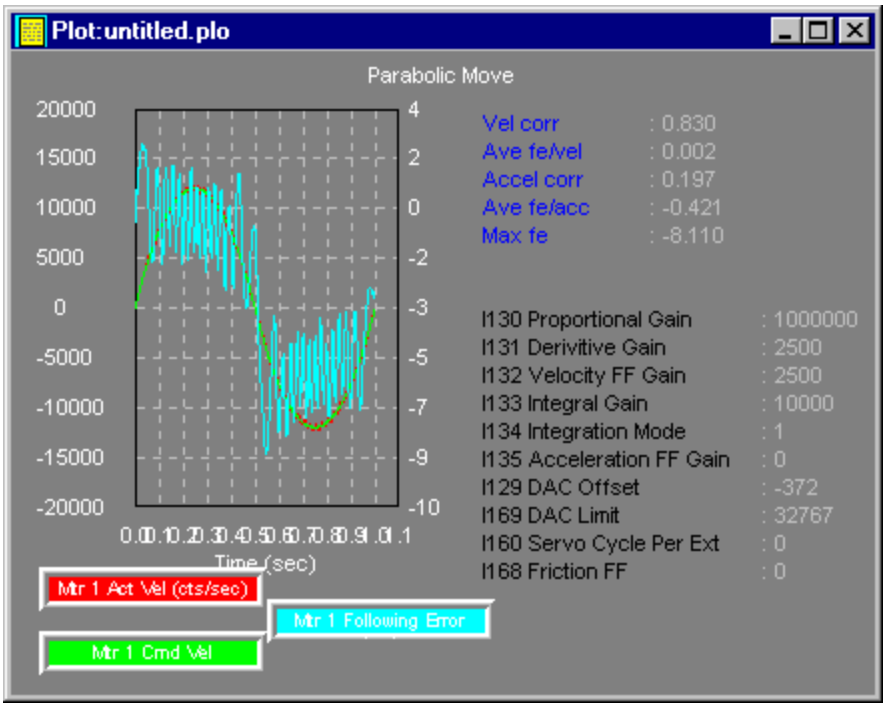
In the first figure below, we see the plot of a parabolic move using a step size of 4000 counts and a step time of 500 milliseconds. Here, position is plotted against time, with the statistics of the following error shown. The following error gets as large as 112 counts, and it is virtually perfectly correlated to velocity (Vel corr = 0.913). It shows very low correlation to acceleration (Acc corr = 0.226). The next two figures show the plot of position, and velocity and following error, versus time for the same move. Note that the velocity and following error curves are shaped almost exactly alike. This is a ramification of their high correlation.

Now let's introduce some velocity feed forward. As a rule of thumb, choose Kvff (I*x*32) to be equal to Kd (I*x*31). In this case, Kvff = 2500. By looking at the following error curve, it no longer resembles the velocity curve (correlation of 0.211). In addition, the maximum following error is reduced from 112 to 9.6 counts!

Modified parabolic move with increased Kvff (Ix32).

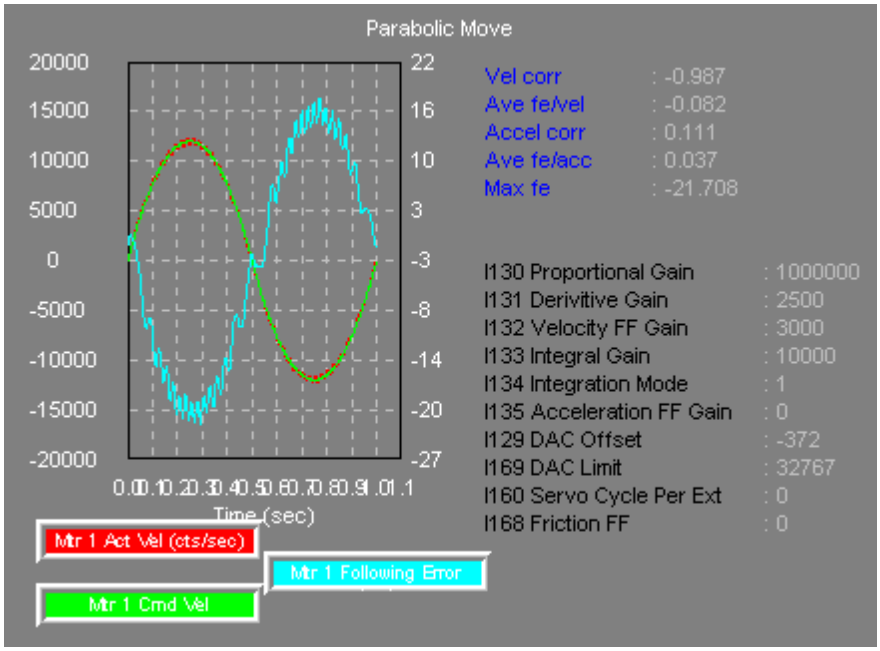


Modified parabolic move with Ix34 = 1.

