

Time Base Synchronization Module

Command reference and application guide

Time Base Synchronization Module:
Command reference and application guide

Version 1.00

Microstar Laboratories, Data Acquisition Processor, DAP, xDAP, DAPstudio, and DAPview are trademarks of Microstar Laboratories, Inc. *Windows* is a trademark of the Microsoft Corporation.

Copyright © 2013, Microstar Laboratories, Inc.

All rights reserved. No part of this manual may be copied, reproduced, or translated to another language without prior written consent of Microstar Laboratories, Inc.

Microstar Laboratories, Inc.
2265 116 Avenue N.E.
Bellevue, WA 98004
Tel: (425) 453-2345
Fax: (425) 453-3199
<http://www.mstarlabs.com>

Part Number: MSTBTSMOD100

Table of Contents

1. Introduction.....	3
Internal timing and external reference timing.....	4
Sample and fractional-sample time intervals.....	5
2. Installing the Time Base Synchronization module.....	6
3. Observing Reference Time.....	7
Time-base analysis using IRIG-B.....	7
Time-base analysis using GPS.....	7
Time-base analysis using a precision oscillator.....	8
4. Resampling.....	9
5. Application Examples.....	10
Application 1: Alignment to power frequency rates	11
Application 2: Alignment to iDSC sampling rates	13
Application 3: Alignment to digital audio rates	15
Application 4: Aligning measurements to IRIG-B	17
6. Command Reference.....	21
Appendix I. Resampling Technology.....	32
Appendix II. Format of Raw IRIG-B Messages.....	35
Modulation schemes.....	35
Information bit encoding in IRIG signals.....	35
Encoding of bits in IRIG signals.....	36
Details of data fields.....	36

1. Introduction

Data acquisition distributed over time and distance can be difficult. Consider the following scenarios:

- Your task is to collect data and report some simple statistical tests periodically over a long period of time. This is trivial, but only if the data acquisition and logging agree on what *periodically* means. If the rates for sampling and the rates for data collection and logging differ by just a few parts per million, over the course of millions of samples, the disparity in rates accumulates into a serious problem.
- You can avoid rate problems by having the centralized data collection *poll the data acquisition* devices for data. But if you do this, you lose continuity of measurements.
- A process protocol must be implemented for *polled* or *triggered* data collection, with the centralized station issuing requests and then waiting for responses. Remote stations don't know what they are supposed to do until the request arrives, so they tend to spin idle a lot of the time. And whatever they do after a request arrives, it can't be very complicated or the response cannot be returned quickly.
- For physically distributed stations, it is difficult to align data sets and determine what events appear in the records for each station. If they are going at the same rate, why not just count sampling intervals? That only works if the sampling is truly *exactly* at the same rate.
- Activity at all stations can be kept at the same pace by driving every station with the same reference time signal. However, the signals typically available to be distributed over a wide area are either very complicated (e.g. IRIG-B receivers) or very slow (e.g., a distributed digital timing pulse). The rates of these systems (typically 1 pulse to 100 pulses per second) are very often not the timing intervals you want for your sampling hardware operation, so an additional timer is involved somehow, and now you have synchronization problems.

Difficulties like these can be solved by applying the Time Base Synchronization Module. The purpose of this module is to enable you to:

1. Use the hardware clock at the natural rate for hardware operation, but synchronize the data (rather than the hardware) to an external reference clock.
2. Operate the data acquisition at any rate you want, specified relative to the external reference.
3. Accept simple voltage or digital pulse signals.
4. Accept broadcast timing signals from sources such as IRIG-B or GPS.

In effect, your signals are recorded as if the sampling were driven directly by the external time base, but at an arbitrary rate otherwise unsupportable by that time base.

Internal timing and external reference timing

An xDAP system provides you with many configurable sampling options. To guarantee that the timing is accurate in all of these configurations, they are governed by an on-board, crystal-controlled digital oscillator operating at a single fixed rate. To give the illusion of “configurable sample timing” a very fast timer rate is used, and all configurable hardware intervals are then exact multiples of this hardware interval. Once synchronized, any timing skew between data acquisition activities remains rigidly fixed. We call the user-adjustable internal timing that the xDAP uses the *sampling time*. It is configured by specifying the sampling *SCAN* interval. Thus, all sampling activity is governed by the sampler timing clock, with equal time intervals between samples as determined by the sampler clock and scan configuration.

When you use an *external time base signal*, you cannot depend that it will stay aligned with the internal sampling clock over an extended time. (Even international timing standards based on an atomic clock or on planetary rotation diverge from each other given sufficient time.) The best you can do is observe differences and account for them. Then you can relate the internal and external timing, even if you can't force them to be the same.

A simple model can account for the differences between the internal DAP and external time base timing.

1. The sampling starts at some position relative to the reference time base. A constant offset term describes this location.
2. The time base and the DAP sampling rate both run at known rates. But the time base is considered absolutely accurate, while the actual sampling rate can differ a small amount from its nominal rate. The primary reason for the rate discrepancy is a tolerance on the absolute accuracy of the sampling timer chips, with variations typically in the tens of parts-per-million.

For example, if the sampling rate is nominally 10000 samples per second, an xDAP system might execute 10000.129 sampling cycles per external time base second on the average. This is not a big discrepancy, but after many thousands of seconds pass, eventually the hardware sampling will deliver one more full sample than would be expected according to the external time base. The mapping model accounts for the observed differences.

Once the relationship is established, it is possible to calculate equally-spaced intervals within the 1-second reference timing pulses, then map these positions back to the corresponding positions in the hardware sample stream.

Sample and fractional-sample time intervals

Sample counts are presented to a precision of 1/1000 sample. If the hardware clock were perfect, samples would occur only at the ends of sampling intervals, so *concerns about fractions of sample interval would be irrelevant*. Suppose for example that there are 10000 samples captured per second. Then those 10000 samples would be captured exactly on 100 microsecond time intervals, aligned perfectly to the sampler clock edges.

When time intervals are determined relative to the external time base, these intervals are not restricted to those locations where the sampler hardware performs its sampling. Then it can be relevant to consider *fractions of a sample* to reach a location at an intermediate position between samples captured directly by the sampling clock.

2. Installing the Time Base Synchronization module

It is assumed that you have used the Standard or Professional edition of the DAPtools software, and run its installer program to place a copy of the Time Base Synchronization module on your local disk drive.

To install the module, run the “Data Acquisition Processor” applications from the Windows control panel. Go to the *Modules* tab. Click on *DAPL 3000* in the display window. In the lower left, click the *Add* button. Leave the default options in the checkboxes on the left side. In the lower right, click the *Browse* button, and navigate in the tree dialog to locate your copy of the `tbtsm.dlm` module. When you click the *Open* button, you will be returned to the *Modules* page. Click *OK*. The module will be loaded to your xDAP system. In addition, its configuration will be noted in the Windows registry, and if you shut down Windows or your xDAP unit, the `tbtsm.dlm` module will be reloaded automatically at future restarts.

3. Observing Reference Time

To base data samples on an external clock, you still need to capture samples, and that means that you must configure an input sampling process. One of the things that this sampling process will do, to establish the association between its timing and the time base, will be to sample the reference timing signal and watch for changes in that signal. This processing performs a *timing analysis* to watch the number of samples occurring per reference timing interval. For perfect rate alignment, this number would be an infinitely repeating pattern. Any break in this pattern provides information about disparity between the nominal and actual sampling rate. Not only does this tracking provide accurate estimates, but it produced dynamic corrections if any long-term relative drift in the sampling rate is detected.

The timing source can be one of the following:

- a once per second digital timing pulse isolated from an IRIG-B time reference
- a once per second digital timing pulse captured by a GPS receiver
- an N pulse per second rectangular wave pulse train generated by an arbitrary digital time source

As with all sampling, there is uncertainty in the exact positions of the samples. You can see the arrival of a reference timing pulse shortly after it happens, but you can't see the arrival of that pulse shortly before it happens, so this uncertainty is not balanced. To keep this timing uncertainty low, you want the hardware sampling to run at a significantly faster rate than the reference clock. As long as sampling does not approach absolute limits that overload the processor, there is no harm from going fast.

