

# UBAfxP Reference Manual

Paul H. Milenkovic <sup>1</sup>

Department of Electrical and Computer Engineering

University of Wisconsin-Madison

1415 Johnson Drive

Madison, Wisconsin 53706

Copyright 2006

All Rights Reserved

<sup>1</sup>Please refer inquiries to Paul Milenkovic, 118 Shiloh Dr., Madison, Wisconsin 53705-2433, U.S.A., 608/833-7956, <http://userpages.chorus.net/cspeech>, [cspeech@chorus.net](mailto:cspeech@chorus.net)

## Abstract

UBAfxP is the ActiveX library for the synchronized display of the acoustic waveform, acoustic analysis traces, and an x-y display of point-parameterized articulatory data. The analysis traces include fundamental frequency, RMS or dB intensity, time-frequency spectrogram with formant-track overlay, and the time-slice spectrum. UBAfxP is derived from TF32, the time-frequency analysis software program for 32-bit Windows for the display and measurement of speech or other audio-frequency waveforms. UBAfxP is usable with Matlab, wxPython, Visual Basic 6.0, Visual Studio .NET C# and Visual Basic .NET, Delphi, or other development software under Microsoft Windows compatible with ActiveX.

The UBAfxP library contains a single ActiveX control called UBeamAfx, which packages a complete acoustic and articulatory viewer application. The goal is to automate the data reduction of synchronized audio and articulatory recordings. The ActiveX control is usable within a scripting environment such as Matlab or Python, and the control has interface functions allowing access to waveform samples, analysis traces, statistical summaries, and articulatory coordinates presented in the displays.

UBeamAfx was designed for use with data collected with the University of Wisconsin X-ray Microbeam facility, but it also works with a commonly-used binary format for data collected with a magnetic articulograph.

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Functions and Properties</b>	<b>4</b>
2.1	Global . . . . .	5
2.2	Wave plot . . . . .	6
2.2.1	File commands . . . . .	6
2.2.2	Screen, cursor indices, and waveform playback . . . . .	8
2.2.3	Waveform measurement . . . . .	12
2.3	F0 plot . . . . .	14
2.4	RMS plot . . . . .	16
2.5	Time-frequency plot . . . . .	18
2.6	Label plot . . . . .	20
2.7	XY plot . . . . .	22
2.8	Spec plot . . . . .	24

## 1 Introduction

The University of Wisconsin X-ray Microbeam project has collected and disseminated data consisting of acoustic recordings synchronized with the acquisition of time-varying pellet marker positions that track the movement of the tongue, jaw, and lips during speech. The acoustic recordings are stored in binary-format files while the articulatory measurements are stored in alphanumeric text files. The project has also disseminated a version of the TF32 program with an articulatory viewer for viewing and labeling these data along with a command-line program called DX for extracting pitch, formants, and articulatory traces at labeled intervals into files for further data reduction.

Can speech data be directly viewed, labeled, and extracted within a software package used for data reduction without storing intermediate values in collections of files? Microsoft Windows has a facility called ActiveX for distributing visual displays in a software library in a manner that these displays can be presented in any of a number of different software packages. A software package hosting an ActiveX control can call functions for direct transfer of data displayed in that control.

The UBeamAfx ActiveX control within the UBAfxP software library packages elements of the TF32 program as an acoustic and articulatory viewer. While UB is shorthand for "Microbeam" in the parlance of staff members of the project, the viewer can display magnetic articulograph data as well. This viewer can appear within a Matlab figure window or within a Web page displayed by Microsoft Internet Explorer or within any other software application running under Windows that is able to host ActiveX controls. When the viewer is displayed within a Matlab figure window, the user can make cursor selections and interact with the waveform, analysis, and articulatory displays much as with the stand-alone TF32 program. Furthermore, Matlab function, invoked from the Matlab command line or from M-file scripts can invoke functions on the UBeamAfx control to copy waveform, analysis, and articulatory samples at specified locations or intervals into Matlab variables.

UBAfxP is distributed as the ActiveX module `UBAfxP.ocx`, which may be installed on a Windows computer by copying to a directory of your choice and by invoking the command

```
C:\myactivex>regsvr32 UBAfxP.ocx
```

and uninstalled with the command

```
C:\myactivex>regsvr32 /u UBAfxP.ocx
```

The Matlab M-file called `AArView.m` can be invoked from the Matlab command line according to

```
aa = AArView
```

This action will place an acoustic articulatory viewer in the default Matlab figure window, and it will create a Matlab handle-variable called `aa` that provides access to the data presented in that viewer.

