Project Documentation
Document TN-0174
Draft 1

Synchronizing Motion to ATST TRADS using Delta Tau and External Time Base

A. Ferayorni
Instrumentation Group

October 2011

Advanced Technology Solar Telescope 950 N. Cherry Avenue Tucson, AZ 85719
Phone 520-318-8102 atst@nso.edu http://atst.nso.edu Fax 520-318-8500

REVISION SUMMARY:

1. Date: 21 October 2011
   Revision: Draft 1
   Changes: Original version
2. Date:
   Revision:
   Changes:

Table of Contents

Audience

The intended audience of this document are instrument developers that use the Delta Tau Power PMAC for motion control and require precise synchronization between mechanism motions and images being captured by cameras. In particular, those with use cases that involve triggering moves at periods greater than 1s.

1. BACKGROUND

The Instrument Controller (IC) and Camera System (CS) are independent systems that require precise synchronization for many observing use cases. The TRADS TSync hardware allows independent systems to stay synchronized during an observation. For example, the IC can configure the camera to continuously take a frame set every 3 seconds starting at an absolute time. The CS will use its TSync hardware to ensure execution begins at that absolute time and continues in sync with TRADS time. While the CS is taking data, the IC may need to move mechanisms between frame sets. Thus the IC needs to use its TSync hardware to control precise execution of mechanism moves.

The IC has many options for commanding a motor to move. The SIF MotionController provides a move command that can be issued by the IC sequencer. This approach is the simplest, but is non-deterministic and thus not appropriate for observations requiring precise synchronization with the CS. The SIF AdvancedMotionController provides a real-time move command that sequences several moves based on receipt of an external trigger, such as those generated by the TRADS TSync card installed in the IC computer. This approach works well if the period of the trigger generation is within the capabilities of the TSync card's general-purpose outputs, which is currently 1 second.

For use cases that require triggering at periods greater than 1 second, an external counter/timer card can be used. This hardware would generate an output pulse after counting a specified number of input pulses received from the TSync card. The drawback to this approach is the requirement for another piece of hardware, and thus additional software interfaces, maintenance, and potential point of failure.

One solution to the long period trigger use case is to have the TSync card send a single trigger pulse to the Delta Tau Power PMAC at the start of the observation sequence, and let the Delta Tau sequence moves based on offsets relative to the time of the start pulse. Unfortunately, the internal clock of the Delta Tau is only accurate to 100PPM, resulting in 8.64s of drift over the course of a day, or 360 milliseconds in an hour. For long observations (1-4 hours) this drift would not be acceptable for synchronization with cameras.

An ideal solution to the long period trigger use case is to have the time base of the Delta Tau Power PMAC motion controller continuously synchronized to the time of the TSync hardware. Unfortunately, the TSync hardware requires a PCIe bus installation, which is not available on the Delta Tau. Luckily, the Power PMAC provides **External Time Base** (ETB) functionality that allows it to replace its internal time base with that of an external source signal, such as one generated by the TSync card. Once ETB is in place, sequences of moves and delays can be performed in precise amounts of time in sync with the external time source.

The remainder of this document will explain how to setup a Delta Tau Power PMAC to use the TSync card as an external time base. It will also show how to use the AdvancedMotionController custom program execution capabilities to run a motion program under external time base control, and present the performance results from testing this approach.

1.1 REFERENCES

Synchronizing Power PMAC to External Events
ACC24E3 Hardware Manual
TSync User's Manual
TSync Driver Guide

1.2 REQUIRED HARDWARE

1) TSync PCIe-PTP card installed with premium breakout cable
2) Delta Tau Power PMAC with available digital encoder channel on an axis interface board (i.e. ACC24E3)

2. SETUP

As a starting point, you should read the Delta Tau document "**Synchronizing Power PMAC to External Events**". The approach we will be using is referred to as "External Time-Base Control" or "Electronic Cams" in that document.

2.1 STEP 1) VERIFY AXIS INTERFACE CHANNEL IS NOT BEING USED

It is important to verify that the axis interface channel associated with the encoder channel being used is not connected to a motor. Since we are using the encoder channel for ETB, the associated axis interface channel should remain unused. For example, on an ACC24E3 there are 4 axis interface channels, each with their own encoder input. If we use the fourth encoder input for ETB, we must ensure that no motor is connected to the fourth axis interface. Failure to do this could result in damage to a motor connected to the axis interface of the channel used for ETB.

