Anti-Alias Multichannel Module

Command reference and application guide

Anti-Alias Multichannel Module:
Command reference and application guide

Version 1.00

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: MSAAMMOD100

Table of Contents

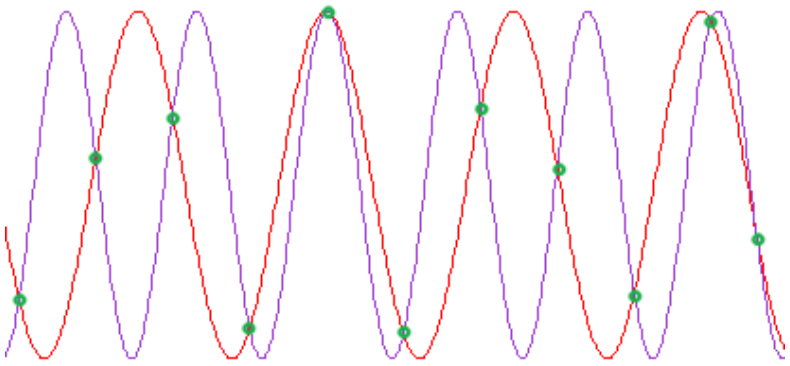
1. Introduction

The downloadable Anti-Aliasing Multichannel Module (AAMM.DLM) for xDAP systems provides a digital approach to anti-aliasing filtering, with amplitude and phase accuracy that is impossible using traditional analog anti-alias filtering schemes.

The following is an example of applying the AAMFILT anti-alias filtering command on a bank of 4 audio signal channels, processing high resolution signal data captured at 192000 samples per second, and then reducing to a 44100 samples per second rate with zero aliasing and zero flat-band distortion.

```
CONSTANT   oldfreq double = 192000
CONSTANT   newfreq double = 44100
CONSTANT   nchannels word = 4
PIPES    pAudio word
. . .
   AAMFILT ( IPipes(0..3), nchannels, oldfreq, newfreq, pAudio )
```

When capturing samples of sinusoidal signals by sampling at equal discrete-time intervals, multiple sinusoidal signal frequencies can produce exactly the same sets of samples. Once a signal has been digitized, there is no way to determine from the sampled data set alone which frequencies were the ones actually present prior to digitizing.



Given only the sample values shown in green, how could you know which of the two waveforms was the one sampled?

The only way to make the measurements unambiguous is to have practical constraints guaranteeing that the signals were restricted to a certain known band prior to digitizing.

> As an aside, it is possible to take sparse measurements of a very high frequency signal, and knowing the frequency band of the input signal, shuffle the measured sample values appropriately to reconstruct an accurate and very high resolution picture of the waveform. Modern digital oscilloscopes apply this idea extensively. However, *this is not the intent of the AAMM module*.

For most measurements of sound, vibration, temperature, pressure, and so forth, physical constraints limit changes in the measurements to relatively low frequencies, starting near zero and extending upward through a limited band. Under these constraints, the frequencies are uniquely represented in the sampled data set provided that there are no frequencies in the original signal higher than the *Nyquist Limit* at ½ the sampling rate. Or, another way to say this, only frequencies for which there at least 2 samples per waveform cycle can be distinguished unambiguously.

It is tempting to say: “Well, I know that I will have some high frequency noise, but I can get rid of that by averaging my data sets.” In some cases, that is true, but those are special cases in which the undesirable high frequencies are *uncorrelated,* that is to say, white noise. Noise sources are not always so uncorrelated. Some examples of when things can be vulnerable:

- sensors pick up a wider bandwidth than the desired signal band
- noise results from power harmonics and is closely related to 50 or 60 Hz
- noise results from floor or fixture vibrations with distinct resonance modes

In these cases, the high frequency noise will *alias*, appearing in the sampled data as if bands of lower frequencies were present, when in fact they were not. These hazards are avoided by using the AAMFILT anti-aliasing command and using it consistently.

2. AAMFILT Performance Characteristics

This section discusses the technical characteristics of the anti-alias filtering. This is for purposes of reference. When using the AAMFILT processing, you will not need to think about any of this – except for knowing that it works.

No prior aliasing assumption

The AAMFILT processing should be applied to a signal sampled fast enough that the entire spectrum is covered, hence no frequencies present in your signal can cause aliasing. If this assumption is not true, there is a possibility that some aliasing has already occurred during sampling at the very high rate, and after that, there is no way to compensate. It is strongly recommended that you check your system as discussed in **Appendix 2** to determine whether this potential hazard is real.

