Processor Audio SDK Setup Guide****

Rev. 0.30

08 Dec 2016

|  |
| --- |
| **Revision History** |
| **Version** | **Date** | **Description of changes** | **Author(s)** |
| 0.10 | 23-Aug-16 | Initial version | Frank Livingston |
| 0.20 | 30-Aug-16 | Added gmake clean to UART LLD build instructions | Frank Livingston |
| 0.30 | 08-Dec-16 | Updated for AR2 | Frank Livingston |

**Table of Contents**

[1 Introduction 5](#_Toc468992244)

[2 Required Hardware 5](#_Toc468992245)

[3 Software Installation 5](#_Toc468992246)

[3.1 Code Composer Studio (CCS) 5](#_Toc468992247)

[3.2 Processor SDK-RTOS 6](#_Toc468992248)

[3.3 Open Source Package 6](#_Toc468992249)

[3.4 Firmware Deliverable Package 7](#_Toc468992250)

[3.5 Dolby IP Packages 7](#_Toc468992251)

[3.6 CCS Update 7](#_Toc468992252)

[3.7 PASDK OS Dev Build and Alpha Communication Tools 8](#_Toc468992253)

[4 Emulator Firmware Update 8](#_Toc468992254)

[5 Software Build 9](#_Toc468992255)

[5.1 PASDK OS Dev Libraries 9](#_Toc468992256)

[5.2 UART, SPI, and I2C Low Level Drivers (LLDs) 10](#_Toc468992257)

[5.3 K2G Platform Library 10](#_Toc468992258)

[5.4 DSP Application 10](#_Toc468992259)

[5.5 ARM Application 11](#_Toc468992260)

[6 Application Execution and Test 11](#_Toc468992261)

[6.1 I/O Configuration 11](#_Toc468992262)

[6.2 Connect EVM 12](#_Toc468992263)

[6.2.1 ADC 8-Ch Input and DAC 8-Ch Output 12](#_Toc468992264)

[6.2.2 S/PDIF Input and DAC 8-Ch Output 12](#_Toc468992265)

[6.2.3 HDMI Input and DAC Output 13](#_Toc468992266)

[6.3 Load and Execute Code 13](#_Toc468992267)

[6.4 PyAlpha Python Scripts for Alpha Communication 14](#_Toc468992268)

[6.5 Alpha Commands for AE0 Configuration/Status 14](#_Toc468992269)

[7 Known Issues 15](#_Toc468992270)

# Introduction

This document provides installation and setup instructions for the Alpha-2 Release (AR2) of Processor Audio SDK (PASDK) 1.0 for K2G provided by Texas Instruments, Inc. (TI). AR2 incorporates several packages: (1) the Open Source (OS) package; (2) the Firmware Deliverable (FD) package; and (3) the Dolby Atmos IP packages. The AR2 features provided by these packages are detailed below.

1. OS package
	* Audio input drivers: HDMI input, and 8-channel ADC or S/PDIF (optical only) input.
	* Audio output drivers: up to 16-channel DAC output.
	* Control I/O drivers: UART, SPI and I2C.
	* PCM decoder (executing on A15).
	* PCM encoder with DEL and VOL ASP phases.
	* DM, FIL, SRC and AE0 ASP’s.
	* Framework supporting I-topology, decoders on ARM, and ASP’s on DSP.
2. FD package
	* SNG decoder (executing on ARM).
	* BM2, DEM and GEQ3 ASP’s.
3. Dolby Atmos IP packages
	* Dolby Intrinsics 1.8.1.1.
	* Dolby Digital Plus (DDP) decoder 4.7.2.
	* MAT-PCM decoder 2.1.0 and MAT-THD decoder 3.3.2.
	* Channel Audio Renderer (CAR) 1.0.0.
	* Object Audio Renderer (OAR) 1.0.0.
	* Bass Management (BMDA) from SIDK v1.2.

# Required Hardware

The hardware listed below is required for complete AR2 functionality:

1. Mistral K2G Rev. D EVM (SPRW274).
2. Mistral K2G EVM Audio Daughter Card Rev. B (SPRW279).
3. Momentum Data Systems (MDS) HDMI repeater kit, including:
	* MDS HSR41P HDMI Repeater module Rev. A.
	* MDS DA10X Interface board (IFB) Rev. A.
	* 5V AC to USB power supply for IFB and HSR41P.
	* Short 26- and 34-wire ribbon cables for connecting HSR41P and IFB.
	* Long 34-wire ribbon cable for connecting IFB and Audio Daughter Card.