2.2 STEP 2) CONNECT TSYNC GPO TO PPMAC ENCODER INPUT CHANNEL

You will need to determine which General Purpose Output pin will be used to generate the timing signal that will be wired to the PPMAC encoder input channel. Next, connect a wire from that pin of the TSync GPO break out cable to the PPMAC encoder channel pin for A positive quadrature (or PULSE+). A ground wire should be connected from the corresponding TSync GPO break out cable ground pin to the ground of the PPMAC encoder channel. Refer to the TSync User's Manual for pinout information of the premium breakout cable. Refer to the hardware manual for your Delta Tau axis interface board for pin out information.

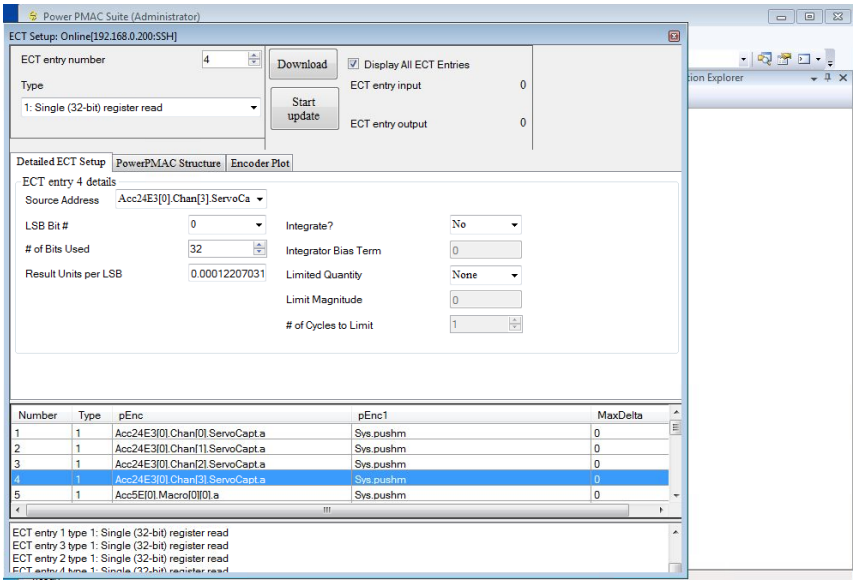
The images below show the connections used for our testing in the solar lab. For our tests GPO 1 of the TSync card was used, which corresponds to pin 2 of the GPO D9 connector of the premium breakout cable. Our Delta Tau PPMAC unit uses an ACC24E3 axis interface board, which has 4 digital encoder inputs. We used the fourth encoder channel input for our tests. The encoder channel input uses a 12 pin terminal block with pin 1 as the A positive quadrature (or PULSE+) input. A ground wire was twisted with the signal wire and connected to pin 7 (GND) of the GPO D9 and pin 8 (GND) of the ACC24E3 channel 4 encoder terminal block.

2.3 STEP 3) CONFIGURE THE DIGITAL ENCODER CHANNEL FEEDBACK TYPE

Using the Delta Tau IDE, go to the **Delta Tau->Configure->Encoder Conversion Table** menu. When the encoder conversion table (ECT) window opens up, select the ECT entry associated with the encoder input you are using.

We will now set the ECT table entry type using the drop down box in the upper left of the ECT window. For external time base we will use the 1/T extension feature of the ECT. This can be done in hardware or software, depending on the type of PMAC ASIC in the axis interface board. For PMAC style 3 ASICS (i.e. ACC24E**3**), the 1/T extension is done in hardware. Therefore we would set the ECT table entry type to **Type 1 Single Register Read**. For PMAC Style 2 ASICS (i.e. ACC24E**2**), the 1/T extension is done in software. Therefore we would set the ECT table entry type to **Type 3 Software 1/T Extension**.

The screenshot below shows how we set the ECT type for our tests in the solar lab. Since we are using an ACC24E3, we have a PMAC style 3 ASIC, and the 1/T extension is done in hardware. Thus ECT table entry 4, which corresponds to the fourth encoder channel, is set to Type 1 Single Register Read.



