

# A2B STACK ON LINUX USER GUIDE

<b>Document Status</b>	Approved
<b>Approved by</b>	ASH

ANALOG DEVICES, INC.

[www.analog.com](http://www.analog.com)

## Revision List

**Table 1: Revision List**

Revision	Date	Description
0.1	04-May-2017	Draft Version
0.2	12-May-2017	Absorbing review comments
1.0	12-May-2017	Baselined for Rel16.0.0
1.1	5-Oct-2017	Updated CCES version and minor updates in section 6.1.1.2 ,6.1.2.1 & 7.2.1 – Release 17.0.0
2.0	5-Oct-2017	Baselined for Rel17.0.0
2.1	7-May-2018	Updated for Release 19.0.0 to reflect the following changes 1. Folder Re-organization 2. Makefile Project Changes 3. Linux Add-in version references removed. 4. Board Specific Plugin support removed
2.2	17-May-2018	Addressed review comments
3.0	06-June-2018	Baselined for Rel19.0.0
3.1	17-Oct-2018	Updated for Release 19.2.0 to reflect the following changes 1. Added section on A2B Linux command line tools
3.2	26-Oct-2018	Addressed review comments
3.3	10-Dec-2018	Restructured few sections.
4.0	12-Dec-2018	Approved for 19.2.0 release
4.1	29-Aug-2019	Updated for 19.3.0 release
4.2	30-Aug-2019	Addressed review comments
5.0	30-Aug-2019	Approved and Baselined for Rel19.3.0
5.1	15-Feb-2023	Updated for Release 19.4.4 to reflect the following changes: 1) Support for Yocto distribution a) Download, build, develop and install Yocto Linux (release/yocto-2.1.0) on Host PC b) Run Yocto Linux on the ARM core of the ADSP-SC5xx family of processors 2) Demo applications for following platforms – a) ADSP-SC589-mini b) ADSP-SC598
5.2	09-Mar-2023	Addressed review comments
6.0	16-Mar-2023	Approved and Baselined for Rel19.4.4

## **Copyright, Disclaimer Statements**

### **Copyright Information**

Copyright (c) 2009-2023 Analog Devices, Inc. All Rights Reserved. This software is proprietary and confidential to Analog Devices, Inc. and its licensors. This document may not be reproduced in any form without prior, express written consent from Analog Devices, Inc.

### **Disclaimer**

Analog Devices, Inc. reserves the right to change this product without prior notice. Information furnished by Analog Devices is believed to be accurate and reliable. However, no responsibility is assumed by Analog Devices for its use; nor for any infringement of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under the patent rights of Analog Devices, Inc.

## Table of Contents

<b>Revision List.....</b>	<b>2</b>
<b>Copyright, Disclaimer Statements .....</b>	<b>3</b>
<b>Table of Contents.....</b>	<b>4</b>
<b>List of Figures .....</b>	<b>5</b>
<b>List of Tables.....</b>	<b>6</b>
<b>1 Introduction.....</b>	<b>7</b>
1.1 Scope.....	7
1.2 Organization of the Guide .....	8
<b>2 A2B Linux Package Details .....</b>	<b>9</b>
<b>3 System Requirements .....</b>	<b>10</b>
3.1 Hardware Requirements .....	10
3.1.1 PC.....	10
3.1.2 A2B Evaluation Boards .....	10
3.2 Software Requirements.....	11
3.2.1 Windows Host .....	11
3.2.2 Linux Host .....	11
<b>4 System Set Up.....</b>	<b>12</b>
4.1 Hardware Setup .....	12
4.2 Yocto Software Setup on Linux Host.....	12
4.3 Additional Packages.....	12
<b>5 Enabling Remote Processor option in Yocto.....</b>	<b>13</b>
5.1 Kernel Configuration .....	13
<b>6 A2B Stack and Demo application build instructions.....</b>	<b>15</b>
6.1 Building and Installing the SDK .....	15
6.2 Adding recipe to build application in Yocto .....	16
6.2.1 Creating recipe.....	16
6.2.2 Adding Recipe to ramboot Image .....	17
6.3 Makefile based build for SC5XX.....	18
6.4 Makefile based build for a different platform .....	19
<b>7 SC589-Mini Linux Demo .....</b>	<b>23</b>
7.1 Demo Set Up .....	23