The functions and properties of the `UBeamAfx` control from the `UBAfxP.ocx` ActiveX library can now be accessed with the `aa` handle variable within Matlab. The command

```
aa.OpenWaveFile('D:\vol1\ubdb\jw12\tp009.acc')
```

opens the waveform file `tp009.acc` from the drive and directory `D:\vol1\ubdb\jw12` into the display. One can optionally specify

```
s = aa.OpenWaveFile('D:\vol1\ubdb\jw12\tp009.acc')
```

This action sets the value of Boolean variable `s` to `True` if the file is successfully opened, the `False` otherwise. The command

```
aa.FileName
```

reads the `FileName` property variable and returns the text string giving the last-opened file name. Assigning the property variable

```
aa.FileName = 'D:\vol1\ubdb\jw12\tp009.acc'
```

is an alternate way of opening a waveform into the display.

The list of ActiveX functions can be viewed within Matlab by

```
invoke(aa)
```

while the list of property variables can be viewed with

```
set(aa)
```

What follows is a listing of all of these functions and property variables along with descriptions of their purpose. For information on how to interact with the visual display that appears within a Matlab figure window, please refer to the TF32 User's Manual.

## 2 Functions and Properties

The UBAfxP ActiveX library contains the single ActiveX control called UBeamAfx. While this arrangement is less flexible than having multiple, interacting ActiveX controls, a single control simplifies layout within a Matlab figure window by the AArView.m script. All interactions with the acoustic, analyser, and articulatory displays take place through functions and properties of UBeamAfx.

An ActiveX control has functions that are callable by programs or scripts – in the case of Matlab, by commands or by M-file scripts. An ActiveX control also has properties. Properties look like variables which can be assigned or set, but in reality a property refers to a pair of functions – a Get function that returns a value of the specified type and a Set function that takes one value of that type as its argument. In Matlab, one does not see the Get and Set functions – one only sees a property variable that can be on either the left or right side of the assignment-operator equal sign.

While there is only one interface to the UBeamAfx control, the functions (also called methods) and properties break down into logical groupings: global, wave plot, F0 plot, RMS plot, time-frequency spectrogram with formant-track overlay, text labels, x-y articulatory plot, and time-slice spectrum.

## 2.1 Global

The following apply to the collection of UBeamAfx displays as a whole.

### **CopytoClipboard**

#### **aa.CopytoClipboard**

Copies the plots in the visible wave trace displays – waveform, F0, RMS, time-frequency spectrogram with optional formant-track overlay, labels – as a Windows metafile image to the system clipboard. That image can be pasted into Microsoft Word or other application capable of accepting metafile images. Includes only those plots selected by checkboxes for display. Does not include the x-y articulatory display or the time-slice spectrum – those displays have their own Copy buttons.

## 2.2 Wave plot

### 2.2.1 File commands

The following methods open waveforms from files into the display.

#### OpenWaveDlg

```
s = aa.OpenWaveDlg;
```

Displays a dialog for opening a waveform file or a waveform file list for display (same as `TF32 Files Open` command). Optional return variable `s` is `true` if the user selects a valid file. A valid file is either a waveform file in one of the supported formats or a waveform list file naming a list of waveform files.

#### OpenWaveFile

```
s = aa.OpenWaveFile(fname)
```

Opens the named file for display without displaying a dialog. Variable `fname` may be a string variable or a quoted string. Optional return variable `s` is `true` if `fname` specifies a valid file.

#### SelectPrev

```
aa.SelectPrev
```

Selects the previous waveform file in a list (as does `TF32 Files Prev` command). If a waveform file has been opened into the display, selects the previous waveform file in the same directory in alphabetical order. If a waveform list has been opened into the display, selects the previous waveform file in that list.

#### SelectNext

```
aa.SelectNext
```

Selects the next waveform file in a list (as does `TF32 Files Next` command). If a waveform file has been opened into the display, selects the previous waveform file in the same directory in alphabetical order. If a waveform list has been opened into the display, selects the next waveform file in that list.

