

Title	Delta Tau / PEWIN Primer
Author	Benjamin Hornberger
Date	2009-04-09
Revision	A
Applicable to	All systems using Delta Tau UMAC / Turbo PMAC2 Motion Controllers

1 Introduction

This document describes basic usage of the PEWIN32Pro (a.k.a. PEWIN) program for diagnostics, maintenance and troubleshooting of the Delta Tau PMAC motion controller. For further information, consult the Delta Tau documentation (in particular the Software Reference Manual and the Turbo PMAC User Manual).

Note that here we do not describe any particular Delta Tau configuration or system-specific aspects. Refer to the documentation of your system or contact Xradia for details.

2 Starting the program

There is a shortcut to the program in the PEWIN32Pro submenu of the Windows XP start menu (see Figure 1).

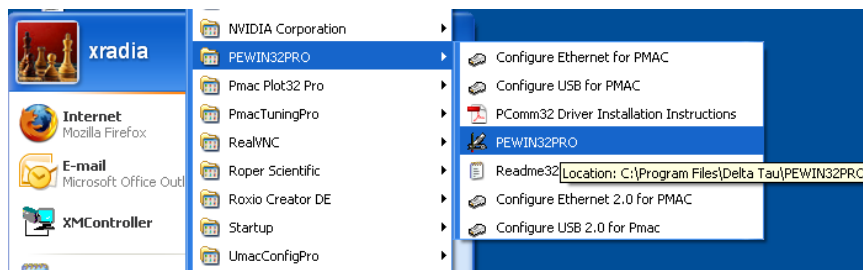


Figure 1: Starting PEWIN32Pro.

3 Program overview

Figure 2 shows a screenshot of a typical PEWIN session. Depending on the task at hand, different subwindows can be opened. The relevant ones will be described below in detail. Note that several instances of a type of subwindow can be opened, for example to monitor the status of two motors at the same time.

A single instance of PEWIN can communicate with several Delta Tau controllers that might be connected to the PC on which the program is running. In this case, each subwindow is tied to one controller, which is listed in the title of the subwindow. Right-clicking onto the title bar of the window allows switching the PMAC.

For some types of windows (e.g. watch window, position window) you can change the font size from the context menu (right-click). This is useful if you want to observe the values from a distance, while working on the instrument.

A note on PEWIN terminology: The words Upload / Download are used always from the point of view of the controller, i.e. *upload* means from the controller to the PC (where the PEWIN program runs), and *download* means from the PC to the controller.