2.4 STEP 4) SET THE ENCODER CONVERSION SCALE FACTOR

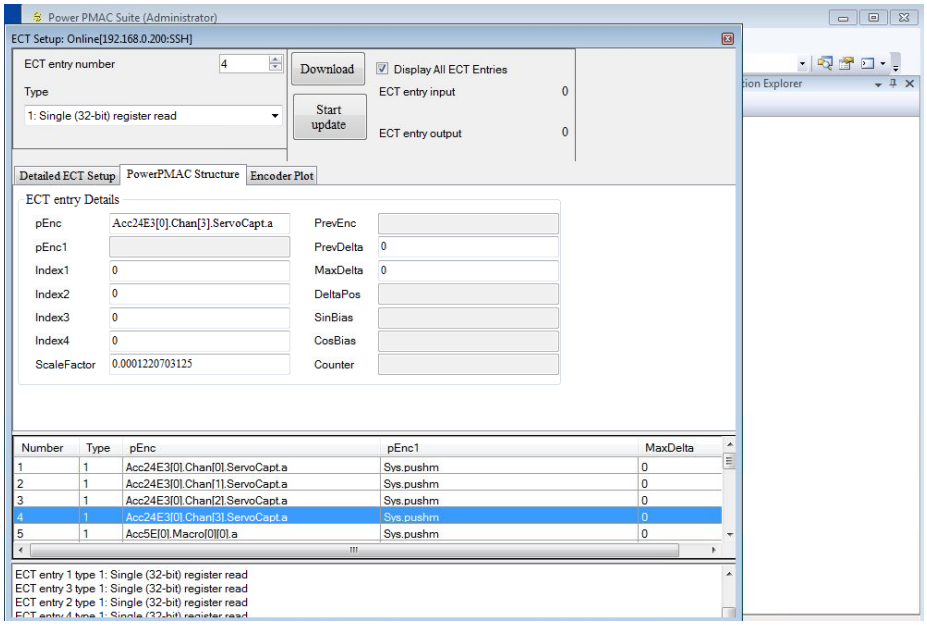
Next, while the ECT entry you are using is still selected, click on the "**PowerPMAC Structure**" tab of the ECT Setup window. Here we will set the ScaleFactor for the ECT entry. The other settings (pEnc, and pEnc1) should already be correct, but you may refer to the screenshot below and your hardware manual for guidance if they are not. The formula for setting the ScaleFactor is as follows:

Scale Factor = 1/ (2^n * RTIF)

Where:
n = number of bits fractional count supported by the encoder board
Ex: ACC24E2 uses 9 bits, ACC24E3 uses 8 bits. Refer to your manual.
RTIF = Real-time input frequency, which is the number of pulses per millisecond being transmitted by the TSync card.
Ex: 32 pulses per millisecond (32kHz) would correspond to a 31.25 microsecond period square wave coming from the TSync. 50% duty cycle should be used.

For our tests, we used an ACC24E3 board (thus n=8) and a 32 kHz signal from the TSync (thus RTIF=32). Therefore…

ScaleFactor = 1/(256 * 32) = 0.0001220703125



2.5 STEP 5) CONFIGURE COORDINATE SYSTEM TO USE EXTERNAL TIME BASE

Now that the encoder channel is receiving the external signal and the ECT entry is converting it correctly you are ready to use it. To do this we must configure the desired coordinate system(s) to use the external time base instead of the internal time base. This is done by setting the **pDesTimeBase** attribute of the coordinate system to point to the address of the ECT entry's **DeltaPos**, which holds the number of counts that occurred during the last servo cycle.

Coord[x].pDesTimeBase = EncTabl[x].DeltaPos.a

NOTE: To turn off external time base, and **change back to internal time base** you would do the following:

Coord[x].pDesTimeBase = Coord[x].DesTimeBase.a

2.6 STEP 6) ADJUST THE TIME BASED SLEW

Each coordinate system has an attribute that limits the amount of change in time-base value allowed each servo cycle. The default value for this attribute is very small, so that drastic changes in time-base do not cause sudden velocity movement. However, when using external time base we need to adjust this parameter to allow for full synchronization to the master signal. Therefore, the value should be changed from the default of 0.00001 to 1.0 as follows:

Coord[x].TimeBasedSlew = 1.0

NOTE: This value should be reverted to the default (0.00001) if the coordinate system is switched back to internal time base.

3. MOTION CONTROL

Now that the connections are in place and the coordinate system(s) configured to use it as an external time base, we are ready to execute motor moves.

