VocalFusion Stereo Control Users Guide

IN THIS DOCUMENT

1 Introduction

This document describes how to use the provided command-line utilities to read and update the user configurable parameters on *VocalFusion Stereo* devices and evaluation kits.

The underlying mechanism for the control is provided by the *lib_device_control* library. This library implements host and device side APIs to provide acknowledged transport of control messages. A range of control transport mechanisms are available including USB, I^{2}C and xSCOPE. This allows changing of the *VocalFusion Stereo* parameters at runtime to meet specific application needs and available interfaces on the host.

The *XVF3500 DSP Databrief*[1] details the DSP processing of the VocalFusion stack.

2 Prerequisites

- This document assumes familiarity with the XMOS xCORE architecture, the XMOS tool chain and the xC language. Documentation related to these aspects which are not specific to this guide are linked to in the references appendix.
- For descriptions of XMOS related terms found in this document please see the XMOS Glossary[2].

[1] https://www.xmos.com/published/xvf3500-dsp-databrief
[2] http://www.xmos.com/published/glossary

Publication Date: 2018/6/28 Document Number: XM012738A

3 Controllable entities

The voice algorithm suite supports linear arrays to provide two stereo channels with up to 180° coverage and has been developed for far-field voice control in Stereo TV and/or video applications. All the pre-compiled firmware binaries have control enabled.

The code for control is guarded by pre-processor macros. Figure 1 summarizes the macros required to enable or disable control over a particular transport.

Figure 1: Preprocessor macros to disable/enable control

Transport	Defines to set to 1	Makefile arguments
USB	BECLEAR_CONTROL_USB	`-D BECLEAR_CONTROL_USB=1`
I^2C	BECLEAR_CONTROL_I2C	`-D BECLEAR_CONTROL_I2C=1`
xSCOPE	BECLEAR_CONTROL_XSCOPE	`-f xscope -D BECLEAR_CONTROL_XSCOPE=1`

When enabling xSCOPE control, in addition to enabling the control code using `BECLEAR_CONTROL_XSCOPE`, the tools-provided xSCOPE library must be linked into the application using the `-fxscope` argument.

The *VocalFusion Stereo* firmware broadly divides its microphone processing into two logical stages: Acoustic Echo Cancellation (AEC) and Beamforming & Post-processing (BAP). The *VocalFusion Stereo* device has two AEC blocks, one for each channel. Each block resides on a different tile and each has multiple control parameters associated with it. I/O control interfaces (such as I^2C) are connected to each logical block via xC interfaces, an array called `i_control` at time of writing. There are three in stereo devices, AEC12, AEC34 and BAP.

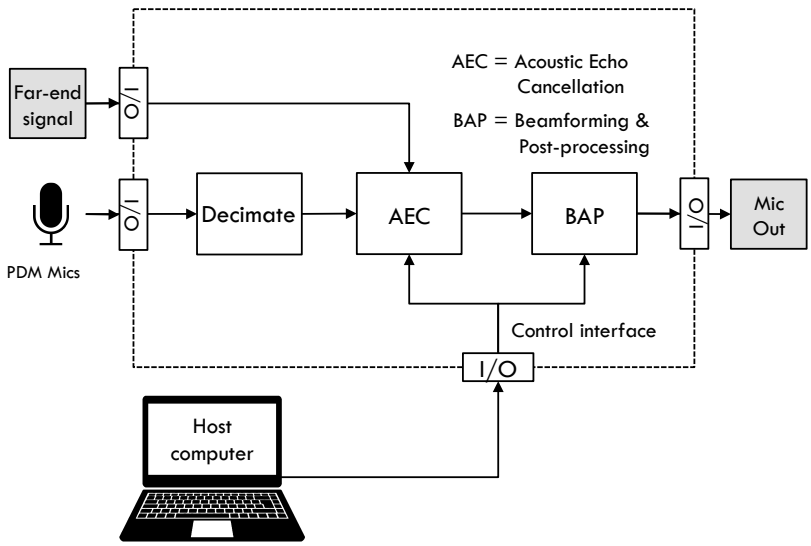

Figure 2: High level view of parameter control

A list of the available parameters that are runtime controllable can be found in the tables in appendix §9.2.