If you have well-shielded and properly terminated signal lines, you should find that there are no aliasing hazards. Unterminated signal lines can "float" away from ground easily, and are relatively vulnerable to coupled interference from RF or digital circuit sources. A typical termination would be a 10K to 100K ohm load resistor from each signal line to ground, at the data acquisition end of the signal cable. If this is insufficient, a bypass capacitor of perhaps 100 to 1000 pF in parallel with the load resistor can provide additional high frequency attenuation.

In rare cases, simple bypassing is not sufficient, because the interfering frequencies are driven by too much signal power. For these cases, some hardware filtering is required. Using the AAMFILT processing retains an advantage, because small, inexpensive components will work fine for the low-order supplemental filters.

Flat band

For a full-range signal, within the frequency band from 0 to 80% of the Nyquist frequency for the output rate you specify, the frequency response gain is 1.0 within one converter count. This is equivalent to *passband flatness* of approximately ±0.00025 dB.

For example, if you specify that an output sample rate is 5000 Hz, you can think of your data as being completely accurate to 16 bits from 0 to 4000 Hz.

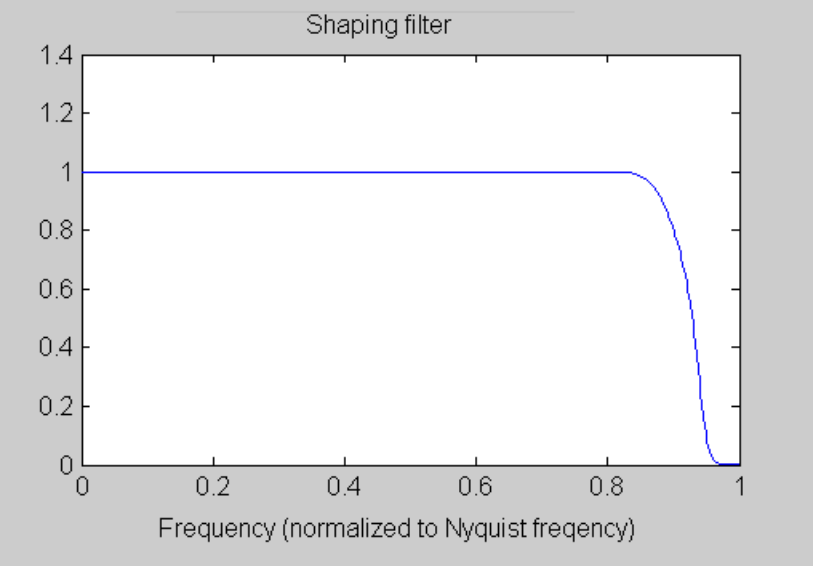
Noise

Because the input data are in a fixed point format, every value is adjusted to an integer value. These adjustments have a very similar effect to adding low-level white noise to data streams. When this noise interacts with the anti-aliasing filters, it can result in occasional random errors of 1 or 2 converter counts, in addition to the plus or minus 1 count expected in each original sample.

Transition Band

Isolating a flat band perfectly, removing all possible frequencies that could alias, and extending this all the way to the theoretical Nyquist limit, would require a filter of infinite size. Since this is clearly not practical, there is a transition band from 80% to 100% of the Nyquist frequency. In this band, signals are attenuated.

To give a repeatable frequency characteristic in the transition zone, every filter configuration finishes with a *shaping filter* that enforces a consistent frequency rolloff relative to the final sampling rate. The following illustration shows this characteristic, with location 1.0 on the x-axis representing the location of the Nyquist limit at half of the final sampling rate.



The shaping characteristic is not “brick wall.” Attenuation begins gradually at the edge of the flat band and reaches the effective half-power point near 90% of the Nyquist frequency (exact location is not specified). Attenuation increases rapidly beyond that point.

Aliasing

Aliasing of a rejected frequency of full range amplitude leaves less than 1 converter count of residual noise within the flat band from 0 to 80% of the Nyquist frequency at the final output rate. In practice, this is zero aliasing. If you had a noise signal at full amplitude, there would be no dynamic range left for the signal you are trying to measure. Aliasing effects this small will be hidden by the bit quantization chatter inherent in the discrete analog-to-digital converter.

