DELTA TAU DATA SYSTEMS, INC.

TURBO PMAC/PMAC2
SOFTWARE REFERENCE ADDENDUM
FOR V1.938, V1.939, & V1.940 FIRMWARE

June, 2003

Delta Tau Data Systems 21314 Lassen St. Chatsworth, CA 91311
Ph: (818) 998-2095 Fax: (818) 998-7807
e-mail: support@deltatau.com Website: www.deltatau.com

NEW IDEAS IN MOTION....

TABLE OF CONTENTS

Setting Up Turbo PMAC for MLDT Feedback

Introduction

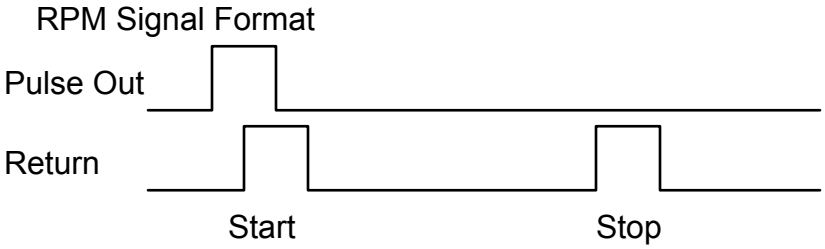
A Turbo PMAC with PMAC2-style ICs can provide direct interface to magnetostrictive linear displacement transducers (MLDTs), such as MTS's Temposonics brand. MLDTs can provide absolute position information in rugged environments; they are particularly well suited to hydraulic applications. In this interface Turbo PMAC provides a periodic excitation pulse output to the MLDT, receives the echo pulse that returns at the speed of sound in the transducer, and very accurately measures the time between these pulses, which is directly proportional to the distance of the moving member from the stationary base of the transducer. The timer therefore contains a position measurement.

Interface Type

MLDTs are available with several different interface formats; for this interface, a format with "external excitation" is required, because Turbo PMAC provides the excitation pulse. Usually this format has an "RS-422" interface, because the excitation and echo pulses are at RS-422 levels. The Turbo PMAC MLDT interface inputs and outputs are at RS-422 levels.

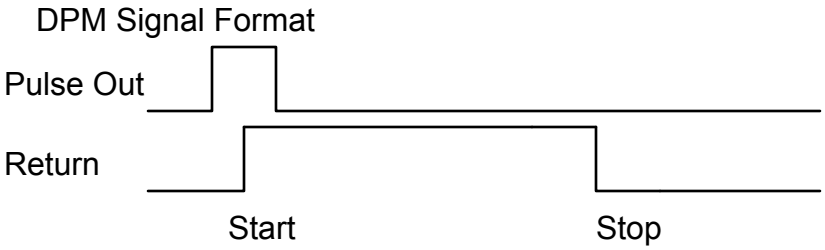
Signal Formats

There are two common signal formats of the external excitation type; MTS calls them "RPM" and "DPM". In the RPM format there are two short pulses returned from the MLDT: an immediate "start" pulse, and a delayed "stop" pulse.



Since Turbo PMAC uses the first rising signal edge returned after the falling edge of the output pulse to latch the timer, the key setup issue in this format is to make sure that the output pulse width is large enough so that the falling edge of the output pulse occurs after the rising edge of the return line's start pulse (see "PFM Pulse Width", below).

In the DPM format, there is only one long pulse returned from the MLDT.



The rising edge of the return pulse in the DPM format is the equivalent of the rising edge of the start pulse in the RPM format. The falling edge of the return pulse in the DPM format is the equivalent to the rising edge of the stop pulse in the RPM format. Because Turbo PMAC is expecting a rising signal edge to latch the timer, in this signal format the return signals should be inverted so that the '+' output of the MLDT is wired into Turbo PMAC's '-' input, and vice versa.

Hardware Setup

The PULSEn output that is commonly used to command stepper drives is used as the excitation signal for the MLDT; the CHAn input that is typically part of encoder feedback is used to accept the response. The PULSEn output is an RS-422 style differential line-drive pair. The CHAn input is an RS-422 style differential line receiver pair. The use of differential pairs for both inputs and outputs is strongly encouraged for the common-mode noise rejection it provides.

Remember that in the DPM signal format or equivalent (see above), the '+' output of the MLDT should be wired into the CHAn- input, and the '-' output of the MLDT should be wired into the CHAn+ input.

Turbo PMAC Hardware-Control Parameter Setup

PFMCLK Frequency: I7m03, I6803, MI903, MI907, MI993

The pulse output uses Turbo PMAC's pulse frequency modulation (PFM) feature. The PFM circuitry generates periodic output pulses by repeatedly adding a command value into an accumulator. When the accumulator overflows, an output pulse is generated.

The addition of the command value into the accumulator is performed once per PFMCLK cycle. The PFMCLK frequency is governed by I7m03 for the channels on Servo IC m, I6803 for the supplemental channels on MACRO IC 0, and MI903, MI907, or MI993 for channels on a MACRO Station. The default frequency of the PFMCLK for all channels is 9.83 MHz; this frequency should be suitable for all MLDT applications.

PFM Output Frequency: Mxx07, MI926

The pulse output frequency for a channel is controlled by both the PFMCLK frequency and the PFM command value for the channel, which is the C-output register for that channel. When used for stepper motor applications, the PFM command value is determined by the instantaneous command velocity and the gains of the simulated servo loop on Turbo PMAC; Ixx02 tells Turbo PMAC to write this to the PFM command register.

For MLDT use, we will write to the PFM command register once on power-up/reset with an M-variable. (If the interface is on a MACRO Station, the Station's firmware will do this automatically, using the saved value of node-specific variable MI926.) The suggested M-variable definition for this register is Mxx07. The following table shows the registers for these suggested definitions:

PFM Pulse Output Addresses (Y-registers)

IC# - Chan#	0 - 1	0 - 2	0 - 3	0 - 4	1 - 1	1 - 2	1 - 3	1 - 4
Mxx07->Y:	**$078004**	**$07800C**	**$078014**	**$07801C**	**$078104**	**$07810C**	**$078114**	**$07811C**
IC# - Chan#	2 - 1	2 - 2	2 - 3	2 - 4	3 - 1	3 - 2	3 - 3	3 - 4
Mxx07->Y:	**$078204**	**$07820C**	**$078214**	**$07821C**	**$078304**	**$07830C**	**$078314**	**$07831C**
IC# - Chan#	4 - 1	4 - 2	4 - 3	4 - 4	5 - 1	5 - 2	5 - 3	5 - 4
Mxx07->Y:	**$079204**	**$07920C**	**$079214**	**$07921C**	**$079304**	**$07930C**	**$079314**	**$07931C**
IC# - Chan#	6 - 1	6 - 2	6 - 3	6 - 4	7 - 1	7 - 2	7 - 3	7 - 4
Mxx07->Y:	**$07A204**	**$07A20C**	**$07A214**	**$07A21C**	**$07A304**	**$07A30C**	**$07A314**	**$07A31C**
IC# - Chan#	8 - 1	8 - 2	8 - 3	8 - 4	9 - 1	9 - 2	9 - 3	9 - 4
Mxx07->Y:	**$07B204**	**$07B20C**	**$07B214**	**$07B21C**	**$07B304**	**$07B30C**	**$07B314**	**$07B31C**
IC# - Chan#	M0 – 1*	M0 – 2*						
Mxx07->Y:	**$078414**	**$07841C**						

A sample M-variable definition is:

```
M107->Y:$078004,8,16,S  ; Channel 1 PFM (C) command value
```

Note: The PFM command registers are actually 24-bit registers; the suggested definitions just use the high 16 bits, providing enough resolution for our purposes here. If you use the register as a 24-bit value, e.g. **`M107->Y:$078004,0,24,S`***, it takes numerical values 256 times higher than the equations shown here. When using the MACRO Station, MI926 is a 24-bit, not 16-bit value.*

The frequency of the pulse output should produce a period just slightly longer than the longest expected response time for the echo pulse. For MLDTs, the response time is approximately 0.35 μsec/mm (9 μsec/inch). On an MLDT 1500 mm (~60 in) long, the longest response time is approximately 540 μsec; a recommended period between pulse outputs for this device is 600 μsec, for a frequency of 1667 Hz.

To produce the desired pulse output frequency from a Turbo PMAC IC, the following formula can be used (assuming a 16-bit M-variable definition):

$$OutputFreq(kHz) = \frac{Mxx07}{65{,}536} PFMCLK\_Freq(kHz)$$

or:

$$Mxx07 = 65{,}536 * \frac{OutputFreq(kHz)}{PFMCLK\_Freq(kHz)}$$

To produce a pulse output frequency of 1.667 kHz with the default PFMCLK frequency of 9.83 MHz, we calculate:

$$Mxx07 = 65{,}536 * \frac{1.667}{9{,}380} \cong 11$$

To write this value to the register, a power-on PLC routine is suggested; this can also be done with on-line commands from the host computer. Turbo PMAC firmware will not automatically write this value to the register.

To produce the desired pulse output frequency from a channel on the MACRO Station, the following formula can be used for node-specific 24-bit variable MI926:

$$OutputFreq(kHz) = \frac{MI926}{16{,}777{,}216} PFMCLK\_Freq(kHz)$$

or:

$$MI926 = 16{,}777{,}216 * \frac{OutputFreq(kHz)}{PFMCLK\_Freq(kHz)}$$

To produce a pulse output frequency of 1.667 kHz with the default PFMCLK frequency of 9.83 MHz, we calculate:

$$MI926 = 16{,}777{,}216 * \frac{1.667}{9{,}380} = 2{,}982$$

Note: The servo update time for the motor using the MLDT should be at least as high as the output time set here (the servo frequency should be as low or lower than the output frequency).

PFM Pulse Width: I7m04, I6804, MI904, MI908, MI994

The width of the output pulse is controlled by the PFMCLK frequency with I7m04 for the channels on Servo IC m, I6804 for the channels on MACRO IC 0, or MI904, MI908, or MI994 for the ICs on a MACRO Station. This I-variable specifies the pulse width as the number of PFMCLK cycles. At the default PFMCLK frequency of 9.83 MHz, the default value of 15 produces a 1.5-μsec output pulse width. This should be satisfactory for most MLDT devices. When using the RPM format or equivalent (see *Signal Format*, above), the pulse width must be large enough to enclose the rising edge of the returned "start" pulse – that is, it must be longer than the delay between the output pulse and the returned start pulse.

PFM Format Select: I7mn6, I68n6, MI916

The output format of channel signals is controlled by variable I7mn6 for Servo IC m Channel n, by I68n6 for MACRO IC 0 Channel n, or by node-specific variable MI916 on a MACRO Station. In order for the C-register circuitry of Channel n to output a PFM pulse train rather than a PWM pulse train, variable this variable must be set to 2 or 3. Most commonly, it will be set to 3, so that the A and B registers for Channel n output DAC signals rather than PWM.

Note: You cannot use one channel of Turbo PMAC simultaneously for direct PWM control of a motor, and for MLDT pulse generation. Direct PWM control of a motor automatically writes to the channel's A, B, and C registers every phase cycle.

MLDT Feedback Select: I7mn0, I68n0, MI910

The decoding of the signals on the "encoder" inputs is controlled by I7mn0 for Servo IC m Channel n, I68n0 for MACRO IC 0 Channel n, or by node-specific variable MI910 for a channel on a MACRO Station For proper decoding of the MLDT signal, this variable must be set to 12. With this setting, the pulse timer is cleared to zero at the falling edge of the output pulse. It then counts up at 117.96 MHz until a rising edge on the return pulse is received, at which time the timer's value is latched into a memory-mapped register that the processor can read. This register is the X-register at the base address of each channel. The addresses are shown in tables below, where they are used.

Note: The MLDT feedback uses the same circuitry that would be used for quadrature encoder feedback on that channel, so you cannot simultaneously connect an encoder and MLDT to the same channel's feedback on Turbo PMAC. In this mode, it is the pulse *timer* that is used as a position measurement for feedback, not the pulse *counter* that is used with encoders. The counter still registers the number of pulses returned, but does not represent a position measurement here.

Conversion Table Processing Setup – MACRO Station Interface

If the MLDT is connected to the MACRO Station, the data must be processed through conversion tables on both the Station and the Turbo PMAC (although the table on the Turbo PMAC really just copies the data off the ring). The conversion table on the MACRO Station is contained in setup variables MI120 – MI151. Each entry to process an MLDT timer takes three consecutive of these variables ("lines" of the table).

Station Entry First Line: Method and Address

These timer registers are processed as "parallel feedback" just as an absolute encoder would be. We use the $3 format – parallel data from "Y" register, with filtering. The following table shows the values for this first line of the entry for each channel on a MACRO Station.

Machine Interface Channel	Entry for Stack Board	Entry for Backplane Board	Machine Interface Channel	Entry for Stack Board	Entry for Backplane Board
1	$30C000	$30C040	6	$30C028	$30C068
2	$30C008	$30C048	7	$30C030	$30C070
3	$30C010	$30C050	8	$30C038	$30C078
4	$30C018	$30C058	9	$30C090	--
5	$30C020	$30C060	10	$30C098	--

Station Entry Second Line: Bits-Used Mask

The second line is a 24-bit value showing which bits of the source register are to be used. Each bit used is set to 1. Set this line to $07FFFF to use the low 19 bits of the timer register. Note that this does not necessarily limit the absolute position range to 19 bits (2^{19} LSBs, or 524,288 LSBs); the full range is set by the number of bits read by Ixx10 and Ixx95 at power-up/reset; this can be up to 23 bits for a range of 2^{23} LSBs, or 8,388,608 LSBs (see below).

Station Entry Third Line: Maximum Change

The third line of the conversion table entry is the "Maximum Change" filter value. This value, expressed in least-significant bits per servo interrupt, specifies the maximum change in the source data that will be accepted as real. This is an important protection against extra or missing return pulses. It should be set to a value slightly greater than the maximum real velocity expected in the application, expressed in LSBs of the timer per phase interrupt.

Motor Node Feedback Address: MI10x

To pass the resulting value in the conversion table back to the Turbo PMAC as the position feedback value for the "xth" motor on the Station, variable MI10x on the Station must be set to the address of the result. The following table shows the results

Last Entry Variable	Result Address	Last Entry Variable	Result Address	Last Entry Variable	Result Address	Last Entry Variable	Result Address
MI120	$0010	MI128	$0018	MI136	$0020	MI144	$0028
MI121	$0011	MI129	$0019	MI137	$0021	MI145	$0029
MI122	$0012	MI130	$001A	MI138	$0022	MI146	$002A
MI123	$0013	MI131	$001B	MI139	$0023	MI147	$002B
MI124	$0014	MI132	$001C	MI140	$0024	MI148	$002C
MI125	$0015	MI133	$001D	MI141	$0025	MI149	$002D
MI126	$0016	MI134	$001E	MI142	$0026	MI150	$002E
MI127	$0017	MI135	$001F	MI143	$0027	MI151	$002F

With proper ring setup, this causes the data to be sent back to the Turbo PMAC, where the PMAC's own conversion table can copy the value into internal memory.

Turbo PMAC Entry First Line: Method and Address

These MACRO-node registers are processed as "parallel feedback" just as an absolute encoder would be. However, filtering is not required, so the $2 format can be used. Because the data has already been shifted to contain 5 bits of fraction, bit 19 is set to 1 to disable another shift. The following table shows the possible entry first lines.

Entries for MACRO Node Register 0 Position Feedback (Y-registers)

IC# - Node#	0 - 0	0 - 1	0 - 4	0 - 5	0 - 8	0 - 9	0 - 12	0 - 13
Entry 1st Line	**$2F8420**	**$2F8424**	**$2F8428**	**$2F842C**	**$2F8430**	**$2F8434**	**$2F8438**	**$2F843C**
IC# - Node#	1 - 0	1 - 1	1 - 4	1 - 5	1 - 8	1 - 9	1 - 12	1 - 13
Entry 1st Line	**$2F9420**	**$2F9424**	**$2F9428**	**$2F942C**	**$2F9430**	**$2F9434**	**$2F9438**	**$2F943C**
IC# - Node#	2 - 0	2 - 1	2 - 4	2 - 5	2 - 8	2 - 9	2 - 12	2 - 13
Entry 1st Line	**$2FA420**	**$2FA424**	**$2FA428**	**$2FA42C**	**$2FA430**	**$2FA434**	**$2FA438**	**$2FA43C**
IC# - Node#	3 - 0	3 - 1	3 - 4	3 - 5	3 - 8	3 - 9	3 - 12	3 - 13
Entry 1st Line	**$2FB420**	**$2FB424**	**$2FB428**	**$2FB42C**	**$2FB430**	**$2FB434**	**$2FB438**	**$2FB43C**

Turbo PMAC Entry Second Line: Bit Width and Offset

The second line of the conversion table entry determines the "bit width" and the "offset" of the source data. This should be set to $018000, specifying use of all 24 bits of the timer (bit width of $018, or 24 decimal, offset of $000, or 0 decimal).

Table Result Register

The result of this conversion is in the X-register of the last (second) line of the entry. Any function using this value should address this register. For example, if this were the first entry in the table, which starts at $003501, the result would be in X:$003502.

Example

To process MLDTs brought into the first two channels of a MACRO Station's first backplane axis-interface board (Servo IC 2 Channels 1 and 2 only), with a maximum change of 16 LSBs per phase interrupt, the following MI-variable and I-variable values could be set:

```
MS0,MI120=$30C040 ; Parallel filtered read of IC 0 Chan 1 timer
MS0,MI121=$07FFFF ; Use 19 bits starting at bit 0
MS0,MI122=16      ; Max change of 16 LSBs per phase
MS0,MI123=$30C048 ; Parallel filtered read of IC 0 Chan 2 timer
MS0,MI124=$007FFF ; Use 19 bits starting at bit 0
MS0,MI125=16      ; Max change of 16 LSBs per phase

MS0,MI101=$0012   ; Use first result for first motor feedback
MS0,MI102=$0015   ; Use second result for second motor feedback

I8000=$2F8420     ; Parallel unfiltered read of Node 0 Register 0
I8001=$018000     ; Read 24 bits starting at bit 0
I8002=$2F8424     ; Parallel unfiltered read of Node 1 Register 0
I8003=$018000     ; Read 24 bits starting at bit 0
```

The resulting values from these conversions are in the following registers:

Channel 1 -- X:$003502 (Address of I8001)
Channel 2 -- X:$003504 (Address of I8003)

Motor variables Ixx03 and Ixx04 should contain the addresses of these resulting values to use the MLDT for position- and velocity-loop feedback, respectively.

Conversion Table Processing Setup – Turbo PMAC Interface

If the MLDT is connected directly to the Turbo PMAC, the timer value must be processed through the Turbo PMAC's conversion table to be used for servo feedback.

Turbo PMAC Entry First Line: Method and Address

These timer registers are processed as "parallel feedback" just as an absolute encoder would be. Using the Executive program Encoder Table Configure window, you select the "Parallel Y Data with Filtering" conversion method. If writing directly to the table, this is the $3 format. The following table shows the values for this first line of the entry for each channel on a Turbo PMAC2.

Entries for PMAC2-Style IC Timer Registers (Y-registers)

IC# - Chan#	0 - 1	0 - 2	0 - 3	0 - 4	1 - 1	1 - 2	1 - 3	1 - 4
Entry 1st line	**$378000**	**$378008**	**$378010**	**$378018**	**$378100**	**$378108**	**$378110**	**$378118**
IC# - Chan#	2 - 1	2 - 2	2 - 3	2 - 4	3 - 1	3 - 2	3 - 3	3 - 4
Entry 1st line	**$378200**	**$378208**	**$378210**	**$378218**	**$378300**	**$378308**	**$378310**	**$378318**
IC# - Chan#	4 - 1	4 - 2	4 - 3	4 - 4	5 - 1	5 - 2	5 - 3	5 - 4
Entry 1st line	**$379200**	**$379208**	**$379210**	**$379218**	**$379300**	**$379308**	**$379310**	**$379318**
IC# - Chan#	6 - 1	6 - 2	6 - 3	6 - 4	7 - 1	7 - 2	7 - 3	7 - 4
Entry 1st line	**$37A200**	**$37A208**	**$37A210**	**$37A218**	**$37A300**	**$37A308**	**$37A310**	**$37A318**
IC# - Chan#	8 - 1	8 - 2	8 - 3	8 - 4	9 - 1	9 - 2	9 - 3	9 - 4
Entry 1st line	**$37B200**	**$37B208**	**$37B210**	**$37B218**	**$37B300**	**$37B308**	**$37B310**	**$37B318**
IC# - Chan#	M0 – 1*	M0 – 2*						
Entry 1st line	**$378410**	**$378418**						

Turbo PMAC Entry Second Line: Bit Width and Offset

The second line of the conversion table entry determines the "bit width" and the "offset" of the source data. This should be set to $013000, specifying use of the low 19 bits of the timer (bit width of $013, or 19 decimal, offset of $000, or 0 decimal). Note that this does not necessarily limit the absolute position range to 19 bits (2^{19} LSBs, or 524,288 LSBs); the full range is set by the number of bits read by Ixx10 and Ixx95 at power-up/reset; this can be up to 23 bits for a range of 2^{23} LSBs, or 8,388,608 LSBs (see below).

Turbo PMAC Entry Third Line: Maximum Change

The third line of the conversion table entry is the "Maximum Change" filter value. This value, expressed in least-significant bits per servo interrupt, specifies the maximum change in the source data that will be accepted as real. This is an important protection against extra or missing return pulses. It should be set to a value slightly greater than the maximum real velocity expected in the application, expressed in LSBs of the timer per servo interrupt.

Table Result Register

The result of this conversion is in the X-register of the last (third) line of the entry. Any function using this value should address this register. For example, if this were the first entry in the table, which starts at $003501, the result would be in X:$003503.

Example

To set up the encoder conversion table to process MLDTs on Servo IC 0 Channels 1 to 4 only, with a maximum change of 32 LSBs per servo interrupt, the following I-variable values could be set:

```
I8000=$378000     ; Parallel filtered read of IC 0 Chan 1 timer
I8001=$013000     ; Use 19 bits starting at bit 0
I8002=32          ; Max change of 32 LSBs per servo
I8003=$378008     ; Parallel filtered read of IC 0 Chan 2 timer
I8004=$013000     ; Use 19 bits starting at bit 0
I8005=32          ; Max change of 32 LSBs per servo
I8006=$378010     ; Parallel filtered read of IC 0 Chan 1 timer
I8007=$013000     ; Use 19 bits starting at bit 0
```

```
I8008=32          ; Max change of 32 LSBs per servo
I8009=$378018     ; Parallel filtered read of IC 0 Chan 2 timer
I8010=$013000     ; Use 19 bits starting at bit 0
I8011=32          ; Max change of 32 LSBs per servo
```

The resulting values from these conversions are in the following registers:

Channel 1 -- X:$003503 (Address of I8002)
Channel 2 -- X:$003506 (Address of I8005)
Channel 3 -- X:$003509 (Address of I8008)
Channel 4 -- X:$00350C (Address of I8011)

Motor variables Ixx03 and Ixx04 should contain the addresses of these resulting values to use the MLDT for position- and velocity-loop feedback, respectively.

Setting Up for Power-On Absolute Position

Because MLDTs are absolute position sensors, no homing search move is required if they are used for position feedback. Turbo PMAC variables Ixx10 and Ixx95 can be used to establish this absolute position.

Absolute Power-On Position Address: Ixx10

Ixx10 tells Turbo PMAC what register to read for power-on absolute position for Motor xx. For MLDT feedback, the timer register for the interface channel is used for this, just as it is used for ongoing feedback. The following table shows Ixx10 values for these registers:

Ixx10 for PMAC2-Style IC Timer Registers (Y-registers)

IC# - Chan#	0 - 1	0 - 2	0 - 3	0 - 4	1 - 1	1 - 2	1 - 3	1 - 4
Ixx10	**$078000**	**$078008**	**$078010**	**$078018**	**$078100**	**$078108**	**$078110**	**$078118**
IC# - Chan#	2 - 1	2 - 2	2 - 3	2 - 4	3 - 1	3 - 2	3 - 3	3 - 4
Ixx10	**$078200**	**$078208**	**$078210**	**$078218**	**$078300**	**$078308**	**$078310**	**$078318**
IC# - Chan#	4 - 1	4 - 2	4 - 3	4 - 4	5 - 1	5 - 2	5 - 3	5 - 4
Ixx10	**$079200**	**$079208**	**$079210**	**$079218**	**$079300**	**$079308**	**$079310**	**$079318**
IC# - Chan#	6 - 1	6 - 2	6 - 3	6 - 4	7 - 1	7 - 2	7 - 3	7 - 4
Ixx10	**$07A200**	**$07A208**	**$07A210**	**$07A218**	**$07A300**	**$07A308**	**$07A310**	**$07A318**
IC# - Chan#	8 - 1	8 - 2	8 - 3	8 - 4	9 - 1	9 - 2	9 - 3	9 - 4
Ixx10	**$07B200**	**$07B208**	**$07B210**	**$07B218**	**$07B300**	**$07B308**	**$07B310**	**$07B318**
IC# - Chan#	M0 – 1*	M0 – 2*						
Ixx10	**$078410**	**$078418**						

If the interface to the MLDT is on a MACRO Station, Ixx10 on the Turbo PMAC just contains the number of the controller's MACRO node used for interface. The following table shows the possible settings:

Ixx10 for MACRO IC Servo Nodes (Y-registers)

IC# - Node#	0 - 0	0 - 1	0 - 4	0 - 5	0 - 8	0 - 9	0 - 12	0 - 13
Ixx10	**$000100**	**$000001**	**$000004**	**$000005**	**$000008**	**$000009**	**$00000C**	**$00000D**
IC# - Node#	1 - 0	1 - 1	1 - 4	1 - 5	1 - 8	1 - 9	1 - 12	1 - 13
Ixx10	**$000010**	**$000011**	**$000014**	**$000015**	**$000018**	**$000019**	**$00001C**	**$00001D**
IC# - Node#	2 - 0	2 - 1	2 - 4	2 - 5	2 - 8	2 - 9	2 - 12	2 - 13
Ixx10	**$000020**	**$000021**	**$000024**	**$000025**	**$000028**	**$000029**	**$00002C**	**$00002D**
IC# - Node#	3 - 0	3 - 1	3 - 4	3 - 5	3 - 8	3 - 9	3 - 12	3 - 13
Ixx10	**$000030**	**$000031**	**$000034**	**$000035**	**$000038**	**$000039**	**$00003C**	**$00003D**

Note that for IC 0 Node 0, Ixx10 is not set to $000000; this would cause no absolute position to be read. Instead, it is set to $000100 to permit the absolute read through this node.

Absolute Power-On Position Format: Ixx95

Ixx95 tells Turbo PMAC how to interpret the data in the register specified in Ixx10. To read the MLDT timer directly on a Turbo PMAC, a 23-bit parallel-read format is used. To specify this, set Ixx95 to $170000 ($17 is 23 decimal).

To specify this when the MLDT interface is on a MACRO Station, set Ixx95 to $740000, and set MI11x on the MACRO Station to $17xxxx, where "xxxx" represents the Y-address of the timer register on the MACRO Station (e.g. $C000, C008, etc.)

Position Reference Offset: Ixx26

When used with an absolute position read, Ixx26 specifies the difference between the sensor's zero position and the motor zero position. With an MLDT, the sensor's zero position is at the transmitter/receiver module, outside the possible range of motion of the sensor. If it is desired that the motor's zero position be within the range of travel, Ixx26 should be used to define the offset between sensor and motor zero. Ixx26 has units of 1/16 count, where a count here is an LSB of the timer.

Power-On Mode: Ixx80

Ixx80 controls whether the absolute position read for Motor xx is performed immediately at power-on/reset, or whether the absolute position read is delayed until a **$*** or **$$*** command is issued. Because the output frequency for the MLDT is not established yet when a read would be done immediately at power-on, it should be delayed with an MLDT. This requires that Bit 2 of Ixx80 be set to 1, making the value of Ixx80 equal to 4 if the motor is not to be enabled immediately on power-on/reset, or equal to 5 if the motor is to be enabled immediately.

Scaling the Feedback Units

Motor Units

For a motor set up with MLDT feedback, a "count" is one increment (LSB) of the timer. The physical length of this increment, which is the resolution of the measurement, is dependent on the speed of the return pulse in the MLDT, and the frequency of the timer in the Turbo PMAC. The speed of the return pulse (the speed of sound in the metal) varies from device to device, but is

always approximately the inverse of 0.35 μsec/mm, or 9.0 μsec/inch. The frequency of the timer is 117.96 MHz. The resolution can be calculated as:

$$\mathrm{Re}\,solution\left(\frac{length}{increment}\right) = \frac{1}{TimerFreq}\left(\frac{\mu\sec}{increment}\right) * \mathrm{Re}\,turnSpeed\left(\frac{length}{\mu\sec}\right)$$

$$\cong \frac{1}{117.96}\frac{\mu\sec}{increment} * \frac{1}{0.35}\frac{mm}{\mu\sec} \cong 0.024\frac{mm}{increment}\left(41.3\frac{increments}{mm}\right)$$

$$\cong \frac{1}{117.96}\frac{\mu\sec}{increment} * \frac{1}{9.0}\frac{inches}{\mu\sec} \cong 0.00094\frac{inches}{increment}\left(1060\frac{increments}{inch}\right)$$

Axis User Units

Typically the axis assigned to the motor using the MLDT for feedback will be given engineering units of millimeters or inches. The scaling is done in the axis definition statement. With the above example, an axis definition statement to create axis user units of millimeters would be:

```
#1->41.3X
```

An axis definition statement to create axis user units of inches would be:

```
#1->1060X
```

Turbo PMAC General Purpose I/O Use

Turbo PMAC(1) General-Purpose I/O (JOPTO) Port

The JOPTO port on aTurbo PMAC(1) (J5 on Turbo PMAC-PC, -PCI, and -VME) provides eight general-purpose digital inputs and eight general-purpose digital outputs. Each input and each output has its own corresponding ground pin in the opposite row. The 34-pin connector was designed for easy interface to OPTO-22 or equivalent optically isolated I/O modules. Delta Tau's Accessory 21F is a six-foot cable for this purpose. Note that these I/O points are not optically isolated on the Turbo PMAC itself.

Hardware Characteristics

A Turbo PMAC(1) is shipped standard with a ULN2803A sinking (open-collector) output IC for the eight outputs. These outputs can sink up to 100 mA each, but must have a pull-up resistor to go high.
Do not connect these outputs directly to the supply voltage, or damage to the Turbo PMAC will result from excessive current draw.

The user can provide a high-side voltage (+5 to +24V) into Pin 33 of the JOPTO connector, and allow this to pull up the outputs by connecting pins 1 and 2 of Jumper E1. Jumper E2 must also connect pins 1 and 2 for a ULN2803A sinking output.

Warning: Having Jumpers E1 and E2 set wrong can damage the IC.

It is possible for these outputs to be sourcing drivers by substituting a UDN2981A IC for the ULN2803A. This IC (U3 on the Turbo PMAC-PC, U26 on the Turbo PMAC-Lite, U33 on the Turbo PMAC-VME) is socketed, and so may easily be replaced. For this driver, pull-down resistors should be used. With a UDN2981A driver IC, Jumper E1 must connect pins 2 and 3, and Jumper E2 must connect pins 2 and 3.

Warning: Having Jumpers E1 and E2 set wrong can damage the IC.

Jumper E7 controls the configuration of the eight inputs. If it connects pins 1 and 2 (the default setting), the inputs are biased to high-side voltage (+5V to +24V) for the "OFF" state, and they must be pulled low for the "ON" state. If E7 connects pins 2 and 3, the inputs are biased to ground for the "OFF" state, and must be pulled high for the "ON" state. In either case, a high voltage is interpreted as a “0” by the Turbo PMAC software, and a low voltage is interpreted as a “1”.

Software Access

These inputs and outputs are typically accessed in software through the use of M-variables. In the suggested set of M-variable definitions, variables M1 through M8 are used to access outputs 1 through 8, respectively, and M11 through M18 to access inputs 1 through 8, respectively. This port maps into Turbo PMAC’s memory space at Y-address $078802.

Turbo PMAC(1) Multiplexed I/O (JTHW) Port

The Multiplexer port on the JTHW (J3) connector of a Turbo PMAC(1) has eight input lines and eight output lines. The output lines can be used to multiplex large numbers of inputs and outputs on the port, and Delta Tau provides accessory boards and software structures (special M-variable definitions TWB, TWD, TWR, and TWS) to capitalize on this feature. Up to 32 of the multiplexed I/O boards may be daisy-chained on the port, in any combination.

