Wave Synchronization Module

Command reference and application guide

Wave Synchronization Module:
Command reference and application guide

Version 1.01

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: MSWSMOD101

Table of Contents

1. Introduction

Suppose you have set up an IEEE 1588 software system for your multi-station network. You have mapped out all of your timing signals relative to your centralized timing authority, so that you can know the time offsets between any station and any other with extreme precision. Now you are ready to take some well-synchronized measurements. Despite unpredictable networking lags, you can still tag all of your measurements with their exact time of capture. But extending this scheme down to every individual load cell, thermocouple, accelerometer, and so forth is a difficult and costly exercise.

Sometimes it makes better sense to collect measurements for many channels synchronized to a local station clock, and leave it to the local collection station to coordinate timing with the rest of the network environment. This leaves more flexibility for deciding how to apply the shared local clock to coordinate the lower-level and less-expensive sensor devices. It also leaves the local station more flexibility for how to independently coordinate the timing. Here are three ways the local station might choose to synchronize its local process timing with the other devices in its network.

- On an industrial scale, passage of time is often determined by counting AC power system cycles, in the manner of a typical "electric wall clock." This is not absolutely accurate, and the time resolution is modest, but the power grid is periodically corrected to keep excursions in frequency and phases bounded over time. All local stations stay locked to the common signal. This kind of solution can be feasible when keeping plant activity together is critical but short term timing drift away from the rest of the world does not matter.
- A centralized, accurate, stable timing oscillator can provide a local time base. Measurements can then be associated with that oscillator to synchronize local activity. If further coordination with external networking is needed, the central station that generates the reference time base must enforce the alignment to the networked time standard.
- A highly stable and accurate reference time base can be derived from a standardized timing receiver such as satellite GPS or an IRIG-B beacon. A precision tracking oscillator then derives and distributes an aligned timing signal to the local stations and devices that need it.

All of these options are subject to the limitation that the timing signal rarely has waveforms or cycle rates usable for direct control of sampling: too slow, digital transitions not clean enough, etc.

The WSM (Wave Synchronization Module) addresses cases in which a reference time signal has the form of a (nominally) fixed-frequency sine wave. The WSM allows xDAP hardware to perform its sampling in the ordinary way, using its highly configurable, very stable, but independent clock. The WSM processing then observes both the reference timing waveform and other measured signals together as data. The signal values can be mathematically associated with any phase locations relative to the timing signal waveform. This makes the data sets independent of the original sampling rate, and dependent only on the reference timing and the accuracy of tracking it. Data streams can then be evaluated at any desired locations, not restricted to the cycle rates provided by the external reference clock signal, or to the hardware sample rates of initial data capture.

2. Installation

To install the Wave Synchronization 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 `wsm.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 `wsm.dlm` module will be reloaded automatically at future restarts.

3. Application Examples

This section describes some applications using the Wave 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: Synchronizing to distribution power frequency

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 sum of currents from the three phases, and it does not need to be measured.)

A sampling interval with 400.0 samples per cycle is selected, producing a 24000 samples per second scan rate on each channel, yielding excellent time resolution, and allowing enough extra degrees of freedom for filters to improve data quality effectively.

Accurate measurements of AC voltages and currents can be obtained from a sampled data set, but only when the analyzed data blocks exactly match the lengths of the power cycles. If the sampling rate is slightly off, there is a phase-dependent bias in the computed estimates, due to the fact that a fraction of a cycle is inadvertently included or excluded from the analysis.

On its own, xDAP hardware can get close to the desired nominal frequency, but not exactly right. To get 400 samples per 1/60 second, the SCAN command interval must be 41.66666666 microseconds. With a resolution of 20 nanoseconds in specifying the scan interval, the nearest available choices for the xDAP hardware are 41.660 or 41.680 microseconds. Depending on the choice, the number of samples collected per cycle will average too high or too low. This is not random, and it doesn't take long to accumulate what appears to be a significant phase shift between sampled data set and the actual waveforms. Even more problematic, under relatively moderate conditions, the power grid frequency can routinely drift away from the ideal 60 Hz by 0.1% or more for short term, seriously compounding the data alignment problems and degrading the measurement quality.

Synchronizing to the prevailing power line frequency, and analyzing exactly cycle-by-cycle, eliminates this bias. Using the Time Base and Synchronization module and using one of the AC voltage phases as the timing reference signal, you can obtain **exactly** the number of samples you want per waveform cycle.