3.1 TSYNC CONFIGURATION

Use the TimeBaseController to setup the TSync card. The following is a Jython script that configures a TimeBaseController deployed to a container under the name atst.vbi.test.tbc. The configuration used here is to output a 32 kHz 50% duty cycle square wave signal on GPO 1.

```
import time
from atst.base.util import Misc


# Connect to the time base controller
tbc = App.connect("atst.vbi.test.tbc")
prefix = "atst.vbi.test.tbc."

# Reset the board
reset = Configuration("reset")
reset.insert(Attribute(prefix + "reset", "doreset"))
print "Resetting the TSync board..."
Misc.submitAndWait(tbc, reset)
print

#Config TBC to generate ETB signal
print "Setting GPO mode for pin 1 to direct"
writeConfig = Configuration("writeConfig")
attr = "gpoMode"
newValue = ["1", "DIRECT"]
writeConfig.insert(Attribute(prefix + attr, newValue))
Misc.submitAndWait(tbc, writeConfig)

print "Setting GPO direct value for pin 1 to 1"
writeConfig = Configuration("writeConfig")
attr = "gpoDirectValue"
newValue = ["1", "1"]
writeConfig.insert(Attribute(prefix + attr, newValue))
Misc.submitAndWait(tbc, writeConfig)

print "Enabling GPO for pin 1"
writeConfig = Configuration("writeConfig")
attr = "gpoEnable"
newValue = ["1", "ENABLE"]
writeConfig.insert(Attribute(prefix + attr, newValue))
Misc.submitAndWait(tbc, writeConfig)

print "Disabling GPO for pin 1"
writeConfig = Configuration("writeConfig")
attr = "gpoEnable"
newValue = ["1", "DISABLE"]
writeConfig.insert(Attribute(prefix + attr, newValue))
Misc.submitAndWait(tbc, writeConfig)
```

```
print "Setting GPO mode for pin 1 to square wave"
writeConfig = Configuration("writeConfig")
attr = "gpoMode"
newValue = ["1", "SQUARE_WAVE"]
writeConfig.insert(Attribute(prefix + attr, newValue))
Misc.submitAndWait(tbc, writeConfig)

print "Setting square wave config for pin 1"
writeConfig = Configuration("writeConfig")
attr = "sqWaveConfig"
newValue = ["1", "0", "31250", "15625", "FALLING"]
writeConfig.insert(Attribute(prefix + attr, newValue))
Misc.submitAndWait(tbc, writeConfig)

print "Enabling GPO for pin 1"
writeConfig = Configuration("writeConfig")
attr = "gpoEnable"
newValue = ["1", "ENABLE"]
writeConfig.insert(Attribute(prefix + attr, newValue))
Misc.submitAndWait(tbc, writeConfig)

print "done enabling square wave"
print
```

3.2 CHECK ETB SIGNAL

Use an oscilloscope to verify that the square wave signal coming out of the TSync GPO is as expected.

3.3 CHECK ETB PROCESSING

Once the TSync is generating the correct signal, you can verify the Delta Tau is processing it correctly. The key attribute we want to check is **EncTable[x].DeltaPos**, where x is the ECT table entry number used to setup ETB. The value of this attribute represents the change in pulse count for a servo cycle. If ETB is configured correctly, the value should be close to the value of **Sys.ServoPeriod**. This indicates that the number of milliseconds per servo cycle being calculated from the ETB is close to the number of milliseconds per servo cycle calculated using the Delta Tau's internal clock. The two should be close but **not** exactly the same, because our external clock is *more* accurate than the Delta Tau's internal clock.

3.4 BASIC TEST

Once you have the input signal running, the first simple test you will want to perform is to simply command a linear timed move in the coordinate system using external time base. The following are example command to type into the Delta Tau IDE terminal to perform a move of motor X in coordinate system 3 to position 100 in 3 seconds.

```
&3
cpx linear abs x100 tm3000
```

3.5 ADVANCED TEST

A more advanced test involves running a motion program that continuously executes a move followed by a delay. If you have a digital IO board installed in your Delta Tau rack, an output pulse can be generated coincident with the start and stop of a move. These pulses can then be time stamped by the TSync card so that performance of the system can be checked. Refer to Appendix A for the code required for this test.

NOTE: If you are using a version of TimeBaseController **prior to 1.11**, the time stamp data is returned with a resolution to the millisecond. To record timestamp data with nanosecond resolution it is recommended that you do one of the following:

