SOFTWARE REFERENCE MANUAL

Turbo PMAC / PMAC2

V1.941 – V1.945 Firmware Addendum

3Ax-01.942-xSxx

August 28, 2008

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

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 exposed to hazardous or conductive materials and/or environments, we cannot guarantee their operation.

REVISION HISTORY				
REV.	**DESCRIPTION**	**DATE**	**CHG**	**APPVD**
1	ADDED V 1.943 FIRMWARE ADDENDUM	08/15/07	CP	C. WILSON
2	ADDED V1.944 & V1.945 FIRMWARE ADDENDA	08/28/08	CP	C. WILSON

Table of Contents

USER MANUAL ADDITION

Coordinate System "Time Base" and "Feedrate Override"

Turbo PMAC provides two methods for the user to vary the execution speeds and times of commanded trajectories from those set by I-variables and/or motion-program parameters. There are several reasons to use these features. One of the most common is to provide the machine operator with an interactive "feedrate override" capability, permitting him to use a potentiometer or rotary switch to set a percentage override value in real time compared to the programmed speed. Another use is to synchronize move execution to the motion of a master encoder for functionality such as electronic cams.

Both methods work by varying the numerical time-update value in interpolation calculations from the (constant) actual physical time elapsed. The first method, time-base control, operates at the servo-rate interpolation stage, after all other trajectory calculations. It is the method that must be used to synchronize move execution to the travel of a master encoder. However, because it operates after all other trajectory calculations, it affects other trajectory features such as acceleration (acceleration varies inversely with the square of the time-base override value). For this reason, some users do not feel that it is suitable as an interactive feedrate override control.

The second method, segmentation override, operates at the coarse interpolation, or segmentation stage. Because it acts before the acceleration control of the special lookahead buffer, the segmentation override can vary velocity without affecting acceleration. This makes it optimal for interactive feedrate override control. It should be noted that this override control operates only on Turbo PMAC move modes that are segmented: LINEAR and CIRCLE-mode moves with segmentation active (Isx13 > 0). This method was introduced in Turbo PMAC firmware revision 1.941, released in 2005.

Coordinate System "Time-Base"

Each coordinate system has its own "time base" that helps control the speed of interpolated moves in that coordinate system. Turbo PMAC's interpolation routines increment an "elapsed-time" register every servo cycle. While the true time for the servo cycle is set in hardware for the entire system (by jumpers E98, E29-E33, and E3-E6 on a Turbo PMAC1, by hardware-control variables I7m00, I7m01, and I7m02 for a Turbo PMAC2 without MACRO, or by I6800, I6801, and I6802 for a Turbo PMAC2 with MACRO) and does not change for a given application, the value of time added to the "elapsed-time" register for a coordinate system each servo cycle is just a number in a memory register. *It does not have to match the true physical time for the cycle.*

The units for the time base register are such that 2^{23} (8,388,608) equals 1 millisecond. The default value for the time-base register is equal to the value of I10. The factory default value for I10 of 3,713,707 represents the default physical servo cycle time of 442 microseconds.

If the value of the time base register is changed from I10, interpolated moves will move at a different speed from that programmed. Many people call this capability "feedrate override". Note that the physical time does not change, so servo loop dynamics remain unchanged.

Each coordinate system has a variable Isx93 that contains the address of the register that the coordinate system uses for its time base. With the default value of Isx93, the coordinate system gets its time base information from a register that is set by `%` commands from the host computer. A `%100` command puts a value equal to I10 in this register; a `%50` command puts a value equal to I10/2 in this register.

Note: A motor that is not assigned to any coordinate system uses Coordinate System 1's time base value for executing its jogging and homing moves. If you do not want this to be the case, you must assign the motor to a different coordinate system, even if you do not need to write a motion program to run this motor.

Regardless of the source of the time base information, a `%` query command cause PMAC to report back the value of the present time base expressed as a percentage of I10.

Time base information can come from other sources. The most common alternative to command-sourced time base is external frequency-sourced time base, where the time base value is proportional to the frequency of a master encoder. This provides a powerful position-synchronized slaving mechanism that is commonly called "electronic cam".

See instructions for using an external time base, in the section *Synchronizing Turbo PMAC to External Events*.

Coordinate System Segmentation Override

For LINEAR and CIRCLE mode moves, Turbo PMAC performs a two-stage interpolation process. (This is required for the execution of CIRCLE-mode moves; it is optional if only LINEAR-mode moves are executed, and without cutter-radius compensation.) At the first, coarse, interpolation stage, called segmentation, the complex calculations such as the circle trigonometric calculations, inverse-kinematic transformations, and lookahead acceleration control calculations are performed. Then a simple fine-interpolation algorithm (cubic B-spline) is performed at the servo-cycle rate to generate the instantaneous commanded positions for each servo cycle.

The segmentation period for segmented moves is set by coordinate system I-variable Isx13, scaled in milliseconds. That is, every Isx13 milliseconds (without override), Turbo PMAC will perform coarse interpolation calculations to compute an intermediate segment point for each axis in the move, advancing Isx13 milliseconds in the move equations.

With override control at the segmentation stage, the numerical value of the time update value does not have to match the physical time elapsed per segment. For example, if Isx13 were set to 10 to define a physical segment time of 10 milliseconds, but a value of 5 milliseconds were used for the time-update calculations in the segmentation algorithms, an override of 50% would be produced. Because this override function is performed before the acceleration control function of the special lookahead buffer, the combination of segmentation override and lookahead acceleration control yields an override capability that controls velocity but maintains acceleration.

Segmentation override control occurs after acceleration profiles commanded by TA and TS (Isx87 and Isx88) times are created in the generation of the basic move equations. Therefore these acceleration times vary inversely with the override (and the rates they produce vary inversely with the square of the override). However, in applications utilizing segmentation override and lookahead acceleration control, these times are usually set very small (because the lookahead is used to control rates of acceleration), and used mainly to define the size of corner blends in the path. Significantly, the operation of the segmentation override and lookahead acceleration control functions do not change the size or shape of these corner blends.

I-variable Isx15 controls the segmentation override for the coordinate system. Isx15 has a range of -1.0 to +1.0 representing override values of 0% to 200%. With Isx15 at the default value of 0.0, the override value is 100% (real time). I-variable Isx16 controls the slew rate for changes in the working value of Isx15, expressed in units of Isx15 per segment. This permits the user to command large step changes in Isx15, but still create smooth changes in the actual override. Global I-variable I12 should be set to 1 to enable “time spline” calculations in the lookahead algorithm to ensure smoothness in transitions, and Isx16 should be set small enough so that 2-to-1 velocity changes in a single segment cannot be commanded.

Note that is it possible, but not advisable to use segmentation override and the override of time-base control simultaneously. For example, if Isx15 were set to –0.5 to achieve a segmentation override of 50%, and time-base control set a 60% override at the fine-interpolation stage, the net override would be 30% of programmed velocity.

Blending and Cornering Control

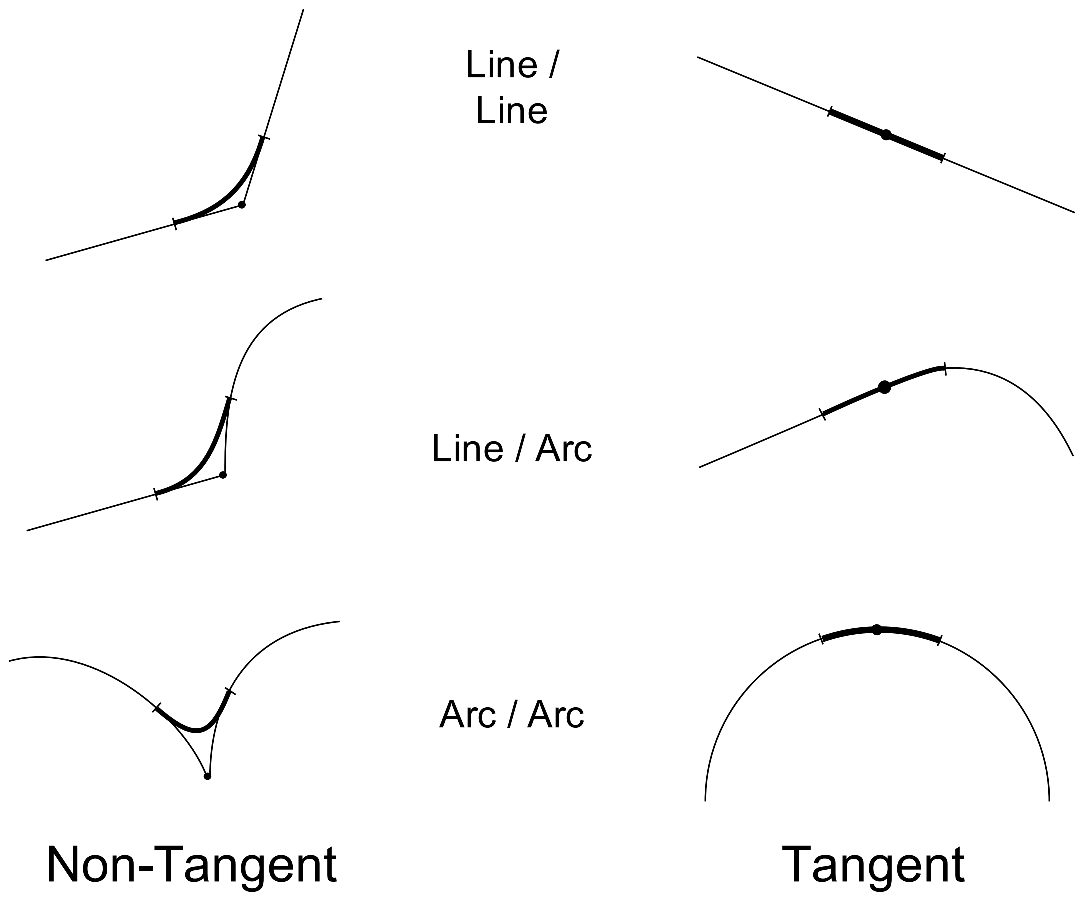
Turbo PMAC has sophisticated features to govern how and when LINEAR and CIRCLE-mode moves are blended together, particularly in path-based applications. These features permit you to govern precisely but easily the nature, size, and shape of corners formed by consecutive moves.

Normal Blending

By default, consecutive LINEAR and CIRCLE-mode moves are blended together without stopping. The blending occurs over the acceleration time Ta, which is the larger of the declared TA (Isx87) or 2 times TS (2*Isx88). The blending occurs half in the incoming move and half in the outgoing move. A rounded path corner starts a distance F*Ta/2 along the path before where the sharp corner would be if blending were disabled, and ends an equivalent distance along the path on the outgoing move. In other words, the blend starts where the deceleration to stop of the first move would begin without blending, and ends where the acceleration from stop of the second move would end.

The following diagram shows the paths generated by the blending function for LINEAR and CIRCLE mode moves, distinguishing between “tangent” and “non-tangent” blends. The paths are identical in either direction.

Blending Between Lines and Arcs



Disabling All Blending

Setting Isx92 to 1 disables all blending. Each move comes to a complete stop before the next move is calculated and executed. This is the equivalent of adding a DWELL 0 to the program between each move. Turbo PMAC only checks Isx92 at the beginning of execution of a motion program. If you desire to change this in the middle of a motion program, you must change the "continuous motion request" coordinate system status bit (bit 4 of the first coordinate system status word) directly. The suggested M-variable for this bit is Msx84, and it is set to 0 to disable blending.

Disabling Blending Based on Corner Angle

It is also possible to disable blending selectively based on the angle of the corner formed by the incoming and outgoing moves. Isx83 sets the threshold of the corner angle between those that are blended and those that are not. It is expressed as the cosine of the maximum change in directed angle at the corner that will be blended. At sharper corners (larger change in directed angle, smaller cosine), the moves will not be blended together. At shallower corners (smaller change in directed angle, larger cosine), the moves will be blended together. If Isx83 were set to 0.707, a 45^o change in directed angle would be the threshold between blended and unblended corners.

The corner angle is evaluated in the plane in XYZ-space that has been specified by the most recent **NORMAL** command. The default plane is the XY-plane. If the corner has a component out of the specified plane, it is the projection of the corner into the specified plane that is evaluated.

If blending is disabled at a corner where an arc move has been automatically added by Turbo PMAC's cutter radius compensation algorithm (outside corner with change in directed angle having a cosine less than Isx99), the stop can either be at the beginning or end of this added arc. If Isx84 is at the default value of 0, the stop will be at the end of this arc. If Isx84 is set to 1, the stop will be at the beginning of this arc.

Additionally, it is possible to set a corner-angle threshold beyond which a dwell is automatically added before starting the outgoing move. This threshold is set by Isx85, again as the cosine of the change in directed angle of the threshold corner. Isx85 should be set to a value less than or equal to Isx83, because it is only evaluated if blending is disabled at a corner due to Isx83. If the algorithm determines that a dwell should be added, Isx82 specifies the time for that dwell, expressed in real-time interrupt periods.