# Software Installation

## Code Composer Studio (CCS)

AR2 was developed using CCS 6.1.2.00015 on Windows. Download the Off-line Installer for this version of CCS at the following URL:

<http://processors.wiki.ti.com/index.php/Download_CCS#Code_Composer_Studio_Version_6_Downloads>

Install CCS to C:\ti. During CCS installation:

* For “Processor Support”, select “Multi Core Processors (all)”
* For “Select Debug Probes”, select “TI XDS Debug Probe Support” and “Spectrum Digital Debug Probes and Boards”

AR2 was developed using the code generation tools listed below. These tools are installed as part of CCS.

* GNU ARM Tools 4.8.0.2014q3-20140805
* C6000 v8 Compiler Tools: 8.1.0

## Processor SDK-RTOS

AR2 was developed using Processor SDK-RTOS 2.0.2.11 for K2G on Windows. Release information for this version of Processor SDK-RTOS is provided at the URL below, along with links to the installer:

<http://software-dl.ti.com/processor-sdk-rtos/esd/K2G/02_00_02_11/index_FDS.html>

Install Processor SDK-RTOS to C:\ti. Among others, this should install the tools in the following list:

* SYS/BIOS 6.45.01.29
* XDAIS 7.24.00.04
* XDC Tools 3.32.00.06

## Open Source Package

The AR2 OS package is distributed using a self-extracting executable. Follow the steps below to install the software.

1. Double click the executable to initiate the installation process. Continue to the License Agreement.
2. Accept the terms of the click-wrap license agreement.
3. Change the installation destination folder to c:\ti\processor\_audio\_sdk\_1\_00\_00\_01 in the Choose Destination Location window. Complete the installation.

Upon completion of the installation, the processor\_audio\_sdk\_1\_00\_00\_01 folder contains the sub-folders described in the table below.

|  |  |
| --- | --- |
| **Folder** | **Contents** |
| docs\ | User documentation |
| pasdk\ | PASDK OS Dev (paf\)ARM application (common\, shared\, test\_arm\DSP application (common\, shared\, test\_dsp\) |
| psdk\_cust\ | IPC engineering release 3.43 (ipc\_3\_43\_00\_00\_eng\)Enhanced PDK K2G 1.0.1 (pdk\_k2g\_1\_0\_1\_0\_eng\) |
| scripts\ | Build scripts |
| tools\ | Alpha communication tools |

## Firmware Deliverable Package

The AR2 FD package is distributed as a .zip archive. To install the software, unzip the package .zip file to c:\ti\processor\_audio\_sdk\_1\_00\_00\_01\pasdk\paf. The FD package root folder is “pa”, so unzipping the file should overwrite the pa folder in paf and overwrite or create the sub-folders described in the table below.

|  |  |
| --- | --- |
| **Folder** | **Contents** |
| pa\asp\rel\ | ASP IP (bm2, dem, geq3) header files |
| pa\build | ARM and DSP IP libraries |
| pa\dec\rel\sng1 | Decoder IP (sng1) header files |
| pa\docs | User documentation |

## Dolby IP Packages

The AR2 Dolby IP packages are distributed using a select-extracting executable. Follow the steps below to install the software.

1. Double click the executable to initiate the installation process. Continue to the License Agreement.
2. Accept the terms of the click-wrap license agreement.
3. Accept the default installation folder, or change the folder to the desired location of the Dolby IP packages (e.g. c:\ti\processor\_audio\_sdk\_1\_00\_00\_01\release\dhip\_package). Complete the installation.
4. Upon completion of the installation, unzip all Dolby IP .zip packages (bmda, car, oar, ddp, intrinsics, matted, and oar) to c:\ti\processor\_audio\_sdk\_1\_00\_00\_01\dolby\_ip\dh-ip. After unzipping the Dolby IP .zip packages, the dh-ip folder should contain the sub-folders described in the table below.

|  |  |
| --- | --- |
| **Folder** | **Contents** |
| asp\ | ASP IP (bmda, car, oar) header files |
| build\ | ARM and DSP Dolby IP libraries |
| dec\ | Decoder IP (ddp, mat-thd) header files |
| DOC\ | User documentation |