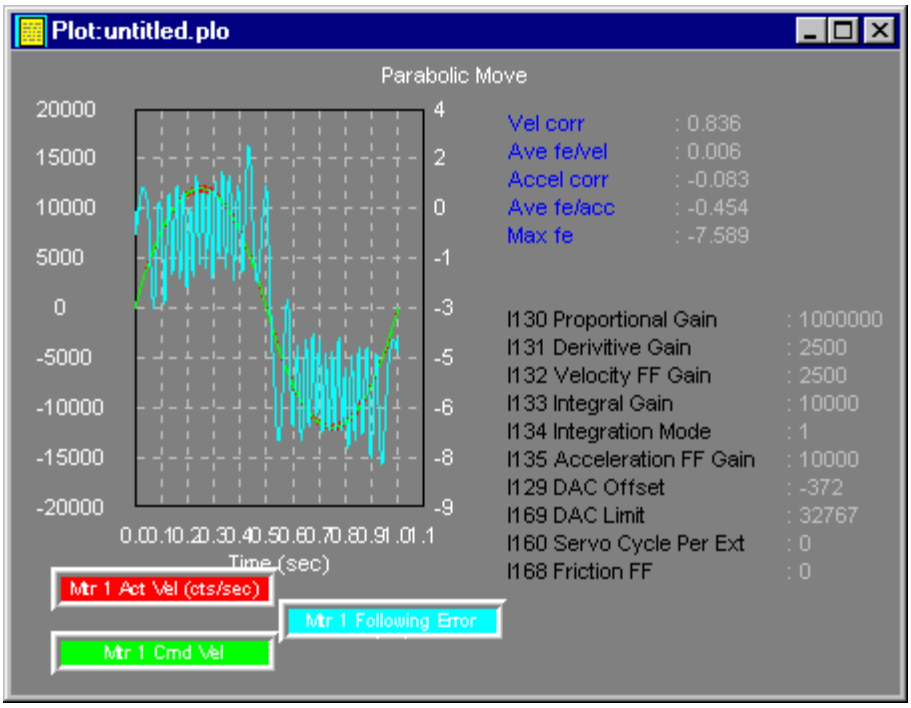


Now we instruct the integration in the servo loop to be active only when the motor is not be commanded to move. To do this, set I*x*34, the integration mode, equal to 1. The improvement is seen above. If we continue to increase Kvff, then we have the following behavior where it again resembles the velocity curve, but in an inverted manner.

Modified parabolic move with too much Kvff.



Modified parabolic move with some Kaff.



Finally, a bit of acceleration feedforward, Kaff (I*x*35), gives us an almost perfectly tuned motor!

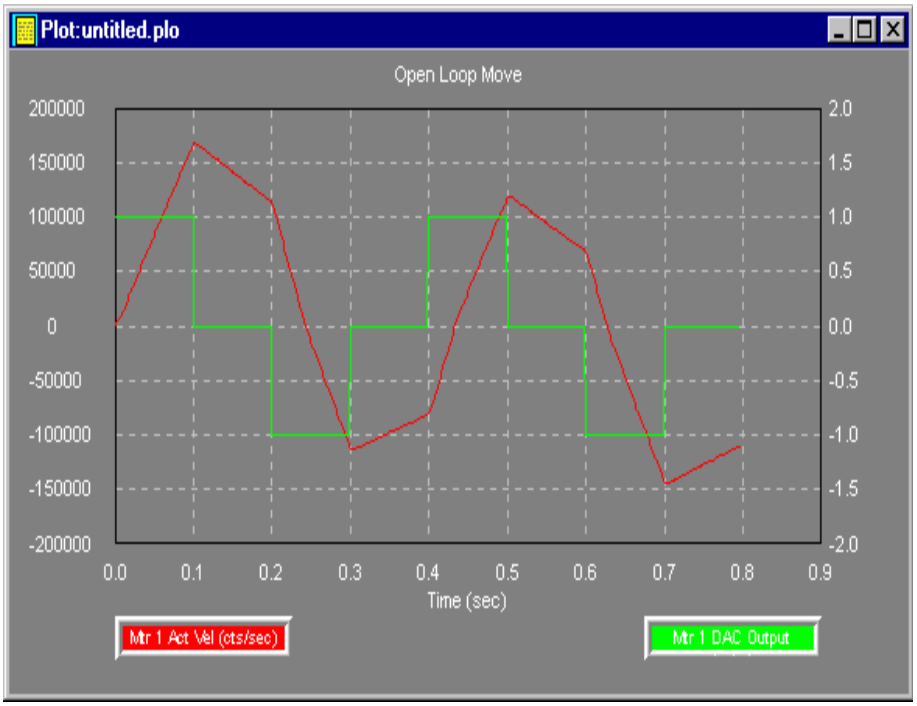
Open Loop Amplifier Tuning

This section of the program allows you to program a repetitive sequence of open-loop outputs from PMAC using its O command feature. This is very useful for tuning amplifiers, particularly those with tachometer feedback. It can take the place of special hardware devices that perform the same purpose. In addition, it has safety features that no hardware device can have: automatic shutdown on exceeding (programmable) velocity and position limits. It also uses data gathering and plotting to display the results of the stimulation (velocity, position, and acceleration versus time), eliminating in many cases the need for an oscilloscope to do the tuning.