Solution: Connect the 7 measurement channels to xDAP input signal lines that are sampled simultaneously. Pick one of the voltage channels as the *reference phase* providing the timing information. Pick a sampling rate 12.0 microseconds, so that there are plenty of samples for good time resolution. Configure the WAVESCAN command to expect the 12.0 microsecond sampling interval and a 60.0 Hz timing reference frequency. The WAVESCAN command performs its timing analysis and places a stream of timing results into a data transfer pipe. The TBRESAMP command can then use this timing information to determine where the sample locations are on all of the signal channels, equally spaced, at the timing interval specified but aligned to the *reference phase*. Note that if the system frequency drifts, so does your time reference, so there will be 400 equally-spaced samples along each waveform cycle regardless of how the frequency drifts.

DAPL configuration:

```
idefine MslInput
   channels 7      // Seven channels to measure
   set IP0 d0      // First measured channel, "Reference Phase"
   set IP1 d2
   set IP2 d4
   set IP3 d6
   set IP4 d8
   set IP5 d10
   set IP6 d12     // Last measured channel
   scan  12.0
end

pipes  pTimeAnalysis  double

// Reference timer rate in cycles per second
constant  reffreq     double  =  60.0
// Hardware sample interval, as configured, microseconds
constant  smpinterval double  =  12.0
// Final sample interval desired, microseconds (based on reffreq)
constant  newinterval double  =  41.6666666666667

pdefine aligning
  // Analyze the timing reference signal
  WAVESCAN(IP0, smpinterval, reffreq, pTimeAnalysis)
  // Evaluate seven signals at the desired sample locations
  TBRESAMP(IPipes(0..6),7, pTimeAnalysis, newinterval, $BinOut)
end
```

See the `WVSApp1.dms` application file for DAPstudio.

Application 2: Large-scale system measurements with multiple xDAPs

This example uses a common local time-base oscillator as a source for coordinating data capture on multiple xDAP stations – in this case, absolute alignment to an external world time standard does not matter. A large airframe structure is monitored. There are 10 xDAP stations, each monitoring 30 channels of vibration data from strain sensors. The analysis must preserve frequencies from 0 to 2500 Hz accurately, both in magnitude and phase on each channel, delivering data at 20000 samples/second aligned to the reference timing signal, for every signal, at every station. Alignment of sampling is critical to avoid artificial phase distortions. The local time-base oscillator generates a sinusoidal timing signal at 1000 Hz, high enough to give good timing resolution, but low enough to avoid signal propagation distortions along the timing cable.

The sampling rate is set at 100000 samples per second on each channel. Simple bypass terminations on each cable line eliminate frequencies at 100 kHz and above, so there is no exposure to aliasing and expensive anti-aliasing filters are unnecessary. The 100 samples per wave cycle for observing the sinusoidal timing oscillator yields excellent time resolution and phase alignment.

The initial sampling rate is higher than the final desired 20000 samples per second. To avoid introducing new aliasing effects during the rate reduction, each signal channel (including the reference timing channel) is processed by a transversal (*Finite Impulse Response*) digital filter that preserves all frequencies up to 5000 Hz with extreme accuracy, and attenuates all frequencies above 15000 Hz to effectively zero. The data stream can then be decimated to a 20000 sample per second rate with no aliasing hazards.

The samples at 100000 samples per second are actually captured in four separate banks of 8 channels at 2.5 microsecond intervals. The staggered 2.5 microsecond delays introduce a precise and predictable phase shift proportional to frequency. For the maximum 2500 Hz frequencies, there are 0.00000 radians delay for the first bank, 0.03927 radians delay for the second bank, 0.07854 radians delay for the third bank, and 0.11781 radians delay at the fourth bank. These shifts can be corrected during later analysis, but we will apply an alternate approach that removes these delays at the time of data capture.

Start-up timing causes a complication. There is no way to know when individual xDAP stations are started, and after that, when exactly each individual xDAP unit is actually commanded to run. So a special triggering strategy is applied. Every station is started, and runs continuously, but initially all data are discarded. A separate triggering signal is delivered to each xDAP station in addition to the regular timing signal. This triggering signal is normally near 0V, but is raised to approximately 5V when the central station is ready to begin collecting measurements. This rising edge of this signal is guaranteed to occur only **after** a rising-edge zero crossing on the AC timing wave. Each station begins analyzing data following this trigger event, eliminating ambiguity about which AC cycle is the first.

Solution: Each xDAP station watches 32 signal channels. One signal is the triggering signal and one signal is the timing waveform. The sampling and processing are started, monitoring data but discarding everything until the timing signal pulse arrives. At this point, timing on each xDAP station aligns to the next timing waveform zero crossing and matches it in rate and phase.

Raw data samples are first filtered to eliminate all of the undesirable high frequencies. Then filtered data streams are collected and adjusted by MTSFILT to remove the 2.5 microsecond delays between the 8-channel banks. The TBRESAMP command then aligns the data set to the reference clock,

decimating the data streams to the target 20000 samples per second rate.