Blend Disable Stop Condition

When blending is disabled between moves, it is possible just to bring the commanded trajectories to a momentary stop before starting the next move, ignoring what the actual positions are. It is also possible to wait for all motors in the coordinate system to be "in-position" as defined by motor variables Ixx28 and Ixx88 before starting the next move. This decision is controlled by Isx81 for the coordinate system. If Isx81 is at the default value of 0, only the commanded trajectories are brought to a stop. If Isx81 is set greater than 0, the program will wait for all motors in the coordinate system to come in-position first. In this case, the value of Isx81 set the "time-out" value; if all coordinate-system motors have not achieved in-position status within Isx81 real-time interrupt periods of the stop in the commanded trajectory, the program will stop with a run-time error.

New/Revised I-Variable Descriptions

I26 UMAC Electrical MACRO Enable {New}

Range:	0 – $F (0 – 15)
Units:	none
Default:	0

I26 controls whether the MACRO ICs on UMAC ACC-5E boards are configured for optical-fiber MACRO transceivers or RJ-45 electrical MACRO transceivers. I26 is a 4-bit value, with each bit corresponding to one of the four MACRO ICs whose addresses are specified by I20 – I23. If the bit is set to the default value of 0, that MACRO IC is configured for a fiber transceiver. If the bit is 1, that MACRO IC is configured for an RJ-45 electrical transceiver.

Bit "n" of I26 controls the configuration of the MACRO IC addressed by variable I2n, which is called MACRO IC "n" (e.g. bit 0 controls the IC of I20, MACRO IC 0). If fewer than 4 MACRO ICs are present, the settings of bits of I26 corresponding to "missing" MACRO ICs are not important.

I26 is used only on power-up/reset, so to change the effect of I26, the user must change the value, issue a **SAVE** command, and reset or cycle power on the UMAC.

Examples:

```
I26=$0          ; All MACRO ICs use fiber
I26=$F          ; All MACRO ICs use RJ-45
I26=$3          ; MACRO ICs 0 & 1 use RJ-45, MACRO ICs 2 & 3 use fiber
```

I27 Alternate TWS Input Format {New}

Range:	0 – 1
Units:	none
Default:	0

I27 controls how the Turbo PMAC interprets incoming data on a TWS-format M-variable read from an ACC-34 or similar serial-interface I/O board. If I27 is set to the default value of 0, PMAC expects the serial input data on the DAT0 signal line. If I27 is set to 1, PMAC expects the serial input data on the DAT7 signal line.

The DAT7 line is separated more from the output clock line on the same cable; the use of DAT7 by setting I27 to 1 and making the appropriate jumper setting on the I/O board makes it possible to use a longer cable without too much coupling interference between signals.

On the ACC-34AA, jumper E23 must be connect pins 1 and 2 to support the default setting of I68 = 0; it must connect pins 2 and 3 to support the setting of I27 = 1. On the ACC-76 and ACC-77 "P-Brain" boards, jumper E1 should be ON to support the default setting of I27 = 0; jumper E8 should be ON to support the setting of I27 = 1. Older boards of this class do not support settings of I27 = 1.

I36 Enable/Abort Separation Control {New}

Range: 0 – 1

Units: none

Default: 0

I36 controls whether the abort commands in Turbo PMAC can also enable disabled motors, or whether separate enable commands are required. If I36 is at the default value of 0, the abort commands (coordinate-system-specific **A** and global **<CTRL-A>**) will enable a disabled motor, bringing it to a closed-loop zero-velocity state. If I36 is set to 1, these commands will have no effect on a disabled motor. In this case, separate commands (coordinate-system-specific **E** and global **<CTRL-E>**) will be required to enable disabled motors.

Setting I36 to 1 permits simpler logic in implementing an emergency stop routine using the A or <CTRL-A> commands, since the routine will not have to worry about the disabled-motor case.

Note that if the motor is a synchronous (zero-slip – Ixx78 = 0) motor commutated by Turbo PMAC (Ixx01 bit 0 = 1), a phase referencing is required after power-up/reset before the motor can be enabled. This is done automatically on power-up/reset if Ixx80 for the motor is set to 1 or 3, or subsequently with the motor-specific **$** command, or the coordinate-system-specific **$$** command. These enabling commands do not cause a phase referencing to be performed.

The following table shows the action of each of these commands with I36 = 0:

Command	Starting State: Disabled	Starting State: Open-Loop Enabled	Starting State: Closed-Loop Enabled
A	Enable, close loop at present position	Close loop at present position	Decelerate to stop using Ixx15
<CTRL-A>	Enable, close loop at present position	Close loop at present position	Decelerate to stop using Ixx15
E	Enable, close loop at present position	Close loop at present position	(no action)
<CTRL-E>	Enable, close loop at present position	Close loop at present position	(no action)

The following table shows the action of each of these commands with I36 = 1:

Command	Starting State: Disabled	Starting State: Open-Loop Enabled	Starting State: Closed-Loop Enabled
A	(no action)	Close loop at present position	Decelerate to stop using Ixx15
<CTRL-A>	(no action)	Close loop at present position	Decelerate to stop using Ixx15
E	Enable, close loop at present position	Close loop at present position	(no action)
<CTRL-E>	Enable, close loop at present position	Close loop at present position	(no action)

I38 In-Line CALL Enable {New}

Range:	0 – 1
Units:	none
Default:	0

I38 controls the timing of the execution of a **CALL** command that follows motion commands on the same program line. Normally, as soon as the program counter jumps to a new program line, as it does on a subprogram call, program calculations are suspended until the start of execution of a new move.

If I38 is set to 1, the jump to the subprogram does not cause suspension of program calculation. This permits some further calculations to be completed immediately following the calculation of programmed motion. Setting I38 to 1 permits the proper use of a **CREAD** (coordinate read) command in a called subroutine to store the just-computed axis move endpoints with synchronous M-variable assignments, so that they are easily available when the move is actually being executed, facilitating calculation of data such as “distance to go”.

All calculations to be completed immediately following the calculation of programmed motion must be on the first line of the subroutine. A jump to a new line in the subroutine will suspend motion program calculations.

If I38 is set to the default value of 0, the jump to the subprogram causes suspension of program calculation until the start of execution of a new move.

I43 Auxiliary/Main Serial Port Parser Disable {Revised}

Range:	0 - 3
Units:	none
Default:	0

I43 controls whether Turbo PMAC firmware automatically parses the data received on the main serial port and/or the Option 9T auxiliary serial port as commands or not. I43 is a 2-bit value; bit 0 controls the auxiliary serial port, and bit 1 controls the main serial port. If the bit of I43 is set to the default value of 0, Turbo PMAC automatically tries to interpret the data received on the port as Turbo PMAC commands. However, if the bit of I43 is set to 1, it will not try to interpret this data as commands, permitting the user’s application software to interpret this data as required for the application.

With the parser for the port disabled, the **CMDA** (for the auxiliary serial port) or the **CMDS** (for the main serial port) statement for issuing commands as if they came from the port cannot be used. However, the **SENDS** or **SENDA** statement for sending messages out the port can still be used with the parser for the port disabled.

The following table shows the possible values of I43 and their effects:

I43	Main Serial Port Parser	Auxiliary Serial Port Parser
0	Enabled	Enabled
1	Enabled	Disabled
2	Disabled	Enabled
3	Disabled	Disabled

This ability to disable the automatic command parser permits the auxiliary port to be interfaced to other devices such as vision systems, which send out data on their serial ports, and not have the Turbo PMAC try to interpret this data as commands. The user can then interpret the input string in application software using M-variable pointers, typically with indirect addressing techniques.

Serial Port	Command Queue	Command Pointer	Response Queue	Response Pointer
Auxiliary	X:$001C00 - $001CFF	X:$000034	Y:$001C00 - $001CFF	Y:$000034
Main	X:$003600 - $0036FF	X:$FFFFE6	Y:$003600 - $0036FF	X:$FFFFE3

Only the low byte (bits 0 – 7) of each word in the command and response queues is used. The command pointer contains the location of the next character to be input to the Turbo PMAC. The response pointer does not need to be used if responses are issued using the **`SENDA`** or **`SENDS`** commands; only if responses are "manually" placed in the response queue. If assembling responses manually on the main serial port, the response must start at Y:$003600, the number of bytes to output must be placed in X:$FFFFE1, and bit 23 of X:$FFFFE0 must be set to 1 to enable the output.

I65 User Configuration Variable {New}

Range: 0 – 16,777,215

Units: none

Default: 0

I65 is a variable that has no automatic function in Turbo PMAC. Because its factory default value is 0, setting it to a non-zero value as part of the downloaded configuration file provides an easy way of later verifying that the configuration has been loaded in a particular card.

Since this variable has no automatic function, how this variable is utilized (if it is utilized at all) is completely up to the user. The same value may be downloaded to every controller, just for later verification of the presence of the download. Alternately, it may be used to identify different optional configurations, or as an electronic serial number.

I67 Modbus TCP Buffer Start Address {New}

Range: $0 – $03FFFF

Units: Turbo PMAC addresses

Default: 0

I67 enables the Modbus TCP interface in Turbo PMAC software and reports the starting address of the 256-word Modbus buffer in Turbo PMAC memory. To enable the Modbus TCP interface on the Turbo PMAC's Ethernet port, the following conditions must apply:

1. The Ethernet physical interface must be present
2. The Modbus TCP firmware for the Ethernet processor must be installed
3. V1.941 or newer Turbo PMAC firmware must be installed
4. A user buffer of 256 or more words must have been defined with the DEFINE UBUFFER command
5. I67 must be set to a value greater than 0.

The user can set I67 to any value greater than 0 to enable the Modbus TCP buffer. When this is done, PMAC will automatically set I67 to the address of the start of the 256-word Modbus buffer. In the standard Turbo PMAC CPU/memory configuration (Option 5C0), this address will be $010700, so the buffer will occupy the addresses $010700 - $0107FF.

A **`SAVE`** command must be issued with I67 at a non-zero value in order for the Modbus TCP buffer to be active after subsequent power-up or reset operations.

I69 Modbus TCP Software Control Panel Start Address {New}

Range:	$0 – $03FFFF
Units:	Turbo PMAC addresses
Default:	0

I69 enables and specifies the address of the start of the Modbus TCP software control panel in Turbo PMAC. I69 permits a software control panel to be commanded over the Modbus TCP link, typically from a PLC, using part of the user buffer created with the **`DEFINE UBUFFER`** command and reserved for Modbus TCP use with I67. If I69 is set to a value greater than 0, this software control panel is enabled. Typically, I69 is set to a value 128 ($80) greater than the value of I67, so this control panel starts at an address 128 higher than the beginning of the entire Modbus TCP buffer. For example, if the beginning of the Modbus buffer were at $010700, I69 could be set to $010780.

The software control panel occupies 26 long words of Turbo PMAC memory. The structure functions of the Modbus panel are equivalent to those for the DPRAM software control panel, which are documented in the Memory and I/O Map chapter of the Software Reference Manual at their default addresses of $060000 - $060019.

The operation of the Modbus control panel is independent of that for the DPRAM control panel (which is controlled by I2). One, neither, or both of these control panels may be active at one time.

Ixx96 Motor xx Command Output Mode Control {Revised}

Range:	0 - 3
Units:	none

Default: 0

Ixx96 controls aspects of how Turbo PMAC writes to the command output register(s) specified in Ixx02. Ixx96 is a two-bit value; bit 0 controls either the bipolar/unipolar nature of the output, whether the commutation algorithm is closed-loop or open-loop, or whether the current-loop algorithm is for a brushless or brush motor (dependent on the settings of Ixx01 and Ixx82); bit 1 controls whether the integrator of the PID position/velocity servo algorithm is in the position loop or the velocity loop.

If bit 0 of Ixx01 is set to 0 (no Turbo PMAC commutation for Motor xx), and bit 0 of Ixx96 is set to 0, the single command value from the Turbo PMAC servo is written to the register specified by Ixx02 as a signed (bipolar) value.

For PMAC(1)-style Servo ICs only, if bit 0 of Ixx01 is set to 0 and bit 0 of Ixx96 is set to 1, then the command output value is the absolute value (magnitude) of what the servo calculates, and the sign (direction) is output on the AENAn/DIRn line of the set of flags addressed by Ixx25 (polarity determined by jumper E17 or E17x). In this case, bit 16 of Ixx24 should also be set to 1 to *disable* the amplifier-enable function for that line. For PMAC2-style Servo ICs, this sign-and-magnitude mode is not supported.