You will need to set the open loop magnitude, which is expressed as a percentage (0 - 100%) of the value stored in I*x*69. I*x*69 holds the maximum allowable output magnitude (in binary bits) which can range from 0 to 32767 corresponding to 0 to 10 volts. The factory default for Ix69 is 20480 (which is about 6.3 volts), but many users reset this to the full 32767. The open loop time specifies the time duration of the "on state" (when a positive or negative value is written to the DAC output) for the open loop move. The Open loop zero time specifies the time duration of the "off state" (when a value of zero (0) is written to the DAC output) for the open loop move. No of reps specifies the number of repetitions for the open loop move cycle. A value of zero (0) indicates infinite repetitions until the space bar is pressed.

When the open loop program is executed, the moving motor's velocity will be limited to the velocity limit as specified by I*x*16. This is intended to protect the system by preventing it from ever moving too fast. This feature depends on having proper encoder feedback to the card, of course. The open loop program is terminated if the velocity limit is exceeded in either the positive or the negative direction. In addition, if either the hardware or the software position limits are exceeded, the open loop program will stop. The hardware limits are the external limit switches connected to PMAC. The software limits are defined by I*x*13 and I*x*14 for the positive and negative position limits, respectively.

Open Loop Move



Auto-Tuning

Here are some guidelines for doing auto-tuning on a motor.

1. Jog the motor to a safe place where both forward and reverse motions of at least twice the size of the Maximum Travel Limit are acceptable.
2. With a low gain PD loop closed adjust I*x*29 (for steppers first adjust I*x*79) so that offsets are substantially reduced. You may use the Calibrate Offsets menu option to do this.
3. Select the Tuning option from the Configure menu, choose the desired motor, and enter the auto-tune dialog box.
4. Select the correct amplifier type (current or velocity loop).
5. Select the desired bandwidth and natural frequency for the closed loop system. You may want to use the Auto-select bandwidth option to choose a bandwidth for you.
6. Select the type of feed forward and integral action desired.
7. Select the Begin Tuning button and wait for the tuner to give you the suggested gains. If gains appear to be numerically reasonable, you may elect to accept them immediately by selecting Implement Now, or you may want to implement them later for comparison to your original gains by selecting the Implement Later button. Selecting either Implement Now or Close will return you to the auto-tune dialog box.
8. Return to the main tuning dialog box by selecting the Close button.
9. Using the auto-tuner gain recommendations for I*x*30 and I*x*31 and with I*x*32, I*x*33, and I*x*35 equal to zero carry out a step response test. If the response is satisfactory in terms of natural frequency and damping ratio, then set I*x*32, I*x*33 and I*x*35 to their respective recommended values and carry out a parabolic response test. If the response is satisfactory in terms of its adherence to the desired goals then you are finished tuning! In the event the response is unsatisfactory, you may try auto-tuning again using different design parameters. If the actual response was too oscillatory, reduce the desired bandwidth. If the actual response was too sluggish, increase the desired bandwidth. The damping ratio should be left at or around one.

Digital Current Loop Auto-Tuning

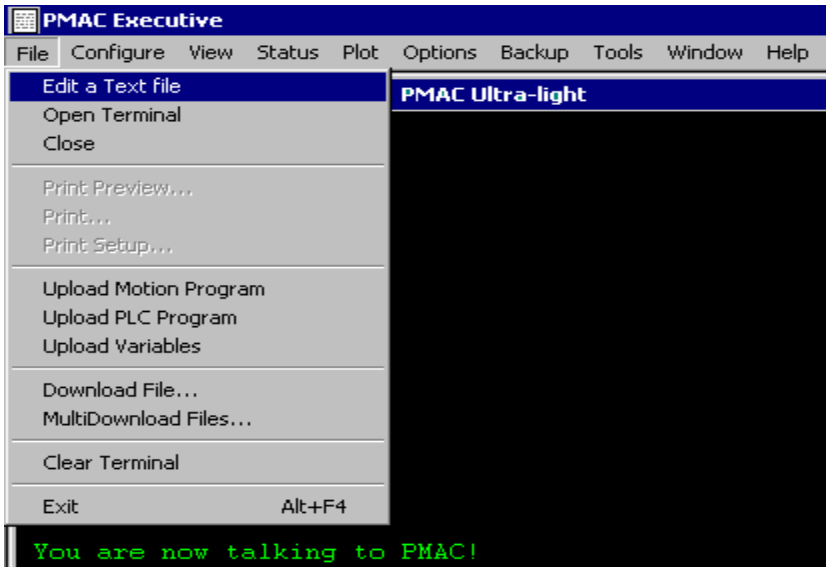
This is for the PMAC II using digital amplifiers. This feature exists in our DOS version, but has not been implemented yet in the Windows version.