DAPL configuration:

```
idefine MslInput
   channels 32
   set IP0 d0     // The timing sinusoid channel
   set IP1 d1     // The triggering pulse channel
   set IP2 d2     // First of 30 vibration signal channels
   set IP3 d3
   . . .
   set IP30 d30
   set IP31 d31   // The last of the 30 vibration signal channels
   scan  10.0     // 10 microseconds cover 4 channel groups of 8
end

// Digital lowpass FIR filter design from DAPstudio
// Taps: 41, Scale: 4, Window: Kaiser with alpha=4.8
// Low Cutoff: 0.12  Transition: 0.12
vector vFilter word = (-12, -28, -40, -19, 64, 221, 413, 531,
  415, -75, -957, -2029, -2842, -2778, -1248, 2059, 6963,
  12732, 18215, 22158, 23593, 22158, 18215, 12732, 6963, 2059,
  -1248, -2778, -2842, -2029, -957, -75, 415, 531, 413, 221,
  64, -19, -40, -28, -12)

trigger  tBegin  mode=normal

pipes  pTiming   double    // Results of timing analysis
pipes  pRaw      word      // Intermediate unfiltered data

pipes  pRef      word      // Separated signal channels
pipes  pTrig     word
pipes  pR1       word
pipes  pR2       word
   . . .
pipes  pR30      word

pipes  pF1       word      // Intermediate filtered data
pipes  pF2       word
   . . .
pipes  pF30      word      // Last of 30 filtered signals
pipe   pMerged   word      // Reorganized data
pipe   pAligned  word      // Data aligned to reference clock


// Nominal sample interval for 100000 samples per second rate
constant  smpinterval double  =    10.0
// Reference timer rate, 1000 Hz
constant  reffreq double      =  1000.00
// Desired final 20000 sample/second rate, 50 microsecond interval
constant  newinterval double  =    50.00
```

```
// Data processing
pdefine aligning
  // Wait for triggering pulse above 4V to retain raw data
  LIMIT(IPipe1,INSIDE,13107,32767,tBegin,INSIDE,13107,32767)
  WAIT(IPipe(0..31), tBegin, 0, pRaw)

  // Separate the raw signal channels for filtering
  SEPARATE(pRaw,  \
    pRef, pTrig, pR1,  pR2,  pR3,  pR4,  pR5,  pR6,  \
    pR7,  pR8,   pR9,  pR10, pR11, pR12, pR13, pR14, \
    pR15, pR16,  pR17, pR18, pR19, pR20, pR21, pR22, \
    pR23, pR24,  pR25, pR26, pR27, pR28, pR29, pR30)

  // Filter the raw measurement samples, preserving 0-5000 Hz
  FIRFILTER(pRaw1,vFilter,41,4,1,-1,pF1)
  FIRFILTER(pRaw2,vFilter,41,4,1,-1,pF2)
  . . .
  FIRFILTER(pRaw30,vFilter,41,4,1,-1,pF30)

  // Reorganize channels in groups as sampled simultaneously
  MERGE( \
    pRef,  pF1,  pF3,  pF5,  pF7,  pF9,  pF11, pF13, \
    pTrig, pF2,  pF4,  pF6,  pF8,  pF10, pF12, pF14, \
    pF15,  pF17, pF19, pF21, pF23, pF25, pF27, pF29, \
    pF16,  pF18, pF20, pF22, pF24, pF26, pF28, pF30, \
      pMerged )

  // Cancel the 2.5 microsecond delays between channel banks
  MTSFILT(pMerged,32,8,  1,  pAligned)

  // Analyze the timing reference waveform
  WAVESCAN(pRef, smpinterval, reffreq, pTiming)

  // Decimate and time-align the data channels to final rate,
  // and send results to host system
  TBRESAMP(pAligned,32, pTiming, newinteval, $BinOut)

end
```

See the `WVSApp2.dms` application file for DAPstudio.

Application 3: Polling for streamed data

This application uses an xDAP system to collect streaming data. The central station wants to collect data for a 500-point FFT power spectrum analysis every second. The central station does not have resources for performing the DSP transform calculations, and expects these to be already completed in the delivered data. A data block of spectrum results is expected immediately after sending out polling message once per second.

The central station and the xDAP, operating on independent clocks, might have a slightly different idea about what the 1.0 polling interval is, a little longer or a little shorter than the nominal 1.0 seconds. Suppose that the xDAP happens to be operating a few parts per million faster than the nominal once-per-second rate of the central station. Over the course of time, the xDAP will collect excess samples beyond those needed, gradually accumulating a backlog of old history in buffer memory. As the collected measurement values backlog in buffer memory, the delivered results start to lag behind actual time in a very mysterious way.