If bit 0 of Ixx01 is set to 1 (Turbo PMAC commutation enabled for Motor xx), Ixx82 is set to 0 (Turbo PMAC current loop disabled for Motor xx), and bit 0 of Ixx96 is set to 0, Turbo PMAC will perform the normal closed-loop commutation for Motor xx. If bit 0 of Ixx01 is set to 1, Ixx82 is set to 0, and bit 0 of Ixx96 is set to 1, then Turbo PMAC's commutation performs the special "direct microstepping" algorithm. In this algorithm, the magnitude of the command from the servo does not affect the magnitude of the phase command outputs; it simply controls their frequency.

If bit 0 of Ixx01 is set to 1 (Turbo PMAC commutation enabled for Motor xx), Ixx82 is set to a value greater than 0 (Turbo PMAC current loop enabled for Motor xx), and bit 0 of Ixx96 is set to 0, Turbo PMAC will perform the normal direct-PWM control with both direct and quadrature current loops closed, for a 3-phase motor. If bit 0 of Ixx01 is set to 1, Ixx82 is set to a value greater than 0, and bit 0 of Ixx96 is set to 1, Turbo PMAC will perform direct-PWM control for a brush motor, truly closing only the quadrature current loop, and repeatedly zeroing the direct current-loop registers.

In non-Turbo PMACs, this function is controlled by bit 16 of Ix02.

If bit 1 of Ixx96 is at the default value of 0 (making Ixx96 equal to 0 or 1), the integrator of the PID position/velocity servo loop is in the position loop, acting on the position following error. This setting generally provides better tracking control, but worse disturbance rejection. If bit 1 of Ixx96 is 1 (making Ixx96 equal to 2 or 3), the integrator is inside the velocity loop, acting on the velocity error. This setting generally provides better disturbance rejection, but worse tracking control.

In revisions V1.940 and older of Turbo PMAC firmware, Ixx96 was a one-bit value, only including the function of bit 0.

Isx11 Coordinate System 'x' User Countdown Timer 1 {Revised}

Range:	-8,388,608 - 8,388,607
Units:	servo cycles
Default:	0

Isx11 provides an automatic countdown timer for user convenience. If Coordinate System 'x' is activated by I68, Isx11 will count down one unit per servo cycle. The user may write to this variable at any time, and it will count down from that value. Typically user software will then wait until the variable is less than another value, usually zero. The software accessing Isx11 does not have to be associated with Coordinate System 'x'.

Isx11 is a signed 24-bit variable, providing a range of -2^{23} (-8,388,608) to $+2^{23}-1$ (8,388,607). If active, it counts down continually until it reaches its maximum negative value of -8,388,608. It will not roll over. Most people will just use the positive range, writing a number representing the number of servo cycles for the period to the variable, then waiting for it to count down past 0.

If Isx14 is set to a non-zero value when the **MOVETIME** command is issued to this coordinate system, Isx11 will automatically be written to with a value of the number of servo cycles equal to the time left in the commanded move minus Isx14 milliseconds. This lets the user easily monitor Isx11 to find out when the move is Isx14 milliseconds from the end.

The following code shows how Isx11 could be used in a PLC to turn on an output for a fixed period of time.

```
M1=1              ; Set the output
I5111=2259        ; Set the timer to 1 second (2259 servo cycles)
WHILE (I5111>0)   ; Wait for timer to count down
ENDWHILE
M1=0              ; Clear the output
```

Isx14 Coordinate System 'x' End-of-Move Anticipation Time {New}

Range:	(floating-point)
Units:	milliseconds
Default:	0.0 (disabled)

Isx14, if set to a non-zero value, defines the time in relative to the end of a commanded move at which a dedicated timer will reach zero. This can be used to trigger events in anticipation of the end of the move, as in the case of firing a punch. If Isx14 is negative, the timer will reach zero before the end of the commanded move; if Isx14 is positive, the timer will reach zero after the end of the commanded move.

If Isx14 is set to a non-zero value when the on-line command **MOVETIME** is issued to this coordinate system, when Turbo PMAC calculates the time left in the commanded move (in milliseconds at 100%) for response, it will then convert this to true time using the present %

override value in force at this time, add Isx14 to this "true-time" value, convert the sum to servo cycles using the % value, and place this value in the Isx11 countdown timer for the coordinate system. This timer will then automatically decrement each subsequent servo cycle, and therefore reach a value of 0 at a time Isx14 milliseconds before the end of the commanded move (presuming the % value does not change in the meantime).

Note that the **MOVETIME** command can be issued from a PLC program, and if it is issued through a **CMD""** statement rather than a **CMDx""** statement, no actual response string is sent to any port.

If Isx14 is set to 0.0, these additional calculations will not be performed when the **MOVETIME** command is issued.

It is the user's responsibility to avoid any conflicts with other possible uses of the Isx11 timer.

The following PLC code shows how this feature might be used to trigger an action 20 milliseconds before the end of a commanded move.

```
I5114=-20                  ; 20msec before end
I5111=8000000              ; Set to large value
CMD"&1MOVETIME"            ; Trigger calculation action
WHILE (I5111>0)            ; Wait for timer to count down
ENDWHILE
M1=1                       ; Trigger action
```

Isx15 Coordinate System 'x' Segmentation Override {New}

Range:	-1.0 – 0.9999999
Units:	0 – 200%
Default:	0.0

Isx15 permits a "feedrate override" of segmented moves (LINEAR and CIRCLE mode moves with Isx13 > 0) in Coordinate System 'x'. The override percentage can be expressed as:

$$Override(\%) = (Isx15 + 1.0) * 100\%$$

At the default value for Isx15 of 0.0, the override value is 100%, so trajectories are calculated at their programmed speed. At the minimum value for Isx15 of -1.0, the override value is 0%, and the trajectory is "frozen" so no motion of the segmented move will occur. At the maximum value for Isx15 of 0.9999999, the override value is essentially 200%, so trajectories will be calculated at twice their programmed speed.

To calculate the value of Isx15 required for a given override value, the following equation can be used:

$$Isx15 = [Override(\%) / 100\%] - 1.0$$

The effect of override control with Isx15 is similar to that of override control with Turbo PMAC's time-base control (%) feature, but there are several important differences.

First, because the segmentation override control occurs before the lookahead buffer, the acceleration control of the lookahead buffer is independent of the segmentation override value. If the programmed acceleration times are kept small, acceleration rates are essentially unaffected by the segmentation override value. By contrast, the time-base control occurs after the lookahead acceleration control, so override control with the time-base feature will always affect acceleration as well as velocity.

Second, because the trajectories calculated with the segmentation override are (typically) passed through the lookahead buffer, changes in the segmentation override value are delayed in execution by the length of the lookahead buffer. Changes in override through time-base control are instantaneous.

Third, time-base control can be used to provide complete synchronization to a master encoder, but segmentation override cannot.

The user may write to Isx15 at any time, and it will immediately affect segmentation calculations. As noted above, the effect on actual motion is delayed by the length of the lookahead buffer. The rate of change in the active override value is limited by the Isx16 override slew rate parameter. Global variable I12 should be set to 1 to enable “time splining” in lookahead for smooth transitions, and whatever algorithm is setting Isx15 should ensure that no instantaneous change in resulting velocity of greater than 2-to-1 magnitude results.

If the user attempts to write a value greater than 0.9999999 to Isx15, it will saturate at 0.9999999. If the user attempts to write a value less than -1.0 to Isx15, it will saturate at -1.0.

The value of Isx15 at power-up/reset is always 0.0, representing 100% override. The user cannot save the present value of Isx15 to non-volatile flash memory.

Isx16 Coordinate System ‘x’ Segmentation Override Slew {New}

Range:	0 – 0.9999999
Units:	Change in Isx15 per segment
Default:	0.0

Isx16 controls the rate of change of the segmentation override value commanded by Isx15 for Coordinate System ‘x’. When the user changes the Isx15 command value, Turbo PMAC will change the actual override value by the magnitude of Isx16 each motion segment until the new value of Isx15 is reached.

When used with the special lookahead buffer active, the user should set Isx15 so that there cannot be an instantaneous change in segment velocity of greater than 2-to-1 magnitude (of non-zero velocities) between adjacent segments, before lookahead. This will permit the “time-spline” algorithm in lookahead enabled with I12 = 1 to generate smooth acceleration trajectories.

If Isx16 is set to 0.0 when Isx15 is changed, there is no slew rate control in the segmentation override, and override will immediately change to the new value of Isx15 in the next segment calculated.

Example:

For a change in override of 100% (e.g. Isx15 changing from 0.0 to -1.0) in 1.0 seconds with a 5 millisecond segmentation time (Isx13 = 5), we would want the effective value of the override parameter to change by 0.005 each segment, so Isx16 should be set to 0.005.

Isx78 Coordinate System ‘x’ Maximum Circle Acceleration {New}

Range:	Non-negative floating-point
Units:	(user position units) / (Isx90 feedrate time units)^2
Default:	0 (disabled)

Isx78, if set to a positive value, specifies the maximum centripetal acceleration that Turbo PMAC will permit for a feedrate-specified (F) CIRCLE-mode move in Coordinate System ‘x’. If the move as programmed would yield a higher centripetal acceleration, Turbo PMAC will automatically lower the programmed speed for the move so that the Isx78 limit is not exceeded. The centripetal acceleration is expressed as:

$$A = \frac{V^2}{R}$$

This limitation is done at the initial move calculation time, so it is not required to use the special lookahead buffer in conjunction with Isx78. It still may be desirable to use the special lookahead buffer, especially to manage the tangential acceleration into and out of a reduced-speed arc move.

The most important difference between limiting centripetal acceleration with Isx78 and limiting it with the individual motor Ixx17 acceleration limits has to do with the exact path generated. Turbo PMAC’s circular interpolation algorithms introduce a radial error term that can be described by:

$$E = \frac{V^2 T^2}{6R}$$

where V is the speed along the arc determined by the motion program (possibly limited by Isx78), T is the Isx13 segmentation time, and R is the radius of the arc. If Isx78 is used to limit the speed of the arc, this error will be reduced. However, if the special lookahead is used to limit the speed, the error will be as large as if the arc move were run at full speed.

Isx78 is expressed in the user length units for the linear axes (usually millimeters or inches) divided by the square of the user “feedrate time units” set by Isx90 for the coordinate system (usually seconds or minutes).

Example 1:

You want to limit the centripetal acceleration to 1.0g with Isx78. Your length units are millimeters, and your time units are seconds. Isx78 can be calculated as follows:

$$Isx78 = 1.0g * \frac{9.8\frac{m}{s^2}}{g} * \frac{1000mm}{m} = 9800\left(\frac{mm}{s^2}\right)$$

Example 2:

You want to limit your circular interpolation calculation errors to 0.001 inches. Your length units are inches, your time units are minutes, and your Isx13 segmentation time is 10 milliseconds. Isx78 can be calculated as follows:

$$Isx78 = \frac{V^2}{R} = \frac{6E}{T^2} = \frac{6 * 0.001in}{0.01^2 s^2} * \frac{60^2 s^2}{\min^2} = 216000\left(\frac{in}{\min^2}\right)$$

Example 3:

Your system is capable of 10 m/s^2 acceleration (about 1g). Your length units are millimeters, your time units are minutes, and your Isx13 segmentation time is 2 milliseconds. Isx78 can be calculated as follows:

$$Isx78 = 10\frac{m}{s^2} * \frac{1000mm}{m} * \frac{3600s^2}{\min^2} = 36{,}000{,}000\left(\frac{mm}{\min^2}\right)$$

At this setting, your maximum circular interpolation calculation errors can be computed as:

$$E = \frac{V^2T^2}{6R} = Isx78 * \frac{T^2}{6} = \frac{36{,}000{,}000}{6}\frac{mm}{\min^2} * 0.002^2 s^2 * \frac{\min^2}{60^2 s^2} = 0.0067mm$$

Isx79 Coordinate System ‘x’ Rapid Move Mode Control {New}

Range:	0 .. 1
Units:	none
Default:	0

Isx79 controls how Turbo PMAC computes the speed of “shorter” axes in a multi-axis RAPID-mode move. If Isx79 is set to the default value of 0, Turbo PMAC will compute the ratio of move distance to rapid-speed (Ixx16 or Ixx22, depending on the setting of Ixx90) for each motor in the move. Only the motor with highest distance/speed ratio will actually be commanded at its specified speed. The commanded speeds for other motors will be lessened so that they have the same ratio of distance to speed, yielding the same move time for all motors (before acceleration and deceleration are added). This makes the move path in a Cartesian system at least approximately linear (and truly linear if the acceleration and deceleration times set by Ixx19 – Ixx21 are the same).

If Isx79 is set to 1, all motors involved in the multi-axis move will be commanded to move at their specified speed. This means that motors with lower distance/speed ratios will finish sooner than those with higher ratios, and the path in a Cartesian system will not be linear in the general case.

In both cases, each motor involved in the multi-axis move will use its own acceleration parameters Ixx19 – Ixx21 to calculate its acceleration and deceleration profile.