In the transition band, total removal of aliasing is not guaranteed. Residual aliasing is possible within the notch region above 95% of the Nyquist frequency. However, the shaping filter will enforce at least -60 dB attenuation, so in practice it is extremely difficult to produce a measurable alias response, even within this band.

Phase

The phase response is 0.0 ± 0.01 radians from 0 through 80% of the Nyquist frequency.

The phase errors are largest near the 80% of Nyquist frequency limit, so if you use only part of the flat band, your phase errors will be less.

Phase response is unspecified in the band above 80% of the Nyquist frequency.

Time Delay

The zero phase shift is achieved by shifting and aligning data streams. These shifts are not visible in the data sets. However, they can affect the moment in real time when the data become available for transfer to the PC host or to some other subsystem. This delay can be observed as a *transfer latency,* or as an *input-output response latency.* If you need to account for this latency, for purposes such as aligning xDAP data samples with samples reported by other sampling devices clocked at the same time, you can run the `latency.exe` console application, included with the AAMM.DLM downloadable module. Start a Windows command console window. Use the `CD` command to locate the folder containing the `latency.exe` command, and make this the current working folder. Start the program by typing the command `latency.` It will prompt for a decimation factor; type in the one you intend to use, and press the *Enter* key. The latency, in units of input sample time units and output sample time units, will be displayed. Press the *Ctrl-Z* keys in combination to end the session.

3. Application Examples

This section describes some typical applications using the *Anti-Aliasing Multichannel Module.* Keep in mind that these are *preprocessing* applications, executed by embedded DAPL 3000 processing on-the-fly, so that all of this is completed independently, before your application program receives any data.

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, and then incrementally adapting them. Or if you prefer to work in a different environment, 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 0: Anti-aliasing response (simulation)

This configuration can be used "straight out of the box" to examine computed results of the AAMFILT command before you have your application set up.

This application generates a mathematically pure sinusoidal waveform at an adjustable frequency. The rate reduction / anti-aliasing processing is then applied to this signal. The original signal and the reduced signal are then plotted, so that you can see the results of the anti-alias and rate reduction processing.

> Reducing the sample rate means that there are fewer samples per waveform cycle. Plotting the original data set and the reduced data set on the same plot, using the same "x axis" resolution, is not really valid; when viewed in this manner, the rate reduction looks like a shift to a higher frequency. Actually, the frequency is the same as before – it is really the points that are shown too close together. Take this into consideration when viewing the plot graphics.

To run this test, open the DAPstudio Configuration window, click the Processing tab, and then the Declarations sub-tab. You can highlight and edit the values of the following three configuration variables.

1. The `oldrate` variable specifies the number of samples per second used to capture the original high-rate data.
2. The `newrate` variable specifies the number of samples per second you want in your reduced data set (without introducing any aliasing). The ratio of `oldrate` to `newrate` is the *decimation factor*.
3. The `wavelength` variable specifies the number of samples per waveform you want in your test sine wave.

In the original application configuration, there are 100000 samples per second, to be reduced to 10000 samples per second, a *decimation factor* of 10. The sine waveform is initially configured with 28 samples per wave cycle. After the sample rate reduction by a factor of 10, the 28 samples per wave cycle will be reduced to 2.8 samples per cycle. This barely resembles the original wave, and yet because there more than 2.5 samples per cycle (thus below 80% of the Nyquist limit), the reduced data stream remains valid with its magnitude unchanged.

- Click the Start! item in the main menu and see what happens in the plot display for the original and reduced rate signals. Click Stop! in the main menu to complete the test run.
- Return to the Declarations sub-tab, and modify the `wavelength` to 17 samples per cycle. After the rate reduction by factor 10, the result is 1.7 samples per cycle, less than 2. Ordinarily, this reduction would cause aliasing, but in this case the AAMFILT processing removes frequencies like this that might alias.

See the `AAMApp0.dms` application file for DAPstudio.