Alternately, the 8 inputs and 8 outputs on the port can be used directly by assigning M-variables to the I/O points themselves. This port maps into Turbo PMAC's memory space at Y-address $078801. The suggested M-variable definitions for this use are M40 to M47 for the 8 outputs, and M50 to M57 for the 8 inputs.

Turbo PMAC(1) Control Panel Port

The control-panel port on the JPAN connector (J2 on Turbo PMAC-PC, -PCI, -VME) of a Turbo PMAC(1) is a 26-pin connector with dedicated control inputs, dedicated indicator outputs, a quadrature encoder input, and an analog input. The control inputs are active low in their automatic function with internal pull-up resistors. They have predefined functions unless the control-panel-disable I-variable (I2) has been set to 1. If this is the case, they may be used as general-purpose inputs by assigning M-variable to their corresponding memory-map locations (bits of Y address $078800).

Control-Panel Inputs

The JOG-/, JOG+/, PREJ/ (return to pre-jog position), and HOME/ inputs affect the motor selected by the FDPn/ lines (see below). The STRT/ (run), STEP/, STOP/ (abort), and HOLD/ (feed hold) inputs affect the coordinate system selected by the FDPn/ lines.

The four low-true BCD-coded input lines FDP0/ (LSBit), FDP1/, FDP2/, and FDP3/ (MSBit) form a low-true BCD-coded nibble that selects the active motor and coordinate system (simultaneously). These are usually controlled from a single 4-bit motor/coordinate-system selector switch. Variable I59 "bank selects" which group of 8 motors and coordinate systems can be specified by the switch. The motor selected with these input lines will respond to the motor-specific inputs. It will also have its position following function turned on (*Ixx06 bit 0 is automatically set to 1!*); the motor just de-selected has its position following function turned off (*Ixx06 bit 0 is automatically set to 0*).

It is not a good idea to change the selector inputs while holding one of the jog inputs low. Releasing the jog input will then *not* stop the previously selected motor. This can lead to a dangerous situation.

Control-Panel Outputs

There are five dedicated low-true outputs on the JPAN connector, usually used to light LEDs. They are BRLD/ (buffer-request LED), IPLD/ (in-position LED), EROR/ (fatal following error LED), F1LD/ (1st -- warning -- following error LED), and F2LD/ (which goes true when the watchdog timer trips). BRLD/, ERLD/, and F2LD/ are global status lines. IPLD/ and F1LD/ are coordinate-system specific status lines. If I2=0, they refer to the panel-selected coordinate system (by FDPn/ and I59). If I1=1, they refer to the host-selected coordinate system (&n).

If I2=0 but no coordinate system is selected (all FPDn/ inputs are floating or pulled high), these lines can be used as general purpose outputs, addressed as bits 20-23 of Y:$078802.

Turbo PMAC2 General-Purpose I/O (JIO) Port

The JIO port on a Turbo PMAC2 (on ACC-5E for a UMAC Turbo) has 32 discrete digital I/O lines for general-purpose use. The lines are configurable by byte for input or output (on the DSPGATE2 I/O IC, the lines are individually configurable for input or output, but the buffer ICs are only byte-configurable), and individually configurable for inverting or non-inverting format.

Hardware Characteristics

When configured as an output, each line has a 5V CMOS totem-pole driver. This driver can sink or source up to 20 mA. There is a 10 kΩ pull-up resistor to 5V on each line for input purposes, but the driver IC can hold the line high or low despite this resistor. When configured as an input, the buffer IC presents a high-impedance input either sinking or sourcing; no significant current will flow. The pull-up resistor on the line will bias the line high in the absence of anything actively pulling the line low at significantly lower impedance.

Note: Because all of these lines default to inputs at power-up/reset, any lines to be used as outputs will pull to +5V at power-up/reset until software configures them as outputs.

Suggested M-Variables

Note: In a UMAC Turbo system, it is possible to have the DSPGATE2 IC driving the JIO port at a base address other than the standard $078400. However, this is unlikely, so the following discussion assumes the standard base address of $078400 for this IC.

The 32 I/O lines are memory-mapped into PMAC's address space in registers Y:$078400 and Y:$078401. Typically these I/O lines are accessed individually with M-variables. A complete list of the suggested M-variables is shown in the Software Reference; a few are shown here:

```
M0->Y:$078400,0          ; I/O00 Data Line; J3 Pin 1
M1->Y:$078400,1          ; I/O01 Data Line; J3 Pin 2
…
M23->Y:$078400,23        ; I/O23 Data Line; J3 Pin 24
M24->Y:$078401,0         ; I/O24 Data Line; J3 Pin 25
M25->Y:$078401,1         ; I/O25 Data Line; J3 Pin 26
…
M31->Y:$078401,7         ; I/O31 Data Line; J3 Pin 32
```

Direction Control

The direction control bit for each of these I/O bits is in the corresponding bit in the matching X register. For example, the direction control bit for I/O03 is located at X:$078400,3; the direction control bit for I/O30 is located at X:$078401,6. Because the buffer ICs can only be switched by byte, it is best to define 8-bit M-variables for the direction control. Suggested definitions are:

```
M32->X:$078400,0,8            ; Direction control for I/O00 to I/O07
```

```
M34->X:$078400,8,8            ; Direction control for I/O08 to I/O15
M36->X:$078400,16,8           ; Direction control for I/O16 to I/O23
M38->X:$078401,0,8            ; Direction control for I/O24 to I/O31
```

These M-variables should take values of 0 or 255 ($FF) only; 0 sets the byte to input, 255 sets the byte to output.

In addition, the bi-directional buffer IC for each byte has a direction control line accessible as a software control bit. These control lines and bits must match the ASIC direction bits. The buffer direction control bits are at PMAC address Y:$070800, with bits 0 to 3 controlling the four bytes of the JIO port. A bit value of 0 specifies input; 1 specifies output. Suggested M-variable definitions are:

```
M33->Y:$070800,0        ; Buffer direction control for I/O00 to I/O07
M35->Y:$070800,1        ; Buffer direction control for I/O08 to I/O15
M37->Y:$070800,2        ; Buffer direction control for I/O16 to I/O23
M39->Y:$070800,3        ; Buffer direction control for I/O24 to I/O31
```

In the default configuration automatically set at power-up/reset, I/O00 to I/O31 are set up as inputs (M32 through M39 = 0). This is done for maximum safety; no lines can be forced into an undesirable high or low state. Any of these lines that are to be used as outputs must be changed to outputs by user programs (usually this is done in PLC 1 acting as a "reset" PLC, scanning through once on power-up/reset, then disabling itself).

Inversion Control

Each line on the JIO port is individually controllable as to whether it is an inverting I/O point (0=+5V; 1=0V) or a non-inverting I/O point (0=0V; 1=+5V). Registers X:$078404 and X:$078405 contain the inversion control bits:

X:$078404 bits 0 to 23 control I/O00 to I/O23, respectively
X:$078405 bits 0 to 7 control I/O24 to I/O31, respectively

A value of 0 in the control bit sets the corresponding I/O point as non-inverting. A value of 1 in the control bits sets the corresponding I/O point as inverting. At power-up/reset, PMAC automatically sets all of the I/O points on the JIO port as non-inverting.

Alternate Uses

Each general-purpose I/O point on the JIO port has an alternate use as a supplemental fixed-use I/O point on a supplemental machine interface channel (1* or 2*). The points are individually controllable as to general-purpose use or fixed use by control registers Y:$078404 and Y:$078405. Refer to these registers in the memory-I/O map to see the alternate uses of each point. At power-up/reset, Turbo PMAC automatically sets up all of the I/O points on the port for general-purpose use.

Note: The direction-control of the buffer ICs must be set properly for the alternate uses of the I/O points, just as for the general-purpose I/O uses.

Turbo PMAC2 Multiplexed I/O Port (JTHW)

The JTHW multiplexer port has 16 discrete digital I/O lines for general-purpose use. Most people will use them in the default configuration of 8 inputs and 8 outputs, and to support multiplexed I/O accessories from Delta Tau. When used in this manner, no special setup of the I/O points is required. However, if it is desired to use these I/O points directly, the following discussion explains their use.

The lines are configurable by byte for input or output (on the DSPGATE2 I/O IC, the lines are individually configurable for input or output, but the buffer ICs are only byte-configurable), and individually configurable for inverting or non-inverting format.

Note: Variable I20 for a Turbo PMAC2 specifies the base address for the first "MACRO IC" in the system, the DSPGATE2 IC that controls the JTHW multiplexer port. This variable must be set correctly for the automatic multiplexer port functions to work correctly. This base address is almost always $078400, and the following discussions assume that the system is set up this way.

Hardware Characteristics

When configured as an output, each line has a 5V CMOS totem-pole driver. This driver can sink or source up to 20 mA. There is a 10 kΩ pull-up resistor to 5V on each line for input purposes, but the driver IC can hold the line high or low despite this resistor. When configured as an input, the buffer IC presents a high-impedance input either sinking or sourcing; no significant current will flow. The pull-up resistor on the line will bias the line high in the absence of anything actively pulling the line low at significantly lower impedance.

Suggested M-Variables

The 16 I/O lines are memory-mapped into PMAC's address space in register Y:$078402. These lines are typically used as a unit with specially designed multiplexing I/O accessories and appropriate multiplexing M-variables (TWB, TWD, TWR, and TWS formats), in which case Turbo PMAC handles the direct control of these I/O lines automatically. However, these lines can also be accessed individually with M-variables. The complete list of M-variables is shown in the Software Reference; a few are shown here:

```
M40->Y:$078402,8        ; SEL0 Line; J2 Pin 4
…
M47->Y:$078402,15       ; SEL7 Line; J2 Pin 18
M48->Y:$078402,8,8,U    ; SEL0-7 Lines treated as a byte
M50->Y:$078402,0        ; DAT0 Line; J2 Pin 3
…
M57->Y:$078402,7        ; DAT7 Line; J2 Pin 17
M58->Y:$078402,0,8,U    ; DAT0-7 Lines treated as a byte
```

Direction Control

In the default configuration automatically set at power-up/reset, DAT0 to DAT7 are set up as non-inverting inputs; SEL0 to SEL7 are set up as non-inverting outputs with a zero (low voltage) value. If any of the multiplexer port accessories are to be used, this configuration must not be changed.

The direction control bit for each of these I/O bits is in the corresponding bit in the matching X register. For example, the direction control bit for DAT3 is located at X:$078402,3; the direction control bit for SEL6 is located at X:$078402,14. Because the buffer ICs can only be switched by byte, it is best to define 8-bit M-variables for the direction control. Suggested definitions are:

```
M60->X:$078402,0,8            ; Direction control for DAT0 to DAT7
M62->X:$078402,8,8            ; Direction control for SEL0 to SEL7
```

These M-variables should take values of 0 or 255 ($FF) only; 0 sets the byte to input, 255 sets the byte to output.

In addition, the bi-directional buffer IC for each byte has a direction control line accessible as a software control bit. These control lines and bits must match the ASIC direction bits. In the PC-bus versions of the Turbo PMAC, the buffer direction control bits are at Turbo PMAC address Y:$070800, with bits 4 and 5 controlling the two bytes of the JTHW port. A bit value of 0 specifies input; 1 specifies output. Suggested M-variable definitions are:

```
M61->Y:$070800,4        ; Buffer direction control for DAT0 to DAT7
M63->Y:$070800,5        ; Buffer direction control for SEL0 to SEL7
```

In the VME-bus versions of the Turbo PMAC, the buffer direction control bits are at Turbo PMAC address Y:$070802, with bits 0 and 1 controlling the two bytes of the JTHW port. A bit value of 0 specifies input; 1 specifies output. Suggested M-variable definitions are:

```
M61->Y:$070802,0        ; Buffer direction control for DAT0 to DAT7
M63->Y:$070802,1        ; Buffer direction control for SEL0 to SEL7
```

If it is desired to change either of these I/O bytes, it must be done by user programs (usually this is done in PLC 1 acting as a “reset” PLC, scanning through once on power-up/reset, then disabling itself).

Inversion Control

Each line on the JTHW port is individually controllable as to whether it is an inverting I/O point (0=+5V; 1=0V) or a non-inverting I/O point (0=0V; 1=+5V). Register X:$078406 contains the inversion control bits:

X:$078406 bits 0 to 7 control DAT0 to DAT7, respectively
X:$078406 bits 8 to 15 control SEL0 to SEL7, respectively

A value of 0 in the control bit sets the corresponding I/O point as non-inverting. A value of 1 in the control bits sets the corresponding I/O point as inverting. At power-up/reset, PMAC automatically sets all of the I/O points on the JTHW port as non-inverting. To use any of the multiplexed I/O accessory boards on the JTHW port, all I/O points on the port must be left non-inverting.

Alternate Uses

Each general-purpose I/O point on the JTHW port has an alternate use as a supplemental fixed-use I/O point on a supplemental machine interface channel (1* or 2*). The points are individually controllable as to general-purpose use or fixed use by control register Y:$078406.

Refer to this register in the memory-I/O map to see the alternate uses of each point. At power-up/reset, Turbo PMAC automatically sets up all of the I/O points on the port for general purpose use.

Note: Because of the byte-wide direction-control buffer ICs, it is not possible to use all of the I/O points on the JTHW in their alternate uses.

Turbo PMAC2 Analog Input (JANA) Port

The analog input (JANA) port is present only if Option 12 is ordered for the Turbo PMAC2. Option 12 provides 8 12-bit analog inputs (ANAI00-ANAI07). Option 12A provides 8 additional 12-bit analog inputs (ANA08-ANAI15) for a total of 16 inputs.

Hardware Characteristics

The analog inputs can be used as unipolar inputs in the 0V to +5V range, or bipolar inputs in the -2.5V to +2.5V range. Each input has a 470Ω input resistor in-line, and a 0.033 μF resistor to ground. This provides a 16 μsec time constant on each input line.

The analog-to-digital converters on Turbo PMAC require +5V and -12V supplies. These supplies are not isolated from digital +5V circuitry on PMAC. If the Turbo PMAC2 is plugged into the bus (ISA, PCI, or VME), these supplies are taken from the bus power supply. In a standalone application, these supplies must be brought in on terminal block TB1.

The -12V and matching +12V supply voltages are available on the J1 connector to supply the analog circuitry providing the signals. The +12V supply is not used by Turbo PMAC; it is merely passed through to the J1 connector for convenience. If use of this supply is desired, it must either come from the bus supply through Turbo PMAC's bus connector, or from TB1.

Multiplexing Principle

Only one pair of analog-to-digital converter registers is available to the Turbo PMAC processor at any given time. The data appears to the processor at address Y:$078800. The data from the selected analog input 0 to 7 (ANAI00-ANAI07) appears in the low 12 bits; the data from the selected analog input 8 to 15 (ANAI08-ANAI15) appears in the high 12 bits (this data is only present if Option 12A has been ordered).

The input is selected and the conversion is started by writing to this same word address Y:$078800. A value of 0 to 7 written into the low 12 bits selects the analog input channel of that number (ANAI00-ANAI07) to be converted in unipolar mode (0V to +5V). A value of 0 to 7 written into the high 12 bits selects the analog input channel numbered 8 greater (ANAI08-ANAI15) in unipolar mode. If the value written into either the low 12 bits or the high 12 bits is 8 higher (8 to 15), the same input channel is selected, but the conversion is in bipolar mode (-2.5V to +2.5V).

De-multiplexing I-Variables

Turbo PMAC I-variables I5060 – I5096 permit the automatic "de-multiplexing" of these multiplexed A/D converters (and of multiplexed A/D converters on external ACC-36 and ACC-59 boards as well).

I5060 controls the number of A/D converter pairs accessed in the de-multiplexing "ring", up to 16 pairs. Variables starting at I5061, and possibly up to I5076, specify the Turbo PMAC address of each ADC pair to be read. The addresses of all 8 pairs on the JANA port are located at $078800.

Variables starting at I5081, and possibly up to I5096, specify which pair of ADCs at the address specified by the I-variable numbered 20 lower (e.g. I5081 for I5061) is read, and how it is to be converted. I5081 – I5096 are 24-bit values, represented by 6 hexadecimal digits. Legitimate values are of the format $00*m*00*n*, where *m* and *n* can take any hex value from 0 through F.

For the on-board Option 12 & 12A ADCs on a Turbo PMAC2, the *m* value determines which of the inputs ANAI08 to ANAI15 that come with Option 12A is to be read, and how it is to be converted, according to the following formulas:

m = *ANAI#*–8 ; 0 to +5V unipolar input
m = *ANAI#* ; -2.5V to +2.5V bipolar input

For the on-board Option 12 & 12A ADCs on a Turbo PMAC2, the *n* value determines which of the inputs ANAI00 to ANAI07 that come with Option 12A is to be read, and how it is to be converted, according to the following formulas:

n = *ANAI#* ; 0V to +5V unipolar input
n = *ANAI#*+8 ; -2.5V to +2.5V bipolar input

The results of this A/D de-multiplexing are placed in registers at addresses Y:$003400 to Y:$00341F, using bits 12 to 23 of these registers. The value of the A/D converter found in the low 12 bits of the source register is placed in the register with the even-numbered address; the value of the A/D converter found in the high 12 bits of the source register is placed in the register with the odd-numbered address. Refer to the Turbo PMAC memory map or I5061 – I5076 description for details. Suggested M-variables for the result registers are M5061 – M5076.

In operation, Turbo PMAC reads one ADC pair each phase cycle and copies it into the appropriate memory registers. Therefore, it reads each ADC pair every I5060 phase cycles. If these values are used a feedback for a servo loop, the loop should not be closed more often than the ADC is read.

Using the Position-Compare Feature on Turbo PMAC

The Turbo PMAC has powerful "position compare" functions built into its Servo ICs, particularly with the PMAC2-style "DSPGATE1" Servo ICs. The position-compare feature provides a very fast and accurate digital output based on the actual position counter -- when the hardware counter in the IC becomes equal to the "compare" register, the output toggles immediately. This output is often used to trigger a measurement device or to fire a laser.

Because the triggering of the compare output is a pure hardware function (although the setup is done in software), there are no software delays to affect the accuracy. The compare output is accurate to the exact count at any speed.

The position-compare function is implemented in software-configurable hardware registers in the Servo ICs. The software configuration gives the function its flexibility; the hardware circuitry gives the function its speed.

Where the position-compare outputs are brought out, and the nature of the driver circuit, is dependent on the individual hardware used. Consult the Hardware Reference manual for your product. In some cases, the driver IC is socketed, so you have a choice of circuits to use. These outputs, typically labeled "EQUn", can be used to drive external hardware, such as the triggers for scanning and measurement equipment.

Scaling and Offset of Position-Compare Registers

The position-compare registers are scaled in encoder counts. A count is defined by the channel Encoder Decode I-variable I7mn0, typically with 4 counts per cycle or line. The compare registers are always referenced to the position at the most recent power-up or reset, called "Encoder Zero". This reference does not change with motor homing, which changes "Motor Zero", or axis offsets, which change "Axis Zero". See the section *"Converting from Motor and Axis Coordinates",* below, for more details.

The position-compare registers, like the hardware encoder counters, are 24-bit values that can, and often do, "roll over" in the normal course of operation. The user must keep track of any rollover effects. Note, however, that by assigning a 24-bit M-variable to a compare register, the act of writing to this M-variable automatically truncates a longer value properly to this word length.

Setup On a PMAC(1)-Style Servo IC

Each encoder counter in the PMAC(1)-style "DSPGATE" Servo IC has a position-compare function. On the Turbo PMAC(1)-PC and the Turbo PMAC(1)-PCI, several of the position-compare outputs may be used to interrupt the host computer over the backplane bus.

The position-compare circuitry for each channel has a 24-bit compare register, three control bits, and one status bit. The compare register contains the position count value at which the output is to be toggled. Note that this is a write-only register; if you read the same address, you read the "captured position" value.

The control bits are:

1. Compare output-enable bit: This must be set to 1 to get the compare signal output from the IC. (The internal status bit always works).

2. Compare flag latch control bit: If this bit is set to 0, the output is "transparent" – it is on only for the single count when the position counter equals the value in the compare register. If this bit is set to 1, the output is latched – it turns on when the counter becomes equal to the compare value, and stays on until this latch bit is cleared. This is the more common mode of use.

3. Compare output invert control bit: If this bit is set to 0, the "off" state is a low voltage out of the Servo IC, and the "on" state is a high voltage. (Depending on the output driver circuitry, the actual output from the card may be inverted from this.) If this bit is set to 1, the "off" state is a high voltage and the "on" state is a low voltage. Note that this control bit can be used by itself to make the compare output a general-purpose digital output, or in some cases, a software-driven interrupt to the host computer.

The single status bit is the position-compare flag, which shows the present state of the compare output logic (and if the output is enabled, of the physical output). It is set to 1 if the compare output logic state is "on", and to a 0 if the output logic state is "off".

There are no firmware functions for the automatic use of the position-compare circuitry. The user's application software must deal with the compare registers and control/status bits directly. These are almost always accessed with M-variables, and the file of suggested M-variables includes these definitions. For the first channel of a Turbo PMAC(1) (Servo IC 0 Channel 1), these definitions are:

```
M101->X:$078001,0,24,S  ; 24-bit position counter register
M103->X:$078003,0,24,S  ; 24-bit position compare register
M111->X:$078000,11,1    ; Compare flag latch control
M112->X:$078000,12,1    ; Compare output enable control
M113->X:$078000,13,1    ; Compare output invert control
M116->X:$078000,16,1    ; Compare logic status
```

To preload a compare position, simply write a value to the compare register (e.g. **M103=1250**).

Example: We have an array of 50 points in P1 through P50 that represent the distances from a starting position at which we want compare outputs. Program code (running just a single time that could start this process is:

```
P100=1              ; Select first point of array
P101=M101           ; Read and store starting position
M103=P101+P(P100)   ; Write first relative position to compare
M111=1              ; Set for latched output
M112=1              ; Enable compare output
```

Repeatedly executing program code (probably in a PLC 0 or PLCC 0) that could work through the array is:

```
WHILE (P100<51)     ; Loop until 50 points completed
  IF (M116=1)       ; Reached last compare position?
    P100=P100+1     ; Select next point
    M103=P101+P(P100)   ; Write next relative position to compare
```

```
    M111=0          ; Clear output by making transparent
    M111=1          ; Set for latched output again
  ENDIF
ENDWHILE
```

Setup On a PMAC2-Style Servo IC

Each encoder counter in PMAC2-style Servo IC has a position compare function. Furthermore, the first encoder counter (Channel 1) in each Servo IC can use the position compare circuitry from any of the other channels on the ASIC, so it can utilize up to 4 independent compare circuits.

On the Turbo PMAC2-PC and the Turbo PMAC2-PCI, the compare outputs for Channel 1 and Channel 5 (if present) can also be used to interrupt the host computer over the backplane bus.

In addition, there is a memory-mapped status bit for the output that Turbo PMAC software can access with M-variables for its own use.

If the Servo IC is on a Turbo PMAC, or directly connected to it, the registers and control/status bits are accessed with user-defined M-variables. If the Servo IC is on a MACRO Station, these registers and bits are accessed with pre-defined Station node-specific MI-variables for the MACRO node matched to this interface channel.

The position compare circuitry for each channel is based on three memory-mapped registers:

- Compare A (Turbo suggested M-variable Mxx08; MACRO Station **MS{node},MI925**)
- Compare B (Turbo suggested M-variable Mxx09; MACRO Station **MS{node},MI924**)
- Compare Auto-Increment (Turbo suggested M-variable Mxx10; MACRO Station **MS{node},MI922**)

When the encoder counter value matches the value in either the channel's Compare A Register or Compare B Register, the compare output is toggled from the existing state; either from 0 to 1, or from 1 to 0. The toggling occurs on the transition from "equal" to "not-equal" in either direction. For instance, if the compare register contains 100, the toggling of the output would occur on the transition from 100 to 101 counts in the positive direction, or on the transition from 100 to 99 counts in the negative direction.

In addition, when the output is toggled by one of the compare registers, the other register is immediately incremented by the amount in the auto-increment register. If the output is toggled by moving in the positive direction, the value in the auto-increment register is added to the other compare register. If the output is toggled by moving in the negative direction, the value in the auto-increment register is subtracted from the other compare register.

If the auto-increment register is non-zero, all of the compare edges should be at least two counts apart. This means that the Compare A and Compare B registers should not be less than two counts apart, and the minimum non-zero value for the auto-increment register should be 4.

There are three control bits for each channel. They are:

1. *Compare Channel Select Bit*: (Turbo I-variable I7mn1; MACRO Station **MS{node},MI911**) This control bit determines whether the compare circuitry for the channel acts on the encoder for that channel, or the encoder for Channel 1 of this Servo IC. (Note that this bit does nothing for Channel 1.) This control bit has been assigned an I-variable – I7mn1 for Servo IC m Channel n. Note that when multiple compare circuits have been assigned to Channel 1 of a Servo IC, the compare output for the first channel is the logical OR of all of the compare logical outputs assigned to Channel 1.

2. *Compare Direct-Write (Initial State) Value*: (Turbo suggested M-variable Mxx12; MACRO Station **MS{node},MI929**) This control bit permits the user to set the state of the compare output for the channel directly, either as an initial state for a compare sequence, or to use the output for general-purpose use, or for a software-driven interrupt. A "1" here will force a high voltage on the output of the Servo IC; a "0" will force a low voltage. (Depending on the output driver used, this may be inverted on the output of the controller itself.) It is the user's responsibility to determine whether he is in his desired "0" region or the "1" region when he uses the direct-write feature. Writing to this bit alone does not set the output; this does not happen until the write is enabled, using the next control bit.

3. *Compare Direct-Write Enable*: (Turbo suggested M-variable Mxx11; MACRO Station **MS{node},MI928**) Writing a 1 to this bit forces the value of the direct-write bit onto the compare output line. As soon as the output state is set, this bit automatically sets itself back to 0.

The single status bit is the compare output status, which reflects the state of the output at any instant. A "1" here indicates a high voltage out of the Servo IC; a "0" indicates a low voltage. Depending on the output driver used in a particular configuration, the voltage sense may be inverted out of the control card.

There are no firmware functions for the automatic use of the position-compare circuitry. The user's application software must deal with the compare registers and control/status bits directly. These are almost always accessed with M-variables (or MACRO Station setup MI-variables), and the file of suggested M-variables includes these definitions. For the first channel of a Turbo PMAC2 (Servo IC 0 Channel 1), these definitions are:

```
M101->Y:$078001,0,24,S  ; Hardware position count value (counts)
M108->Y:$078007,0,24,S  ; Position compare A value (counts)
M109->X:$078007,0,24,S  ; Position compare B value (counts)
M110->X:$078006,0,24,S  ; Compare auto-increment Value (counts)
M111->X:$078005,11      ; Position compare write enable
M112->X:$078005,12      ; Position compare direct-write
                        ; (initial-state) value
M113->X:$078000,9       ; Position compare output status
```

Refer to the Suggested M-variable list in the Software Reference Manual for equivalent definitions for other channels.

Setting Up for a Single Pulse Output

If just a single compare pulse is desired (not using the auto-increment feature), the following steps should be taken:

- Write the encoder value at the front edge into the Compare A register

- Write the encoder value at the back edge into the Compare B register
- Set the Auto-Increment register to zero
- Set the initial state with the direct-write feature
    - Write a value to the initial state bit
    - Write a '1' to the direct-write enable bit (this is self-clearing to 0)
- Start the move that will cause the compare function

Example: With the axis sitting still at about encoder position 100, and a '1' value of position compare desired between encoder positions 1000 and 1010, the following code could be used:

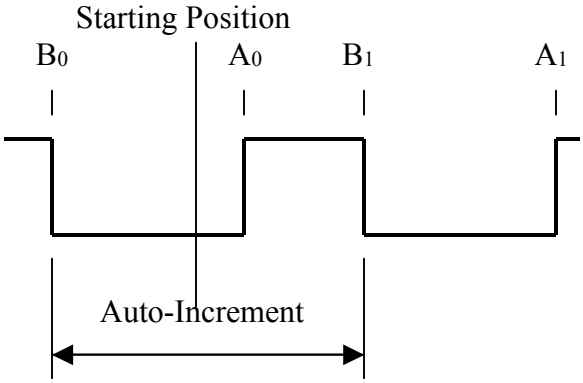
```
M108=1000          ; Set front end compare in A
M109=1010          ; Set back end compare in B
M110=0             ; No auto-increment
M112=0             ; Prepare initial value of 0
M111=1             ; Enable direct write (resets immediately to zero)
{Command to start the move}
```

Setting Up for Multiple Pulse Outputs

By using the auto-increment feature, it is possible to create multiple compare pulses with a single software setup operation. When the auto-increment register is a non-zero value, its value is automatically added to or subtracted from one compare register's value when the other compare value is matched. PMAC keeps track of the direction of incrementing, so only positive values should be used in the auto-increment register, even if the encoder will be counting in the negative direction.

The setup for multiple pulses is like the setup for a single pulse, except that a non-zero value must be entered into the auto-increment register, and the value entered for the back edge must be that of the first back edge minus the auto-increment if the move will be positive, or that of the first back edge plus the auto-increment value if the move will be negative.

In other words, the starting values to the two compare registers must "bracket" the starting position. When either compare value is matched by the encoder counter, the other compare value is incremented in the direction of movement.



Example: Starting from the above example, desiring the compare output on between 1000 (A_0) and 1010 (B_1) counts, but adding an auto-increment value of 2000 counts, with a starting position of about 100 counts, program code to start the sequence could be:

```
M110=2000          ; Auto-increment of 2000 encoder counts
M108=1000          ; First front edge (A0) at 1000 counts
M109=1010-M110     ; First back edge (B1) at 1010 counts
M112=0             ; Prepare initial value of 0
M111=1             ; Enable direct write (resets immediately to zero)
```

Extended-Count Compare

Starting in the "D" revision of the DSPGATE1 PMAC2-style Servo IC (released mid-2001), the compare circuitry has the capability of triggering the output at a fractional count value, not just at the nearest whole number of counts. This IC can create a fractional count value every encoder-sample clock (SCLK) cycle using 1/T timer-based techniques and use this extended count value in the position-compare (and capture) circuitry.

This capability is particularly valuable when using a sinusoidal encoder through an interpolator circuit, such as the ACC-51EC. The interpolator greatly increases the resolution of the servo feedback, but does not by itself increase the resolution of the compare function. Note, however, that use of a sinusoidal encoder and interpolator is not required for the extended compare function.

This extended count mode is enabled for a channel by setting bit 18 of the channel's control word to 1. Starting in the V1.937 revision of Turbo PMAC firmware (although undocumented for that revision), this bit has been assigned to I-variable I7mn9 for Servo IC m Channel n. Otherwise an M-variable can be assigned to this bit. A suggested M-variable for this bit for Servo IC 0 Channel 1 is:

```
M113->X:$078005,18,1    ; Extended-count mode enable
```

When this mode is enabled, the meaning of the two 24-bit timer registers for the channel changes. (This means that you cannot use the traditional 1/T count extension in the conversion table for this channel.) In extended-count mode these two registers contain four 12-bit values. The lower half of each register contains the fractional count value for the position compare function. Suggested M-variables for these values for Servo IC 0 Channel 1 are:

```
M188->Y:$078000,0,12,U  ; Compare A fractional count
M189->Y:$078001,0,12,U  ; Compare B fractional count
```

These registers must be used as unsigned values. Treated this way, these registers take a value of 0 to 4095, representing a (whole) count value 1/4096 as big.

The auto-increment function is still available in extended-count mode, but the auto-increment value is limited to integer numbers of counts.

In operation in this mode, the user writes to both the standard (integer) compare register and the fractional compare register. In this mode, the integer value written to the compare register is offset by one count from the count value at which the compare output will toggle. The direction of this offset is dependent on the direction of motion. If a value of *n* counts is written to the integer compare register, the compare output will toggle at *n+1* counts when moving in the positive direction, or *n-1* counts when moving in the negative direction. To compensate for this, simply add or subtract 1 count depending on the direction of motion. For instance, to get a compare edge at 743.25 counts when moving in the positive direction, write 742 (=743-1) to the integer compare register, and write 1024 (=0.25*4096) to the fractional compare register.

Example: P108 and P109 are floating-point variables representing the first count values where the compare output is to be turned on, then off. P110 is an integer value representing the auto-increment length in counts. Motion is to be in the positive direction. Program code to start the sequence could be:

```
M110=P110                       ; Auto-increment value
M108=INT(P108)-1                ; Compare A integer component
M188=(P108-(M108+1))*4096       ; Compare A fractional component
M109=INT(P109)-1-M110           ; Compare B integer component
M189=(P109-(M109+1+M110))*4096      ; Compare B fractional component
M112=0                          ; Prepare initial value of 0
M111=1                          ; Enable direct write
```

Converting from Motor and Axis Coordinates

The compare registers are scaled in encoder counts, referenced to the position at the latest power-up/reset. Typically, the user wants to work in either motor coordinates, still in counts and referenced to the home position, or in axis coordinates, in engineering units (mm, inches, degrees) and referenced to a user-set origin.