Isx81 Coordinate System 'x' Blend Disable In-Position Time-Out {New}

Range:	0 .. 8,388,607
Units:	real-time interrupt periods
Default:	0 (disabled)

Isx81, if set to a positive value, specifies that when blending is disabled between programmed moves, Turbo PMAC will determine that all axes in the coordinate system will be "in-position" before starting execution of the next move in the motion program. In this case, the value of Isx81 specifies the "time-out" value for the in-position check, expressed in real-time interrupt periods (I8+1 servo cycles). If all axes in the coordinate system are not in-position within this time from the end of the incoming commanded move, the program will be stopped with a run-time error.

If Isx81 is set to 0, Turbo PMAC merely brings the commanded trajectory to a momentary stop before starting the next move in the cases where blending is disabled; it does not verify that any of the actual positions have reached this point. (This setting is fully compatible with firmware versions from before Isx81 was implemented.)

The in-position check as specified by Isx81 is performed after programmed moves that are never blended (RAPID-mode moves, programmed homing-search moves), or after moves that can be blended, but for which blending has been disabled by the clearing of the coordinate system "continuous motion request" status/control bit (directly or from Isx92), or because the corner is sharper than the angle specified by Isx83.

The in-position check enabled by Isx81 is the "background" check function that uses the Ixx28 in-position band for each motor, and the Ixx88 number of scans for each motor. When all axes in the coordinate system have been verified to be "in position", the coordinate-system "in-position" status bit is set. After the in-position check, a dwell of the time specified by Isx82 is inserted before execution of the next programmed move is started.

Isx82 Coordinate System 'x' Blend Disable Dwell Time {New}

Range:	0 .. 8,388,607
Units:	real-time interrupt periods
Default:	0

Isx82 specifies the dwell time that is automatically inserted between programmed moves when blending is disabled. This dwell time is inserted after programmed moves that are never blended (RAPID-mode moves, programmed homing-search moves), or after moves that can be blended, but for which blending has been disabled by the clearing of the coordinate system "continuous motion request" status/control bit (directly or from Isx92), or because the corner is sharper than the angles specified by Isx83 and Isx85. (If the angle is sharper than that specified by Isx83 but not sharper than that specified by Isx85, blending will be disabled at the corner, but no dwell will be added.) The units of this dwell are in real-time interrupt periods (I8+1 servo cycles), not milliseconds as in a directly programmed dwell.

If Isx81 is 0, this dwell time is inserted immediately after the end of commanded execution of the previous move. If Isx81 is greater than 0, this dwell time is inserted after all axes in the coordinate system have been verified to be "in position".

During this automatically inserted dwell time (if greater than 0), a coordinate-system "dwell-in-progress" status bit (Y:$002x40 bit 5, distinct from the motor "dwell-in-progress" status bits from a directly programmed dwell) will be set.

Isx83 Coordinate System 'x' Corner Blend Break Point {New}

Range:	-1.0 .. 0.9999 (floating-point)
Units:	cosine
Default:	0.0 (disabled)

Isx83, if set to a non-zero value, controls the threshold in Coordinate System 'x' between corner angles for which LINEAR and CIRCLE-mode moves are directly blended together, and those for which the axes are stopped in between the incoming and outgoing moves for the corner. Isx83 is only used if the "continuous motion request" (CMR) coordinate-system status/control bit is set to 1 permit blending. (This can be set by Isx92=0 at the beginning of a motion program, or can be set directly afterwards.) If the CMR bit is 0 (usually from Isx92=1), no blending occurs between any moves, and Isx83 is not used.

If Isx83 is set to 0.0, no decision on blending is made based on corner angle. In this case, the decision on blending is made by the "continuous motion request" coordinate-system status/control bit. (If a threshold corner angle of 90^o is desired – for which the Isx83 would be 0.0 – Isx83 should be set to a value that is not exactly 0.0, for example to 0.00001).

Isx83 is expressed as the cosine of the change in directed angle between the incoming and outgoing moves. (The change in directed angle is equal to 180^o minus the included angle of the corner.) As such, it can take a value between -1.0 and +1.0. If the two moves have the same directed angle at the move boundary (i.e. they are moving in the same direction), the change in directed angle is 0, and the cosine is 1.0. As the change in directed angle increases, the corner gets sharper, and the cosine of the change in directed angle decreases. For a total reversal, the change in directed angle is 180^o, and the cosine is -1.0. The change in directed angle is evaluated in the plane defined by the **NORMAL** command (default is the XY-plane); if the corner also involves axis movement perpendicular to this plane, it is the projection of movement into this plane that matters.

If the cosine of the change in directed angle at a corner is less than Isx83 (a large change in directed angle; a sharp corner), Turbo PMAC will automatically disable blending between the two moves, and bring the commanded trajectory to a stop at the end of the incoming move. If Isx81 is greater than 0, Turbo PMAC then verifies that all motors in the coordinate system are "in position". Next, if the cosine of this angle is also less than Isx85, a dwell of Isx82 milliseconds is executed. Finally, the outgoing move is automatically executed. From the time the incoming commanded move begins, through any in-position settling and added dwell, until the commanded outgoing move starts, the coordinate-system status bit “sharp corner stop” (Y:$002x40 bit 6) will be set.

If the cosine of the change in directed angle at a corner is greater than Isx83 (a small change in directed angle; a gradual corner), Turbo PMAC will directly blend the incoming and outgoing moves with its normal blending algorithms.

The operation of Isx83 is independent of the operation of the similar function of Isx99, which controls for outside corners in 2D cutter-radius compensation whether an arc move will be added based on the change in directed angle. Isx83 works regardless of whether cutter-radius compensation is active or not, or whether the corner is an inside or outside corner when cutter-radius compensation is active. If this is an outside compensated corner with an added arc, the corner angle is based on the moves without the added arc (i.e. the uncompensated moves).

Example
If it is desired that motion in Coordinate System 1 be stopped if the change in directed angle is greater than 30^o (included angle less than 150^o), then I5183 should be set to 0.866, because cos $\Delta\theta$ = cos 30^o = 0.866.

Isx84 Coordinate System ‘x’ Outside Corner Stop Point Control {New}

Range:	0 .. 1
Units:	none
Default:	0

Isx84 controls where the programmed movement stops when blending is disabled due to a corner sharper than the Isx83 threshold on an outside corner in 2D cutter-radius compensation with an added arc at the corner. If Isx84 is 0, the programmed movement will stop at the end of the added arc; if Isx84 is 1, the programmed movement will stop at the beginning of the added arc.

If blending is disabled due to single-step operation (S), program termination (Q or /), or continuous-motion-request bit cleared (usually from Isx92=1), programmed movement will always stop at the end of the added arc, regardless of the setting of Isx92=1.

Isx85 Coordinate System ‘x’ Corner Dwell Break Point {New}

Range:	-1.0 .. 0.9999 (floating-point)
Units:	cosine
Default:	0.0 (disabled)

Isx85, if set to a non-zero value, controls the threshold in Coordinate System ‘x’ between corner angles for which LINEAR and CIRCLE-mode moves are made without an automatically intervening dwell, and those for which a dwell of Isx82 real-time-interrupt periods is automatically added. Isx85 is only used if the “continuous motion request” (CMR) coordinate-system status/control bit is set to 1 permit blending. (This can be set by Isx92=0 at the beginning of a motion program, or can be set directly afterwards.) If the CMR bit is 0 (usually from Isx92=1), no blending occurs between any moves, and Isx85 is not used.

The corner angle is only evaluated against Isx85 if the blending at the corner has been disabled due to the action of Isx83. This means that Isx85 must be set to a value less than Isx83 if it is to have any effect.

If Isx85 is set to 0.0, no dwell is ever automatically added. (If a threshold corner angle of 90^o is desired – for which the Isx85 would be 0.0 – Isx85 should be set to a value that is not exactly 0.0, for example to 0.00001).

Isx85 is expressed as the cosine of the change in directed angle between the incoming and outgoing moves. (The change in directed angle is equal to 180^o minus the included angle of the corner.) As such, it can take a value between -1.0 and +1.0. If the two moves have the same directed angle at the move boundary (i.e. they are moving in the same direction), the change in directed angle is 0, and the cosine is 1.0. As the change in directed angle increases, the corner gets sharper, and the cosine of the change in directed angle decreases. For a total reversal, the change in directed angle is 180^o, and the cosine is -1.0. The change in directed angle is evaluated in the plane defined by the **NORMAL** command (default is the XY-plane); if the corner also involves axis movement perpendicular to this plane, it is the projection of movement into this plane that matters.

If the cosine of the change in directed angle at a corner is less than Isx85 (a large change in directed angle; a sharp corner), Turbo PMAC will automatically add a dwell of Isx82 real-time-interrupt periods before the outgoing move is started. From the time the incoming commanded move begins, through any in-position settling and added dwell, until the commanded outgoing move starts, the coordinate-system status bit "sharp corner stop" (Y:$002x40 bit 6) will be set.

If the cosine of the change in directed angle at a corner is greater than Isx85 (a small change in directed angle; a gradual corner), Turbo PMAC will not automatically add a dwell.

The operation of Isx85 is independent of the operation of the similar function of Isx99, which controls for outside corners in 2D cutter-radius compensation whether an arc move will be added based on the change in directed angle. Isx85 works regardless of whether cutter-radius compensation is active or not, or whether the corner is an inside or outside corner when cutter-radius compensation is active. If this is an outside-compensated corner with an added arc, the corner angle is based on the moves without the added arc (i.e. the uncompensated moves).

Example
If it is desired that motion in Coordinate System 1 be stopped if the change in directed angle is greater than 45^o (included angle less than 135^o), then I5185 should be set to 0.707, because $\cos \Delta\theta = \cos 45^o = 0.707$.

Isx92 Coordinate System 'x' Move Blend Disable {Revised Description}

Range:	0 .. 1
Units:	none

Default: 0

Isx92 controls whether programmed moves for Coordinate System 'x' are automatically blended or not. If Isx92 is set to 0, programmed moves that can be blended together – LINEAR, SPLINE, and CIRCLE-mode – are blended together with no intervening stop, unless Isx83 is set to 1 non-zero value and the corner formed at the juncture of LINEAR and/or CIRCLE-mode moves is sharper than the Isx83 threshold. Upcoming moves are calculated during the current moves.

If Isx92 is set to 1, even moves that can be blended together are not blended. The commanded trajectory is brought to a stop at the end of each programmed move. If Isx81 is greater than 0, Turbo PMAC then verifies that all motors in the coordinate system are "in position". Next, a dwell of Isx82 real-time interrupt periods is executed. After the dwell, the next move is calculated and executed.

Isx92 is only evaluated when the **`R`** or **`S`** command is given to start program execution. It is used to set the coordinate system's "continuous motion request" (CMR) status/control bit, which is checked each time a blending decision is to be made. (If Isx92 is 1, the CMR bit is set to 0; if Isx92 is 0, the CMR bit is set to 1.) To change the mode of operation while the program is running, the CMR bit must be changed directly using an M-variable. Msx84 is the suggested M-variable for this bit for Coordinate System 'x' (M5184->X:$002040,4 for C.S. 1).

I8000 - I8191 Conversion Table Setup Lines {Revised, revised description}

Range: $000000 - $FFFFFF

Units: Modified Turbo PMAC Addresses

…

The following table summarizes the content of entries in the Encoder Conversion Table:

Method Digit	# of lines	Process Defined	Mode Switch	1st Additional Line	2nd Additional Line
\$0	1	1/T Extension of Incremental Encoder	None	-	-
\$1	1	ACC-28 style A/D converter (high 16 bits, no rollover)	0 = signed data 1 = unsigned data	-	-
\$2	2	Parallel Y-word data, no filtering	0 = normal shift 1 = unshifted	Width/Offset Word	-
\$3	3	Parallel Y-word data, with filtering	0 = normal shift 1 = unshifted	Width/Offset Word	Max Change per Cycle
\$4	2	"Time Base" scaled digital differentiation	None	Time Base Scale Factor	-
\$5	2	Integrated ACC-28 style A/D converter	0 = signed data 1 = unsigned data	Input Bias	-
\$6	2	Parallel Y/X-word data, no filtering	0 = normal shift 1 = unshifted	Width/Offset Word	-
\$7	3	Parallel Y/X-word data, with filtering	0 = normal shift 1 = unshifted	Width/Offset Word	Max Change per Cycle
\$8	1	Parallel Extension of Incremental Encoder	0 = PMAC(1) IC 1 = PMAC2 IC	-	-
\$9	2	Triggered Time Base, frozen	0 = PMAC(1) IC 1 = PMAC2 IC	Time Base Scale Factor	-
\$A	2	Triggered Time Base, running	0 = PMAC(1) IC 1 = PMAC2 IC	Time Base Scale Factor	-
\$B	2	Triggered Time Base, armed	0 = PMAC(1) IC 1 = PMAC2 IC	Time Base Scale Factor	-
\$C	1	Incremental Encoder, no or HW 1/T extension	0 = No Extension 1= HW 1/T Ext	-	-
\$D	3/5	Low-pass filter of parallel data	0 = Exponential filter 1 = Tracking Filter	Max Change per Cycle	Filter Gain (Inverse Time Constant); (5) Integral Gain
\$E	1	Sum or difference of entries	None	-	-
\$F	-	(Extended entry – type determined by 1st digit of 2nd line)	-	-	-
\$F/\$0	3	High-Resolution Interpolator Feedback	0 = PMAC(1) IC 1 = PMAC2 IC	\$0 Method digit & Address of 1st A/D converter	A/D Bias Term
\$F/\$1	5	High-Resolution Interpolator Diagnostic	-	-	Active A/D Bias Term
\$F/\$2	2	Byte-wide parallel Y-word data, no filtering	0 = normal shift 1 = unshifted	\$2 and Width/Offset Word	-
\$F/\$3	3	Byte-wide parallel Y-word data, with filtering	0 = normal shift 1 = unshifted	\$3 and Width/Offset Word	Max Change per Cycle
\$F\$4	3	Resolver Conversion	0 = CW 1 = CCW		A/D Bias Term

…

Low-Pass Filter Entries ($D): The $D entry is used to create one of two types of low-pass filters on a word of input data to provide smoothing of noisy measurements. The two types of filter are distinguished by bit 19 of the first setup line of the entry. If the bit-19 mode-switch bit is 0, typically making the second hex digit $0, the filter is a simple exponential filter. If the bit-19 mode-switch bit is 1, typically making the second hex digit $8, the filter is a more sophisticated tracking filter that includes an integrator to eliminate steady-state errors.