4 Building the command line utility

XMOS provides example command-line utilities for controlling parameters inside the *VocalFusion Stereo* firmware.

The command-line utilities are supplied as source with Makefiles for building under multiple platforms. The utility may be built from the source using commonly available compilers. The following instructions provide step-by-step guides to building the binary/executable.

Open a xTIMEcomposer command-line window and navigate to the directory containing the command-line utility:

```
lib_xbeclear/host/control/
```

And then use the following to build:

Microsoft Visual Studio is required to compile the host control utility on Windows. It has been tested with Microsoft Visual Studio Community 2015 (Version 14.0.23107.0 D14REL):

```
nmake /f Makefile.Win32
```

The above command will build two binaries `vfctrl_xscope` and `vfctrl_usb` which support xSCOPE and USB control transports.

Xcode is required to compile the host utility on macOS:

```
make -f Makefile.OSX
```

The above command will build two binaries `vfctrl_xscope` and `vfctrl_usb` which support xSCOPE and USB control transports.

g++ is required to compile the host utility on Linux.

For linux x86 hosts:

```
make -f Makefile.Linux64
```

The above command will build two binaries `vfctrl_xscope` and `vfctrl_usb` which support xSCOPE and USB control transports.

For linux ARM running on a Raspberry Pi 3:

```
make -f Makefile.Pi
```

The above command will build two binaries `vfctrl_usb` and `vfctrl_i2c` which support USB and I^2C control transports.

Linux and Pi builds list dependencies in their respective makefiles. These can be installed with by doing *apt-get install* on Debian derived systems. Typically, these are libusb, libreadline and libncurses5 with headers. Please refer to the makefile itself for details.

5 Windows Driver installation for USB control

The *VocalFusion Stereo* device with USB host connection is a composite USB device which requires a driver to be installed for the control interface when used with Windows. The driver is installed by locating the device in Device manager. Select **Control Panel ▶ Device Manager**. For other OS's or embedded hosts this step is not required.

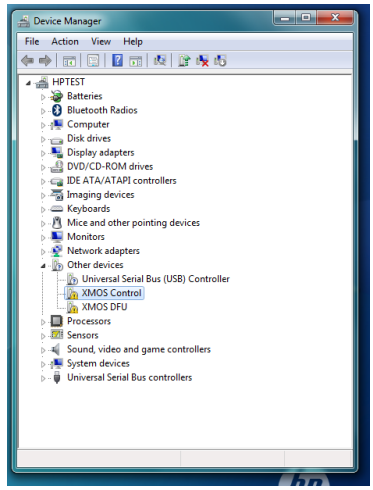

Figure 3: Device before driver install

When you plug in the board on a fresh system, two unknown devices will appear, *XMOS Control* and *XMOS DFU*. Right click on *XMOS Control* and select **Update Driver Software...**.

Select **Browse for driver software on your computer**, or equivalent, and select the `lib_xbeclear/host/control/libusb/Win32/driver` folder.

The driver should successfully install and create a device called *XMOS Microphone Array Control*.

Note that the *XMOS DFU* device remains unrecognized at this point. DFU host utilities are available for Raspberry Pi, Linux and macOS, please see *VocalFusion Stereo software design guide*.

The Windows host is now ready to send control requests to the device.