#### PrevEnabled

```
enabled = aa.PrevEnabled
```



Return variable `enabled` equals `true` if a file selection for the `SelectPrev` method is available.

### **NextEnabled**

```
enabled = aa.NextEnabled
```

Return variable `enabled` equals `true` if a file selection for the `SelectNext` method is available.

### **FileName**

```
aa.FileName = fname  
fname = aa.FileName
```

Assigning the `FileName` property to string `fname` opens that wave file or wave file list into the display. Assigning `fname` to the `FileName` property reads the file name of the waveform opened into the display using `OpenWaveDlg`, `OpenWaveFile`, or by assigning to the `FileName` property.

### 2.2.2 Screen, cursor indices, and waveform playback

The following methods get and set the initial and final cursor markers along with the placement of the displayed screen interval within the waveform buffer extent. Integer index values are keyed to the primary channel – the first channel in the display. The number of samples of the primary channel is returned by `Get_primary_msamp` and the interval between samples in milliseconds is returned by `Get_primary_ms`. Valid integer index positions are 0 through `Get_primary_msamp - 1`. Indices are reported in ms by multiplying the integer index by `Get_primary_ms`.

Index values may be set either with type integer or type double. In Matlab,

```
aa.Set_icur(1000)
```

sets the initial cursor to sample number 1000 (ms value is  $1000 * \text{primary\_ms}$ ) while

```
aa.Set_icur(1000.0)
```

sets the initial cursor to ms value 1000.0. Index values are always returned as type double (8-byte, double-precision floating-point value) in units of ms.

#### SetScreen

```
aa.SetScreen(cursors)
```

Sets the initial and final display screen indices, sets cursors within that placement of the displayed screen interval. Argument `cursors` specifies a variable-length array with elements in the order `iscn`, `fscn`, `icur`, and `fcu`. Supplying a 2-element array sets the initial and final screen positions; supplying a 4-element array additionally sets the cursor positions; elements beyond 4 are ignored. An array of type `double` specifies ms values; an array of type `integer` specifies cursor index values. Maintain the relationship  $\text{iscn} \leq \text{icur} < \text{fcu} \leq \text{fscn}$ . Set `icur = iscn` to remove the initial cursor from view, set `fcu = fscn` to remove the left cursor from view.

#### Get\_iscn

```
iscn = aa.Get_iscn
```

Returns the ms-value of the left edge of the waveform display window.

#### Get\_fscn

```
fscn = aa.Get_fscn
```

Returns the ms-value of the right edge of the waveform display window.

**Get\_icur**

```
icur = aa.Get_icur
```

Returns the ms-value of the initial cursor – returns the left edge of the screen if the initial cursor is not visible.

**Set\_icur**

```
aa.Set_icur(icur)
```

Sets the initial cursor – sets the final cursor to the right edge of the screen if past the final cursor placement. Specifies ms if `icur` of type `double`; specifies an index value if `icur` of type `integer`.

**Get\_fcur**

```
fcur = aa.Get_fcur
```

Gets the ms-value of the final cursor – returns the index of the right edge of the screen if the final cursor is not visible.

**Set\_fcur**

```
aa.Set_fcur(fcur)
```

Sets the index of the final cursor – sets the initial cursor to the left edge of the screen if prior to the initial cursor placement. Specifies ms if `fcur` of type `double`; specifies an index value if `fcur` of type `integer`.

**Get\_primary\_msamp**

```
msamp = aa.Get_primary_msamp
```

Returns the maximum number of samples in the primary (first) waveform channel. Valid cursor integer indices are 0 through `Get_primary_msamp - 1`; valid cursor ms values are 0.0 through  $(\text{Get\_primary\_msamp} - 1) * \text{Get\_primary\_ms}$ .

**Get\_primary\_ms**

```
ms = aa.Get_primary_ms;
```

Returns the interval between samples in milliseconds of the primary (first) waveform channel. The quantity `Get_primary_msamp` times `Get_primary_ms` is the time span of the waveform buffer.

The following commands and properties control waveform playback of cursor-selected interval of the selected channel

**PlayWave**

```
aa.PlayWave(to_end)
```

Plays the waveform starting at the initial cursor position (or left edge of the screen if the initial cursor is not visible). Argument `to_end` of `False` plays to the final cursor position or to the right edge of the screen. Argument `to_end` of `True` plays past the final cursor and right edge of the screen all the way to the end of the waveform buffer, scrolling the waveform display interval as this takes place.