Two values are needed to convert motor position to encoder position for compare calculations. First is the "home capture offset", the encoder position captured at the home trigger, a value PMAC stores for future use. The second is the "home-offset" variable Ixx26, which contains the difference between the trigger position and the home position. The relationship is:

EncoderPosition = *MotorPosition* + *HomeCaptureOffset* + *HomeOffset*

The home capture offset is a 24-bit signed integer, expressed in counts. It is best accessed with a 24-bit signed M-variable. The register and suggested M-variable for Motor 1 is:

```
M173->Y:$0000CE,0,24,S  ; #1 Encoder home capture position (cts)
```

The home offset I-variable Ixx26 is in units of 1/16 count, so it should be divided by 16 before adding to the motor position.

For example, for Motor 1 using Encoder 1 in a PMAC2-style Servo IC, if you want to set the Compare 1A register to trigger at motor position +1500 counts, you could use the following command, on-line, in a motion program, or in a PLC program:

```
M108=1500+M173+I126/16
```

The conversion from axis position to motor position involves a scale factor and an offset, with the following equation:

MotorPosition = *ScaleFactor* * *AxisPosition* + *Offset*

The scale factor is specified in the axis definition statement in counts per engineering unit; it should be constant for an application. You can specify the scale factor directly in your equation, or you can access the actual register that PMAC is using with suggested M-variables:

```
Mxx91        ; Axis scale factor for X, U, A, B, or C assigned to Motor xx
Mxx92        ; Axis scale factor for Y or V assigned to Motor xx
Mxx93        ; Axis scale factor for Z or W assigned to Motor xx
```

The offset is the position bias term, with suggested M-variable Mx64, and units of 1/(Ixx08*32) counts. It is set equal to the axis definition offset (usually 0) on power-up/reset and homing. It can be changed after this with on-line command **{axis}=** or motion program command **PSET**.

For example, with Motor 1 assigned to the X-axis, if you want to set the Compare 1A register to trigger at +2.5 engineering units from the axis origin, you can compute motor position in counts as:

```
P1=M191*2.5+M164/(I108*32)
```

Then you can set the actual compare register with:

```
M108=P1+M173+I126/16
```

Note that if the expression on the right-hand side of this equation had produced a result outside of the range of the compare register, the act of writing to the M-variable assigned to this register would automatically truncate the value properly to the 24-bit range so that the compare function will work properly. 24-bit signed registers have a range of -8,388,608 to +8,388,607. If you attempted to write a value of +9,388,608 – one million counts past the rollover point – into this register, the resulting value would be -7,388,608, which would be reached one million counts after the rollover.

Turbo PMAC Mathematical Features

Mathematical Operators

+

Function: Addition

The + sign implements the addition of the numerical values preceding and following it.

Multiplication, division, modulo (remainder), and bit-by-bit "and" operations have higher priority than addition, subtraction, bit-by-bit "or", and bit-by-bit "exclusive-or" operations. Operations of the same priority are implemented from left to right.

The + sign may not be used as a "unary" operator to emphasize that the positive value of the following variable or constant is to be used (e.g. **P1=+P2**). This syntax will be rejected with an error.

Execution time, 80 MHz CPU: 2.9 µsec interpreted, 1.2 µsec compiled floating-point, 0.08 µsec compiled fixed-point

-

Function: Subtraction, negation

The - sign implements the subtraction of the numerical value following it from the numerical value preceding it. If there is no numerical value preceding it, it causes the negation of the numerical value following it.

Multiplication, division, modulo (remainder), and bit-by-bit "and" operations have higher priority than addition, subtraction, bit-by-bit "or", and bit-by-bit "exclusive-or" operations. Operations of the same priority are implemented from left to right.

Execution time, 80 MHz CPU: 2.9 µsec interpreted, 1.2 µsec compiled floating-point, 0.08 µsec compiled fixed-point

*

Function: Multiplication

The * sign implements the multiplication of the numerical values preceding and following it.

Multiplication, division, modulo (remainder), and bit-by-bit "and" operations have higher priority than addition, subtraction, bit-by-bit "or", and bit-by-bit "exclusive-or" operations. Operations of the same priority are implemented from left to right.

Execution time, 80 MHz CPU: 3.0 µsec interpreted, 1.0 µsec compiled floating-point, 0.13 µsec compiled fixed-point

/

Function: Division

The / sign implements the division of the numerical value preceding it by the numerical value following it. Unless the division is executed in a compiled PLC on a line with only L-variables and integers, the division operation is always a floating-point calculation (even if integer values are used). The quotient is computed as a floating-point value and used as such in subsequent calculations in the same expression; if it is then stored to an integer, it is rounded at the time of storage.

If the division operation is performed as an integer operation in a compiled PLC (only L-variables and integer constants on the line), the quotient is computed as an integer (rounded to the nearest integer value) and used as such in subsequent calculations in the same expression.

Multiplication, division, modulo (remainder), and bit-by-bit "and" operations have higher priority than addition, subtraction, bit-by-bit "or", and bit-by-bit "exclusive-or" operations. Operations of the same priority are implemented from left to right.

If the divisor is equal to 0, the result will saturate at $+/-2^{47}$ ($+/-2^{23}$ for an integer division in a compiled PLC). No error will be reported, and the program will not stop. It is the programmer's responsibility to check for possible division-by-zero errors.

Execution time, 80 MHz CPU: 3.6 µsec interpreted, 1.6 µsec compiled floating-point, 0.73 µsec compiled fixed-point

Examples

(L1 and M1 are integer variables; P1 is a floating-point variable)

Command	Result
`P1=10*2/3`	6.666666667
`P1=10*(2/3)`	6.666666667
`M1=10*2/3`	7
`M1=10*(2/3)`	7
`L1=10*2/3`	7
`L1=10*(2/3)`	10

%

Function: Modulo (remainder)

The % sign causes the calculation of the remainder due to the division of the numerical value preceding it by the numerical value following it. Unless the division is executed in a compiled PLC on a line with only L-variables and integers, the division operation is always a floating-point calculation (even if integer values are used). The quotient is computed as a floating-point value, then truncated to the next lowest (i.e. toward -∞) integer so the remainder can be computed.

If the divisor "n" is a positive value, the modulo result is in the range $0 \leq \text{Result} < n$. If the divisor "n" is a negative value, the modulo result is in the range $-n \leq \text{Result} < n$.

Multiplication, division, modulo (remainder), and bit-by-bit "and" operations have higher priority than addition, subtraction, bit-by-bit "or", and bit-by-bit "exclusive-or" operations. Operations of the same priority are implemented from left to right.

If the divisor is equal to 0, the division will saturate and the modulo result will be 0. No error will be reported, and the program will not stop. It is the programmer's responsibility to check for possible division-by-zero errors.

Execution time, 80 MHz CPU: 3.3 µsec interpreted, 1.4 µsec compiled floating-point, 0.78 µsec compiled fixed-point

Examples

Operation	Result
`11%4`	3
`-11%4`	1
`11%-4`	3
`-11%-4`	-3
`3%2.5`	0.5
`-3%2.5`	2
`3%-2.5`	-2
`-3%-2.5`	2

&

Function: Bit-by-bit "and"

The & sign implements the bit-by-bit logical "and" of the numerical value preceding it and the numerical value following it. A given bit of the result is equal to 1 if and only if the matching bits of both operands are equal to 1. The operation is done both on integer bits and fractional bits (if any).

Multiplication, division, modulo (remainder), and bit-by-bit "and" operations have higher priority than addition, subtraction, bit-by-bit "or", and bit-by-bit "exclusive-or" operations. Operations of the same priority are implemented from left to right.

This bit-by-bit "and" operator that logically combines the bits of numerical values is not to be confused with the **AND** command, which logically combines conditions.

Execution time, 80 MHz CPU: 3.4 µsec interpreted, 1.4 µsec compiled floating-point, 0.10 µsec compiled fixed-point

Examples

Operation	Result
3&1	1
3&2	2
3&3	3
3&4	0
3&-3	1
0.875&1.75	0.75
0.875&-1.75	0.25

|

Function: Bit-by-bit "or"

The | sign implements the bit-by-bit logical "or" of the numerical value preceding it and the numerical value following it. A given bit of the result is equal to 1 if the matching bit of either operand is equal to 1. The operation is done both on integer bits and fractional bits (if any).

Multiplication, division, modulo (remainder), and bit-by-bit "and" operations have higher priority than addition, subtraction, bit-by-bit "or", and bit-by-bit "exclusive-or" operations. Operations of the same priority are implemented from left to right.

This bit-by-bit "or" operator that logically combines the bits of numerical values is not to be confused with the **OR** command, which logically combines conditions.

Execution time, 80 MHz CPU: 3.2 µsec interpreted, 1.4 µsec compiled floating-point, 0.10 µsec compiled fixed-point

Examples

Operation	Result
4|3	7
3|2	3
3|3	3
$F0|$4	$F4
3|-3	-1
0.5|0.375	0.875
0.875|-1.75	-0.375

^

Function: Bit-by-bit "exclusive or"

The ^ sign implements the bit-by-bit logical "exclusive or" (xor) of the numerical value preceding it and the numerical value following it. A given bit of the result is equal to 1 if and only if the matching bits of the two operands are different from each other. The operation is done both on integer bits and fractional bits (if any).

Multiplication, division, modulo (remainder), and bit-by-bit "and" operations have higher priority than addition, subtraction, bit-by-bit "or", and bit-by-bit "exclusive-or" operations. Operations of the same priority are implemented from left to right.

Execution time, 80 MHz CPU: 3.1 µsec interpreted, 1.4 µsec compiled floating-point, 0.10 µsec compiled fixed-point

Examples

Operation	Result
2^1	3
2^2	0
5^7	2
$AA^$55	$FF
3^-3	-2
0.5^0.875	0.375

Mathematical Functions

ABS

Function: Absolute value
Syntax: **ABS({expression})**
Domain: All real numbers
Domain units: User-determined
Range: Non-negative real numbers
Range units: User-determined

ABS implements the absolute-value, or magnitude, function, of the mathematical expression contained inside the following parentheses.

Execution time, 80 MHz CPU: 3.1 µsec interpreted, 0.6 µsec compiled

Examples

```
P8=ABS(P7)              ; Computes magnitude of P7
```

```
IF(Q200!=0)        ; Divide by 0 check
  Q240=ABS(Q200)/Q200   ; Computes sign (-1 or 1) of Q200
ELSE
  Q240=0                ; Sign value is 0
ENDIF
```

ACOS

Function:	Trigonometric arc-cosine
Syntax:	**ACOS({expression})**
Domain:	-1.0 to +1.0
Domain units:	none
Range:	0 to Pi radians (0 to 180 degrees)
Range units:	Radians/degrees

ACOS implements the inverse cosine, or arc-cosine, function, of the mathematical expression contained inside the following parentheses.

This function returns values in degrees if I15 is set to the default value of 0; it returns values in radians if I15 is set to 1.

If the argument inside the parentheses is outside of the legal domain of –1.0 to +1.0, an arbitrary value will be returned. No error will be reported, and the program will not stop. It is the programmer's responsibility to check for possible domain errors.

Execution time, 80 MHz CPU: 6.7 μsec interpreted, 4.3 μsec compiled

Examples

```
P50=ACOS(P48/P49) ; Computes angle whose cos is P48/P49
C(ACOS(Q70/10))         ; Move C axis to specified angle
```

ASIN

Function:	Trigonometric arc-sine
Syntax:	**ASIN({expression})**
Domain:	-1.0 to +1.0
Domain units:	none
Range:	0 to Pi radians (0 to 180 degrees)
Range units:	Radians/degrees

ASIN implements the inverse sine, or arc-sine, function, of the mathematical expression contained inside the following parentheses.

This function returns values in degrees if I15 is set to the default value of 0; it returns values in radians if I15 is set to 1.

If the argument inside the parentheses is outside of the legal domain of –1.0 to +1.0, an arbitrary value will be returned. No error will be reported, and the program will not stop. It is the programmer's responsibility to check for possible domain errors.

Execution time, 80 MHz CPU: 7.0 µsec interpreted, 4.7 µsec compiled

Examples

```
P50=ASIN(P48/P49) ; Computes angle whose sin is P48/P49
C(ASIN(Q70/10))         ; Move C axis to specified angle
```

ATAN

Function:	Trigonometric arc-tangent
Syntax:	**ATAN({expression})**
Domain:	All real numbers
Domain units:	none
Range:	-Pi/2 to +Pi/2 radians (-90 to +90 degrees)
Range units:	Radians/degrees

ATAN implements the standard inverse tangent, or arc-tangent, function, of the mathematical expression contained inside the following parentheses. This standard arc-tangent function returns values only in the +/-90-degree range; if a full +/-180-degree range is desired, the **ATAN2** function should be used instead.

This function returns values in degrees if I15 is set to the default value of 0; it returns values in radians if I15 is set to 1.

Execution time, 80 MHz CPU: 5.9 µsec interpreted, 3.5 µsec compiled

Examples

```
P50=ATAN(P48/P49) ; Computes angle whose tan is P48/P49
C(ATAN(Q70/10))         ; Move C axis to specified angle
```

ATAN2

Function:	Two-argument trigonometric arc-tangent
Syntax:	**ATAN2({expression})**
Domain:	All real numbers in both arguments
Domain units:	none
Range:	-Pi to +Pi radians (-180 to +180 degrees)
Range units:	Radians/degrees

ATAN2 implements the expanded (two-argument) inverse tangent, or arc-tangent, function, of the mathematical expression contained inside the following parentheses, and the value of variable Q0 for the coordinate system used. (If this function is used inside a PLC program, make sure the desired coordinate system has been selected with the **ADDRESS** command.)

This expanded arc-tangent function returns values in the full +/-180-degree range; if only the +/-90-degree range is desired, the standard **ATAN** function should be used instead. The **ATAN2** function makes use of the signs of both arguments, as well as the ratio of their magnitudes, to extend the range to a full 360 degrees. The value in the parentheses following **ATAN2** is the "sine" argument; the value in Q0 is the "cosine" argument.

This function returns values in degrees if I15 is set to the default value of 0; it returns values in radians if I15 is set to 1.

If both arguments for the **ATAN2** function are equal to exactly 0.0, an internal division-by-zero error will result, and an arbitrary value will be returned. No error will be reported, and the program will not stop. It is the programmer's responsibility to check for these possible domain errors.

Execution time, 80 MHz CPU: 5.1 µsec interpreted, 3.5 µsec compiled

Examples

```
Q30=-0.707              ; "Cosine" argument
Q31=-0.707              ; "Sine" argument
Q32=ATAN(Q31/Q30) ; Single-argument arctangent
Q32                     ; Query resulting value
45                      ; Returns value in +/-90 range
Q0=Q30                  ; Prepare "cosine" for ATAN2
Q33=ATAN2(Q31)          ; Two-argument arctangent
Q33                     ; Query resulting value
-135                    ; Note different result

Q0=M163-M161            ; X target - X present position
Q1=M263-M261            ; Y target - Y present position
IF (ABS(Q0)>0.001 OR ABS(Q1)>0.001) ; Div by 0 check
  Q2=ATAN2(Q1)          ; Calculate directed angle
ENDIF
```

COS

Function:	Trigonometric cosine
Syntax:	**COS({expression})**
Domain:	All real numbers
Domain units:	Radians/degrees
Range:	-1.0 to +1.0
Range units:	none

COS implements the trigonometric cosine function of the mathematical expression contained inside the following parentheses.

This function interprets its argument in degrees if I15 is set to the default value of 0; it interprets its argument in radians if I15 is set to 1.

Execution time, 80 MHz CPU: 5.6 µsec interpreted, 3.2 µsec compiled

Examples

```
P60=COS(30)          ; Computes cosine of 30
X(Q80*COS(Q81))          ; Move X axis to calculated value
```

EXP

Function:	Exponentiation (e^x)
Syntax:	**EXP({expression})**
Domain:	All real numbers
Domain units:	User-determined
Range:	Positive real numbers
Range units:	User-determined

EXP implements the standard exponentiation function of the mathematical expression contained inside the following parentheses, raising "e" to the power of this expression.

To implement the y^x function, use $e^{x \ln(y)}$ instead.

Execution time, 80 MHz CPU: 5.6 µsec interpreted, 3.2 µsec compiled

Examples

```
P20=EXP(P19)             ; Raises e to the power of P19
P3=EXP(P2*LN(P1))        ; Raises P1 to the power of P2
```

INT

Function:	Truncation to integer
Syntax:	**INT({expression})**
Domain:	All real numbers
Domain units:	User-determined
Range:	All integers
Range units:	User-determined

INT implements the truncation to integer function of the mathematical expression contained inside the following parentheses. The truncation is always done in the negative direction.

Note that while the result is an integer number, it is still represented as a floating-point value.

Execution time, 80 MHz CPU: 3.3 µsec interpreted, 1.0 µsec compiled

Examples

```
P50=2.5                  ;
P51=INT(P50)             ; Take INT of positive value
P51                      ; Query resulting value
2                        ; Next lower integer value
P52=INT(-P50)            ; Take INT of negative value
```

```
P52                     ; Query resulting value
-3                      ; Next lower integer value
```

LN

Function:	Natural logarithm
Syntax:	**EXP({expression})**
Domain:	Positive real numbers
Domain units:	User-determined
Range:	All real numbers
Range units:	User-determined

LN implements the natural logarithm (logarithm base "e") function of the mathematical expression contained inside the following parentheses.

To implement the logarithm using another base, divide the natural logarithm of the value by the natural logarithm of the base ($\log_y x = \ln x / \ln y$). The natural logarithm of 10 is equal to 2.302585.

If the argument inside the parentheses is outside of the legal domain of positive numbers, a 0 value will be returned. No error will be reported, and the program will not stop. It is the programmer's responsibility to check for possible domain errors.

Execution time, 80 MHz CPU: 4.3 µsec interpreted, 1.4 µsec compiled

Examples

```
P19=LN(P20)         ; Takes the natural log of P20
P6=LN(P5)/LN(10)        ; Takes the log base 10 of P5
```

SIN

Function:	Trigonometric sine
Syntax:	**SIN({expression})**
Domain:	All real numbers
Domain units:	Radians/degrees
Range:	-1.0 to +1.0
Range units:	none

SIN implements the trigonometric sine function of the mathematical expression contained inside the following parentheses.

This function interprets its argument in degrees if I15 is set to the default value of 0; it interprets its argument in radians if I15 is set to 1.

Execution time, 80 MHz CPU: 5.6 µsec interpreted, 3.2 µsec compiled

Examples

```
P60=SIN(30)         ; Computes cosine of 30
Y(Q80*SIN(Q81))           ; Move Y axis to calculated value
```

SQRT

Function:	Square root
Syntax:	**SQRT({expression})**
Domain:	Non-negative real numbers
Domain units:	User-determined
Range:	Non-negative real numbers
Range units:	User-determined

SQRT implements the positive square-root function of the mathematical expression contained inside the following parentheses.

If the argument inside the parentheses is outside of the legal domain of non-negative numbers, an arbitrary value will be returned. No error will be reported, and the program will not stop. It is the programmer's responsibility to check for possible domain errors.

Execution time, 80 MHz CPU: 3.7 μsec interpreted, 1.4 μsec compiled

Examples

```
P19=SQRT(P20)             ; Takes the square root of P20
P50=SQRT(P8*P8+P9*P9)     ; Pythagorean theorem calculation
```

TAN

Function:	Trigonometric tangent
Syntax:	**TAN({expression})**
Domain:	All real numbers except +/-(2N-1)*90 degrees
Domain units:	Radians/degrees
Range:	All real numbers
Range units:	none

TAN implements the trigonometric tangent function of the mathematical expression contained inside the following parentheses.

This function interprets its argument in degrees if I15 is set to the default value of 0; it interprets its argument in radians if I15 is set to 1.

If the argument inside the parentheses approaches +/-(2N-1)*90 degrees (+/-90, 270, 450, etc.), the **TAN** function will "blow up" and a very large value will be returned. If the argument inside the parentheses is exactly equal to one of these quantities, an internal division-by-zero error will occur and the resulting value will saturate at $+/-2^{47}$. No error will be reported, and the program will not stop. It is the programmer's responsibility to check for possible domain errors.

Execution time, 80 MHz CPU: 6.8µsec interpreted, 4.5 µsec compiled

Examples

```
P60=TAN(30)        ; Computes cosine of 30
Y(Q80*TAN(Q81))          ; Move Y axis to calculated value
```

New/Revised I-Variables/Descriptions

I12 Lookahead Time Spline Enable

Range: 0 - 1
Units: none
Default: 0

I12 permits the enabling of a new lookahead technique called "time splining". If I12 is set to 1, all coordinate systems that are executing lookahead will use this technique. If I12 is set to 0, none of them will.

"Time splining" permits smoother transitions from one vector velocity to another during lookahead when there is little or no change in direction. As long as the commanded vector velocity going into lookahead does not change by more than a factor of two in a single Isx13 segment, the velocity change will be made without any velocity "undershoot". Without this technique, large changes in vector velocity that have to be extended by lookahead can cause significant velocity undershoot.

Setting I12 to 1 adds a small but potentially significant computational load to the lookahead calculations.

I30 Compensation Table Wrap Enable

Range: 0 - 1
Units: none
Default: 0

I30 controls whether the compensation tables entered into Turbo PMAC will automatically "wrap" or not. This affects position ("leadscrew"), backlash, and torque compensation tables. If I30 is set to 0, when a table is downloaded to PMAC, the compensation correction at motor position 0 is always set to 0. In this case, if smooth rollover of the table is desired, the last entry of the table must explicitly be set to 0.

If I30 is set to 1, the last entry of the table also becomes the correction at motor position 0, automatically yielding a smooth rollover of the table, and permitting non-zero corrections at the rollover point.

I30 affects table values only as they are being downloaded to Turbo PMAC; it does not affect the values of tables already in Turbo PMAC's memory.

I37 Additional Wait States

Range:	$000000 - $032403
Units:	Instruction cycle wait states (by bit)
Default:	$000000

I37 controls the number of "wait states" added to the factory default values when the Turbo PMAC processor accesses external memory or memory-mapped I/O devices. Wait states are the number of instruction cycles the processor idles when reading from or writing to a register of memory or I/O. On power-up/reset, Turbo PMAC automatically sets the number of wait states based on the programmed CPU frequency as set by I54. Under certain circumstances, particularly in accessing third-party devices, more robust operation may be obtained by increasing the number of wait states from the factory default values (at the cost of slightly slower operation.

I37 is divided into four parts, each controlling the wait states for a different area of the memory and I/O map. Bits 0 and 1 control the number of added wait states for I/O devices (such as ASICs and A/D converters; dual-ported RAM also counts as an I/O device) mapped into Y-registers. Bits 16 and 17 control the number of added wait states for I/O devices mapped into X-registers. With 2 bits each, up to 3 wait states can be added to these accesses; generally these are both set to the same value.

Bit 10 of I37 controls the number of added wait states for "P" (program, or machine-code) memory register access. Bit 13 controls the number of added wait states for "X" and "Y" (data) memory register access. As single-bit values, they can add only one wait state to these memory accesses. Generally, these are both set to the same value.

I37 is used at power-up/reset only, so to change the number of I/O wait states, change the value of I37, issue a **SAVE** command, and reset the Turbo PMAC. At power-up/reset, Turbo PMAC automatically adds the value of I37 to the value from its internal look-up table to set the number of I/O wait states. The resulting number of wait states for different areas of the memory and I/O map is in internal CPU register X:$FFFFFB.

Examples

```
I37=$020002     ; Add 2 wait states to X and Y I/O access.
I37=$002400     ; Add 1 wait state to X/Y and P memory access
I37=$032403     ; Add 3 wait states for I/O, 1 for memory
```

I44 PMAC Ladder Program Enable {Special Firmware Only}

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

I44 controls whether the "PMAC Ladder" graphical PLC programs that can be used with optional firmware are running or not. If I44 is set to 1, any PMAC ladder programs that have been downloaded into Turbo PMAC program memory are active. If I44 is set to 0, these programs will not execute, even if they are present.

If the firmware does not support these "PMAC Ladder" PLC programs, I443 cannot be changed from 0.

Ixx02 Motor xx Command Output Address {revised description}

Range: $000000 - $FFFFFF
Units: Turbo PMAC Addresses

Turbo PMAC(1) Ixx02 Defaults

Ixx02	Value	Register	Ixx02	Value	Register
I102	$078003	PMAC DAC1	I1702	$079203	2nd ACC-24P/V DAC1
I202	$078002	PMAC DAC2	I1802	$079202	2nd ACC-24P/V DAC2
I302	$07800B	PMAC DAC3	I1902	$07920B	2nd ACC-24P/V DAC3
I402	$07800A	PMAC DAC4	I2002	$07920A	2nd ACC-24P/V DAC4
I502	$078103	PMAC DAC5	I2102	$079303	2nd ACC-24P/V DAC5
I602	$078102	PMAC DAC6	I2202	$079302	2nd ACC-24P/V DAC6
I702	$07810B	PMAC DAC7	I2302	$07930B	2nd ACC-24P/V DAC7
I802	$07810A	PMAC DAC8	I2402	$07930A	2nd ACC-24P/V DAC8
I902	$078203	1st ACC-24P/V DAC1	I2502	$07A203	3rd ACC-24P/V DAC1
I1002	$078202	1st ACC-24P/V DAC2	I2602	$07A202	3rd ACC-24P/V DAC2
I1102	$07820B	1st ACC-24P/V DAC3	I2702	$07A20B	3rd ACC-24P/V DAC3
I1202	$07820A	1st ACC-24P/V DAC4	I2802	$07A20A	3rd ACC-24P/V DAC4
I1302	$078303	1st ACC-24P/V DAC5	I2902	$07A303	3rd ACC-24P/V DAC5
I1402	$078302	1st ACC-24P/V DAC6	I3002	$07A302	3rd ACC-24P/V DAC6
I1502	$07830B	1st ACC-24P/V DAC7	I3102	$07A30B	3rd ACC-24P/V DAC7
I1602	$07830A	1st ACC-24P/V DAC8	I3202	$07A30A	3rd ACC-24P/V DAC8

Turbo PMAC2 Ixx02 Defaults

Ixx02	Value	Register	Ixx02	Value	Register
I102	$078002	PMAC2 DAC/PWM1A	I1702	$079202	2nd ACC-24P/V2 DAC/PWM1A
I202	$07800A	PMAC2 DAC/PWM2A	I1802	$07920A	2nd ACC-24P/V2 DAC/PWM2A
I302	$078012	PMAC2 DAC/PWM3A	I1902	$079212	2nd ACC-24P/V2 DAC/PWM3A
I402	$07801A	PMAC2 DAC/PWM4A	I2002	$07921A	2nd ACC-24P/V2 DAC/PWM4A
I502	$078102	PMAC2 DAC/PWM5A	I2102	$079302	2nd ACC-24P/V2 DAC/PWM5A
I602	$07810A	PMAC2 DAC/PWM6A	I2202	$07930A	2nd ACC-24P/V2 DAC/PWM6A
I702	$078112	PMAC2 DAC/PWM7A	I2302	$079312	2nd ACC-24P/V2 DAC/PWM7A
I802	$07811A	PMAC2 DAC/PWM8A	I2402	$07931A	2nd ACC-24P/V2 DAC/PWM8A
I902	$078202	1st ACC-24P/V2 DAC/PWM1A	I2502	$07A202	3rd ACC-24P/V2 DAC/PWM1A
I1002	$07820A	1st ACC-24P/V2 DAC/PWM2A	I2602	$07A20A	3rd ACC-24P/V2 DAC/PWM2A
I1102	$078212	1st ACC-24P/V2 DAC/PWM3A	I2702	$07A212	3rd ACC-24P/V2 DAC/PWM3A
I1202	$07821A	1st ACC-24P/V2 DAC/PWM4A	I2802	$07A21A	3rd ACC-24P/V2 DAC/PWM4A
I1302	$078302	1st ACC-24P/V2 DAC/PWM5A	I2902	$07A302	3rd ACC-24P/V2 DAC/PWM5A
I1402	$07830A	1st ACC-24P/V2 DAC/PWM6A	I3002	$07A30A	3rd ACC-24P/V2 DAC/PWM6A
I1502	$078312	1st ACC-24P/V2 DAC/PWM7A	I3102	$07A312	3rd ACC-24P/V2 DAC/PWM7A
I1602	$07831A	1st ACC-24P/V2 DAC/PWM8A	I3202	$07A31A	3rd ACC-24P/V2 DAC/PWM8A

Turbo PMAC2 Ultralite Ixx02 Defaults

Ixx02	Value	Register	Ixx02	Value	Register
I102	$078420	MACRO IC 0 Node 0 Reg. 0	I1702	$07A420	MACRO IC 2 Node 0 Reg. 0
I202	$078424	MACRO IC 0 Node 1 Reg. 0	I1802	$07A424	MACRO IC 2 Node 1 Reg. 0
I302	$078428	MACRO IC 0 Node 4 Reg. 0	I1902	$07A428	MACRO IC 2 Node 4 Reg. 0
I402	$07842C	MACRO IC 0 Node 5 Reg. 0	I2002	$07A42C	MACRO IC 2 Node 5 Reg. 0
I502	$078430	MACRO IC 0 Node 8 Reg. 0	I2102	$07A430	MACRO IC 2 Node 8 Reg. 0
I602	$078434	MACRO IC 0 Node 9 Reg. 0	I2202	$07A434	MACRO IC 2 Node 9 Reg. 0
I702	$078438	MACRO IC 0 Node 12 Reg. 0	I2302	$07A438	MACRO IC 2 Node 12 Reg. 0
I802	$07843C	MACRO IC 0 Node 13 Reg. 0	I2402	$07A43C	MACRO IC 2 Node 13 Reg. 0
I902	$079420	MACRO IC 1 Node 0 Reg. 0	I2502	$07B420	MACRO IC 3 Node 0 Reg. 0
I1002	$079424	MACRO IC 1 Node 1 Reg. 0	I2602	$07B424	MACRO IC 3 Node 1 Reg. 0
I1102	$079428	MACRO IC 1 Node 4 Reg. 0	I2702	$07B428	MACRO IC 3 Node 4 Reg. 0
I1202	$07942C	MACRO IC 1 Node 5 Reg. 0	I2802	$07B42C	MACRO IC 3 Node 5 Reg. 0
I1302	$079430	MACRO IC 1 Node 8 Reg. 0	I2902	$07B430	MACRO IC 3 Node 8 Reg. 0
I1402	$079434	MACRO IC 1 Node 9 Reg. 0	I3002	$07B434	MACRO IC 3 Node 9 Reg. 0
I1502	$079438	MACRO IC 1 Node 12 Reg. 0	I3102	$07B438	MACRO IC 3 Node 12 Reg. 0
I1602	$07943C	MACRO IC 1 Node 13 Reg. 0	I3202	$07B43C	MACRO IC 3 Node 13 Reg. 0

UMAC Turbo Ixx02 Defaults

Ixx02	Value	Register	Ixx02	Value	Register
I102	$078202	1st ACC-24E2x DAC/PWM1A	I1702	$07A202	5th ACC-24E2x DAC/PWM1A
I202	$07820A	1st ACC-24E2x DAC/PWM2A	I1802	$07A20A	5th ACC-24E2x DAC/PWM2A
I302	$078212	1st ACC-24E2x DAC/PWM3A	I1902	$07A212	5th ACC-24E2x DAC/PWM3A
I402	$07821A	1st ACC-24E2x DAC/PWM4A	I2002	$07A21A	5th ACC-24E2x DAC/PWM4A
I502	$078302	2nd ACC-24E2x DAC/PWM1A	I2102	$07A302	6th ACC-24E2x DAC/PWM1A
I602	$07830A	2nd ACC-24E2x DAC/PWM2A	I2202	$07A30A	6th ACC-24E2x DAC/PWM2A
I702	$078312	2nd ACC-24E2x DAC/PWM3A	I2302	$07A312	6th ACC-24E2x DAC/PWM3A
I802	$07831A	2nd ACC-24E2x DAC/PWM4A	I2402	$07A31A	6th ACC-24E2x DAC/PWM4A
I902	$079202	3rd ACC-24E2x DAC/PWM1A	I2502	$07B202	7th ACC-24E2x DAC/PWM1A
I1002	$07920A	3rd ACC-24E2x DAC/PWM2A	I2602	$07B20A	7th ACC-24E2x DAC/PWM2A
I1102	$079212	3rd ACC-24E2x DAC/PWM3A	I2702	$07B212	7th ACC-24E2x DAC/PWM3A
I1202	$07921A	3rd ACC-24E2x DAC/PWM4A	I2802	$07B21A	7th ACC-24E2x DAC/PWM4A
I1302	$079302	4th ACC-24E2x DAC/PWM1A	I2902	$07B302	8th ACC-24E2x DAC/PWM1A
I1402	$07930A	4th ACC-24E2x DAC/PWM2A	I3002	$07B30A	8th ACC-24E2x DAC/PWM2A
I1502	$079312	4th ACC-24E2x DAC/PWM3A	I3102	$07B312	8th ACC-24E2x DAC/PWM3A
I1602	$07931A	4th ACC-24E2x DAC/PWM4A	I3202	$07B31A	8th ACC-24E2x DAC/PWM4A

Ixx02 tells Motor xx which register or registers to which it writes its command output values. It contains the address of this register or the first (lowest addresses) of these multiple registers. This determines which output lines transmit the command output signals.

