Introduction

This document provides information on tuning the IRTF servos, and documents the custom configuration of the PMAC controller. The PMAC controller is a PCI board in the TCS3 computer that drives the HA and DEC servo motors.

Section 1, is primary written by TCS3 Servo Engineer:

1. Tuning the IRTF Servos

Section 2 thur 5, are relevant to the TCS3 programmer as it details software setup and interactive with the PMAC. These sections are:

2. What is the PMAC?
3. Memory Map
4. Using Analog Tachometers.
5. Dual DAC on Each Axis
6. Initialize the absolute position (apos)

To understand the IRTF servo system, you must under the PMAC's role, it configuration, and parameter used to tune the system.

1. Tuning the IRTF TCS3 Servo System

1.1. Type of motion

For Slewing/MP, uses J=STEP to perform a point-to-point move.

For Tracking, adjust velocity using 'Ixx22=vel J+' to keep pmac position close to tcs3 vtcs position.

1.2. Tuning Goals

For tracking we wanted to keep the motor following error under 0.2 arcsecond band 95% of the time, And under 0.1 , 70% of the time, Or
Following error peak-to-peak values under 0.20 arcsec.
Following error standard dev. values under 0.03 arcsec.

We also wish to keep the velocity stable: not varying by more that 1.5 arcsecond/sec.
Velocity peak-to-peak values under 1.50 arcsec/sec.
Velocity standard dev. values under 0.30 arcsec.

During tracking, we used 30 arcseond offset as the standard value to tune to. Our setting time goal is 2 seconds.

Based on TCS1 servo performance was measured in hist/0710/11oct_tcs1_data/.
For comparison, here is some TCS1 performance values:

```
        FollowingErr    Velocity
  Date   P-P    STD     P-P    STD
  ----   ----   ----    ----   ----
2006/01  0.14   0.03    1.30   0.29   // tcs1 servo, but not tracking.
2007/10  0.38   0.05    3.50   0.46   // tcs1 observing at night.
2008/02  0.43   0.07    3.44   0.61   // tcs1 observing during daytime observing 26Feb.
2008/03  0.37   0.07    3.21   0.58   // 070306, after fixing TCS1 broken tac signal
```

The TCS1 offset deadtime is 2.0 seconds.

1.3. HA tracking (track & MV mode)

Auto PID values are 130000 5000 40. Auto Vel_ff is 25000.

P (Ix30):

P can be as low as 75000 up to 200000. 130000 give stiffer responds. Beyond a 140,000 the tracking following error stats starts to increase.

D (x31):

High D is needed to dampen offsets.
Lower D give better offset performance.
Higher D give better tracking performance.
A value of 40 balances tracking and offset performance.

I (Ix33):

Lots of I is good to settle the motor after an offset. Look like 5000 is needed since tracking East need lots of I to setting.

The TCS set Ix34=1 (turning off the integrator) when rtcs_ferr is > 0.2 arcsec; This effectly turns off I during an offset. Without this the I-term would need to be lower: > 2500.

Ix63, Motor XX integration Limit: Since high I values can make a system unstable. For HA, Ix63 is set to 2000, to limit the amount of gain. I-values settles to about 1800 during tacking + offsets.

Vel_FF (Ix32)
Velocity Feed forward (Ix32) is extremely helpful during offsets; it is part of the autoPID table.

Other Terms

Needed to modify the default Jog/Home S-Curve Time (Ix21) of 50 for tracking. 50ms will cause a ~50ms latency when an offset occurs. For immediate response to offset, rtcs sets Ix21=1.

Acceleration controlled using Ixx19. During Tracking it is set to 800 as/s^2.

The TCS3 rtcs will limit the PMAC velocity to 400 as/s.

The rtcs has the ability to set Ix34=1 (disabling Integration) at the time an offset occurs. Ix34 is set to 0 (enabling Integration) when the following error of the PMAC's dpos and vtcs is less that 0.2 arcseconds. ADDITIONAL COMMENTS NEEDED.

The deadband feature (Ix64, Ix65) was tested, but found not very useful. The motor_following_error cross back and forth zero during offset. Using deadband make it offshoot more.

Ix35 (accel FeedForward) default is 0. Setting higher doesn't seem to help tracking or offsetting much. At values above 7000, it make offset worse (axis goes to fast, and starts overshooting).