If a waveform is opened into the display with two channels of the same sampling rate, duration, range, and units, playback takes place in stereo. If not, the first channel is played in mono.

**PlayStop**

```
aa.PlayStop
```

Stops any playback that is in progress.

### 2.2.3 Waveform measurement

The following functions designate a waveform channel in the display – the allowed channel number is the range 1 to NChan.

#### Get\_readout

```
readout_str = aa.Get_readout(ch)
```

Gets the string naming the measurement readout mode of the designated channel. Empty string: no readout; other values are `endpoints`, `stats`, and `RMS_dB`.

#### Set\_readout

```
aa.Set_readout(ch, readout_str);
```

Sets measurement readout mode of the designated channel.

#### GetReadoutValues

```
value_array = aa.GetReadoutValues(ch)
```

Returns an array of double values of the numeric readouts appearing in the channel display for the selected mode. Returns an empty array if no readout is selected.

#### GetSampleSlice

```
sample_array = aa.GetSampleSlice(ch, slice)
```

Returns an array of waveform sample values for the designated channel for the interval selected by `slice`. If `slice` is a two-element array, selects the initial and final index of that interval. Integer values select integer indices; double value specify ms indices. If `slice` is a scalar value, the interval is from that index position to the end of the waveform. Specify

```
sample_array = aa.GetSampleSlice(ch, 0)
```

to return the entire channel contents.

#### Get\_sample\_ms

```
ms = aa.Get_sample_ms(ch)
```

Returns the interval between waveform samples in ms – `aa.Get_sample_ms(1)` same as `aa.Get_primary_ms`.

## **HPF**

`aa.HPF(ch, f_c, newchan)`

Applies a non-causal zero-phase frequency-sample design high-pass filter to the designated waveform channel. Argument `f_c` is the cutoff frequency in Hz. If argument `newchan` is `false`, overwrites the designated channel with the filter output; otherwise, stores the filter output in a new waveform channel added to the display. This filter is useful for removing DC-offsets or other low-frequency artifacts the waveform channel before measuring dB readouts

### **2.3 F0 plot**

The F0 plot displays the pitch trace computed from the selected waveform channel. If an F0 plot has been edited and saved to a file, the edited version will be retrieved. The F0 plot can be configured by right clicking on the F0 display and making selections from a dialog.



**F0\_readout**

```
aa.F0_readout = readout_str  
readout_str = aa.F0_readout
```

Assign this property to set the readout mode of the F0 plot. Empty string: no readout; other values are `endpoints`, and `stats`. Read this property to determine the readout mode selected either by assigning the property or by setting the readout from the F0 dialog.

**F0ReadoutValues**

```
value_array = aa.F0ReadoutValues
```

Returns an array of double values of the numeric readouts appearing in the F0 display for the selected mode. Returns an empty array if no readout is selected.

**F0IndexSamples**

```
sample_list = aa.F0IndexSamples(index_list)
```

Returns F0 sample values at the specified index positions. Integers specify integer sample indices; double values specify ms sample indices. The index list can be a scalar, array, or matrix, and the returned sample list will be a scalar, array, or matrix of the same dimensions and structure giving the F0 value or values and the supplied indices. Invoking

```
sample_list = aa.F0IndexSamples(aa.LblIndexList('mylabel'))
```

is a way to read all of the F0 values marked by text label `mylabel` in the label plot – see description of the label plot function `LblIndexList`.

## 2.4 RMS plot

The RMS plot displays a trace of the RMS or dB value of the selected waveform channel, computed by sliding a window over the waveform. The RMS plot can be configured by right clicking

### RMS\_readout

```
aa.RMS_readout = readout_str
readout_str = RMS_readout
```

Assign this property to set the readout mode of the RMS plot. Empty string: no readout; other values are **endpoints**, and **stats**. Read this property to determine the readout mode selected either by assigning the property or by setting the readout from the RMS dialog.

### RMS\_use\_dB

```
aa.RMS_use_dB = enable
enable = aa.RMS_use_dB
```

Assign property to True to enable the dB mode of the RMS plot. Assign False to disable dB mode and revert to RMS. Read this property to determine the dB mode set by assigning this property or set from the RMS dialog.