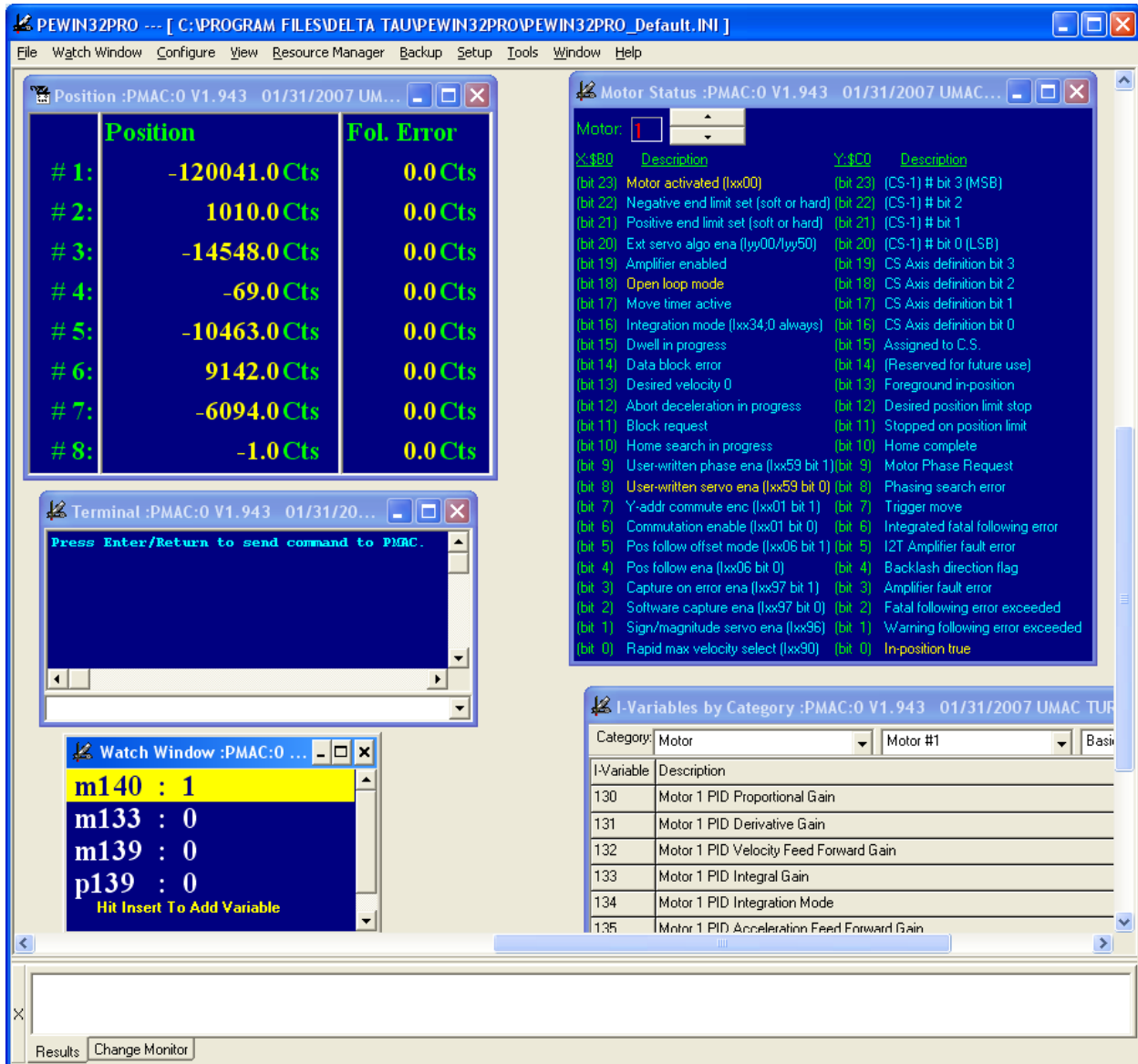


Figure 2: PEWIN program window.

3.1 The terminal

The most important interface to the controller is the terminal (see Figure 3). If there is no terminal on the screen, or you want to open an additional one, go to View → Terminal.

Here you can type commands that are sent to the controller. Above the command line, previous commands are displayed along with the response from the controller.

Note that the command line has a history feature that exhibits some awkward behavior. You can try using the up and down arrows (similar to a Unix shell) to recall previously-used commands, but you won't always find the desired one. Typically it helps to type the first few letters of the desired command and then use the up/down arrows.

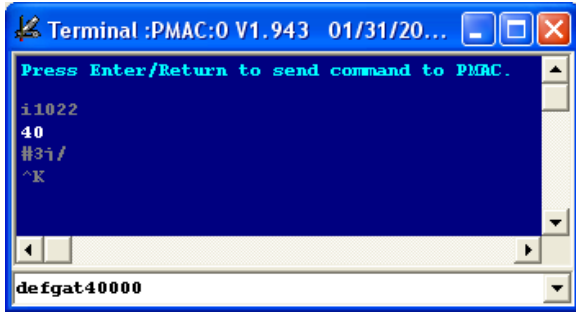


Figure 3: PEWIN terminal.

Note that several commands can be typed in one line, e.g. “#1j=2000 #2j=1000” (see below). The space between the commands can even be omitted.

3.2 Positions window

The positions window (see Figure 4) displays motor positions, typically in encoder counts (see the note about scale factors below). Go to View → Position to open a new window. In the standard configuration, it displays position of eight motors at a time. You can display a different set of eight motors by clicking into the window and pressing Page Up / Page Down, or by right-clicking and selecting “Next Set of Motors”.

