XVF3610 Voice Processor - Product Description

Release: 5.7.2

Publication Date: 2023/01/23

Table of Contents

The XMOS VocalFusion® XVF3610 family of voice processors uses microphone array processing to capture clear, high-quality voice from anywhere in the room. XVF3610 processors use highly optimised digital signal processing algorithms implementing 'barge-in', point noise and ambient noise reduction to increase the Signal-to-Noise Ratio (SNR) achieving a reliable voice interface whatever the environment.

The XVF3610 processor is designed for seamless integration into consumer electronic products requiring voice interfaces for Automatic Speech Recognition (ASR), communications or conferencing. In addition to its class-leading voice processing, the XVF3610 voice processor provides a comprehensive set of interfaces and configuration options to simplify the integration of a voice interface into a wide range of system architectures. This includes specific features required in TV and set-top box applications, including audio switching and digital inputs and outputs that support switches and LED indicators.

Note: This document is valid for XVF3610 version 5.7

1 XVF3610 Product Variants

The XVF3610 voice processor executes a firmware image that is either read from a flash memory device or loaded by a host processor. The Device Firmware Upgrade (DFU) function of the processor allows in field upgrade ensuring all products can benefit from the latest releases. While the voice processor is running, this configuration can be modified by the host system over the XVF3610 control interface. The control interface also allows the host system to control peripheral devices and obtain status information from the device and its digital inputs.

Three variants of the XVF3610 are available and they have been optimised for different application use cases. These three variants require different firmware to be loaded onto the device.

The application use cases are described in more detail in the following sections:

- XVF3610-INT (**INT**-egrated): this variant is used as a Voice interface integrated into the product and it has the following features:
  - Far-field voice interface
  - Audio interface: I^2S (Slave)
  - Control interface: I^2C (Slave)
  - Device Firmware Upgrade: I^2C (Slave)
- XVF3610-UA (**U**-SB **A**-ccessory): this variant is used as a USB plug-in voice accessory and integrated products using USB and it has the following features:
  - Far-field voice interface
  - Audio interfaces: USB UAC1.0 for far-field voice and reference audio (and optionally I^2S Master),
  - Control interface: USB2.0 Full Speed
  - Device Firmware Upgrade: USB
- XVF3610-UA-HYBRID (**U**-SB **A**-ccessory **HYBRID**): this variant is used as a USB plug-in voice accessory and integrated products using USB and it has the following features:
  - Far-field voice interface
  - Audio interfaces: USB UAC1.0 for far-field voice and I^2S Master for reference audio,
  - Control interface: USB2.0 Full Speed
  - Device Firmware Upgrade: USB

The bit width of the USB audio interface is configurable for both the input and output channels. The default value is 16 bits, but it can be set to 24 and 32 as well. The I^2S interface is always 32-bit wide. In case of the XVF3610-UA, the bit width of the optional I^2S Master interface is 32 bits, but if the width of USB channel from the device to the host is lower than 32, the I^2S samples are right padded with zeros.

The sample rates of both the USB and I^2S interfaces can be set to 16 or 48kHz, the default value is 48kHz.

2 System Architecture

2.1 System Block Diagrams

2.1.1 XVF3610-INT Configuration

The XVF3610-INT device has been optimized for integration on a system board. A standard I²C interface is provided to enable the main processor on the system board to configure and monitor the XVF3610-INT. The processed voice signal is output over an I²S bus to the host system and the XVF3610 receives its I²S audio reference signal for the Acoustic Echo Cancellation function.

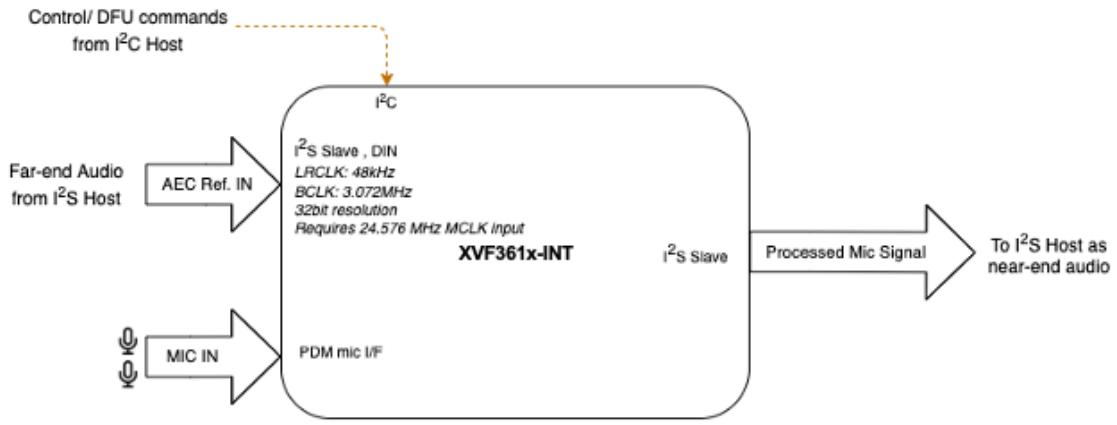

Fig. 2.1: XVF3610-INT Integrated configuration

2.1.2 XVF3610-UA and XVF3610-UA-HYBRID Configurations

The XVF3610-UA and XVF3610-UA-HYBRID devices replace the I²C interface of the XVF3610-INT with a USB2.0 compliant PHY which supports a UAC1.0 audio device for the processed audio output. In the XVF3610-UA device the reference signal input is sent over the same USB interface, while XVF3610-UA-HYBRID uses an I²S master interface. In addition, the USB device supports a standard USB Endpoint 0 for device control and a standard USB HID for status events. On both devices, an I²S master interface is also available on the device to output an audio signal to an external audio device.