7.2	Connections .....	23
7.2.1	Evaluation boards .....	24
7.2.2	Ethernet and UART Connection to Linux Host.....	24
7.2.3	Audio In/Out .....	24
7.3	Installing the Yocto Sources for ADSP-SC589-Mini.....	24
7.4	Booting Linux on SC589-Mini .....	25
7.5	Running the A2B SHARC(Core1) application.....	27
7.6	Running the A2B ARM(Core0) demo application .....	28
<b>8</b>	<b>ADSP-SC598 Linux Demo .....</b>	<b>29</b>
8.1	Demo Set Up .....	29
8.2	Connections .....	29
8.2.1	Evaluation boards .....	30
8.2.2	Ethernet and UART Connection to Linux Host.....	30
8.2.3	Audio In/Out .....	30
8.3	Installing the Yocto Sources for ADSP-SC598 .....	30
8.4	Booting Linux on SC598 .....	31
8.5	Running the A2B SHARC(Core1) application.....	33
8.6	Running the A2B ARM(Core0) demo application .....	34
<b>9</b>	<b>Demo Application and A2B stack features on Linux.....</b>	<b>35</b>
9.1	Build Time options.....	35
9.1.1	Utilizing a custom schematic .....	35
9.1.2	Trace file and Level .....	35
9.1.3	Sequence chart feature .....	36
9.2	Command Line options .....	36
9.2.1	Specifying Sequence chart file and generating UML diagram.....	36
9.2.2	Enabling debug mode .....	37
9.2.3	Running as a daemon .....	38
9.2.4	Enabling audio using A2B sound card .....	38
9.3	Run Time options.....	38
9.3.1	Rediscovery .....	39
9.3.2	Register dump of nodes .....	39
9.3.3	Line Fault messages .....	40
<b>10</b>	<b>Appendix A.....</b>	<b>41</b>
10.1	A2B Linux Command Line Tools .....	41

## List of Figures

Figure 1: A2B stack on Linux architecture .....	7
Figure 3 : Remote Proc option in Linux Configuration.....	13
Figure 2 : SDK Poky version.....	16
Figure 4: Demo block diagram.....	23
Figure 5: Demo hardware connections .....	23
Figure 6: Output of DHCP command in terminal.....	25
Figure 7: u-boot console output .....	26
Figure 8: Editing boot arguments in u-boot.....	26
Figure 9: Load Core1 application by remote proc (on SC589-Mini) .....	27
Figure 10: A2B Linux demo application on ADSP-SC589 Mini .....	28
Figure 11: A2B system with ADSP-SC598 as Host .....	29
Figure 12: ADSP-SC598 Demo Hardware Connections .....	29
Figure 13: ADSP-SC598 u-boot console output .....	31
Figure 14: ADSP-SC598 U-Boot console output .....	32
Figure 15: Editing boot arguments in u-boot.....	33
Figure 16: Load Core1 application by remoteproc (on SC598) .....	34
Figure 17: A2B Linux demo application on ADSP-SC598.....	34
Figure 18: Enabling Debug Mode in application .....	38
Figure 19: Running application as a daemon.....	38
Figure 20: Enabling audio using A2B sound card .....	38
Figure 21: Initiating Rediscovery .....	39
Figure 22: Master node register dump and Slave node register dump.....	40
Figure 23: Line fault during discovery and Line fault post discovery .....	40
Figure 24: A2B System with Linux PC as Host .....	41

## List of Tables

Table 1: Revision List .....	2
Table 2: a2b-linux directory details .....	9
Table 3: a2b-linux/a2b-adsp-sc5xx-linux directory details.....	9
Table 4: Makefile Build Output Folders.....	18
Table 5: Top level Makefile parameters.....	21
Table 6: Terminology.....	42
Table 7: References .....	42

The demo A2B application on Linux is built by re-implementing A2B Stack's Platform Abstraction Layer (PAL) for Linux operating system. The architecture of A2B Stack Software PAL and demo application on Linux is as shown in Figure 1. The PAL layer on Linux resides in user space. It utilizes the standard system call services of Linux Kernel to interact with the hardware. Linux exports standard system calls to the user space for timer, i2c, memory management etc. Hence the PAL layer on Linux and demo application can be configured to be run on any platform supporting a port of Linux.