### RMS\_window\_ms

```
aa.RMS_window_ms = ms
ms = aa.RMS_window_ms
```

Assign this property to set the the duration of the sliding window (in ms) used to generate the dB or RMS trace. Read this property to determine the window duration set by assigning this property or set from the RMS dialog.

### RMS\_Hamming

```
aa.Hamming = enable
enable = aa.RMS_Hamming
```

Assign property to True to use a sliding Hamming window in the dB or RMS trace generation. Assign False to revert to a rectangular window. Read this property to determine the window type set by assigning this property or set from the RMS dialog.

**RMSReadoutValues**

```
value_array = aa.RMSReadoutValues
```

Returns an array of double values of the numeric readouts appearing in the RMS display for the selected mode. Returns an empty array if no readout is selected.

**RMSIndexSamples**

```
sample_list = aa.RMSIndexSamples(index_list)
```

Returns RMS or dB sample values at the specified index positions. Integers specify integer sample indices; double values specify ms sample indices. The index list can be a scalar, array, or matrix, and the returned sample list will be a scalar, array, or matrix of the same dimensions and structure giving the RMS or dB value or values at the supplied indices.

```
sample_list = aa.RMSIndexSamples(aa.LblIndexList('mylabel'))
```

is a way to read all of the dB or RMS values marked by text label `mylabel` in the label plot – see description of the label plot function `LblIndexList`.

## 2.5 Time-frequency plot

The time-frequency plot computes a gray-scale time-frequency spectrogram from a selected waveform channel. Right click the display to change plot settings using a dialog.

### TF\_LPC

```
aa.TF_LPC = enable
enable = aa.TF_LPC
```

Assign property to True to enable the formant-track overlay of the time-frequency plot. Assign False to remove the formant tracks from the display. Read this property to determine the presence of formant tracks. Formant tracks are computed from the selected waveform channel; formants may be edited and saved to a file, and the edited formant tracks are automatically retrieved when the same waveform is opened into the display.

### TF\_ncoef

```
aa.TF_ncoef = int_value
int_value = aa.TF_ncoef
```

This property determines the number of LPC coefficients used to compute formant tracks – default number is 2 times the waveform sample rate in kHz plus 4 more for spectrum shaping. This number of coefficients is used when computing formant tracks – retrieved formant tracks are based on the number of coefficients used when they were computed.

### TFComputeFormants

```
aa.TFComputeFormants
```

Computes formant values over the cursor-selected interval – same as Compute button in the time-frequency dialog.

### TFSmoothFormants

```
aa.TFSmoothFormants
```

Recomputes formants values over the cursor-selected interval using LPC log area-ratio smoothing and assigns formant tracks to LPC poles using a dynamic programming algorithm – same as Smooth button in the time-frequency dialog.

**TFTrackFormants**

```
aa.TFTrackFormants
```

Takes current LPC pole locations by reassigns formant tracts to LPC poles over the cursor-selected interval using the same dynamic programming algorithm as Smooth – same as Track button in the time-frequency dialog.

**TFComputed**

```
is_computed = aa.TFComputed
```

Returns True if the LPC formant tracks are computed; returns False if the LPC formant tracks have been retrieved from a file saving editing changes or have been edited with Smooth or Track.

**TFIndexFormants**

```
formant_list = aa.TFIndexFormants(index_list)
```

Returns formant values at the specified index positions when formant tracks are enabled. Integers specify integer sample indices; double values specify ms sample indices. The index list can be a scalar, array, or matrix, and the returned values will be triples of formants F1, F2, and F3 at the index values. If the index list is a Matlab row vector, the return value will be a row vector of triples (F1(idx1), F2(idx1), F3(idx1), F1(idx2), F2(idx2), F3(idx2), . . .). If the index list is a Matlab column vector (matrix transpose of a row vector), the return matrix will have rows for each index position and three columns: formants F1, F2, and F3. If the index list is a Matlab matrix, the number of columns gets expanded by 3 to contain the formant triples.

```
sample_list = aa.TFIndexFormants(aa.LblIndexList('mylabel'))
```

is a way to read all of the formant values marked by text label `mylabel` in the label plot – see description of the label plot function `LblIndexList`.

## 2.6 Label plot

The label plot displays text labels along with initial and final index markers. Right click on the label display to select dialogs for displaying and marking labels.