Time-base analysis using IRIG-B

To perform a timing analysis on an IRIG-B time signal, route the signal from your IRIG-B receiver to an available digital input pin. Or alternatively, if you do not have or do not want to use a digital input pin, digital logic levels are perfectly acceptable as an analog input, and you can use one of the xDAP analog input channels to receive the IRIG-B signal. The signal must be in the “level shift” digital logic form, not an amplitude modulated carrier form. The information passed to an IRIGB processing command includes the following.

- The sample stream captured from the IRIG-B signal
- The nominal sample rate of the hardware sampler

The pulse rate of the IRIG-B signal is guaranteed by timing standards.

The processing strips the IRIG-B signal into two parts.

1. A timing analysis pipe receives information about the absolute start time, if known, and the rate mapping between sampling time units and reference time-base units.
2. Optional raw message bits as extracted from the raw IRIG signal can be routed to other processing one packet at a time, as received.

Time-base analysis using GPS

GPS receivers provide two kinds of signals, an accurate 1 pulse-per-second timing signal, and a serial data stream (typically RS-422) delivering various general-purpose navigation information. Unlike

IRIG-B, the timing and message signals produced by the hardware arrive via separate lines.

- A clean, “rectangular pulse” logic signal indicates the start of each 1-second timing interval.
- A serial line presents supplementary navigation message texts modulated within the original pulse, but delivered via a separate signal pin.

Since the navigational data is not embedded with the output timing data as in the case of IRIG-B, the Time Base and Synchronization Package does not process the encoded serial data. The timing output is treated as a typical serialized digital signal.

To obtain the GPS information, route the GPS timing output signal to an xDAP digital input pin. Or alternatively, if you do not have or do not want to use digital input pins, the digital logic signal can be processed as an analog input.

Time-base analysis using a precision oscillator

While 1-second intervals are very helpful for relating external timing pulses to “true” elapsed time according to the time base, mathematically, the pulse rate of the reference signal can be something else, provided the rate is accurate and known exactly. For example, some IRIG receivers do not provide the original IRIG signal, but instead provide ordinary digital pulse sequences synchronized to the IRIG timing. You do not need to worry about how these schemes work, or how the oscillator determines the timing intervals, as long as the timing rate remains steady and accurate. The analysis must include an extra parameter for the configurable timing rate.

4. Resampling

Timing analysis will establish the correspondence between positions relative to the observed time base and samples captured by the on-board digitizer. *Resampling* can apply that information to deliver sample values aligned with the reference time base, which could occur at *fractional sample positions* given samples captured on the hardware sampling clock.

If the sampling rate happens to be an exact integer multiple of the desired reference-based rate, and is phase-aligned to the reference clock signal, you have the ideal situation. Every desired sample location, as referenced to the time base, then aligns perfectly to one of the samples captured directly by the hardware. For this, resampling can be as simple as discarding any extra sample values you don't need. But in general, slight rate and “phase” discrepancies will mean that there is no such perfect alignment. What happens when the desired positions relative to the time base no longer align exactly to any of the many available samples? This is the problem of *signal reconstruction* or *interpolation*. Fortunately, this is one of the best understood aspects of *digital signal processing*. Signal sample values can be calculated to a high degree of accuracy at any location, at or between previously measured sample locations. While not perfect, the accuracy of this reconstruction is so good that you can't distinguish these errors from the errors that happen anyway when continuous analog measurements are rounded to the nearest 16-bit integer values by digitizer hardware. For more information about the DSP calculations, see Appendix A.

Once resampling capabilities are available, it is no longer necessary for sample locations to be dictated by the hardware. One second intervals can be divided into any desired fractions of a second. Those calculated positions can be mapped to locations in the hardware sample stream, and resampling applied to evaluate each of these locations. The computations are fast. The sampled values that you get in this *resampled stream* will be indistinguishable from samples directly driven by hardware synchronized to the external timing signal. The difference with resampling is that you don't need complicated and temperamental hardware devices such as clock multipliers, digital counters, and phase-locked loops.

5. Application Examples

This section describes some typical applications using the Time Base Synchronization Module.

Each application described in this section has a corresponding DAPstudio configuration file. You can begin configuring new applications by connecting real signals, running these applications to verify that they work, and then incrementally adapting them. Or if you prefer, you can use the DAPstudio *Save DAPL* option to export the configuration in a text script file form, which can then be modified or used directly within any software environment.

Application 1: Alignment to power frequency rates

The goal is to measure voltage and current in the three phases of a Y-connected 3-phase power transformer. Current and voltage for each phase are measured simultaneously. Also, to facilitate analysis of phase imbalance, the voltage from the Y-connection point of the transformer to a common reference ground is also measured. (Current in the common-ground path must equal the algebraic sum of currents from the three phases, and need not be measured.)

To get an accurate measurement that is less influenced by noise, lots of samples are taken, with the clean data streams to be reduced later. The measurements are supposed to be based on the North American standard frequency, 60 Hz. 1000 samples per cycle would yield a scan rate of 60000 samples per second on each channel.

The xDAP can get close to the desired frequency, but not exactly right. To get 1000 samples per 1/60 second, the SCAN interval must be 16.6666666667 microseconds between scans. With a resolution of 20 nanoseconds in specifying the scan interval, the best available choices for the xDAP are 16.66 or 16.68 microseconds. Thus, the number of samples collected to span 2 power cycles will be off by one sample short or excess, depending on the configuration. This is not random, and it doesn't take long to accumulate what appears to be a significant phase shift between sampled data and the actual waveforms.

Using the Time Base and Synchronization module and a time base reference that produces 60 timing pulses per second, you can obtain exactly the sample rate you want, 1000 samples per waveform cycle, even if the waveform cycle is somewhat longer or shorter than expected.

Solution: Connect the 7 measurement channels and the time base channel to xDAP input signal lines that are sampled simultaneously. Choose a sampling rate that is simple but in the general neighborhood of the rate you want. In this case, sample with a scan interval of 12.8 seconds, corresponding to 78125.0 samples per second on each channel. The timing signal swings between -5V and +5V, which will digitize from -16384 to +16384. Configure the OSCSCAN command to observe the timing rates. Use the TBRESAMP command to obtain the signal values at equal time intervals 60000 times per second aligned to the time base.

DAPL configuration:

```
idefine MslInput
  channels 8
  set IP0 d0      // Seven measurement channels
  set IP1 d2
  set IP2 d4
  set IP3 d6
  set IP4 d8
  set IP5 d10
  set IP6 d12
  set IP7 d14    // Time base channel
  scan 12.80
end

pipes pTimeAnalysis double
constant voltlow word = -16384
constant volthigh word = 16384

// Nominal sample rate 78125 samples per second
constant smpinterval double = 12.800
// Reference timer rate 60 cycles per second
constant refinterval double = 16666.666666666666
// Resample rate 1000 samples per reference timer cycle
constant newinterval double = 16.666666666666666

pdefine aligning
  // Analyze the timing reference signal
  OSCSCAN(IPipe7, voltlow, volthigh, \
    smpinterval, refinterval, pTimeAnalysis)
  // Evaluate signal at the desired sample locations
  TBRESAMP(IPipes(0..6),7, pTimeAnalysis, newinterval, $BinOut)
end
```

See the TBSApp1.dms application file for DAPstudio.

Application 2: Alignment to iDSC sampling rates

iDSC boards provide sampling at certain fixed scan rates. For example, if a desirable sampling rate is 48000 samples per second on each channel, you can't do that. The nearest sampling rates the iDSC supports are 38400 and 51200 samples per second.