DAPL configuration:

```
// Set the frequency and rate reduction here
CONSTANT  oldrate  double = 100000.0
CONSTANT  newrate  double = 10000.0
CONSTANT  wavelength word = 28

PIPES   pHirate  word
PIPES   pTimed   word
PIPES   pReduced word

// Use the sample hardware to provide display timing
IDEFINE  timing
  CHANNELS    1
  SET   IP0  D0
  TIME     1000
  COUNT   50000
END

// Calculate the reduced rate signal
PDEFINE  simulate
  // Generate the pure waveform prior to rate adjustment
  COSINEWVE(24000, wavelength, pHirate)
  // Deliver data at a controlled rate for simulation
  pTimed = pHirate + IP0*0
  // Apply waveform rate adjustment and anti-alias
  AAMFILT(pTimed, 1, oldrate, newrate, pReduced)
  COPY(pTimed, pReduced, $BinOut)
END
```

Application 1: 8-channel acquisition with anti-aliasing

This application processes accelerometer data from 8 channels for a collision test. The relevant bandwidth ranges from 0 to 1000 Hz. In a traditional configuration, this application would use a bank of hardware anti-alias lowpass filters with corner frequency of approximately 10 kHz, with a sampling frequency approximately 25 kHz, roughly at the limit of the frequency band that the accelerometer devices can generate. While feasible, this is tricky because the filters distort frequencies inside the bands that are captured by the digitizer. In this application, the AAMFILT command is used to avoid the hardware filtering and associated distortions.

Solution: Connect the 8 accelerometer channels to 8 differential input connectors. These are sampled simultaneously at a high rate, 50000 samples per second from each channel. This is higher than the highest accelerometer output frequency or any plausible interfering signals. To represent a 1000 Hz band unambiguously, the sampling rate must be no less than 2000 samples per second. To avoid pushing the theoretical limits, a 5000 sample per second rate is selected as the final data rate. Configure an `AAMFILT` command to reduce the sampling rate from 50000 sample/second rate to 5000 samples per second. This processing will represent frequencies from 0 through 2000 Hz with no attenuation, phase distortion, or aliasing.

DAPL configuration:

```
idefine MslInput
   channels 8
   set IP0 d0      // Eight accelerometer channels
   set IP1 d2
   set IP2 d4
   set IP3 d6
   set IP4 d8
   set IP5 d10
   set IP6 d12
   set IP7 d14
   scan  20.00     // 50000 samp/sec, each channel
end

pipes pAAlias word
constant  highrate  double  =  50000.0
constant  newrate   double  =  5000.0

pdefine filtering
  // Receive and AA filter data at high rate
  AAMFILT(IPipes(0..7), 8, highrate, newrate, pAAlias)
  // Transfer data to host at desired reduced rate
  COPY(pAAlias, $BinOut)
end
```

See the `AAMApp1.dms` application file for DAPstudio.

Application 2: Precise audio capture with rate calibration

This application captures two channels of audio data. In addition to the usual rate reduction with anti-aliasing, this further illustrates how subtle rate adjustments can be applied at the same time to improve timing accuracy.

To be sure of accurate audio representation, each channel is sampled at a very high rate. The very high frequencies catch the noise and distortion; there is no audio information there. After capture, the extraneous high frequencies are removed using AAMFILT processing, eliminating any possibility of aliasing, as the data rate is reduced to a slower and more conventional one for storage.

There are two practical complications that interfere with the timing accuracy of this application.

1. The desired sampling rate is 144000 samples per second – three times faster than commonly-used 48000 samples per second rate. To get exactly 144000 samples per second, the xDAP must be configured to sample at a scan interval of 6.9444444 microseconds. The digital sampling controls in the xDAP digitizers can be configured for a programmable interval of 6.944 microseconds, but not 6.9444444. Thus, samples are captured at a rate of 144009.21 samples per second rather than 144000 samples per second, slightly faster than nominal. This difference is taken into account when specifying the initial sampling rate.
2. The sampling clock rate on the xDAP equipment is rated to better than ± 50 parts per million. However, most of this is a constant rate offset; short term dynamic *drift* is much lower than this, particularly over a very short time period. By running a sampling configuration for 10 minutes at 100000 samples per second, and comparing to a highly accurate time base, 60 million plus 1320 samples were observed, where nominally 60 million were expected. This experiment tells us that for this one particular xDAP unit in its current operating configuration, the sample timing is running at 22 ppm faster than reference time, a factor 1.000022 . To compensate, the nominal output rate is reduced by this factor.