1.4. HA Slewing (slew & MP mode)

AutoPID values are 80000 5 50.
During slew we don't care about following error, so minimal I-gain of 5 is used.

Acceleration is controlled using Ix21 (Jog/Home S-curve time)

```
if( vel < 400 as/s)
   ix21 = 2000 (2 seconds)
else
   ix21 = 4000 (4 seconds)
```

At speeds above 400 as/s the pmac is unable to maintain a stable velocity. We use the S-Curve feature to provide a very relaxed acceleration curve for the telescope. Once the slew reaches 1800 as/s, the velocity should varies under 20 as/s, and the following error should be peak-to-peak should be < 10 as.
TCS1 actually used a different board for slewing. They did not close the loop using position (as the pmac does), but just has a simple velocity limiting circuit. Thus it was able to have a higher acceleration rate, and did not suffer from this unstableness at higher speeds.

The PC will limit the PMAC velocity to 1800 as/s.

1.5. Dec tracking

PID values are 100000 10000 35.

P-term.
Reasonable tracking (rates < 1as/s) can be achieved with P=80000. However, during offset the dec axis trails the pmac profile. At the end of the offset, the following error can be many arcseconds away (relying on I to close the error). A higher P, 100000, is needed to keep this following error smaller.

D-term.
D is set to 35.

I-term.
After a N offset, I settles to about +200 (near zenith).
After a S offset, I settles to about -600 (near zenith)
The system work better with NoIOnOffset being ON, otherwise the I-values drifts away from its needed tracking value. Ix33 is set relatively high, I=10000, to allow the Dec to settle quickly after an offset.

Due to this high gain, I263=2500. The Motor Integrator Limit is need to keep the system from going unstable.

The PC will limit the PMAC velocity to 400 as/s.

1.6. Dec Slewing

PID values are 50000 5 10.
Acceleration is controlled using Ix21 (Jog/Home S-curve time)
Ix21 = 1000 (1 seconds).

The PC will limit the PMAC velocity to 1800 as/s.

1.7. Comment s on D values when using the Tachometer as the velocity sensor.

The pmac is setup to use the Tachometer as the velocity loop sensors. The TAC a provide better velocity data, but at a reduced resolution. It was estimated that switching between the encoder and TAC, the D would need to be reduce by 120x. So a value of D=10,000 using the encoder would result in D=83 when using the tachometer.

In fact, during 2007-sept/oct, we (mistakenly) had the pmac configured to used the encoder (I104=$3501). Here is a note from then:
"High D is need to dampen when offseting. Going over 10000 provides better damping, but the motor start to fighting with each other more, and we saw the motors fight each other after a slew. A value of 7000 is good as it balances good tracking with reasonable damping for offsets."
When going to the velocity loop, the P general increased to 1.3x its values, vs the non-velocity loop value.

Also see section 4, Using Analog Tachometers.

1.7. Summary of PMAC tuning Ixx Variables

```
Maximum Jog/Home Accel     Ixx19  default=0.15625 count/m^2; *Set at begining of
track/slew.
Jog/Home Accel time        Ixx20  default=0 (ixx21 is in control). (NOT USED BY TCS3)
Jog/Home S-curve time      Ixx21  default=50 *rtcs commands in realtime.

PID Proportional Gain      Ixx30 * rtcs commands in realtime (if autopid is ON)
PID Integral Gain          Ixx33 * rtcs commands in realtime (if autopid is ON)
PID Integral Mode          Ixx34 * default=1; change to 0 in setup.
                                   0 means On all the time.
PID Derivative Gain        Ixx31 * rtcs commands in realtime (if autopid is ON)

PID Velocity FF Gain       Ixx32 * rtcs commands in realtime (in autopid table)
PID Accel FF Gain          Ixx35

PID Notch Filter Coe N1    Ixx36
PID Notch Filter Coe N2    Ixx37
PID Notch Filter Coe D1    Ixx38
PID Notch Filter Coe D2    Ixx39



Max Jog/Home Acceleration  Ixx19 *  rtcs commands in realtime (track, slew, mv,mp)
Max Jog/Home Accel Time    Ixx20
Max Jog/Home S-Curve Time  Ixx21 *  rtcs commands in realtime (track, slew, mv, mp)
Max Jog Speed              Ixx22 *  rtcs commands in realtime (Track, Slew, MV, MP)

Abort Deceleration Rate    Ixx15

Net Desired Pos Filter Gain Ixx40
Desired Position Limit Band Ixx41

Motor Integration Limit    Ixx63    * Set as part of setup. default=4,194,304.
                                     I163=2000 (HA axis).
                                     I263=2500 (Dec axis).

Deadband Gain Factor       Ixx64    default=0
Deadband Size              Ixx65    default=0

Position Error Limit       Ixx67    default=4,194,304 (=262,144 counts)

Motor Friction Feedforward Ixx68    default=0

Output Command Limit       Ixx69 *  13106 -> we limit the DAC output to 4v..
```