## CCS Update

Update the CCS installation as below:

1. Launch CCS and create a new workspace (e.g. c:\ti\processor\_audio\_sdk\_1\_00\_00\_01\pasdk\workspace\_v6\_1).
2. Allow installation of newly discovered products located in c:\ti.
	* Ignore any messages about unsigned content
	* Restart CCS when prompted
3. Update Keystone2 device support
	* Select Help→Installation Details from the CCS menu
	* Select Keystone2 device support and click Uninstall.
	* Restart CCS when prompted.
	* Select Help→Install New Software.
	* Select “--All Available Sites--” from the “Work with:” drop-down menu.
	* Uncheck “Show only the latest version of the available software”.
	* Click the right arrow next to Keystone2 Device Support to display the Keystone2 device support version options.
	* Select the second version 1.1.5 Keystone2 device support option. Continue to and accept the license agreement.
	* Restart CCS when prompted.
4. Update Emulators
	* Select Help→Check for updates from the CCS menu
	* Update Spectrum Digital Emulators to 5.2.0.14 or greater
	* Update TI Emulators to 6.0.407.3 or greater
	* Restart CCS when prompted
5. Update RTSC Product discovery path to access to IPC engineering release 3.43 and Enhanced PDK K2G 1.0.1
	* Select Window→Preferences
	* Select Code Composer Studio→RTSC→Products
	* Click Add next to Product Discovery Path, add folder C:\ti\processor\_audio\_sdk\_1\_00\_00\_01\psdk\_cust

## PASDK OS Dev Build and Alpha Communication Tools

Download and install the additional tools below. Sed and Python are required to build the PASDK OS DEV software contained in the ti\processor\_audio\_sdk\_1\_00\_00\_01\pasdk\paf folder. Python is also required for alpha command control I/O over using the PyAlpha tool. The PyAlpha tool is further described below in Section 6.4.

1. Obtain Sed 4.2.1 from <http://gnuwin32.sourceforge.net/packages/sed.htm>,

executable installer, sed-4.2.1-setup.exe.

Install to c:\PA\_Tools\GnuWin32 (or change installation location in c:\ti\processor\_audio\_sdk\_1\_00\_00\_01\pasdk\setup\_env.bat).

1. Obtain Python 2.7.6 from <https://www.python.org/download/releases/2.7.6>

executable installer, python-2.7.6.amd64.msi.

Install to c:\PA\_Tools\Python27 (or change installation location in c:\ti\processor\_audio\_sdk\_1\_00\_00\_01\pasdk\setup\_env.bat).

# Emulator Firmware Update

The K2G EVM on-board emulator firmware must be updated for correct emulator operation. Update the firmware as described below.

1. Connect the EVM to the host PC using the provided USB Mini B to A plug type cable. Connect the Mini B plug on the cable to XDS\_USB connector J1 on the EVM.
2. Connect the EVM to power using the supplied 12V power supply.
3. Apply power to the EVM by pushing the SW1 switch to the “ON” position.
4. Open a DOS prompt on the PC. At the DOS prompt, issue these commands:
> cd c:\ti\ccsv6\ccs\_base\common\uscif\xds2xx
> update\_xds2xx.bat xds200
5. The batch file output should appear as shown below.
Updating CPLD ...
.
Updating Firmware ...
.
Rebooting ...
.
Reading Configuration ...
.
Check swRev is 1.0.0.8 or higher
.
boardRev=4
ipAddress=0.0.0.0
ipConfig=dhcp
ipGateway=0.0.0.0
ipNetmask=0.0.0.0
productClass=XDS2XX
productName=XDS200
serialNum=00:0E:99:03:90:0F
swRev=1.0.0.8
hostCPU=AM1802
emuCtrlType=Bit bang
extMemType=SDRAM
portUSB=true
portENET=false
portWIFI=false
portRS232=false
EnableUSBSerial=false
CurrentMeasure=false
Press any key to continue . . .