The following block diagram illustrates the typical configuration for the XVF3610-UA device:

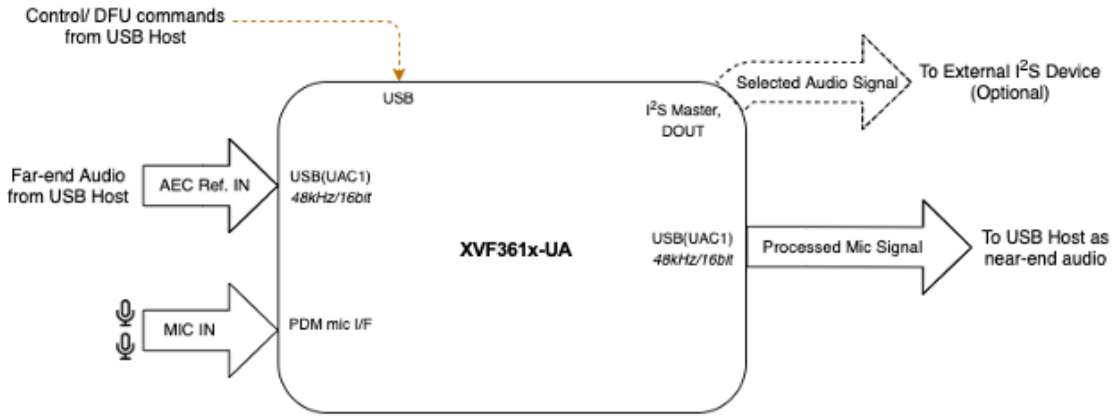

Fig. 2.2: XVF3610-UA Configuration for USB-only use case

The following block diagram illustrates the typical configuration for the XVF3610-UA-HYBRID device:

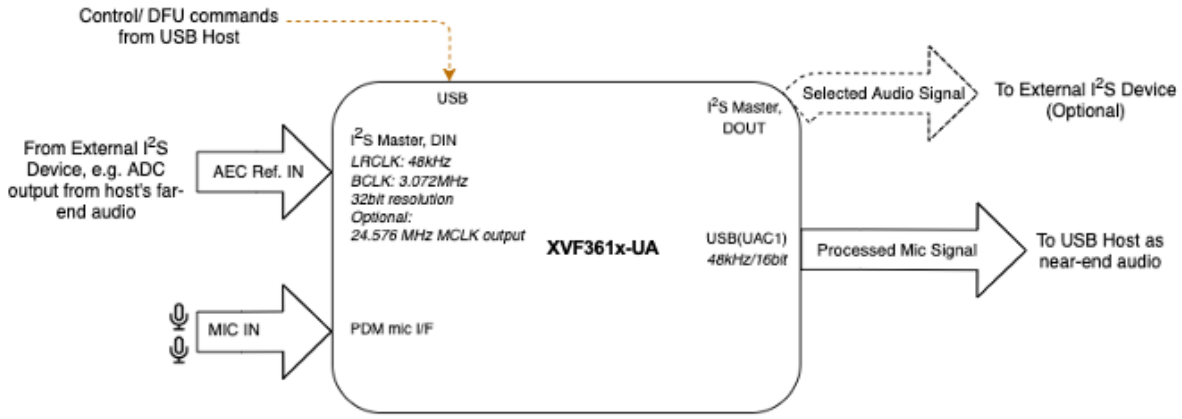

Fig. 2.3: XVF3610-UA-HYBRID Configuration using USB and I²S audio

2.2 Device firmware and configuration

The operation of the XVF3610 device is controlled through a firmware image that is loaded onto the device when it is powered up.

Two modes of operation are supported:

1. The firmware image can either be stored in a QSPI Flash device which is read by the XVF3610 processor automatically,
2. The firmware image is downloaded to the XVF3610 processor over the SPI interface by the host processor on the system board.

Selection of the boot mode is made via setting the QSPI_D1/BOOTSEL pin on the device as described in the boot modes section of the XVF3610 datasheet.

The firmware image configures the XVF3610 into a standard, default operational mode. This mode can be modified at startup via a set of configuration parameters that are stored in the flash device along with the firmware in the XVF3610 Data Partition. These commands can be used to reconfigure the device during startup, and also initialise other devices attached to it.

If the device firmware is downloaded from the host, then the data partition is not required and the device is configured directly over the control interface.