No Commutation: If Turbo PMAC is not commutating Motor xx (Ixx01=0 or 2), only one command output value is calculated, which is written to the register at the address specified in Ixx02. If Ixx01 is set to 0, this register is a Y-register; if Ixx01 is set to 2, this register is an X-register. Almost all output registers on PMAC are Y-registers; the only common use of X-register outputs is in the Type 0 MACRO protocol.

On Turbo PMAC(1) boards, if Ixx01 is set to 0 or 2 and Ixx96 is set to 1, then only the magnitude of the command is written to the register specified by Ixx02; the sign of the command is written to bit 14 of the flag register specified by Ixx25, which is usually the AENA/DIR output. If this sign-and-magnitude mode is used, bit 16 of Ixx24 should be set to 1 so this bit is not used for the amplifier-enable function. Sign-and-magnitude mode does not work with PMAC2-style Servo ICs.

The default values listed above are usually suitable for commanding single analog outputs (velocity or torque mode) when the Turbo PMAC is not commutating the motor.

Commutation, No Current Loop: If Turbo PMAC is commutating Motor xx (Ixx01=1 or 3), but not closing its current loop (Ixx82=0), two command output values are calculated, which are written to the Y-register at the address specified in Ixx02, plus the Y-register at the next higher address.

The default values listed above are usually suitable for commanding analog output pairs when the Turbo PMAC is commutating the motor, but not closing the current loop.

Commutation and Current Loop: If Turbo PMAC is commutating Motor xx (Ixx01=1 or 3) and closing its current loop (Ixx82>0), three command output values are calculated, which are written to the Y-register at the address specified in Ixx02, plus the Y-registers at the next two higher addresses.

The default values listed above are usually suitable for commanding three-phase PWM sets when the Turbo PMAC is commutating the motor, and closing the current loop.

Pulse Frequency Output: One common application type for which the default value of Ixx02 cannot be used is the direct pulse-and-direction output for stepper motor drives (Turbo PMAC2 only). This mode uses the 'C' output register alone for each channel, and I7mn6 for Servo IC **m** Channel **n** must be set to 2 or 3 to get pulse frequency output. In this case, the following values should be used:

Turbo PMAC2 Ixx02 Pulse Frequency Output Settings

Servo IC #	Chan. 1	Chan. 2	Chan. 3	Chan. 4	Notes
0	$078002	$07800A	$078012	$07801A	1st IC on board PMAC2, 3U stack
1	$078102	$07810A	$078012	$07801A	2nd IC on board PMAC2, 3U stack
2	$078202	$07820A	$078212	$07821A	1st ACC-24E2x, 1st IC on 1st ACC-24P/V2
3	$078302	$07830A	$078312	$07831A	2nd ACC-24E2x, 2nd IC on 1st ACC-24P/V2
4	$079202	$07920A	$079212	$07921A	3rd ACC-24E2x, 1st IC on 2nd ACC-24P/V2
5	$079302	$07930A	$079312	$07931A	4th ACC-24E2x, 2nd IC on 2nd ACC-24P/V2
6	$07A202	$07A20A	$07A212	$07A21A	5th ACC-24E2x, 1st IC on 3rd ACC-24P/V2
7	$07A302	$07A30A	$07A312	$07A31A	6th ACC-24E2x, 2nd IC on 3rd ACC-24P/V2
8	$07B202	$07B20A	$07B212	$07B21A	7th ACC-24E2x, 1st IC on 4th ACC-24P/V2
9	$07B302	$07B30A	$07B312	$07B31A	8th ACC-24E2x, 2nd IC on 4th ACC-24P/V2

MACRO Type 1 Command Outputs: To write command outputs to MACRO registers for Type 1 MACRO devices such as the Delta Tau MACRO Station, the values of Ixx02 shown above as defaults for the Turbo PMAC2 Ultralite can be used.

MACRO Type 0 Command Outputs: To write single velocity or torque command outputs to MACRO registers for Type 0 MACRO drives such as the Performance Controls FLX Drive and the Kollmorgen FAST Drive, the values of Ixx02 in the following table should be used. Each value can select two registers (e.g. for Node 0 and Node 2). To select the lower-numbered node's register, which is a Y-register in Turbo PMAC, Ixx01 should be set to 0; to select the higher-numbered node's register, which is a Y-register, Ixx01 should be set to 2.

Ixx02 for Type 0 MACRO Commands

Ixx02	Value	Register	Ixx02	Value	Register
I102	$078423	MACRO IC 0 Node 0/2 Reg. 3	I1702	$07A423	MACRO IC 2 Node 0/2 Reg. 3
I202	$078427	MACRO IC 0 Node 1/3 Reg. 3	I1802	$07A427	MACRO IC 2 Node 1/3 Reg. 3
I302	$07842B	MACRO IC 0 Node 4/6 Reg. 3	I1902	$07A42B	MACRO IC 2 Node 4/6 Reg. 3
I402	$07842F	MACRO IC 0 Node 5/7 Reg. 3	I2002	$07A42F	MACRO IC 2 Node 5/7 Reg. 3
I502	$078433	MACRO IC 0 Node 8/10 Reg. 3	I2102	$07A433	MACRO IC 2 Node 8/10 Reg. 3
I602	$078437	MACRO IC 0 Node 9/11 Reg. 3	I2202	$07A437	MACRO IC 2 Node 9/11 Reg. 3
I702	$07843B	MACRO IC 0 Node 12/14 Reg. 3	I2302	$07A43B	MACRO IC 2 Node 12/14 Reg. 3
I802	$07843F	MACRO IC 0 Node 13/15 Reg. 3	I2402	$07A43F	MACRO IC 2 Node 13/15 Reg. 3
I902	$079423	MACRO IC 1 Node 0/2 Reg. 3	I2502	$07B423	MACRO IC 3 Node 0/2 Reg. 3
I1002	$079427	MACRO IC 1 Node 1/3 Reg. 3	I2602	$07B427	MACRO IC 3 Node 1/3 Reg. 3
I1102	$07942B	MACRO IC 1 Node 4/6 Reg. 3	I2702	$07B42B	MACRO IC 3 Node 4/6 Reg. 3
I1202	$07942F	MACRO IC 1 Node 5/7 Reg. 3	I2802	$07B42F	MACRO IC 3 Node 5/7 Reg. 3
I1302	$079433	MACRO IC 1 Node 8/10 Reg. 3	I2902	$07B433	MACRO IC 3 Node 8/10 Reg. 3
I1402	$079437	MACRO IC 1 Node 9/11 Reg. 3	I3002	$07B437	MACRO IC 3 Node 9/11 Reg. 3
I1502	$07943B	MACRO IC 1 Node 12/14 Reg. 3	I3102	$07B43B	MACRO IC 3 Node 12/14 Reg. 3
I1602	$07943F	MACRO IC 1 Node 13/15 Reg. 3	I3202	$07B43F	MACRO IC 3 Node 13/15 Reg. 3

Ixx10 Motor xx Power-On Servo Position Address {revised description}

Range:	$000000 - $FFFFFF
Units:	Turbo PMAC or Multiplexer Port Addresses
Default:	$0

Ixx10 controls whether Turbo PMAC reads an absolute position sensor for Motor xx on power-up/reset and/or with the **$*** or **$$*** commands. If an absolute position read is to be done, Ixx10 specifies what register is read for that absolute position data. Ixx95 specifies how the data in this register is interpreted.

If Ixx10 is set to 0, no absolute power-on/reset position read is performed. The power-on/reset position is considered to be zero, even if an absolute sensor reporting a non-zero value is used. Ixx10 should be set to 0 when an incremental position sensor is used; a homing search move is typically then executed to establish a position reference.

If Ixx10 is set to a non-zero value, an absolute position read is performed for Motor xx at power-on/reset, from the register whose location is specified in Ixx10 (unless Bit 2 of Ixx80 is set to 1). This is either the address of a Turbo PMAC register, the multiplexed data address on the Multiplexer Port, or the *number* of the MACRO node on the Turbo PMAC, depending on the setting of Ixx95. The motor's position is set to the value read from the sensor location the Ixx26 "home" offset value.

Ixx10 is used only on power-on/reset, when the **$*** command is issued for the motor, or when the **$$*** command is issued for the coordinate system containing the motor. To get a new value of Ixx10 to take effect, either the **$*** or **$$*** command must be issued, or the value must be stored to non-volatile flash memory with the **SAVE** command, and the board must be reset.

Note: Variable Ixx81 (with Ixx91) performs the same power-on position read function for the phasing (commutation) algorithm.

R/D Converter Read: If Ixx95 is set to a value from $000000 to $070000, or from $800000 to $870000, the address specified in Ixx10 is a Multiplexer Port address. Turbo PMAC will read the absolute position from an ACC-8D Opt 7 Resolver-to-Digital Converter board at that port address, as set by DIP switches on the board. Ixx95 specifies which R/D converter at that address is read, and whether it is treated as a signed or unsigned value.

If Ixx99 is greater than 0, the next R/D converter at that port address is also read as a second geared-down resolver, with Ixx99 setting the gear ratio. If Ixx98 is also greater than 0, the next R/D converter past that one at the same port address is read as a third geared-down resolver, with Ixx98 setting the gear ratio.

In this mode, bits 1 through 7 of Ixx10 match the settings of DIP-switches SW1-2 through SW1-8, respectively, on the ACC-8D Opt 7 R/D Converter board. A CLOSED (ON) switch represents a 0 value; an OPEN (OFF) switch represents a 1 value. Bit 0 and bits 9 through 23 of Ixx10 are always set to 0 in this mode; bit 8 is only set to 1 if all other bits are 0.

The following table shows the common Multiplexer Port addresses that can be used. Note that address 0 uses an Ixx10 value of $000100, because Ixx10=0 disables the absolute position read function.

Ixx10 for ACC-8D Opt. 7 Resolver/Digital Converter
(Ixx95=$000000 - $070000, $800000 - $870000) Addresses are Multiplexer Port Addresses

Board Mux. Addr.	Ixx10	Board Mux. Addr.	Ixx10	Board Mux. Addr.	Ixx10	Board Mux. Addr.	Ixx10
0	$000100	64	$000040	128	$000080	192	$0000C0
8	$000008	72	$000048	136	$000088	200	$0000C8
16	$000010	80	$000050	144	$000090	208	$0000D0
24	$000018	88	$000058	152	$000098	216	$0000D8
32	$000020	96	$000060	160	$0000A0	224	$0000E0
40	$000028	104	$000068	168	$0000A8	232	$0000E8
48	$000030	112	$000070	176	$0000B0	240	$0000F0
56	$000038	120	$000078	184	$0000B8	248	$0000F8

Parallel Word Read: If Ixx95 is set to a value from $080000 to $300000, from $480000 to $700000, from $880000 to $B00000, or from $C80000 to $F00000, the address specified in Ixx10 is a Turbo PMAC memory-I/O address, and Turbo PMAC will read the parallel word at that address. The least significant bit ("count") is expected at bit 0 of the address. The bit width (8 to 48 bits), the format (signed or unsigned), and the register type (X or Y) are determined by Ixx95.

The common sources for this type of read are ACC-14 parallel I/O expansion boards, and the MLDT timer registers. The following tables show the settings of Ixx10 for these devices.

Ixx10 Values for ACC-14D/V Registers
(Ixx95=$080000 to $300000 [unsigned], $880000 to $B00000 [signed])

Register	ACC-14 Select Jumper	Ixx10	Register	ACC-14 Select Jumper	Ixx10
1st ACC-14D/V Port A	E12	$078A00	4th ACC-14D/V Port A	E15	$078D00
1st ACC-14D/V Port B	E12	$078A01	4th ACC-14D/V Port B	E15	$078D01
2nd ACC-14D/V Port A	E13	$078B00	5th ACC-14D/V Port A	E16	$078E00
2nd ACC-14D/V Port B	E13	$078B01	5th ACC-14D/V Port B	E16	$078E01
3rd ACC-14D/V Port A	E14	$078C00	6th ACC-14D/V Port A	E17	$078F00
3rd ACC-14D/V Port B	E14	$078C01	6th ACC-14D/V Port B	E17	$078F01

Ixx10 for PMAC2-Style MLDT Timer Registers (Ixx95=$180000)

Servo IC #	Chan. 1	Chan. 2	Chan. 3	Chan. 4	Notes
0	$078000	$078008	$078010	$078018	1st IC on board PMAC2, 3U stack
1	$078100	$078108	$078010	$078018	2nd IC on board PMAC2, 3U stack
2	$078200	$078208	$078210	$078218	1st ACC-24E2x, 1st IC on 1st ACC-24P/V2
3	$078300	$078308	$078310	$078318	2nd ACC-24E2x, 2nd IC on 1st ACC-24P/V2
4	$079200	$079208	$079210	$079218	3rd ACC-24E2x, 1st IC on 2nd ACC-24P/V2
5	$079300	$079308	$079310	$079318	4th ACC-24E2x, 2nd IC on 2nd ACC-24P/V2
6	$07A200	$07A208	$07A210	$07A218	5th ACC-24E2x, 1st IC on 3rd ACC-24P/V2
7	$07A300	$07A308	$07A310	$07A318	6th ACC-24E2x, 2nd IC on 3rd ACC-24P/V2
8	$07B200	$07B208	$07B210	$07B218	7th ACC-24E2x, 1st IC on 4th ACC-24P/V2
9	$07B300	$07B308	$07B310	$07B318	8th ACC-24E2x, 2nd IC on 4th ACC-24P/V2

It can also be used for registers in the 3U-format ACC-3E1 (for 3U Turbo Stack systems) and ACC-14E (for UMAC Turbo systems) boards. In this case the last hex digit of Ixx91 must be set to a non-zero value to specify the byte-wide bus of these boards. The following tables show Ixx10 values for these boards.

Ixx10 Values for ACC-3E1 Registers in 3U Turbo Stack Systems
(Ixx95=$08000x to $30000x [unsigned], $88000x to $B0000x [signed])

ACC-3E1 Address Jumper	E1	E2	E3	E4
Ixx10 Value	$07880x	$07890x	$078A0x	$078B0x

Ixx10 Values for ACC-14E Registers in UMAC Turbo Systems
(Ixx95=$08000x to $30000x [unsigned], $88000x to $B0000x [signed])

DIP-Switch Setting	SW1-1 ON (0) SW1-2 ON (0)	SW1-1 OFF (1) SW1-2 ON (0)	SW1-1 ON (0) SW1-2 OFF (1)	SW1-1 OFF (1) SW1-2 OFF (1)
SW1-3 ON (0) SW1-4 ON (0)	$078C0x	$078D0x	$078E0x	$078F0x
SW1-3 OFF (1) SW1-4 ON (0)	$079C0x	$079D0x	$079E0x	$079F0x
SW1-3 ON (0) SW1-4 OFF (1)	$07AC0x	$07AD0x	$07AE0x	$07AF0x
SW1-3 OFF (1) SW1-4 OFF (1)	$07BC0x	$07BD0x	$07BE0x	$07BF0x

SW1-5 & 6 must be ON (0). ON means CLOSED; OFF means OPEN.

The final digit, represented by an 'x' in both of these tables, can take a value of 0 to 5, depending on which I/O point on the board is used for the least significant bit (LSB):

Ixx10 Last Hex Digit 'x'	Pin Used for LSB	Pin Used for LSB	Pin Used for LSB
x=0	I/O00-07	I/O48-55	I/O96-103
x=1	I/O08-15	I/O56-63	I/O104-111
x=2	I/O16-23	I/O64-71	I/O112-119
x=3	I/O24-31	I/O72-79	I/O120-127
x=4	I/O32-39	I/O80-87	I/O128-135
x=5	I/O40-47	I/O88-95	I/O136-143

ACC-28 A/D Converter Read: If Ixx95 is set to $310000 or $B10000, the address specified by Ixx10 is a Turbo PMAC 'Y' memory-I/O address, and Turbo PMAC will read the data in the high 16 bits of that address as the absolute position (the LSB – one "count" – is in bit 8). This format is intended for the ACC-28A and ACC-28B A/D converters.

The following table shows the settings of Ixx10 for these registers.

Ixx10 Values for PMAC(1)-Style ADC Registers
(Ixx95=$B10000 for ACC-28A, Ixx95=$310000 for ACC-28B)

Register	PMAC	1st ACC-24P/V	2nd ACC-24P/V	3rd ACC-24P/V	4th ACC-24P/V
ADC1	$078006	$078206	$079206	$07A206	$07B206
ADC2	$078007	$078207	$079207	$07A207	$07B207
ADC3	$07800E	$07820E	$07920E	$07A20E	$07B20E
ADC4	$07800F	$07820F	$07920F	$07A20F	$07B20F
ADC5	$078106	$078306	$079306	$07A306	$07B306
ADC6	$078107	$078307	$079307	$07A307	$07B307
ADC7	$07810E	$07830E	$07930E	$07A30E	$07B30E
ADC8	$07810F	$07830F	$07930F	$07A30F	$07B30F

Ixx10 Values for PMAC2-Style ADC Registers using ACC-28B
(Ixx95=$B10000)

Register	PMAC2	1st ACC-24x2	2nd ACC-24x2	3rd ACC-24x2	4th ACC-24x2
ADC 1A	$078005	$078205	$079205	$07A205	$07B205
ADC 1B	$078006	$078206	$079206	$07A206	$07B206
ADC 2A	$07800D	$07820D	$07920D	$07A20D	$07B20D
ADC 2B	$07800E	$07820E	$07920E	$07A20E	$07B20E
ADC 3A	$078015	$078215	$079215	$07A215	$07B215
ADC 3B	$078016	$078216	$079216	$07A216	$07B216
ADC 4A	$07801D	$07821D	$07921D	$07A21D	$07B21D
ADC 4B	$07801E	$07821E	$07921E	$07A21E	$07B21E
ADC 5A	$078105	$078305	$079305	$07A305	$07B305
ADC 5B	$078106	$078306	$079306	$07A306	$07B306
ADC 6A	$07810D	$07830D	$07930D	$07A30D	$07B30D
ADC 6B	$07810E	$07830E	$07930E	$07A30E	$07B30E
ADC 7A	$078115	$078315	$079315	$07A315	$07B315
ADC 7B	$078116	$078316	$079316	$07A316	$07B316
ADC 8A	$07811D	$07831D	$07931D	$07A31D	$07B31D
ADC 8B	$07811E	$07831E	$07931E	$07A31E	$07B31E

Ixx10 Values for ACC-28E Registers in UMAC Turbo Systems
(Ixx95=$B10000)

DIP-Switch Setting	SW1-1 ON (0) SW1-2 ON (0)	SW1-1 OFF (1) SW1-2 ON (0)	SW1-1 ON (0) SW1-2 OFF (1)	SW1-1 OFF (1) SW1-2 OFF (1)
SW1-3 ON (0) SW1-4 ON (0)	$078C0x	$078D0x	$078E0x	$078F0x
SW1-3 OFF (1) SW1-4 ON (0)	$079C0x	$079D0x	$079E0x	$079F0x
SW1-3 ON (0) SW1-4 OFF (1)	$07AC0x	$07AD0x	$07AE0x	$07AF0x
SW1-3 OFF (1) SW1-4 OFF (1)	$07BC0x	$07BD0x	$07BE0x	$07BF0x

SW1-5 & 6 must be ON (0). ON means CLOSED; OFF means OPEN.

The final digit, represented by an 'x' in both of these tables, can take a value of 0 to 3, depending on which ADC channel on the ACC-28E is used (x = Channel - 1).

Sanyo Absolute Encoder Read: If Ixx95 is set to $320000 or $B20000, the address specified in Ixx10 is a Turbo PMAC memory-I/O address, and Turbo PMAC will read the absolute position from an ACC-49 Sanyo Absolute Encoder Converter board at that address. Ixx95 specifies whether this position is treated as a signed or unsigned value.

The following table shows the possible settings of Ixx10 for ACC-49 Sanyo Absolute Encoder Converter boards.

Ixx10 Values for ACC-49 Sanyo Absolute Encoder Converter (Ixx95=$320000, $B20000)
Addresses are Turbo PMAC Memory-I/O Addresses

Enc. # on Board	Ixx10 for E1 ON	Ixx10 for E2 ON	Ixx10 for E3 ON	Enc. # on Board	Ixx10 for E4 ON	Ixx10 for E5 ON	Ixx10 for E6 ON
Enc. 1	$078A00	$078B00	$078C00	Enc. 3	$078D00	$078E00	$078F00
Enc. 2	$078A04	$078B04	$078C04	Enc. 4	$078D04	$078E04	$078F04

Yaskawa Absolute Encoder Read: If Ixx95 is set to $710000 or $F10000, the address specified in Ixx10 is a Multiplexer Port address, and Turbo PMAC will read the absolute position from an ACC-8D Opt 9 Yaskawa Absolute Encoder Converter board at that port address, as set by DIP switches on the board. Ixx95 specifies whether it is treated as a signed or unsigned value.

In this mode, bits 3 through 7 of Ixx10 match the settings of DIP switches SW1-1 through SW1-5, respectively, of the ACC-8D Opt 9 Yaskawa converter board. A CLOSED switch represents a bit value of 0; an OPEN switch represents a bit value of 1. Bits 0 through 2, and bits 8 though 23, of Ixx10 are always set to 0 in this mode.

The following table shows the Multiplexer Port addresses that can be used and the matching values of Ixx10. Note that address 0 uses an Ixx10 value of $000100, because Ixx10=0 disables the absolute position read function.

Ixx10 for ACC-8D Opt. 9 Yaskawa Absolute Encoder (Ixx95=$710000, $F10000)
Addresses are Multiplexer Port Addresses

Board Mux. Addr.	Ixx10 for Enc. 1	Ixx10 for Enc. 2	Ixx10 for Enc. 3	Ixx10 for Enc. 4	Board Mux. Addr.	Ixx10 for Enc. 1	Ixx10 for Enc. 2	Ixx10 for Enc. 3	Ixx10 for Enc. 4
0	$000100	$000002	$000004	$000006	128	$000080	$000082	$000084	$000086
8	$000008	$00000A	$00000C	$00000E	136	$000088	$00008A	$00008C	$00008E
16	$000010	$000012	$000014	$000016	144	$000090	$000092	$000094	$000096
24	$000018	$00001A	$00001C	$00001E	152	$000098	$00009A	$00009C	$00009E
32	$000020	$000022	$000024	$000026	160	$0000A0	$0000A2	$0000A4	$0000A6
40	$000028	$00002A	$00002C	$00002E	168	$0000A8	$0000AA	$0000AC	$0000AE
48	$000030	$000032	$000034	$000036	176	$0000B0	$0000B2	$0000B4	$0000B6
56	$000038	$00003A	$00003C	$00003E	184	$0000B8	$0000BA	$0000BC	$0000BE
64	$000040	$000042	$000044	$000046	192	$0000C0	$0000C2	$0000C4	$0000C6
72	$000048	$00004A	$00004C	$00004E	200	$0000C8	$0000CA	$0000CC	$0000CE
80	$000050	$000052	$000054	$000056	208	$0000D0	$0000D2	$0000D4	$0000D6
88	$000058	$00005A	$00005C	$00005E	216	$0000D8	$0000DA	$0000DC	$0000DE
96	$000060	$000062	$000064	$000066	224	$0000E0	$0000E2	$0000E4	$0000E6
104	$000068	$00006A	$00006C	$00006E	232	$0000E8	$0000EA	$0000EC	$0000EE
112	$000070	$000072	$000074	$000076	240	$0000F0	$0000F2	$0000F4	$0000F6
120	$000078	$00007A	$00007C	$00007E	248	$0000F8	$0000FA	$0000FC	$0000FE

MACRO Absolute Position Read: If Ixx95 contains a value from $720000 to $740000, or from $F20000 to $F40000, the value specified in Ixx10 is a MACRO node number, and Turbo PMAC will obtain the absolute power-on position through the MACRO ring. Ixx95 specifies what type of position data is used, and whether it is treated as a signed or unsigned value.

The MACRO node number is specified in the last two hex digits of Ixx10. The second-to-last digit specifies the MACRO IC number 0 to 3 (1, 2, and 3 exist only on Ultralite versions of the Turbo PMAC2, or a UMAC Turbo with ACC-5E). Note that the MACRO IC number on the Turbo PMAC does not necessarily match the ring master number for that IC, although it often will. The last digit specifies the MACRO node number 0 to 15 (0 to F hex) in that IC. This function is only supported in nodes 0, 1, 4, 5, 8, 9, 12 (C), and 13 (D).

The following table shows the required values of Ixx10 for all of the MACRO nodes that can be used. Note that MACRO IC 0 Node 0 uses an Ixx10 value of $000100, because Ixx10=0 disables the absolute position read function.

Ixx10 for MACRO Absolute Position Reads
(Ixx95=$720000 - $740000, $F20000 - $F40000)
Addresses are MACRO Node Numbers

MACRO Node Number	Ixx10 for MACRO IC 0	Ixx10 for MACRO IC 1	Ixx10 for MACRO IC 2	Ixx10 for MACRO IC 3
0	$000100	$000010	$000020	$000030
1	$000001	$000011	$000021	$000031
4	$000004	$000014	$000024	$000034
5	$000005	$000015	$000025	$000035
8	$000008	$000018	$000028	$000038
9	$000009	$000019	$000029	$000039
12	$00000C	$00001C	$00002C	$00003C
13	$00000D	$00001D	$00002D	$00003D

If obtaining the absolute position through a Delta Tau MACRO Station or equivalent, MACRO Station setup variable MI11x for the matching node must be set properly to obtain the type of information desired.

Ixx24 Motor xx Flag Mode Control

Range: $000000 - $FFFFFF
Units: none
Default: $000000 (Turbo PMAC(1) boards)
$000001 (non-Ultralite Turbo PMAC2 boards)
$840001 (Turbo PMAC2 Ultralite boards)

Ixx24 specifies how the flag information in the register(s) specified by Ixx25, Ixx42, and Ixx43 is used. Ixx24 is a set of 24 individual control bits – bits 0 to 23. Currently bits 0 and 11 to 23 are used.

It is easier to specify this parameter in hexadecimal form. With I9 at 2 or 3, the value of this variable will be reported back to the host in hexadecimal form.

Bit 0: Flag Register Type Bit: If bit 0 is set to zero, the Turbo PMAC expects the flag registers to be in the format of a PMAC(1)-style Servo IC. Bit 0 should be set to 0 for any flags on-board a Turbo PMAC(1), an ACC-24P, or an ACC24V.

If bit 0 is set to one, the Turbo PMAC expects the flag registers to be in the format of a PMAC2-style Servo IC. Bit 0 should be set to 1 for any flag register on-board a Turbo PMAC2, an ACC-24P2, an ACC-24V2, an ACC-24E2, or coming from a MACRO Station.

If multiple flag registers are specified by non-zero settings of Ixx42 and/or Ixx43, all registers must be of the same format.

Bit 11: Capture with High-Resolution Feedback Bit: If bit 11 is set to zero when hardware position capture is used in a triggered move such as a homing-search move, the captured data (whether whole-count only or including sub-count data) is processed to match servo feedback of "normal" resolution (5 bits of fractional count data per hardware whole count). This setting is appropriate for digital quadrature feedback or for "low-resolution" interpolation of a sinusoidal encoder.

If bit 11 (value $800, or 2,048) is set to one when hardware position capture is used in a triggered move, the captured data (whether whole-count only or including sub-count data) is processed to match servo feedback of "high" resolution (10 bits of fractional count data per hardware whole count). This setting is appropriate for "high-resolution" interpolation of a sinusoidal encoder through an ACC-51x interpolator.

Bit 12: Sub-Count Capture Enable Bit: If bit 12 is set to zero when hardware position capture is used in a triggered move such as a homing-search move, only the whole-count captured position register is used to establish the trigger position. This setting must be used with PMAC(1)-style Servo ICs, and with PMAC2-style Servo ICs older than Revision "D" (Revision "D" ICs started shipping in early 2002).

If bit 12 (value $1000, or 4,096) is set to one when hardware position capture is used in a triggered move, both the whole-count captured position register and the estimated sub-count position register are used to establish the trigger position. A PMAC2-style Servo IC of Revision "D" or newer must be used for this mode, and I7mn9 for the channel used must be set to 1 to enable the hardware sub-count estimation. This setting is typically used for registration or probing triggered moves with interpolated sinusoidal encoder feedback. (Even with interpolated sinusoidal encoder feedback, homing search moves will probably be done without sub-count captured data, to force a home position referenced to one of the four "zero-crossing" positions of the sine/cosine signals.)

Bit 13 Error Saturation Control Bit: If bit 13 is set to zero, when the motor's following error exceeds the Ixx67 position-error limit, the error is simply truncated by the limit parameter.
If bit 13 (value $2000, or 8,192) is set to 1, when the motor's following error exceeds the Ixx67 position-error limit, the excess is put in the "master position" register for the motor, so it is eventually recoverable.

Bit 14: Continue on Desired Position Limit Bit: If bit 14 is set to zero when desired position limits are enabled (bit 15=1), and desired position within the lookahead buffer exceeds a position limit, Turbo PMAC will stop execution of the program at the point where the motor reaches the limit.

If bit 14 (value $4000, or 16,384) is set to one when desired position limits are enabled (bit 15=1) (e.g. I224=$C000), and desired position within the lookahead buffer exceeds a position limit, Turbo PMAC will continue execution of the program past the point where the motor reaches the limit, but will not let the desired motor position exceed the limit.

Bit 15: Desired Position Limit Enable Bit: If bit 15 is set to zero, Turbo PMAC does not check to see whether the desired position for this motor exceeds software overtravel limits.
If bit 15 (value $8000, or 32,768) is set to one (e.g. I324=$8001), Turbo PMAC will check desired position values for this motor against the software overtravel limits as set by Ixx13, Ixx14, and Ixx41. If inside the special lookahead buffer, Turbo PMAC will either come to a controlled stop along the path at the point where the desired position reaches the limit, or continue the program with desired position saturated at the limit, depending on the setting of bit 14. If not inside the special lookahead buffer, Turbo PMAC will issue an Abort command when it sees that the desired position has exceeded a position limit.

Bit 16: Amplifier Enable Use Bit: With bit 19 equal to zero – the normal case – the AENAn output is used as an amplifier-enable line: off when the motor is "killed", on when it is enabled. If bit 16 (value $10000, or 65,536) is set to one (e.g. I1924=$10001), this output is not used as an amplifier-enable line. On PMAC(1)-style channels, it could then be used as a direction output for magnitude and direction command format if Ixx96 is set to 1. Also, by assigning an M-variable to the AENAn output bit, general-purpose use of the this output is possible on either Turbo PMAC(1) or PMAC2 if this bit is set.

Bit 17: Overtravel Limit Use Bit: With bit 17 equal to zero – the normal case – the two hardware overtravel limit inputs must read 0 (drawing current) to permit commanded motion in the appropriate direction. If there are not actual (normally closed or normally conducting) limit switches, the inputs must be hardwired to ground.

If bit 17 (value $20000, or 131,072) is set to one (e.g. I1924=$20000), Motor xx does not use these inputs as overtravel limits. This can be done temporarily, as when using a limit as a homing flag. If the hardware overtravel limit function is not used at all, these inputs can be used as general-purpose inputs by assigning M-variables to them.