# Software Build

## PASDK OS Dev Libraries

The ARM and DSP applications depend on several libraries located in the ti\processor\_audio\_sdk\_1\_00\_00\_01\pasdk\paf folder. Detailed instructions for building any of the libraries in this folder are contained in C:\ti\processor\_audio\_sdk\_1\_00\_00\_01\pasdk\paf\ReadMe.txt. However, batch files for building only the necessary libraries for the ARM/DSP applications are provided in the C:\ti\processor\_audio\_sdk\_1\_00\_00\_01\scripts folder. To build the required libraries:

1. Open a DOS prompt.
2. From the DOS prompt, execute:
> cd C:\ti\processor\_audio\_sdk\_1\_00\_00\_01\scripts
> setup\_env.bat
> build\_paf\_libs.bat

## UART, SPI, and I2C Low Level Drivers (LLDs)

The enhanced PDK K2G 1.0.1 includes UART, SPI, and I2C serial port LLDs for control I/O. Each LLD supports CPU interrupt-callback mode and the UART LLD also supports EDMA dma-callback mode. Control I/O is implemented in the DSP application using the UART LLD in dma-callback mode. To build the serial port LLD libraries:

1. Open a DOS prompt.
2. From the DOS prompt, execute:
> cd C:\ti\processor\_audio\_sdk\_1\_00\_00\_01\scripts
> build\_pdk\_libs.bat

## K2G Platform Library

Configuration and control of the K2G EVM and Audio Daughter Card require the K2G platform library supplied with the enhanced PDK K2G 1.0.1. To build the K2G platform library:

1. If necessary, re-launch CCS using the workspace created above in Section 3.6.
2. Import the “platform\_lib\_evmk2g” CCS project into the CCS workspace. The project is located in C:\ti\processor\_audio\_sdk\_1\_00\_00\_01\psdk\_cust\pdk\_k2g\_1\_0\_1\_0\_eng\packages\ti\platform\evmk2g\platform\_lib.
3. Rebuild the project using the Debug build profile.
	* Right-click “platform\_lib\_evmk2g” project in CCS Project Explorer, select Build Configurations→Set Active→Project→Debug.
	* Right-click “platform\_lib\_evmk2g” project in CCS Project Explorer, select Rebuild Project.

## DSP Application

To build the DSP application:

1. Import the “test\_dsp” CCS project into the CCS workspace. The project is located in C:\ti\processor\_audio\_sdk\_1\_00\_00\_01\pasdk\test\_dsp.
2. Ensure the correct component software versions are selected by RTSC.
	* Right-click “test\_dsp” project in CCS Project Explorer, select Properties
	* On left-hand side of window, Select General. Select RTSC tab on the right-hand side of General window.
	* Confirm the following are installed and selected: (1) EDMA3 LLD 2.12.1; (2) SYS/BIOS 6.45.1.29; (3) System Analyzer (UIA Target) 2.0.3.43; (4) XDAIS 7.24.0.04; and (5) k2g PDK 1.0.1
	* For IPC, select version 3.42.0.02 and then (re-)select version 3.43.0.00\_eng.
	* Click OK.
3. Rebuild the project using the Debug\_pkgs build profile.
	* Right-click “test\_dsp” project in CCS Project Explorer, select Build Configurations→Set Active→Project→Debug\_pkgs.
	* Right-click “test\_dsp” project in CCS Project Explorer, select Rebuild Project.

## ARM Application

To build the ARM application:

1. Import the “test\_arm” CCS project into the CCS workspace. The project is located in C:\ti\processor\_audio\_sdk\_1\_00\_00\_01\pasdk\test\_arm.
2. Ensure the correct component software versions are selected by RTSC.
	* Right-click “test\_arm” project in CCS Project Explorer, select Properties
	* On left-hand side of window, Select “General”. Select “RTSC” tab on the right-hand side of “General” window.
	* Confirm the following are installed and selected: (1) SYS/BIOS 6.45.1.29; (2) System Analyzer (UIA Target) 2.0.3.43; and (3) XDAIS 7.24.0.04.
	* For IPC, select version 3.42.0.02 and then (re-)select version 3.43.0.00\_eng.
	* Click “OK”.
3. Rebuild the project using the Debug\_pkgs build profile.
	* Right-click “test\_dsp” project in CCS Project Explorer, select Build Configurations→Set Active→Project→Debug\_pkgs.
	* Right-click “test\_dsp” project in CCS Project Explorer, select Rebuild Project.