Suppose that you have an existing application that depends on the iDSC sample rate of 51200 samples per second. You want to replace this application with an xDAP application that runs at exactly the same sample rate. The 51200 samples per second rate corresponds to 19.53125 microseconds per scan. If you could match that rate exactly, the careful analysis methods in your software would remain valid. However, the xDAP hardware can directly support a scan interval of 19.52 or 19.54 samples per second. These rates differ from the ideal approximately 0.05%. The difference isn't big, but over the course of many data blocks, it soon accumulates into a serious discrepancy.

Using the Time Base and Synchronization module with an arbitrary time base oscillator, you can obtain sampling at arbitrary rates that match iDSC rates but otherwise could not be directly supportable by the hardware SCAN command and an xDAP system.

For this example, there are eight signals that must be sampled simultaneously at the 51200 samples per second rate on each channel, matching the iDSC board. These channels are measured simultaneously. An additional analog signal channel is used to bring in the precision oscillator signal.

Solution: This assumes an xDAP model that samples 8 channels simultaneously. Connect the signal measurement channels to pins D0, D2, . . . , D14. The time base signal connects to input pin D1, and measurements of this channel alternate with the measurements of the other channels. Choose a SCAN interval of 10.0 microseconds to capture all of the measurements, including both the signal and the timing. This rate is arbitrary but faster than the desired interval of 19.53125 microseconds per scan. The timing signal swings between almost 0V and almost +5V, represented as 0 to 16000 converter counts. The precision time base oscillator provides 1000 timing pulses per second. The 10 microsecond scan interval is plenty fast to observe the 1000 microsecond timing pulses without difficulty, providing a high-resolution signal to the OSCTIME command. Use the results of the timing analysis to locate the signal values at equally spaced time intervals, 51200 times per second.

DAPL configuration:

```
idefine MslInput
  channels 10
  set IP0 d0      // Eight iDSC-like signal channels
  set IP1 d2
  set IP2 d4
  set IP3 d6
  set IP4 d8
  set IP5 d10
  set IP6 d12
  set IP7 d14
  set IP8 d1      // Time base channel
  set IP9 d3      // unused
  scan 10.0
end

pipes pTiming double

// Timing oscillator signal levels
constant Vlow word = 0
constant Vhigh word = 16000

// Nominal sample rate 100000 samples per second
constant smpinterval double = 10.0
// Reference timer period for 1000 Hz, in microseconds
constant refinterval double = 1000.00
// Resample interval to match iDSC: 52000 samples/sec rate
constant newinterval double = 19.53125

pdefine aligning
  // Analyze the timing reference signal
  OSCSCAN(IPipe8, Vlow, Vhigh, smpinterval, refinterval, pTiming)
  // Evaluate signal at the desired sample locations
  TBRESAMP(IPipes(0..7),8, pTiming, newinterval, $BinOut)
end
```

See the TBSApp2.dms application file for DAPstudio.

Application 3: Alignment to digital audio rates

Audio signals are often recorded at 48000 samples per second (or multiples of that rate). The eventual goal is often to reduce that oversampled data by frequency-limiting smoothing, followed by a decimation, to yield a final rate of 44100 samples per second for digital audio recording. This is tricky enough, but if multiple tracks from different sources are to be mixed, the rate and phase must remain very stable and accurate in every signal channel, or different sources will be out of phase with each other when played back, producing strange auditory effects.

One common difficulty is that the common recording rates are not exactly compatible. For example, if the signal is originally sampled at 48000 samples per second, it requires a reduction by factor 0.91875 to achieve a final rate of 44100 samples/second. This is a rather awkward ratio.

Using the Time Base and Synchronization module, you can use the 44.1 kHz, 96000 kHz, or any other sampling frequency for your reference timing. We will show how a stable and accurate *A 440* tuning oscillator can be used to produce the desired sampling rates.

Solution: The time reference is provided by a very accurate and stable 440 Hz tuning reference oscillator. First capture the data at a rate somewhat higher than the required intermediate rate of 88200 samples per second. The OSCSCAN command performs the timing analysis, and the TBRESAMP command reduces the data rate to exactly 88200 samples per second. Apply a digital lowpass filter that cuts off any frequencies above 22 kHz in the sampled data, and then there is no aliasing or loss of information when reducing to the final 44100 samples per second audio rate.

The original samples are captured by the hardware at intervals of 6.00 microseconds. The intermediate 88200 samples per second are at intervals of 11.33786848 microseconds. The accurate 440 Hz timing signal period is 2,272.727272 microseconds.

The reference time signal is sinusoidal; this is not what you want for the OSCSCAN command timing analysis, which expects a *digital* (two-level) signal. However, if the waveform signal is a very accurate high-level voltage signal, receiving it with an additional gain of 20 results in a severely “clipped” data set that looks very much like a digital waveform. This is not ideal, but sufficient as a timing reference signal.

The digital filtering is done using the FIRFILTER command, as provided by the DAPL system, to cut off frequencies above 22 kHz, while simultaneously decimating the rate to the final target rate of 44100 samples per second.

DAPL configuration:

```
idefine MslInput
  channels 3
  set IP0 d0          // First audio channel
  set IP1 d2          // Second audio channel
  set IP2 d4 20       // Time base 440 Hz channel, with gain
  scan 6.0
end

pipes pTimeAnalysis double
pipes pOversamp word
pipes pOversamp1 word, pOversamp2 word
pipes pFiltered1 word, pFiltered2 word

constant voltlow      word    = -30000
constant volthigh     word    = 30000

// Nominal hardware sample rate, 200000 samples per second
constant smpinterval double = 6.000
// Reference timer rate, 440 cycles per second
constant refinterval double = 2272.727272
// Resample rate 88200 samples per second
constant newinterval double = 11.33786848

// DAPstudio filter design for the lowpass decimation 2 filter
// Taps: 37 (35), Scale: 1, Type: Lowpass, Window: Hamming
// Low Cutoff/Slope: 0.46/0.08 Sum: 1.000436 Abs Sum: 1.537970
vector HALFBAND word = (21, 0, -50, 0, 120, 0, -258, 0, 501,
  0, -910, 0, 1631, 0, -3186, 0, 10330, 16384, 10330, 0,
  -3186, 0, 1631, 0, -910, 0, 501, 0, -258, 0, 120, 0, -50,
  0, 21)

pdefine aligning
  // Analyze the timing reference signal
  OSCSCAN(IPipe2, voltlow, volthigh, \
    smpinterval, refinterval, pTimeAnalysis)
  // Evaluate both audio channels at the 88200 sample/sec rate
  TBRESAMP(IPipes(0,1),2, pTimeAnalysis, newinterval, pOversamp)
  // Separate the two audio channels
  SKIP(pOversamp, 0,1,2, pOversamp1)
  SKIP(pOversamp, 1,1,2, pOversamp2)
  // Apply halfband anti-alias filtering and decimate to 44100 rate
  // using the DAPL system FIRLOWPASS command
  FIRFILTER(pOversamp1,HALFBAND,35,1,2,-1,pFiltered1)
  FIRFILTER(pOversamp2,HALFBAND,35,1,2,-1,pFiltered2)
  // Recombine streams for transfer at 44100 samples/sec in each
  MERGE(pFiltered1, pFiltered1, $BinOut);
end
```

See the TBSApp3.dms application file for DAPstudio.

Application 4: Aligning measurements to IRIG-B

This example discusses a more advanced application that uses other features of the DAPL system in addition to the Time Base and Synchronization package.

For this application, the goal is to record underground geological disturbances over a long period of time. If something significant happens, blocks of data are recorded from a large grid of 12 geophones. If *nothing* of any significance happens, an occasional data record is captured to maintain baseline statistics about background noise and verify sensor health. Frequencies of interest range from 0.1 Hz to 100 Hz.