## 1.1 Scope

The scope of the document is to provide steps to build and run the A2B Stack Software and demo application in Linux running on ADSP- SC589-Mini and SC598 EZ-Kits. Guidelines are also provided to port the demo application on Linux for a different platform. The documents also cover the various features of demo application and A2B Stack Software in Linux.

## **1.2 Organization of the Guide**

Section 1: Introduction to A2B Stack Software and application on Linux.

Section 2: Covers the details of the various folders in the A2B Stack Software for Linux.

Section 3: List the various system requirements for running the demo on ADSP-SC5XX EZ-Kit.

Section 4: Details the various steps in setting up the system with respect to hardware connections and software installation.

Section 5: Details various steps to enable Remote Processor option in Yocto

Section 6: Covers the details on SDK installation, A2B Stack and Demo application build steps

Section 7: Provides the steps to build, flash image and run the demo application on ADSP-SC589 Mini EZ-Kit.

Section 8: Provides the steps to build, flash image and run the demo application on ADSP-SC598 EZ-Kit.

Section 9: Details the features of A2B Stack Software and demo application on Linux.



## 2 A2B Linux Package Details

After the installation of A2B Linux package, the various files specific to the demo application on Linux are located in the `/opt/analog/a2b-software/X.Y.Z/Target/examples/demo/a2b-linux` folder. The various folders in the `a2b-linux` folder are explained in Table 2.

**Table 2: a2b-linux directory details**

Folder Name	Purpose
a2b-adsp-sc5xx-linux	This folder contains the sources for a2bstack PAL implementation on Linux and A2B Stack Software Linux demo application for SC589-mini and SC598 Platforms. It also contains the audio routing application files and associated CCES project on Windows for the SHARC core of ADSP-SC589-mini and SC598 Platforms.

Here use following directory < a2b-adsp-sc5xx-linux > based on your platform

a2b-adsp-sc589-mini-linux

a2b-adsp-sc598-linux

The various folders in the `/opt/analog/a2b-software/X.Y.Z/Target/examples/demo/a2b-linux/a2b-adsp-sc5xx-linux` folder are explained in Table 3.

**Table 3: a2b-linux/a2b-adsp-sc5xx-linux directory details**

Folder Name	Purpose
a2b-app_Core1	This folder contains Windows CCES project for audio routing application for SHARC core (Core 1).
a2b-app-linux_Core0	This folder contains the Linux CCES project for A2B Stack Software demo application in Linux running on ARM core (Core 0).
binaries	This folder contains pre-built demo application binaries for SHARC core (Core 1) and ARM Core (Core 0) running Linux
Makefiles	Contains the Makefiles required to build the complete ARM demo application in Linux using the <i>make</i> command

## 3 System Requirements

### 3.1 Hardware Requirements

#### 3.1.1 PC

1. Windows Host
2. Linux Host running Ubuntu 20.04 64 bit.

The configuration of PCs should meet the requirements for CCES Windows & Linux version as mentioned in CCES release notes [3]. Please refer [2] to download the CCES package. Yocto Linux port for ADSP-SC5XX platforms is available in [4], ADSP-SC589 Mini is available in [9] and ADSP-SC598 is available in [10].