The simpler exponential filter, which is a three-line entry in the table, is suitable for the smoothing of noisy master data used for electronic gearing (position following) or electronic cams (external time base). However, it will produce lags even in the steady state (e.g. at constant velocity), so it is usually not suitable for smoothing servo feedback data because of the delays it introduces.

The more complex tracking filter, which is a five-line entry in the table, is suitable for smoothing either master data or feedback data, because its integrator eliminates steady-state errors. Still, its filtering can introduce delays in responding to dynamic changes (e.g. accelerations), so it needs to be set up carefully. This software tracking filter is dynamically equivalent to the hardware tracking filters common in resolver-to-digital converter ICs. It is commonly used to smooth the results of direct conversion of sinusoidal encoders.

Warning: Attempting to enter a tracking filter into a Turbo PMAC with firmware that does not support it (V1.940 or older) will result in disturbing or disabling any subsequent entries in the table as well as the filter entry.

Exponential Filter ($Dxxxxx, bit 19 = 0)

The equation of the exponential filter executed every servo cycle *n* is:

$$\begin{gathered} If\ [In(n) - In(n-1)] > Max\_change,\ In(n) = In(n-1) + Max\_change \\ If\ [In(n) - In(n-1)] < -Max\_change,\ In(n) = In(n-1) - Max\_change \\ Out(n) = Out(n-1) + (K/2^{23})*[In(n)-Out(n-1)] \end{gathered}$$

In, *Out*, and *K* are all signed 24-bit numbers (range -8,388,608 to 8,388,607). The difference *[In(n)-Out(n-1)]* is truncated to 24 bits to handle rollover properly.

The time constant of the filter, in servo cycles, is $(2^{23}/K)-1$. The lower the value of K, the longer the time constant.

No shifting action is performed. Any operations such as 1/T interpolation should have been done on the data already, so the source register for this filter is typically the result register of the previous operation in the conversion table.

Method/Address Word: The first setup line (I-variable) of an exponential filter entry contains a 'D' in the first hex digit (bits 20 – 23), a '0' in bit 19, and the address of the source X-register in bits 0 – 18. If it is desired to execute an exponential filter on the contents of a Y-register, the contents of the Y-register must first be copied to an X-register in the conversion table with a "parallel" entry ($2) higher in the table. The source addresses for exponential filter entries are almost always from the conversion table itself (X:$003501 – X:$0035BC). Since bits 16 – 18 of conversion table registers are 0, this makes the entire second hex digit of this line '0'. For example, to perform an exponential filter on the result of the fourth line of the table, the first setup line of the filter entry would be $D03504.

Maximum Change Word: The second setup line (I-variable) of an exponential filter entry contains the value "max change" that limits how much the entry can change in one servo cycle. The units of this entry are whatever the units of the input register are, typically 1/32 of a count. For example, to limit the change in one servo cycle to 64 counts with an input register in units of 1/32 count, this third line would be 64*32 = 2048.

Filter Gain Word: The third setup line (I-variable) of an exponential filter entry contains the filter gain value K, which sets a filter time constant T_f of $(2^{23}/K)$-1 servo cycles. Therefore, the gain value K can be set as $2^{23}/(T_f+1)$. For example, to set a filter time constant of 7 servo cycles, the filter gain word would be 8,388,608/(7+1) = 1,048,576.

Result Word: The output value of the exponential filter is placed in the X register of the third line of the conversion table entry. An operation that uses this value should address this third register; for example Ixx05 for position following, or the source address for a time-base conversion-table entry (to keep position lock in time base, this filter must be executed *before* the time-base differentiation, not afterward).

Tracking Filter ($Dxxxxx, bit 19 = 1)

For the tracking filter, the equation of the filter every servo cycle *n* is:

$$\begin{gathered}
\text{If } [In(n) - In(n-1)] > Max\_change,\ In(n) = In(n-1) + Max\_change \\
\text{If } [In(n) - In(n-1)] < -Max\_change,\ In(n) = In(n-1) - Max\_change \\
Err(n) = In(n) - Out(n-1) \\
Temp1 = (Kp/2^{23}) * Err(n) \\
Int(n) = Int(n-1) + (Ki/2^{23}) * Err(n) \\
Out(n) = Out(n-1) + Temp1 + Int(n)
\end{gathered}$$

In, *Out*, K_p and K_i are all signed 24-bit numbers (range -8,388,608 to 8,388,607). The difference *[In(n)-Out(n-1)]* is truncated to 24 bits to handle rollover properly.

The time constant of the filter, in servo cycles, is $(2^{23}/K_p)$-1. The lower the value of K_p, the longer the time constant.

No shifting action is performed. Any operations such as 1/T interpolation should have been done on the data already, so the source register for this filter is typically the result register of the previous operation in the conversion table.

Method/Address Word: The first setup line (I-variable) of a tracking filter entry contains a 'D' in the first hex digit (bits 20 – 23), a 1 in the bit-19 mode-switch bit, and the address of the source X-register in bits 0 – 18. If it is desired to execute a tracking filter on the contents of a Y-register, the contents of the Y-register must first be copied to an X-register in the conversion table with a "parallel" entry ($2) higher in the table. The source addresses for tracking filter entries are almost always from the conversion table itself (X:$003501 – X:$0035BC). Since bits 16 – 18 of conversion table registers are 0, this makes the entire second hex digit of this line '8'. For example, to perform an tracking filter on the result of the fourth line of the table, the first setup line of the filter entry would be $D83504.

Maximum Change Word: The second setup line (I-variable) of a tracking filter entry contains the value “max change” that limits how much the entry can change in one servo cycle. The units of this entry are whatever the units of the input register are, typically 1/32 of a count. For example, to limit the change in one servo cycle to 64 counts with an input register in units of 1/32 count, this third line would be 64*32 = 2048.

Filter Proportional Gain Word: The third setup line (I-variable) of a tracking filter entry contains the filter proportional gain value K_p, which sets a filter time constant T_f of $(2^{23}/K_p)$-1 servo cycles. Therefore, the proportional gain value K_p can be set as $2^{23}/(T_f+1)$. For example, to set a filter time constant of 7 servo cycles, the filter proportional gain word would be 8,388,608/(7+1) = 1,048,576.

Reserved Setup Word: The fourth setup line (I-variable) of a tracking filter entry is reserved for future use. It is not presently used, and can be set to 0.

Filter Integral Gain Word: The fifth setup line (I-variable) of a tracking filter entry contains the filter integral gain value K_i, which determines how quickly the integrated error contributes to the filter output. Each servo cycle, the amount $(Ki/2^{23}) * Err(n)$ is added to the integrator and to the filter output.

Result Word: The output value of the tracking filter is placed in the X-register of the fifth line of the conversion table entry. An operation that uses this value should address this fifth register; for example Ixx03 for position-loop feedback, or the source address for a time-base conversion-table entry (to keep position lock in time base, this filter must be executed *before* the time-base differentiation, not afterward).

High-Resolution Interpolator Entries ($F/$0): An ECT entry in which the first hex digit of the first line is $F and the first hex digit of the second line is $0 processes the result of a high-resolution interpolator for analog “sine-wave” encoders, such as the ACC-51. This entry, when used with a high-resolution interpolator, produces a value with 4096 states per line. The entry must read both an encoder channel for the whole number of lines of the encoder, and a pair of A/D converters to determine the location within the line, mathematically combining the values to produce a single position value.

Encoder Channel Address: The first line of the three-line entry contains $F in the first hex digit and the base address of the encoder channel to be read in the low 19 bits (bits 0 to 18). If the bit-19 mode switch of the line is set to 0, Turbo PMAC expects a PMAC(1)-style Servo IC on the interpolator, as in the ACC-51P. If the bit-19 mode switch bit is set to1, Turbo PMAC expects a PMAC2-style Servo IC on the interpolator, as in the ACC-51E, 51C, and 51P2.

The following table shows the possible entries when PMAC(1)-style Servo ICs are used, as in the ACC-51P.

High-Res Interpolator Entry First Lines for PMAC(1)-Style Servo ICs

Servo IC #	Channel 1	Channel 2	Channel 3	Channel 4
2	$F78200	$F78204	$F78208	$F7820C
3	$F78300	$F78304	$F78308	$F7830C
4	$F79200	$F79204	$F79208	$F7920C
5	$F79300	$F79304	$F79308	$F7930C
6	$F7A200	$F7A204	$F7A208	$F7A20C
7	$F7A300	$F7A304	$F7A308	$F7A30C
8	$F7B200	$F7B204	$F7B208	$F7B20C
9	$F7B300	$F7B304	$F7B308	$F7B30C

The following table shows the possible entries when PMAC2-style Servo ICs are used, as in the ACC-51E, 51C, or 51P2:

High-Res Interpolator Entry First Lines for PMAC2-Style Servo ICs

Servo IC #	Channel 1	Channel 2	Channel 3	Channel 4
2	$FF8200	$FF8208	$FF8210	$FF8218
3	$FF8300	$FF8308	$FF8310	$FF8318
4	$FF9200	$FF9208	$FF9210	$FF9218
5	$FF9300	$FF9308	$FF9310	$FF9318
6	$FFA200	$FFA208	$FFA210	$FFA218
7	$FFA300	$FFA308	$FFA310	$FFA318
8	$FFB200	$FFB208	$FFB210	$FFB218
9	$FFB300	$FFB308	$FFB310	$FFB318

Note that by setting the bit-19 mode switch to 1, the second hex digit changes from “7” to “F”.

A/D Converter Address: The second setup line (I-variable) of the entry contains $0 in the first hex digit and the base address of the first of two A/D converters to be read in the low 19 bits (bits 0 to 18). The second A/D converter will be read at the next higher address. The following table shows the possible entries when the ACC-51P, with PMAC(1) style Servo ICs, is used:

High-Res Interpolator Entry Second Lines for PMAC(1)-Style Servo ICs

Servo IC #	Channel 1	Channel 2	Channel 3	Channel 4
2	$078202	$078206	$07820A	$07820E
3	$078302	$078306	$07830A	$07830E
4	$079202	$079206	$07920A	$07920E
5	$079302	$079306	$07930A	$07930E
6	$07A202	$07A206	$07A20A	$07A20E
7	$07A302	$07A306	$07A30A	$07A30E
8	$07B202	$07B206	$07B20A	$07B20E
9	$07B302	$07B306	$07B30A	$07B30E

The following table shows the possible entries when PMAC2-style Servo ICs are used, as in the ACC-51E, 51C, or 51P2:

High-Res Interpolator Entry Second Lines for PMAC2-Style Servo ICs

Servo IC #	Channel 1	Channel 2	Channel 3	Channel 4
2	$078205	$07820D	$078215	$07821D
3	$078305	$07830D	$078315	$07831D
4	$079205	$07920D	$079215	$07921D
5	$079305	$07930D	$079315	$07931D
6	$07A205	$07A20D	$07A215	$07A21D
7	$07A305	$07A30D	$07A315	$07A31D
8	$07B205	$07B20D	$07B215	$07B21D
9	$07B305	$07B30D	$07B315	$07B31D

Sine/Cosine Bias Word: The third setup line (I-variable) in a high-resolution sinusoidal-encoder conversion feedback entry contains bias-correction terms for the sine and cosine ADC values. The high twelve bits (the first three hex digits) contain the bias-correction term for the sine input; the low twelve bits (the last three hex digits) contain the bias-correction term for the cosine input. Each 12-bit section should be treated as a signed 12-bit value (so if the most significant of the 12 bits is a 1, the bias value is negative).