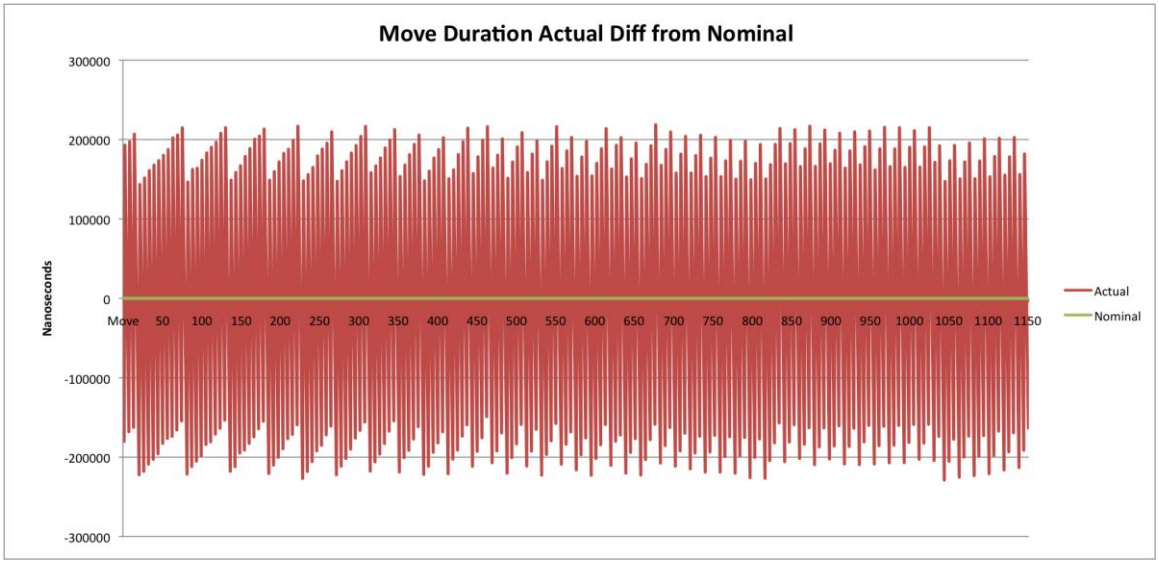
1) Copy the TSync example code (part of the TSync driver installation) for HW_GetTSSingle.c and modify it to read timestamps repeatedly. This will allow you to see the nanosecond resolution of the timestamp, which is not available when reading timestamp back through the TimeBaseController.

2) Modify the following in TimeBaseController…

```
364c364
<                     resultString = new String[] {timeStamp.toString()};
---
>                     resultString = new String[] {timeStamp.toNanoString()};
403c403
<                         resultString[i] = new String(timeStamps[i].toString());
---
>                         resultString[i] = new String(timeStamps[i].toNanoString());
```

4. PERFORMANCE

Tests were run using external time base and data recorded to verify accuracy. In summary, using external time base we saw no significant cumulative drift (i.e. <1ms) over 8 hours. The graph below shows the actual recorded times versus the nominal times. The mean error of the actual times compared to the nominal times was 111 microseconds, with a standard deviation of 64 microseconds.



Full test result data can be found in the excel file: TNXXX_DeltaTau_External_Time_Base.xlsx.