### **Lbl\_nlabel**

```
count = aa.Lbl_nlabel
```

Returns the number of labels.

### **Lbl\_str**

```
str = aa.Lbl_str(idx)
```

Returns the label text of the selected label – label selection variable `idx` is in the range 1 to `nlabel`.

### **Lbl\_initl\_ms**

```
ms = aa.Lbl_initl_ms(idx)
```

Returns the initial ms index of the selected label – label selection variable `idx` is in the range 1 to `nlabel`.

### **Lbl\_final\_ms**

```
ms = aa.Lbl_final_ms
```

Returns the final ms index of the selected label – label selection variable `idx` is in the range 1 to `nlabel`.

### **LblIndexList**

```
index_list = aa.LblIndexList(label_string)
```

Returns a matrix of index values for the collection of labels having the specified label string. The rows of that matrix are the pairs of index values for successive labels while the two columns are the initial and final index positions. This function is useful for making measurements at a collection of times marked with the same label string. The returned index list can be directly supplied to `F0IndexSamples`, `RMSIndexSamples`, `TFIndexFormants`, or `XYIndexPel` to read out sets for F0, RMS/dB, formant, or articulatory x-y positions at a list of labeled times.

**Lbl\_idx**

```
idx = aa.Lbl_idx(label_string, occur)
```

Returns the label index in the range 1 to nlabel of the specified label string. Integer occur is in the range 1 to max\_occur: 1 returns the first occurrence of label\_string, 2 returns the second occurrence of label\_string. The returned label index may be supplied to Lbl\_initl\_ms or Lbl\_final\_ms to retrieve times associated with the specified label string and occurrence. Returns 0 if the label string and occurrence cannot be located. Invoking

```
str = aa.Lbl_str(aa.Lbl_idx('mylabel', 1))
```

returns the value mylabel if at least 1 occurrence of mylabel exists while

```
ms = aa.Lbl_initl_ms(aa.Lbl_idx('mylabel', 1))
```

returns the initial ms value of that label.

**Lbl\_max\_occur**

```
max_occur = aa.Lbl_max_occur(label_string)
```

Returns the number of occurrences of the specified label string; returns 0 if the label string cannot be found.

## 2.7 XY plot

The XY plot displays midsagittal articulatory positions of pellet markers of the lips, tongue, and jaw, and this display is synchronized with the acoustic recording in Channel 1 of the display. The articulatory positions are read from an ASCII files that accompanies the acoustic waveform file. The XY plot has buttons for configuring the display, displaying traces of articulatory movement, and for linking the articulatory cursor with the waveform display. The time frame of an articulatory position is displayed as a black cursor on the waveform display, and this black cursor sweeps across the waveform to indicate the time position during playback.

### XYIndexPel

```
xyv_list = aa.XYIndexPel(ipel, index_list)
```

Returns pellet marker values for the given pellet number at the specified index positions. The pellet numbers are in the order they appear in the numeric readouts of the XY plot: for Microbeam data, 1 is upper lip, 2 is lower lip, 3-6 are tongue starting with 3 at the tongue tip, 7 is jaw incisor, and 8 is jaw molar. As for the index values, integers specify integer sample indices; double values specify ms sample indices. The index list can be a scalar, array, or matrix, and the returned values will be triples of x, y, and v where position x, y is in mm and pellet speed v is in mm/s. If the index list is a Matlab row vector, the return value will be a row vector of triples (x(idx1), y(idx1), v(idx1), x(idx2), y(idx2), v(idx2), . . .). If the index list is a Matlab column vector (matrix transpose of a row vector), the return matrix will have rows for each index position and three columns: x, y, and v. If the index list is a Matlab matrix, the number of columns gets expanded by 3 to contain the pellet triples.

```
xyv_list = aa.XYIndexPel(3, aa.LblIndexList('mylabel'))
```

is a way to read all of the values for pellet number 3 (tongue tip in Microbeam data) marked by text label `mylabel` in the label plot – see description of the label plot function `LblIndexList`.

### XYMsAlign

```
ms = aa.XYMsAlign(index_ms)
```

Returns the nearest pellet marker time sample value to the specified index ms value. In Microbeam data, the pellet markers are sampled every 6.866 ms starting at  $t = 0$ . The function `XYIndexPel` returns interpolated values of pellets – if you want the exact values for pellets, determine the ms time of the nearest pellet marker frame to the time you want using the `XYMsAlign` function. For example,



```
xyv = aa.XYIndexPel(3, aa.XYMsAlign(aa.Get_icur))
```

returns x, y, and v values for pellet 3 (tongue tip) at the nearest pellet frame to location of the initial waveform cursor.