Or take the opposite case. The xDAP happens to be operating a few parts per million slower than the nominal once-per-second rate of the central station. The xDAP can't send data that is doesn't have, leading eventually to a missed response to a polling request.

The problems arise because the system polling and sampling operate with separate clocks – the definition of *one second* is slightly different at the two locations. If the rates of issuing polling requests and producing responses can be made to stay aligned to the same reference rate over time, rate inconsistency problems do not occur. Since extremely high precision and resolution are not necessary for this, the shared timing can be based on the AC waveform for the common power distribution supplying both systems.

The sampling rate necessary to track the power waveforms accurately is much higher than the data rate needed for the FFT analysis. This is not a problem – DAPL system processing can decimate the data to obtain the desired rate for the analysis. This is done in a way that avoids aliasing and preserves the validity and accuracy in the reported spectrum terms. Then, an FFT analysis is performed on 1000-term blocks. The power spectrum results are redundant in the second half of a 1000-term transform, so the extraneous terms are discarded to yield the desired 500-term power spectrum spanning from 0 to 500 Hz.

Solution. Configure the xDAP system to capture data at approximately 6000 samples per second, based on the xDAP local clock. The AC supply waveform and the measured signal are captured simultaneously. At this rate, there are about 100 samples per AC waveform cycle, yielding plenty of data for an accurate timing analysis. Apply a timing analysis to the AC waveform cycle. Using that timing analysis, resample the waveform signal so that the 100 samples per waveform exactly align to the waveforms. Apply digital filtering to reduce the data rate from 6000 samples per second to 2000 samples per second without any aliasing damage. This filtering preserves the frequency band from 0 to 500 Hz perfectly. Perform a 2000 point FFT power spectrum analysis for each second of data. Discard all attenuated and redundant frequency bands, and report the required 500 accurate power spectrum terms to the host as soon as the work is completed, so the information is ready for the host to deliver to the central data collection system when the next polling message arrives.

DAPL configuration:

```
idefine MslInput
   channels 2
   set IP0 d0     // The signal to be measured and analyzed
   set IP1 d1     // The observations of the 60 Hz waveform
   scan  166      // Roughly 6000 samples per second, noncritical
end

pipes  pTiming   double    // Results of timing analysis
pipes  pAligned  word      // Intermediate full-rate data
pipes  pReduced  word      // Reduced rate data
pipes  pPowSpect float     // Separated signal channels
pipes  p500Hz    float     // Desired power spectrum data

// Specified sampling interval, microseconds
constant  smpinterval double  =    166.00
// Reference timer rate, 60 Hz
constant  reffreq double      =     60.00
// Desired microsecond sample time interval, 2000 samples/second
constant  newinterval double  =    500.00

// Data processing
pdefine powerpoll
  // Analyze the 60 Hz reference waveform
  WAVESCAN(IPipe1, smpinterval, reffreq, pTiming)

  // Align signal measurement stream to reference timing
  TBRESAMP(IPipe0,1, pTiming, newinterval, pAligned)

  // Use a decimating lowpass filter to safely reduce the
  // data rate to 2000 aligned samples/sec without aliasing
  FIRLOWPASS(pAligned,3,pReduced)

  // Perform a 2000 sample FFT analysis, obtaining a 1000 term
  // power spectrum block.
  MIXRFFT(2000,HAMMING,pReduced,HALF,POWER,pPowSpect)

  // Retain only the first 500 (accurate) terms from each block
  SKIP(pPowSpect, 0,500,500, p500Hz);
  MERGE(p500Hz,$BinOut)
end
```

See the `WVSApp3.dms` application file for DAPstudio.

Application 4: Monitoring power frequency drift

This application reverses the normal assumptions. Usually, when you have a time base signal, it is a trusted resource, calibrated to a high degree of accuracy according to an even better time reference. But when you are using a power system waveform for timing, the frequency can drift, and for the short term it is much less steady than the sampling clock hardware. Short term frequency drift can be observed in the number of samples at the original sampling rate needed to span each observed reference wave cycle.

In this application, the xDAP internal sampling is deemed more accurate and stable than the power system frequency observed. Assuming a reversal of roles, the sampling clock is treated as the trusted reference. The timing analysis determines how much the observed sine wave timing signal deviates from the parameters specified for the WAVESCAN command processing, based on the sampling clock, and reports this information in a separate output stream.