Because of potentially very long periods of continuous operation, slight differences in sampling clock rates could accumulate into major misalignment between data sets collected at multiple stations covering a large geographic area. To avoid this problem, each data collection station aligns its data to the time reference signal from an IRIG-B satellite receiver.

The IRIG-B signal provides 1/10 second timing pulses that are precisely positioned. However, the IRIG timing does not provide timing intervals suitable for directly clocking the data acquisition. The xDAP data acquisition provides the stable, high-rate timing required for high resolution sampling. If left free-running, however, these local timers would gradually drift away from the time reference. The processing of the Time Base Synchronization package provides the means for keeping the data acquisition at these stations synchronized.

Blocks of data are selected from the processed data stream when events occur and sent for logging or other processing.

To detect a low frequency rumble at 1/10 Hz, a data block must span at least 10 seconds. Detecting frequencies to 100 Hz requires a sample rate of at least 200 samples per second. Thus, a typical data block for purposes of a frequency analysis will be 2000 samples. When a significant disturbance occurs, data blocks are recorded for as long as the disturbance persists.

A data block is recorded every 10 minutes when no significant event is observed, to provide information about background noise.

As data rates are reduced from the high rates to lower rates for logging, noise captured at the high sampling resolution can potentially result in aliasing artifacts. Additional processing is required while reducing the data rates, to eliminate exposure to aliasing hazards.

Solution: To improve the time resolution, the initial sample rate is boosted to about 20000 samples per second on each channel – much faster than the final recorded data set requires. The timing signal is analyzed by the IRIGSCAN command, which determines how to track the IRIG-B time base signal. The data stream is then processed by the TBRESAMP command, producing blocks of signal data at a partially-reduced 4000 samples per second rate aligned to the IRIG timing signal.

The intermediate rate is still much faster than what is needed for data recording. The AAMFILT command is applied to avoid aliasing the captured noise as the data rate is reduced to the required 200 samples per second in each block, with the data blocks remaining aligned to the 1-second intervals reported by the IRIG time base receiver.

The RMS command monitors the signal level present in each block, producing one value per block. There are 12 channels with 200 samples per second in each, yielding blocks of 2400 samples. If the signal level reported by the RMS command is **greater than** a given threshold, the LIMIT command generates a triggering event. The TRIGSCALE command realigns each event posted by the LIMIT command, adjusting it to the beginning of the corresponding 2400-sample block of data. The WAIT command waits for events, and when one arrives, it selects the indicated block of 2400 aligned samples, copying them to the event data communication channel. Otherwise, data block are discarded.

There is one RMS value produced for each one-second time interval processed. Since the processing is aligned, counting one RMS value corresponds exactly to 1 elapsed second. Thus, this “event timestamp” identifies the second containing the start of the event data. This 32-bit number is transferred via a second communication channel. There is one “event timestamp” value for each data block retained.

To determine when nothing significant is happening, two conditions must be satisfied. First, the opposite of the previous condition must occur – the signal level reported by the RMS command must be **less than** the threshold that signifies an event. Second, the number of elapsed seconds must be a multiple of 600 so that reports will occur no more than once per 10 minute interval. These two conditions are combined by the TAND command. When both events occur at the same place, the block of data representing background noise is selected and copied to the third communication channel.

DAPL configuration:

```
// This configuration requires that the host control panel
// application configures the following extra communication channels.
//   CpOut2      long data stream for timestamp information
//   CpOut3      word data stream for background noise data

#define ScanInputs
  channels 13
  set IP0  d0    1    // Timing signal
  set IP1  d1    20   // 12 sensor channels, measured with gain
  set IP2  d2    20
  set IP3  d3    20
  set IP4  d4    20
  set IP5  d5    20
  set IP6  d6    20
  set IP7  d7    20
  set IP8  d8    20
  set IP9  d9    20
  set IP10 d10   20
  set IP11 d11   20
  set IP12 d12   20

  // Time interval for roughly 20000 scans/sec ==> 52 microseconds
  scan 52.0
end
```

DAPL configuration continued:

```
// pipes used for intermediate data streams
pipes    pTimeAnalysis  double
pipes    pAligned       word
pipes    pAntiAlias     word

// IRIG - TTL signal levels
constant voltlow       word    = 450
constant volthigh     word    = 14600

// Nominal hardware sample interval for about 20000 samples / second
constant smpinterval  double = 52.00
// Resample intermediate interval for 4000 samples / second
constant newinterval  double = 500.00

// Data rates for anti-alias processing
// Samples per second before reduction
constant newrate      double = 4000.00
// Samples per second after reduction
constant outrate      double = 200.00

// Triggering for events
trigger   tActivity    mode=normal
trigger   tSignal      mode=normal
trigger   tSigBlock    mode=normal

// Triggering for non-events
trigger   tInactivity  mode=normal
trigger   tTenMin      mode=normal
trigger   tNoise       mode=normal
trigger   tNoiseBlock  mode=normal

// Triggering levels
pipe      pRMS         word
constant cNoiseLev    word = 400
constant cMaxInt      word = 32767

pdefine monitoring
// Analyze the timing reference signal channel
IRIGSCAN(IPipe0, voltlow, volthigh, smpinterval, pTimeAnalysis)
// Align sensor signals to time base at intermediate sample rate
TBRESAMP(IPipes(1..12),12, pTimeAnalysis, newinterval, pAligned)
// Perform reduction to final rate with anti-aliasing safety
AAMFILT(pAligned,12, newrate,outrate, pAntiAlias)

// Detect activity and deliver data blocks to event data channel
RMS(pAntiAlias,2400,pRMS)
LIMIT(pRms, INSIDE,cNoiseLev,cMaxInt, tActivity)
TRIGSCALE(tActivity, 1,2400,0, tSigBlock)
WAIT(pAntiAlias, tSigBlock,0,2400, $BinOut)

// Report activity timestamps to the event time channel
TSTAMP(tActivity,CpOut2)
```

DAPL configuration continued:

```
// Detect lack of activity and deliver background noise data
LIMIT(pRMS, OUTSIDE,cNoiseLev,cMaxInt, tInactivity)
TGEN(600,tTenMin)
TAND(tInactivity,tTenMin, tNoise)
TRIGSCALE(tNoise, 0,2400,1 tNoiseBlock)
WAIT(pAntiAlias, tNoiseBlock,0,2400, Cp3Out)

end
```

See the TBSApp4.dms application file for DAPstudio.

6. Command Reference

The commands in this module can be categorized as *timing analysis* or *resampling analysis*.

Timing analysis

These commands locate timing transitions in signals observed by xDAP sampling, building a timing correspondence stream that associates sample numbers with reference time in seconds. For signals that include additional encoded information, optional output streams can preserve this separately.

OSCSCAN – Analyze digital timing oscillator waveform

IRIGSCAN – Analyze IRIG-B time base signal and extract encoded message

Resampling analysis

This command takes samples at the original sampling rate, and a stream of timing analysis data. The two are applied to generate a new stream of samples relative to the time base at a new specified rate.

TBRESAMP – Resample to obtain data synchronized to the time base reference signal

Processing Command

Module **TBTSM :: IRIGSCAN**

Perform a timing analysis and optional message extraction for an IRIG-B input signal .

Syntax

```
IRIGSCAN( IRIGIN, BITMASK, RSAMP, [TCODE,] PTIMING [,PMSG] )  
IRIGSCAN( IRIGIN, LOWLEV, HIGHLEV, RSAMP, [TCODE,] PTIMING [,PMSG] )
```

Parameters

IRIGIN

Pipe with input samples captured from an IRIG timing signal.

WORD PIPE

BITMASK

Mask to select IRIG signal bit when IRIGIN is from a digital input port.

WORD CONSTANT

LOWLEV

Nominal signal-low level when IRIGIN is from an analog input port.

WORD CONSTANT

HIGHLEV

Nominal signal-high level when IRIGIN is from an analog input port.

WORD CONSTANT

RSAMP

Sampling scan interval used to digitize IRIG timing signal, microseconds.