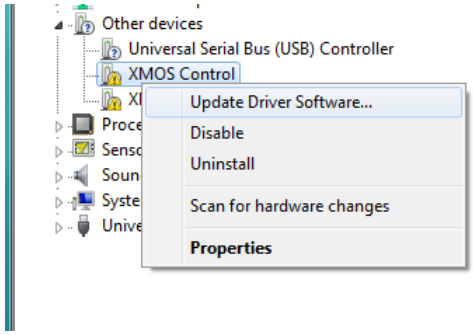

Figure 4: Update the driver

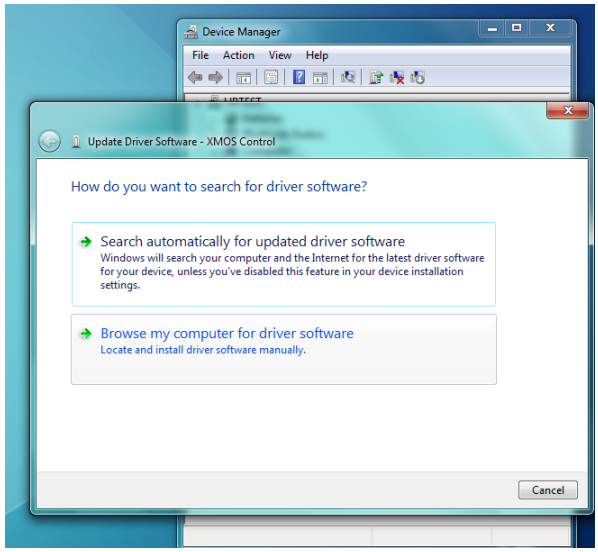

Figure 5: Locate the driver

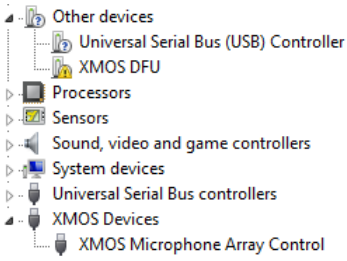

Figure 6: Once the driver has been installed

5.1 Windows Driver Issues

1. If the driver installation hangs (seen on Windows 7):

   Disable checking the driver store. **Control Panel ▶ Change device installation settings ▶ No let me choose what to do ▶ Never install driver software from Windows Update**

2. If Windows Security says it can't verify the publisher of the driver:

   Select **Install this driver software anyway**. This often results in a working driver on Windows 7.

3. There is a known issue with the Windows device driver where, on systems running Windows 10 Anniversary Update and newer, the driver will not install correctly. This is because the driver signing method used is not the new attestation signing. The issue is being reviewed. In the mean-time, disabling driver signing checking allows the driver to install.

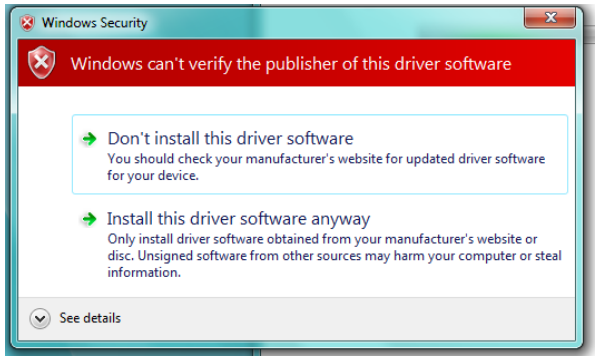

Figure 7: Windows Security

6 I²C control setup on Raspberry Pi 3

You can control device parameters via I²C, the device being an I²C slave at bus address 0x2c. The I²C control implementation requires support for clock stretching and bit-banging.

To setup the Raspberry Pi, please follow the instructions in the *VocalFusion Stereo Setup*[3].

To connect the *VocalFusion Stereo* board to the Raspberry Pi, please attach the pins in the following manner (see also Figure 8). The full pinout of the Raspberry Pi 3 header can be viewed from the link below[4].

- Attach the Raspberry Pi SDA pin 3 (BCM2) to pin 9 (X0D24) of Expansion Header J5.

[3] https://github.com/xmos/vocalfusion-stereo-setup
[4] https://pinout.xyz/pinout/i2c

- Attach the Raspberry Pi SCL pin 5 (BCM3) to pin 12 (X0D25) to Expansion Header J5.
- Ground must be shared - available on Raspberry Pi pin 6 to one of pins 2, 6, 8, 14, 15, 16 of Expansion Header J5.

Now run the command:

```
i2cdetect -y 1
```

to check that the Raspberry Pi host can see the device before running the host application. It should show a device at address 0x2c. If it does not check your wiring between the Raspberry Pi and the *VocalFusion Stereo* board and ensure you are running a version of firmware with I^{2}C control enabled (i2cctl is contained in the binary name).