Bits 18 and 19: MACRO Node Use Bits: Bits 18 (value $40000, or 262,144) and 19 (value ($80000, or 524,288) of Ixx24 specify what flag information is connected directly to Turbo PMAC hardware channels, and what information comes through the MACRO ring into a MACRO auxiliary register. The following table shows the possible settings of these two bits and what they specify:

Bit 19	Bit 18	Capture Flags	Amp Flags	Limit Flags
0	0	Direct	Direct	(don't care)
0	1	Thru MACRO	Thru MACRO	(don't care)
1	0	Direct	Thru MACRO	(don't care)
1	1	Thru MACRO	Direct	(don't care)

If the amplifier flags are connected through the MACRO ring, bit 23 of Ixx24 must be set to 1 to designate a high-true amplifier fault, which is the MACRO standard. When using a MACRO auxiliary register for the flags, Ixx25, Ixx42, or Ixx43 should contain the address of a holding register in RAM, not the actual MACRO register. Refer to the descriptions of those variables for a list of the holding register addresses. Turbo PMAC firmware automatically copies between the holding registers and the MACRO registers as enabled by I70, I72, I74 and I76, for MACRO ICs 0, 1, 2, and 3, respectively. I71, I73, I75, and I77 must be set properly to determine whether the Type 0 or Type 1 MACRO protocol is being used on the particular node (all Delta Tau products use Type 1).

Bit 20: Amplifier Fault Use Bit: If bit 20 of Ixx24 is 0, the amplifier-fault input function through the FAULTn input is enabled. If bit 20 (value $100000, or 1,048,576) is 1 (e.g. I1924=$100000), this function is disabled. General-purpose use of this input is then possible by assigning an M-variable to the input.

Bits 21 & 22: Action-on-Fault Bits: Bits 21 (value $200000, or 2,097,152) and 22 (value $400000, or 4,194,304) of Ixx24 control what action is taken on an amplifier fault for the motor, or on exceeding the fatal following error limit (as set by Ixx11) for the motor:

Bit 22	Bit 21	Function
Bit 22=0	Bit 21=0:	Kill all PMAC motors
Bit 22=0	Bit 21=1:	Kill all motors in same coordinate system
Bit 22=1	Bit 21=0:	Kill only this motor
Bit 22=1	Bit 21=1:	(Reserved for future use)

Regardless of the setting of these bits, a program running in the coordinate system of the offending motor will be halted on an amplifier fault or the exceeding of a fatal following error.

Bit 23: Amplifier-Fault Polarity Bit: Bit 23 (value $800000, or 8,388,608) of Ixx24 controls the polarity of the amplifier-fault input. A zero in this bit specifies that a zero read in the fault bit means a fault; a one in this bit specifies that a one read in the fault bit means a fault. The actual state of the input circuitry for a fault depends on the actual interface circuitry used. If a Delta Tau-provided optically isolated fault interface is used, when the fault driver from the amplifier is drawing current through the isolator, either sinking or sourcing, the fault bit will read as zero; when it is not drawing current through the isolator, the fault bit will read as one.

In both the standard direct-PWM interface and the standard MACRO interface, bit 23 should be set to one, to specify that a one in the fault bit means a fault. (The actual polarity of the signal into the remote MACRO Station is programmable at the station).

Bit 23 is only used if bit 20 of Ixx24 is set to 0, telling Turbo PMAC to use the amplifier fault input.

Ixx25 Motor xx Flag Address {revised description}

Range: $000000 - $FFFFFF
Units: Turbo PMAC Addresses
Default:

Turbo PMAC(1) Ixx25 Defaults

Ixx25	Value	Register	Ixx25	Value	Register
I125	$078000	PMAC Flag Set 1	I1725	$079200	2nd ACC-24P/V Flag Set 1
I225	$078004	PMAC Flag Set 2	I1825	$079204	2nd ACC-24P/V Flag Set 2
I325	$078008	PMAC Flag Set 3	I1925	$079208	2nd ACC-24P/V Flag Set 3
I425	$07800C	PMAC Flag Set 4	I2025	$07920C	2nd ACC-24P/V Flag Set 4
I525	$078100	PMAC Flag Set 5	I2125	$079300	2nd ACC-24P/V Flag Set 5
I625	$078104	PMAC Flag Set 6	I2225	$079304	2nd ACC-24P/V Flag Set 6
I725	$078108	PMAC Flag Set 7	I2325	$079308	2nd ACC-24P/V Flag Set 7
I825	$07810C	PMAC Flag Set 8	I2425	$07930C	2nd ACC-24P/V Flag Set 8
I925	$078200	1st ACC-24P/V Flag Set 1	I2525	$07A200	3rd ACC-24P/V Flag Set 1
I1025	$078204	1st ACC-24P/V Flag Set 2	I2625	$07A204	3rd ACC-24P/V Flag Set 2
I1125	$078208	1st ACC-24P/V Flag Set 3	I2725	$07A208	3rd ACC-24P/V Flag Set 3
I1225	$07820C	1st ACC-24P/V Flag Set 4	I2825	$07A20C	3rd ACC-24P/V Flag Set 4
I1325	$078300	1st ACC-24P/V Flag Set 5	I2925	$07A300	3rd ACC-24P/V Flag Set 5
I1425	$078304	1st ACC-24P/V Flag Set 6	I3025	$07A304	3rd ACC-24P/V Flag Set 6
I1525	$078308	1st ACC-24P/V Flag Set 7	I3125	$07A308	3rd ACC-24P/V Flag Set 7
I1625	$07830C	1st ACC-24P/V Flag Set 8	I3225	$07A30C	3rd ACC-24P/V Flag Set 8

Turbo PMAC2 Ixx25 Defaults

Ixx25	Value	Register	Ixx25	Value	Register
I125	$078000	PMAC2 Flag Set 1	I1725	$079200	2nd ACC-24P/V2 Flag Set 1
I225	$078008	PMAC2 Flag Set 2	I1825	$079208	2nd ACC-24P/V2 Flag Set 2
I325	$078010	PMAC2 Flag Set 3	I1925	$079210	2nd ACC-24P/V2 Flag Set 3
I425	$078018	PMAC2 Flag Set 4	I2025	$079218	2nd ACC-24P/V2 Flag Set 4
I525	$078100	PMAC2 Flag Set 5	I2125	$079300	2nd ACC-24P/V2 Flag Set 5
I625	$078108	PMAC2 Flag Set 6	I2225	$079308	2nd ACC-24P/V2 Flag Set 6
I725	$078110	PMAC2 Flag Set 7	I2325	$079310	2nd ACC-24P/V2 Flag Set 7
I825	$078118	PMAC2 Flag Set 8	I2425	$079318	2nd ACC-24P/V2 Flag Set 8
I925	$078200	1st ACC-24P/V2 Flag Set 1	I2525	$07A200	3rd ACC-24P/V2 Flag Set 1
I1025	$078208	1st ACC-24P/V2 Flag Set 2	I2625	$07A208	3rd ACC-24P/V2 Flag Set 2
I1125	$078210	1st ACC-24P/V2 Flag Set 3	I2725	$07A210	3rd ACC-24P/V2 Flag Set 3
I1225	$078218	1st ACC-24P/V2 Flag Set 4	I2825	$07A218	3rd ACC-24P/V2 Flag Set 4
I1325	$078300	1st ACC-24P/V2 Flag Set 5	I2925	$07A300	3rd ACC-24P/V2 Flag Set 5
I1425	$078308	1st ACC-24P/V2 Flag Set 6	I3025	$07A308	3rd ACC-24P/V2 Flag Set 6
I1525	$078310	1st ACC-24P/V2 Flag Set 7	I3125	$07A310	3rd ACC-24P/V2 Flag Set 7
I1625	$078318	1st ACC-24P/V2 Flag Set 8	I3225	$07A318	3rd ACC-24P/V2 Flag Set 8

Turbo PMAC2 Ultralite Ixx25 Defaults

Ixx25	Value	Register	Ixx25	Value	Register
I125	$003440	MACRO Flag Register Set 0	I1725	$003460	MACRO Flag Register Set 32
I225	$003441	MACRO Flag Register Set 1	I1825	$003461	MACRO Flag Register Set 33
I325	$003444	MACRO Flag Register Set 4	I1925	$003464	MACRO Flag Register Set 36
I425	$003445	MACRO Flag Register Set 5	I2025	$003465	MACRO Flag Register Set 37
I525	$003448	MACRO Flag Register Set 8	I2125	$003468	MACRO Flag Register Set 40
I625	$003449	MACRO Flag Register Set 9	I2225	$003469	MACRO Flag Register Set 41
I725	$00344C	MACRO Flag Register Set 12	I2325	$00346C	MACRO Flag Register Set 44
I825	$00344D	MACRO Flag Register Set 13	I2425	$00346D	MACRO Flag Register Set 45
I925	$003450	MACRO Flag Register Set 16	I2525	$003470	MACRO Flag Register Set 48
I1025	$003451	MACRO Flag Register Set 17	I2625	$003471	MACRO Flag Register Set 49
I1125	$003454	MACRO Flag Register Set 20	I2725	$003474	MACRO Flag Register Set 52
I1225	$003455	MACRO Flag Register Set 21	I2825	$003475	MACRO Flag Register Set 53
I1325	$003458	MACRO Flag Register Set 24	I2925	$003478	MACRO Flag Register Set 56
I1425	$003459	MACRO Flag Register Set 25	I3025	$003479	MACRO Flag Register Set 57
I1525	$00345C	MACRO Flag Register Set 28	I3125	$00347C	MACRO Flag Register Set 60
I1625	$00345D	MACRO Flag Register Set 29	I3225	$00347D	MACRO Flag Register Set 61

UMAC Turbo Ixx25 Defaults

Ixx02	Value	Register	Ixx02	Value	Register
I102	$078200	1st ACC-24E2x (IC 2) Flag Set 1	I1702	$07A200	5th ACC-24E2x (IC 6) Flag Set 1
I202	$078208	1st ACC-24E2x (IC 2) Flag Set 2	I1802	$07A208	5th ACC-24E2x (IC 6) Flag Set 2
I302	$078210	1st ACC-24E2x (IC 2) Flag Set 3	I1902	$07A210	5th ACC-24E2x (IC 6) Flag Set 3
I402	$078218	1st ACC-24E2x (IC 2) Flag Set 4	I2002	$07A218	5th ACC-24E2x (IC 6) Flag Set 4
I502	$078300	2nd ACC-24E2x (IC 3) Flag Set 1	I2102	$07A300	6th ACC-24E2x (IC 7) Flag Set 1
I602	$078308	2nd ACC-24E2x (IC 3) Flag Set 2	I2202	$07A308	6th ACC-24E2x (IC 7) Flag Set 2
I702	$078310	2nd ACC-24E2x (IC 3) Flag Set 3	I2302	$07A310	6th ACC-24E2x (IC 7) Flag Set 3
I802	$078318	2nd ACC-24E2x (IC 3) Flag Set 4	I2402	$07A318	6th ACC-24E2x (IC 7) Flag Set 4
I902	$079200	3rd ACC-24E2x (IC 4) Flag Set 1	I2502	$07B200	7th ACC-24E2x (IC 8) Flag Set 1
I1002	$079208	3rd ACC-24E2x (IC 4) Flag Set 2	I2602	$07B208	7th ACC-24E2x (IC 8) Flag Set 2
I1102	$079210	3rd ACC-24E2x (IC 4) Flag Set 3	I2702	$07B210	7th ACC-24E2x (IC 8) Flag Set 3
I1202	$079218	3rd ACC-24E2x (IC 4) Flag Set 4	I2802	$07B218	7th ACC-24E2x (IC 8) Flag Set 4
I1302	$079300	4th ACC-24E2x (IC 5) Flag Set 1	I2902	$07B300	8th ACC-24E2x (IC 9) Flag Set 1
I1402	$079308	4th ACC-24E2x (IC 5) Flag Set 2	I3002	$07B308	8th ACC-24E2x (IC 9) Flag Set 2
I1502	$079310	4th ACC-24E2x (IC 5) Flag Set 3	I3102	$07B310	8th ACC-24E2x (IC 9) Flag Set 3
I1602	$079318	4th ACC-24E2x (IC 5) Flag Set 4	I3202	$07B318	8th ACC-24E2x (IC 9) Flag Set 4

Ixx25 tells Turbo PMAC what registers it will access for its position-capture flags, and possibly its overtravel-limit input flags and amplifier enable/fault flags, for Motor xx. If Ixx42 is set to 0, Ixx25 specifies the address of the amplifier flags; if Ixx42 is set to a non-zero value, Ixx42 specifies the address of the amplifier flags. If Ixx43 is set to 0, Ixx25 specifies the address of the overtravel limit flags; if Ixx43 if set to a non-zero value, Ixx43 specifies the address of the overtravel limit flags. Variable Ixx24 tells which of the flags from the specified register(s) are to be used, and how they are to be used.

The addresses for the standard flag registers are given in the default table, above. The following tables show settings by register if you wish to change from the default.

Ixx25 Addresses for PMAC(1)-Style Servo ICs

Servo IC #	Chan. 1	Chan. 2	Chan. 3	Chan. 4	Notes
0	$078000	$078004	$078008	$07800C	1st IC on board PMAC
1	$078100	$078104	$078108	$07810C	2nd IC on board PMAC
2	$078200	$078204	$078208	$07820C	1st IC on 1st ACC-24P/V

3	$078300	$078304	$078308	$07830C	2nd IC on 1st ACC-24P/V
4	$079200	$079204	$079208	$07920C	1st IC on 2nd ACC-24P/V
5	$079300	$079304	$079308	$07930C	2nd IC on 2nd ACC-24P/V
6	$07A200	$07A204	$07A208	$07A20C	1st IC on 3rd ACC-24P/V
7	$07A300	$07A304	$07A308	$07A30C	2nd IC on 3rd ACC-24P/V
8	$07B200	$07B204	$07B208	$07B20C	1st IC on 4th ACC-24P/V
9	$07B300	$07B304	$07B308	$07B30C	2nd IC on 4th ACC-24P/V

Bit 0 of Ixx24 must be set to 0 to use PMAC(1)-style Servo ICs.

Ixx25 Addresses for PMAC2-Style Servo ICs

Servo IC #	Chan. 1	Chan. 2	Chan. 3	Chan. 4	Notes
0	$078000	$078008	$078010	$078018	1st IC on board PMAC2, 3U stack
1	$078100	$078108	$078010	$078018	2nd IC on board PMAC2, 3U stack
2	$078200	$078208	$078210	$078218	1st ACC-24E2x, 1st IC on 1st ACC-24P/V2
3	$078300	$078308	$078310	$078318	2nd ACC-24E2x, 2nd IC on 1st ACC-24P/V2
4	$079200	$079208	$079210	$079218	3rd ACC-24E2x, 1st IC on 2nd ACC-24P/V2
5	$079300	$079308	$079310	$079318	4th ACC-24E2x, 2nd IC on 2nd ACC-24P/V2
6	$07A200	$07A208	$07A210	$07A218	5th ACC-24E2x, 1st IC on 3rd ACC-24P/V2
7	$07A300	$07A308	$07A310	$07A318	6th ACC-24E2x, 2nd IC on 3rd ACC-24P/V2
8	$07B200	$07B208	$07B210	$07B218	7th ACC-24E2x, 1st IC on 4th ACC-24P/V2
9	$07B300	$07B308	$07B310	$07B318	8th ACC-24E2x, 2nd IC on 4th ACC-24P/V2

Bit 0 of Ixx24 must be set to 1 to use PMAC2-style Servo ICs.

Ixx25 Addresses for MACRO Flag Holding Registers

IC Node #	MACRO IC 1	MACRO IC 2	MACRO IC 3	MACRO IC 4	Notes
0	$003440	$003450	$003460	$003470	MACRO Flag Register Sets 0, 16, 32, 48
1	$003441	$003451	$003461	$003471	MACRO Flag Register Sets 1, 17, 33, 49
4	$003444	$003454	$003464	$003474	MACRO Flag Register Sets 4, 20, 36, 52
5	$003445	$003455	$003465	$003475	MACRO Flag Register Sets 5, 21, 37, 53
8	$003448	$003458	$003468	$003478	MACRO Flag Register Sets 8, 24, 40, 56
9	$003449	$003459	$003469	$003479	MACRO Flag Register Sets 9, 25, 41, 57
12	$00344C	$00345C	$00346C	$00347C	MACRO Flag Register Sets 12, 28, 44, 60
13	$00344D	$00345D	$00346D	$00347D	MACRO Flag Register Sets 13, 29, 45, 61

Bit 0 of Ixx24 must be set to 1 to use MACRO flag holding registers

Bits 18 and 19 of Ixx24 specify what flag information comes directly into Turbo PMAC and what comes through the MACRO ring. The following table explains the possible settings:

Bit 19	Bit 18	Capture Flags	Amp Flags	Limit Flags
0	0	Direct	Direct	(don't care)
0	1	Thru MACRO	Thru MACRO	(don't care)
1	0	Direct	Thru MACRO	(don't care)
1	1	Thru MACRO	Direct	(don't care)

Typically, the position-capture flags will be on the same hardware channel as the position feedback encoder for the motor. If you wish to use the hardware-captured position for a Turbo PMAC triggered-move function such as a homing search move, Ixx25 must specify flags of the same hardware channel as the position feedback encoder specified with Ixx03 through the encoder conversion table, whether digital quadrature feedback, or interpolated sinusoidal feedback.

In the case of sinusoidal-encoder feedback through an ACC-51x high-resolution interpolator, if hardware position-capture capability is desired, the position-capture flags will be specified as being on the ACC-51x using Ixx25, and the amplifier flags will be specified as being on the output channel using Ixx42; the overtravel-limit flags will probably be specified as being on the same channel as the outputs, using Ixx43.

For the position-capture function, variables I7mn2 and I7mn3 for Servo IC *m* Channel *n* of the channel selected (or node-specific variables MI912 and MI913 on a MACRO Station) specify which edges of which signal(s) for the channel will cause the position-capture trigger.

The overtravel-limit inputs specified by Ixx25 or Ixx43 must read as 0 in order for Motor xx to be able to command movement in the direction of the limit unless bit 17 of Ixx24 is set to 1 to disable their action. With Delta Tau interface circuitry with optical isolation on the flags, this means that the switches must be drawing current through the opto-isolators, whether sinking or sourcing.

Whether the address of the amplifier flags is specified with Ixx25 or Ixx42, the polarity of the amplifier-fault input is determined by bit 23 of Ixx24 and the polarity of the amplifier-enable output must be determined with the hardware interface.

Ixx42 Motor xx Amplifier Flag Address

Range: $000000 - $FFFFFF
Units: Turbo PMAC Addresses
Default: $0

Ixx42, if set to a non-zero value, specifies the address of the amplifier-enable output flag and amplifier-fault input flag, independently of position-capture flags and overtravel-limit flags, for Motor xx. If Ixx42 is set to 0, Ixx25 specifies the address of the amplifier flags as well as the position-capture flags, and possibly the overtravel-limit flags, for Motor xx. This maintains backward compatibility with older firmware revisions in which Ixx42 was not implemented.

Whether the address of the amplifier flags is specified with Ixx25 or Ixx42, the polarity of the amplifier-fault input is determined by bit 23 of Ixx24 and the polarity of the amplifier-enable output must be determined with the hardware interface.

If amplifier flags are specified separately using Ixx42, they must use the same type of ICs as does Ixx25, those specified by bit 0 of Ixx24.

Bits 18 and 19 of Ixx24 specify whether the amplifier flags and the capture flags are connected directly to Turbo PMAC circuitry, or interface to it through the MACRO ring as shown in the following table:

Bit 19	Bit 18	Capture Flags	Amp Flags
0	0	Direct	Direct
0	1	Thru MACRO	Thru MACRO
1	0	Direct	Thru MACRO
1	1	Thru MACRO	Direct

The following tables show the standard addresses that can be used for Ixx42.

Ixx42 Addresses for PMAC(1)-Style Servo ICs

Servo IC #	Chan. 1	Chan. 2	Chan. 3	Chan. 4	Notes
0	$078000	$078004	$078008	$07800C	1st IC on board PMAC
1	$078100	$078104	$078108	$07810C	2nd IC on board PMAC
2	$078200	$078204	$078208	$07820C	1st IC on 1st ACC-24P/V
3	$078300	$078304	$078308	$07830C	2nd IC on 1st ACC-24P/V
4	$079200	$079204	$079208	$07920C	1st IC on 2nd ACC-24P/V
5	$079300	$079304	$079308	$07930C	2nd IC on 2nd ACC-24P/V
6	$07A200	$07A204	$07A208	$07A20C	1st IC on 3rd ACC-24P/V
7	$07A300	$07A304	$07A308	$07A30C	2nd IC on 3rd ACC-24P/V
8	$07B200	$07B204	$07B208	$07B20C	1st IC on 4th ACC-24P/V
9	$07B300	$07B304	$07B308	$07B30C	2nd IC on 4th ACC-24P/V

Bit 0 of Ixx24 must be set to 0 to use PMAC(1)-style Servo ICs.

Ixx42 Addresses for PMAC2-Style Servo ICs

Servo IC #	Chan. 1	Chan. 2	Chan. 3	Chan. 4	Notes
0	$078000	$078008	$078010	$078018	1st IC on board PMAC2, 3U stack
1	$078100	$078108	$078010	$078018	2nd IC on board PMAC2, 3U stack
2	$078200	$078208	$078210	$078218	1st ACC-24E2x, 1st IC on 1st ACC-24P/V2
3	$078300	$078308	$078310	$078318	2nd ACC-24E2x, 2nd IC on 1st ACC-24P/V2
4	$079200	$079208	$079210	$079218	3rd ACC-24E2x, 1st IC on 2nd ACC-24P/V2
5	$079300	$079308	$079310	$079318	4th ACC-24E2x, 2nd IC on 2nd ACC-24P/V2
6	$07A200	$07A208	$07A210	$07A218	5th ACC-24E2x, 1st IC on 3rd ACC-24P/V2
7	$07A300	$07A308	$07A310	$07A318	6th ACC-24E2x, 2nd IC on 3rd ACC-24P/V2
8	$07B200	$07B208	$07B210	$07B218	7th ACC-24E2x, 1st IC on 4th ACC-24P/V2
9	$07B300	$07B308	$07B310	$07B318	8th ACC-24E2x, 2nd IC on 4th ACC-24P/V2

Bit 0 of Ixx24 must be set to 1 to use PMAC2-style Servo ICs.

Ixx42 Addresses for MACRO Flag Holding Registers

IC Node #	MACRO IC 1	MACRO IC 2	MACRO IC 3	MACRO IC 4	Notes
0	$003440	$003450	$003460	$003470	MACRO Flag Register Sets 0, 16, 32, 48
1	$003441	$003451	$003461	$003471	MACRO Flag Register Sets 1, 17, 33, 49
4	$003444	$003454	$003464	$003474	MACRO Flag Register Sets 4, 20, 36, 52
5	$003445	$003455	$003465	$003475	MACRO Flag Register Sets 5, 21, 37, 53
8	$003448	$003458	$003468	$003478	MACRO Flag Register Sets 8, 24, 40, 56
9	$003449	$003459	$003469	$003479	MACRO Flag Register Sets 9, 25, 41, 57
12	$00344C	$00345C	$00346C	$00347C	MACRO Flag Register Sets 12, 28, 44, 60
13	$00344D	$00345D	$00346D	$00347D	MACRO Flag Register Sets 13, 29, 45, 61

Bit 0 of Ixx24 must be set to 1 to use MACRO flag holding registers.

Ixx43 Motor xx Overtravel-Limit Flag Address

Range: $000000 - $FFFFFF
Units: Turbo PMAC Addresses
Default: $0

Ixx43, if set to a non-zero value, specifies the address of the overtravel-limit input flags, independently of position-capture flags and amplifier flags, for Motor xx. If Ixx43 is set to 0, Ixx25 specifies the address of the overtravel-limit flags as well as the position-capture flags, and possibly the amplifier flags, for Motor xx. This maintains backward compatibility with older firmware revisions in which Ixx43 was not implemented.

If overtravel limit flags are specified separately using Ixx43, they must use the same type of ICs as Ixx25, as specified by bit 0 of Ixx24.

The following tables show the standard addresses that can be used for Ixx42.

Ixx43 Addresses for PMAC(1)-Style Servo ICs

Servo IC #	Chan. 1	Chan. 2	Chan. 3	Chan. 4	Notes
0	$078000	$078004	$078008	$07800C	1st IC on board PMAC
1	$078100	$078104	$078108	$07810C	2nd IC on board PMAC
2	$078200	$078204	$078208	$07820C	1st IC on 1st ACC-24P/V
3	$078300	$078304	$078308	$07830C	2nd IC on 1st ACC-24P/V
4	$079200	$079204	$079208	$07920C	1st IC on 2nd ACC-24P/V
5	$079300	$079304	$079308	$07930C	2nd IC on 2nd ACC-24P/V
6	$07A200	$07A204	$07A208	$07A20C	1st IC on 3rd ACC-24P/V
7	$07A300	$07A304	$07A308	$07A30C	2nd IC on 3rd ACC-24P/V
8	$07B200	$07B204	$07B208	$07B20C	1st IC on 4th ACC-24P/V
9	$07B300	$07B304	$07B308	$07B30C	2nd IC on 4th ACC-24P/V

Bit 0 of Ixx24 must be set to 0 to use PMAC(1)-style Servo ICs.

Ixx43 Addresses for PMAC2-Style Servo ICs

Servo IC #	Chan. 1	Chan. 2	Chan. 3	Chan. 4	Notes
0	$078000	$078008	$078010	$078018	1st IC on board PMAC2, 3U stack
1	$078100	$078108	$078010	$078018	2nd IC on board PMAC2, 3U stack
2	$078200	$078208	$078210	$078218	1st ACC-24E2x, 1st IC on 1st ACC-24P/V2
3	$078300	$078308	$078310	$078318	2nd ACC-24E2x, 2nd IC on 1st ACC-24P/V2
4	$079200	$079208	$079210	$079218	3rd ACC-24E2x, 1st IC on 2nd ACC-24P/V2
5	$079300	$079308	$079310	$079318	4th ACC-24E2x, 2nd IC on 2nd ACC-24P/V2
6	$07A200	$07A208	$07A210	$07A218	5th ACC-24E2x, 1st IC on 3rd ACC-24P/V2
7	$07A300	$07A308	$07A310	$07A318	6th ACC-24E2x, 2nd IC on 3rd ACC-24P/V2
8	$07B200	$07B208	$07B210	$07B218	7th ACC-24E2x, 1st IC on 4th ACC-24P/V2
9	$07B300	$07B308	$07B310	$07B318	8th ACC-24E2x, 2nd IC on 4th ACC-24P/V2

Bit 0 of Ixx24 must be set to 1 to use PMAC2style Servo ICs.

Ixx43 Addresses for MACRO Flag Holding Registers

IC Node #	MACRO IC 1	MACRO IC 2	MACRO IC 3	MACRO IC 4	Notes
0	$003440	$003450	$003460	$003470	MACRO Flag Register Sets 0, 16, 32, 48
1	$003441	$003451	$003461	$003471	MACRO Flag Register Sets 1, 17, 33, 49
4	$003444	$003454	$003464	$003474	MACRO Flag Register Sets 4, 20, 36, 52
5	$003445	$003455	$003465	$003475	MACRO Flag Register Sets 5, 21, 37, 53
8	$003448	$003458	$003468	$003478	MACRO Flag Register Sets 8, 24, 40, 56
9	$003449	$003459	$003469	$003479	MACRO Flag Register Sets 9, 25, 41, 57
12	$00344C	$00345C	$00346C	$00347C	MACRO Flag Register Sets 12, 28, 44, 60
13	$00344D	$00345D	$00346D	$00347D	MACRO Flag Register Sets 13, 29, 45, 61

Bit 0 of Ixx24 must be set to 1 to use MACRO flag holding registers.

Ixx71 Motor xx Counts per N Commutation Cycles {revised}

Range: 0 – 16,777,215
Units: counts
Default: 1000

For a Turbo PMAC-commutated motor, this parameter defines the size of a commutation cycle in conjunction with Ixx70 (counts/cycle = Ixx71/Ixx70). The meaning of a "count" used in this parameter is defined by the encoder-decode variable I7mn0 for the commutation feedback device. If a "times-4" decode is used, a *count* is one-fourth of an encoder *line*.

When the commutation position feedback is received over the MACRO ring, the units of the feedback are typically 1/32 of a count, so Ixx71 should be in units of 1/32 count in this case.

A commutation cycle, or electrical cycle, consists of two poles (one pole pair) of a multiphase motor.

Note: In firmware revisions V1.938 and older, the maximum value of Ixx71 was 8,388,607.

Examples

1. A four-pole brushless motor with a 1000-line-per-revolution encoder and "times-4" decode has 2 commutation cycles per revolution and 4000 counts per revolution. Therefore, either Ixx70=2 and Ixx71=4000 could be used, or Ixx70=1 and Ixx71=2000.

2. A linear motor has a 60.96-mm (2.4-inch) electrical cycle. An encoder with a 40 micron pitch is wired directly into PMAC and "times-4" decode is used. Ixx70 can be set to 1 and Ixx71 can be calculated as:

$$Ixx71 = 60.96\frac{mm}{cycle} * \frac{line}{0.04mm} * 4\frac{counts}{line} = 6096\frac{counts}{cycle}$$

3. An 8-pole brushless motor has an 8192-line encoder that is wired into a Compact MACRO Station with "times-4" decode. The position data is sent back to PMAC in the MACRO "Type 1" protocol, with units of 1/32 count. If Ixx70 is set to 4 (for 4 electrical cycles per revolution), Ixx71 can be calculated as:

$$Ixx71 = 8192\frac{lines}{rev} * \frac{rev}{4 - cycles} * 4\frac{counts}{line} * 32\frac{(1/32count)}{count} = 262{,}144\frac{(1/32count)}{line}$$

See Also

I-variables Ixx01, Ixx70, Ixx72-Ixx83
Setting Up Turbo PMAC Commutation

Ixx75 Motor xx Phase Position Offset {revised}

Range:	0 – Ixx71 (up to 16,777,215)
Units:	Counts * Ixx70
Default:	0

Ixx75 tells Turbo PMAC the distance between the zero position of an absolute sensor used for power-on phase position (specified by Ixx81 and Ixx91) and the zero position of Turbo PMAC's commutation cycle. It is used to reference the phasing algorithm for a PMAC-commutated motor with an absolute sensor (Ixx81 > 0). If Ixx80 bit 0 is 1 (Ixx80 = 1 or 3), this is done automatically during the power-up/reset cycle. It will also be done in response to a **$** on-line command to the motor, or a **$$** on-line command to the coordinate system containing the motor.

Ixx75 is also used by the **SETPHASE** command (on-line, motion-program, or PLC-program). When the **SETPHASE** command is given, the value of Ixx75 is immediately copied directly into the motor's phase position register. This operation is typically used to correct the phasing, usually at the encoder index pulse, after an initial rough phasing (e.g. from Hall commutation sensors).

The proper value for this parameter can be found with a simple procedure that should be done with an *unloaded* motor, after satisfactory operation has been achieved using a power-on phasing search.

- Define an M-variable to the absolute sensor if you have one.
- Define an M-variable to the internal phase position register. Mxx71 is the suggested M-variable.
- Give the motor an **O0** command.
- Put a bias (a magnitude of 2000 is usually good) on the A phase (higher-numbered DAC of a pair for Turbo PMAC(1)) by setting Ixx29; use a positive bias if Ixx82>0 for digital current loop closure, or if Ixx82=0 and Ixx72>1024 (e.g. 1365 or 1536); use a negative bias if Ixx82=0 and Ixx72<1024 (e.g. 683 or 512).
- Also put a bias in the opposite direction of the same magnitude on the B phase by setting Ixx79. The motor should lock in on a position like a stepper motor.
- Now remove the A-phase bias by setting Ixx29 back to zero, or at least to the value you have found to force zero current in the phase, and the motor should lock in on another position. This position is the zero position of the phasing cycle.
- If you have an absolute sensor, after you are sure the motor has settled, read the position of the absolute sensor by querying its M-variable value. Then:
  - Take the negative of this value, multiply it by Ixx70, and put the resulting value in Ixx75.
  - Now, with Ixx79 returned to zero or the proper bias, and Ixx81 pointing to the absolute sensor, give the motor a **$** command. The motor should be properly phased.
- If you are doing this so you can use the **SETPHASE** command at a known position such as the index, set the internal phase position register to 0 with Mxx71. Then:
  - Return Ixx79 to zero or the proper bias, and close the loop with a **J/** command.

  - Now move to the reference position (e.g. do a homing search move with the index pulse as the trigger) and make sure you are settled there with minimal following error (some integral gain should be used).
  - Read the value of Mxx71 at this point and set Ixx75 to this value.
- Remember to save these variable values before doing a full reset on the card.

Note: In firmware revisions V1.938 and older, the range of Ixx75 was –8,388,608 to +8,388,607. In revisions V1.939 and newer, if a negative value for Ixx75 is specified (-Ixx71 ≤ Ixx75 ≤ 0), Turbo PMAC stores it as (Ixx71 + Ixx75). This maintains the same effect, and therefore keeps backward compatibility, *provided that the Ixx71 commutation cycle size parameter has already been specified correctly* (which would happen automatically in the downloading of a standard configuration backup file). If a value for Ixx75 less than –Ixx71 or greater than Ixx71 is specified, Turbo PMAC will reject this with an error.