Solution: Connect the 2 microphone preamplifier signals to input channels that are sampled simultaneously. Set the sampling rate to capture samples at a scan interval of 6.944 microseconds. Tell the `AAMFILT` command that the input rate is the nominal 144009.21 sample/second, consistent with the scan time configuration. Instead of specifying the nominal target rate of 44100 samples per second for the reduced stream, apply a timing correction factor of 1.0/1.000022, specifying an output sample rate of 44099.03 samples per second. The adjustment factor results in a final output data rate extremely close to the ideal 44100 samples per second audio rate.

DAPL configuration:

```
idefine MslInput   // Calibrated: “fast by factor 1.000022”
   channels 2
   set IP0 d0      // Two precision audio channels
   set IP1 d2
   scan  6.944     // Faster than ideal 144000 samples/sec
end

pipes pAudio word

// new rate = 44100 samples per second after rate offset
constant  highrate  double  =  144009.21
constant  audrate   double  =   44099.03

pdefine filtering
  // Receive and AA filter data at high rate simultaneously
  AAMFILT(IPipes(1,3), 2, highrate, audrate, pAudio)
  // Transfer data to host at compensated target rate
  COPY(pAudio, $BinOut)
end
```

See the `AAMApp2.dms` application file for DAPstudio.

4. Command Reference

The AAMM.DLM module contains only one processing command, the AAMFILT command.

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 AAMM.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 AAMM.DLM module will be reloaded automatically at future restarts.

Processing Command

Module **AAMM :: AAMFILT**

Apply data rate reduction to multiple data channels without aliasing.

Syntax

```
AAMFILT( PINMUX, NCHAN, OLDRATE, NEWRATE, POUTMUX )
```

Parameters

PINMUX
: Input data, the form of multiplexed signal samples
WORD PIPE

NCHAN
: Number of multiplexed channels in the PINMUX stream
WORD CONSTANT

OLDRATE
: Sample rate, samples per second, in each input channel
DOUBLE CONSTANT

NEWRATE
: Sample rate, samples per second, in each output channel
DOUBLE CONSTANT

POUTMUX
: Output data, multiplexed like input data, at new rate
WORD PIPE

Description

An AAMFILT command reduces the data sample rate, for multiple data channels, applying the required digital filtering to avoid exposure to aliasing. The input samples arrive via pipe PINMUX, in groups of size NCHAN with one sample for each input channel, and with NCHAN in the range 1 to 256 channels. The original sampling rate at which the samples in each channel were captured is specified by parameter OLDRATE. The new sample rate at which the output samples are to be produced is provided by parameter NEWRATE. The samples for the reduced-rate sample stream are placed into the POUTMUX pipe, organized in the same groups as the input data.

The OLDRATE parameter should accurately reflect the rate at which each channel of the original data stream is captured. If you have calibrated the sample timing for your Data Acquisition Processor, include any timing adjustment factor in the rate value specified. The parameter NEWRATE can be an arbitrary rate equal to or less than the original sample rate. It is not necessary for this to be a special fraction of the input rate, or of the Data Acquisition Processor digital time base frequency. The *decimation factor* is defined as the ratio OLDRATE/NEWRATE. To be more precise, it is this decimation ratio, and not the absolute values of the two terms, that matters. So, for example, if OLDRATE is set to 3.0 and NEWRATE is set to 2.0, the decimation factor is 1.5, or equivalently, the rate is reduced to 2/3 of the original rate.

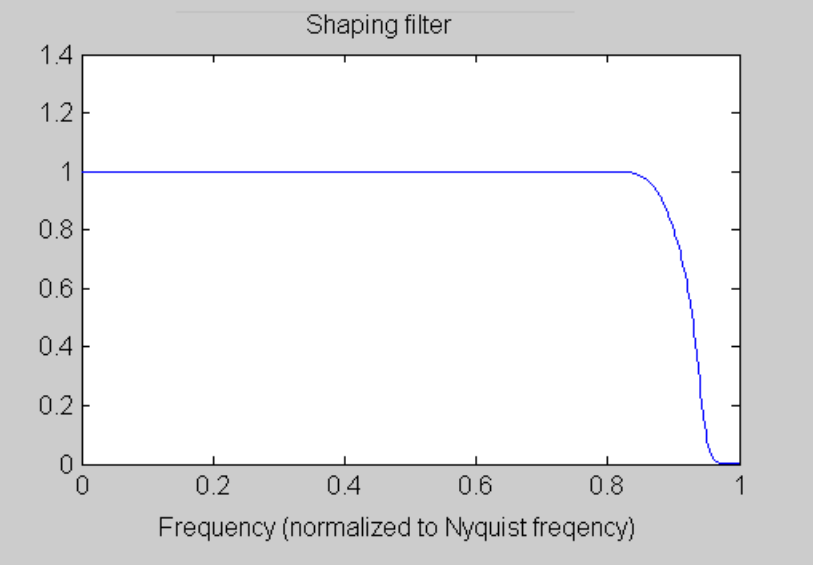
With arbitrary rate changes, nothing guarantees that samples in the output stream align exactly with samples of the input data stream (after the first sample). However, by virtue of being "lowpass filters," the anti-aliasing filters supply the necessary smoothness properties for accurate digital reconstruction of signal values at any desired locations. Consequently, it does not matter if your hardware did not capture a sample at *the correct location* (figuretively speaking) for the given new sample rate. The filtering is of course not perfect, but with errors smaller than the rounding chatter in the least significant bit (LSB) of your analog to digital converter hardware, you will not be able to tell the difference.