FLOAT CONSTANT | DOUBLE CONSTANT

TCODE

Option to record absolute timecode information parsed from IRIG encoding.

KEYWORD, one of: *ABSOLUTE*, *ELAPSED*

PTIMING

Pipe receiving the output timing analysis information to be used by other tasks.

DOUBLE PIPE

PMSG

Optional pipe to receive raw message bits parsed from IRIG encoded IRIG signal.

LONG PIPE

Description

An **IRIGSCAN** task captures time reference data from an unmodulated (voltage level-shifted) signal from an IRIG-B time-base receiver or generator device. The relationship between the sampling rate of this stream and the signal transitions observed in this stream is used to construct a timing analysis used by other processing tasks.

The samples captured from the IRIG-B signal are read from pipe IRIGIN. The BITMASK parameter, or the LOWLEV and HIGHLEV parameters, determine how to interpret the logic-low and logic-high levels on the input signal. Parameter RSAMP specifies the scan rate used to obtain the raw samples of the IRIG signal, and also initializes internal timing. The optional parameter TCODE can be used to

select whether timestamps are in absolute timecodes or relative to the start of the run. The timing information is written to the `PTIMING` output pipe in a compact form for other tasks to use for data resampling. For advanced software systems already configured for this, the optional `PMSG` output pipe can deliver the raw unprocessed message bits from the IRIG-B signal.

Select the appropriate input specification parameters for the type of input signal you use:

- If the signal is a conventional TTL-level digital signal, directly connected to a pin on a digital port, set the `BITMASK` number so that there is a 1-bit at the position where the input signal is wired to the digital port (a logical AND mask), with 0-bits at all other positions in the mask.
- If the signal is connected to one of the analog input pins, specify the input pipe channel that has the analog voltage samples. Be careful that your signal voltage range is within safe limits of your DAP input pin – if not, use an appropriate attenuation network. For the `LOWLEV` parameter, specify the nominal low voltage level, as it would be digitized given your analog channel configuration. Similarly, for the `HIGHLEV` parameter, specify the nominal high voltage level, as it would be digitized. The processing will classify all signal samples as either logic-low or logic-high, depending on whether the input voltage is closer to the nominal low or high voltage level.

The sampling rate for observing the IRIG signal must meet these conditions:

1. The sampling is done at regular time intervals. That is, it cannot be used with hardware triggered or burst-mode sampling.
2. The sampling scan rate is at least 1000 samples per second in each channel. Sample lower than this and accurate capture of the IRIG signal is uncertain.

The IRIG input signal must be sampled in the same input sampling configuration as all of the other measured data channels. In this way, it can establish which measurements in the data streams correspond to which positions in the IRIG timing pulse. Typically, the IRIG timing signal is observed once per scanning pass through the input sampling configuration channel list. For this normal case, the value of the `RSAMP` parameter will be exactly the same as the interval you specify on the `SCAN` command in your sampling configuration.

If your equipment supports the IRIG 200-04 (2004) standard, the IRIG signal can include modulated information bits for year, month, day, hour, minute, and second of calendar time. If you then specify the `ABSOLUTE` timecode option, the encoded information can be parsed from the input signal and used to determine an absolute timecode value, locating the data set in absolute time relative to the beginning of a standard epoch beginning with January 1, year 1 CE. Otherwise, you must use the `ELAPSED` timecode option (the default), which begins the timing analysis with relative timecode 0 at the beginning of operation when the `START` command takes effect. The default option is `ELAPSED`.

Some host software systems can interpret the raw message bits provided in the IRIG-B signal by the receiver or generator equipment. If you specify the optional `PMSG` pipe parameter, the raw bits will be collected into a 96-bit buffer. The bits are delivered as 3 32-bit integers. There are actually only 74 of the bits, aligned to the least significant bit, that contain information, as defined by the IRIG standards. The remaining high-order bits are filled with 0. The buffer is transferred in little-endian style, low order 32 bits first, then middle 32 bits, then high order 32 bits. If the bits are reassembled by the host, and shifted out from the low order bit one at a time, this will reproduce the same sequence of encoded data bits as received in the IRIG signal.

Examples

```
IRIGSCAN( IPipe9, $0010, 40.0, pTime )
```

The IRIG-B timing signal has TTL voltage levels, and samples from the digital port arrive at bit position 4, masked by the hexadecimal bit pattern \$0010, on a digital port. There are 10 channels, and the last of these input channels, `IPipe9`, receives the raw samples. The nominal timing interval for these samples is 40 microseconds, as specified in the sampling configuration, though the actual timing as determined by the time base might differ slightly. The results of the timing analysis go to the `pTime` pipe, and from there can be routed to other tasks. Absolute timecodes are not required, so the default *ELAPSED* option is accepted, and the optional stream for IRIG message bits is omitted.

```
IRIGSCAN( IPipe0, 983, 14746, 25.0, ABSOLUTE, pTime, Cp3Out )
```

The IRIG-B timing signal is read as one of a group of analog input channels collected simultaneously. The signal logic levels are TTL compatible, and observed to be approximately 0.3 volts low and 4.5 volts high. Measured on the voltage range -10 to +10 volts, these levels digitize to 983 and 14746 respectively. The IRIG signal is digitized the same time that every channel is captured, at 25 microsecond scan intervals, as set by the `SCAN` command. The input channel pipe `IPipe0` is the one that receives the raw samples from the IRIG-B receiver. The results of the timing analysis go to the `pTime` pipe, and from there can be routed to other tasks. Two advanced features are used. The IRIG-B receiver provides encoded date-time information, and the *ABSOLUTE* timecode option is selected so that the absolute calendar time is used when later encoding timestamps. The application uses calendar information from the raw IRIG-B messages for purposes of displays, so the `IRIGSCAN` command sends the raw message bits to the host via the `Cp3Out` communication pipe, which was configured in the Windows control panel to transfer 32-bit (*LONG*) data.

Processing Command

Module **TBTSM :: OSCSCAN**

Perform a timing analysis on a digital, squared-pulse input timing signal.

Syntax

```
OSCSCAN( OSCIN, BITMASK, SAMPTIME, REFTIME, PTIMING )  
OSCSCAN( OSCIN, LOWLEV, HIGHLEV, SAMPTIME, REFTIME, PTIMING )
```

Parameters

OSCIN

Pipe with input samples captured from a digital time base signal.
WORD PIPE

BITMASK

Mask to select IRIG signal bit when OSCIN is from a digital input port.
WORD CONSTANT

LOWLEV

Nominal signal-low level when OSCIN is from an analog input port.
WORD CONSTANT

HIGHLEV

Nominal signal-high level when OSCIN is from an analog input port.
WORD CONSTANT

SAMPTIME

Sampling scan interval used to digitize the time base signal, microseconds.
FLOAT CONSTANT | DOUBLE CONSTANT

REFTIME

Time base clock interval for each high-low cycle, microseconds.
FLOAT CONSTANT | DOUBLE CONSTANT

PTIMING

Pipe receiving the output timing analysis information to be used by other tasks.
DOUBLE PIPE

Description

An **OSCCAN** task captures time reference data from a stable timing oscillator signal. The relationship between the sampling rate of the input data stream and the signal transitions observed in this stream is used to construct a timing analysis used by other processing tasks. The input signal from pipe OSCIN can arrive as analog or digital data. The BITMASK parameter, or the LOWLEV and HIGHLEV parameters, will determine how to interpret the sampled signal data. Parameters SAMPTIME and REFTIME specify nominal timing rates for xDAP sampling and time base clocks, and are used to initialize internal tracking. The timing information is written to the PTIMING output pipe in a compact form for other tasks to use for data resampling and timestamp recording.

Select the appropriate input specification parameters for the type of input signal you use:

- If the signal is a conventional TTL-level digital signal, directly connected to a pin on a digital port, set the BITMASK number so that there is a 1-bit at the position where the input signal is

wired to the digital port (a logical AND mask), with 0-bits at all other positions in the mask.