Solution: The time reference signal to be tested is derived from an AC distribution main at the European grid frequency of 50 Hz. The sampling is performed at intervals of 266.0 microseconds per sample. This results in approximately 75 samples per waveform cycle. WAVESCAN command accounts for all of the fractions, so the exact value of the sampling interval doesn't matter. The configuration specifies a second, optional output pipe where the WAVESCAN command can save its analysis results. The normal timing analysis stream is mandatory, but it can be discarded and ignored. The alternate output data consists of three values per observed cycle, and the field of interest is the second term, the current operating frequency relative to the sampling clock. If the frequency indicated in the output data exactly matches the reference signal nominal value as specified by the WAVESCAN task parameter, the power frequency currently matches the nominal 50 Hz exactly. If the reported values are lower or higher, this indicates that the reference frequency seems to be correspondingly lower or higher relative to the sampling which is deemed steady. Output values are reported in `float` data type, with units of Hz.

The analysis represents all of the frequency drift as if it were due to the power system frequency. Actually, a few parts per million of frequency shift could be due to frequency error in the crystal-controlled sampling clock rate. There is no way to determine the cause of this discrepancy from the data samples.

DAPL configuration:

```
idefine MslInput
   channels 1
   set ipipe0   d0    // Waveform to check
   time     266
end

pipes  pTimeAnalysis  double
pipes  pWaveAnalysis  double
pipes  pFreq          double
pipes  pFloatFreq     float

// Reference timer rate in cycles per second
constant  reffreq     double  =  60.0
// Hardware sample interval, as configured, microseconds
constant  smpinterval double  =  266.0

pdefine freqtest
  // Analyze the timing reference signal
  WAVESCAN(IP0, smpinterval, reffreq, pTimeAnalysis, pWaveAnalysis)
  // We don't need the usual timing analysis data
  DISCARD(pTimeAnalysis)
  // Select the frequency field to send to the host in float type
  SKIP(pWaveAnalysis, 1,1,2, pFreq)
  pFloatFreq = pFreq
  MERGE(pFloatFreq, $Binout)
end
```

See the `WVSApp4.dms` application file for DAPstudio.

4. Technology of Timing Signal Tracking

Sampling and external reference signals

An xDAP system provides many configurable sampling options. To guarantee that the timing is accurate in all of these configurations, processing is 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. All configurable hardware timing intervals are then exact multiples of this hardware interval. Once started, timing intervals for data acquisition action remains rigidly fixed. We call the xDAP's user-adjustable sample timing period the *sampling time interval.* It is configured by specifying the sampling `SCAN` interval. Thus, sampling activity is determined by the sampler timing clock, with precise and equal time intervals between samples as determined by the sampler clock and used according to the scan configuration.

When you have any independent *external time base signal,* you cannot depend that it will start aligned with the xDAP internal sampling clock, and you cannot depend that the external rate will match the sampling clock timer exactly over an extended time. (Even international timing standards based on an atomic clock or on planetary rotation diverge from each other given sufficient time.)

For the particular case that the external time base signal is a sinusoidal waveform, there is rich frequency and phase information, but the time base signal itself is unsuitable for direct digital control of sampling processes. xDAP processing can observe the timing, and account for what happens sample-by-sample in observations of the timing reference signal. Then locations in other sampled data channels captured at the same time can be associated with locations along the timing wave. The association is represented as a *timing model* or *timing information stream.* The timing model tracks the amplitude, phase, and frequency of the reference signal. Once this relationship is known, it can be used to guide other kinds of data analysis on-the-fly.

Fractional-sample intervals

If a perfect relationship existed between the sampling clock and observed waveform cycles, each cycle could align in such a way that each reference-signal cycle begins exactly at a zero crossing boundary and exactly at a captured sample – and the reference waveform would show phase 0.000 and value 0.000 at this point. In reality, the true waveform zero crossing points are expected to occur at some intermediate points between the ones where the hardware timing clock captures samples.

Using the combined information spanning the total shape of the waveform, which is known to be sinusoidal, an interpolation analysis can estimate a best match for the entire cycle. This makes the analysis impervious to small random variations, typically accurate to about 1/1000 of a sampling time interval.

Limitations of accuracy

To the extent that sampling is not perfect, and the reference timing waveform is not a perfect sine wave, small variations will result in tracking that is not quite at the theoretical ideal. The tracking process requires time to distinguish the effects of a phase shift from the effects of a frequency shift. Random real noise and tiny numerical errors result in small displacements between the model and actual waveform.

Particularly when the reference timing is derived from an AC power source, it is likely that there will be harmonic distortions in the waveform shape. Even in signals that appear very clean, amplitudes of these distortion components might total 5% to 10% of the fundamental reference signal frequency. Harmonics shift the apparent locations of waveform zero crossings, interfering with phase estimates, thus interfering with the process of locating the waveforms. The distortion of wave peaks influences amplitude estimates. Distortion also causes local rate-of-rise abnormalities that interfere with frequency estimates.