# Application Execution and Test

## I/O Configuration

AR2 supports boot- and run-time I/O configuration. Boot-time I/O configuration is determined by the at-boot I/O shortcuts in the CUS\_ATBOOT\_S definition in c:\ti\processor\_audio\_sdk\_1\_00\_00\_01\pasdk\test\_dsp\application\itopo\evmk2g\atboot.c, while run-time I/O configuration is performed by invoking I/O shortcuts via alpha commands. The available Input and Output shortcuts are defined in C:\ti\processor\_audio\_sdk\_1\_00\_00\_01\pasdk\test\_dsp\application\itopo\evmk2g\io.c and described in the tables below.

|  |  |
| --- | --- |
| **Input Configuration** | **Input Shortcut** |
| ADC 8-ch(w/ DAC 8-ch input) | execPAIInAnalog |
| S/PDIF | execPAIInDigital |
| Stereo HDMI(PCM or DDP) | execPAIHDMIInStereo |
| HDMI (MAT-THD/PCM) | execPAIInHDMI |

|  |  |
| --- | --- |
| **Output Configuration** | **Output Shortcut** |
| DAC 8-ch(w/ ADC 8-ch input) | execPAIOutAnalogSlave |
| DAC (8-ch) | execPAIOutAnalog |
| DAC (12-ch) | execPAIOutAnalog12Ch |
| DAC (16-ch) | execPAIOutAnalog16Ch |

## Connect EVM

1. Turn off power to the EVM using SW1 on the EVM. Disconnect the 12V power supply.
2. Assemble the Audio Daughter Card (assembly instructions are packaged with Audio Daughter Card).
3. Attach the Audio Daughter Card to the K2G EVM. The Audio Daughter Card connects to the EVM through the JP1 “Audio Expansion” and J12 “Serial Expansion” connectors on EVM.
4. Connect the PC and K2G EVM using DB9 serial RS-232 cable delivered with EVM.
	* A USB to UART FTDI cable can be used on the PC side of the connection.
	* Attach the DB9 connector to the P1 “UART0” connector on the EVM.
	* Push the SW12-UART0\_DB9\_EN switch to the “ON” position.
5. Connect the EVM emulation port to the PC as described in Section 4.
6. Connect the MDS HSR41P to the MDS IFB.
	* Connect HSR41P J4 “Main” header to the IFB J5 “Main” header using the short 26-wire ribbon cable. Ensure the red wire on the ribbon cable is connected between HSR41P J4 Pin1 and IFB J5 Pin1.
	* Connect HSR41P J7 “Sec” header to the IFB J2 “Sec” header using the short 34-wire ribbon cable. Ensure the red wire on the ribbon cable is connected between HSR41P J7 Pin1 and IFB J2 Pin1.
7. Connect the MDS IFB to the Audio Daughter Card. Connect IFB J3 “McASP0” header to Audio Daughter Card “I2S Header” using the supplied long 34-wire ribbon cable. Ensure the red wire on the ribbon cable is connected between IFB J3 Pin1 and Audio Daughter Card “I2S Header” Pin1 (note the “I2S Header” is only 32 pins, so the two wires on the ribbon cable opposite Pin1 are unused).
8. Connect HDMI cable from audio source (e.g. Blu-ray player) to HSR41P J6 “IN”. Connect HDMI cable to audio sink (e.g. HDMI enabled monitor) from HSR41P J8 “HDMI OUT”.
9. Connect 5V AC to USB power supply to IFB power connector J1.
10. Re-connect the 12V power supply to the EVM. Turn on power to the EVM using SW1.

### ADC 8-Ch Input and DAC 8-Ch Output

Connect ADC inputs to the ADC\_IN0-3 input jacks on the Audio Daughter Card. Processed PCM samples will be output to the corresponding DAC output jacks DAC\_OUT0-3 on the Daughter Card.

### S/PDIF Input and DAC 8-Ch Output

Connect S/PDIF input to the MOD1 Digital Input (Optical) connector on the Audio Daughter Card. Processed PCM samples will be output to the DAC output jacks DAC\_OUT0-3 on the Daughter Card.