3 Voice Processing Pipeline

3.1 Overview

The core of the XVF3610 voice processor is a high-performance audio processing pipeline that takes its input from a pair of microphones and executes a series of signal processing algorithms to extract a voice signal from a complex soundscape. The audio pipeline can accept a reference signal from a host system which is used to perform Acoustic Echo Cancellation (AEC) to remove audio being played by the host. The audio pipeline provides two different output channels - one that is optimized for Automatic Speech Recognition systems and the other for voice communications.

Flexible audio signal routing infrastructure and a range of digital inputs and outputs enable the XVF3610 to be integrated into a wide range of system configurations, that can be configured at start up and during operation through a set of control registers.

In addition, the XVF3610-UA and project|-UA-HYBRID variants support a standard USB PHY interface which supports a UAC audio device and device control over USB. The following sections describe the voice pipeline and the surrounding infrastructure in more detail.

3.2 Audio Processing Pipeline

The audio processing pipeline is common to both the XVF3610-UA and XVF3610-INT firmware variants. The signal processing chain is described below, with individual blocks and usage described in more detail in subsequent sections.

The XVF3610 audio processing pipeline takes inputs from a pair of MEMS Pulse Density Modulation (PDM) microphones and uses advanced signal processing to create audio streams suitable for use in Automatic Speech Recognition (ASR) and voice communication applications. The pipeline enhances the captured audio stream using a set of complementary signal enhancement and noise reduction processes.

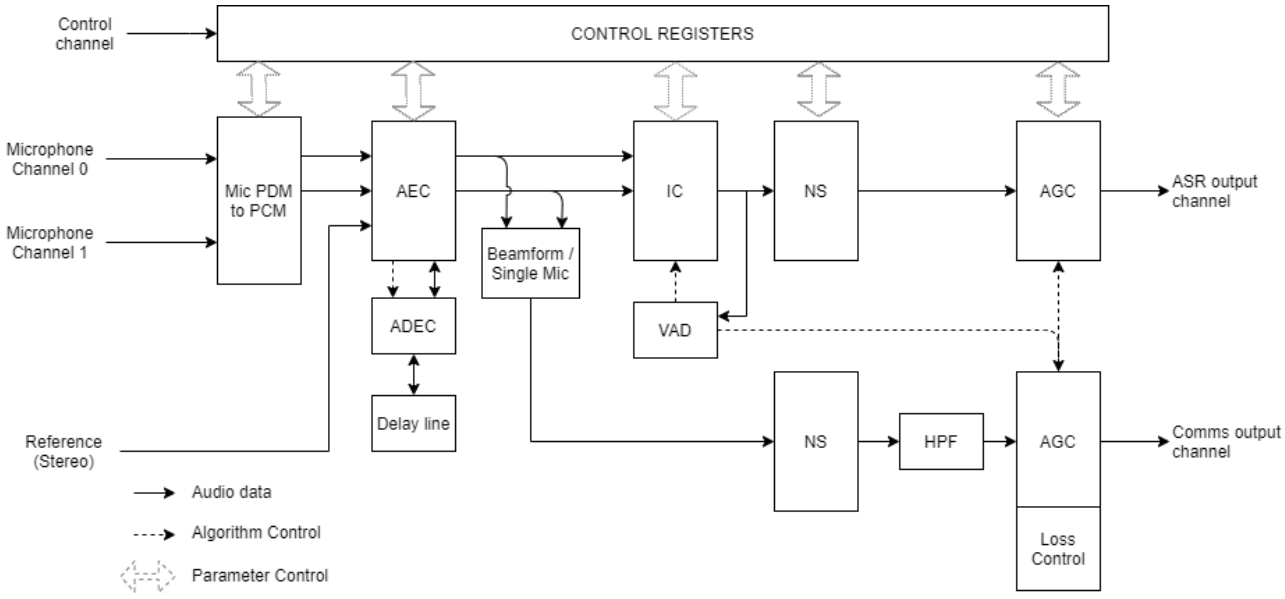

Fig. 3.1: The XVF3610 audio pipeline

The pipeline takes its input from a pair of low-cost PDM microphones and converts this signal to PCM for further processing:

- **Acoustic Echo Cancellation (AEC):** Continuously modelling the room acoustics allows the AEC to remove audio being played into the room by the product of which the XVF3610 is a component of. A reference copy of the audio is provided to the AEC in order for it to accurately estimate the echo.
- **Automatic Delay Estimation & Control (ADEC):** Automatically monitors and compensates for the delay between the reference audio and the echo received by the microphone.

Following echo cancellation, the ASR and communications paths diverge to permit parameter tuning appropriate for the individual audio output use cases.

- **Interference Cancellation (IC):** Suppresses static noise from point sources such as cooker hoods, washing machines, or radios for which there is no reference audio signal available.
- **Voice Activity Detection (VAD):** Controls adaption the IC and AGC to optimise output for near-end speech.
- **Noise Suppression (NS):** Suppresses diffuse noise from sources whose frequency characteristics do not change rapidly over time (i.e., diffuse stationary noise).
- **Automatic Gain Control (AGC):** Controls the audio output level via separate AGC channels for Automatic Speech Recognition (ASR) and communications output. The VAD is used to prevent gain changes during speech to improve speech recognition performance.