5. APPENDIX A

Jython script for setting up TimeBaseController and executing a custom program on the Delta Tau.

```
import time
from atst.base.util import Misc
from atst.base.util.ra import NullAction

# Create the null action
curAction = NullAction()

# Connect to the time base controller
tbc = App.connect("atst.vbi.test.tbc")
prefix = "atst.vbi.test.tbc."

# Reset the board
reset = Configuration("reset")
reset.insert(Attribute(prefix + "reset", "doreset"))
print "Resetting the TSync board..."
Misc.submitAndWait(tbc, reset)
#time.sleep(2.0)
print

# Test the time
attr = "time"
newValue = "2000-01-01 00:00:01.000"
print "Testing attribute: " + attr
# Read the current value
readTable = AttributeTable()
readTable.insert(Attribute(prefix + attr))
readTable = tbc.get(readTable)
print "Current attribute = " + readTable.getString(prefix + attr)
# Write a new value
writeConfig = Configuration("writeConfig")
writeConfig.insert(Attribute(prefix + attr, newValue))
print "Writing " + newValue + " as new value"
Misc.submitAndWait(tbc, writeConfig)
# Read the value back
time.sleep(2.0)
readTable = AttributeTable()
readTable.insert(Attribute(prefix + attr))
readTable = tbc.get(readTable)
print "New attribute = " + readTable.getString(prefix + attr)
print



#Test etb config for tbc
print "Setting GPO mode for pin 1 to direct"
writeConfig = Configuration("writeConfig")
```

```
attr = "gpoMode"
newValue = ["1", "DIRECT"]
writeConfig.insert(Attribute(prefix + attr, newValue))
Misc.submitAndWait(tbc, writeConfig)
#time.sleep(2.0)
print "Setting GPO direct value for pin 1 to 1"
writeConfig = Configuration("writeConfig")
attr = "gpoDirectValue"
newValue = ["1", "1"]
writeConfig.insert(Attribute(prefix + attr, newValue))
Misc.submitAndWait(tbc, writeConfig)
#time.sleep(2.0)
print "Enabling GPO for pin 1"
writeConfig = Configuration("writeConfig")
attr = "gpoEnable"
newValue = ["1", "ENABLE"]
writeConfig.insert(Attribute(prefix + attr, newValue))
Misc.submitAndWait(tbc, writeConfig)
#time.sleep(2.0)
print "Disabling GPO for pin 1"
writeConfig = Configuration("writeConfig")
attr = "gpoEnable"
newValue = ["1", "DISABLE"]
writeConfig.insert(Attribute(prefix + attr, newValue))
Misc.submitAndWait(tbc, writeConfig)
#time.sleep(2.0)
print "Setting GPO mode for pin 1 to square wave"
writeConfig = Configuration("writeConfig")
attr = "gpoMode"
newValue = ["1", "SQUARE_WAVE"]
writeConfig.insert(Attribute(prefix + attr, newValue))
Misc.submitAndWait(tbc, writeConfig)
#time.sleep(2.0)
print "Setting square wave config for pin 1"
writeConfig = Configuration("writeConfig")
attr = "sqWaveConfig"
newValue = ["1", "0", "31250", "15625", "FALLING"]
writeConfig.insert(Attribute(prefix + attr, newValue))
Misc.submitAndWait(tbc, writeConfig)
#time.sleep(1.0)
print "Enabling GPO for pin 1"
writeConfig = Configuration("writeConfig")
attr = "gpoEnable"
newValue = ["1", "ENABLE"]
writeConfig.insert(Attribute(prefix + attr, newValue))
Misc.submitAndWait(tbc, writeConfig)
#time.sleep(1.0)
print "done enabling square wave"
print

time.sleep(1.0)
```

```
#-------------------------------------------------------
# Setup GPI timestamping
#
# Mask interrupts
print "Masking interrupts"
prefix = "atst.vbi.test.tbc."
writeConfig = Configuration("writeConfig")
attr = "interruptMask"
newValue = ["GPI_EVENT", "0", "ENABLE"]
writeConfig.insert(Attribute(prefix + attr, newValue))
Misc.submitAndWait(tbc, writeConfig)
time.sleep(1.0)
newValue = ["GPI_EVENT", "1", "ENABLE"]
writeConfig.insert(Attribute(prefix + attr, newValue))
Misc.submitAndWait(tbc, writeConfig)
time.sleep(1.0)


# Disable GPI timestamps
print "Disable GPI timestamps"
prefix = "atst.vbi.test.tbc."
writeConfig = Configuration("writeConfig")
attr = "gpiTimeStampEnable"
newValue = ["0", "DISABLE"]
writeConfig.insert(Attribute(prefix + attr, newValue))
Misc.submitAndWait(tbc, writeConfig)
time.sleep(1.0)
newValue = ["1", "DISABLE"]
writeConfig.insert(Attribute(prefix + attr, newValue))
Misc.submitAndWait(tbc, writeConfig)
time.sleep(1.0)


# Clear the host and GPI 0, 1 timestamps
print "Clear the GPI timestamps"
prefix = "atst.vbi.test.tbc."
writeConfig = Configuration("writeConfig")
attr = "timeStampClear"
newValue = ["GPI_0"]
writeConfig.insert(Attribute(prefix + attr, newValue))
Misc.submitAndWait(tbc, writeConfig)
time.sleep(1.0)
newValue = ["GPI_1"]
writeConfig.insert(Attribute(prefix + attr, newValue))
Misc.submitAndWait(tbc, writeConfig)
time.sleep(1.0)


# Setup GPIs
print "Setup the GPI edges"
prefix = "atst.vbi.test.tbc."
```

```
writeConfig = Configuration("writeConfig")
attr = "gpiEdge"
# Falling edge on GPI 0 indicated move is complete
newValue = ["0", "FALLING"]
writeConfig.insert(Attribute(prefix + attr, newValue))
Misc.submitAndWait(tbc, writeConfig)
time.sleep(1.0)
# Rising edge in GPI 1 indicates move starting
newValue = ["1", "RISING"]
writeConfig.insert(Attribute(prefix + attr, newValue))
Misc.submitAndWait(tbc, writeConfig)
time.sleep(1.0)

# Setup interrupt masks for GPIs
print "Setup interrupt masks for GPIs"
prefix = "atst.vbi.test.tbc."
writeConfig = Configuration("writeConfig")
attr = "interruptMask"
newValue = ["GPI_EVENT", "0", "DISABLE"]
writeConfig.insert(Attribute(prefix + attr, newValue))
Misc.submitAndWait(tbc, writeConfig)
time.sleep(1.0)
newValue = ["GPI_EVENT", "1", "DISABLE"]
writeConfig.insert(Attribute(prefix + attr, newValue))
Misc.submitAndWait(tbc, writeConfig)