See Also

I-variables Ixx01, Ixx70 – Ixx74, Ixx76 – Ixx83
Setting Up Turbo PMAC Commutation

Ixx82 Motor xx Current-Loop Feedback Address {revised description}

Range:	$000000 – $FFFFFF
Units:	Turbo PMAC Y-addresses
Default:	$0

Ixx82 tells Turbo PMAC which addresses to read to get its current feedback values for Motor xx if Turbo PMAC is closing the current loop for this motor. Turbo PMAC must be performing the commutation for the motor (Ixx01=1) if it is to close the current loop as well.

A zero value for Ixx82 tells PMAC not to close the current loop for this motor. In this case, PMAC either outputs one velocity or torque command value (Ixx01 bit 0 = 0), or two phase-current command values (Ixx01 bit 0 = 1), usually represented as analog voltages.

A non-zero value for Ixx82 automatically triggers current loop execution in the phase interrupt, using the current values found in the registers specified by Ixx82. Typically these registers are analog-to-digital converter (ADC) registers in a PMAC2-style Servo IC, or MACRO feedback registers containing copies of ADC registers in a MACRO Station.

Digital current loop closure on the Turbo PMAC requires a set of three consecutive command output registers. Generally this requires writing to either a PMAC2-style Servo IC or a MACRO IC.

When Ixx01 is set to 1, Turbo PMAC performs the phase commutation for this motor, computing two phase current commands based on the position/velocity servo command and the magnetization current value. If Ixx82>0, these commands are compared to the two actual current values read from the address specified by Ixx82, and the next *lower* address. It executes a PI filter on the current loops and outputs three voltage command values to the address specified by

Ixx02 and the next two higher addresses. These are typically the PWM commands for the three half-bridges of a brushless motor power stage.

When the digital current loop is used for drives connected directly to the Turbo PMAC2, the typical values for Ixx82 are:

Turbo PMAC2 Ixx82 Typical Settings

Ixx82	Value	Register	Ixx82	Value	Register
I182	$078006	PMAC2 ADC1B	I1782	$079206	2nd ACC-24x2 ADC1B
I282	$07800E	PMAC2 ADC2B	I1882	$07920E	2nd ACC-24x2 ADC2B
I382	$078016	PMAC2 ADC3B	I1982	$079216	2nd ACC-24x2 ADC3B
I482	$07801E	PMAC2 ADC4B	I2082	$07921E	2nd ACC-24x2 ADC4B
I582	$078106	PMAC2 ADC5B	I2182	$079306	2nd ACC-24x2 ADC5B
I682	$07810E	PMAC2 ADC6B	I2282	$07930E	2nd ACC-24x2 ADC6B
I782	$078116	PMAC2 ADC7B	I2382	$079316	2nd ACC-24x2 ADC7B
I882	$07811E	PMAC2 ADC8B	I2482	$07931E	2nd ACC-24x2 ADC8B
I982	$078206	1st ACC-24x2 ADC1B	I2582	$07A206	3rd ACC-24x2 ADC1B
I1082	$07820E	1st ACC-24x2 ADC2B	I2682	$07A20E	3rd ACC-24x2 ADC2B
I1182	$078216	1st ACC-24x2 ADC3B	I2782	$07A216	3rd ACC-24x2 ADC3B
I1282	$07821E	1st ACC-24x2 ADC4B	I2882	$07A21E	3rd ACC-24x2 ADC4B
I1382	$078306	1st ACC-24x2 ADC5B	I2982	$07A306	3rd ACC-24x2 ADC5B
I1482	$07830E	1st ACC-24x2 ADC6B	I3082	$07A30E	3rd ACC-24x2 ADC6B
I1582	$078316	1st ACC-24x2 ADC7B	I3182	$07A316	3rd ACC-24x2 ADC7B
I1682	$07831E	1st ACC-24x2 ADC8B	I3282	$07A31E	3rd ACC-24x2 ADC8B

When the digital current loop is used for drives connected to the Turbo PMAC2 Ultralite through a MACRO station, the typical values for Ixx82 are:

Turbo PMAC2 Ultralite Ixx82 Typical Settings

Ixx82	Value	Register	Ixx82	Value	Register
I182	$078422	MACRO IC 0 Node 0 Reg. 2	I1782	$07A422	MACRO IC 2 Node 0 Reg. 2
I282	$078426	MACRO IC 0 Node 1 Reg. 2	I1882	$07A426	MACRO IC 2 Node 1 Reg. 2
I382	$07842A	MACRO IC 0 Node 4 Reg. 2	I1982	$07A42A	MACRO IC 2 Node 4 Reg. 2
I482	$07842E	MACRO IC 0 Node 5 Reg. 2	I2082	$07A42E	MACRO IC 2 Node 5 Reg. 2
I582	$078432	MACRO IC 0 Node 8 Reg. 2	I2182	$07A432	MACRO IC 2 Node 8 Reg. 2
I682	$078436	MACRO IC 0 Node 9 Reg. 2	I2282	$07A436	MACRO IC 2 Node 9 Reg. 2
I782	$07843A	MACRO IC 0 Node 12 Reg. 2	I2382	$07A43A	MACRO IC 2 Node 12 Reg. 2
I882	$07843E	MACRO IC 0 Node 13 Reg. 2	I2482	$07A43E	MACRO IC 2 Node 13 Reg. 2
I982	$079422	MACRO IC 1 Node 0 Reg. 2	I2582	$07B422	MACRO IC 3 Node 0 Reg. 2
I1082	$079426	MACRO IC 1 Node 1 Reg. 2	I2682	$07B426	MACRO IC 3 Node 1 Reg. 2
I1182	$07942A	MACRO IC 1 Node 4 Reg. 2	I2782	$07B42A	MACRO IC 3 Node 4 Reg. 2
I1282	$07942E	MACRO IC 1 Node 5 Reg. 2	I2882	$07B42E	MACRO IC 3 Node 5 Reg. 2
I1382	$079432	MACRO IC 1 Node 8 Reg. 2	I2982	$07B432	MACRO IC 3 Node 8 Reg. 2
I1482	$079436	MACRO IC 1 Node 9 Reg. 2	I3082	$07B436	MACRO IC 3 Node 9 Reg. 2
I1582	$07943A	MACRO IC 1 Node 12 Reg. 2	I3182	$07B43A	MACRO IC 3 Node 12 Reg. 2
I1682	$07943E	MACRO IC 1 Node 13 Reg. 2	I3282	$07B43E	MACRO IC 3 Node 13 Reg. 2

UMAC Turbo Ixx82 Typical Settings

Ixx82	Value	Register	Ixx82	Value	Register
I182	$078206	1st ACC-24E2 ADC1B	I1782	$07A206	5th ACC-24E2 ADC1B
I282	$07820E	1st ACC-24E2 ADC2B	I1882	$07A20E	5th ACC-24E2 ADC2B
I382	$078216	1st ACC-24E2 ADC3B	I1982	$07A216	5th ACC-24E2 ADC3B
I482	$07821E	1st ACC-24E2 ADC4B	I2082	$07A21E	5th ACC-24E2 ADC4B
I582	$078306	2nd ACC-24E2 ADC1B	I2182	$07A306	6th ACC-24E2 ADC1B
I682	$07830E	2nd ACC-24E2 ADC2B	I2282	$07A30E	6th ACC-24E2 ADC2B
I782	$078316	2nd ACC-24E2 ADC3B	I2382	$07A316	6th ACC-24E2 ADC3B
I882	$07831E	2nd ACC-24E2 ADC4B	I2482	$07A31E	6th ACC-24E2 ADC4B
I982	$079206	3rd ACC-24E2 ADC1B	I2582	$07B206	7th ACC-24E2 ADC1B
I1082	$07920E	3rd ACC-24E2 ADC2B	I2682	$07B20E	7th ACC-24E2 ADC2B
I1182	$079216	3rd ACC-24E2 ADC3B	I2782	$07B216	7th ACC-24E2 ADC3B
I1282	$07921E	3rd ACC-24E2 ADC4B	I2882	$07B21E	7th ACC-24E2 ADC4B
I1382	$079306	4th ACC-24E2 ADC1B	I2982	$07B306	8th ACC-24E2 ADC1B
I1482	$07930E	4th ACC-24E2 ADC2B	I3082	$07B30E	8th ACC-24E2 ADC2B
I1582	$079316	4th ACC-24E2 ADC3B	I3182	$07B316	8th ACC-24E2 ADC3B
I1682	$07931E	4th ACC-24E2 ADC4B	I3282	$07B31E	8th ACC-24E2 ADC4B

If Ixx82>0, the following variables must be set properly for correct operation of the digital current loop:

- Ixx61: Current-Loop Integral Gain
- Ixx62: Current-Loop Forward-Path Proportional Gain
- Ixx66: PWM Scale Factor
- Ixx72: Commutation Phase Angle
- Ixx76: Current-Loop Back-Path Proportional Gain
- Ixx84: Current-Loop Feedback Mask Word

Ixx83 Motor xx Commutation Position Address {revised description}

Range: $000000 - $FFFFFF
Units: Turbo PMAC addresses
Default values:

Turbo PMAC(1) Ixx83 Defaults

Ixx83	Value	Register	Ixx83	Value	Register
I183	$078001	PMAC Encoder 1	I1783	$079201	2nd ACC-24P/V Encoder 1
I283	$078005	PMAC Encoder 2	I1883	$079205	2nd ACC-24P/V Encoder 2
I383	$078009	PMAC Encoder 3	I1983	$079209	2nd ACC-24P/V Encoder 3
I483	$07800D	PMAC Encoder 4	I2083	$07920D	2nd ACC-24P/V Encoder 4
I583	$078101	PMAC Encoder 5	I2183	$079301	2nd ACC-24P/V Encoder 5
I683	$078105	PMAC Encoder 6	I2283	$079305	2nd ACC-24P/V Encoder 6
I783	$078109	PMAC Encoder 7	I2383	$079309	2nd ACC-24P/V Encoder 7
I883	$07810D	PMAC Encoder 8	I2483	$07930D	2nd ACC-24P/V Encoder 8
I983	$078201	1st ACC-24P/V Encoder 1	I2583	$07A201	3rd ACC-24P/V Encoder 1
I1083	$078205	1st ACC-24P/V Encoder 2	I2683	$07A205	3rd ACC-24P/V Encoder 2
I1183	$078209	1st ACC-24P/V Encoder 3	I2783	$07A209	3rd ACC-24P/V Encoder 3
I1283	$07820D	1st ACC-24P/V Encoder 4	I2883	$07A20D	3rd ACC-24P/V Encoder 4
I1383	$078301	1st ACC-24P/V Encoder 5	I2983	$07A301	3rd ACC-24P/V Encoder 5
I1483	$078305	1st ACC-24P/V Encoder 6	I3083	$07A305	3rd ACC-24P/V Encoder 6
I1583	$078309	1st ACC-24P/V Encoder 7	I3183	$07A309	3rd ACC-24P/V Encoder 7
I1683	$07830D	1st ACC-24P/V Encoder 8	I3283	$07A30D	3rd ACC-24P/V Encoder 8

Turbo PMAC2 (Non-Ultralite) Ixx83 Defaults

Ixx83	Value	Register	Ixx83	Value	Register
I183	$078001	PMAC2 Encoder 1	I1783	$079201	2nd ACC-24P/V2 Encoder 1
I283	$078009	PMAC2 Encoder 2	I1883	$079209	2nd ACC-24P/V2 Encoder 2
I383	$078011	PMAC2 Encoder 3	I1983	$079211	2nd ACC-24P/V2 Encoder 3
I483	$078019	PMAC2 Encoder 4	I2083	$079219	2nd ACC-24P/V2 Encoder 4
I583	$078101	PMAC2 Encoder 5	I2183	$079301	2nd ACC-24P/V2 Encoder 5
I683	$078109	PMAC2 Encoder 6	I2283	$079309	2nd ACC-24P/V2 Encoder 6
I783	$078111	PMAC2 Encoder 7	I2383	$079311	2nd ACC-24P/V2 Encoder 7
I883	$078119	PMAC2 Encoder 8	I2483	$079319	2nd ACC-24P/V2 Encoder 8
I983	$078201	1st ACC-24P/V2 Encoder 1	I2583	$07A201	3rd ACC-24P/V2 Encoder 1
I1083	$078209	1st ACC-24P/V2 Encoder 2	I2683	$07A209	3rd ACC-24P/V2 Encoder 2
I1183	$078211	1st ACC-24P/V2 Encoder 3	I2783	$07A211	3rd ACC-24P/V2 Encoder 3
I1283	$078219	1st ACC-24P/V2 Encoder 4	I2883	$07A219	3rd ACC-24P/V2 Encoder 4
I1383	$078301	1st ACC-24P/V2 Encoder 5	I2983	$07A301	3rd ACC-24P/V2 Encoder 5
I1483	$078309	1st ACC-24P/V2 Encoder 6	I3083	$07A309	3rd ACC-24P/V2 Encoder 6
I1583	$078311	1st ACC-24P/V2 Encoder 7	I3183	$07A311	3rd ACC-24P/V2 Encoder 7
I1683	$078319	1st ACC-24P/V2 Encoder 8	I3283	$07A319	3rd ACC-24P/V2 Encoder 8

Turbo PMAC2 Ultralite Ixx83 Defaults

Ixx83	Value	Register	Ixx83	Value	Register
I183	$078420	MACRO IC 0 Node 0 Reg. 0	I1783	$07A420	MACRO IC 2 Node 0 Reg. 0
I283	$078424	MACRO IC 0 Node 1 Reg. 0	I1883	$07A424	MACRO IC 2 Node 1 Reg. 0
I383	$078428	MACRO IC 0 Node 4 Reg. 0	I1983	$07A428	MACRO IC 2 Node 4 Reg. 0
I483	$07842C	MACRO IC 0 Node 5 Reg. 0	I2083	$07A42C	MACRO IC 2 Node 5 Reg. 0
I583	$078430	MACRO IC 0 Node 8 Reg. 0	I2183	$07A430	MACRO IC 2 Node 8 Reg. 0
I683	$078434	MACRO IC 0 Node 9 Reg. 0	I2283	$07A434	MACRO IC 2 Node 9 Reg. 0
I783	$078438	MACRO IC 0 Node 12 Reg. 0	I2383	$07A438	MACRO IC 2 Node 12 Reg. 0
I883	$07843C	MACRO IC 0 Node 13 Reg. 0	I2483	$07A43C	MACRO IC 2 Node 13 Reg. 0
I983	$079420	MACRO IC 1 Node 0 Reg. 0	I2583	$07B420	MACRO IC 3 Node 0 Reg. 0
I1083	$079424	MACRO IC 1 Node 1 Reg. 0	I2683	$07B424	MACRO IC 3 Node 1 Reg. 0
I1183	$079428	MACRO IC 1 Node 4 Reg. 0	I2783	$07B428	MACRO IC 3 Node 4 Reg. 0
I1283	$07942C	MACRO IC 1 Node 5 Reg. 0	I2883	$07B42C	MACRO IC 3 Node 5 Reg. 0
I1383	$079430	MACRO IC 1 Node 8 Reg. 0	I2983	$07B430	MACRO IC 3 Node 8 Reg. 0
I1483	$079434	MACRO IC 1 Node 9 Reg. 0	I3083	$07B434	MACRO IC 3 Node 9 Reg. 0
I1583	$079438	MACRO IC 1 Node 12 Reg. 0	I3183	$07B438	MACRO IC 3 Node 12 Reg. 0
I1683	$07943C	MACRO IC 1 Node 13 Reg. 0	I3283	$07B43C	MACRO IC 3 Node 13 Reg. 0

UMAC Turbo Ixx83 Defaults

Ixx82	Value	Register	Ixx82	Value	Register
I182	$078201	1st ACC-24E2 Encoder 1	I1782	$07A201	5th ACC-24E2 Encoder 1
I282	$078209	1st ACC-24E2 Encoder 2	I1882	$07A209	5th ACC-24E2 Encoder 2
I382	$078211	1st ACC-24E2 Encoder 3	I1982	$07A211	5th ACC-24E2 Encoder 3
I482	$078219	1st ACC-24E2 Encoder 4	I2082	$07A219	5th ACC-24E2 Encoder 4
I582	$078301	2nd ACC-24E2 Encoder 1	I2182	$07A301	6th ACC-24E2 Encoder 1
I682	$078309	2nd ACC-24E2 Encoder 2	I2282	$07A309	6th ACC-24E2 Encoder 2
I782	$078311	2nd ACC-24E2 Encoder 3	I2382	$07A311	6th ACC-24E2 Encoder 3
I882	$078319	2nd ACC-24E2 Encoder 4	I2482	$07A319	6th ACC-24E2 Encoder 4
I982	$079201	3rd ACC-24E2 Encoder 1	I2582	$07B201	7th ACC-24E2 Encoder 1
I1082	$079209	3rd ACC-24E2 Encoder 2	I2682	$07B209	7th ACC-24E2 Encoder 2
I1182	$079211	3rd ACC-24E2 Encoder 3	I2782	$07B211	7th ACC-24E2 Encoder 3
I1282	$079219	3rd ACC-24E2 Encoder 4	I2882	$07B219	7th ACC-24E2 Encoder 4
I1382	$079301	4th ACC-24E2 Encoder 1	I2982	$07B301	8th ACC-24E2 Encoder 1
I1482	$079309	4th ACC-24E2 Encoder 2	I3082	$07B309	8th ACC-24E2 Encoder 2
I1582	$079311	4th ACC-24E2 Encoder 3	I3182	$07B311	8th ACC-24E2 Encoder 3
I1682	$079319	4th ACC-24E2 Encoder 4	I3282	$07B319	8th ACC-24E2 Encoder 4

For a motor commutated by Turbo PMAC (Ixx01 = 1 or 3), Ixx83 tells Turbo PMAC where to read its commutation (phasing) position information for Motor xx every commutation cycle. This can be a different address from that used for power-on/reset phasing position, which is determined by Ixx81. If Turbo PMAC is not commutating Motor xx (Ixx01 = 0 or 2), Ixx83 is not used.

Ixx83 contains the address of the register to be read. If Ixx01 bit 1 is set to 0 (Ixx01 = 1), the register is the X-register at that address. If Ixx01 bit 1 is set to 1 (Ixx01 = 3), the register is the Y-register at that address.

For Turbo PMAC boards with on-board encoder circuitry, Ixx83 typically contains the address of the "phase position" encoder register for encoder *x*; this is the default. Since these registers have 'X' addresses, Ixx01 is set to 1.

For Turbo PMAC2 Ultralite boards, Ixx83 typically contains the address of a MACRO node's position feedback register; this is the default. Since PMAC2 can only commutate over MACRO using nodes with 'Y' addresses, Ixx01 is set to 3 in these cases.

I7m06 Servo IC m ADC Strobe Word {revised description}

Range: $000000 - $FFFFFF
Units: Serial Data Stream (MSB first, starting on rising edge of phase clock)
Default: $FFFFFE

I7m06 controls the ADC strobe signal for all machine interface channels on Servo IC m. The 24-bit word set by I7m06 is shifted out serially on the ADC_STROB lines, MSB first, one bit per ADC_CLK cycle starting on the rising edge of the phase clock.

In revisions "D" and newer of the DSPGATE1 Servo IC (beginning shipments in 2002), bit 0 (the LSB) of I7m06 is a control bit that determines whether the Servo IC will expect "header" information on the return data streams that precedes the numerical data from the ADCs. If bit 0 is 0, no header information is expected, and the low output from this bit is held until the next rising edge of the phase clock. This setting must be used on all earlier revisions of the DSPGATE1 Servo IC.

In revisions "D" and newer, if bit 0 of I7m06 is 1, up to 4 bits of header information can be accepted on the returned serial data streams from the ADCs (as with the ADCs in Delta Tau's Geo power block amplifiers). These bits are "rolled over" and end up in bits 0 – 3 of the ADC register in the Servo IC, and the numerical data ends up with its MSB in bit 23 of the ADC register. If fewer than 4 header bits are expected, the beginning of the strobe word should be delayed by setting the first bit(s) of I7m06 to 0. Specifically, if $(4 - n)$ header bits are expected, the first n bits of I7m06 should be set to 0. In this setting, the ADC_STROB output is taken low and held low after bit 0 is shifted out.

The first bit that is a "1" creates a rising edge on the ADC_STROB output that is typically used as a "start-convert" signal. Some A/D converters just need this rising edge for the conversion; others need the signal to stay high all of the way through the conversion. Intermediate bits of the ADC_STROB output can be used to transmit other information in some applications.

The default I7m06 value of $FFFFFE is suitable for use with Delta Tau "Quad Amps", most third-party direct-PWM amplifiers, and with ACC-28B A/D converters. A value of $3FFFFF is appropriate for Delta Tau "Geo" power-block amplifiers.

I7mn9 Servo IC m Channel n Hardware-1/T Control

Range: 0 – 1
Units: none
Default: 0

I7mn9 controls whether the "hardware-1/T" functionality is enabled for Channel n of a PMAC2-style Servo IC m. If I7mn9 is set to the default value of 0, the hardware-1/T functionality is disabled, permitting the use of the "software-1/T" position extension that is calculated by default

with encoder conversion method $0. If I7mn9 is set to 1, the hardware-1/T functionality is enabled (if present on the IC), and the software-1/T cannot be used.

The hardware-1/T functionality is present only on Revision D and newer of the PMAC2-style DSPGATE1 IC, released at the beginning of the year 2002. Setting I7mn9 to 1 on an older revision IC does nothing – software-1/T functions can still be used. However, it is strongly recommended that I7mn9 be left at 0 in this case, to prevent possible problems when copying a configuration to newer hardware.

When the hardware-1/T functionality is enabled, the IC computes a new fractional-count position estimate based on timers every SCLK (encoder sample clock) cycle. This permits the fractional count data to be used for hardware capture and compare functions, enhancing their resolution. The sub-count position-capture data can be used automatically in Turbo PMAC triggered-move functions if bit 12 of Ixx24 is set to 1. This is particularly useful when the IC is used on an ACC-51 high-resolution analog-encoder interpolator board. However, it replaces the timer registers at the first two "Y" addresses for the channel with fractional count position data, so the traditional software-1/T method of the conversion table cannot work if this is enabled.

If you enable the hardware-1/T functionality, and want to be able to use 1/T interpolation in your servo loop, you must use the hardware-1/T extension method ($C method digit with the mode switch bit set to 1) in the encoder conversion table.

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

Range: $000000 - $FFFFFF
Units: Modified Turbo PMAC Addresses
Defaults:

Turbo PMAC(1) Defaults

I-Var.	Setting	Meaning	I-Var.	Setting	Meaning
I8000	$078000	1/T Extension of Encoder 1	I8004	$078100	1/T Extension of Encoder 5
I8001	$078004	1/T Extension of Encoder 2	I8005	$078104	1/T Extension of Encoder 6
I8002	$078008	1/T Extension of Encoder 3	I8006	$078108	1/T Extension of Encoder 7
I8003	$07800C	1/T Extension of Encoder 4	I8007	$07810C	1/T Extension of Encoder 8

I8008 - I8191 = 0

Turbo PMAC2 Defaults

I-Var.	Setting	Meaning	I-Var.	Setting	Meaning
I8000	$078000	1/T Extension of Encoder 1	I8004	$078100	1/T Extension of Encoder 5
I8001	$078008	1/T Extension of Encoder 2	I8005	$078108	1/T Extension of Encoder 6
I8002	$078010	1/T Extension of Encoder 3	I8006	$078110	1/T Extension of Encoder 7
I8003	$078018	1/T Extension of Encoder 4	I8007	$078118	1/T Extension of Encoder 8

I8008 - I8191 = 0

Turbo PMAC2 Ultralite Defaults

I-Var.	Setting	Meaning	I-Var.	Setting	Meaning
I8000	$2F8420	MACRO Node 0 Reg. 0 Read	I8008	$2F8430	MACRO Node 8 Reg. 0 Read
I8001	$018000	24 bits, bit 0 LSB	I8009	$018000	24 bits, bit 0 LSB
I8002	$2F8424	MACRO Node 1 Reg. 0 Read	I8010	$2F8434	MACRO Node 9 Reg. 0 Read
I8003	$018000	24 bits, bit 0 LSB	I8011	$018000	24 bits, bit 0 LSB
I8004	$2F8428	MACRO Node 4 Reg. 0 Read	I8012	$2F8438	MACRO Node 12 Reg. 0 Read
I8005	$018000	24 bits, bit 0 LSB	I8013	$018000	24 bits, bit 0 LSB
I8006	$2F842C	MACRO Node 5 Reg. 0 Read	I8014	$2F843C	MACRO Node 13 Reg. 0 Read
I8007	$018000	24 bits, bit 0 LSB	I8015	$018000	24 bits, bit 0 LSB

I8016 - I8191 = 0

I8000 to I8191 form the 192 setup lines of the Turbo PMAC's Encoder Conversion Table (ECT). The main purpose of the ECT is to provide a pre-processing of feedback and master data to prepare it for use by the servo loop. It can also be used to execute certain simple calculations at the servo update frequency.

Each I-variable occupies a fixed register in the Turbo PMAC's memory map. The register addresses are important, because the results of the ECT are accessed by address.

The ECT has two halves: setup and results. The "setup" half resides in Turbo PMAC's Y-memory, and can be accessed through these 192 I-variables. The "result" half resides in Turbo PMAC's X-memory. Each of the 192 I-variables has a matching result X-register at the same numerical address. If the entry consists of more than one line, the last line has the final result; any previous lines contain intermediate results.

The entries in the ECT are almost always set up through the table's configuration menu in the PMAC Executive program.

The following table shows the address of each ECT I-variable.

I-Variable	Address	I-Variable	Address	I-Variable	Address	I-Variable	Address
I8000	$003501	I8048	$003531	I8096	$003561	I8144	$003591
I8001	$003502	I8049	$003532	I8097	$003562	I8145	$003592
I8002	$003503	I8050	$003533	I8098	$003563	I8146	$003593
I8003	$003504	I8051	$003534	I8099	$003564	I8147	$003594
I8004	$003505	I8052	$003535	I8100	$003565	I8148	$003595
I8005	$003506	I8053	$003536	I8101	$003566	I8149	$003596
I8006	$003507	I8054	$003537	I8102	$003567	I8150	$003597
I8007	$003508	I8055	$003538	I8103	$003568	I8151	$003598
I8008	$003509	I8056	$003539	I8104	$003569	I8152	$003599
I8009	$00350A	I8057	$00353A	I8105	$00356A	I8153	$00359A
I8010	$00350B	I8058	$00353B	I8106	$00356B	I8154	$00359B
I8011	$00350C	I8059	$00353C	I8107	$00356C	I8155	$00359C
I8012	$00350D	I8060	$00353D	I8108	$00356D	I8156	$00359D
I8013	$00350E	I8061	$00353E	I8109	$00356E	I8157	$00359E
I8014	$00350F	I8062	$00353F	I8110	$00356F	I8158	$00359F
I8015	$003510	I8063	$003540	I8111	$003570	I8159	$0035A0
I8016	$003511	I8064	$003541	I8112	$003571	I8160	$0035A1
I8017	$003512	I8065	$003542	I8113	$003572	I8161	$0035A2
I8018	$003513	I8066	$003543	I8114	$003573	I8162	$0035A3
I8019	$003514	I8067	$003544	I8115	$003574	I8163	$0035A4
I8020	$003515	I8068	$003545	I8116	$003575	I8164	$0035A5
I8021	$003516	I8069	$003546	I8117	$003576	I8165	$0035A6
I8022	$003517	I8070	$003547	I8118	$003577	I8166	$0035A7
I8023	$003518	I8071	$003548	I8119	$003578	I8167	$0035A8
I8024	$003519	I8072	$003549	I8120	$003579	I8168	$0035A9
I8025	$00351A	I8073	$00354A	I8121	$00357A	I8169	$0035AA
I8026	$00351B	I8074	$00354B	I8122	$00357B	I8170	$0035AB
I8027	$00351C	I8075	$00354C	I8123	$00357C	I8171	$0035AC
I8028	$00351D	I8076	$00354D	I8124	$00357D	I8172	$0035AD
I8029	$00351E	I8077	$00354E	I8125	$00357E	I8173	$0035AE
I8030	$00351F	I8078	$00354F	I8126	$00357F	I8174	$0035AF
I8031	$003520	I8079	$003550	I8127	$003580	I8175	$0035B0
I8032	$003521	I8080	$003551	I8128	$003581	I8176	$0035B1
I8033	$003522	I8081	$003552	I8129	$003582	I8177	$0035B2
I8034	$003523	I8082	$003553	I8130	$003583	I8178	$0035B3
I8035	$003524	I8083	$003554	I8131	$003584	I8179	$0035B4
I8036	$003525	I8084	$003555	I8132	$003585	I8180	$0035B5
I8037	$003526	I8085	$003556	I8133	$003586	I8181	$0035B6
I8038	$003527	I8086	$003557	I8134	$003587	I8182	$0035B7
I8039	$003528	I8087	$003558	I8135	$003588	I8183	$0035B8
I8040	$003529	I8088	$003559	I8136	$003589	I8184	$0035B9
I8041	$00352A	I8089	$00355A	I8137	$00358A	I8185	$0035BA
I8042	$00352B	I8090	$00355B	I8138	$00358B	I8186	$0035BB
I8043	$00352C	I8091	$00355C	I8139	$00358C	I8187	$0035BC
I8044	$00352D	I8092	$00355D	I8140	$00358D	I8188	$0035BD
I8045	$00352E	I8093	$00355E	I8141	$00358E	I8189	$0035BE
I8046	$00352F	I8094	$00355F	I8142	$00358F	I8190	$0035BF
I8047	$003530	I8095	$003560	I8143	$003590	I8191	$0035C0

Table Structure: The ECT consists of a series of "entries", with each entry creating one processed ("converted") feedback value. An entry in the ECT can have 1, 2, or 3 lines, with each line containing a 24-bit setup word (I-variable) in Y-memory, and a 24-bit result register in X-memory. Therefore, each entry contains 1, 2, or 3 of these 24-bit I-variables. The final result is always in the X-memory register matching the *last* I-variable in the entry.

The variables that commonly contain the address of the last line of the entry are Ixx03 Motor xx Position-Loop Feedback Address, Ixx04 Motor xx Velocity-Loop Feedback Address, Ixx05 Motor xx Master Position Address and Isx93 Coordinate System 'x' Time-Base Address.

The addresses for these variables can be specified directly using the above table (e.g. **I103=$3501**) or by reference to the table I-variable with the special on-line command **I{constant}=@I{constant}**, which sets the first I-variable to the address of the second (e.g. **I103=@I8000**).

Entry First Line: The first line's setup register (I-variable) in each entry consists of a source address in the low 19 bits (bits 0 – 18), which contains the Turbo PMAC address of the raw data to be processed, a possible mode switch in bit 19, and a "method" value in the high 4 bits (first hex digit), which specifies how this data is to be processed. If the first line (I-variable) in the entry is $000000, this signifies the end of the active table, regardless of what subsequent entries in the table (higher numbered I-variables) contain.

Entry Additional Lines: Depending on the method, 1 or 2 additional lines (I-variables) may be required in the entry to provide further instructions on processing.

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 extension	None	-	-
$D	3	Exponential filter of parallel data	None	Max Change per Cycle	Filter Gain (Inverse Time Constant)
$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	0 = PMAC(1) IC 1 = PMAC2 IC	$0 Method digit & Address of 1st A/D converter	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

Incremental Encoder Entries ($0, $8, $C): These three conversion table methods utilize the incremental encoder registers in the Servo ICs. Each method provides a processed result with the units of (1/32) count – the low 5 bits of the result are fractional data.

Software 1/T Extension: With the $0 method, the fractional data is computed by dividing the "Time Since Last Count" register by the "Time Between Last 2 Counts" register. This technique is known as "1/T extension", and is the default and most commonly used method. It can be used with a digital incremental encoder connected directly to the Turbo PMAC, through either PMAC(1)-style or PMAC2-style Servo ICs.

Note: 1/T extension with 8 bits of fractional resolution (units of 1/256 count) can be gotten using the intermediate result value of the "triggered time-base" conversion in "running mode". This intermediate result is in the first line of the entry. If used for position data, one true count of the position is considered by Turbo PMAC software to be 8 counts.