In general, when digitizing an analog signal at some frequency `Fs`, frequency bands of width `Fs` centered at multiples of frequency `Fs` will alias onto frequencies centered at frequency 0. Your sampled data will be completely free of any aliasing effects only if all frequency bands higher than the *Nyquist frequency* at `Fs/2` are empty at the time the analog signals are sampled.

Well-designed applications will sample a little faster than the absolute theoretical limits, and this means that some of the frequencies close to the new Nyquist frequency are not meaningful to the application. Some aliasing effects can exist in this high range and this will not damage the band that you care about. ***You must guarantee that there is no harmful aliasing present in the band you care about when you sample your data initially at the high sampling rate.*** In most cases, a well-terminated signal line with a loading resistor, sometimes supplemented by a small bypass capacitor, will have no aliasing hazards. In a few cases, supplemental hardware filters remain necessary to remove energetic high frequencies.

Once you have the input streams sampled at a high frequency, free of harmful aliasing effects, then you can apply the `AAMFILT` command to reduce the rate to a more desirable and useful lower rate, without introducing any new aliasing. The `AAMFILT` command removes any potentially harmful frequencies from the original high-rate data, prior to the decimation.

The `AAMFILT` command establishes a consistent high frequency rolloff shape close to the new Nyquist frequency, by applying a final shaping filter. The following graph illustrates the frequency response characteristic imposed on the output data streams.



This is the spectrum envelope you will get for any decimation factor that you specify, relative to your final sampling rate. The shaping filter has the following characteristics:

- Flat response within +- 0.0003 dB (16^{th} bit) from 0 through 80% of the Nyquist frequency
- Stopband edge at approximately 95% of the final Nyquist frequency
- Phase accuracy +- 0.01 radians from 0 through 80% of the Nyquist frequency
- Phase response unspecified above 80% of the Nyquist frequency

Example

```
CONSTANT   sampling  double = 125000.0
CONSTANT   final     double = 48000.0

AAMFILT(IPipe(0,1), 2, sampling, final, PAAFILT)
```

You want accurate sampling at 48000 samples per second for two channels using extended range microphones. Your microphones cannot track frequencies above 50000 Hz, but there are some frequencies between 24000 Hz and 50000 Hz that could cause aliasing problems if you sampled at the desired 48000 samples per second directly. Instead, you capture the two channels of digitized data at a higher rate. A sampling rate of 125000 samples/second is chosen; specify a `SCAN 8.0` (microseconds) sample interval in the DAPL configuration. Set the `sampling` parameter to 125000.0 samples per second, and specify the `final` desired rate to be 48000 samples per second after alias filtering and rate reduction. The reduced data channels are placed into the `PAAFILT` pipe. The anti-aliasing filters clean out the bands above the new Nyquist limit of 24000 Hz, guaranteeing that there is no aliasing as the high-rate data stream is reduced to the rate you specify.

Appendix I. Digital Anti-Alias Technology

If you are unsure how digital signal processing can work for anti-aliasing, given what you already know about this topic, this section is for you. Don't just read about it, put this to the test.

Doesn't the valid sampling require bandlimiting filters?

No it doesn't! Not always. Filtering is only required when the signal you are processing digitally is not already suitably bandlimited. That is, high frequencies can alias and hurt you... but not if they don't exist.