	Position	Fol. Error
# 1:	-120041.0 Cts	0.0 Cts
# 2:	1010.0 Cts	0.0 Cts
# 3:	-14548.0 Cts	0.0 Cts
# 4:	-69.0 Cts	0.0 Cts
# 5:	-10463.0 Cts	0.0 Cts
# 6:	9142.0 Cts	0.0 Cts
# 7:	-6094.0 Cts	0.0 Cts
# 8:	-1.0 Cts	0.0 Cts

Figure 4: Positions window showing position and following error of motors 1-8.

The context menu (right-click into the window) offers the following useful options:

- Display the following error (actual minus commanded position).
- Display velocity.
- Select the next eight motors (“Next set of motors”).
- Display position of all 32 motors (excludes following error / velocity display)
- Change “Position and Velocity Units” lets you apply a scale factor (counts per unit). This is useful if you want to display real units (e.g. microns) instead of encoder counts. Note that this has to be done for each motor individually.

3.3 Motor status window

The motor status window displays status bits for a motor. Go to View → Motor Status to open a new window. The motor can be selected with the up/down arrows or by pressing Page Up / Page Down. Active (high) bits are highlighted in yellow. Relevant bits are (mostly self-explanatory):

- Negative / positive limit set. Note that this doesn't distinguish between soft and hard limits.
- Amplifier enabled (i.e. motor is in closed loop)
- Stopped on position limit. This means the previous motion was stopped on a limit, even though the limit switch might not be engaged any more (see “negative / positive limit set” above). Sometimes the limit switches are not active any more if the stage runs a little bit beyond during deceleration.
- Fatal / Warning following error exceeded
- In-position true

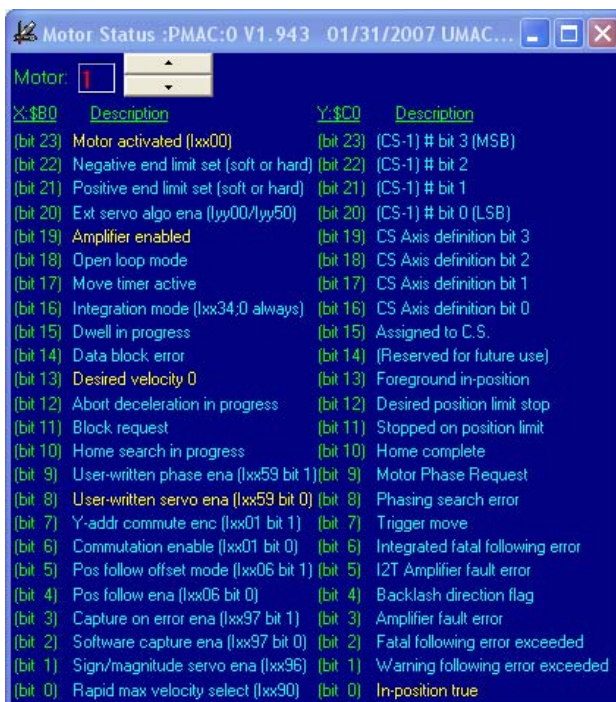


Figure 5: Motor status window.

3.4 Watch window

The watch window allows real-time monitoring of variables (see Sec. 5 below). Go to View → Watch Window to open a new watch window. Click into the window and then press the Insert key on the keyboard to add single variables, or multiple variables (i.e. a range of variables).

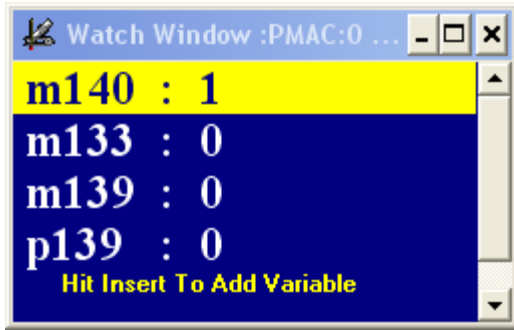


Figure 6: Watch window.