The pipeline has been designed to minimise the need to tune and modify these functions. However, if required for specific use cases, these later sections of this document provide details of the relevant parameters and processes.

3.3 ASR and Communication Processing

The audio pipeline discussed above produces two separate audio streams, one specifically tuned for integration with keyword and ASR services and the other designed for conferencing and communication applications. Both

processed audio streams are available simultaneously on the left and right channels of the USB and I²S audio outputs. The default configuration is as follows:

Table 3.1: Default channel mapping (both USB and I²S)

Channel	Default
[0] Left	ASR Automatic Speech Recognition
[1] Right	Communications

In situations where an ASR is used to invoke a call it may be necessary to continually monitor the ASR channel for a 'end call' intent. The parallel output of both ASR and Communications processed streams allow the combination of high-quality calling audio with the tuned ASR capability.

The IO_MAP configuration parameter (see *Signal flow and processing* section) allows users to also configure both channels to be ASR or Communications if required.

3.4 XVF3610-INT - For integrated voice interface applications

The XVF3610-INT product embeds the core audio processing pipeline in an audio infrastructure that supports rate conversion, filtering and signal routing. This infrastructure is controllable by the host system via a set of control registers. In addition, the XVF3610-INT provides a set of peripheral interfaces to the host system to other devices, e.g. digital inputs, LEDs, SPI peripherals etc.

The peripheral interfaces supported include an interface to an optional QSPI Flash device containing the XVF3610 firmware and configuration information that is loaded by the processor on startup.

The system architecture of the XVF3610-INT is shown below:

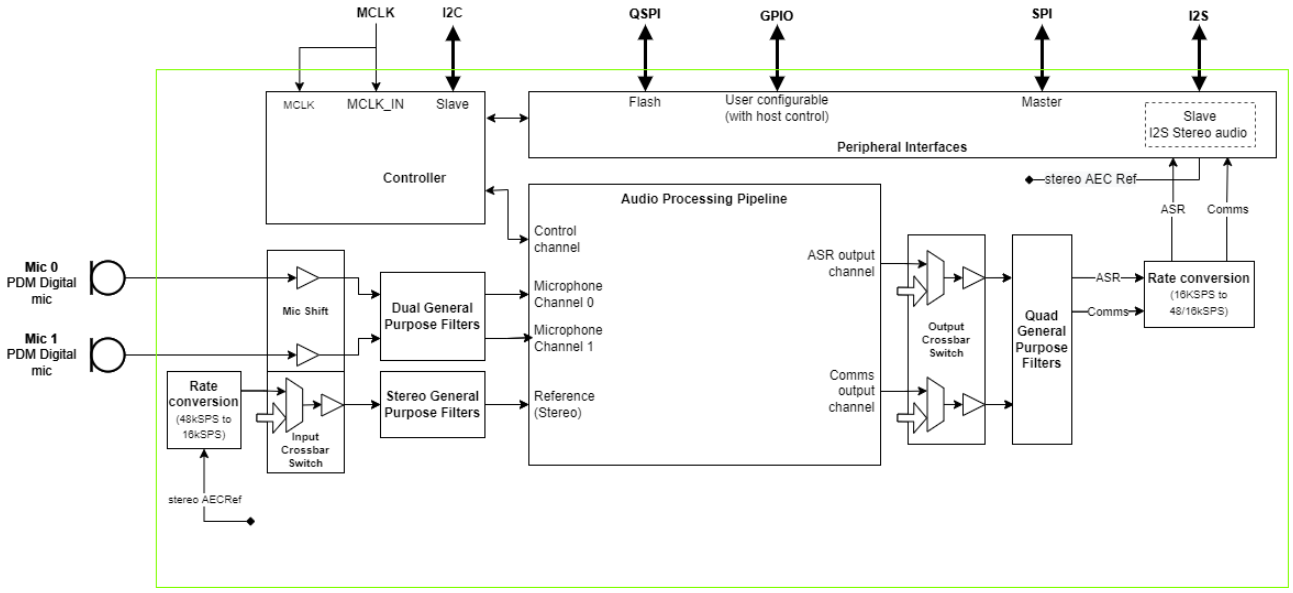

Fig. 3.2: XVF3610-INT System architecture

3.5 XVF3610-UA and XVF3610-UA-HYBRID - For USB accessory voice interface applications

The XVF3610-UA variant includes the same audio infrastructure as the XVF3610-INT, but it includes a USB interface that implements a UAC1.0 audio device to interface to the host system. The USB interface also supports an Endpoint 0 control channel, and a USB HID to signal input events to the host.