#### 3.1.2 A2B Evaluation Boards

- ADSP-SC589-Mini EZ-Kit BOM Rev 1.5 or greater
  - Evaluation Board
  - 12V Power Supply 1.5A
  - Twisted Pair A2B Cable with DuraClick connectors
  - 1 Ethernet cable – RJ45 Cat5E
  - ICE-1000 programmer with USB cable
  - USB Micro-B to Standard-A cable
- ADSP-SC598 SOM Board
  - EV-SOMCRR-EZKIT Rev A or later
  - EV-SC598-SOM Rev B or later
  - Twisted Pair A2B Cable with Molex H-DAC connector
  - 1 Ethernet cable – RJ45 Cat5E
  - ICE-1000 programmer with USB cable
  - USB Micro-B to Standard-A cable
- EVAL-AD2428WB1BZ Rev2.0 package
  - Evaluation Board EVAL-AD2428WB1BZ (Stereo In, Stereo Out, Stereo Microphones)

- Twisted Pair A2B Cable with DuraClick connectors (30cm or 4m)
  - 3.5mm Audio cable or male-male adapter
- EVAL-AD2428WC1BZ Rev2.1 package
  - Evaluation Board EVAL-AD2428WC1BZ (3 Microphones)
  - Twisted Pair A2B Cable with DuraClick connectors (30cm or 4m)
- Passive speakers for Audio Output.
- One audio source device (e.g. iPods)
- Two audio sink devices (e.g. Speakers)
- 3.5mm Audio cable or male-male adaptor

## **3.2 Software Requirements**

### **3.2.1 Windows Host**

- Cross-Core Embedded Studio 2.11.0 or later

### **3.2.2 Linux Host**

- Cross-Core Embedded Studio 2.11.0 or later
- Yocto Linux port for ADSP-SC5xx processors

## 4 System Set Up

### 4.1 Hardware Setup

For details on jumper settings of the evaluation boards refer to Section 3 of A2B Quick Start Guide [1].

### 4.2 Yocto Software Setup on Linux Host

Yocto port for SC589-mini is available on a different repository as compared to SC598 platform. Make sure that you follow below steps carefully to install the Yocto source for all the supported platforms. These steps will enable you run RAM Boot based images on the Target platform.

Follow Section 7 to run Yocto Linux images, A2B demo application on SC589-Mini platform and Section 8 to run on SC598 platform. Below are the high level steps to be followed:

1. Download the latest sources of **Yocto Linux for ADSP-SC5xx processors** available online. Refer [9] for ADSP-SC589-Mini and [10] for ADSP-SC598 platforms. Follow the steps mentioned in the above documents to setup Linux Host PC and build the required Target Image files.
2. Generated image files can be found in build/tmp/deploy/images/<MACHINE> directory, which are useful for booting Yocto Linux and executing demo applications in target. Here <MACHINE> refers to adsp-sc589-mini OR adsp-sc598-som-ezkit
3. Cross check U-Boot ldr files are available for flashing U-Boot on ADSP-SC5XX EZ-Kits
  - *u-boot-sc589-ezkit.ldr* for ADSP-SC589 Mini EZ-Kit
  - *Stage1-boot.ldr* and *stage2-boot.ldr* for ADSP-SC598 EZ-Kit
4. Boot image file(fitImage) which includes Kernel, DTBs and Overlays; RAMDISK FILE(adsp-sc5xx-ramdisk-adsp-sc5xx-ezkit.cpio.xz.u-boot) for ADSP-SC5XX EZ-Kits.
5. Copy the build images into “/tftpboot” folder and continue to run RAM Boot.

### 4.3 Additional Packages

Several packages are required in the Linux host which include minicom (for serial port access), TFTP server for transferring Linux images to target over Ethernet. For more details refer to the “Installing Required Packages” section in Yocto Linux Getting Started page [9] for SC589-Mini and [10] for SC598.

## 5 Enabling Remote Processor option in Yocto

Remote processor option needs to be enabled in order to load the SHARC executable to SHARC core. The steps to enable remote processor option in Yocto build configuration are explained here.

### 5.1 Kernel Configuration

Follow the steps below to update the kernel configuration

1. Open a terminal in Linux host.
2. Give the below commands to change configurations and enable remote processor support