By right-clicking and selecting “Thread priority” you can increase / decrease the update speed. Note that for each update cycle PEWIN needs to query the variable value from the controller, so that many variables / watch windows and / or a fast update cycle can put a considerable load onto the controller and the communication bus (USB / Ethernet). This, by the way, applies to an open PEWIN instance in general – it is recommended to close PEWIN when it is not in use.

The highlighted variable can be removed from the watch window using the Delete key.

4 Moving motors

One of the most frequent jobs done in PEWIN is moving motors for diagnostics or other purposes. Here we only describe “jogging” of motors (not motion programs or other more sophisticated means of moving). Note that jogging is always in encoder counts, not “real” units (microns etc.).

Relevant commands to be used in the terminal are (omit the colon when typing the command):

- `#nj/`: Enable motor number n (enable amplifier and put in closed loop)
- `#nj=x`: Move motor number n to position x (absolute position in counts)
- `#nj:x`: Move motor number n by x counts from the current commanded position (can be positive or negative)
- `#nk`: Kill (stop and disable) motor number n
- `^k` (i.e. Ctrl-k): Kill (stop and disable) all motors. **This is the most important emergency command!**

Note that `#n` is actually a separate command “make number n the current motor”. Once that is done, all subsequent commands apply to that motor, so it doesn’t have to be typed each time. However, note the following warning:

If XMController is running at the same time, it will constantly poll motor positions and therefore change the current motor. That means, while XMController is running you *must* type `#n` for every single command, otherwise the behavior is unpredictable!

5 Variables

The following types of variables exist on the Delta Tau controller: I-, P-, Q- and M- variables. For each type, there are 8192 variables, i.e. I0 ... I8191, P0 ... P8191, Q0 ... Q8191 and M0 ... M8191.

In the following we give an overview of the variables. For specific meaning of individual variables, consult the Turbo PMAC Software Reference Manual.

5.1 I-variables

I-variables are configuration variables for the Delta Tau controller, each with a predefined meaning. The I-variables that you are most likely to encounter are the motor setup variables $I_{xx}00$ through $I_{xx}99$, where $xx = 1 \dots 32$ is the motor number. They include the motor motion parameters (velocity, acceleration, ...), tuning (PID, feed forward gains etc.), limits and more.

Another set of relevant I-variables are the channel-specific Servo-IC setup variables in the I7000 range that define the hardware configuration of a motor channel. One Servo-IC (motor driver card) typically serves four motors, so the Servo-IC setup variables are typically denoted $I7mnx$, where $m = 2 \dots 9$ is the servo IC number, $n = 1 \dots 4$ is the channel on that IC, and the different x 's are different setup variables for that channel. Note that m runs from 2 to 9 for the motor driver cards because ICs 0 and 1 are on board the processor card. The mn for all 32 motors are listed in the table below for reference.

Servo IC	Channel	$I7mnx$	Motor #
2	1	$I721x$	1
	2	$I722x$	2
	3	$I723x$	3
	4	$I724x$	4
3	1	$I731x$	5
	2	$I732x$	6
	3	$I733x$	7
	4	$I734x$	8
4	1	$I741x$	9
	2	$I742x$	10
	3	$I743x$	11
	4	$I744x$	12
5	1	$I751x$	13
	2	$I752x$	14
	3	$I753x$	15
	4	$I754x$	16
6	1	$I761x$	17
	2	$I762x$	18
	3	$I763x$	19
	4	$I764x$	20
7	1	$I771x$	21
	2	$I772x$	22
	3	$I773x$	23
	4	$I774x$	24
8	1	$I781x$	25
	2	$I782x$	26
	3	$I783x$	27
	4	$I784x$	28
9	1	$I791x$	29
	2	$I792x$	30
	3	$I793x$	31
	4	$I794x$	32

5.2 P-variables

P-variables are for general use, i.e. they have no predefined meaning and are not used by the Delta Tau controller for its own purposes. The user is free to use them for his own purposes, for examples as status

variables / flags / switches, for calculations etc. P-variables are often used in PLC programs (see Sec. 6). If necessary, consult your system documentation or Xradia directly for the P-variables used in your system.

5.3 Q-variables

Q-variables are currently not used in Xradia systems, so they are not covered here.