PMACEDITOR

Start PMACEditor

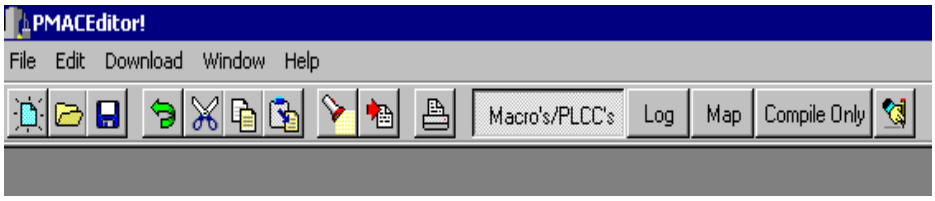
PMACEditor is a text editor provided for users convenience. It is a separate application and can be launched from the **START** menu in PEWIN32 programs group.

PMACEditor can, however, be launched from PMAC Executive as well through the **EDIT A TEXT FILE** in the **FILE** menu during an active terminal session.



How Menus Work

PMACEditor uses a static menuing system. However, most of the menus are highlighted during an active file. The standard menu displayed when the editor has focus, looks like this:

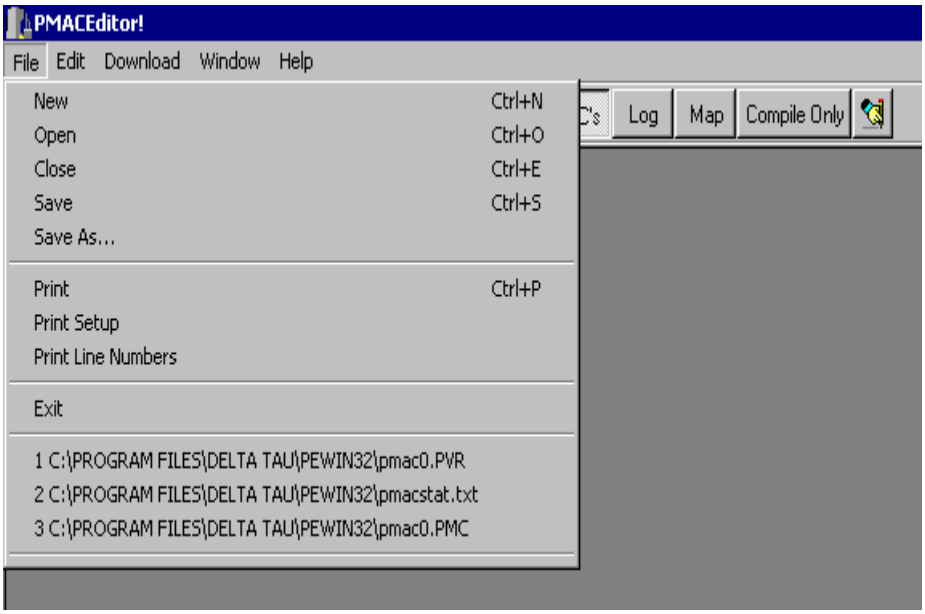


Most of the PMACEditor menus are similar to editors one with about the same attributes. There is also a menu bar beneath the Editor bar that has quick buttons for the PMACEditor compiler.

 This button compiles and downloads the file (MACRO/PLC) to the PMAC.

File Menu

The File menu handles the transfer of files or programs to and from PMACEdtor as well as printing of these files.



NEW - will allow you to create a new PMC, or PLC file.

OPEN - will open a PMC, PLC, CFG file.

CLOSE - will close the highlighted file window.

SAVE – will save the highlighted window.

SAVE AS - will save the current window to a new file.

PRINT - allows you to print the editors contents.