Filtering is applied to reject these disturbances as much as possible, but the cancellation is not perfect. Thus you should not depend on absolute accuracy. With a clean reference signal, the errors in its parameter tracking will typically approach 1 part in 100000 (i.e., 5 fully trustworthy digits), and could be better. There is zero long term drift. When you have less clean signals, you will need to test for how much the signal noise and distortions influence local tracking accuracy in your special situation.

Frequency resolution

To get adequate localization of the waveform, your sampling rate should allow for 50 or more samples per reference waveform cycle. Excessive numbers of samples per waveform is not a problem, and if that situation occurs, the very high data rates are reduced internally for processing efficiency.

Resampling

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

If the hardware sampling rate happens to be an exact integer multiple of the desired reference-based rate, and if the sampling and reference clocks are somehow phase-aligned, you have the ideal situation. All samples you want coincide with samples that the hardware captures. But in general, slight rate and "phase" discrepancies will mean that there is no perfect alignment of sampling and evaluation locations. This leads to 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 locations between previously measured sample locations. The accuracy of this reconstruction is good enough that you can't distinguish signal reconstruction errors from the errors that occur anyway when continuous analog measurements are rounded to the nearest bit 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 complicated and temperamental sample-clocking devices such as clock multipliers, digital counters and phase-locked loops. The computations are fast. The sample values that you get in the *resampled stream* will be indistinguishable from samples synchronized by hardware to the external timing signal.

Power analysis

The timing analysis produces estimates of amplitude and frequency as a side effect. This information can be reported optionally. Since the waveform tracking is quite accurate, a power analysis based on that waveform model can be quite accurate as well.

5. Command Reference

This section describes details of the individual commands. There is only one command, WAVESCAN, provided by the Wave Synchronization Module (**wsm.dlm**). This command is used so often with the time base resampling command TBRESAMP from the Time Base Resampling Module (**tbrsm.dlm**) that the command description for that command is repeated here.

Processing Command

Module **WSM :: WAVESCAN**

Perform a timing analysis and optionally report signal parameters of a sinusoidal waveform.

Syntax

```
WAVESCAN( REFIN, SMPTIME, REFFREQ, PTIMING [, PPROPS] )
```

Parameters

`REFIN`
: Single pipe with input samples captured from a sinusoidal reference timing signal
WORD PIPE

`SMPTIME`
: Sampling scan interval used to digitize timing signal, microseconds
FLOAT CONSTANT | DOUBLE CONSTANT

`REFFREQ`
: Nominal frequency of the reference timing waveform in Hertz
FLOAT CONSTANT | DOUBLE CONSTANT

`PTIMING`
: Pipe receiving the output timing analysis information to be used by other tasks
DOUBLE PIPE

`PPROPS`
: Optional cycle-by-cycle report of sine wave parameters of reference signal
DOUBLE PIPE

Description

A **`WAVESCAN`** task analyzes reference data from samples of a clean, high-level sinusoidal voltage signal received from pipe `REFIN`. The `SMPTIME` parameter specifies the sampling time interval of the timing data stream, in units of microseconds per sample, as might appear on a `SCAN` command line. The `REFFREQ` parameter specifies the reference frequency of the timing source in Hz. Specifications for both the sampling rate and the timing signal frequency should be reasonably accurate. A significant discrepancy between specified and actual rates will be diagnosed as an error and cause shutdown of the task before it can fully start.

For best results, the timing signal must be a low-distortion, low-impedance sinusoidal wave spanning a significant portion of the converter full signal range. Full tracking accuracy requires a highly accurate and clean waveform signal, as one would expect from an instrument grade signal source. Though less accurate, typically within a few thousandths of a radian, tracking also works with signals that drift in amplitude, frequency, and harmonic distortion, such as a typical AC power distribution voltage waveform. Tracking is typically robust to short term temporary faults, such as temporary cable removal or “switching events” disturbances that are typical of power systems. However, under severe, rapid-fire, repeated faults, it is possible to “completely lose” phase alignment between the sample stream and timing reference. If your timing signal has been subjected to severe disturbances, it is recommended that you stop the system and then restart it to allow accurate phase realignment. If you cannot do that,

another option is to resample your time base signal, and perform a single-frequency DFT analysis on that signal, to measure its phase shift. This can allow you to correct your data sets for the permanent phase shift at a later time.