5.4 M-variables

M-variables give direct access to PMAC memory registers. Typically M-variables are assigned a memory address so that the value of that register can be retrieved or modified easily. The Delta Tau controller usually doesn't require specific M-variable assignments for its operation (it accesses memory directly), but it is strongly recommended to make use of the *suggested* M-variable definitions (see Software Reference Manual) when reading memory registers in PLCs etc. These suggested M-variable definitions assign variables to most relevant motor parameters (like actual and commanded position, limit status, in-position bit etc.) in the Mxx01 to Mxx99 range for motor number $xx = 1$ to 32.

To query the value of an M-variable in the terminal, type Mxxx, where xxx is the variable number. To query the memory address that an M-variable is assigned to, type Mxxx->.

If an M-variable is assigned to *, that means it is not assigned to a specific memory register but “self-assigned”. In this case, the variable can still be used for the user's purpose, although there is no particular advantage over using a P-variable for the same purpose (it is even slower).

5.5 Querying and assigning variable values

You can query the value of any variable in the terminal by typing the variable (e.g., I122). You can assign values using the equal sign, e.g. I122=50.

For each category of variables, there is also a window available in the Configure menu to display and edit the variable values (see Figure 7 for the M-variable version). Two general warnings:

- The values will not refresh automatically when variables are changed outside of the window, you need to click the Refresh button. The exception is the M-variable window when the “Update Periodically” box is checked.
- Remember to press Enter after editing a value, or clicking outside the field. Otherwise the value will not be changed on the controller.

For P- and Q-variables, you can simply read and set the current values in the configure window.

For M-variables you can also see the definition (memory address), and when checking the “Use Suggested M-var. Description” box, that description will also be shown (be aware though that this doesn't guarantee the variable is actually assigned to the appropriate register – that needs to be verified separately!).

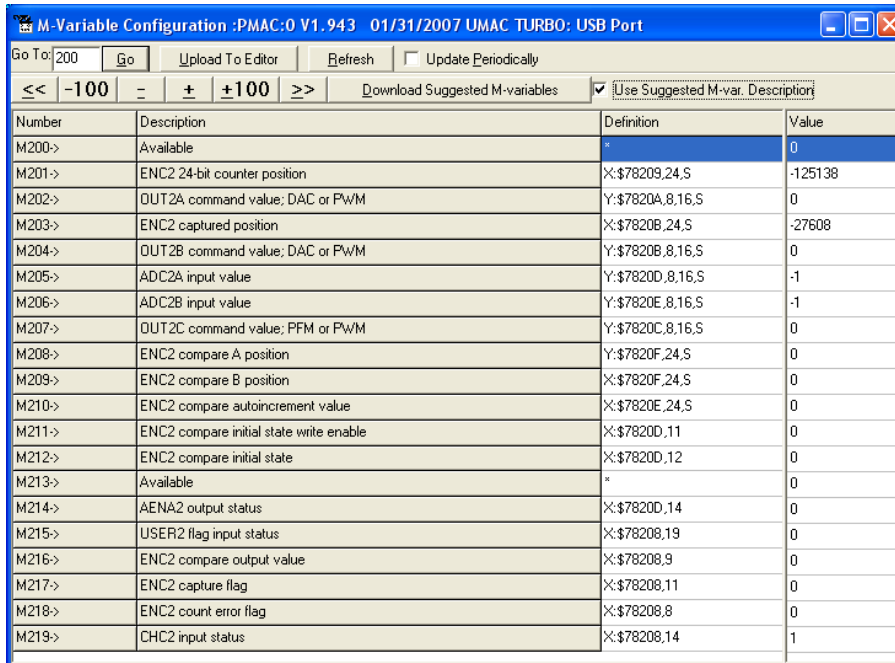


Figure 7: Configure M-variables dialog.

For I-variables, there are separate windows available to view them by number or by category. Both options display the standard I-variable description. The category option (see Figure 8) is particularly useful to compare the configuration for different motors: The subcategory, e.g. “Movement”, can stay selected while going from one motor to another.