### **XYMsMax**

```
ms = aa.XYMsMax
```

Returns the maximum ms value of the pellet markers.

### **XYMsPos**

```
ms = aa.XYMsPos
```

Returns the ms time of the current pellet marker frame as shown by the position of the black cursor on the waveform plot and by the articulatory display in the XY plot.

## 2.8 Spec plot

The spec plot displays the time-slice spectrum of the cursor-selected interval of a waveform channel.

### Spec\_align\_curs

```
aa.Spec_align_curs = enable
enable = aa.Spec_align_curs
```

Assign property to True to align the spectrum window extending from the initial waveform cursor towards the final waveform cursor up to a maximum of 1024 waveform samples.

### Spec\_LTA

```
aa.Spec_LTA = enable
enable = aa.Spec_LTA
```

Assign property to True compute a long-term average Fourier spectrum for a window centered at the initial cursor, moving in increments of 50 percent overlap, and ending at the final cursor. The long-term average spectrum requires the Hamming window or rectangular window Fourier modes.

### Spec\_align\_initl

```
aa.Spec_align_initl = enable
enable = aa.Spec_align_initl
```

Assign property to True to align the spectrum window centered at the initial waveform cursor position.

### Spec\_align\_final

```
aa.Spec_align_final = enable
enable = aa.Spec_align_final
```

Assign property to True to align the spectrum window centered at the final waveform cursor position.

### Spec\_window\_ms

```
aa.Spec_window_ms = ms
ms = aa.Spec_window_ms
```

For all other modes besides `Spec_align_curs`, assign this property to the ms length of the spectrum window. For all modes, read this property to determine the ms length of the spectrum window.

### **Spec\_Preem**

```
aa.Spec_Preem = enable
enable = aa.Spec_Preem
```

Assign property to `True` for waveform preemphasis in calculation the spectrum; assign `False` to remove preemphasis.

### **Spec\_Hamming**

```
aa.Spec_Hamming = enable
enable = aa.Spec_Hamming
```

Assign property to `True` to enable the Hamming-window Fourier spectrum.

### **Spec\_rect**

```
aa.Spec_rect = enable
enable = aa.Spec_rect
```

Assign property to `True` to enable the rectangular-window Fourier spectrum.

### **Spec\_LP**

```
aa.Spec_LP = enable
enable = aa.Spec_LP
```

Assign property to `True` to enable the covariance-method LPC spectrum.

### **Spec\_DCT**

```
aa.Spec_DCT = enable
enable = aa.Spec_DCT
```

Assign property to `True` to enable discrete-cosine transform smoothing of the Fourier spectrum – requires Hamming or rect window Fourier modes.

### **Spec\_Mmt**

```
aa.Spec_Mmt = enable
enable = aa.Spec_Mmt
```

Assign property to True to enable spectrum moments calculation and spectrum-moments smoothing of the Fourier spectrum – requires Hamming or rect window Fourier modes.

### **Spec\_avg**

```
aa.Spec_avg = enable  
enable = aa.Spec_avg
```

Assign property to True to enable spectrum averaging in 1 kHz bands – requires Hamming or rect window Fourier modes.

### **Spec\_ncoef**

```
aa.Spec_ncoef = int_value  
int_value = aa.Spec_ncoef
```

Assign property to desired number of LPC coefficients. Read property to determine number of LPC coefficients assigned.

### **SpecReadoutValues**

```
values_array = aa.SpecReadoutValues
```

Returns an array of the numeric readouts displayed in the Spec plot – useful for retrieving spectrum moments values in moments mode. The readouts are in the order 1) dB value of waveform time-slice being analyzed, 2) dB value of the windowed and preemphasized waveform, and 3) in moments mode, the moment values in the order mean, std deviation, coefficient of skewness, coefficient of kurtosis.

### **SpecGetdB**

```
dB_array = aa.SpecGetdB
```

Returns the array of spectrum dB values. Returns 513 values where dB\_array(1) in Matlab is DC, dB\_array(513) is at half the sampling frequency of the waveform analyzed.