time.sleep(1.0)

# Enable timestamps
print "Enable timestamps"
prefix = "atst.vbi.test.tbc."
writeConfig = Configuration("writeConfig")
attr = "gpiTimeStampEnable"
newValue = ["0", "ENABLE"]
writeConfig.insert(Attribute(prefix + attr, newValue))
Misc.submitAndWait(tbc, writeConfig)
time.sleep(1.0)
newValue = ["1", "ENABLE"]
writeConfig.insert(Attribute(prefix + attr, newValue))
Misc.submitAndWait(tbc, writeConfig)
time.sleep(1.0)


#-------------------------------------------------------------
#Setup TBC to generate single start pulse

#Test cycle signal config for tbc
print "Setting GPO mode for pin 0 to match time"
writeConfig = Configuration("writeConfig")
attr = "gpoMode"
newValue = ["0", "MATCH_TIME"]
writeConfig.insert(Attribute(prefix + attr, newValue))
Misc.submitAndWait(tbc, writeConfig)
```

```
time.sleep(1.0)

print "Enabling GPO 0 high level match time"
attr = "gpoMatchTimeEnable"
newValue = ["0", "HIGH", "ENABLE"]
writeConfig.insert(Attribute(prefix + attr, newValue))
Misc.submitAndWait(tbc, writeConfig)
time.sleep(1.0)

print "Setting match time config for pin 0 to.."
writeConfig = Configuration("writeConfig")
attr = "matchTimeHi"
newValue = ["0", "2000-01-01 00:01:00.000000000"]
print newValue
writeConfig.insert(Attribute(prefix + attr, newValue))
Misc.submitAndWait(tbc, writeConfig)
time.sleep(5.0)

print "Enabling GPO for pin 0"
writeConfig = Configuration("writeConfig")
attr = "gpoEnable"
newValue = ["0", "ENABLE"]
writeConfig.insert(Attribute(prefix + attr, newValue))
Misc.submitAndWait(tbc, writeConfig)
time.sleep(2.0)

print "Enabling GPO 0 low level match time"
attr = "gpoMatchTimeEnable"
newValue = ["0", "LOW", "ENABLE"]
writeConfig.insert(Attribute(prefix + attr, newValue))
Misc.submitAndWait(tbc, writeConfig)
time.sleep(2.0)

print "Setting match time config for pin 0 to.."
writeConfig = Configuration("writeConfig")
attr = "matchTimeLo"
newValue = ["0", "2000-01-01 00:01:30.000000000"]
print newValue
writeConfig.insert(Attribute(prefix + attr, newValue))
Misc.submitAndWait(tbc, writeConfig)
time.sleep(2.0)
print "Testing attribute: " + attr
# Read the current value
readTable = AttributeTable()
readValue = "0"
readTable.insert(Attribute(prefix + attr, readValue))
readTable = tbc.get(readTable)
print "Current attribute = " + readTable.getString(prefix + attr)
#
time.sleep(2.0)
```

```
#-------------------------------------------------------------
# Connect to the filter controller
print "Connecting to filter..."
filter = App.connect("atst.vbi.test.mc.filter")
prefix = "atst.vbi.test.mc.filter."

writeConfig = Configuration("off")
attr = "mode"
newValue = "off"
writeConfig.insert(Attribute(prefix + attr, newValue))
Misc.submitAndWait(filter, writeConfig)

writeConfig = Configuration("active")
attr = "mode"
newValue = "active"
writeConfig.insert(Attribute(prefix + attr, newValue))
Misc.submitAndWait(filter, writeConfig)

writeConfig = Configuration("custom")
attr = "command"
newValue = "custom"
writeConfig.insert(Attribute(prefix + attr, newValue))
attr = "prog"
newValue = [
"abs",
"linear",
"l1=0",
"l2=7",
"l3 = ldata.coord",
"m(2000 + q2) == 0",
"x(q(l2)) tm(q(l2+1))",
"while(motor[l3].inpos==0) {dwell0}",
"while(m(1000 + q1)==0) {dwell0}",
"m(2000 + q2) == 1",
"delay (q(l2+2))",
"l2 = l2 + 3",
"while(l1 < q5) {",
"  while(l2 < (7 + (3*q6))) {",
"    m(2000 + q2) == 0",
"    x(q(l2)) tm(q(l2+1))",
"    m(2000 + q2) == 1",
"    delay (q(l2+2))",
"    l2 = l2 + 3",
"  }",
"  l1++",
"  l2=7",
"}" ]
writeConfig.insert(Attribute(prefix + attr, newValue))
attr = "progAddr"
newValue = "30"
writeConfig.insert(Attribute(prefix + attr, newValue))
```

```
attr = "progArgs"
newValue = [ "23", "23", "0", "0", "10",
             "2",
             "0","2000","3000",
             "2000","3000","3000" ]
writeConfig.insert(Attribute(prefix + attr, newValue))
raFilter = Misc.submit(curAction, filter, writeConfig)


#-------------------------------------------------------------

# Read in the timestamps
print "Reading in timestamps..."
prefix = "atst.vbi.test.tbc."
msDurCnt=0
oldMsDur="0"
newMsDur="0"
while (1) :

  #print "Reading GPI 1 timestamp count"
  attr = "timeStampCount"
  readValue = "GPI_1"
  readTable = AttributeTable()
  readTable.insert(Attribute(prefix + attr, readValue))
  readTable = tbc.get(readTable)
  #print "Current attribute = " + readTable.getString(prefix + attr)

  if readTable.getString(prefix + attr) != "0" :
    #print "Reading GPI 1 timestamps"
    attr = "timeStamps"
    readValue = "GPI_1"
    readTable = AttributeTable()
    readTable.insert(Attribute(prefix + attr, readValue))
    readTable = tbc.get(readTable)
    newAttr = readTable.getStringArray(prefix + attr)
    print "Start:" + newAttr[0]

  #print "Reading GPI 0 timestamp count"
  attr = "timeStampCount"
  readValue = "GPI_0"
  readTable = AttributeTable()
  readTable.insert(Attribute(prefix + attr, readValue))
  readTable = tbc.get(readTable)
  #print "Current attribute = " + readTable.getString(prefix + attr)

  if readTable.getString(prefix + attr) != "0" :
    #print "Reading GPI 0 timestamps"
    attr = "timeStamps"
    readValue = "GPI_0"
    readTable = AttributeTable()
    readTable.insert(Attribute(prefix + attr, readValue))
    readTable = tbc.get(readTable)
```

```
    newAttr = readTable.getStringArray(prefix + attr)
    print "Stop:" + newAttr[0]

  time.sleep(0.5)



print "done reading timestamps"
print
```