The system architecture of the XVF3610-UA is shown below:

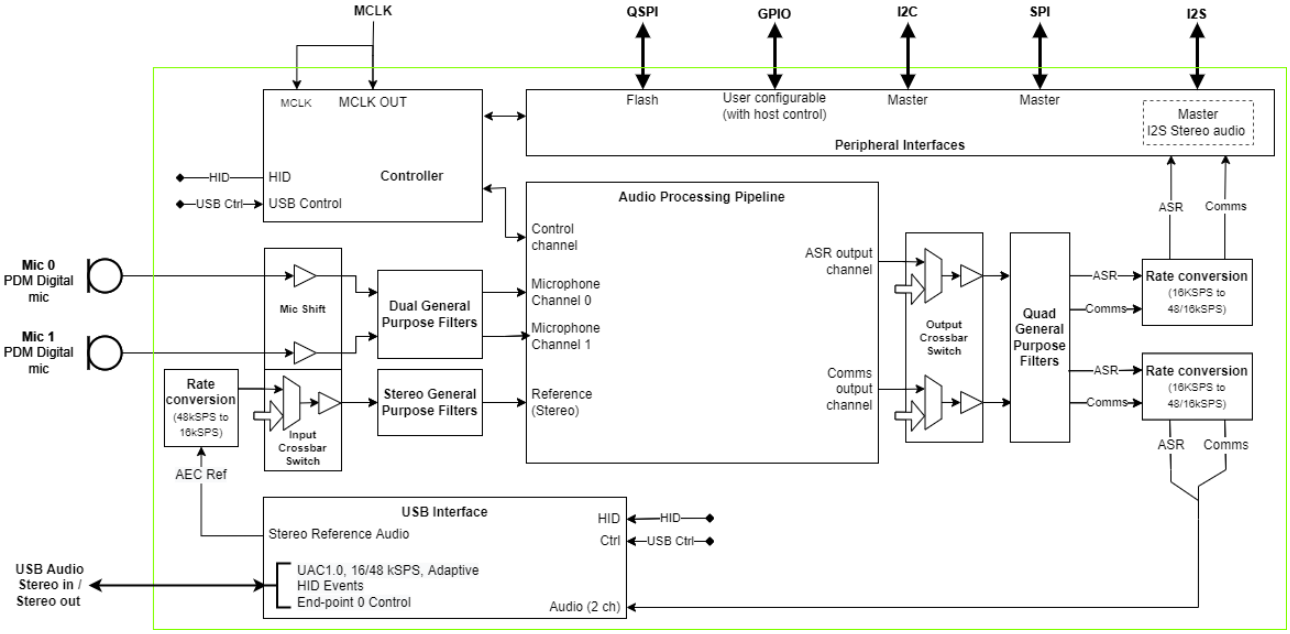

Fig. 3.3: XVF3610-UA System architecture

The XVF3610-UA-HYBRID has the same characteristics of the XVF3610-UA variant, but the reference signal is delivered via I^2S rather than USB.

The system architecture of the XVF3610-UA-HYBRID is shown below:

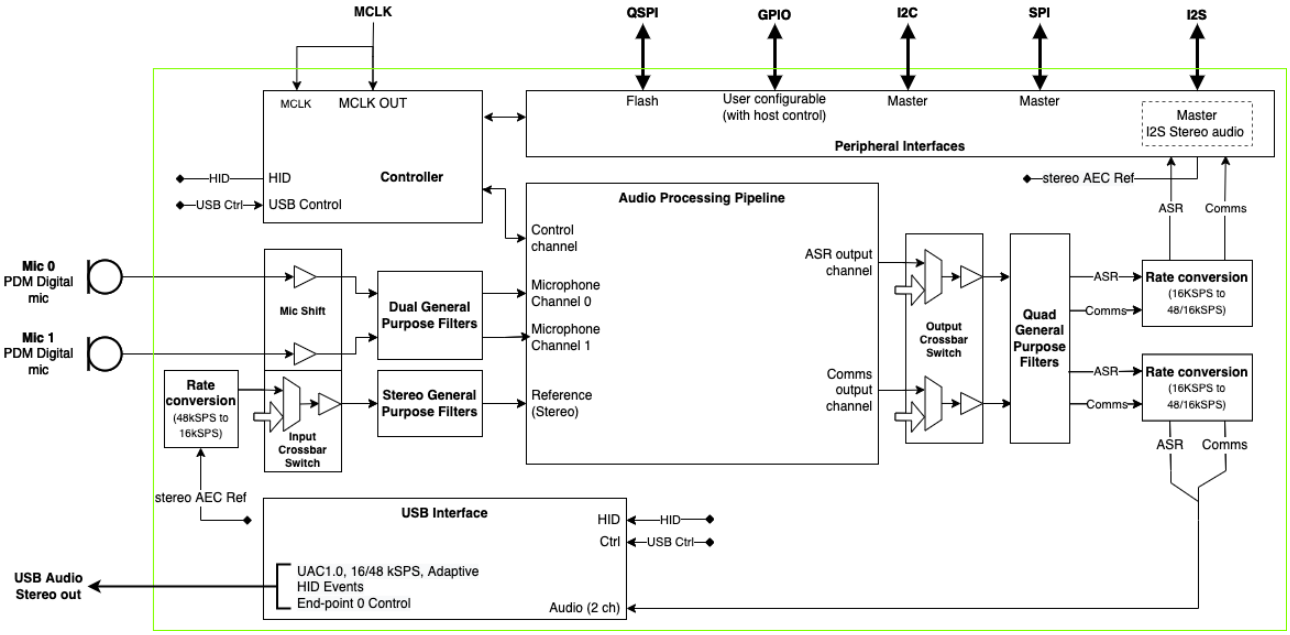

Fig. 3.4: XVF3610-UA-HYBRID System architecture

4 Firmware release package

There are two release packages available for the XVF3610, one for the XVF3610-UA / XVF3610-UA-HYBRID and one for the XVF3610-INT.