2. What is the PMAC?

The PMAC from Delta Tau Data System will perform the servo PID loop. This is a commercial PCI-based servo controller board. The PMAC configuration is:

Pmac controller with options:

400-603657-TRX - Turbo PMAC PCI Lite
5C0-0TURBO-OPT, Turbo CPU option-5C, default CPU-speed/memory config.
302-603657-OPT - On-board 8Kx16 Dual Ported RAM for PCI or USB.

Tac input A/D board:
3A0-602678-10x - ACC-28B, 2-channel A/D converter board
301-00028B-OPT - OPT-1, Additional on board 2-channels A/D converter

Cables:
30P-0ACC8D-OPT - OPT-P, 40 cm (16 inch) cable with 60-pin IDC connector

Purchased but did not use:
306-0ACC8D-OPT - OPT-6, Quad 3-channel encoder isolate board
3D0-602205-10x - ACC-8D, PMAC(1) 4-channel breakout board
3B2-00028B-OPT - OPT-2B, 12-pin input terminal block

Vendor Home Pages is http://www.deltatau.com/

The tcs3 project purchased 3 PMAC. The board as installed as indicated below:

Host	Serial Number
t1	95
t2	93
t1hilo	25

Output querying PMAC on host t2 on 3/18/2008:

```
IOR > CID
602191
IOR > CPU
DSP56303
IOR > DATE
06/11/2003
IOR > SIZE
26556
IOR > TYPE
TURBO1, X4
IOR > VERSION
1.940
IOR > VID
1 DELTA TAU
```

3. Memory Map - TCS3 use of RAM for the PMAC

3.1. PMAC DSP RAM usage - This section documents the non-standard use of DSP RAM.

We used the memory for Motor 30-32 for scratch area or variable storage. This section documents the locations used:

```
Motor 30 $000f00-$000f7f - general purpose
Motor 31 $000f80-$000fff - motor 1 extra space
Motor 32 $001000-$00107f - motor 2 extra space

Motor 1
-------------------------------------------------------------
          y:$000f80 - plc0's m1 command value. I102 = $0f80
          y:$000f81 \ 2 consecutive Y addresses (y:$f81,$f82)
          y:$000f82 /    are needed to hold the APE value for '$*'
   I3160  y:$000f8f - fake ADC (bias value) for ECT for TAC#1

Motor 2
-------------------------------------------------------------
          y:$001000 - plc0's m2 commanded value. I202 = $1000
          y:$001001 \ 2 consecutive Y addresses (y:$1001,$1002)
          y:$001002 /    are needed to hold the APE value for '$*'
   I3260  y:$00100f - fake ADC (bias value) for ECT for TAC#2
```

3.2. PMAC DPR usage - This section documents used of DPR (Dual Ported Ram).

```
2.2.1 Open Buffer.
The following is reserved for open buffer memory:

             PC_Addr  PC_nbytes  DSP_Addr DSP_nwords
OpenBuffer   0x3450   0x0bb0     $60d14   0x2ec

The table documents the used of open buffer memory:

Desc         PC_Addr  PC_nbytes  DSP_Addr DSP_nwords  Notes
-----------  ------   ---------  -------  ---------  ---------------
plc0         0x3450   10*4=40     $60d14  10  (0xA)
```

4. Using Analog Tachometers

The TCS3 can use the motor shaft's DC Tachometer as a velocity sensor in the PMAC control loop. These notes describe the setup for using the TAC output.

4.1 TAC description.

The TCS has 2 DC tachometers per axis. The T3 safety board electronics takes 2 tach DC output, then combines/scale time so the PMAC receives 1 Analog voltage representing velocity.