AAMFILT vs. “traditional” anti-aliasing strategies

These aren't really as different as they might appear.

The traditional strategy can be summarized as follows:

> *Chop away any frequencies that you don't want and that could potentially cause problems. Then digitize at the sample rate you want.*

The AAMFILT strategy can be summarized as follows:

> *Digitize everything at high rates – preserving frequencies that you want and that you don't want. Then discard frequencies you don't want using digital filtering – without aliasing.*

The traditional filtering strategy

The traditional approach is completely valid, but surprisingly difficult and expensive to apply well.

- To avoid distortion, the filter linearity must be extremely good.
- The frequency response characteristic must be extremely flat.
- The phase response must present near-constant group delay to avoid phase distortions that alter waveform shapes.
- The filter design must have very good selectivity so that frequency bands are well isolated.
- The filter hardware must not introduce its own contaminating noise or resonant “ringing.”

Electronic devices that do all of these things well tend to be relatively expensive. They must be designed carefully and built using stable, tight-tolerance components. Once these filter designs are implemented, there is no flexibility. So in practice you probably need to makes some compromise between the frequency band that you would like to analyze and the filter characteristic of an available device.

The fact is, regardless of how good the filter is, some amount of signal corruption is unavoidable, since small gain and phase errors are inherent in all of the commonly used filter characteristics.

As an example, consider an application to measure a signal accurately through a 1000 Hz band. The filter prototype is presumed to roll off at the "normalized corner frequency" 1.0. Suppose that you select a fourth-order Butterworth filter as a compromise between fast rolloff and moderate phase distortion. To remove high frequency noise to a -60dB level (a compromise), you will need to select a normalized sampling frequency *fh* such that

$$1 / (1 + f_h^{\,4}) < 0.001$$

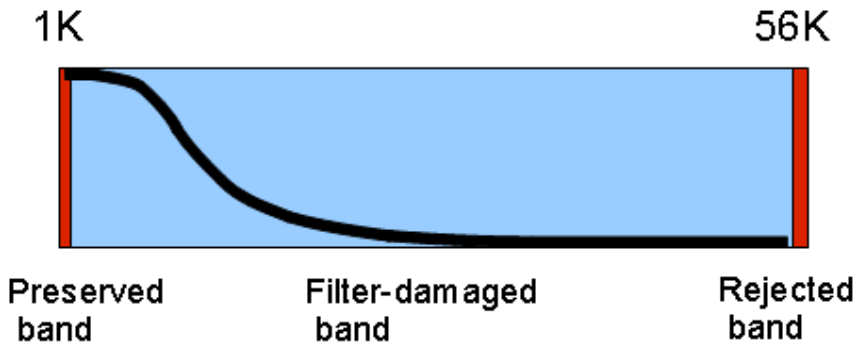
$$f_h > 5.6$$

If you want measurements accurate to at least 14 bits (a compromise), you will then allow very little attenuation at low frequencies.

$$1 / (1 + f_l^{\,4}) > (8191/8192)$$

$$f_l < 0.10$$

Thus, to preserve the normalized frequency normalized sampling frequency $fl = 1$ kHz accurately, your filter cutoff would need to be located at roughly frequency 10000 Hz, and your sampling rate would need to be approximately 56000 samples per second. That rate might seem a little excessive, given that 2000 samples per second is the theoretical band size able to represent the signal with full accuracy, but nevertheless, that is the nature of this particular filter. It often goes unrecognized how much hardware filters can damage signals.



A higher order filter with a steeper cutoff transition will allow the preserved flat band, filter cutoff frequency, and sampling rate to be closer together, so you don't need to sample as fast. For example, you can achieve the same (compromised) performance using an 8th order Butterworth filter if your sampling is at least 7.3 times as fast as the highest frequency you want to preserve – a general "rule of thumb" is to use a factor of 10, allowing for variations in cutoff frequency. Higher order filters tend to be more expensive, however, with compromises in band flatness, phase, and sensitivity.

The AAMFILT Strategy

There was a time when the digital processing required for the AAMFILT strategy was beyond all reason. But time and technology have moved forward.

The xDAP equipment is perfectly happy to capture a small number or a huge number of samples. Since it doesn't care, should you? Actually, *you should,* because you don't want to deal with clogged data channels and masses of redundant data streaming into your application software. Having obtained valid digitized data, it is possible to use digital filtering to eliminate frequencies that you don't care about. This is where you need the AAMFILT command. Optimized filtering and then rate reductions are applied automatically, without introducing any new aliasing or distortion.