### HDMI Input and DAC Output

Connect HDMI input source to the HSR41P repeater module as described above. Processed PCM samples will be output to DAC output jacks DAC\_OUT0-6 on the Daughter Card. The number of active DAC outputs depends on: (1) the selected Output Shortcut (execPAIOutAnalog, execPAIOutAnalog12Ch, or execPAIOutAnalog16Ch); (2) the configured ENC channel map from; and (3) the active renderer channel configuration request. For example, issue the alpha commands listed below to enable 12-channel DAC for channel-based audio using CAR with speaker layout L/R, C, Sub, Ls/Rs, Lrs/Rrs, Ltf/Rtf, Ltr/Rtr:

> python.exe pyalpha -I alpha -h pa\_i13\_evmk2g\_io\_a -p COM1 execPAIOutAnalog12Ch

alpha execPAIOutAnalog12Ch

> python.exe pyalpha -I alpha -h pa\_i13\_evmk2g\_io\_a -p COM1 writeENCChannelMapFrom16(PAF\_LEFT,PAF\_RGHT,PAF\_LSUR,PAF\_RSUR,PAF\_CNTR,PAF\_SUBW,PAF\_LBAK,PAF\_RBAK,PAF\_LTFT,PAF\_RTFT,PAF\_LTRR,PAF\_RTRR,-3,-3,-3,-3)

> python.exe pyalpha -I alpha -h pa\_i13\_evmk2g\_io\_a -p COM1 writeSYSChannelConfigurationRequestSurround4LtfRtfLtrRtr\_1

alpha /\* none \*/

> python.exe pyalpha -I alpha -h pa\_i13\_evmk2g\_io\_a -p COM1 readCARChannelCfgOverride

alpha writeCARChannelCfgOverrideUnknown

## Load and Execute Code

Follow the steps below from within CCS to load and execute the DSP and ARM application code:

1. Click View→Target Configurations.
2. In the Target Configurations window, open the Projects→test\_dsp→targetConfigs folder.
3. Right-click on K2GEVM.ccxml contained in the targetConfigs folder.
4. Select Launch Selected Configuration.
5. In the Debug window, right-click on the C66x and select Connect Target. The output from the GEL code invoked on the target connection can be observed in the Console output window. The final line of this output should read “C66xx: GEL Output: DDR3A initialization complete”.
6. In the Debug window, right-click on the CortexA15 and select Connect Target. The output from the GEL code invoked on the target connection can be observed in the Console output window. The final line of this output should read “CortexA15: GEL Output: A15 non secure mode entered”.
7. In the Debug Window, click on the C66x and then open Run→Load→Load Program. In the Load Program window, click on Browse project, open the “test\_dsp” folder, and click on “test\_dsp.out” and OK to select the program to be loaded. Click on OK in the Load Program window to load the program to the C66x.
8. In the Debug Window, click on the CortexA15, and then open Run→Load→Load Program. In the Load Program window, click on Browse project, open the “test\_arm” folder, and click on “test\_arm.out” and OK to select the program to be loaded. Click on OK in the Load Program window to load the program to the A15.
9. In the Debug window, click on the C66x and then Open Run→Resume to execute the DSP code.
10. In the Debug window, click on the CortexA15 and then Open Run→Resume to execute the ARM code.
11. The CIO Console output window should display memory usage statistics. The final line of output should display the memory usage summary for the MSMC SHM heap.
12. In the Debug Window, click on the CortexA15, and then open Run→Disconnect Target. This step is required for clean audio output (known issue for this release).

## PyAlpha Python Scripts for Alpha Communication

AR2 supports alpha command control over UART using the PyAlpha tool located in c:\ti\processor\_audio\_sdk\_1\_00\_00\_01\tools. PyAlpha is invoked on the command-line in a DOS shell. PyAlpha documentation can be obtained by executing PyAlpha with a “--help” argument as below (note: the Python executable is contained in the DOS shell path for this example).

> cd c:\ti\processor\_audio\_sdk\_1\_00\_00\_01\tools
> python pyalpha --help

usage: PyAlpha [-?] [-h FILE] [-I DIR] [-p PORT] [-b BAUD] [-t DIR] [-bin DIR]

 [-r] [--fast] [--repeat N T N T]

 [ALPHAS [ALPHAS ...]]

Alpha Command Communicator

positional arguments:

 ALPHAS Alpha commands, separated by spaces

optional arguments:

 -?, --help show this help message and exit

 -h FILE, --header FILE

 Add FILE\_a.h to symbol definition list and FILE\_a.hdM

 to symbol decomposition list

 -I DIR, --include DIR

 Add DIR to the include path. [default: ./alpha]

 -p PORT, --port PORT Communication port [default: COM1]

 -b BAUD, --baud BAUD Baud rate [default: 19200]

 -t DIR, --temp DIR Temporary file save location [default: .]