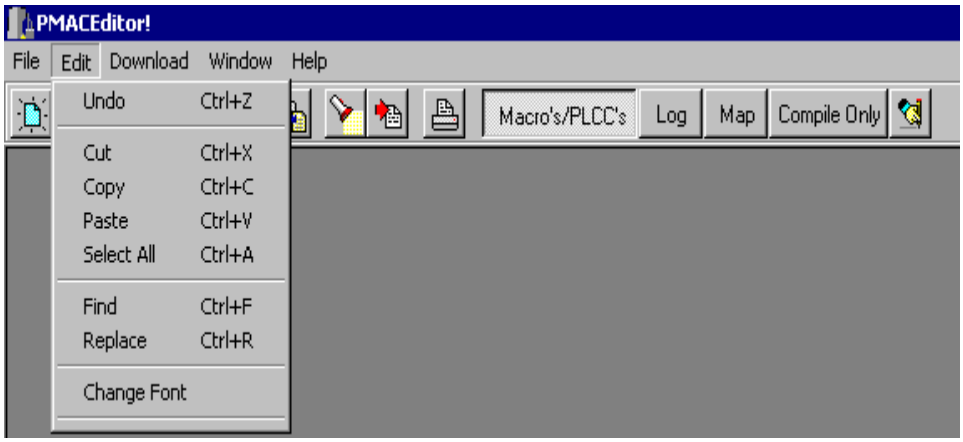
PRINT SETUP - configures your printer.

PRINT LINE NUMBERS - numbers the lines at the print out.

EXIT - closes the program, and is followed by a list of saved in the histroy file.

Edit

Through the Edit menu, the user can do all sorts of changes in the text as if it was a text file and using an editor.



UNDO - to undo your last action.

CUT - first select the text so you can move it to another location or delete it, and then on the **Edit** menu click **Cut**.

COPY - first select the text, you want to paste it or copy in another location, and then on the **Edit** menu click **Copy**.

PASTE - first place the cursor where you want to paste the text that you have cut or copied, and then on the **Edit** menu click **Paste**.

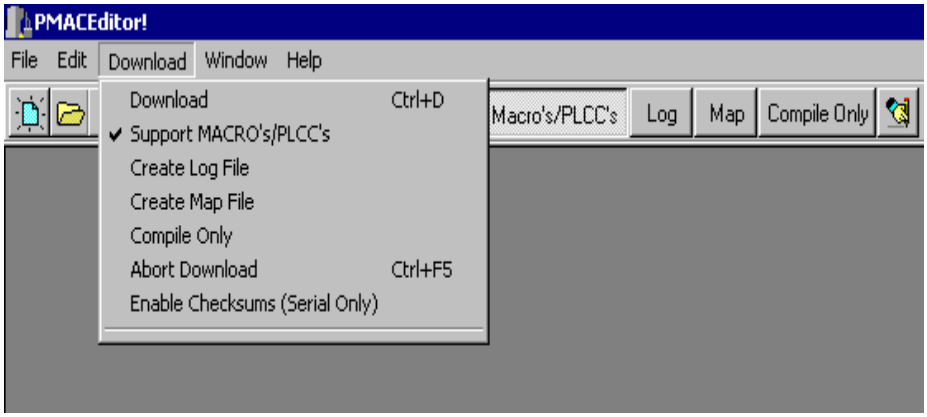
SELECT ALL - You can select all text at once by clicking **Edit** menu, and then clicking **Select All**.

FIND - On the **Edit** menu click **Find**, and type the characters or words you want to find.

REPLACE - On the **Edit** menu click **Replace** and you can search for and replace words or characters with text that you specify.

CHANGE FONT - first select the text that you want to change. Then on the **Format** menu, click **Font**, and change the font.

Download



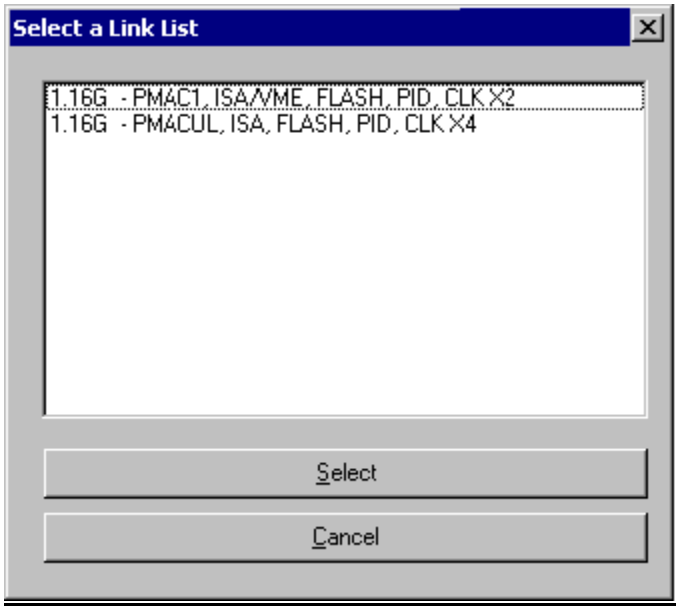
DOWNLOAD - download the file (MACRO or PLC) to PMAC.
SUPPORT MACRO's/PLCC's - when downloading, parse file for #Define statements, and support MACRO's/ PLC's.

CREATE LOG FILE - create a LOG file when downloading, *.LOG.

CREATE MAP FILE - create a MAP file when downloading, *.MAP.

COMPILE ONLY - compiles the file only.