Output timing information is reported once per waveform cycle in the `PTIMING` output pipe. The values placed into the `PTIMING` output pipe are in groups of 3, and do not have a defined interpretation. However, any time the second of the three terms is nonzero and negative, that indicates problems with signal quality or loss of synchronization.

The optional `PPROPS` output pipe can be used to deliver information about the waveform observed on the timing input channel. This is, in effect, a presentation of waveform information maintained in the internal waveform tracking model. A set of three `PPROPS` properties will be sent to the `PPROPS` pipe for each cycle of the timing reference signal. The returned fields are in a `double` data type, and consist of the following in sequence:

1. *Amplitude estimate.* Units are "converter ticks" corresponding to the sampled signals as originally received, but in a mixed fractional form. This is the amplitude of the fundamental frequency peak, excluding waveforms, noise, harmonics, etc.
2. *Frequency estimate.* The frequency is expressed in Hertz. Normally, the signal is considered to be a perfect frequency reference, but this frequency estimate turns that on its head and assumes that the sampling rate is perfect, reporting what the external clock signal appears to be relative to that. If the reference frequency is known to be highly accurate, any changes can be a useful indication of drift in the sampling clock rate.
3. *Cumulative phase discrepancy.* In effect, this is the integral over time of the difference between observed and nominal frequencies. This value would be 0.0 if sampling and reference clock were in perfect agreement. This value gradually increases when the observed frequency in the wave data is faster than expected, relative to the nominal sampling rate, and gradually decreases when the observed frequency in the wave data is slower than expected, relative to the nominal sampling rate. This can be meaningful if sampling is slaved to an absolute timing reference; it would then indicate net phase gain or loss in the timing signal relative to the absolute reference.

When first started, the WAVESCAN command requires a number of samples to estimate the waveform's properties and calculate a reasonable initial state for the wave tracking. Tracking converges relatively fast, but it will take some amount of time to converge to full accuracy. If you need full accuracy from the outset, start the WAVESCAN processing of signals early, allow time for settling, and then use triggering features to select data from the aligned signals.

If you have a signal generator that is synchronized to an IEEE 1588 network timing, the generator can establish time-alignment to the network, and then the xDAP processing can establish time-alignment of its data streams to the generator. In effect, this aligns data sets to the common network time base. However, keep in mind that this alignment is not the same thing as synchronization in real time. For example, data streams captured on multiple channels at 50000 samples per second, using any kind of devices and acquisition equipment, are highly unlikely to be delivered without large data buffering and transfer delays. On some networks, the time of transfer is not deterministic. The moment of capture can be resolved and tagged with a high degree of accuracy, but it is not knowable exactly when information will arrive at other locations in the network in real time.

Examples

```
WAVESCAN( IPipe6, 20.0, 100.0, pTime )
```

Analyze samples of a reference sine wave signal produced by a precision waveform generator and delivered via the input channel pipe `Ipipe6`. The sampling rate for the measurements is 20 microseconds per sample in each channel. The reference signal is 100 Hz. The results of the timing analysis are placed into pipe `pTime`.

```
WAVESCAN( IPipe6, 83.34, 60.0, pTime )
TBRESAMP( Ipipe(2), 1, pTime, 300.00, pResampled )
```

The processing is intended to track the nominal 60 Hz power frequency. The power waveform is reduced to a safe low level by a step-down transformer, and observed on channel `Ipipe6` at a nominal 83.34 microsecond-per-sample scan rate. The `WAVESCAN` command monitors the 60 Hz reference wave with roughly 200 samples per cycle captured along the reference 60 Hz signal. Timing information goes into the `pTime` pipe. The `TBRESAMP` task then accesses the signals from channels `Ipipe2` to `Ipipe5` at exactly 40 samples per 60 Hz cycle, locked to the reference rate, placing the resampled streams in the `pResampled` pipe. Blocks of 1000 resampled values from each channel will be very well suited for a spectrum analysis, showing the first 20 power system harmonics, at intervals of 25 frequency steps between harmonics. Because of almost perfect alignment, no windowing is required for follow-up harmonic analysis.

```
WAVESCAN( IPipe0, 160.0, 60.0, pTime, pWaveParms )
SKIP( pWaveParms, 0, 1, 2, pVoltage )
AVERAGE( pVoltage, 6, pDisplay )
```

Timing information is captured from one 60-Hz reference voltage phase, observed in the input channel pipe `IPipe0`. Measured at 160.0 microsecond time intervals, this provides approximately 104.16 samples per wave cycle – this is plenty to give good time resolution, and the exact number of samples per cycle makes no difference. Timing information is delivered into pipe `pTime` for other tasks to use. Waveform parameter information is issued once per timing cycle to the supplementary output pipe `pWaveParms`. Amplitude is of special interest, so this value is selected using a SKIP command. The stream of amplitude values is reduced using an AVERAGE command with data processed in groups of 6, producing one result per 1/10 second intervals in the pipe `pDisplay`. This is a useful rate for directly updating a *graphical instrument* display.