Release packages and firmware builds are identified via a version number, which follows the standard semantic version specification. The version number format is X.Y.Z, eg 5.2.0, and these numbers have the following meaning.

Table 4.1: Firmware version number structure

Digit	Name	Meaning
X	Major version number	Significant release of the firmware. The control interface may not be backwards compatible with earlier versions
Y	Minor version number	New features added. The control interface is backwards compatible with earlier host applications
Z	Patch version number	Bug fixes for incorrect functionality only. No change to host interface

The release version is contained in the file name of firmware file distribution and can also be read via the control interface using the `GET_VERSION` command.

Each package consists of several directories and files containing released firmware binaries, data-partition tools, host binaries and host source code. A simplified directory structure is shown below.

```
├── bin
│   ├── dfu
│   │   └── <config>
│   ├── spi_boot
│   │   └── <config>
│   └── xflash
│       └── <config>
├── data-partition
│   └── input
└── host
    ├── Linux
    │   └── bin
    ├── MAC
    │   └── bin
    ├── Pi
    │   ├── bin
    │   └── scripts
    ├── Win32
    │   └── bin
    └── src
        ├── dfu
        ├── dpgen
        └── vfctrl
```

Note: <config> in the structure above refers to the configurations supported by the firmware release package. See the *System Architecture* section for further details on these configurations.

4.1 "bin" Directory

Note: Structure has changed in version 5.6 - firmware and corresponding default data partition images are now in the same folder

This directory contains the released firmware for the XVF3610. There are different versions of the firmware which are stored in separate subdirectories:

xflash: Firmware intended for loading from an external flash device. This process is described in *Updating Firmware* below.

dfu: An image that can be used to upgrade an operational system using the Device Firmware Update process.

spi_boot: Firmware for loading from an external host over SPI (XVF3610 is the slave). Please refer to the SPI Slave boot section of the datasheet for connections to the external boot source.

See the ../user_guide/booting/index section of the User Guide for further information on of the process of installing and updating the firmware for the XVF3610.

4.2 "data-partition" Directory

The data partition contains configuration data for the XVF3610 firmware, implemented as a set of commands that are run at boot time. The data partition is created using input command source files and a set of tools which are described in the Data Partition section of this document. The contents of the data-partition directory are as follows:

The root directory contains default data partition image source files (int.json or ua.json) as well as the generic flash device specification 16mbit_12.5mhz_sector_4kb.spispec, data partition generation scripts and short instructions about how to generate data partition binary files.

For convenience, the pre-generated data partition binary files generated from these default data partition image source files are contained in the "bin" directory (see above). These files are suitable for direct programming into the external flash along with the firmware, should the default settings be suitable.

The `input` subdirectory contains short command sequences which are referenced by the data partition image source file when the data partition binary file is generated.

In addition, an output directory is created during the running of the data partition generation script which contains the newly generated data partition binary file.

4.3 "host" Directory

This directory contains files and utilities relating to the host. The various host utilities that perform parameter control, device firmware update (DFU) and data partition generation are provided pre-compiled for Linux (ARM and x86), Windows and MacOS platforms. These binaries can be found in the Linux, Pi, Mac and Win32 directories

along with an additional script for the Raspberry Pi release called send_image_from_rpi.py which provides an example of sending an SPI boot image from the host.

The root of the host directory also contains scripts for unpacking packed signals which can be captured using the controls described in the signal routing section of this document.

Instructions for building the host utilities from the source are also provided in the same directory. The source files for the host utilities are contained in the `src` sub-directory allowing building, modification or integration into other projects.

Within this directory there are three further sub-directories `dfu`, `dpgen` and `vfctrl` which contain the source files (and dependent libraries) for the DFU, data partition generator and parameter control utilities.

5 Required Tools

In order to update the firmware, modify and regenerate Data Partitions and rebuild the host utilities the following tools are required.

5.1 XTC Tools

The XMOS XTC Tools contains a comprehensive suite of tools for compilation, debug and programming of XMOS devices. It is available to download from https://www.xmos.ai/software-tools.

Note: At the time of writing v15.1 of XTC Tools is recommend for programming XVF3610 firmware. More recent versions may be available, but unless specified on the xmos.ai website they will not have been tested and verified for operation with XVF3610.

Further information about the full tool suite, including installation instructions for different platforms is available here in the XTC Tools user guide, available from https://www.xmos.ai/file/tools-user-guide.