The tachs output 12 volt/per radians. Gearing between motor shaft & telescope shaft is 144:1. The safety board electronic divides the voltage input by 5. The 2 tach voltage are sum & divided by 2 before being output to the pmac D/A board. The voltage obtained by the PC can be converted to velocity:

V2AS = 5 / 12 / 144 * R2AS is appox 596.8310658.

15 as/s ~ 0.0251 v
2000 as/s ~ 3.3510 v

The PMAC is fitted with ACC28B, a 4-channel 16-bit analog to digital converter board.

The HA velocity voltage is sent to ADC#1.
The DEC velocity voltage is sent to ADC#3.
(ADC#2 and #4 are unused)

The PMAC firmware provides an Encoder Conversion Table (ECT). The following describes how the ECT is configure to used the HA, DEC voltage input as a velocity signal.

4.2 Definition of ECT. Entries 1-4 are default values.

Entry 1 is provides HA position input using the main shaft encoder.
Entry 2 is provides DEC position input using the main shaft encoder.

```
Entry    Address     Y-Word     Conversion Method
-----------------------------------------------------
  1      Y:$  3501   $078000    1/T extension of location $78000
  2      Y:$  3502   $078004    1/T extension of location $78004
  3      Y:$  3503   $078008    1/T extension of location $78008
  4      Y:$  3504   $07800C    1/T extension of location $7800C
```

4.3 Using "Integrated A/D conversion" method ($5)

Method $5 works by:

1. ADC / 256
2. * 32
3. - bais   store in X:reg

The ADC maps -/+ 10 volts to 0-65536. Therefore the bias term is 32768*32 = 1048576 ($100000)

```
Y:$  3505   $5F8006    Integrated A/D conversion of location $F8006
Y:$  3506   $100000    bias term (-32768*32)
```

The above should work for the PMAC, however, their is a bug in the firmware that prevents values below the bias term to be used. After consulting with PMAC technical support, it was suggested we using the Add/Subtract feature to work around this bug.

4.4 Using "Add/Subtract" method ($E)

Use 'add/subtract' method as a work-around. First we using 2 unused I-variables to hold the bias values. We used I-variable from an unused motor (31,32) so they will be automatically saved. Below is the I-variable setup for the TAC.
For details see c.wilson email:
~tcs3/public_html/tcs3/vendor_info/DeltaTau/ref/040206_cwilson_PMAC_Tach.txt

```
I3160=256*32768 ; y:000f8f hold fake ADC (bias value) for ECT for TAC#1
I3260=256*32768 ; y:00100f hold fake ADC (bias value) for ECT for TAC#2
                ; ECT Entries for TAC#1
I8004=$1F8006   ; Unintegrated unsigned ADC from Y:$F8006 (ADC1)
I8005=$180F8F   ; Unintegrated unsigned ADC from Y:$000F8F(fake ADC = bias value )
I8006=$E90405   ; Integrated results from:  (-I8005) + I8004
                ; ECT Entries for TAC#2
I8007=$1F800E   ; Unintegrated unsigned ADC from Y:$F800E (ADC3)
I8008=$18100F   ; Unintegrated unsigned ADC from Y:$00100F(fake ADC = bias value )
I8009=$E90708   ; Integrated results from:  (-I8008) + I8007
I8010=$0        ; end of table
```

Finally, configure the axes to use the tac data, by setting Ix04 - Velocity Loop Feedback Address:

```
I104=$3507     ; set velocity loop feedback to Tach#1 for motor1 - HA  (I8006 results)
I204=$350A     ; set velocity loop feedback to Tach#2 for motor2 - DEC (I8009 results)
```

4.5 Notes on scaling

The velocity resolution in the PMAC in terms of counts/as is very high 12401 counts/AS. This is extremely high compared to cnt/as of encoder (~20). This is a 620:1 ratio.
See ~/public_html/systems/tcs3/computers/pmac/tach_scaling.xls.

Although the pmac manual suggests we scale Ixx09/Ixx08 some the resolutions are equal, our high ratio make this difficult. Ixx09 is just a pre-scaling factor before the Ixx31 derivative gain. cwilson@deltatau.com suggest we keep Ixx08 at 96, and reduce Ix09
to 1 to increase the efficed resolution on Ixx31. However keeping Ixx08/Ixx09 equal would allow easily switching between single and dual feedback.