Processing Command

Module **WSM :: WAVEREGEN**

Reconstruct an ideal waveform sample stream that tracks the reference timing signal.

Syntax

```
WAVEREGEN( PTIMIN, [PPROPS,] PREGEN )
WAVEREGEN( PTIMIN, AMPLITUDE, PREGEN )
```

Parameters

`PTIMING`
: Pipe of timing analysis information received from a WAVESCAN task
DOUBLE PIPE

`PPROPS`
: Optional wave properties received from a WAVESCAN task
DOUBLE PIPE

`AMPLITUDE`
: Optional constant amplitude specification for the output wave
WORD CONSTANT

`PREGEN`
: Pipe receiving the regenerated pure sine wave signal
DOUBLE PIPE

Description

A **`WAVEREGEN`** task synthesizes a mathematically pure sine wave signal synchronized to the timing information from the `PTIMING` pipe, as produced by a separate **`WAVESCAN`** task. The generated data are placed into the `PREGEN` output pipe, where they can be used as a cleaned signal for retransmission, as a reference signal for software triggering, as displayable data to compare to the original timing signal, or for various other purposes. The phase of the generated wave is locked to the fundamental frequency of the reference waveform, so depending on the nature of the harmonic distortions, the zero-crossing locations of the regenerated wave and the original wave might not align exactly. The regenerated sample stream corresponds sample-for-sample with the samples of the original reference signal analyzed by the **`WAVESCAN`** task.

The `PTIMING` pipe is mandatory. This pipe provides all of the necessary phase, frequency, and alignment information from a separate timing analysis.

The desired amplitude of the output signal can be specified using one of the following optional parameters:

- A `PPROPS` pipe, as produced by the same **`WAVESCAN`** task that produced the timing information. This amplitude is dynamically estimated to track the amplitude of the fundamental frequency of the timing signal, typically within about 0.01% of the true amplitude of the fundamental component. Because of harmonic distortions, the peaks might not match the peak levels in the original timing signal.

- A constant `AMPLITUDE` value, in digitizer units, range 1 to 32767. The output signal is appropriately scaled to have the specified arbitrary peak.

If neither parameter is specified, an arbitrary amplitude of approximately half full range is assumed.

Examples

```
WAVEREGEN( pTiming, pParameters, pRegen )
pNoise = pRefSignal - pRegen
RMS( pNoise, 1000, pRMS )
```

A **`WAVEREGEN`** task is used to rebuild a mathematically ideal waveform matching the fundamental frequency in the reference timing signal. The timing analysis `pTiming` pipe and waveform property `pParameters` pipe provide the data produced by a separate **`WAVESCAN`** task. Differences between the regenerated ideal signal `pRegen` and the original timing signal `pRefSignal` are calculated by a DAPL expression task and placed into the `pNoise` stream. If the original signal is perfect, and synchronization is perfect, all of the differences should be zero. The DAPL system's RMS task calculates the root-mean-squared values of the actual differences, over blocks of 1000 samples. An excessively large number will indicate a signal problem such as distortion or noise interference.

Processing Command

Module **TBRSM :: 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 interval expressed in microsecond units
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. Calculations for each signal channel are processed separately, but the results preserve the multiplexed-data organization. The resulting sample values are placed into the `OUTSTREAM` data pipe, at the new sample 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 in a typical configuration the sampling is relatively fast (for good time resolution) and the aligned data are relatively slower (for efficient representation of signal information without too much redundancy). Both rates are typically much higher than the timing interval rate of the time base signal.

Nothing forces alignment between the reference timing hardware 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 bits.

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 need, 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 = 166.6666666666
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 166.6666666666 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 – retaining 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 rate. 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 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 approximating the *Whitaker-Shannon Theorem* have these too, but the intent is to make the side-effects as small as possible.

Low frequencies should see no detectable gain error effects.

The interpolation filters in the TBRESAMP command deliberately sacrifice accuracy at frequencies higher than 40% of the Nyquist frequency of the new 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 can be elevated enough to make sure that the desired frequency band falls within that suggested 40% of Nyquist frequency bound.

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 no phase distortion effects. However, practical implementation restrictions will sometimes select resampling locations that are 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, localized small time displacements can result in signal variations appearing very similar to low-level random noise. Unlike true random noise, the magnitude of these effects decrease both as frequency and as amplitude decrease, so 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 available 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 pr-calculated 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 meaningful improvement to measurement quality.

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