Parallel Extension: With the $8 method, the fractional data is computed by reading the 5 inputs at bits 19-23 either of the specified address (USERn, Wn, Vn, Un, and Tn flag inputs, respectively) if the mode switch bit of the setup I-variable is set to 1 for PMAC2-style Servo ICs, or of the specified address plus 4 (CHC[n+1], HMFL[n+1], +LIM[n+1], -LIM[n+1], FAULT[n+1]) if the mode switch bit of the setup I-variable is set to 0 for PMAC(1)-style Servo ICs. This technique is known as "parallel extension", and can be used with an analog incremental encoder processed through an ACC-8D Opt 8 Analog Encoder Interpolator board or its equivalent.

No Extension: In the $C method with the mode switch bit set to 0, the fractional data is always set to zero, which means there is no extension of the incremental encoder count. This setting is used mainly to verify the effect of one of the two extension methods. It is also recommended when feeding back the pulse-and-direction outputs for stepper drives.

Hardware 1/T Extension: In the $C method with the mode switch bit set to 1, the fractional data is read from a special timer-based register in the Servo IC that has already computed the fractional-count data in hardware. This feature is only supported in the D-revision or newer (first shipments around the beginning of 2002) of the PMAC2-style "DSPGATE1" Servo ICs. The alternate timer registers for the encoder channel must be selected by setting I7mn9 for the channel to 1.

Using this mode permits timer-based sub-count capture and compare features to be used on this encoder channel.

With any of these three conversion methods, the source address in the low 19 bits (bits 0 - 18) is that of the starting register of the machine interface channel.

The first table below shows the entries for PMAC(1)-style encoder channels. The 'm' in the first hex digit (bits 20 - 23) represents the conversion method ($0, $8, or $C). For the PMAC(1)-style channels, the bit 19 mode switch is always 0, so the second hex digit is always '7' for the hardware registers.

Entries for PMAC(1)-Style Servo ICs

Servo IC #	Chan. 1	Chan. 2	Chan. 3	Chan. 4	Notes
0	$m78000	$m78004	$m78008	$m7800C	1st IC on board PMAC
1	$m78100	$m78104	$m78108	$m7810C	2nd IC on board PMAC
2	$m78200	$m78204	$m78208	$m7820C	1st IC on 1st ACC-24P/V
3	$m78300	$m78304	$m78308	$m7830C	2nd IC on 1st ACC-24P/V
4	$m79200	$m79204	$m79208	$m7920C	1st IC on 2nd ACC-24P/V
5	$m79300	$m79304	$m79308	$m7930C	2nd IC on 2nd ACC-24P/V
6	$m7A200	$m7A204	$m7A208	$m7A20C	1st IC on 3rd ACC-24P/V
7	$m7A300	$m7A304	$m7A308	$m7A30C	2nd IC on 3rd ACC-24P/V
8	$m7B200	$m7B204	$m7B208	$m7B20C	1st IC on 4th ACC-24P/V
9	$m7B300	$m7B304	$m7B308	$m7B30C	2nd IC on 4th ACC-24P/V

The next table shows the entry values for PMAC2-style encoder channels. The 'm' in the first hex digit (bits 20 – 23) represents the conversion method ($0, $8, or $C). The 'n' in the second hex digit (bits 16 – 19) contains the bit 19 mode switch and the start of the source address. For methods $0 (software 1/T extension) and $C (no extension), the bit 19 mode switch is 0, making the second hex digit '7'. For method $8 (parallel extension) or for method $C for hardware 1/T extension, the bit 19 mode switch is 1, changing the second hex digit from '7' to 'F'.

Entries for PMAC2-Style Servo ICs

Servo IC #	Chan. 1	Chan. 2	Chan. 3	Chan. 4	Notes
0	$mn8000	$mn8008	$mn8010	$mn8018	1st IC on board PMAC2, 3U stack
1	$mn8100	$mn8108	$mn8010	$mn8018	2nd IC on board PMAC2, 3U stack
2	$mn8200	$mn8208	$mn8210	$mn8218	1st ACC-24E2x, 1st IC on 1st ACC-24P/V2
3	$mn8300	$mn8308	$mn8310	$mn8318	2nd ACC-24E2x, 2nd IC on 1st ACC-24P/V2
4	$mn9200	$mn9208	$mn9210	$mn9218	3rd ACC-24E2x, 1st IC on 2nd ACC-24P/V2
5	$mn9300	$mn9308	$mn9310	$mn9318	4th ACC-24E2x, 2nd IC on 2nd ACC-24P/V2
6	$mnA200	$mnA208	$mnA210	$mnA218	5th ACC-24E2x, 1st IC on 3rd ACC-24P/V2
7	$mnA300	$mnA308	$mnA310	$mnA318	6th ACC-24E2x, 2nd IC on 3rd ACC-24P/V2
8	$mnB200	$mnB208	$mnB210	$mnB218	7th ACC-24E2x, 1st IC on 4th ACC-24P/V2
9	$mnB300	$mnB308	$mnB310	$mnB318	8th ACC-24E2x, 2nd IC on 4th ACC-24P/V2

Entries for PMAC2 MACRO IC 0

Handwheel Channel #	PMAC2
Channel 1	$mn8410
Channel 2	$mn8418

These are single-line entries in the table, so the next line (I-Variable) is the start of the next entry.

ACC-28 Style A/D Entries ($1, $5): The "A/D" feedback entries read from the high 16 bits of the specified address and shift the data right three bits so that the least significant bit of the processed result in bit 5. Unlike the "parallel feedback" methods, this method will not "roll over" and extend the result.

The $1 method processes the information directly, essentially a copying with shift. The $5 integrates the input value as it copies and shifts it. That is, it reads the input value, shifts it right three bits, adds the bias term in the second line, and adds this value to the previous processed result.

If the bit 19 mode switch of the entry is '0', the 16-bit source value is treated as a signed quantity; this should be used for the ACC-28A. If bit 19 of the entry is '1', the 16-bit value is treated as an unsigned quantity; this should be used for the ACC-28B or the ACC-28E.

The first two tables show the entry values that should be used for ACC-28 boards interfaced to PMAC(1)-style Servo ICs. The 'm' in the first hex digit refers to the method digit -- $1 for un-integrated; $5 for integrated. Note that setting the bit 19 mode switch bit to 1 for the ACC-28B changes the second hex digit from '7' to 'F'.

Entries for PMAC(1)-Style Servo ICs using ACC-28A

Servo IC #	Chan. 1	Chan. 2	Chan. 3	Chan. 4	Notes
0	$m78006	$m78007	$m7800E	$m7800F	1st IC on board PMAC
1	$m78106	$m78107	$m7810E	$m7810F	2nd IC on board PMAC
2	$m78206	$m78207	$m7820E	$m7820F	1st IC on 1st ACC-24P/V
3	$m78306	$m78307	$m7830E	$m7830F	2nd IC on 1st ACC-24P/V
4	$m79206	$m79207	$m7920E	$m7920F	1st IC on 2nd ACC-24P/V
5	$m79306	$m79307	$m7930E	$m7930F	2nd IC on 2nd ACC-24P/V
6	$m7A206	$m7A207	$m7A20E	$m7A20F	1st IC on 3rd ACC-24P/V
7	$m7A306	$m7A307	$m7A30E	$m7A30F	2nd IC on 3rd ACC-24P/V
8	$m7B206	$m7B207	$m7B20E	$m7B20F	1st IC on 4th ACC-24P/V
9	$m7B306	$m7B307	$m7B30E	$m7B30F	2nd IC on 4th ACC-24P/V

Entries for PMAC(1)-Style Servo ICs using ACC-28B

Servo IC #	Chan. 1	Chan. 2	Chan. 3	Chan. 4	Notes
0	$mF8006	$mF8007	$mF800E	$mF800F	1st IC on board PMAC
1	$mF8106	$mF8107	$mF810E	$mF810F	2nd IC on board PMAC
2	$mF8206	$mF8207	$mF820E	$mF820F	1st IC on 1st ACC-24P/V
3	$mF8306	$mF8307	$mF830E	$mF830F	2nd IC on 1st ACC-24P/V
4	$mF9206	$mF9207	$mF920E	$mF920F	1st IC on 2nd ACC-24P/V
5	$mF9306	$mF9307	$mF930E	$mF930F	2nd IC on 2nd ACC-24P/V
6	$mFA206	$mFA207	$mFA20E	$mFA20F	1st IC on 3rd ACC-24P/V
7	$mFA306	$mFA307	$mFA30E	$mFA30F	2nd IC on 3rd ACC-24P/V
8	$mFB206	$mFB207	$mFB20E	$mFB20F	1st IC on 4th ACC-24P/V
9	$mFB306	$mFB307	$mFB30E	$mFB30F	2nd IC on 4th ACC-24P/V

The next table shows the entry values that should be used for ACC-28B boards interfaced to PMAC2-style Servo ICs (ACC-28A is not compatible with these ICs). The 'm' in the first hex digit refers to the method digit -- $1 for un-integrated; $5 for integrated. Note that setting the bit 19 mode switch bit to 1 for the ACC-28B changes the second hex digit from '7' to 'F'.

Entries for PMAC2-Style ADC Registers Using ACC-28B

Register	PMAC2	1st ACC-24P/V2	2nd ACC-24P/V2	3rd ACC-24P/V2	4th ACC-24P/V2
ADC 1A	$mF8005	$mF8205	$mF9205	$mFA205	$mFB205
ADC 1B	$mF8006	$mF8206	$mF9206	$mFA206	$mFB206
ADC 2A	$mF800D	$mF820D	$mF920D	$mFA20D	$mFB20D
ADC 2B	$mF800E	$mF820E	$mF920E	$mFA20E	$mFB20E
ADC 3A	$mF8015	$mF8215	$mF9215	$mFA215	$mFB215
ADC 3B	$mF8016	$mF8216	$mF9216	$mFA216	$mFB216
ADC 4A	$mF801D	$mF821D	$mF921D	$mFA21D	$mFB21D
ADC 4B	$mF801E	$mF821E	$mF921E	$mFA21E	$mFB21E
ADC 5A	$mF8105	$mF8305	$mF9305	$mFA305	$mFB305
ADC 5B	$mF8106	$mF8306	$mF9306	$mFA306	$mFB306
ADC 6A	$mF810D	$mF830D	$mF930D	$mFA30D	$mFB30D
ADC 6B	$mF810E	$mF830E	$mF930E	$mFA30E	$mFB30E
ADC 7A	$mF8115	$mF8315	$mF9315	$mFA315	$mFB315
ADC 7B	$mF8116	$mF8316	$mF9316	$mFA316	$mFB316
ADC 8A	$mF811D	$mF831D	$mF931D	$mFA31D	$mFB31D
ADC 8B	$mF811E	$mF831E	$mF931E	$mFA31E	$mFB31E

The next table shows the entry values that should be used for ACC-28E boards in a UMAC Turbo system. The 'm' in the first hex digit refers to the method digit -- $1 for un-integrated; $5 for integrated. Note that setting the bit 19 mode switch bit to 1 for the ACC-28E changes the second hex digit from '7' to 'F'.

Entries for UMAC ACC-28E ADCs

I/O IC #	SW1-1	SW1-2	SW1-3	SW1-4	Chan. 1	Chan. 2	Chan. 3	Chan. 4
0	ON	ON	ON	ON	$mF8C00	$mF8C01	$mF8C02	$mF8C03
1	OFF	ON	ON	ON	$mF8D00	$mF8D01	$mF8D02	$mF8D03
2	ON	OFF	ON	ON	$mF8E00	$mF8E01	$mF8E02	$mF8E03
3	OFF	OFF	ON	ON	$mF8F00	$mF8F01	$mF8F02	$mF8F03
4	ON	ON	OFF	ON	$mF9C00	$mF9C01	$mF9C02	$mF9C03
5	OFF	ON	OFF	ON	$mF9D00	$mF9D01	$mF9D02	$mF9D03
6	ON	OFF	OFF	ON	$mF9E00	$mF9E01	$mF9E02	$mF9E03
7	OFF	OFF	OFF	ON	$mF9F00	$mF9F01	$mF9F02	$mF9F03
8	ON	ON	ON	OFF	$mFAC00	$mFAC01	$mFAC02	$mFAC03
3	OFF	ON	ON	OFF	$mFAD00	$mFAD01	$mFAD02	$mFAD03
4	ON	OFF	ON	OFF	$mFAE00	$mFAE01	$mFAE02	$mFAE03
5	OFF	OFF	ON	OFF	$mFAF00	$mFAF01	$mFAF02	$mFAF03
6	ON	ON	OFF	OFF	$mFBC00	$mFBC01	$mFBC02	$mFBC03
7	OFF	ON	OFF	OFF	$mFBD00	$mFBD01	$mFBD02	$mFBD03
8	ON	OFF	OFF	OFF	$mFBE00	$mFBE01	$mFBE02	$mFBE03
9	OFF	OFF	OFF	OFF	$mFBE00	$mFBE01	$mFBE00	$mFBE03

Integration Bias: The $5 integrated format requires a second line to specify the bias of the A/D converter. This bias term is a signed quantity (even for an unsigned A/D converter), with units of 1/256 of the LSB of the 16-bit A/D converter. This value is *subtracted* from the reading of the ADC before the integration occurs.

For example, if there were an offset in a 16-bit ADC of +5 LSBs, this term would be set to 1280. If no bias is desired, a zero value should be entered here. If the conversion is unsigned, the result after the bias is not permitted to be less than zero. This term permits reasonable integration, even with an analog offset.

Parallel Feedback Entries ($2, $3, $6, $7): The "parallel feedback" entries read a word from the address specified in the low 19 bits (bits 0 to 18) of the first line. The four methods in this class are:

- $2: Y-word parallel, no filtering (2-line entry)
- $3: Y-word parallel, with filtering (3-line entry)
- $6: Y/X-word parallel, no filtering (2-line entry)
- $7: Y/X-word parallel, with filtering (3-line entry)

The Bit-19 mode switch in the first line controls whether the least significant bit (LSB) of the source register is placed in Bit 5 of the result register ("normal shift"), providing the standard 5 bits of (non-existent) fraction, or the LSB is placed in Bit 0 of the result register ("unshifted"), creating no fractional bits.

Normally, the Bit-19 mode switch is set to 0 to place the source LSB in Bit 5 of the result register. Bit 19 is set to 1 to place to source LSB in Bit 0 of the result register for one of three reasons:

- The data already comes with 5 bits of fraction, as from a Compact MACRO Station.
- The normal shift limits the maximum velocity too much ($V_{max}<2^{18}$ LSBs per servo cycle)
- The normal shift limits the position range too much (Range$<\pm 2^{47}$/Ix08/32 LSBs)

Unless this is done because the data already contains fractional information, the "unshifted" conversion will mean that the motor position loop will consider 1 LSB of the source to be 1/32 of a count, instead of 1 count.

Width/Offset Word: The second setup line (I-variable) of a parallel read entry contains the width of the data to be read, and the location of the LSB. This 24-bit value, usually represented as 6 hexadecimal digits, is split evenly into two halves, each of 3 hex digits. The first half represents the width of the parallel data in bits, and can range from $001 (1 bit wide – not of much practical use) to $018 (24 bits wide).

The second half of the line contains the bit location of the LSB of the data in the source word, and can range from $000 (Bit 0 of the Y-word at the source address is the LSB), through $017 (Bit 23 of the Y-word at the source address), and $018 (Bit 24, which is Bit 0 of the next word, is the LSB) to $02F (Bit 47, which is Bit 23 of the next word, is the LSB).

If the LSB bit location exceeds 23, or the sum of the LSB bit location and the bit width exceeds 24, the source data extends into the "next word". If the method character is $2 or $3, the next word is the Y-word at the source address + 1. If the method character is $6 or $7, the next word is the X-word at the source address.

For example, to use 20 bits starting at bit 0 (bits 0 – 19) of the Y-word of the source address, this word would be set to $014000. To use all 24 bits of the X-word of the source address, this word

would be set to $018018. To use 24 bits starting at bit 12 of the specified address (with the highest 12 bits coming from the X-word or the next higher Y-address, this word would be set to $01800C.

Maximum Change Word: If the method character for a parallel read is $3 or $7, specifying "filtered" parallel read, there is a third setup line (I-variable) for the entry. This third line contains the maximum change in the source data in a single cycle that will be reflected in the processed result, expressed in LSBs per servo cycle. The filtering that this creates provides an important protection against noise and misreading of data. This number is effectively a velocity value, and should be set slightly greater than the maximum true velocity ever expected.

ACC-14: The Accessory 14 family of boards is often used to bring parallel data feedback to the Turbo PMAC, such as that from parallel absolute encoders, and from interferometers. The following table shows the first line of the entries for ACC-14D/V boards connected to a Turbo PMAC controller over a JEXP expansion port cable:

Entries for ACC-14D/V Registers

Register	First Line Value	Register	First Line Value
1st ACC-14D/V Port A	$m78A00	4th ACC-14D/V Port A	$m78D00
1st ACC-14D/V Port B	$m78A01	4th ACC-14D/V Port B	$m78D01
2nd ACC-14D/V Port A	$m78B00	5th ACC-14D/V Port A	$m78E00
2nd ACC-14D/V Port B	$m78B01	5th ACC-14D/V Port B	$m78E01
3rd ACC-14D/V Port A	$m78C00	6th ACC-14D/V Port A	$m78F00
3rd ACC-14D/V Port B	$m78C01	6th ACC-14D/V Port B	$m78F01

MACRO Position Feedback: When position feedback is received through the MACRO ring, the MACRO input registers are treated as parallel-data feedback. The following table shows the first line of the entries for MACRO position feedback registers.

Entries for Type 1 MACRO Position Feedback Registers

Register	First Line Value	Register	First Line Value
MACRO IC 0 Node 0 Reg. 0	$2F8420	MACRO IC 2 Node 0 Reg. 0	$2FA420
MACRO IC 0 Node 1 Reg. 0	$2F8424	MACRO IC 2 Node 1 Reg. 0	$2FA424
MACRO IC 0 Node 4 Reg. 0	$2F8428	MACRO IC 2 Node 4 Reg. 0	$2FA428
MACRO IC 0 Node 5 Reg. 0	$2F842C	MACRO IC 2 Node 5 Reg. 0	$2FA42C
MACRO IC 0 Node 8 Reg. 0	$2F8430	MACRO IC 2 Node 8 Reg. 0	$2FA430
MACRO IC 0 Node 9 Reg. 0	$2F8434	MACRO IC 2 Node 9 Reg. 0	$2FA434
MACRO IC 0 Node 12 Reg. 0	$2F8438	MACRO IC 2 Node 12 Reg. 0	$2FA438
MACRO IC 0 Node 13 Reg. 0	$2F843C	MACRO IC 2 Node 13 Reg. 0	$2FA43C
MACRO IC 1 Node 0 Reg. 0	$2F9420	MACRO IC 3 Node 0 Reg. 0	$2FB420
MACRO IC 1 Node 1 Reg. 0	$2F9424	MACRO IC 3 Node 1 Reg. 0	$2FB424
MACRO IC 1 Node 4 Reg. 0	$2F9428	MACRO IC 3 Node 4 Reg. 0	$2FB428
MACRO IC 1 Node 5 Reg. 0	$2F942C	MACRO IC 3 Node 5 Reg. 0	$2FB42C
MACRO IC 1 Node 8 Reg. 0	$2F9430	MACRO IC 3 Node 8 Reg. 0	$2FB430
MACRO IC 1 Node 9 Reg. 0	$2F9434	MACRO IC 3 Node 9 Reg. 0	$2FB434
MACRO IC 1 Node 12 Reg. 0	$2F9438	MACRO IC 3 Node 12 Reg. 0	$2FB438
MACRO IC 1 Node 13 Reg. 0	$2F943C	MACRO IC 3 Node 13 Reg. 0	$2FB43C

Note that the bit-19 mode switch has been set to 1 so that the data out of the MACRO node is not shifted. This changes the second hex digit from '7' to 'F'. Type 1 MACRO feedback comes with fractional count information in the low 5 bits, so it does not need to be shifted.

The second line of an entry for MACRO feedback should be $018000 to specify the use of 24 bits ($018) starting at bit 0 ($000).

When performing commutation of motors over the MACRO ring, it is advisable to get servo position feedback data not directly from the MACRO ring registers, as shown above, but from the motor's "previous phase position" register instead. This is where the commutation algorithm has stored the position it read from the ring (with Ixx83) for use in its next cycle.

Using this register prevents the possibility of jitter if the conversion table execution can be pushed too late in the cycle. The following table shows the first line of the conversion table entry for each motor's "previous phase position" register:

Entries for Turbo PMAC Previous Phase Position Registers

Motor #	First Line Value	Motor #	First Line Value	Motor #	First Line Value	Motor #	First Line Value
1	$2800B2	9	$2804B2	17	$2808B2	25	$280CB2
2	$280132	10	$280532	18	$280932	26	$280D32
3	$2801B2	11	$2805B2	19	$2809B2	27	$280DB2
4	$280232	12	$280632	20	$280A32	28	$280E32
5	$2802B2	13	$2806B2	21	$280AB2	29	$280EB2
6	$280332	14	$280732	22	$280B32	30	$280F32
7	$2803B2	15	$2807B2	23	$280BB2	31	$280FB2
8	$280432	16	$280832	24	$280C32	32	$281032

Note that the bit 19 mode switch has been set to 1 so that the data out of the previous phase position register from the MACRO ring is not shifted. This changes the second hex digit from '0' to '8'. Type 1 MACRO feedback comes with fractional count information in the low 5 bits, so it does not need to be shifted.

The second line of an entry for "previous phase position" feedback should be $018000 to specify the use of 24 bits ($018) starting at bit 0 ($000).

MLDT Feedback: PMAC2-style Servo ICs have the ability to interface directly to magnetostrictive linear displacement transducers (MLDTs), outputting the excitation pulse, receiving the echo pulse, and measuring the time between the two. This time is directly proportional to the distance. For this feedback the "time between last two counts" register is used like an absolute encoder. The following table shows the first line of the parallel feedback entry for each channel's timer register:

Entries for PMAC2-Style MLDT Timer Registers

Servo IC #	Chan. 1	Chan. 2	Chan. 3	Chan. 4	Notes
0	$378000	$378008	$378010	$378018	1st IC on board PMAC2, 3U stack
1	$378100	$378108	$378010	$378018	2nd IC on board PMAC2, 3U stack
2	$378200	$378208	$378210	$378218	1st ACC-24E2x, 1st IC on 1st ACC-24P/V2
3	$378300	$378308	$378310	$378318	2nd ACC-24E2x, 2nd IC on 1st ACC-24P/V2
4	$379200	$379208	$379210	$379218	3rd ACC-24E2x, 1st IC on 2nd ACC-24P/V2
5	$379300	$379308	$379310	$379318	4th ACC-24E2x, 2nd IC on 2nd ACC-24P/V2
6	$37A200	$37A208	$37A210	$37A218	5th ACC-24E2x, 1st IC on 3rd ACC-24P/V2
7	$37A300	$37A308	$37A310	$37A318	6th ACC-24E2x, 2nd IC on 3rd ACC-24P/V2
8	$37B200	$37B208	$37B210	$37B218	7th ACC-24E2x, 1st IC on 4th ACC-24P/V2
9	$37B300	$37B308	$37B310	$37B318	8th ACC-24E2x, 2nd IC on 4th ACC-24P/V2

The second line in an MLDT entry should be $013000 to specify the use of 19 bits ($013) starting at bit 0 ($000).

The third line in an MLDT entry should contain a number slightly greater than the maximum velocity ever expected, expressed as timer increments per servo cycle. An increment of the 120 MHz timer represents about 0.024mm (0.0009 in) on a typical MLDT device. This value represents the maximum change in position reading that will be passed through the conversion table in a single servo cycle, and it provides an important protection against missing or spurious echo pulses.

Time-Base Entries ($4, $9, $A, $B): A time-base entry performs a scaled digital differentiation of the value in the source register. It is most often used to perform "electronic cam" functions, slaving a motion sequence to the frequency of a master encoder. There are two types of time-base entries: "untriggered" and "triggered". An untriggered time base does not provide a specific starting point in the master source data. A triggered time base starts the differentiation upon receipt of a hardware trigger on the master encoder's channel, referenced to the position captured by that trigger. This can be used to create an absolute synchronization between the master position and the slave trajectory.

Time-base entries are two-line entries. The first setup line (I-variable) contains the method digit and the address of the source-data register. The second setup line (I-variable) contains the "time-base scale factor". The first result line contains the intermediate result value of the source data, saved for the next cycle to be able to compute the differentiation. The second result line contains the final result, which is the differentiated value. Most commonly this result is used as the time-base source for a coordinate system, so Isx93 for the coordinate system points to this second line.

Untriggered Time Base ($4): In an untriggered time-base entry, the first setup line (I-variable) contains a "4" in the method digit (bits 20 – 23) and the address of the source register in bits 0 – 18. The source register is almost always the result register of an incremental encoder entry (e.g. 1/T) higher in the table (addresses $3501 to $35C0). Refer to the table above, which lists the addresses of each line in the encoder conversion table. For example, to use the result of the fourth line of the conversion table as a source, this I-variable would be $403504.

The second setup line (I-variable) is the "time-base scale factor" which multiplies the differentiated source value. The final result value equals *2 * Time-Base-Scale-Factor * (New Source Value - Old Source Value)*. "New Source Value" and "Old Source Value" (stored from the previous servo cycle) are typically in units of 1/32 of a count, the usual scaling of a 1/T encoder conversion result.

When this time base entry is used to calculate a frequency-based time base for a coordinate system, the TBSF should be set to 2^{17}/Real-Time Input Frequency (131,072/RTIF), where the Real-Time Input Frequency (RTIF) in counts per millisecond, is the frequency at which motion trajectories using this time base will execute at the programmed speed or in the programmed time. The motion sequence to be slaved to this frequency should be written assuming that the master is always generating this real-time input frequency (so always moving at the "real-time speed"). The true speed of trajectories using this time base will vary proportionately with the actual input frequency.

Example

The application requires the use of Encoder 4 on board a Turbo PMAC2 as an untriggered time-base master for Coordinate System 1. The real-time input frequency is selected as 256 counts/msec. The conversion table starts with 8 single-line entries in I8000 – I8007, with the 4th line (I8003) doing a 1/T conversion of Encoder 4.

```
; Setup on-line commands
I8003=$078018    ; 1/T conversion of Encoder 4
I8008=$403504    ; Unriggered time base from 1/T encoder
I8009=512        ; TBSF=131072/256
I5193=@I8009     ; C.S.1 use I8009 result for time base
```

Triggered Time Base ($9, $A, $B): A "triggered" time-base entry is like a regular "untriggered" time-base entry, except that it is easy to "freeze" the time base, then start it exactly on receipt of a trigger that captures the "starting" master position or time.

In a triggered time-base entry, the first setup line (I-variable) contains a '9' 'A' or 'B' in the method digit (bits 20 – 23), depending on its present state. It contains the address of the source register in bits 0 – 18. The source register for triggered time base must be the starting (X) address for one of the machine interface channels of a Servo IC. The bit 19 mode switch must be set to 0 if a PMAC(1)-style Servo IC ("DSPGATE") is addressed; it must be set to 1 if a PMAC2-style Servo or MACRO IC ("DSPGATE1" or "DSPGATE2") is addressed. Note that setting bit 19 to 1 changes the second hex digit of the I-variable typically from '7' to 'F'.

The second setup line (I-variable) is the "time-base scale factor" which multiplies the differentiated source value. The final result value (when running) equals *512 * Time-Base-Scale-Factor * (New Source Count - Old Source Count)*. "New Source Count" and "Old Source Count" are the values of the addressed encoder counter, in counts.

When this time-base entry is used to calculate a frequency-based time base for a coordinate system, the TBSF should be set to 2^{14}/Real-Time Input Frequency (16,384/RTIF), where the Real-Time Input Frequency (RTIF) in counts per millisecond, is the frequency at which motion trajectories using this time base will execute at the programmed speed or in the programmed time. (Note that the TBSF is 1/8 of the value for an untriggered time base, because the triggered time base creates an extra 3 bits [8x] of fractional information with its 1/T extension.) The motion sequence to be slaved to this frequency should be written assuming that the master is always generating this real-time input frequency (so always moving at the "real-time speed"). The true speed of trajectories using this time base will vary proportionately with the actual input frequency.

A triggered time-base entry in Turbo PMAC automatically computes the "1/T" count extension of the input frequency itself before the differentiation. It computes this to 1/256 of a count. This is compared to the 1/32 of a count that the separate 1/T encoder extension uses. The extra fractional information can reduce the quantization noise created by the differentiation and provide smoother operation under external time base.

Note: The intermediate result in the first line of a triggered time-base entry contains the undifferentiated 1/T extension of the source encoder position, in units of 1/256 of a count. This value can be used as feedback data or master position data, with more resolution than the standard 1/T extension.

In use, the method digit (comprising bits 20-23 of the first line) is changed as needed by setting of the I-variable. Triggered time base has three states, "frozen", "armed", and "running", all of which must be used to utilize the triggering feature.

First, the method digit is set to $9 (e.g. I8010=$978008) before the calculations of the triggered move are started, to freeze the time base (and therefore the motion) while the move calculations are done. This is typically done in the user's motion program. When this entry is in the frozen state, the table reads the channel's capture position register each servo cycle to ensure the triggering logic is reset for the next capture. The final result of the entry is always 0 when frozen.

Note: In a Turbo PMAC application with a light computational load, it is possible that the entry will not be in the "frozen" state during a servo interrupt, and the table will not get a chance to reset the trigger logic. Therefore, it is advisable to reset the triggering logic explicitly in the user program with a "dummy" read of the channel's captured position register, which is the X-register with an address 3 greater than the address specified in the entry (e.g. X:$07800B if the entry specifies $078008). The suggested M-variable for the captured position register is Mxx03.

Next, the method digit is set to $B (e.g. I8010=$B78008) after the calculations of the triggered move are finished, to "arm" the time base for the trigger. This is typically done in a PLC program that simply looks to see if the entry is frozen and changes it to the armed state. The final result of the entry is always 0 when armed.

In the armed state, the Table checks every servo cycle for the channel's trigger bit to be set. When the Table sees the trigger (the capture trigger for the machine interface channel as defined by I7mn2 and I7mn3 for Servo IC m Channel n, or by I68n2 and I68n3 for MACRO IC 0 Channel n), it automatically sets the method digit to $A for "running" time base. It uses the position captured by the trigger as the starting position ("time zero") for the running time base. (Those using this method for the reduced quantization noise may simply leave the method digit at $A.)

The following tables show the possible 1st-line entries for triggered time base (running mode):

Triggered Time-Base Entries for PMAC(1)-Style Servo ICs (Running State)

Servo IC #	Chan. 1	Chan. 2	Chan. 3	Chan. 4	Notes
0	$A78000	$A78004	$A78008	$A7800C	1st IC on board PMAC
1	$A78100	$A78104	$A78108	$A7810C	2nd IC on board PMAC
2	$A78200	$A78204	$A78208	$A7820C	1st IC on 1st ACC-24P/V
3	$A78300	$A78304	$A78308	$A7830C	2nd IC on 1st ACC-24P/V
4	$A79200	$A79204	$A79208	$A7920C	1st IC on 2nd ACC-24P/V
5	$A79300	$A79304	$A79308	$A7930C	2nd IC on 2nd ACC-24P/V
6	$A7A200	$A7A204	$A7A208	$A7A20C	1st IC on 3rd ACC-24P/V
7	$A7A300	$A7A304	$A7A308	$A7A30C	2nd IC on 3rd ACC-24P/V
8	$A7B200	$A7B204	$A7B208	$A7B20C	1st IC on 4th ACC-24P/V
9	$A7B300	$A7B304	$A7B308	$A7B30C	2nd IC on 4th ACC-24P/V

Triggered Time-Base Entries for PMAC2-Style Servo ICs (Running State)

Servo IC #	Chan. 1	Chan. 2	Chan. 3	Chan. 4	Notes
0	$AF8000	$AF8008	$AF8010	$AF8018	1st IC on board PMAC2, 3U stack
1	$AF8100	$AF8108	$AF8010	$AF8018	2nd IC on board PMAC2, 3U stack
2	$AF8200	$AF8208	$AF8210	$AF8218	1st ACC-24E2x, 1st IC on 1st ACC-24P/V2
3	$AF8300	$AF8308	$AF8310	$AF8318	2nd ACC-24E2x, 2nd IC on 1st ACC-24P/V2
4	$AF9200	$AF9208	$AF9210	$AF9218	3rd ACC-24E2x, 1st IC on 2nd ACC-24P/V2
5	$AF9300	$AF9308	$AF9310	$AF9318	4th ACC-24E2x, 2nd IC on 2nd ACC-24P/V2
6	$AFA200	$AFA208	$AFA210	$AFA218	5th ACC-24E2x, 1st IC on 3rd ACC-24P/V2
7	$AFA300	$AFA308	$AFA310	$AFA318	6th ACC-24E2x, 2nd IC on 3rd ACC-24P/V2
8	$AFB200	$AFB208	$AFB210	$AFB218	7th ACC-24E2x, 1st IC on 4th ACC-24P/V2
9	$AFB300	$AFB308	$AFB310	$AFB318	8th ACC-24E2x, 2nd IC on 4th ACC-24P/V2

Entries for PMAC2 MACRO IC 0

Handwheel Channel #	PMAC2
Channel 1	$AF8410
Channel 2	$AF8418

Example

The application requires the use of Encoder 4 on board a Turbo PMAC2 as a triggered time base master for coordinate system 1. It is to be triggered by the rising edge of its index channel. The real-time input frequency is selected as 256 counts/msec. The conversion table starts with 8 single-line entries in I8000 – I8007.

```
; Setup on-line command
I8008=$AF8018    ; Triggered time base from PMAC2 channel 4
I8009=64         ; TBSF=16384/256
I7042=1          ; Servo IC 0 Channel 4 trigger on rising index
I5193=@I8009     ; C.S.1 use I8009 result for time base
M403->X:$07801B,0,24,S     ; Channels' captured position register

; Motion program segment
DWELL 0          ; Stop any lookahead
I8008=$9F8018    ; Freeze the time base
P403=M403        ; Dummy read to ensure capture logic reset
X10              ; Calculate first move

; PLC program segment
IF (I8008=$9F8018)    ; If frozen
  I8008=$BF8018       ; Then arm
ENDIF
```

Exponential-Filter Entries ($D): The $D entry is used to create an exponential filter on a word of input data. This is particularly useful for smoothing master position values in position following (electronic gearing) or external time-base (electronic cam) applications, especially when the slave is "geared up" from the master; i.e. the slave moves more than one count for each count of the master, where it can significantly smooth the motion of the following axis. Exponential filters are seldom used on feedback position values, because the delay introduced by the filter has a destabilizing effect on the servo loop.