4.6 I/M-Variables to help look at values for debugging.

Defaults are:

```
I8000 = $078000   ; 1/T Extension of Encoder 1
I8001 = $078004   ; 1/T Extension of Encoder 2
I8002 = $078008   ; 1/T Extension of Encoder 3
I8003 = $07800c   ; 1/T Extension of Encoder 4
I8004 = $0        ; end of table

M-variables to look at ADC and ETC:

m500->x:$3501,0,24    ; 'result' field for I8000 - 1/T encoder 2
m501->x:$3502,0,24    ; 'result' field for I8001 - 1/T encoder 2
m502->x:$3503,0,24    ; 'result' field for I8002 - 1/T encoder 2
m503->x:$3504,0,24    ; 'result' field for I8003 - 1/T encoder 2
m504->x:$3505,0,24    ; 'result' field for I8004
m505->x:$3506,0,24    ; 'result' field for I8005
m506->x:$3507,0,24    ; 'result' field for I8006  - ADC#1 integrated value
m507->x:$3508,0,24    ; 'result' field for I8007
m508->x:$3509,0,24    ; 'result' field for I8008
m509->x:$3510,0,24    ; 'result' field for I8009  - ADC#3 integrated value

m510->y:$78006,0,24   ; points to ADC value
```

4.7. Switching back & forth:

Switch to using tachometers in the velocity loop:

```
I104=$3507     ; set velocity loop feedback to Tach#1 for motor1 - HA  (I8006 results)
I204=$350A     ; set velocity loop feedback to Tach#2 for motor2 - DEC (I8009 results)
```

To return to defaults Ixx04 values (using Inc encoder for vel loop input):

```
I104=$3501     ; restore default velocity loop feedback value
I204=$3502     ; restore default velocity loop feedback value
```

Notes about D-term values.

When using default, D term is positive and in the range of 6000.

When using tac inputs, note the analog values are scale and inverted so the resolution is 120x less. So 50 would be equivalent to about 6000 (in the default mode).

5. Dual DAC on Each Axis

Notes on implementing the Dual DAC output per TCS Axis on the PMAC

5.1. DAC assignments

Each telescope axis has 2 motor requiring 2 pmac DAC per axis. We will assign the pmac DAC outputs as so:

```
 HA:  DAC1,DAC3    ; default for motor 1 and 3
DEC:  DAC2,DAC4    ; default for motor 2 and 4
```

The general scheme is to have a PLC program run every servo cycle with a simple algorithm:

```
dac0 = PID output from PMAC motor
dac1 = motor 1 for axis
dac2 = motor 2 for axis
base1 = backlash value for dac1.
```

```
base2 = backlash value for dac2.

; HA : dac0 neg for west; pos for east
;      dac1 is west (negative)
;      dac2 is east (positive)
if( dac0 < 0 )
   dac1 = -base1 + dac0;     ; go west
   dac2 =  base2;            ; backlash
else
   dac1 = -base1;            ; backlash
   dac2 =  base2 + dac0;     ; go east

; DEC: dac0 neg for north; pos for south
;      dac1 is north (negative)
;      dac2 is south (positive)
if( dac0 < 0 )
   dac1 = -base1 + dac0;     ; go north
   dac2 =  base2;            ; backlash
else
   dac1 = -base1;            ; backlash
   dac2 =  base2 + dac0;     ; go south
```

Based on some tcs1 engineering data, the following anti backlash torque was determined:

The west motor DAC range is 0-negative, base is about -0.31
The east motor DAC range is 0-positive, base is about 0.35

The north motor DAC range is 0-negative, base is about -0.30
The south motor DAC range is 0-positive, base is about 0.43

For TCS3 we will use a value of 0.30 for All Axis.

We use separate base values (base1&2) to allow different base values for each motor, and to include a bias value to handle any calibration. Each PMAC card needs it own values (for example if you want 0.30 volts on each value). The D/A conversion can vary by 0.05 volt on each DAC.

5.2. PMAC implementation notes:

The dual_dac program will run as PLC0.
Set I05 to 3 to enable foreground & background PLC.
Set I08 to 0 to run PLC0 every servo cycle.
DPR Variable Open buffer will provide the storage space need for the program. See section 3, memory map, for details.