Each 12-bit bias-correction term should contain the value opposite that which the high 12 bits of the matching A/D converter report when they should ideally report zero. In action, the bias term will be added to the high 12 bits of the corresponding ADC reading before subsequent calculations are done.

For example, if the bias-correction word were set to $004FFA, the sine bias correction would be +4 LSBs of a 12-bit ADC, and the cosine bias correction would be -6 LSBs ($FFA = -6) of a 12-bit ADC. In use, 4 12-bit LSBs would be *added to* the sine reading, and 6 12-bit LSBs would be *subtracted from* the cosine reading each cycle before further processing.

In most cases, the bias-correction word will be determined automatically by a high-resolution "diagnostic" entry (format $F/$1) in the conversion table. The result of that diagnostic entry, containing both bias corrections, can simply be copied into this setup word.

Note: In firmware revisions 1.940 and older, the bias word contained a single 24-bit bias term that was added to both the sine and the cosine terms.

Conversion Result: The result of the conversion is placed in the X-register of the third line of the entry. Careful attention must be paid to the scaling of this 24-bit result. The least significant bit (Bit 0) of the result represents 1/4096 of a line of the sine/cosine encoder.

When Turbo PMAC software reads this data for servo use with Ixx03, Ixx04, Ixx05, or Isx93, it expects to find data in units of 1/32 of a "count". Therefore, PMAC software regards this format as producing 128 "counts" per line. (The fact that the hardware counter used produces 4 counts per line is not relevant to the actual use of this format; this fact would only be used when reading the actual hardware counter for commutation or debugging purposes.)

Example: This format is used to interpolate a linear scale with a 40-micron pitch (40µm/line), producing a resolution of about 10 nanometers (40,000/4096), used as position feedback for a motor. PMAC considers a "count" to be 1/128 of a line, yielding a count length of 40/128 = 0.3125 µm. To set user units of millimeters for the axis, the axis scale factor would be:

$$AxisScaleFactor = \frac{1mm}{UserUnit} * \frac{1000\mu m}{mm} * \frac{count}{0.3125\mu m} = 3200\frac{counts}{UserUnit}$$

High-Resolution Interpolation Diagnostic Entry ($F/$1): An ECT entry in which the first hex digit of the first line is $F and the first hex digit of the second line is $1 produces either vector magnitude or analog-input bias terms for the sine and cosine inputs of a sinusoidal encoder or resolver. This is a five-line entry. These result values can be used to verify proper setup and interface of the encoder and to optimize the accuracy of the conversion during initial setup, and/or to check for loss of the encoder during the actual application. Bit 0 of the second setup line determines whether the result produced is the sum of the squares of the two analog inputs (bit 0 = 0) or the bias terms for the analog inputs (bit 0 = 1).

Method/Address Setup Word: The first setup line (I-variable) of the five-line entry contains $F in the first hex digit, and the address of the first of the two A/D converters in the low 19 bits (bits 0 – 18). The second A/D converter will be read at the next higher address.

The following table shows the possible entries when the ACC-51P, with PMAC(1) style Servo ICs, is used:

High-Res Interpolator Diagnostic Entry First Lines for PMAC(1)-Style Servo ICs

Servo IC #	Channel 1	Channel 2	Channel 3	Channel 4
2	$F78202	$F78206	$F7820A	$F7820E
3	$F78302	$F78306	$F7830A	$F7830E
4	$F79202	$F79206	$F7920A	$F7920E
5	$F79302	$F79306	$F7930A	$F7930E
6	$F7A202	$F7A206	$F7A20A	$F7A20E
7	$F7A302	$F7A306	$F7A30A	$F7A30E
8	$F7B202	$F7B206	$F7B20A	$F7B20E
9	$F7B302	$F7B306	$F7B30A	$F7B30E

The following table shows the possible entries when PMAC2-style Servo ICs are used, as in the ACC-51E, 51C, or 51P2:

High-Res Interpolator Diagnostic Entry First Lines for PMAC2-Style Servo ICs

Servo IC #	Channel 1	Channel 2	Channel 3	Channel 4
2	$F78205	$F7820D	$F78215	$F7821D
3	$F78305	$F7830D	$F78315	$F7831D
4	$F79205	$F7920D	$F79215	$F7921D
5	$F79305	$F7930D	$F79315	$F7931D
6	$F7A205	$F7A20D	$F7A215	$F7A21D
7	$F7A305	$F7A30D	$F7A315	$F7A31D
8	$F7B205	$F7B20D	$F7B215	$F7B21D
9	$F7B305	$F7B30D	$F7B315	$F7B31D

Diagnostic Mode Setup Word: The second setup line (I-variable) of the five-line entry contains $1 in the first hex digit and $0000 in the second through fifth hex digits. Bits 0 and 1 in the sixth hex digit control the diagnostic mode (bits 2 and 3 should be left at 0). If bit 0 is set to 0 (making the word $100000), the entry computes the sum of squares of the sine and cosine ADCs, permitting monitoring of the vector magnitude of the inputs.

If bit 0 is set to 1, the entry computes the bias in the sine and cosine terms as the negative of average of the maximum positive and maximum negative values found for each term. Each cycle it checks the present readings against the logged maximum and minimum values, changing these values if necessary, then computing the averages and the resulting bias word. If bit 1 is set to 0, the maximum and minimum values are cleared. This setting (second setup word set to $100001) is used to start a test to determine the bias compensation word. As soon as Turbo PMAC starts accumulating maximum and minimum values (the next servo cycle), bit 1 is set to 1, making this second setup word equal to $100003. If you want to start a new test, for example after a circuit adjustment, you must set bit 1 to 0 again by setting this setup word back to $100001.

Active Bias Correction Setup Word: The third setup line (I-variable) of the five-line entry contains the sine and cosine bias terms that are used in the sum-of-squares calculations. Two signed 12-bit bias terms are combined in a 24-bit word. The sine bias-correction term is in the high 12 bits (bits 12 – 23); the cosine bias-correction term is in the low 12 bits (bits 0 – 11). These terms match the high 12 bits from the corresponding A/D converters. This word does not necessarily match the bias "result" term derived from using this entry to determine a suggested bias correction, or the bias correction used in the "feedback" table entry for the encoder or resolver.

Reserved Setup Words: The fourth and fifth setup lines of this entry type are reserved for future use, and should be left at 0.

Result Word (Sum of Squares): When bit 0 of the second setup line is 0, the final (fifth) result word contains the sum of squares of the biased sine and cosine measurements for the most recent servo cycle.

$$Result = (SineADC + SineBias)^2 + (Cosine\ ADC + CosineBias)^2$$

The values *SineADC* and *CosineADC* are read from the A/D converters at the address specified in the first setup line. The values *SineBias* and *CosineBias* are read from the third setup line.

To understand the scaling of the result word, it is best to think of all four of the values as being normalized, that is, having a valid range of -1.0 to +1.0. With small bias terms, the sum of squares result would have a possible normalized value of 0.0 to +2.0. When read as an unsigned integer, this register has a range of 0 to 16,777,215 ($FFFFFFF), corresponding to a normalized range of 0.0 to 2.0.

When the encoder and interpolator circuitry, or the resolver and excitation circuitry, are working properly, the sum of squares should have a normalized value of +0.25 to +0.9999 (2,097,152 to 8,388,607, or $200000 to $7FFFFF). If the resulting normalized value is greater than or equal to +1.0 (8,388,608, or $800000), meaning that the most significant bit (bit 23) is set to 1, at any point in the cycle, this indicates that saturation has occurred in at least one of the readings due to either too large a signal or a significant bias. This should be corrected before using this sensor in actual operation.

If the result has a normalized value of less than +0.25 (2,097,152, or $200000), meaning that bits 23, 22, and 21 are all 0, at low sensor frequencies, the signals are too small to get full resolution from the result, and this should be corrected before using this sensor in actual operation. Many sinusoidal encoders do have a reduction in signal magnitude of up to one-half at their highest frequencies, reducing the magnitude of this square term by three-quarters, and this is acceptable.

It is possible to monitor this term in the actual application to check for loss of the encoder. If the inputs are no longer driven externally, for example because the cable has come undone, the positive and negative input pair to the ADC will pull to substantially the same voltage, and the output of the ADC will be a very small number, resulting in a small magnitude of the sum of squares in at least part of the cycle. (If both signals cease to be driven externally, the sum of squares will be small over the entire cycle). The high four bits (bits 20 – 23) of the sum-of-squares result can be monitored, and if the four-bit value goes to 0, it can be concluded that the encoder has been "lost", and the motor should be "killed".

Ideally, the magnitude of the sum-of-squares result should be constant throughout the sine/cosine cycle, at least at constant frequency. If there is significant variation, this is an indication of signal imperfection. In most cases, the most important imperfection is a DC bias on the sine and/or cosine signals. This entry can be used in its alternate format to determine the optimal bias correction. Once that bias correction has been determined (the result word in that format), it can be copied into the active correction setup word for the diagnostic entry, and the entry put back into sum-of-squares mode, as an important verification that a good bias correction has been determined.

A/D Bias Result Word: When bit 0 of the second setup line is 1, the final (fifth) result word contains the suggested bias correction word containing the bias correction terms for the sine and cosine terms. This 24-bit value, containing two signed 12-bit correction terms, can be copied into the third setup word for the interpolator diagnostic entry for confirmation of its effect, and to the third line of the interpolator feedback entry, or the resolver feedback entry, for actual use. The sine bias-correction term is in the high 12 bits (bits 12 – 23); the cosine bias-correction term is in the low 12 bits (bits 0 – 11).

In this mode, the encoder should be moved for several seconds (motion by hand is OK) to ensure good sampling of maximums and minimums of both waveforms and accurate bias-correction terms. It is probably best to do this test with the amplifier disabled to prevent the possibility of noise distorting the maximum and minimum readings.

Resolver Conversion Entry ($F/$4): An ECT entry in which the first hex digit of the first line is $F and the first hex digit of the second line is $4 converts the result of a pair of resolver sine/cosine A/D converters (ADCs) to a resolver angle value with 14-bit resolution.

The $E entry converts the sine and cosine resolver feedback values processed through the Geo PMAC's A/D converter (ADC) registers to a 14-bit resolver angle value.

Method/Address Word: The first setup line of a resolver conversion entry contains $F in the first hex digit and the Y-address of the first ADC register to be read in the low 19 bits (bits 0 – 18). The next ADC register is read at the next higher Y-address. If bit 19 of the line is set to 0, the conversion creates a "clockwise" rotation sense. If bit 19 of the line is set to 1, the conversion creates a "counter-clockwise" rotation sense.

For example, if the first ADC register is at Y:$078C00, the first line would be set to $F78C00 for a clockwise rotation sense, or to $FF8C00 for a counterclockwise rotation sense.

Excitation Address Setup Word: The second setup line in a resolver conversion entry contains $4 in the first hex digit, and the Y-address of the excitation value register in the low 19 bits (bits 0 – 18), used to correlate the excitation and the feedback values. Multiple resolver channels can use the same excitation register. For example, if the excitation address is at Y:$078C10, the second setup line would be set to $478C10

Sine/Cosine Bias Setup Word: The third setup line in a resolver conversion entry contains bias-correction terms for the sine and cosine ADC values. The high twelve bits (the first three hex digits) contain the bias-correction term for the sine input; the low twelve bits (the last three hex digits) contain the bias-correction term for the cosine input. Each 12-bit section should be treated as a signed 12-bit value (so if the most significant of the 12 bits is a 1, the bias value is negative).

Each 12-bit bias-correction term should contain the value opposite that which the high 12 bits of the matching A/D converter report when they should ideally report zero. In action, the bias term will be added to the high 12 bits of the corresponding ADC reading before subsequent calculations are done.

In most cases, the bias-correction word will be determined automatically by a high-resolution “diagnostic” entry (format $F/$1) in the conversion table. The result of that diagnostic entry, containing both bias corrections, can simply be copied into this setup word.

For example, if the bias-correction word were set to $004FFA, the sine bias correction would be +4 LSBs of a 12-bit ADC, and the cosine bias correction would be -6 LSBs ($FFA = -6) of a 12-bit ADC. In use, 4 12-bit LSBs would be *added to* the sine reading, and 6 12-bit LSBs would be *subtracted from* the cosine reading each cycle before further processing.

In most cases, the bias-correction word will be determined automatically by an analog “diagnostic” entry in the conversion table (method $F/$1). The result of that diagnostic entry, containing both bias corrections, can simply be copied into this setup word.