5.2 Updating Firmware

The XVF3610 Voice Processor is provided in three pre-compiled builds (UA, UA-HYBRID and INT) and as such only requires the usage of one of the XTC Tools programming tools, specifically `xflash`. This operates as a command-line application, to create the boot image, and if using flash, program the boot image to the attached device.

An XTAG debugger must be connected to the XVF3610 for flash programming operations. Refer to the Development Kit User Guide for information on using XTAG connections to XVF3610 development kits.

The basic form of the `xflash` command for flash image creation and programming with a data partition is as follows (note multiple lines have been used for clarity, but command should be executed on single line).

```
xflash --boot-partition-size 0x100000 --factory [Application executable (.xe)] --data [Data␣
↪partition description (.bin)]
```

Note: Boot over SPI from a host processor uses a specific image which is supplied in the release package. No data partition is included as configuration command are assumed to be supplied by the host controller used.

- Application executable (.xe) The .xe file is a boot image provided with a VocalFusion® release package in one of the supported configurations (UA, UA-HYBRID or INT product variants).
- Data partition description (.bin) The .bin file is a data partition description either supplied in the release package (UA, UA-HYBRID or INT) or customised as described later in this guide.

Warning: Running XTC Tools on macOS Catalina or above may trigger a security alert. The process to resolve this is detailed here : https://www.xmos.ai/file/running-XTimecomposer-on-macos-catalina/.

5.3 Python 3

Some operations, such as running the SPI boot example on the Raspberry Pi, require the use of Python 3 (v3.7 onward is recommended). Python can be downloaded from http://python.org/downloads.

5.4 Host build tools

The release package contains executable versions of the host tools. The source code is also indluded to provide the option for users to customise these tools. In order to build the host utilities, the use of a platform-specific compiler is required.

The host utilities are built with the *x86 Native Tools Command Prompt for VS* which is installed as part of the *Build Tools for Visual Studio*. This can be downloaded from Microsoft website (at the time of writing latest versions available here: https://visualstudio.microsoft.com/downloads. It is important to ensure that the optional *C++ CMake tools for Windows* are included when setting up the installation.

For cross-platform support *vfctrl_usb* uses *libusb*. This requires the installation of a driver for use on a Windows host. Driver installation should be done using a third-party installation tool like Zadig (https://zadig.akeo.ie/).

Depending on the distribution and version of Linux used, the following packages may need to be installed:

```
sudo apt-get install -y build-essential
sudo apt-get install -y pkg-config
sudo apt-get install -y libusb-1.0-0-dev
```

The XCode Command Line tools are required to build in on macOS. The following command can be used to install the tools.

```
xcode-select --install
```

6 Configuration and control

The XVF3610 family of processors are designed to provide a far-field voice interface to a host system or processor in speech recognition and communication applications, either closely integrated to the main processor or as a USB accessory. As such the XVF3610 provides boot mechanisms from either an external QSPI flash or by the host processor over SPI.

To facilitate control in both boot configurations and to allow the specification of the default behaviour, the XVF3610 implements two mechanisms for control and parameterisation. The first is the Control Interface which is a direct connection between the host and the XVF3610 and is operational at runtime. The second is the Data Partition which is held in flash and contains configuration data to parameterise the XVF3610 on boot up. Both mechanisms be used by a host application to control the behaviour of the device.

7 Command-line interface (vfctrl)

To allow command-line access to the control interface on the XVF3610 processor, the **vfctrl** (**V**ocal**F**usion **C**on**tr**o**l**) utility is provided as part of the release package.

Two versions of this utility are provided for control of the device (a third is used internally by the Data Partition generation process).

Table 7.1: vfctrl versions and platforms

Version	Function	Host platforms supported
vfctrl_usb	Control of XVF3610-UA and XVF3610-UA-HYBRID over a USB interface	Windows - MacOS - Linux - Raspberry Pi OS
vfctrl_i2c	Control of XVF3610-INT over I^2C interface	Raspberry Pi OS

Source code for the utility is also provided for compilation for other host devices if required.

Note: For cross-platform support vfctrl_usb uses libusb. While this is natively supported in macOS and most Linux distributions, it requires the installation of a driver for use on a Windows host. Driver installation should be done using a third-party installation tool like Zadig (https://zadig.akeo.ie/).

8 vfctrl syntax

The general syntax of the command line tool, when used for device control, is as follows:

```
vfctrl_usb <COMMAND> [ arg 1] [arg 2]....[arg N] ['# Comment']
```

The <COMMAND> is required and is used to control the parameters of the device. Each command either reads or writes parameters to the XVF3610 device. Commands that read parameters begin with GET_. Commands that write parameters begin with SET_.

The available commands are described in detail in the command reference section of the user guide, and a summary table of all the parameters is provided.

Following the <COMMAND> there are a number of optional arguments [arg 1]..[arg N] which depend on the specific parameter. These are detailed in the command tables later in the document.

If the <COMMAND> is a GET_ command, the output of the operation is printed to the terminal as in the example below:

```
vfctrl_usb GET_GPI
GET_GPI: 13
```

The number and type of arguments depend on the command and these are detailed in the command tables. Arguments are integer numbers separated by a space. For setting some parameters that require floating-point data, the numbers have to be first converted to a Q format and then transferred as integers.