To cal base value: Base_volts * 2^16/20v.
For example,
0.3v : 0.3v * 2^16/20 v = 983
1.0v : 1.0v * 2^16/20 v = 3276.8

PMAC pdac value scaled by << 8, ie: +/10 V = +-32767 * 256.
Plc0 shift the cmd by 8 (cmd/256), so internal it is working with 16-bit data (range -32768 to 32767). When writing the result to the DAC memory locate the results (adc1,adc2) are * 256 to scale back to pmac's format.
The base value can be set by changing the mvariables:

```
M8071=983               ; m1_base1
M8072=983               ; m1_base2
M8081=983               ; m2_base1
M8082=983               ; m2_base2
```

The software set the base values startup, see pm_initialize().
.
The Ixx02 pmac defaults are:

```
ixx02 default  DAC    Use in dual DAC code
----  ------   ---    ----------------------------
I102  $078003  DAC1   Motor1 DAC reassigned to WEST (neg)
I202  $078002  DAC2   Motor2 DAC reassigned to NORTH (neg)
I302  $07800B  DAC3   Motor3 DAC reassigned to EAST (pos)
I402  $07800A  DAC4   Motor4 DAC reassegned to SOUTH (pos)
```

Motor Ixx02 Command Output Address will be modified to reference mx_adc0 in DPR.

```
I102 = $0f80    ; default is DAC1, address $78003
I202 = $1000    ; default is DAC2, address $78002
```

plc0 will
read motor command values, m1 at y:$0f00, m2 at y:$0f01
write adjusted DAC0 values to DAC1&2.
write values to DPR for feedback

5.2. Switching back to single output mode, and other plc notes

4.2.1 Set to Single DAC per axis (factory default)

```
disable plc 0         ; disable plc 0
I102 = $78003
I202 = $78002
```

4.2.1 Set to Dual DAC per axis (TCS3 default).

```
I102 = $0f80    ; memory location for plc0. default is $78003
I202 = $1000    ; memory location for plc0. plc0. default is $78002
enable plc 0    ; enable plc 0
```

6. How the TCS3 initializes PMAC's absolute position (apos)

6.1. How it done

The pmac lack a command to directly initialize the axis's absolute value. PMAC assumes the AbsPosValue is captured in memory and the '$*' command will set the axis' position. Since TCS3 will have the linux host set the apos, we will write the APE to pmac RAM and issue the '$*' command.

6.2. I-variable setup

```
I195=$a00000    ; ape value is a Signed 32-bit word
I110=$f81       ; ape value is located at y:$f81,y:$f82

I295=$a00000    ; ape value is a Signed 32-bit word
I210=$1001      ; ape value is located at y:$1001,y:$1002

Ixx95 (position format) is set:
   bit    23  0x0080 0000 equals  0x1 for signed values.
   bit 16-21  0x002f 0000 equals 0x20 for 32-bits data value.

Ixx10 (position address) holds the address of the data:

Ref: Ixx95 I-variable reference in Turbo PMAC Software Reference.
```

6.3. The setup files (ic/rtcs/m/setup) contain these values:

```
;------------------------------------------------------------------------
; apv related setup. See notes/apos_initialize.txt documentation
;------------------------------------------------------------------------
M3100->Y:$000f81,0,24,U     ; low 24-bits for 32-bit APE position (motor1)
M3101->Y:$000f82,0,24,U     ; high 8-bits for 32-bit APE position (motor1)

M3200->Y:$001001,0,24,U     ; low 24-bits for 32-bit APE position (motor2)
M3201->Y:$001002,0,24,U     ; high 8-bits for 32-bit APE position (motor2)

I195=$a00000    ; ape value is a Signed 32-bit word
I110=$f81       ; ape value is located at y:$f81,y:$f82

I295=$a00000    ; ape value is a Signed 32-bit word
I210=$1001      ; ape value is located at y:$1001,y:$1002
```

6.4. Example (setting motor 1 to -8388609, or 0xff7f ffff):

```
M3100=$7fffff
M3101=$ff
$*
```

6.5. TCS3 function

The pm_apos_set() function in pm.c will set the apos for the tcs3. This should be executing only when the tcs3 is in idle mode.