Result Word: The output value of the resolver conversion is placed in the 24-bit X-register of the third line of the conversion table entry. The values in bits 5 – 16 of the result word contain the high 12 bits of the calculated arctangent of the bias-corrected sine and cosine values from the resolver. Because PMAC software considers the value in bit 5 to be a “count” for its scaling purposes, this conversion returns resolver position values of a 12-bit conversion (4096 “counts” per cycle of the resolver).

However, because the conversion uses dual 14-bit converters and the arctangent calculations are valid to 15 bits, the result contains additional resolution in bits 0 – 4 that PMAC software considers to have “fractional”, but still real, count resolution. If the electromagnetic noise levels are low and the signals use near the full scale of the ADCs, a repeatable 14-bit resolution (16,384 states per cycle of the resolver) can be achieved.

Bits 17 – 23 of the result contain cycle data from software extension of the result to multiple resolver cycles. If the result is then used for feedback or master data, it will be further extended in the motor algorithms.

This resolver conversion is a direct, and not a tracking, conversion. As such, it is more dynamically responsive, but also more susceptible to measurement noise. If a more noise-immune result is desired, at the cost of some dynamic responsiveness (but still no steady-state tracking errors), a digital tracking filter can be implemented on this result with another conversion table entry (format $D8). The result of that filter entry can then be used as the feedback or master data.

New/Revised On-Line Command Descriptions

<CTRL-A> {Revised}

Function:	Abort all programs and moves
Scope:	Global
Syntax:	ASCII value 1D, $01

This command causes all closed-loop motors in Turbo PMAC to begin immediately to decelerate to a stop, aborting any currently running motion programs. It also brings any open-loop enabled motors to an enabled zero-velocity closed-loop state at the present position. If global I-variable I36 is set to 0, it will also enable any disabled motors, bringing them to a zero-velocity closed-loop state at the present position. However if I36 is set to 1, it will have no effect on disabled motors; the **`<CTRL-E>`** command should be used for these instead.

Each closed-loop motor will decelerate from its present command velocity to zero velocity at a rate defined by its own motor I-variable Ixx15. Note that a multi-axis system may not stay on its programmed path during this deceleration. If the time-base (override) value used by a motor is exactly 0% when the **`<CTRL-A>`** command is given, the motor will abort at the present position even if the command velocity is not zero; otherwise a ramp-down trajectory will be computed using Ixx15 and executed using the override value.

A **`<CTRL-A>`** stop to a program is not meant to be recovered from gracefully, because the axes will in general not stop at a programmed point. An on-line **`J=`** command may be issued to each motor to cause it to move to the end point that was programmed when the abort occurred. Then the program(s) can be resumed with an **`R`** (run) or **`<CTRL-R>`** command.

To stop a motion sequence in a manner that can be recovered from easily, use instead the Quit (**`Q`** or **`<CTRL-Q>`**), the Hold (**`H`** or **`<CTRL-O>`**), the Quick Stop (**`\`**) or the Halt (**`/`**) commands.

When Turbo PMAC is set up to power on with all motors killed (Ixx80 = 0) and with I36 set to 0, this command can be used to enable all of the motors (provided that they are not synchronous motors commutated by Turbo PMAC – in that case, the motors should be enabled with the **`$`** or **`$$`** phase-referencing command).

For multiple cards on a single serial daisy-chain, this command affects all cards on the chain, regardless of the current software addressing.

See Also

Stop Commands (Making Your Application Safe)
On-line commands **`A`**, **`E`**, **`<CTRL-E>`** **`$`**, **`$$`**, **`/`**, **`\`**, **`J=`**, **`H`**, **`<CTRL-O>`**, **`Q`**, **`<CTRL-Q>`**
I-variables I36, Ixx15, Ixx80.

<CTRL-E> {New}

Function:	Enable disabled motors

Scope: Global

Syntax: ASCII value 5D,$05

This command enables all of the disabled motors on the Turbo PMAC, closing the position loop at the present actual position. If a motor is open-loop enabled, it closes the position loop at the present actual position. It has no effect on closed-loop enabled motors.

If I36 is set to 1, the **`<CTRL-A>`** (abort all) command does not enable disabled motors, so the **`<CTRL-E>`** command is used for enabling all motors together. If I36 is set to 0, either the **`<CTRL-A>`** or **`<CTRL-E>`** command could be used.

Note that if the motor is a synchronous (zero-slip – Ixx78 = 0) motor commutated by Turbo PMAC (Ixx01 bit 0 = 1), a phase referencing is required after power-up/reset before the motor can be enabled. This is done automatically on power-up/reset if Ixx80 for the motor is set to 1 or 3, or subsequently with the motor-specific **`$`** command, or the coordinate-system-specific **`$$`** command. The **`<CTRL-E>`** command does not cause a phase referencing to be performed on any motor.

The coordinate-system-specific **`E`** command performs the comparable action for just the motors of the addressed coordinate system.

See Also

On-line commands **`A`**, **`<CTRL-A>`** **`E`**, **`$`**, **`$$`**
I-variables I36, Ixx80.

? {Revised}

Function: Report motor status words

Scope: Motor-specific

Syntax: **`?`**

…

Sixth character returned:

…

Bit 1 ***Integrator in Velocity Loop***: This bit is 1 when bit 1 of Ixx96 is set to 1 and the PID integrator is inside the velocity loop, acting on the velocity error. This bit is 0 when bit 1 of Ixx96 is 0 and the PID integrator is in the position loop, acting on the position error. (In firmware revisions V1.940 and older, this status bit was bit 0 of Ixx96.)

Bit 0 ***Alternate Command-Output Mode***: This bit is 1 when bit 0 of Ixx96 is set to 1 and the motor's commands are output in the alternate mode. If Ixx01 bit 0 is 1, this means that open-loop direct-microstepping commutation is performed instead of the normal closed-loop commutation. If Ixx01 bit 0 is 0, this means that the motor's non-commutated output is formatted as a sign-and-magnitude signal pair, instead of a single bipolar signal output. This bit is 0 when bit 0 of Ixx96 is set to 0 and the motor's commands are output in the standard mode. (In firmware revisions V1.940 and older, this status bit was the Ixx90 rapid speed control.)

??? {Revised}

Function:	Report global status words
Scope:	Global
Syntax:	**???**

…

Tenth character returned:

…

Bit 10 ***MACRO Ring Check Active***: This bit is 1 if Turbo PMAC is actively checking the MACRO ring for errors (I80 > 0). It is 0 otherwise.

Bit 9 ***MACRO Ring Active***: This bit is 1 if the MACRO ring is active, having passed the ring check functions specified by I80 – I82. It is 0 otherwise.

Bit 8 (Reserved for future use)

Eleventh character returned:

Bit 7 ***Fast Save Flash Block Error***: This bit is 1 if the most recent FSAVE or FREAD command failed due to a bad block of flash memory. It is 0 otherwise.

Bit 6 ***Fast Save Sector Clear Error***: This bit is 1 if a sector of flash memory for "fast save" parameters was being cleared (a process that takes 1 – 2 seconds, occurring after every 256th **FSAVE** command) when the most recent **FSAVE** or **FREAD** command occurred causing that command to fail. It is 0 otherwise.

Bit 5 ***MACRO Ring Break Message Received***: This bit is 1 if the Turbo PMAC has received a message from another station on the MACRO ring that that station detected a ring break immediately upstream of it. It is 0 otherwise.

Bit 4 ***MACRO Ring Break Detected***: This bit is 1 if the Turbo PMAC has detected a break in the MACRO ring immediately upstream of it. It is 0 otherwise.

Twelfth character returned:

Bit 3 ***Modbus TCP Active***: This bit is 1 if the Modbus TCP interface is active (I67 > 0). It is 0 otherwise.

Bits 1-2 (Reserved for future use)

Bit 0: ***Emergency Stop***: This bit is 1 if the motors on the integrated Turbo PMAC controller/amplifier have been halted by the dedicated Emergency Stop input. It is 0 otherwise.

A {Revised}

Function:	Abort all programs and moves in currently addressed coordinate system
Scope:	Coordinate-system specific
Syntax:	**`A`**

This command causes all closed-loop motors defined in the addressed coordinate system to begin immediately to decelerate to a stop, aborting the currently running motion program (if any). It also brings any open-loop enabled motors in the coordinate system to an enabled zero-velocity closed-loop state at the present position. If global I-variable I36 is set to 0, it will also enable any disabled motors in the coordinate system, bringing them to a zero-velocity closed-loop state at the present position. However if I36 is set to 1, it will have no effect on disabled motors; the **`E`** command should be used for these instead.

Each closed-loop motor in the coordinate system will decelerate from its present command velocity to zero velocity at a rate defined by its own motor I-variable Ixx15. Note that a multi-axis system may not stay on its programmed path during this deceleration. If the time-base (override) value for the coordinate system is exactly 0% when the **`A`** command is given, the motor will abort at the present position even if the command velocity is not zero; otherwise a ramp-down trajectory will be computed using Ixx15 and executed using the override value.

An **`A`** (abort) stop to a program is not meant to be recovered from gracefully, because the axes will in general not stop at a programmed point. An on-line **`J=`** command may be issued to each motor to cause it to move to the end point that was programmed when the abort occurred. Then the program(s) can be resumed with an **`R`** (run) command.

To stop a motion sequence in a manner that can be recovered from easily, use instead the Quit (**`Q`** or **`<CTRL-Q>`**), the Hold (**`H`** or **`<CTRL-O>`**), the Quick Stop (**`\`**) or the Halt (**`/`**) commands.

Example

```
B1R             ; Start Motion Program 1
A               ; Abort the program
#1J=#2J=        ; Jog motors to original move-end position
R               ; Resume program with next move
```

See Also

Stop Commands (Making Your Application Safe)
Control-Panel Port STOP/ Input (Connecting Turbo PMAC to the Machine)
I-variables I36, Ixx15, Ixx80
On-line commands **`<CTRL-A>`**, **`<CTRL-E>`**, **`E`**, **`H`**, **`J/`**, **`K`**, **`Q`**
JPAN connector pin 10

E {New}

Function: Enable disabled motors

Scope: Coordinate-system specific

Syntax: **E**

This command enables the disabled motors of the addressed coordinate system, closing the position loop at the present actual position. If a motor in the coordinate system is open-loop enabled, it closes the position loop at the present actual position. It has no effect on closed-loop enabled motors.

If I36 is set to 1, the **A** (abort) command does not enable disabled motors, so the **E** command is used for enabling the motors of a coordinate system. If I36 is set to 0, either the **A** or **E** command could be used.

Note that if the motor is a synchronous (zero-slip – Ixx78 = 0) motor commutated by Turbo PMAC (Ixx01 bit 0 = 1), a phase referencing is required after power-up/reset before the motor can be enabled. This is done automatically on power-up/reset if Ixx80 for the motor is set to 1 or 3, or subsequently with the motor-specific **$** command, or the coordinate-system-specific **$$** command. The **E** command does not cause a phase referencing to be performed on any motor.

The global **<CTRL-E>** command performs the comparable action for all of the motors on Turbo PMAC.

See Also

On-line commands **A**, **<CTRL-A>** **<CTRL-E>**, **$**, **$$**
I-variables I36, Ixx80.

FREAD {New}

Function: Read of parameters stored with **FSAVE**

Scope: Global

Syntax: **FREAD**

The **FREAD** command causes Turbo PMAC to copy the last values of P8107 – P8191 stored to a special sector of non-volatile flash memory with a **FSAVE** command back into the variables in active RAM. Values stored for these variables with the **FSAVE** command are never copied back into active RAM without an **FREAD** command.

On power-up/reset, Turbo PMAC automatically copies the last values stored in the main bank of flash memory by the most recent full **SAVE** command into active RAM (along with those for all of the other variables). These values may not be the same as the last values stored with the **FSAVE** command. A subsequent **FREAD** command will overwrite these values with the last values stored with an **FSAVE** command.

If nothing has been saved to these sectors of flash memory since it was last cleared with an **FSAVECLEAR** command, all values of P8107 – P8191 will be set to 0, but no error will be reported. If the sector of flash memory is bad, this command will be rejected with an error (reporting ERR022 if I6 is set to 1 or 3).

FSAVE {New}

Function: Fast save of certain parameters

Scope: Global

Syntax: **FSAVE**
FSAV

The **FSAVE** command causes Turbo PMAC to do a fast copy of the values of the highest 85 P-variables (P8107 – P8191) from active memory (RAM) to special sectors of non-volatile flash memory. This save operation of 512 bytes occurs in a few milliseconds rather than the several seconds for a full **SAVE** operation, so can be done without disrupting machine operation. (However, during the approximately 5 milliseconds it takes to execute the **FSAVE** command, no other background tasks, including safety checks, are performed. Foreground tasks such as servo loop closure are performed during the **FSAVE** process.) It is intended to permit the quick storage of machine-state information for proper start-up after a quick, or especially an unexpected, power-down.