- If the signal is connected to one of the analog input pins, specify the input pipe channel that has the analog voltage samples. Be careful that your signal voltage range is within safe limits for your xDAP input pin. For the `LOWLEV` parameter, specify the nominal low voltage level, as it would be digitized given your analog channel configuration. Similarly, for the `HIGHLEV` parameter, specify the nominal high voltage level, as it would be digitized. The processing will classify all signal samples as either logic-low or logic-high, depending on whether the input voltage is closer to the nominal low or high voltage level. An exact match is not necessary.

The two timing parameters specify the expected signal rates. Typically, the timing signal is observed once per scanning pass through the input sampling configuration channel list. For this normal case, the value of the `SAMPTIME` parameter will be exactly the same as the interval you specify on the `SCAN` command in your sampling configuration. The value of the `REFTIME` parameter must be the period of the time base reference clock, expressed in units of microseconds:

```
reftime == 1.0e6 / clock frequency
```

This is a little unusual, but it represents the reference clock timing in the same units as the sampling. The largest supported `REFTIME` parameter value is 1 second, which would be specified as 1.0e6 microseconds.

The sampling for observing the time reference signal must meet these conditions:

1. The sampling is done at regular time intervals. That is, it cannot be used with hardware triggered or burst-mode sampling. You can use software triggering on the data streams after resampling.
2. The sampling must be fast enough that there is more than one sample captured in each of the two signal levels, to guarantee that every logic level is reliably observed. For typical 50% duty cycle oscillator signals, a suggested rule-of-thumb is to specify a sampling rate at least 5 times the reference clock rate. Faster sampling scan rates are better, and they also have the benefits of higher time resolution and better tracking accuracy.
3. The timing input signal must be sampled in the same input sampling configuration as all of the other measured data channels. In this way, it can establish which samples in the data streams correspond to which positions in the timing reference.

The `OSCSCAN` command faces a fundamental challenge of digital signal processing. It can only observe a change in the timing signal level at the time of a sample. If it sees that the timing signal has changed, it can only know that change happened some time later than the previous sample instant. As the `OSCSCAN` command starts to track the time base with greater accuracy, it sees fewer and fewer timer edges arriving unexpectedly late or unexpectedly early, hence it gets less and less information about how to improve its tracking. No samples are ever lost, but at first the sampling rate errors can be as large as the worst case timing tolerance of the xDAP. This accuracy typically improves to better than ± 10 parts per million in the first 30 seconds and better than ± 2 PPM in 2 minutes. So, for special measurements where near perfect tracking of the reference time rate is needed during sampling, allow the xDAP system to run 1 to 2 minutes, discarding the data.

Examples

```
OSCSCAN( IPipe9, $0010, 40.0, 1000.0, pTime )
```

The timing signal data arrive from a digital port, via pin 4, which is selected using the hexadecimal bit mask \$0010. There are 10 channels, and the last of these input channels IPipe9, receives the raw samples. The nominal timing interval for scanning all the channels is 40 microseconds, as specified in the sampling configuration, though the actual timing as defined by the time base might differ slightly. The reference timer provides 1000 pulses per second, so the time interval between pulses is 1000 microseconds. The results of the timing analysis go to the pTime pipe, and from there can be routed to other tasks.

```
OSCSCAN( IPipe0, 983, 14746, 25.0, 1.0e6, pTime )
```

The timing signal from input channel IPipe0 is one of the channels read as a group of analog input channels. The signal levels are observed to be approximately 0.3 volts low and 4.5 volts high. Measured on the voltage range -10 to +10 volts. these levels digitize to 983 and 14746 converter units respectively. The timing signal provides one accurately-aligned timing edge per second, so its time interval is 1 million microseconds. The timing signal is digitized every time that the grouped channels are captured, and since there is only one data group, that means the timing channel is captured at nominally 25 microsecond intervals, as set by the SCAN command. The results of the timing analysis go to the pTime pipe, and from there can be routed to other tasks.

Processing Command

Module **TBTSM :: TBRESAMP**

Obtain samples aligned to an external time base for multiple channels.

Syntax

```
TBRESAMP ( INSTREAM, NCHAN, PTIMING, RSINTERVAL, [METHOD,] OUTSTREAM )
```

Parameters

INSTREAM

Pipe carrying samples from one or more data channels.
WORD PIPE, LONG PIPE, FLOAT PIPE, DOUBLE PIPE

NCHAN

Number of data channels in INSTREAM.
WORD CONSTANT

PTIMING

Pipe providing the results of a timing analysis calculated separately.
DOUBLE PIPE

RSINTERVAL

Desired new sample rate in samples per second.
FLOAT CONSTANT | DOUBLE CONSTANT

METHOD

Option to override method of calculation.
KEYWORD, one of: NONE, FAST, ACCURATE

OUTSTREAM

Pipe receiving the resampled data after processing.
WORD PIPE, LONG PIPE, FLOAT PIPE, DOUBLE PIPE

Description

A TBRESAMP task performs a resampling analysis based on timing analysis of a time base signal, allowing samples originally captured by hardware clocking to be transformed into equivalent, equally-spaced samples at an adjusted rate referenced to the time base. This results in samples at a very precisely known rate, at very precisely known times that do not drift relative to a fixed time base standard.

Samples of data from multiple channels, with one sample for each of NCHAN data channels in each group, are provided by pipe INSTREAM. The PTIMING pipe provides the results of a separate timing analysis that establishes a best-fit correspondence between the original input data samples and the time base rate. Using this information, samples are located in the data stream with RSINTERVAL microseconds between samples in each channel, based on the time standard. The value of RSINTERVAL corresponds to the time interval you would specify on a SCAN statement for capturing data based on the hardware clock. Each channel is processed separately, with assumptions about relative time or phase relationships. The resulting sample values are placed into the OUTSTREAM data pipe, in the same groupings as they arrived from the input pipe, but at the new rate.

The data types of the input and output pipes must match.

The data rate originally used to sample the input signals is arbitrary, but to fully capture information present in the input signal, the aligned sampling rate will typically be somewhat lower than the original input sampling rate.

Both the initial sampling rate and the aligned output rate are typically much higher than the timing interval rate of the time base signal. For example, IRIG-B typically provides 10 timing cycles per second, while GPS equipment typically provides 1 timing cycle per second. You might need to specify some other rate when you use another kind of precision digital oscillator as your time base signal.

Nothing forces alignment between the reference timing pulses and the crystal oscillator that drives sampling. As a result, sample locations as captured by the hardware do not in general align perfectly to the time-locations desired for the resampled data. Evaluation of samples at resampling positions is closely related to the DSP problem of accurate signal interpolation. The resampling uses a balanced interpolation, based on an equal time-horizon of older and newer samples surrounding each position of evaluation, and has an accuracy of approximately 15.5 bits.

At start-up, the TBRESAMP task must locate a timing reference pulse for initial alignment. With IRIG-B signals, this special pulse occurs once per second. For other time-base oscillators, any timing pulse can serve as an initial reference pulse. The first output sample will align to that reference pulse. There could be a considerable delay, in excess of a second, before the first reference timing pulse is located and the first data produced at the new output rate.

At the end of your data collection run, the resampling process typically needs some extra raw data beyond the final point of evaluation that you need to retain for your resampled output stream. Collect more raw input samples than you would otherwise, to allow this extra margin.

The optional `METHOD` parameter is one of the following keyword strings, in all capital letters but without any quote characters.