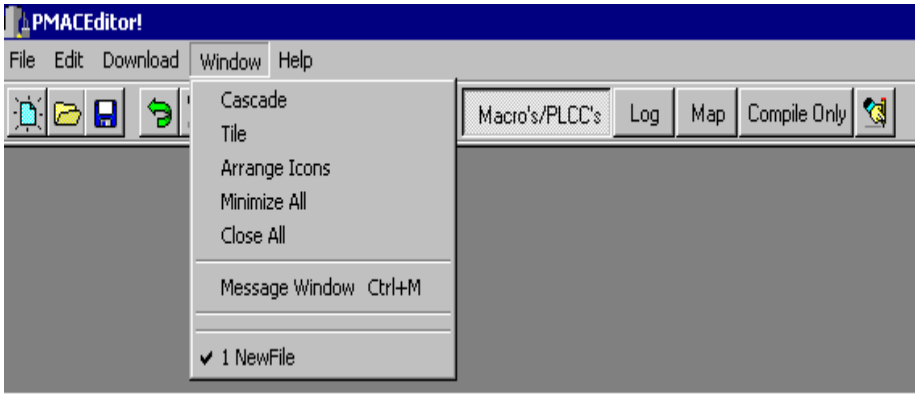
If you have selected the **COMPILE ONLY** mode, when you **DOWNLOAD** the file the below screen will show up.



ABORT DOWNLOAD - aborts the download.
ENABLE CHECKSUMS(Serial only) - this controls how PMAC reacts to serial character errors (parity and framing), if found.

Window

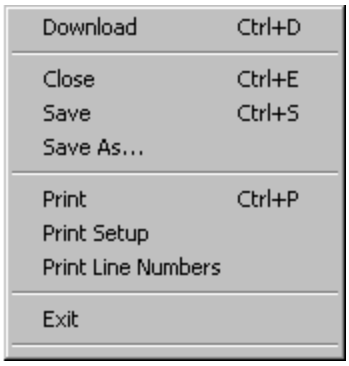
Window related commands:



CASCADE - arrange open windows to overlap.
TILE - arrange all open windows without overlap.
ARRANGE ICONS - arrange all open window icons at bottom of the main window.
MINIMIZE ALL - minimizes all windows.
CLOSE ALL - closes all windows.

Mouse Right Click Menu

Pressing the right button of the mouse at an active editor file, a new menu window will open (see below picture). This menu is a summary of the most common actions at the PMACEditor.



APPENDIX A - UNDERSTANDING PMAC'S WATCHDOG TIMER

PMAC's Watchdog Timer

The PMAC motion control board has an on-board "watchdog timer" (sometimes called a "dead-man timer" or a "get-lost timer") circuit whose job it is to detect a number of conditions that could result in dangerous misfunction, and shut down the card to prevent a misfunction. The philosophy behind the use of this circuit is that it is safer to have the system not operate at all than to have it operate improperly.

Because the watchdog timer "wants" to fail, and many components of the board, both hardware and software, must be working properly to keep it from failing, it may not be immediately obvious what the cause of a watchdog timer failure is. This application note is a guide to examining the possible causes of watchdog failure.

How the Watchdog Timer Works

The hardware circuit for the watchdog timer requires that two basic conditions be met to keep it from tripping. First, it must see a DC voltage greater than approximately 4.75V. If the supply voltage is below this value, the circuit's relay will trip. This prevents corruption of registers due to insufficient voltage. The second necessary condition is that the timer must see a square wave input (provided by the PMAC software) of a frequency greater than approximately 25 Hz. If the card, for whatever reason, due either to hardware or software problems, cannot set and clear this bit repeatedly at this frequency or higher, the circuit's relay will trip.

In the PMAC software, the task that actually writes to the watchdog timer is called the "real-time interrupt" (RTI). This task is supposed to execute at a fixed number of servo cycles. The number of servo cycles is determined by variable I8 -- the RTI executes every I8+1 servo cycles. At the default value of 2, this is every third servo cycle. The servo cycle frequency is divided down from the master clock by an amount determined by the settings of jumpers E98, E29-E33, and E3-E6. At the default settings, the servo frequency is 2.25 KHz, so the default RTI frequency is 750 Hz. If the RTI frequency were to drop below about 50 Hz, it could not provide the 25 Hz square wave, and the timer would trip. With the default servo update rate, an I8 value greater than about 30 will cause the timer to trip.

Every RTI, PMAC reads the 12-bit watchdog timer register (Y register $1F) and decrements the value by 8 -- this toggles bit 3. If the resulting value is not less than zero, it copies the result into a register that forces the bit 3 value onto the watchdog timer. Repeated, this process provides a square-wave input to the watchdog timer.