After a power-up/reset, the values stored with the **FSAVE** command are not copied back into the active registers in RAM until an **FREAD** command is issued to the Turbo PMAC. On power-up/reset, Turbo PMAC automatically copies the last values for these variables stored with a full **SAVE** command to the main bank of flash memory. A subsequent **FREAD** command will overwrite these values with the last values stored with an **FSAVE** command.

After every 256th **FSAVE** command, one of the two flash-memory sectors for this function is filled, and Turbo PMAC must clear the other sector to enable further **FSAVE** commands, an operation that takes 1 – 2 seconds. This does not keep any other Turbo PMAC tasks from operating, but if another **FSAVE** command is executed during this time, it will be rejected with an error (reporting ERR021 if I6 is set to 1 or 3), and bit 6 of global status word Y:$000006 is set to 1. Operation of the flash memory past 100,000 clear cycles cannot be guaranteed, so there should be no more than 25,000,000 **FSAVE** commands issued over the lifetime of the machine.

If the saving operation fails due to a bad flash sector, the command will be rejected with an error (reporting ERR022 if I6 is set to 1 or 3), and bit 7 of global status word Y:$000006 will be set to 1.

The FSAVE command works on all flash-memory ICs being used in manufacturing as of the implementation of the command in 2005. However, it may not work on older Turbo PMACs. Bits 1 and 2 of I4904 must both be set to 1 (i.e. the last hex digit of I4904 must be $6 or $7) to permit operation of the **FASVE** command. If the command is issued to a Turbo PMAC with an older flash-memory IC, the command will be rejected with an error (reporting ERR020 if I6 is set to 1 or 3.)

FSAVECLEAR {New}

Function:	Clear **`FSAVE`** flash-memory sectors
Scope:	Global
Syntax:	**`FSAVECLEAR`** **`FSAVCLR`**

The **`FSAVECLEAR`** command causes Turbo PMAC to clear both of the sectors of flash memory reserved for the **`FSAVE`** "fast save" function. If an **`FREAD`** command is then issued before another **`FSAVE`** command is used to save real data into these flash sectors, all of the variable values will be set to 0.

MOVETIME {Revised}

Function:	Report time left in presently executing move
Scope:	Coordinate-system specific
Syntax:	**`MOVETIME`** **`MVTM`**

The **`MOVETIME`** command causes Turbo PMAC to report the time left in the currently executing commanded move in the addressed coordinate system, scaled in milliseconds (at 100%). If variable Isx14 for the addressed coordinate system is set to a non-zero value, the value of Isx14 is subtracted from this value calculated for response, the difference is converted to servo cycles, and this result is written to the Isx11 automatic countdown timer. This additional functionality facilitates anticipating the end of the commanded move.

This "time remaining" function is not active if the addressed coordinate system is executing moves in the special lookahead buffer.

For the **`MOVETIME`** command to work properly with **`LINEAR`** and **`CIRCLE`**-mode moves, the coordinate system must be in segmentation mode (Isx13>0).

New/Revised Program Command Descriptions

CC4 {New}

Function: Turn Off Cutter Radius Compensation

Type: Motion program (PROG and ROT)

Default: **`CC4`**

This turns off the cutter radius compensation mode, reducing it gradually through the next move. Unlike **`CC0`**, it does not automatically add a zero-distance move if it does not immediately find a programmed lead-out move. **`CC4`** is equivalent to the **`G40`** command of the machine-tool standard RS-274 language.

See Also:

Cutter (Tool) Radius Compensation
Program commands **`CC0`**, **`CC1`**, **`CC2`**, **`CC3`**, **`CCR{data}`**

CREAD {New}

Function: Read latest computed axis destination coordinates

Type: Motion program (PROG and ROT)

Syntax: **`CREAD [({axis}[,{axis}…])]`**

where:

- **`{axis}`** is a character (X, Y, Z, A, B, C, U, V, W) specifying which axis' destination coordinate is to be read

The **`CREAD`** (Coordinate READ) command causes Turbo PMAC to read the most recently computed move-destination coordinates for all axes or the specified axes in the coordinate system running the program, and place these values in particular Q-variables for the coordinate system. From these Q-variables, these values can then be synchronously assigned to floating-point M-variables, so these M-variables will hold the axis destination values when the move is actually executing. This facilitates the calculation of features such as "distance-to-go" reporting.

If no axis list follows **`CREAD`**, all nine axis destination values are placed in Q-variables. If there is an axis list in the command, only the values for the specified axes are placed in Q-variables

The following table shows which Q-variable for the coordinate system is used for each axis destination value:

Axis	Q-Variable	Axis	Q-Variable	Axis	Q-Variable
A	Q91	U	Q94	X	Q97
B	Q92	V	Q95	Y	Q98
C	Q93	W	Q96	Z	Q99

Examples:

```
; Without subroutine
A27 B35 CREAD(A,B) M91==Q91 M92==Q92

; Setup for coordinate read in subprogram
I38=1                ; Delay CALL until move calculated

; Part of main motion program
X10 Y20 CALL 500
X15 Y25 CALL 500

; Subprogram to implement coordinate read & M-var assignment
OPEN PROG 500 CLEAR
CREAD(X,Y) M97==Q97 M98==Q98 RETURN
CLOSE
```

Firmware Update Listing

V1.941 Updates (September 2005)

1. Added new features to manage behavior on cornering and when blending is disabled for other reasons.
    a. Isx81 permits in-position check when blending is disabled (not just momentary stop in commanded velocity).
    b. Isx82 permits an automatic added dwell when blending is disabled.
    c. Isx83 sets corner angle threshold between blended and non-blended corners.
    d. Isx84 determines whether stop is at beginning or end of added arc for outside corners in cutter radius compensation when blending disabled on sharp corner.
    e. Isx85 sets corner angle threshold between non-blended corners where dwell is added and where no dwell is added.
2. Added new coordinate system I-variable Isx14 to provide end-of-move anticipation feature. When **`MOVETIME`** command is given to coordinate system and Isx14 is not equal to 0, the sum of the remaining time and Isx14 is scaled into servo cycles and placed into the Isx11 timer.
3. Added new coordinate-system I-variable Isx78 that specifies maximum centripetal acceleration for circle moves. If programmed feedrate would cause this limit to be violated, feedrate is reduced so as not to violate this limit.
4. Added new coordinate-system I-variable Isx79 to control whether axes with a shorter distance in a multi-axis RAPID move are slowed to finish synchronously with the longest axis, or move at their own RAPID speed.
5. Added new motion program command **`CC4`** to turn off cutter radius compensation, but without automatically adding a zero-distance lead-out move when no programmed lead-out move immediately follows.
6. Added new coordinate-system I-variables Isx15 and Isx16 to support feedrate override at the segmentation stage.
7. Added new global I-variable I38 to permit execution of a single-line subroutine called from the same program line as a move command after the move command before the suspension of program calculations.
8. Added new global “do-nothing” I-variable I65 to permit the user to verify configuration download.
9. Changed reporting of motor I-variable Ixx68 from unsigned to signed (no change in functionality).
10. Corrected action of **`$$$***`** command for Option 5Fx 240 MHz CPUs.
11. Corrected checksum reporting over auxiliary serial port.
12. Corrected problem with opening FORWARD or INVERSE kinematic subroutine when PLC is executing a WHILE loop.
13. Corrected action of **`IDNUMBER`** and **`SID`** commands with Option 5Fx 240 MHz CPUs.
14. Added support for hardware position capture with high-resolution interpolation of sinusoidal encoders over the MACRO ring. Requires V1.117 or newer firmware on 8-axis MACRO Station, V1.203 or newer firmware on 16-axis MACRO Station or MACRO Peripherals.
15. Added support for Modbus Ethernet I/O with new I-variables I67 and I69.
16. Enhanced MACRO ring-break detection

  a. Automatically broadcasts ring-break message if detected here
  b. Automatically responds to received ring-break message
  c. Properly detects ring breaks found at power-up/reset
  d. Accepts **`CLEARFAULT`** (**`CLRF`**) command to reset ring-break fault status

17. Added five-line tracking-filter variant of exponential-filter conversion method ($D) to encoder conversion table, enabled by setting bit 19 of first setup line to 1, permitting low-pass filter without steady-state error – suitable for processing feedback data.
18. Changed bias word (third setup I-variable) for high-resolution sinusoidal encoder conversion ($F/$0) in encoder conversion table from a single 24-bit term used on both sine and cosine to two 12-bit terms, one for each signal.
19. Added five-line diagnostic method entry for highly interpolated sinusoidal encoders ($F/$1) in encoder conversion table to calculate sum-of-squares magnitude or sine/cosine bias word.
20. Added direct resolver conversion method entry ($F/$4) to encoder conversion table.
21. Modified action of "MaxChange" filter in conversion table methods $3, $7, and $F/$3. If MaxChange is exceeded, result is now changed by "LastChange", not "MaxChange", if previous cycle's result was good.
22. Added new global status bits for MACRO ring status and Modbus TCP network.
23. Added capability for disabling automatic parser of main serial port with (new) bit 1 of I43.
24. Added capability for fast save of 85 P-variables to special sectors of flash memory with **`FSAVE`**, **`FREAD`**, and **`FSAVECLEAR`** commands.
25. Added capability to move PID integrator to inside of velocity loop for improved disturbance rejection with (new) bit 1 of Ixx96. This changes meanings of bits 0 and 1 of first motor status word.
26. Added automatic emergency-stop function for Turbo PMAC integrated amplifier configurations (such as Geo Brick).
27. Added new variable I36 to permit separation of abort and enable functions. With I36=1, abort commands do not enable disabled motors. New commands **`E`** and **`<CTRL-E>`** enable disabled motors.

V1.942 Updates (October 2005)

1. Corrected operation of Ixx35 acceleration feedforwardterm, not operating correctly in V1.941 firmware.
2. Permits a CALL statement after a move command on a motion program line to jump to a label in the subprogram without halting program calculations if I38 is set to 1. (In V1.941, this only worked for jumps to the start of the subprogram.)

V1.943 Updates (January 2007)

1. Corrected problem to accept non-standard (ASCII code > 2007) characters in comments.
2. Register X:$003206 contains most recent command error code as 3-character ASCII byte code.
3. Corrected problem in compiled PLC (PLCC) computation of **`INT({var}/1)`** when **`{var}`** =1.
4. Corrected intermittent problem in homing search move when homing command is given while motor is killed.

5. For Geo Brick configuration, does not permit invalid settings of ADC strobe word variables I7m06.
6. Added support for new Modbus Ethernet features (require optional Modbus Ethernet firmware):
    a. New dual-ported RAM control-panel features (PMAC as Modbus slave)
    b. Direct access to PMAC I, M, P, and Q-variables (PMAC as Modbus master or slave)

V1.944 Updates (January 2008)

1. Added new single-bit variable I28 to permit disabling of automatic update of LCD display port. This allows use of DISP0-7 I/O lines for general purpose I/O.
2. Fixed operation of high-resolution sinusoidal-encoder interpolation conversion using the ACC-51S interpolator with Turbo PMAC2A-PC/104 and Turbo PMAC2A-Eth Lite ("Clipper").
3. Will not attempt absolute position read over MACRO ring when there is a MACRO ring fault.
4. After a MACRO-ring "clear fault", will immediately read ring error status bits in I6840 and declare a MACRO-ring fault again before trying any ring operations – this prevents possible lock-ups.
5. Added new control bit (bit 10) to Ixx24 to permit direct-PWM control of two-phase motors.
6. (Geo Brick firmware only) Improved power-on testing of amplifier functions.

V1.945 Updates (June 2008)

1. Added automatic clear of MACRO ring errors before attempting to read absolute power-on servo and/or phase positions. If ring errors persist after clear, do not perform these reads.
2. Added MACROSTASCIIFREQ set and query commands to facilitate reading and writing of ring frequencies for all stations.
3. Added new setup variable I29 to permit alternate address for automatic multiplexer port functions such as TWS-format M-variables for ACC-34 I/O boards.
4. Changed functionality of obsolete MACROAUX (MX), MACROAUXREAD (MXR), and MACROAUXWRITE (MXW) commands so they now support communications from Turbo PMAC acting as ring master to Turbo PMACs acting as MACRO-ring slaves.
5. Changed functionality of I71, I73, I75, and I77 MACRO setup variables. Bit values of 0 no longer specify obsolete "Type 0" slaves; instead support Turbo PMAC as MACRO slave.
6. Added new setup variable Ixx44 to permit motor to run as MACRO slave, accepting cyclical commands over the MACRO ring.
7. If receives MACRO flag status bit indicating multiple active nodes at same ring address, will set global status bit X:$6 bit 7 (configuration error) and kill all motors. If this status bit, which also can be set to indicate a change in the number or type of Servo or MACRO ICs found since last saved configuration, is set, no motors can be enabled.