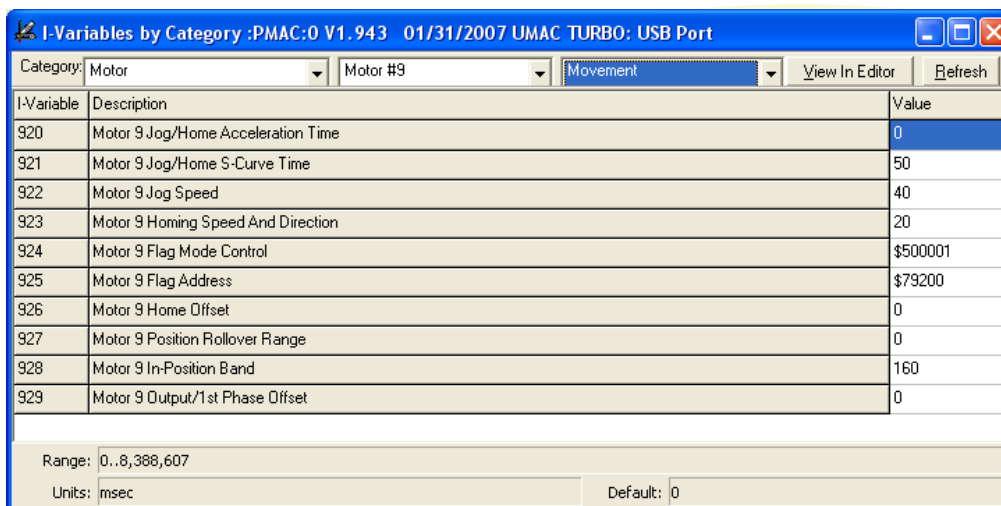


Figure 8: Configure I-variables by category dialog.

5.6 Uploading variables from the controller to the PC

To save a range of variables to a text file, go to File → Upload variables. Select the variable type and the range. Note that for M-variables you can either select the values or the definitions (memory addresses). After clicking Upload, the corresponding text file will open in the PEWIN editor and can be saved.

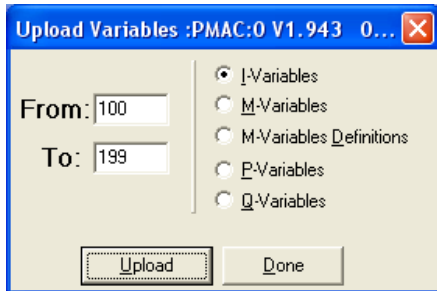


Figure 9: Upload variables dialog.

5.7 Downloading variables from the PC to the controller

If you have a set of variable assignments in a text file that you want to apply on the controller, you can open the file in the PEWIN editor and click the yellow Download button as described in Sec. 6.3.

6 PLC programs

PLC programs (called PLCs because they allow functionality similar to a Programmable Logic Controller) are programs that can be executed on the Delta Tau controller. A typical application would be the monitoring of an input and executing certain commands depending on the status of the input. Xradia uses PLCs to implement homing routines.

We will not discuss PLC programming and syntax here, but general rules how to check PLC status, how to enable them etc.

There can be up to 32 PLCs on the Delta Tau controller, PLC0 through PLC31. PLC0 is a special PLC that is executed at specific times (every m servo cycles, where m can be set through an I-variable). All other PLCs are executed as time allows.

6.1 Enabling / disabling PLCs

A PLC is enabled with the command “enable plc n ”, or “enaplc n ” for short, where n is the PLC number. A PLC is disabled with the command “disable plc n ”, or “displcn” for short. Normally PLCs, once enabled, run indefinitely until disabled manually. However, a PLC can also include a disable command at the end so that it runs only once when enabled.

The command $\wedge d$ (i.e. Ctrl-d) disables all PLCs. **This can be important as an emergency command, but then think whether there are any important PLCs that you want to enable again!**

6.2 Checking PLC status

To check if a PLC is enabled, go to View \rightarrow Program/PLC Status (and Upload) (see Figure 10). The table lists (in the Number column) *all* PLCs that are defined on the controller, and in the Active column tells which ones are currently enabled.