- *FAST* This option is appropriate for most applications, particularly those having some observable random noise, fast processing rates, or signal processing such as filtering. Spectrum analysis results are typically very good. In theory, a small “timing jitter” on roughly the order of 1/1000 sample can exist in locating resample positions, making measurements appear slightly “noisy”. In practice, you will find it difficult to see any effects in your results, as real signal noise masks the artificial noise.
- *ACCURATE* This is the best option for very precise, very clean signals, to obtain the best possible reconstruction of every sample, particularly at very high frequencies. There is almost no timing jitter effect – too small to detect. This option requires roughly twice the computation of the *FAST* method, using a two-stage interpolation. If you have a surplus of CPU capacity, there is no harm from using this method even if you don't otherwise need it.
- *NONE* If your hardware sampling rate is already very much faster than the rate of your final resampled data, the abundant data already provides plenty of resolution, and additional calculations within a tiny fraction of a sample position would be wasted. You can use this option to bypass unnecessary calculations, and select the closest sample as produced directly by the hardware sampler. This is extremely efficient, but almost always produces some visible jitter “noise effects” in signals.

If you omit the `METHOD` parameter, you will get the *FAST* option by default.

Some additional considerations:

1. For best results, the frequencies present in every channel should be band-limited to below 40% of the Nyquist frequency (at least 5 samples captured per cycle). The resampling is somewhat analogous to fitting a smooth curve to the data, and then evaluating along the curve. If the input data set is not smooth, the meaning of the curve fitting operation is unclear. Violating this guideline produces results similar to partial lowpass noise filtering, altering the data but in a way that is sometimes beneficial. If your initial data set is not suitably band-limited, it is likely that you are already subject to aliasing problems in your data sets, even before you apply any TBRESAMP processing.
2. Significantly lowering the sampling rate using `TBRESAMP` command can introduce the same kinds of aliasing effects as any *undersampling* of a signal. The TBRESAMP command has no defenses against aliasing. If you need very large rate reductions, consider applying anti-alias filtering operations such as FIRLOWPASS, or the Anti-Aliasing Multichannel Module for the rate reductions.
3. If subjected to abrupt steps (which would violate the bandwidth condition above), the interpolation process exhibits a transient behavior similar to a frequency-selective digital filter in a neighborhood of the discontinuity. Because the interpolation is balanced forward and backward in time, and applied to data in a buffer, the transient can appear to have a counter-intuitive "before the disturbance arrived" behavior.
4. The timing alignment depends on continuous sampling starting from sample 0. Burst mode processing, in which sampling is physically started and stopped, typically does not work well, carrying stale sample values in filtering buffers from distant previous operations.

Examples

```
CONSTANT Newinterval double = 166.66666666666666
TBRESAMP(IPipe(0..5), 6, pTiming, 6000.0, pAligned)
```

Read voltage and current values for a 3-phase power transformer, from the six input sample pipe channels 0 through 5. The timing analysis results from a separate IRIGSCAN command are routed into this command via pipe `pTiming`. The power system operates at the North American 60 Hz frequency, so to obtain a new sampling rate 6000 samples per second, the time interval between samples at the new rate is 166.66666666666666 microseconds. This aligns the data set to power system rates and can represent harmonic frequencies up to the 50th. Results are delivered to the `pAligned` data pipe.

```
TBRESAMP(IPipe(0..7), 8, pTiming, 50.0, ACCURATE, pStream)
SKIP(pStream, 0, 800, 159200, pSelected)
```

Simulate burst activity aligned to time base intervals. Blocks of representative data are collected from all channels once for each 1-second cycle, to be transferred via a network for centralized logging. 8 sampled input channels are provided by `IPipe(0..7)`. The timing information, from analysis of a precision digital oscillator operating at 400 cycles per second, is routed to the TBRESAMP processing

via pipe `pTiming`. The TBRESAMP processing delivers 20000 samples per per second into each channel, at intervals of 50.0 microseconds, producing 160000 samples per second composite that are sent to the `pStream` data pipe. Exceptional accuracy is required for the calculations, so the `ACCURATE` processing override is specified.

For each channel, the SKIP processing takes 100 samples for every group of 20000 samples in each channel – 800 values out for each 160000 samples it receives. Because of the alignment of rates, the `SKIP` command processing delivers each short burst of data at the beginning of each 1-second time base interval. The selected data are placed into the `pSelected` data pipe.

Appendix I. Resampling Technology

What is resampling?

Resampling is the application of well-established DSP techniques to convert a stream of samples, captured under control of one timing clock, into another stream of samples referred to a different timing signal. You can think of resampling as accurately reconstructing the continuous signal from the available samples, and then selecting a new set of sample positions along this reconstructed signal. Resampling is closely related to the mathematical problem of *numerical interpolation*.

Resampling is widely used in digital music recording, synthesis, and playback. It has been much less commonly used for test and measurement applications, but it is no less valid for these.

When is resampling applicable?

Resampling is appropriate when there is a secondary timing reference, and you want samples aligned to that time reference, rather than the original sampling clock where measurements were captured.

How are resampled values computed?

The resampling calculations are very similar to the calculations that would be done to interpolate between two known points along a straight line segment. For the case of interpolating along a continuously varying signal, many known points (time-sample pairs) are used instead of just two. And the interpolation curve is a higher-order curve, not a straight line.

It is not immediately obvious how to reconstruct a signal with continuously varying curvature at any arbitrary location from discrete samples. However, the *Whittaker-Shannon Interpolation Theorem* gives an explicit formula for doing exactly this. Any continuous signal (subject to bandwidth restrictions) can be reconstructed *perfectly* from its samples.

The practical difficulty in applying the *Whittaker-Shannon Interpolation Theorem* is that a *perfect* reconstruction requires an infinite number of operations using arithmetic with infinite precision. Fortunately, perfect reconstruction is not necessary. There is no such thing as perfect measurements, and attempting an absolutely perfect reconstruction would only reproduce measurement noise in complete detail. Approximations to the *Whittaker-Shannon* formulas with a finite number of terms and finite numerical precision yield excellent results, with the numerical errors masked by the roundoff, nonlinearity, and noise effects that you will experience in real measurement data.

What is a "resampling filter"?

The theoretical *Whittaker-Shannon Interpolation Theorem* formula has the form of a *Finite Impulse Response* (FIR, transversal) digital filter — except for the infinite number of terms involved. A practical filter is obtained by reducing that filter to a finite length by *windowing*, a standard DSP technique. After this, the calculation process is the same as those for an ordinary FIR filtering operation. A slight complication is that a different filter must be constructed for every point to be evaluated.

What is the impact on signal amplitudes?

Just as ordinary frequency-selective filters have frequency-dependent gain properties, interpolation filters have these too, but the intent is to make these side-effects as small as possible.

Low frequencies should see no detectable gain error effects.

The interpolation filters in the Resampling Analysis Modules deliberately sacrifice accuracy at frequencies higher than 40% of the Nyquist frequency of the time-based sampling rate, in favor of best performance at 40% of the Nyquist frequency or below. This restriction does not necessarily compromise an application — the sample rate of the high-density clocked data stream is *oversampled* significantly compared to the time-rate of the encoder edges. This means that the practical bandwidth limitations of the resampling occur at high frequencies beyond those that should be present in the signals you are measuring.

Amplitudes of signals in the low frequency band are preserved within 0.0002 dB — that is, the attenuation effects on a signal of maximum amplitude are less than the bit chatter that you will get anyway from digitizing to 16 bits precision. Frequencies higher than the 40% of Nyquist frequency can still carry useful information, but you must keep in mind that the higher the frequency the greater the attenuation effects, so you cannot depend on the magnitudes to be accurate.

What is the impact on signal phase?

An interpolation filter has absolutely no phase distortion effects. However, practical implementation restrictions will sometimes select resampling locations that only an approximation to the ideal locations, causing small erratic effects on phase that look like a lot like *time jitter*.

Low frequency waveforms can't change very much from one sample to the next, so time displacements that are a tiny fraction of one sample at the original high-resolution have no observable effect. For a wave that is very high frequency and therefore changing very fast, time-quantization produces localized small time shifts with effects very similar to random noise. Unlike true random noise, the effects decrease at lower frequencies and lower amplitudes. For typical real signals, the chatter is masked by natural background noise and is difficult to detect.