If no other task intervened, the RTI would stop toggling the watchdog input after 512 cycles (212/8), because the timer register would have counted down to zero. The task that keeps this from happening is the "housekeeping" function in PMAC's background tasks. The background tasks execute in the time available between higher priority tasks such as servo cycle updates, and RTI tasks. The three main background tasks are responding to host commands, executing PLC programs (1-31), and housekeeping. In the background, PMAC executes one scan through an individual PLC program, then checks to see if there are any complete commands, responding if there are, then executes the housekeeping functions. This cycle is repeatedly endlessly.

Most of the housekeeping functions are safety checks such as following error limits and overtravel limits. When it is done with these checks, PMAC sets the 12-bit watchdog timer register back to its maximum value. As long as this occurs regularly at least every 512 RTI cycles, the watchdog timer will not trip.

The purpose of this two-part control of the timer is to make sure all aspects of the PMAC software are being executed, both in foreground (interrupt-driven) and background. If anything keeps either type of routine from executing, the watchdog will fail quickly.

What to Look for When the Timer Trips

The things that can trip the watchdog timer can be divided into four basic classes: constant (or quickly repeatable) hardware problems, intermittent hardware problems, constant (or quickly repeatable) software problems, or intermittent software problems. The following procedure should help you determine which type of problem it is.

If you think the watchdog timer has tripped -- this is usually suspected when the outputs all turn off and communications to the host is lost -- check the LEDs on PMAC. If both the red and green LEDs are on, the watchdog timer has tripped. If both LEDs are off, you have lost all 5V power to the card. If the green LED is on, and the red LED is off, the watchdog timer has not tripped; there is some other problem.

If the watchdog timer has tripped, reset the card either by taking the INIT/ line on the control panel connector low, then high, or by cycling power. If the red LED stays lit after the reset, you have a constant problem; if it goes off, you have an intermittent problem.

Constant Problem: If the red LED stays lit on reset, you must determine whether the constant problem is in hardware or software. The best way to determine this, is to turn the power off, change jumper E51 from its default state (take OFF for PMAC-STD, put ON for all others), then turn power back on. This re-initializes PMAC, disabling all software settings and programs that could have tripped the timer. If the red LED stays on, you have a constant hardware problem; if it goes off, you have (or at least had) a constant software problem.

Note: Re-initializing the card works by installing the factory-default I-variable values, instead of the values stored in EPROM. You do not lose the values you have stored in EPROM (unless you SAVE the default values); you simply are not using them. If you fix the hardware or jumper setting problem that was causing the watchdog trip, you can recover your old values simply by powering up the card with E51 in its default state.

Constant Hardware Problem: There are several possible causes of a constant hardware problem. The first is low supply voltage. With a voltmeter, probe between +5V and GND on one of the connectors at the card -- for example, between pins 1 and 3 on the machine connector. If you suspect variation in the supply, you may want to use an oscilloscope to catch changes.

If the supply looks OK, you will want to inspect the card itself. Turn off power and remove the card. If there are several circuit boards, make sure that the daughter boards are well seated on the mother board. Look at all the socketed ICs. Make sure that they are firmly in their sockets, with no bent or broken legs. Apply reasonable pressure to each socketed IC. Re-install the card and turn on power.

If the red LED is still on, turn off power and remove the card again. Check the board thoroughly for signs of damage: broken components, damaged circuit board, cracked solder joints, etc. Any of these types of problems will require a factory repair.

Constant Software Problem: If using the re-initialization jumper setting on E51 allows the card to power up without tripping the watchdog timer, the problem that caused the tripping was in software, almost always caused by a particular software function taking too much of the processor's time. Now you must determine which function is causing the problem.

PMAC requires at least 55 microseconds in the servo update time per activated motor. If you have too many active motors and a fast servo update time, you may starve the lower-priority tasks, including watchdog update, for time, and shut down the clock. At the default update rate of 2.25 KHZ, it is possible

to have all 8 motors active. When the card is re-initialized, only motor 1 is left activated (I100=1; I200-I800=0), so the card can run successfully at any servo update rate.

If you suspect that this may be the problem, check what your servo update rate is. First, calculate it from the master clock frequency and the settings of jumpers E98, E29-E33, and E3-E6. Second, you can confirm the frequency setting by probing with an oscilloscope the SERVO clock signal on the serial port connector. If it is true that the servo clock frequency is too fast for the number of activated motors you want, you will need to slow down the servo clock by changing jumper settings before you can re-activate all of your motors.

If the servo update frequency does not appear to be the problem, a PLC program probably is. Re-initialization solves this problem by setting I5 to 0, not permitting any PLC programs to run. PLC 0, which runs as an RTI task, is the most likely culprit, because it can starve the background tasks for time, preventing the housekeeping routine from resetting the timer register. If you suspect PLC 0, you can delete it now (OPEN PLC 0 CLEAR CLOSE), turn off power, change E51 back to its default setting, and turn power back on. If the watchdog does not trip, you have confirmed that the PLC 0 was the problem.