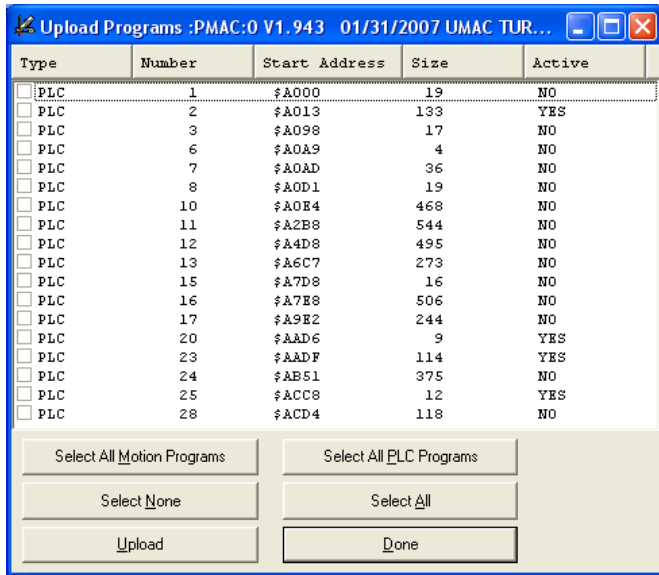


Figure 10: PLC status window.

6.3 Downloading a PLC from a text file

Sometimes it is necessary to load a new PLC to the controller. If you have the PLC in a text file on the PC, you can open it in the PEWIN editor (see Figure 11) by going to File → Open. Note that the txt extension is not selected by default, so you might not see the file at first.

First, if you want to overwrite an existing PLC, make sure it is not enabled currently (see above). Then download it to the controller by clicking the yellow down arrow in the editor toolbar. Check the “Results” output field at the bottom of the PEWIN main window for errors. If there are no errors, you can enable the PLC.

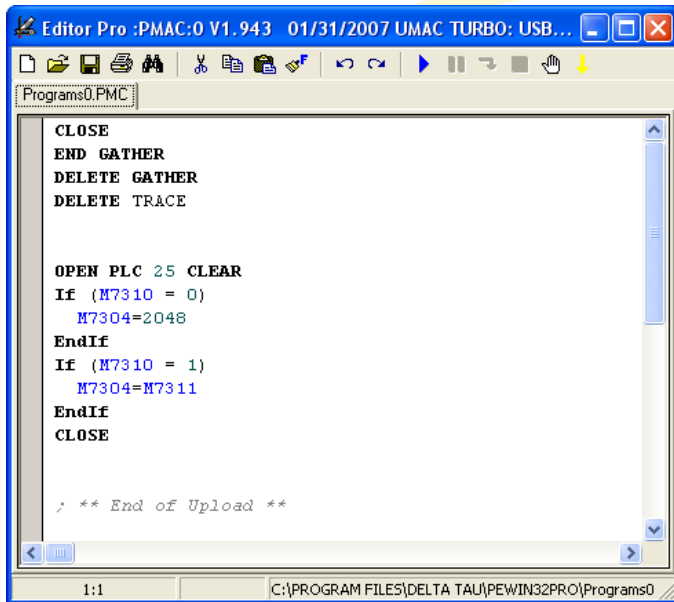


Figure 11: PEWIN Editor with a PLC opened.

7 Maintenance and troubleshooting

7.1 Saving the current configuration to EPROM

After making a configuration change (changing variables, uploading a new PLC, etc.) that you want to be permanent, you need to save the configuration to EPROM to preserve it across the next power cycle. This is done with the “sav” command.

7.2 Resetting the Delta Tau controller

The command “\$\$\$” resets the controller to the last saved configuration (see Sec. 7.1). This can be done to undo any changes made since the last save, or sometimes it helps to fix problems if the controller behaves strangely (like restarting a computer).

7.2.1 Preserving motor positions

Note that after resetting, all motor positions will be zero, i.e. the motors need to be homed. Alternatively all current motor positions can be stored before the reset by recording an image in XMController. After the reset, XMController can assign the positions stored in that image to the motors (Microscope → “Set motor positions based on image” in XMController). It is recommended to turn off all motor amplifiers before storing the positions, and only turn them back on after re-assigning the positions. This will minimize any undesired motion that is not captured by that method.