What are the “interpolation modes” and what is their effect?

Because time-jitter effects can sometimes be large enough to be observable, there are three available operating modes for selecting a resampling location, given a mathematical determination of the desired location.

- Nearest neighbor mode. There are no numerical computations. If the position of the resampled value is different from the position of an available time-based sample, the closest time-based sample position is selected, and its numerical value is used unmodified. This is called the *Nearest Method* in the *Time Base Synchronization* software. This approach is very efficient, but it can produce an observable “phase jitter” in high-frequency signals.

- Precalculated mode. The resampling location is rounded to one of a finite set of locations where precomputed interpolation filters can be applied. The phase shift is approximately 1/1000 of the phase shift that would result from one sample displacement at the high internal sampling rate. To put this in perspective, if a signal has maximum possible amplitude of 32767 counts, at the maximum 40% of the Nyquist limit, the difference in a sample value due to the phase jitter could be no more than 16 converter counts. At more typical lower amplitudes and lower frequencies, the effects are proportionally smaller. This approach is called the *Fast Method* in the *Time Base Synchronization* software. For signals with normal levels, bandwidth, and background noise, or that are subjected to a frequency spectrum analysis, this is the method of choice, and it is selected by default.
- Phase-interpolated mode. The resampling is calculated at two bounding locations for which precomputed interpolation filters are available, and then the final sample value is obtained by linearly interpolating between these two bounding filters. Using this method, *timing jitter* becomes undetectable. However, this computation requires twice the CPU resources of the precalculated mode. This approach is called the *Accurate Method* in the *Time Base Synchronization* software. For extremely clean signals and absolute minimal measurement error, this is the method of choice. In practice, it rarely adds any meaningful improvement to measurement quality.

Appendix II. Format of Raw IRIG-B Messages

This information is based on the **IRIG 200-04 standard**.

IRIG signals are generated by communications/time base equipment that captures timing information from a time reference source and encodes the information as a digital stream. Though the time packets are infrequent and report the time at a low resolution (seconds), the positioning of timing pulses within the packet are very accurate. For the popular IRIG-B message format, there are 10 designated timing pulses per second, with each of these divided into 10 bit-intervals of 1/100 second; so resolution to ± 0.01 seconds is simple. With sophisticated circuits keyed to the timing pulses, time resolution close to microseconds is possible. For the case of Data Acquisition Processors, time resolution is possible within 1 sample scanning interval.

IRIG bit-intervals consist of high-low pulse doublets, each of which covers one cycle of alternation between high and low logic levels. Pulses begin at an accurately-synchronized rising edge at precise timing intervals. The duration (duty cycle) of each pulse is varied to encode information, consequently, the locations of falling edges vary.

Modulation schemes

IRIG receiver/generator equipment can produce either a modulated or unmodulated (level shifted) signal. The *Time Base Synchronization* module requires the unmodulated logic signal, also described as a “level shifted” signal since 0-to-5 volt logic is unbalanced and always positive.

The modulation carrier is a higher frequency sinusoid with frequency a multiple of the fundamental timing frequency of 100 timing pulses per second: typically 10000 Hz. The modulated signal has the advantage of being balanced so that it can be capacitively coupled. The disadvantage is that additional detection logic is required to extract the encoded values. The encoding is amplitude modulation, with logic-low level represented by an amplitude that is 0.3 of the logic-high signal level. In concept, a DAP could sample at a sufficiently high rate and apply some kind of demodulator to recover the logic-high and logic-low portions of the data cycles directly, but this kind of front-end processing is not provided in the Timestamp and Synchronization Module.

The unbalanced voltage-shift output with logic-low and logic-high levels is compatible with serial communications standards such as RS-232 and RS-484. However, the transfer protocols and the packet encoding of these standards do not apply, so the usual serial decode chips cannot be used. Data Acquisition Processors do not require DC voltage balance. The unmodulated signals can be provided either via a digital signal pin at TTL logic levels, or via an analog signal channel.

Information bit encoding in IRIG signals

IRIG signals include timing information and encoded data information.

Since the duty cycle of the signal in each bit-interval is allowed to vary, this can be used to encode information. The pulses in IRIG signals have three forms.

- Synchronizing pulse. No data carried. This pulse is high for 80% of the pulse interval, then low 20% of the pulse interval.

- Data 1 pulse. This pulse is high 50% of the pulse interval, then low 50% of the pulse interval.
- Data 0 pulse. This pulse is high 20% of the pulse interval, then low 80% of the pulse interval.

Encoding of bits in IRIG signals

Sequences of data pulses between the synchronizing pulses are used to encode data (or are padding zero bits). The synchronizing pulses are the last in each group of 10, leaving 9 bits available to carry data in all of the packets except the first, which has an extra synchronizing pulse to indicate alignment with 1-second intervals, thus only 8 bits for carrying data. By extracting the sequence of bits, the encoded information can be parsed from the IRIG signal.

Padding bits are used in various places, in accordance with coding conventions for the fields. For example, numbers are typically represented in a BCD (binary coded decimal digit) form with two digits represented between synchronization pulses, leaving an extra padding 0 bit between the two digits. However, padding bits are not counted as "packet bits" in the standard, and do not need to be preserved in the captured information representations. How to interpret data bits, and where to skip padding bits, is all prescribed by the IRIG standards. The retained information is 74 bits.

Details of data fields

The following tabulation goes field-by-field. The fields are enumerated according to pulses, with pulse 0 being the one that has the special timing marker for 1-second synchronization. Bits are numbered from 1, irrespective of the pulse group, in the order that they are received, in the manner shown in the standard.

Time fields

Seconds, BCD

Short packet, skip the marker pulse.

P0 Bits 1-4: weights 1, 2, 4, 8 (for first digit). Range 0 to 9 used.

Skip padding bit.

Bits 5-7: weights 10, 20, 40 (the second digit). Range 0 to 59 used.

Minutes, BCD

Bits 8-11: weights 1, 2, 4, 8 (for first digit). Range 0 to 9 used.

P1 Skip padding bit.

Bits 12-14: weights 10, 20, 40 (the second digit). Range 0 to 59 used.

Skip padding bit.

Hours, BCD

Bits 15-18: weights 1, 2, 4, 8 (for first digit). Range 0 to 9 used.

P2 Skip padding bit.

Bits 19-20: weights 10, 20 (the second digit). Range 0 to 23 used.

Skip 2 padding bits.

P3 **Days of year, BCD, first two digits**

Bits 21-24: weights 1, 2, 4, 8 (for first digit). Range 0 to 9 used.

Skip padding bit.

Bits 25-28: weights 10, 20, 40, 80 (the second digit). Range 0 to 90 used.
(No trailing pad bits.)

Days of year, BCD, last digit

P4 Bits 29-30: weights 100, 200 (for last digit). Range 0 to 300 used.
Skip 7 padding bits.

Year, last 2 digits, BCD (previously: 9 control bits)

P5 Bits 1-4: weights 1, 2, 4, 8 (for first digit). Range 0 to 9 used.
Bit 5: Skip padding bit.
Bits 6-9: weights 10, 20, 40, 80 (the second digit). Range 0 to 99 used.

P6 Bits 10-18 reserved as control bits.

P7 Bits 19-27 reserved as control bits.

Seconds of day as running binary, low-order bits

Bit 1: weight 2^0

P8 Bit 2: weight 2^1

...

Bit 9: weight 2^8

(Spans range 0 to 511, no padding bits)

Seconds of day as running binary, high bits

Bit 10: weight 2^9

Bit 11: weight 2^{10}

P9 ...

Bit 17: weight 2^{16}

One padding bit

(Spans range 0 to 130560 in steps of 512, range used 0-85888, 86400 seconds/day)

----- *End of Document* -----