Output when this script is run:

```
Resetting the TSync board...

Testing attribute: time
Current attribute = 2000-01-01 00:00:02.742765510
Writing 2000-01-01 00:00:01.000 as new value
New attribute = 2000-01-01 00:00:03.766109270

Setting GPO mode for pin 1 to direct
Setting GPO direct value for pin 1 to 1
Enabling GPO for pin 1
Disabling GPO for pin 1
Setting GPO mode for pin 1 to square wave
Setting square wave config for pin 1
Enabling GPO for pin 1
done enabling square wave

Masking interrupts
Disable GPI timestamps
Clear the GPI timestamps
Setup the GPI edges
Setup interrupt masks for GPIs
Enable timestamps
Setting GPO mode for pin 0 to match time
Enabling GPO 0 high level match time
Setting match time config for pin 0 to..
['0', '2000-01-01 00:01:00.000000000']
Enabling GPO for pin 0
Enabling GPO 0 low level match time
Setting match time config for pin 0 to..
['0', '2000-01-01 00:01:30.000000000']
Testing attribute: matchTimeLo
Current attribute = 0001-01-01 00:01:30.000000000
Connecting to filter...
Reading in timestamps...
Stop:2000-01-01 00:00:35.097
Start:2000-01-01 00:00:38.090
Stop:2000-01-01 00:00:41.090
Start:2000-01-01 00:00:44.089
```

```
Stop:2000-01-01 00:00:46.089
Start:2000-01-01 00:00:49.090
Stop:2000-01-01 00:00:52.090
Start:2000-01-01 00:00:55.089
Stop:2000-01-01 00:00:57.089
Start:2000-01-01 00:01:00.090
Stop:2000-01-01 00:01:03.090
Start:2000-01-01 00:01:06.089
Stop:2000-01-01 00:01:08.089
Start:2000-01-01 00:01:11.091
Stop:2000-01-01 00:01:14.090
Start:2000-01-01 00:01:17.089
Stop:2000-01-01 00:01:19.089
Start:2000-01-01 00:01:22.091
Stop:2000-01-01 00:01:25.091
Start:2000-01-01 00:01:28.089
Stop:2000-01-01 00:01:30.089
Start:2000-01-01 00:01:33.091
Stop:2000-01-01 00:01:36.091
Start:2000-01-01 00:01:39.089
Stop:2000-01-01 00:01:41.089
Start:2000-01-01 00:01:44.091
Stop:2000-01-01 00:01:47.091
Start:2000-01-01 00:01:50.089
Stop:2000-01-01 00:01:52.089
Start:2000-01-01 00:01:55.091
Stop:2000-01-01 00:01:58.091
Start:2000-01-01 00:02:01.089
Stop:2000-01-01 00:02:03.089
Start:2000-01-01 00:02:06.090
Stop:2000-01-01 00:02:09.091
Start:2000-01-01 00:02:12.089
Stop:2000-01-01 00:02:14.089
Start:2000-01-01 00:02:17.090
Stop:2000-01-01 00:02:20.090
```