 -bin DIR Binary file location [default: ./bin]

 -r, --retain Retain temporary files

 --fast Skip post-processing

 --repeat N T N T Repeat an alpha command N times with a T second break

 between

## Alpha Commands for AE0 Configuration/Status

AR2 includes the Audio Example 0 (AE0) ASP. This ASP transforms stereo (Left/Right) PCM into 3stereo (Left/Right/Center) PCM. AE0 can be included in the signal chain by uncommenting Line 63 (63: #define \_AE0\_) in C:\ti\processor\_audio\_sdk\_1\_00\_00\_01\pasdk\test\_dsp\framework\itopo\patch.c and rebuilding the DSP application. AE0 documentation is provided in C:\ti\processor\_audio\_sdk\_1\_00\_00\_01\docs\DA10x\_UG\_AspExample.pdf, and the alpha commands contained in C:\ti\processor\_audio\_sdk\_1\_00\_00\_01\tools\alpha\ae\_a.h can be used to configure AE0 and consult AE0 status.

In the example alpha command sequence below, AE0 is configured to generate 3stereo using a gain of 0.5 (16384 in S16Q15 format) using COM port 1. After disabling AE0, the Master Volume Control is set to -20 (= 40\*0.5 dB). Note the ‘>’ indicates the commands are issued from a DOS shell.

> cd c:\ti\processor\_audio\_sdk\_1\_00\_00\_01\tools
> python.exe pyalpha -I alpha -h pa\_i13\_evmk2g\_io\_a -p COM1 readAEMode

alpha writeAEModeDisable

> python.exe pyalpha -I alpha -h pa\_i13\_evmk2g\_io\_a -p COM1 writeAEModeEnable

alpha /\* none \*/

> python.exe pyalpha -I alpha -h pa\_i13\_evmk2g\_io\_a -p COM1 readAEMode

alpha writeAEModeEnable

> python.exe pyalpha -I alpha -h pa\_i13\_evmk2g\_io\_a -p COM1 readAEScaleQ15

alpha wroteAEScaleQ15,0x0000

> python.exe pyalpha -I alpha -h pa\_i13\_evmk2g\_io\_a -p COM1 writeAEScaleQ15(16384)

alpha

> python.exe pyalpha -I alpha -h pa\_i13\_evmk2g\_io\_a -p COM1 readAEScaleQ15

alpha wroteAEScaleQ15,0x4000

> python.exe pyalpha -I alpha -h pa\_i13\_evmk2g\_io\_a -p COM1 readSYSChannelConfigurationRequest

alpha 0xce20,0x2008,0x010c,0x0000,0x0000,0x0000

> python.exe pyalpha -I alpha -h pa\_i13\_evmk2g\_io\_a -p COM1 writeSYSChannelConfigurationRequestStereo

alpha /\* none \*/

> python.exe pyalpha -I alpha -h pa\_i13\_evmk2g\_io\_a -p COM1 readSYSChannelConfigurationRequest

alpha 0xce20,0x2008,0x0003,0x0000,0x0000,0x0000

> python.exe pyalpha -I alpha -h pa\_i13\_evmk2g\_io\_a -p COM1 writeSYSChannelConfigurationRequestSurround4\_1

alpha /\* none \*/

> python.exe pyalpha -I alpha -h pa\_i13\_evmk2g\_io\_a -p COM1 readSYSChannelConfigurationRequest

alpha 0xce20,0x2008,0x010c,0x0000,0x0000,0x0000

> python.exe pyalpha -I alpha -h pa\_i13\_evmk2g\_io\_a -p COM1 writeAEModeDisable

alpha /\* none \*/

> python.exe pyalpha -I alpha -h pa\_i13\_evmk2g\_io\_a -p COM1 readAEMode

alpha writeAEModeDisable

> python.exe pyalpha -I alpha -h pa\_i13\_evmk2g\_io\_a -p COM1 writeVOLControlMasterN(-40)

alpha /\* none \*/

# Known Issues

* MAT-THD/PCM decoding unstable.
* Decoder status not properly shared between ARM/DSP (cache coherency issue).
* Alpha commands over UART sometimes TIMEOUT.
* SPI and I2C LLDs only support CPU interrupt-callback mode.