This is not trivial. If you have a digitized signal with high frequencies preserved accurately, and you then reduce the sampling rate, you can end up with exactly the same aliasing effect as if capturing directly from the original signal. That is why the AAMFILT processing **does not do that**. After *digital lowpass filters* completely remove any frequencies that could cause aliasing damage, there is nothing left to alias and the rate reductions are safe.

In a sense, this is the same thing that hardware anti-alias filters have always done. There are some significant differences, however.

- There is a major cost difference between applying precise electronic filtering instruments and applying computer bits.
- The rigorous mathematical precision of DSP algorithms is far beyond the capabilities of even the highest quality analog electronic filters.
- The digital filters are completely adjustable, while the hardware filters offer few options.
- System design is much easier. You don't need to worry about filter phase, corner frequencies, attenuation, and so forth. Just sample fast enough.

Returning to the example in which you want accurate measurements of a signal through 1000 Hz, instead of asking *where should the filter corner frequency be,* ask *how much bandwidth is necessary to represent everything in the signal.* Suppose that the answer for this particular application is that the sensors simply cannot respond to anything at 50000 Hz and beyond. Use a sampling rate sufficiently beyond this limit, and that leaves nothing to alias onto the bands that you care about.

In addition to requiring a flat band signal capture through 1000 Hz, suppose that 5000 samples per second is deemed to provide a good representation. To allow a sufficient margin, choose 55000 samples per second as the original sampling rate. Then, simply tell the AAMFILT processing to reduce the sample rate from the initial 55000 samples per second to the desired 5000 samples per second. The appropriate, optimized filtering is applied automatically. Compared to the effort to properly deploy hardware filters, this almost pays for itself in setup time alone.

Appendix II. Checking for high frequency interference

There was an Old Man who supposed
That the street door was partially closed;
  But some very large Rats
  Ate his coats and his hats
While that futile Old Gentleman dozed.

(– Edward Lear, “Book of Nonsense,” 1846)

Don't be futile.

Rather than assuming that the door is shut to high frequency interference that might corrupt your measurements, you should verify that troublesome frequencies do not exist. You can use the application file `AAMspect.dms` with DAPstudio to quickly determine how much (if any) residual exposure you have to aliasing at the time of your high-rate sampling.

Connect one xDAP input channel to the signal that you intend to sample, with signal terminations and signal source as close as possible to conditions you expect for the finished application. The following configuration shows how this signal is sampled and processed.

DAPL configuration:

```
PIPES   pSpect   word
PIPES   pAvSpect word

IDEFINE  highrate
  CHANNELS    1
  SET   IP0  D0     // Select input channel
  TIME      2.0     // 500000 samples/second
END

// Calculate signal power distribution
PDEFINE  simulate
  // Analyze data from high-rate channel
  MIXRFFT(500,FORWARD,HAMMING,IP0,HALF,MAGNITUDE,pSpect)
  // Average over 100 blocks (1/10 second duration)
  BAVERAGE(pSpect, 250,100, pAvSpect)
  // Send spectrum data to host system
  COPY(pAvSpect, $BinOut)
END
```

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

The returned spectrum blocks will show the RMS magnitude of the signal level in 250 *frequency bins*, each of which spans a 1 kHz band, from frequency 0 to 250 kHz.

Frequencies that can be trouble will be centered at the sampling frequency and multiples of it. So, for example, if you propose to sample at 60000 samples per second, you would check the spectrum locations 60, 120, 180, and 240, at the multiples of the proposed sample rate.

At each of these locations, the terms of specific concern extend above and below by the width of the band you want to measure. For example, if you intend a final sampling rate of 5000 samples per second, you would also check the 5 spectrum bins above and below each of the special locations, to verify that there is no significant signal energy in any of these frequencies bands.

It is perhaps worth mentioning that there can be significant high frequencies elsewhere, and these do not matter. They have the same potential to alias, but when they do, they can only damage measurements in frequency bands that your AAMFILT processing will discard. You can sometimes take advantage of this fact when there happens to be a high frequency interference that bypass filters cannot remove completely. Sometimes an adjustment of the initial sampling rate can reposition the frequency bands of interest so that the remaining aliasing has no effects on the frequency bands that you care about.

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