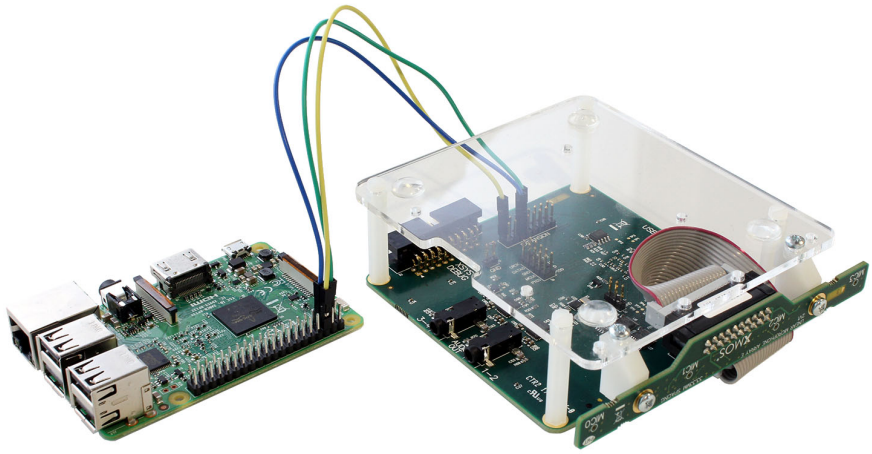

Figure 8: Raspberry Pi 3 connection for I^{2}C control

7 Using the command line utility

Depending on the transport used for control, the binaries have either the suffix _usbctl, _xscopectl or _i2cctl for USB, xSCOPE and I^{2}C control respectively. The binaries for USB and I^{2}C control can be run using `xrun`:

```
xrun <file_name>.xe
```

or `xflash`:

```
xflash  <file_name>.xe
```

while the binaries for xSCOPE control can use `xrun` only and must include some additional parameters:

```
xrun --xscope-port localhost:10101 <file_name>.xe
```

Once this program is running, you can then start the host application:

in Windows:

```
bin/vfctrl_usb.exe
```

in macOS:

```
./bin/vfctrl_usb
```

in Linux:

```
sudo ./bin/vfctrl_usb
```

The following udev rule (belongs in `/etc/udev/rules.d`) allows running the USB utility as a regular user. See customdefines.h to confirm your product ID. Most configurations are USB audio class 1.0 and have PID 0x13. Class 2.0 configurations have PID 0x12.

```
ATTR{idVendor}=="20B1", ATTR{idProduct}=="0013"
GROUP="plugdev"
```

The xSCOPE command-line utility must be run with the IP address and port as second and third command-line parameter, respectively. For example in a Windows machine using a local port 10101:

```
./bin/vfctrl_xscope.exe localhost 10101
```

7.1 Read a control parameter

If no further arguments are added, then the control utility will read the specified parameter and print the current value. For example:

```
$ ./bin/vfctrl_usb RT60
RT60:0.9
```

In case of Stereo SmartTV devices, AEC-related parameters will report two parameters, one for each AEC block, for example:

```
$ ./bin/vfctrl_usb RT60
RT60:0.9 0.9
```

7.2 Write a control parameter

If you wish to write to a parameter, then add the value to be written as the last argument to the command line:

```
$ ./bin/vfctrl_usb AGCONOFF 0
AGCONOFF:0
```

Differently from the reading operations, the AEC commands for Stereo SmartTV devices accept only one value to be written, so that the same value is applied to both of the AEC blocks.

7.3 List control parameters

If the parameter can not be found or the value requested to be written is outside of bounds, then an error message will be shown. To obtain a list of controllable parameters type:

```
$ ./bin/vfctrl_usb parameters
```

This will display available parameters and related details, for example:

```
parameter              type    max          min          r/w          info
---------              ----    ---          ---          ---          ----
AECFREEZEONOFF         int     1            0            read-write   ...
AECNORM                float   16           0.25         read-write   ...
AECPATHCHANGE          int     1            0            read-only    ...
RT60                   float   0.9          0.25         read-only    ...
...
```

Further information can be found by typing the `--help` command:

```
$ ./bin/vfctrl_usb --help
Connected to a Linear Stereo device.
Usage:

 <parameter>                Return the current value of <parameter>
 <parameter> <value>        Assign <value> to <parameter>
 --help (-h)                Display this information
 --help (-h) <parameter>    Display information for <parameter>
 parameters (-p)            Display the list of parameters
 tuning_cmds (-tc)          Display the list of tuning commands
```

7.4 Troubleshooting

- The commands `AGCTIME` and `AGCGAIN` are `read-write`, but the values read back are normally different from the ones written by the user. The value of the `AGCTIME` is used to calculate an internal AGC coefficient, so reading the parameter back returns this internal AGC coefficient. There is currently no defined meaning to the internal `AGCTIME` coefficient value, so it can be ignored. The value of

the AGCGAIN changes and essentially represents the current state of AGC. By definition it will most of the time read back a different value than the write value.