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

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

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.

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) and the address of the source X-register in bits 0 – 18. Bit 19 is not used. 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:$3501 – X:$35CF). 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.

Filter Gain Word: The second 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.

Maximum Change Word: The third 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.

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).

Addition/Subtraction of Entries ($E): The $E entry is used to add the results of two other entries in the Table, possibly after negating one or both of them (which can effectively create subtraction), with the option of integrating the sum. It is a single-line entry.

Control Digit: The second hex digit of the I-variable consists of four independent control bits (bits 19-16) and determines whether the result is integrated or not, whether a second source entry is used or not, and whether each of the source entries is negated before addition or not.

If the bit 19 mode switch bit is 0, which makes the second hex digit 0, the values in the two specified entries are simply added. If the mode switch bit 19 is 1, the sum of the two entries.

If bit 18 is set to 1, the second entry to be added (as specified by bits 8-15) is not used. This permits easy negation (change in sign) of a single entry. If bit 18 is set to 0, the second entry is used.

If bit 17 is set to 1, the second entry to be added (as specified by bits 8-15) is negated before the addition, which means that it is effectively subtracted from the first entry. If bit 17 is not set to 1, the second entry to be added is not negated.

If bit 16 is set to 1, the first entry to be added (as specified by bits 0-7) is negated before the addition, which means that it is effectively subtracted. If bit 16 is not set to 1, the first entry to be added is not negated.

Second Source Offset: Bits 8-15, which form the third and fourth hex digits of the entry, specify the offset from the beginning of the table to the second entry to be used used, as an unsigned 8-bit quantity. The value in these digits should equal the number of the I-variable matching the second entry minus 8000.

First Source Offset: Bits 0-7, which form the fifth and sixth hex digits of the entry, specify the address offset from the beginning of the table to the first entry to be used, as an unsigned 8-bit quantity. The value in these digits should equal the number of the I-variable matching the first entry minus 8000.

Examples:

To add the results of the first two lines in the table, from I8000 and I8001, the I-variable would be $E00100. The ‘E’ specifies addition, the ‘0’ specifies no integration, using the second source, and no negation of either source. The ‘01’ specifies the second line of the table (matching I8001) as the second source, and the final ‘00’ specifies the first line of the table (matching I8000) as the first source.

To subtract the result of the second line (from I8001) of the table from that of the first line (from I8000), the I-variable would be $E20100. The ‘E’ specifies addition, the ‘2’ (0010 binary) specifies no integration, using the second source, negating the second source, but not the first source. The ‘01’ specifies the second line of the table (matching I8001) as the second source, and the final ‘00’ specifies the first line of the table (matching I8000) as the first source.

To invert the 20th line of the table (from I8019), the I-variable would be $E50013. The ‘E’ specifies “addition”, the ‘5’ (0101 binary) specifies no integration, not using the second source, and negating the first source. The ‘00’ is not important, because the second source is not used. The ‘13’ (19 decimal) specifies the result matching I8019 as the first source.

Extended Entries ($F): Encoder conversion table entries in which the first hex digit of the first line is $F are “extended entries”. In these entries, the actual method is dependent on the first digit of the second line. Extended entries are a minimum of 2 lines.

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.

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:

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 line 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:

High-Res Interpolator Entry First 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

A/D Bias Term: The third line of the entry contains the bias in the A/D converter values. This line should contain the value that the A/D converters report when they should ideally report zero. Turbo PMAC subtracts this value from both A/D readings before calculating the arctangent. Many users will leave this value at 0, but it is particularly useful to remove the offsets of single-ended analog encoder signals.

This line is scaled so that the maximum A/D converter reading provides the full value of the 24-bit register (+/-2^{23}, or +/-8,388,608). It is generally set by reading the A/D converter values directly as 24-bit values, computing the average value over a cycle or cycles, and entering this value here.

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:

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

Byte-Wide Parallel Feedback Entries ($F/$2, $F/$3): 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 $2 or $3 processes the result of a parallel data feedback source whose data is in byte-wide pieces in consecutive Y-words. This is used to process feedback from 3U-format parallel-data I/O boards: the ACC-3E in stack form, and the ACC-14E in pack (UMAC) form.

Address Word: The first setup line (I-variable) of the entry contains $F in the first hex digit (bits 20-23). The bit-19 mode-switch bit in the first line controls whether the least significant bit (LSB) of the source register is placed in bit 5 of the result register ("normal shift"), providing the standard 5 bits of (non-existent) fraction, or the LSB is placed in Bit 0 of the result register ("unshifted"), creating no fractional bits.

Normally, the Bit-19 mode switch is set to 0 to place the source LSB in Bit 5 of the result register. Bit 19 is set to 1 to place to source LSB in Bit 0 of the result register for one of three reasons:

- The data already comes with 5 bits of fraction, as from a Compact MACRO Station.
- The normal shift limits the maximum velocity too much ($V_{max}<2^{18}$ LSBs per servo cycle)
- The normal shift limits the position range too much (Range$<\pm 2^{47}$/Ix08/32 LSBs)

Unless this is done because the data already contains fractional information, the "unshifted" conversion will mean that the motor position loop will consider 1 LSB of the source to be 1/32 of a count, instead of 1 count.

Bits 0 to 18 of the first line contain the base address of the parallel data to be read. This is the address of the least significant byte in the parallel feedback word. The following table shows the possible entries when an ACC-3E stack I/O board is used:

Entry First Lines for ACC-3E 3U-Stack I/O Boards

ACC-3E Address Jumper	E1	E2	E3	E4
First-Line Value	$F7880x	$F7890x	$F78A0x	$F78B0x

The following table shows the possible entries when the ACC-14E UMAC I/O board is used:

Entry First Lines for ACC-14E UMAC I/O Boards

DIP-Switch Setting	SW1-1 ON (0) SW1-2 ON (0)	SW1-1 OFF (1) SW1-2 ON (0)	SW1-1 ON (0) SW1-2 OFF (1)	SW1-1 OFF (1) SW1-2 OFF (1)
SW1-3 ON (0) SW1-4 ON (0)	$F78C0x	$F78D0x	$F78E0x	$F78F0x
SW1-3 OFF (1) SW1-4 ON (0)	$F79C0x	$F79D0x	$F79E0x	$F79F0x
SW1-3 ON (0) SW1-4 OFF (1)	$F7AC0x	$F7AD0x	$F7AE0x	$F7AF0x
SW1-3 OFF (1) SW1-4 OFF (1)	$F7BC0x	$F7BD0x	$F7BE0x	$F7BF0x

A switch that is ON is CLOSED; a switch that is OFF is OPEN.

In both of these tables, the second digit should be changed from a '7' to an 'F' if bit 19 is set to 1 to disable the data shift.

The final digit, represented by an 'x' in both of these tables, can take a value of 0 to 5, depending on which I/O point on the board is used for the LSB:

- x=0: I/O00-07 I/O48-55 I/O96-103
- x=1: I/O08-15 I/O56-63 I/O104-111
- x=2: I/O16-23 I/O64-71 I/O112-119
- x=3: I/O24-31 I/O72-79 I/O120-127
- x=4: I/O32-39 I/O80-87 I/O128-135
- x=5: I/O40-47 I/O88-95 I/O136-143

Width/Offset Word: The second setup line (I-variable) of this parallel read entry contains information about what data is to be read starting at the base address. This 24-bit value, usually represented as 6 hexadecimal digits, is split into four parts, as shown in the following table.

Hex Digit	1	2	3	4	5	6
Contents	2 or 3	Bit Width		Byte	LSB Location	

The first hex digit contains a 2 or a 3. If it has a 2, there is no filtering of the data, and the entry is a 2-line entry. If it has a 3, the input data is filtered to protect against noise or data corruption, and the entry is a 3-line entry, with the third line controlling the filtering.

The second and third digits represent the width of the parallel data in bits, and can range from $01 (1 bit wide – not of much practical use) to $18 (24 bits wide). If the value of these digits is from $01 to $08, only the base address in the first line is used. If the value of these digits is from $09 to $10 (16), the base address and the next higher-numbered address are used. If the value of these digits is from $11 to $18 (17 to 24), three addresses starting at the base address are used.

The fourth digit represents which byte of the source words is used. It has three valid values:

- 0: Low byte (bits 0 – 7)
- 1: Middle byte (bits 8 – 15)
- 2: High byte (bits 16 – 23)

The fifth and sixth digits contain the bit location of the LSB of the data in the source word at the base address, and can range from $00 (Bit 0 of the source address is the LSB), through $07 (Bit 7 of the source address is the LSB). To calculate this value, divide the number of the I/O point used for the LSB by 8 and use the remainder here. For example, if I/O19 is used for the LSB, the remainder of 19/16 is 3.

Maximum Change Word: If the method character for a parallel read is $3 or $7, specifying "filtered" parallel read, there is a third setup line (I-variable) for the entry. This third line contains the maximum change in the source data in a single cycle that will be reflected in the processed result, expressed in LSBs per servo cycle. The filtering that this creates provides an important protection against noise and misreading of data. This number is effectively a velocity value, and should be set slightly greater than the maximum true velocity ever expected.

New/Revised On-Line Commands/Descriptions

DEFINE COMP (two-dimensional) {revised description}

Function: Define two-dimensional position compensation table
Scope: Motor-specific
Syntax:
```
DEFINE COMP {Rows}.{Columns}, #{RowMotor}[D],
[#{ColumnMotor}[D], [#{TargetMotor}]],
{RowSpan},{ColumnSpan}

DEF COMP...
```

where:

- **{Rows}** is a positive integer constant representing the number of rows in the table, where each row represents a fixed location of the second (*column*) source motor;
- **{Columns}** is a positive integer constant representing the number of columns in the table, where each column represents a fixed location of the first (*row*) source motor;
- **{RowMotor}** is an integer constant from 1 to 32 representing the number of the first source motor; defaults to addressed motor; if a **D** is specified after the source motor number, the desired position of the motor is used to calculate the correction; otherwise the actual position is used;
- **{ColumnMotor}** is an integer constant from 1 to 32 representing the number of the second source motor; if a **D** is specified after the source motor number, the desired position of the motor is used to calculate the correction; otherwise the actual position is used;
- **{TargetMotor}** is an integer constant from 1 to 32 representing the number of the target motor; defaults to addressed motor;
- **{RowSpan}** is the span of the table, in counts, along the first (row) source motor's travel;
  - **{ColumnSpan}** is the span of the table, in counts, along the second (column) source motor's travel.

Remarks

This command establishes a two-dimensional position compensation table assigned to the addressed motor. The next *(Rows+1)*(Columns+1)-1* constants sent to Turbo PMAC will be placed into this table. This type of table is usually used to correct a motor position (X, Y, or Z-axis) as a function of the planar position of two motors (e.g. X and Y axes). Once defined, the tables are enabled and disabled with the variable I51.

The table "belongs" to the currently addressed motor, and unless otherwise specified in the command line, it will use the addressed motor both as the first-motor source position data and as the target for its corrections. Each motor can only have one table that belongs to it (for a total of 32 tables in one PMAC), but it can act as a source and/or a target for multiple tables.

PMAC will reject this command, reporting an ERR003 if I6=1 or 3, if any COMP buffer exists for a lower numbered motor, or if any TCOMP, BLCOMP, TBUF, ROTARY, or GATHER buffer exists. Any of these buffers must be DELETEd first. COMP buffers must be DEFINEd from high-numbered motor to low-numbered motor, and DELETEd from low-numbered motor to high-numbered motor.

The first source motor must be specified in the command line with **#{RowMotor}**. The second source motor may be specified in the command line with **#{ColumnMotor}**; if it is not specified, Turbo PMAC assumes that the second source motor is the currently addressed motor. The target motor may be specified with **#{TargetMotor}**; if it is not specified, Turbo PMAC assumes that the target motor is the currently addressed motor.

In other words, if only one motor is specified in the command line, it is the first ("row") source motor, and the second ("column") source and target motors default to the addressed motor. If two motors are specified in the command line, the first one specified is the first ("row") source motor, the second is the second ("column") source motor, and the target motor defaults to the addressed motor. If three motors are specified, the first is the first ("row") source motor, the second is the second ("column") source motor, and the third is the target motor. None of these motors is required to be the addressed motor.

It is strongly recommended that you explicitly specify both source motors and the target motor in this command to prevent possible confusion.

The table can operate as a function of either the desired (commanded) or actual position of the source motors. If a '**D**' is entered immediately after the source motor number (which must be explicitly declared here), the table operates as a function of the desired position of the source motor; if no '**D**' is entered, the table operates as a function of the actual position of the source motor. If the target motor is also one of the source motors, it is recommended that desired position be used, especially in high-gain systems, to prevent interaction with the servo dynamics.

The last two items on the command line, **{RowSpan}** and **{ColumnSpan}**, specify the span of the compensation table for the two source motors, "row" and "column" respectively, expressed in encoder counts of those motors. In use, if the source motor position goes outside of the range 0 to **{Span}**, the source position is "rolled over" to within this range along this axis before the correction is computed.

The count spacing between columns in the table is **{RowSpan}** divided by **{Columns}**. The count spacing between rows in the table is **{ColumnSpan}** divided by **{Rows}**. Note carefully the interaction between the row parameters and the column parameters.

On succeeding command lines will be given the actual correction entries of the table, given as integer numerical constants in text form. The units of these entries are 1/16 count, and the entries must be integer values. The first entry is the correction at one column spacing from the zero position of the **RowMotor**, and the zero position of the **ColumnMotor**. The second entry is the correction at two column spacings from the zero position of the **RowMotor**, and the zero position of the **ColumnMotor**, and so on. Entry number "**Columns**" is the correction at **RowSpan** counts of the **RowMotor**, and at the zero position of the **ColumnMotor** (this entry should be zero if you wish to use the table along the edge, to match the implied zero correction at the origin). These entries should be considered as constituting "Row 0" of the table.

The next entry (entry **Columns**+1, the first entry of Row 1) is the correction at the zero position of the **RowMotor**, and one row spacing of the **ColumnMotor**. The following entry is the correction at one column spacing of the **RowMotor** and one row spacing of the **ColumnMotor**. The entry after this is the correction at two column spacing of the **RowMotor** and one row spacings of the **ColumnMotor**., and so on. The last entry of Row 1 (entry 2***Columns**+1) is the correction at one row spacing of the **RowMotor**, and **RowSpan** counts of the **ColumnMotor**.

Subsequent rows are added in this fashion, with the corrections of the entries for Row *n* being at *n* row spacings from the zero position of the **ColumnMotor**. The last row (row **Rows**) contains corrections at **ColumnSpan** counts of the **ColumnMotor**.

The size of the table is limited only by available data buffer space in Turbo PMAC's memory. The following chart shows the order of entries into a 2D table with *r* rows and *c* columns, covering a span along the row motor of *RowSpan*, and along the column motor of *ColSpan*:

	Column Motor Position v	Col 0	Col 1	Col 2	(Col j)	Col *c*
Row Motor Position >		0	RowSpan c	2*RowSpan c		RowSpan
Row 0	0	[0]	E_1	E_2	…	E_c
Row 1	ColSpan r	E_{c+1}	E_{c+2}	E_{c+3}	…	E_{2c+1}
Row 2	2*ColSpan r	E_{2c+2}	E_{2c+3}	E_{2c+4}	…	E_{3c+2}
(Row i)		…	…	…	(E_{ic+I+j})	…
Row *r*	ColSpan	E_{rc+r}	E_{rc+r+1}	E_{rc+r+2}		E_{rc+r+c}

There are several important details to note in the entry of a 2D table:

- The number of rows and number of columns is separated by a period, not a comma.
- The correction to the target motor at the zero position of both source motors is zero by definition. This is an implied entry at the beginning of the table (shown by [0] in the above chart); it should not be explicitly entered.
- Consecutive entries in the table are in the same row (except at row's end) separated by one "column spacing" of the position of the first source ("row") motor.
- Both Row 0 and Row *r* must be entered into the table, so effectively you are entering (*r*+1) rows. If there is any possibility that you may go beyond an edge of the table, matching entries of Row 0 and Row *r* should have the same value to prevent a discontinuity in the correction. Row *r* in the table may simply be an added row beyond your real range of concern used just to prevent possible discontinuities at the edges of your real range of concern.

- Both Column 0 and Column *c* must be entered into the table, so effectively you are entering (*c*+1) columns. If there is any possibility that you may go beyond an edge of the table, matching entries of Column 0 and Column *c* should have the same value to prevent a discontinuity in the correction. Column *c* in the table may simply be an added column beyond your real range of concern used just to prevent possible discontinuities at the edges of your real range of concern.

- Because the outside rows and outside columns must match each other to prevent edge discontinuities, the three explicitly entered corner corrections must be zero to match the implicit zero correction at the first corner of the table.

Examples

```
#1 DEFINE COMP 40.30,#1,#2,#3,300000,400000
                    ; Create table belonging to Motor 1
ERR007              ; Turbo PMAC rejects this command
DELETE GATHER       ; Clear other buffers to allow loading
&1 DELETE ROTARY
&2 DELETE ROTARY
#2 DELETE COMP
#3 DELETE COMP
#4 DEFINE COMP 30.40,#1,#2,#3,400000,300000
                    ; Create same table, now belonging to Motor 4;
                    ; #1 & #2 are sources, #3 is target;
                    ; 30 rows x 40 columns, spacing of 10,000 counts
(1270 entries)      ; (30+1)*(40+1)-1 entries of constants
#3 DEFINE COMP 25.20,#2,#3,#1,200000,250000
                    ; Create table belonging to Motor 3;
                    ; #2 and #3 are sources, #1 is target;
                    ; 25 rows x 20 columns, spacing of 10,000 counts
(545 entries)       ;(25+1)*(20+1)-1 entries of constants
#2 DEFINE COMP 10.10,#1,#4,10000,20000
                    ; Create table belonging to Motor 2;
                    ; #1 and #4 are sources, #2 (default) is target;
                    ; 10 rows x10 columns, spacing of 1000 cts between columns;
                    ; spacing of 2000 cts between rows;
(120 entries)       ; (10+1)*(10+1)-1 entries of constants
#1 DEFINE COMP 12.10,#4,1280,1200
                    ; Create table belonging to Motor 1;
                    ; #4 and #1 (default) are sources, #1 (default) is target;
                    ; 12 rows x 10 columns; spacing of 128 cts between columns;
                    ; spacing of 100 cts between rows
(142 entries)       ; (12+1)*(10+1)-1 entries of constants
I51=1               ; Enable compensation tables
```

Firmware Update Listing

V1.936 Updates (April 2000)

Note: After upgrading an older system to V1.936 and either getting the old configuration back from flash memory or reloading a configuration file, the user should issue an **I20..24=*** *command to set up any MACRO ICs and DPRAM ICs properly. If users had non-zero values in their old configuration for I20 and I21, these values should be entered into I40 and I41 respectively. After this,* **SAVE** *the configuration and reset the board normally.*

1. Added support for UMAC Turbo systems.
2. Added new variables I20 – I23 to specify base addresses of MACRO ICs 0 – 3 respectively, providing flexibility in MACRO ring configurations on UMAC Turbo. These must be set properly to support automatic firmware functions using these ICs, including multiplexer port functions, display port, and I6800 – I6999.
3. Moved old variables I20 (watchdog timer reset value) and I21 (I-variable lockout control) to their proper locations of I40 and I41.
4. Default values of "address" I-variables made more system-specific to reflect what components such as Servo ICs and MACRO ICs are actually found by the processor.
5. Added 3D cutter-radius compensation with new program commands **CC3**, **NX{data}**, **NY{data}**, **NZ(data}**, **TR{data}**, **TX{data}**, **TY{data}**, and **TZ{data}**.
6. Added "altered destination" RAPID mode move on-line command **!{axis}{data}**… to be able to break into currently executing RAPID-mode move and change the move on the fly to a new destination, or execute a RAPID-mode move directly from an on-line command.
7. Extended I49nn controller configuration status I-variables
8. Made the communications ports independent with respect to opening of program buffers. Only the port over which the **OPEN** command was issued can accept buffered program commands, **LIST** the open buffer, **LEARN** points into the open buffer, and **CLOSE** the buffer. Other ports can be used simultaneously for on-line commands.
9. Refined error reporting when **CLOSE**ing a program buffer missing **ENDIF** and/or **ENDWHILE**. Now reports ERR009 (program structure error), and only reports the error on **CLOSE**ing this particular buffer.
10. Extended Ixx91, Ixx95, and I8000 – I8191 to support parallel position reads in byte-wide sections from ACC-3E1 and ACC-14E boards.
11. Corrected problem with hardware position capture over MACRO in V.1933,4,5.
12. Corrected problem in **DELETE GATHER** command that could cause buffer management problems.
13. Corrected problem in cutter compensation in sequencing with non-compensated moves (RAPID, DWELL, out-of-plane).
14. Corrected problem in cutter compensation with CIRCLE mode lead-in moves.
15. Corrected operation of Isx91 default program parameter after **$$$** software reset, and in repeated execution of program.
16. Corrected absolute phase position read of resolver-to-digital converter through MACRO station.

17. Corrected **PMATCH** problem when linear set of axes X, Y, and Z, or U, V, and W were defined "out of order" (e.g. #1->X, #2->Z, #3->Y).
18. Corrected operation of **PMATCH** when called from within a motion program (**CMD "&nPMATCH"**).

V1.937 Updates (November, 2000)

1. Changed I5061 to I5076 A/D de-multiplexing pointer variables to contain the full address of the A/D register, not just the offset from $078800. The old default value of 0 still selects $078800, but $078800 must be added to existing non-zero values to maintain compatibility.
2. Automatically sets I58 to 1, enabling DPRAM ASCII communications, at power-up/reset, if any DPRAM IC is detected.
3. Fixed operation of **J!** command so that commanded position is always rounded to nearest integer number of counts, regardless of the size of the following error.
4. Fixed glitch at the center 1/8-millionth section of long compensation tables (> ½-million counts long).
5. Fixed operation of background variable read buffer in multi-user mode.
6. Added foreground "in-position" check in servo interrupt, enabled by I13=1. Added foreground in-position motor status bit – bit 13 of Y:$0000C0, etc.
7. Permitted disabling of automatic command parsing on Option 9T auxiliary serial port by setting I43 to 1, permitting custom parsing algorithms to be written for serial input of data that is not in PMAC command format.
8. Permitted loading of binary rotary motion program commands from DPRAM to internal buffer as a foreground real-time interrupt task instead of a background task with I45=1.
9. Permitted disabling of A/D de-multiplexing with I5080=0.
10. Implemented alternate rotary-axis rollover mode in which the sign of the specified destination value also specifies the direction to turn to that point. Setting Ixx27 to a negative value enables this mode.
11. Implemented |I|T integrated current limiting function as alternate to existing I^2T integrated current limiting. In |I|T, the magnitude of the current itself, not the square of the current, is integrated and compared to the Ixx58 limit. Setting the Ixx57 continuous current magnitude value to a negative number enables this alternate mode. This more accurately tracks the thermal behavior of a constant voltage-drop device such as an IGBT, whereas I^2T is better for constant-resistance devices such as MOSFETs and motor windings.
12. Implemented automatic clearing of direct current-loop registers to improve direct-PWM control of permanent-magnet brush motors. This mode is enabled by setting Ixx96 to 1 when Ixx01 bit 0 = 1 (enabling commutation) and Ixx82 > 0 (enabling current loop).
13. Implemented **ABR** command, permitting fastest possible abort of currently executing program, and start or restart of a motion program.
14. Implemented "time remaining in move" register and **MOVETIME** query command, to support functions initiated at a fixed time before the end of a commanded move.
15. Implemented **SETPHASE** command (on-line and buffered) to copy Ixx75 phase value into phase position register. Useful for correcting the phase position at a known point (e.g. the index pulse) after an initial rough phasing (e.g. from Hall commutation sensors).
16. Implemented **LOCK** and **UNLOCK** commands (on-line and buffered) to control up to 8 process locking bits that can prevent possible conflict of foreground and background tasks attempting to manipulate the same register.

17. Implemented on-line **I{constant}=@I{constant}** command, permitting the value of one I-variable to be set to the address of another I-variable. The main purpose of this command is to be able to set an address I-variable (e.g. Ixx03, Ixx04, Ixx05, Isx93) to the address of a conversion-table entry without having to look up the address of that entry.
18. Added capability for UMAC Turbo CPUs to generate their own servo and phase clock signals when expected clock source is not found. Keeps watchdog timer from tripping so that new clock source can be established. Bit 3 of X:$000006 set if CPU is generating its own clocks.
19. Resolution of (previously undocumented) real-time clock register L:$000017 changed from 1/256 second to 1/1024 second.
20. Turbo PMACs with extended user data memory options (5x1 or 5x3) have default user buffer of 65,536 words.
21. I52 CPU clock speed multiplier parameter range extended to 14 to support CPU speeds of up to 150 MHz.
22. Resolution of (previously undocumented) real-time interrupt cycle time registers X:$00000B (latest time) and Y:$00000B (maximum time) changed to 2 CPU clock cycles.
23. Added background cycle time registers X:$000022 (latest time) and Y:$000022 (maximum time) with resolution of 2 CPU clock cycles.
24. Internal-use global status bits "Servo Active" (X:$000006 bit 21) and "RTI Active" (X:$000006 bit 23) removed.
25. Implemented anti-windup protection for current-loop integrators. If calculated output is more than 9/8 of saturated output, integrator value is reduced to that which would produce 9/8 of saturation.
26. Corrected operation of cutter compensation for compensated inside corner immediately following inside-corner introduction of compensation.
27. Extended I68n5/I69n5 and I7mn5 encoder variables to support de-multiplexing of hall commutation states from Yaskawa encoder third channel ("B" or newer revision of DSPGATE1/2 Servo/MACRO IC required).
28. Corrected operation of **LIST BLCOMP DEF** and **LIST TCOMP DEF** commands.
29. Corrected algorithm in compiled PLCs for taking INT of a quotient.
30. Corrected deadband gain algorithm for true deadband (Ixx64=-8) with small motor scale factor (Ixx08~1). With pulse-and-direction output, the previous small remaining residual could cause dithering.
31. Added variable I7mn9 for PMAC2-style Servo ICs to enable "hardware-1/T" interpolation. Only useful for "D" revision and newer Servo ICs.

V1.938 Updates (June, 2001)

1. Corrected operation of the **UNDEFINE** command so it only clears axis definitions in the addressed coordinate system.
2. Corrected operation of DPRAM binary rotary buffer download through USB interface.
3. Increased commanded velocity saturation value for jog-to-position and RAPID-mode moves from 256M/Ix08 counts/second to 768M/Ix08 counts/second, consistent with other types of moves.
4. Improved noise immunity in conversion-table algorithms for ACC-51 high-resolution analog-encoder interpolators to decrease chance of "quadrant" errors due to high noise on analog lines.
5. Corrected VME mailbox communications so that responses of more than 15 characters can be read properly.

6. Implemented support for ACC-57E Yaskawa/Mitsubishi absolute encoder interface board in Ixx10 and Ixx95 variables. See ACC-57E manual for details.
7. Implemented support for "I-button" real-time clock/calendar chip on new "Flex" CPU design.
8. Limited number of commands that can possibly be pulled off individual port in one background cycle to 8 to prevent possibility of trapping the background cycle with very high-speed communications.
9. Corrected problem with **DEFINE UBUF** when background PLCs are enabled.
10. Added support for on-board IEC-1131 ladder/sequential-function-chart programs.
11. Added support for "Open Servo" compiled servo algorithms
12. Supported negative values of Ixx99 for Yaskawa Sigma I absolute encoders to permit reversal of the direction sense.
13. Corrected saving and restoring of the value of new variable I7mn9.

V1.939 Updates (March, 2002)

1. Added support for DSP56311 CPU (Option 5Ex).
2. Added new status bit at X:$000006 bit 21 that is set to 1 to indicate that CPU is DSP56311 type (X:$000006 bit 21 also set to 1 in this case).
3. Moved location of main serial-port communications buffer from $001Exx to $0036xx. Moved location of host-bus port communications buffer from $001Fxx to $0037xx. Moved location of synchronous M-variable buffer from $0036xx and $0037xx to $001Exx and $001Fxx. Changes necessary to support DSP56311 (Option 5Ex) properly.
4. Increased maximum value of Ixx71 Commutation Cycle Size variable from 8,388,607 to 16,777,215.
5. Changed range of Ixx75 Phase Position Offset variable from –8,388,608 – +8,388,607 to 0 – 16,777,215. If a negative value of Ixx75 is specified, it is stored as (Ixx71 + Ixx75), which provides the same effect (the proper value of Ixx71 must already be specified for the motor).
6. Added new variable I12 to better support on-the-fly changes in vector feedrate during lookahead.
7. Added new variable I30 to support automatic "wrapping" of compensation tables (the last entry in the table "wraps" to become the correction at zero position as well). Existing documentation incorrectly reported that this was done always in earlier firmware versions, but correction at zero position was always zero, regardless of last entry.
8. Added support for "hardware 1/T" using D-revision or newer PMAC2-style "DSPGATE1" Servo ICs. In new conversion table method ($C with mode bit set), the IC computes the timer-based fractional count value in hardware; the conversion table simply combines it with the whole-count value. This permits use of the alternate timer mode for sub-count capture and compare.
9. Computational efficiency of dual-ported RAM data reporting buffers was improved.
10. Computational efficiency of commutation calculations was improved about 20%.
11. Permitted "foreign" characters (ASCII value > 127) to be accepted in comments (after semi-colon) without causing an error to be reported.
12. Modified timing of multiplexer port interface signals to ACC-34 boards so they will work properly with Option 5Ex 160MHz CPUs.
13. Fixed problem with BREQ "buffer request" interrupt on ISA/PCI bus.

V1.940 Updates (June, 2003)

1. Added support for DSP56321 CPU (Option 5Fx). Range of I52 CPU Frequency Control variable extended to 31 (Option 5Fx can run at up to 240 MHz, with I52=23).
2. Implemented support for separate flag addresses for limits, amp flags, and capture flags with new variables Ixx42 (separate amplifier-flag address) and Ixx43 (separate limit-flag address).
3. Implemented support for sub-count position capture from Revision D PMAC2-style Servo ICs for move-until-trigger functions with bits 11 and 12 of Ixx24.
4. Does not permit enabling of any motors if global "phase clock error" bit (X:$000006 bit 3) is set. Enabling command is rejected in this case with ERR018.
5. Fixed lead-out move problem of 2D cutter compensation.
6. Fixed operation of |I|T protection.
7. Improved operation of Extended Servo Algorithm when saturated.
8. Fixed execution of phasing read so will work correctly even when in an overtravel limit.
9. Fixed listing of **`DISPLAY {variable}`** statement when 0 fractional digits specified.
10. Fixed operation of on-line coordinate-system **`Z`** command.
11. Implemented new variable I37 that can specify additional wait states above the default when accessing memory and/or I/O.