In general, PLC 0 should be kept very short, and usually the shorter the better. PMAC will execute an enabled PLC 0 every RTI, unless it has not finished the RTI tasks (PLC 0 and motion program) that it started in a previous RTI cycle. If a given PLC 0 takes 95% of the time available for it in one RTI cycle, all of the background tasks put together have to run in the remaining 5%, which may not let them keep the watchdog timer happy. Shortening the PLC 0 slightly, so that it only takes 85% of available RTI time, would allow background tasks to run 3 times faster, because they would now have 15% of available time. However, lengthening the PLC 0 slightly, so that it takes 105% of the time available in a single RTI cycle, would cause it to execute only every other RTI cycle, and leave the majority of the second cycle available for background time.

For a background PLC program (PLC 1 to 31) to cause a watchdog failure, it must be extremely long -- probably thousands of lines long, unless it has virtually no time to execute, in which case the real problem is a higher priority task. In general, however, if you have a lot of PLC code, it is better to split it up into several separate PLC programs, because other background tasks, such as communications response and safety checks, will operate more often.

Intermittent Problems: If simply resetting the card without re-initializing it causes the card to operate without immediately tripping the watchdog timer, you have an intermittent problem, in either hardware or software. As we all know from trying to show a car's problem to a mechanic, intermittent problems can be much more difficult to track down.

The key in finding an intermittent problem is identifying the pattern of failure. One of the key questions to ask yourself is, "What has changed since the system worked?". Try changing things back to the way they were before you were getting the failure.

Many of the possible causes of intermittent watchdog failure are fundamentally the same as the constant-problem causes listed above, but only occurring on occasion. For example, a power supply may provide inadequate voltage only when other components draw a heavy load. An important electrical contact may fail intermittently. A PMAC program may take too much time only when it is in a particular section or logical branch.

If a failure occurs repeatedly in the same section of operation of the system, the cause is probably a software time problem (unless this section cuts the supply voltage by drawing increased load). Find out what is special about the software at this point. For instance, maybe PLC 0 had been disabled, and the watchdog tripped when PLC 0 was enabled. Alternately, maybe PLC 0 had been passing over most of its calculations because of the conditional branch it was taking, but the card failed when the condition changed.

Although it is very rare for a motion program to cause a watchdog failure, this does happen on occasion. It is important to understand how the motion program executes. When a Run command is given, and every time the actual execution of programmed moves progresses into a new move, a flag is set saying it is time to do more calculations in the motion program for that coordinate system. At the next RTI, if this flag is set, PMAC will start working through the motion program. Program calculations will continue (which means no background tasks will be executed) until one of the following conditions occurs:

1. The next move or dwell is found and calculated
2. End of, or halt to the program (e.g. `STOP`) is encountered
3. Two jumps backward in the program (from `ENDWHILE` or `GOTO`) are done
4. A `WAIT` statement is encountered (usually in a `WHILE` loop)

If calculations stop on condition 1 or 2, the calculation flag is cleared, and will not be set again until actual motion progresses into the next move (1) or a new Run command is given (2). If calculations stop on conditions 3 or 4, the flag remains set, so calculations will resume at the next RTI. In these cases, where you have an "empty" (no-motion) loop, the motion program acts much like a PLC 0 during this period. These empty loops, which are usually used to wait for a certain condition, provide very fast response to the change in condition, but their fast repetition occupies a lot of CPU time, and can starve the background tasks for time. Particularly if several coordinate systems are executing empty loops at the same time, you can run into serious background time limitations, which can be severe enough to trip the watchdog timer.

If there are a huge number of lines of intensive calculations (e.g. 100) before any move or dwell is encountered, there can be such a long time before background calculations are resumed (more than 512 RTI cycles) it is possible to trip the watchdog timer. If this problem occurs, the calculations should be split apart with short `DWELL` commands to give other tasks time to execute.

APPENDIX B - PEWIN32 BUG LIST

Bug Reports

A text file BUGS.TXT is located in the installation directory of PEWIN32. Here you find what fixes were made from one revision to the next as well as the few known bugs and annoyances that currently exist. If you happen to find a bug not in the list please e-mail the symptom to: SOFTWARE@DELTATAU.COM. Please note the following things in the message:

1. What operating system are you using?
2. What kind of PMAC (1,2, MACRO,MINI,Lite, etc.)
3. Is the problem repeatable, or sporadic?
4. What leads up to the problem

GLOSSARY OF TERMS

On-line Commands

Commands that immediately affect PMAC.

Static Status Screens

PMAC information reports that are not updated in real-time, they are only updated if requested by the user. These include the program status information screen, plc program status information screen and others.

PID

Proportional Integral Derivative

Gains

How much force is applied to the motor?

Run Away

When a motor starts moving without a command being issued by the user.

DPRAM

Dual-Ported RAM, Delta Tau Accessory #xx

INDEX

T

V

W