- *VocalFusion Stereo* board in I²S configuration requires running I²S clocks for I²C control to work. Configurations that are I²S master will always have running clocks. Configurations that are I²S slave need to have a master present supplying the clocks.
- If `i2cdetect -y 1` shows a device present but `vfctrl_i2c` returns the error *rdwr ioctl error -1: No such device or address*, it is a good idea to check that an I²C based *VocalFusion Stereo* device is supplied with valid clocks. Specifically a valid MCLK, BCLK and LRCLK signal from the I²S master. Control requests can only be serviced when the device is properly clocked.

8 Making changes permanent

Once the control mechanism has been used to evaluate the various settings you may want to make "permanent" changes to the default parameter settings.

This must be done in the *VocalFusion Stereo* source code, typically in the `beclear_conf.h` header filer. Further details can be found in the Software Design Guide.

9 Appendix

9.1 Resource utilization

Enabling control within the firmware will utilize additional chip resources. The amount of extra resources utilized depends on the transport chosen and the existing *VocalFusion Stereo* functionality. For example, adding control over USB to a USB Audio connected configuration may consume an additional 5KB of memory, 3 channel ends and no extra processing resource. In contrast, adding control over I²C to an I²S connected configuration may consume an additional 9KB of memory, 5 channels ends and one extra logical core needed to implement the I²C slave peripheral and control decoding.

9.2 Controllable parameters

The tables below detail the control parameters supported in *VocalFusion Stereo*. There are two functional groups: AEC (Adaptive Echo Cancellation) and BF-BP/BAP (BeamForming and Post processing).

You may also view this list by running the command utility with the `parameters` argument.

The user configurable pre-processing parameters are listed in Table 1, the configurable AEC parameters are listed in Table 2 and the configurable Beamformer and Post Processor parameters are listed in Tables 3, 4 and 5.

Table 1: Parameter descriptions for the pre-processing

parameter	type	value	definition
MIC_ATTEN	APES_INT	[-100,0]	MIC input signal attenuator in decibels (read-write) Default: 0dB
AEC_REF_ATTEN	APES_INT	[-100,0]	AEC reference signal attenuator in decibels (read-write) Default: 0dB

parameter	type	value	definition
AECFREEZEONOFF	APES_INT	[0,1]	Adaptive Echo Canceler updates inhibit (read-write). 0 = Adaptation enabled (default) 1 = Freeze adaptation, filter only
AECNORM	APES_FLOAT1	[0.25 .. 16.0]	Limit of AEC filter coefficient values (read-write). Default: 2.0
AECSILENCELEVEL	APES_FLOAT1	[0.0 .. 1.0]	Threshold for signal detection in AEC (read-write). Equivalent range: $[-\infty$ .. $0]$ dBov Default: $1.0 \times 10^{-8} \Rightarrow -80$dBov $(10\log_{10}(1.0 \times 10^{-8}) \approx -80)$
AECSILENCEMODE	APES_INT	[0,1]	AEC far-end silence detection status (read-only). 0 = false (signal detected) 1 = true (silence detected)
HPFONOFF	APES_INT	[0..3]	High-pass Filter on microphone signals (read-write). 0 = OFF 1 = ON - 70 Hz cut-off (default) 2 = ON - 125 Hz cut-off 3 = ON - 180 Hz cut-off
RT60	APES_FLOAT1	[0.250 .. 0.900]	Current RT60 estimate in seconds (read-only).
RT60ONOFF	APES_INT	[0,1]	RT60 Estimation for AES (read-write). 0 = OFF 1 = ON (default)
AECERLMAX	APES_FLOAT1	[1.0 .. 99856.0]	Maximum erl estimate (write-only). Default: 99000.0
MAX_RT60	APES_FLOAT1	[0.0 .. 0.9]	Upper limit for the RT60 estimator in seconds (write-only). Default: 0.9
AEC_REF_DELAY	TYPE_INT	[0 .. 2400]	Parametric delay for the AEC reference samples. Value is given in samples at 16kHz. Default: 0