7.3 Power cycling the Delta Tau controller

In rare cases it can happen that the controller freezes, in which case power cycling is the only option to reset it. There is no power switch, so you need to unplug it. It is recommended to close PEWIN before doing that, otherwise there might be communication problems after restart (see Sec. 7.5).

After power cycling, the last saved configuration will be restored (see Sec. 7.1). All the motor positions will be zero, so that Sec. 7.2.1 applies here as well.

7.4 Backing up the Delta Tau configuration

The entire current Delta Tau configuration can be saved to a text file by going to Backup → Upload configuration (see Figure 12). This is recommended before any major configuration changes. Check the boxes according to what is important to you; the sample shown in the figure should be appropriate in most cases. Depending on how many boxes are checked, this can take several minutes to complete.

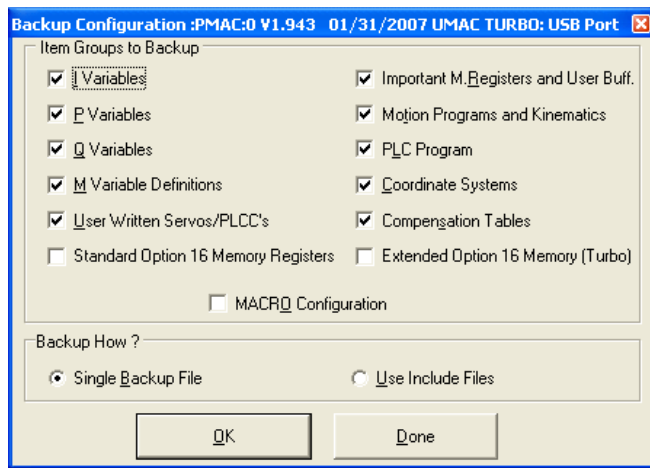


Figure 12: Upload configuration dialog.

7.5 PEWIN cannot communicate with the controller

Sometimes PEWIN cannot communicate with the controller. This can happen in particular when the controller was powered off while PEWIN was running.

- Go to Setup → Force All Windows to Device Number. The device selection dialog opens (see Figure 13).
- If there is a PMAC listed, you can click “Test”. (If not, continue with “Click insert ... below).
- If it says “The PMAC was successfully detected”, it should work.
- If it says “The PMAC could not be detected”, then click Remove and OK, then close PEWIN. If it doesn't let you remove the PMAC, close PEWIN anyway. Open the Windows Task Manager (Ctrl-Alt-Del) and check if there is a process “PMACSE~1.exe” running. If yes, end that process.
- Restart PEWIN and go to Setup → Force All Windows ...
- Click insert. You should see the PMAC. Select it and click OK.
- Click Test. It should now say “The PMAC was successfully detected”.

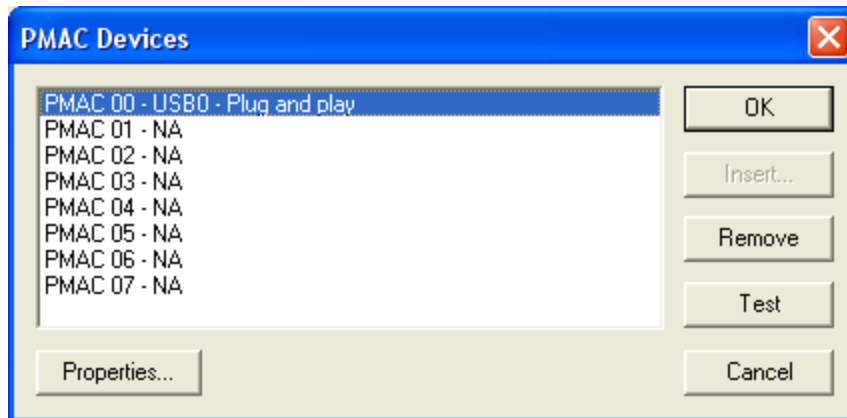


Figure 13: PMAC Devices dialog.