The specification of the Q format for representing floating-point numbers is given in Appendix H.

A secondary form of vfctrl is also available which provides information for developers:

```
vfctrl [options]
```

Where [options] can be:

```
-h, --help : Print help menu

-H, --help-params: Print help menu and the list of all available commands

-d, --dump-params : Print the values of the parameters configured in the device

-n, --no-check-version : Do not check version of firmware image

-f, --cmd-list <filename> : Execute the commands in the given <filename>
```

9 Configuration via Control interface

The XVF3610 Voice Processor contains parameters which can be read and written by the host processor at run time. For information about writing parameters at boot time for initial configuration, please see *Configuration via Data Partition*.

The XVF3610 firmware is provided as pre-compiled builds, UA, UA-HYBRID and INT; the first two builds provide a parameter control mechanism over USB Endpoint 0, whereas the last one uses I^2C as control interface.

Device functions have controllable parameters for the audio pipeline, GPIO, sample rate settings, audio muxing, timing and general device setup and adjustment. Commands support either read using the `GET_` prefix or write using the `SET_` prefix. Controllable parameters may either be readable and writeable, read-only or write-only. Various data types are supported including signed/unsigned integer of either 8b or 32b, fixed point signed/unsigned and floating-point.

In addition, the UA build includes volume controls for input (processed microphone from XVF3610) and output (far-end reference signal). These are USB Audio Class 1.0 compliant controls and are accessed via the host OS audio control panel instead of the XVF3610 control interface. The volumes are initialised to 100% (0dB attenuation) on device power up, which is the recommended setting. The UA-HYBRID build supports similar volume controls for input audio only.

Ensure that the USB Audio input and output volume controls on the host are set to 100% (no attenuation) to ensure proper operation of the device. Some host OS (eg. Windows) may store volume setting in between device connections.

For a comprehensive list of parameters, their data types and an understanding of their function within the device please consult the User Guide section relevant to the function of interest, or Appendix A which summarises all the commands. The full list of commands can be obtained through the use of the -H or –help-params option of the control utility.

```
vfctrl --help-params
```

This dumps a list of commands to the console along with a brief description of the function of each command. The remainder of this section will cover the generic operation of the control interface.

9.1 Control operation

The control interface works by sending a message from the host to the control process within the XVF3610 device. The time required to execute commands can vary, but most will respond within 30ms. Since the commands are fully acknowledged, by design, the control utility blocks until completion. This interface is designed to allow real-time tuning and adjustment but may stall due to bus access or data retrieval.

The control interface consists of two parts a host side application and the device application. These are briefly summarised below.

9.2 Host Application

The example host applications, found in the /host directory in the Release Package, are command-line utilities that accept text commands and, in the case of a read, provides a text response containing the read parameter(s).

Full acknowledgement is included in the protocol and an error is returned in the case of the command not being executed properly or handled correctly by the device.

Example host source code and makefiles for are provided in the release package for x86 Linux, ARM Linux (Raspberry Pi), Windows and Mac platforms along with pre-compiled executables to allow fast evaluation and integration. For more information refer to the *Building the host utilities from source code* section.

9.3 Device Application

The device is always ready to receive commands. The device includes command buffering and an asynchronous mechanism which means that Endpoint 0, NACKing for USB or clock stretching for I^2C is not required. This simplifies the host requirements particularly in the cases where clock stretching is not supported by the host I^2C peripheral.

10 Configuration via Data Partition

The XVF3610 device flash firmware configuration comprises a Boot image and a Data Partition.

- The **Boot image** in the form of an .xe archive is the executable code. It is provided as part of the XVF3610-UA or XVF3610-INT Release Package. This configures the underlying operation of the device.
- The **Data Partition** configures a running Boot image instance at startup with a set of commands which are customisable for the specific application. This contains any command that can be issued at run-time via USB or I^2C, plus some more that are boot-time only. Pre-configured Data Partitions are supplied in the release packages for default operation.

This combination of Boot image and Data Partition allow the functionality of the processor to be configured and defined without requiring any modification or recompilation of base firmware. The commands discussed in subsequent sections can be stored in the Data Partition, for execution at startup redefining the default operation of the device.

11 Development kits

The XVF3610 firmware can execute on the XMOS xcore.ai voice reference platform.

Table 11.1: Development kit variants and firmware pre-loaded

Development kits	Firmware loaded	Notes
XK-VOICE-L71 Voice Reference Design	5.1.1	Should be updated to 5.7.2

Note: Users are recommended to check the website for the latest firmware update, and to follow the instructions to update the stored firmware before use.

The XMOS xcore.ai voice reference platform and host code enable users to build a complete Amazon Alexa end-point with the addition of a Raspberry PI (not supplied). The XVF3610-INT connects to the Raspberry PI using I^2S for audio, and I^2C for control.

Detailed instructions for setting up and using the XVF3610 development kit can be found the Getting Started Guide.