Table 2: Parameter descriptions for the AEC module

parameter	type	value	definition
AGCDESIREDLEVEL	APES_FLOAT1	$[0.0 .. 1.0]$	Target power level of the output signal (read-write). Equivalent range: $[-\infty .. 0]$ dBov Default: $0.001 \Rightarrow -30$ dBov $(10\log_{10} 0.001 \approx -30)$
AGCGAIN	APES_FLOAT1	$[1.0 .. 1000.0]$	Current AGC gain factor (read-write). Equivalent range: $[0 .. 60]$ dB Default: $40.0 \Rightarrow 32$ dB $(20\log_{10} 40.0 \approx 32)$
AGCONOFF	APES_INT	[0,1]	Automatic Gain Control (read-write). 0 = OFF (default) 1 = ON
AGCMAXGAIN	APES_FLOAT1	$[1.0 .. 1000.0]$	Maximum AGC gain factor (read-write). Equivalent range: $[0 .. 60]$ dB Default: $40.0 \Rightarrow 32$ dB $(20\log_{10} 40.0 \approx 32)$
AGCTIME	APES_FLOAT1	$[0.1 .. 1.0]$	Ramp-up/down time-constant in seconds(read-write). Default: 0.9 s
CNIONOFF	APES_INT	[0,1]	Comfort Noise Insertion (read-write). 0 = OFF 1 = ON (default)
ECHOONOFF	APES_INT	[0,1]	Echo suppression (read-write). 0 = OFF 1 = ON (default)
FREEZEONOFF	APES_INT	[0,1]	Adaptive beamformer and postprocessor updates (read-write). 0 = Adaptation enabled (default) 1 = Freeze adaptation, filter only
GAMMA_NN	APES_FLOAT1	$[0.0 .. 3.0]$	Over-subtraction factor of non-stationary noise (read-write). min .. max attenuation (default: 1.1)
GAMMA_NS	APES_FLOAT1	$[0.0 .. 3.0]$	Over-subtraction factor of stationary noise (read-write). min .. max attenuation (default: 1.0)
MIN_NN	APES_FLOAT1	$[0.0 .. 1.0]$	Gain-floor for non-stationary noise suppression (read-write). Equivalent range: $[-\infty .. 0]$ dB Default: $0.3 \Rightarrow -10$ dB $(20\log_{10} 0.3 \approx -10)$
MIN_NS	APES_FLOAT1	$[0.0 .. 1.0]$	Gain-floor for stationary noise suppression (read-write). Equivalent range: $[-\infty .. 0]$ dB Default: $0.15 \Rightarrow -16$ dB $(20\log_{10} 0.15 \approx -16)$
NONSTATNOISEONOFF	APES_INT	[0,1]	Non-stationary noise suppression (read-write). 0 = OFF 1 = ON (default)

Table 3: Parameter descriptions for the BF and PP

parameter	type	value	definition
STATNOISEONOFF	APES_INT	[0,1]	Stationary noise suppression (read-write). 0 = OFF 1 = ON (default)
TRANSIENTONOFF	APES_INT	[0,1]	Transient echo suppression (read-write). 0 = OFF 1 = ON (default)
FSBPATHCHANGE	APES_INT	[0,1]	FSB Path Change Detection (read-only). 0 = false (no path change detected) 1 = true (path change detected)
FSBUPDATED	APES_INT	[0,1]	FSB Update Decision (read-only). 0 = false (FSB was not updated) 1 = true (FSB was updated)
GAMMA_E	APES_FLOAT1	$[0.0 .. 3.0]$	Over-subtraction factor of echo (direct and early components) (read-write). min .. max attenuation (default: 1.0)
GAMMA_ENL	APES_FLOAT1	$[0.0 .. 5.0]$	Over-subtraction factor of non-linear echo (read-write). min .. max attenuation (default: 1.0)
GAMMA_ETAIL	APES_FLOAT1	$[0.0 .. 3.0]$	Over-subtraction factor of echo (tail components) (read-write). min .. max attenuation (default: 1.0)
NLAEC_MODE	APES_INT	[0..2]	Non-Linear AEC training mode (read-write). 0 = OFF (default) 1 = ON - phase 1 2 = ON - phase 2
NLATTENONOFF	APES_INT	[0,1]	Non-Linear echo attenuation (read-write). 0 = OFF (default) 1 = ON
VOICEACTIVITY	APES_INT	[0,1]	Signal energy exceeded a threshold (read-only). 0 = false (no voice activity) 1 = true (voice activity)
BEAMANGLE	APES_FLOAT1	$[-1.0 .. 1.0]$	Center of the beam for desired speech sources (read-write). Equivalent range: $[-90° .. 90°]$ Default: $0.0 \Rightarrow 0°$ $(\sin^{-1} 0.0 = 0°)$
BEAMWIDTH	APES_FLOAT1	$[0.2 .. 1.0]$	Width of the beam for desired speech sources (read-write). Equivalent range:$[23° .. 180°]$ Default: $0.5 \Rightarrow 60°$ $(2 \cdot \sin^{-1} 0.5 = 60°)$
DOAANGLE	APES_INT	[0 .. 180]	DOA angle; current value, orientation depends on build configuration (read-only).
FSBFREEZEONOFF	APES_INT	[0,1]	Adaptive beamformer updates (read-write). 0 = Adaptation enabled (default) 1 = Freeze adaptation, filter only

Table 4: Parameter descriptions for the BF and PP (continued)

parameter	type	value	definition
SPTHRESH	APES_FLOAT1	[0.0 .. 1.0]	Set parameter value for DNNS (read-write). Default: 0.0065
SR_ABSQFLOOR	APES_FLOAT	[0.0 .. 1000.0]	Absolute noise floor for voice activity detection (read-write). Equivalent range: $[-\infty$ .. $60]$ dB Default: $0.0 \Rightarrow -\infty$ dB $(20\log_{10} 0.0 = -\infty)$
SR_GAMMA_NN	APES_FLOAT1	[0.0 .. 3.0]	Gain-floor for non-stationary noise suppression (read-write). min .. max attenuation (default: 1.1)
SR_GAMMA_NS	APES_FLOAT1	[0.0 .. 3.0]	Over-subtraction factor of stationary noise (read-write). min .. max attenuation (default: 1.0)
SR_GAMMA_VAD	APES_FLOAT1	[0.0 .. 1000.0]	Threshold for voice activity detection (read-write). Equivalent range: $[-\infty$ .. $60]$ dB Default: $15 \Rightarrow 23.5$ dB $(20\log_{10} 15 \approx 23.5)$
SR_MIN_NN	APES_FLOAT1	[0.0 .. 1.0]	Gain-floor for non-stationary noise suppression (read-write). Equivalent range: $[-\infty$ .. $0]$ dB Default: $0.3 \Rightarrow -10$ dB $(20\log_{10} 0.3 \approx -10)$
SR_MIN_NS	APES_FLOAT1	[0.0 .. 1.0]	Gain-floor for stationary noise suppression (read-write). Equivalent range $[-\infty$ .. $0]$ dB Default: $0.15 \Rightarrow -16$ dB $(20\log_{10} 0.15 \approx -16)$
SR_NONSTATNOISEONOFF	APES_INT	[0,1]	Non-stationary noise suppression for ASR (read-write). 0 = OFF 1 = ON (default)
SR_STATNOISEONOFF	APES_INT	[0,1]	Stationary noise suppression for ASR (read-write). 0 = OFF 1 = ON (default)
XNLTRAINONOFF	APES_INT	[0,1]	Non-linear matrix training (read-write). 0 = OFF(default) 1 = ON

Table 5: Parameter descriptions for the BF and PP (continued)

9.2.1 References

xCORE-200: The XMOS XS2 Architecture

https://www.xmos.com/published/xs2-isa-specification

XMOS Tools User Guide

http://www.xmos.com/published/xtimecomposer-user-guide

XMOS xCORE Programming Guide

http://www.xmos.com/published/xmos-programming-guide