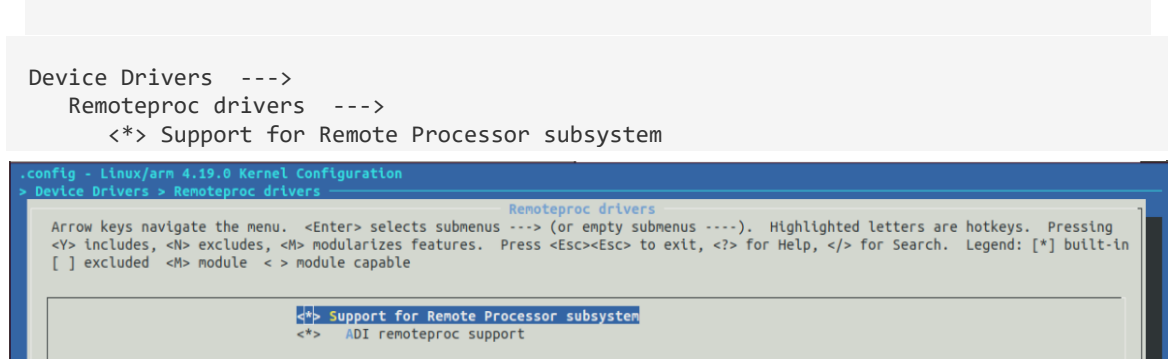
```
$ cd ~/YOCTO_DIRECTORY
$ source setup-environment -m <MACHINE>
```

Here set the <MACHINE> one of the following based on your platform.

adsp-sc589-mini  
adsp-sc598-som-ezkit

Give “*bitbake linux-adi -c menuconfig*” command to change the configurations of Linux (Enable Remote processor option)

Navigate to remote processor option



**Figure 2 : Remote Proc option in Linux Configuration**

Select the “Support for Remote Processor Subsystem” and press “Y” to enable the driver. Do same to enable “ADI remoteproc support” as shown in Figure 3. If these are already marked as enabled then ignore next step (Step 3) and continue to Step 4.

3. Then run “*bitbake linux-adi -C compile*” to generate kernel image fitImage and rootfs file
4. Build the SHARC application (a2b-app\_Core1) in CCES by following steps in link [3] and copy the **.ldr** file to Linux Host PC to the path: “YOCTO\_DIRECTORY/sources/meta-adi/meta-adi-adsp-sc5xx/recipes-kernel/linux-firmware/linux-firmware”

5. Update "meta-adi-adsp-sc5xx/recipes-kernel/linux-firmware/linux-firmware\_%.bbappend" file as below.

```
SUMMARY = "Linux kernel firmware files from ADI distribution"
DESCRIPTION = "These binaries provide kernel support for ADI sc5xx boards"

FILESEXTRAPATHS_prepend := "${THISDIR}/linux-firmware:"
SRC_URI += " \
            file://a2baudiorouter-app-<sc5xx>_Core1.ldr \
            "

do_install_append() {
    install -m 0644 ${WORKDIR}/a2baudiorouter-app-<sc5xx>_Core1.ldr ${D}/lib/firmware/
}
PACKAGES += " \
            ${PN}-fastboot \
            "

FILES_${PN}-fastboot = " \
                        /lib/firmware/a2baudiorouter-app-<sc5xx>_Core1.ldr \
                        "

```

**Note:** <sc5xx> refers to platforms – sc589-mini or sc598

Below are the *ldr* file names based on your platform

- a2baudiorouter-app-sc589-mini\_Core1.ldr
- a2baudiorouter-app-sc598\_Core1.ldr

6. Append "linux-firmware-fastboot" and "linux-firmware-sharc-alsa" to "IMAGE\_INSTALL" in the file YOCTO\_DIRECTORY/sources/meta-adi/meta-adi-adsp-sc5xx/recipes-adi/images/adsp-sc5xx-ramdisk.bb as follows

```
IMAGE_INSTALL = " \
    packagegroup-core-boot \
    packagegroup-base \
    busybox-watchdog-init \
    linux-firmware-fastboot \
    linux-firmware-sharc-alsa \
    "

```

## 6 A2B Stack and Demo application build instructions

Yocto Project Software Development Kit (SDK) is required to build the A2B stack demo application on Linux. It can be built using any of the two methods.

- Adding a Recipe to build application as described in Section 6.2
- Using make command to build application as described in Section 6.3

Both methods use makefile to build the application. Method 1 manually uses make command to build the application while in method 2 a recipe is created and added in the “adi-recipes” so that A2B demo application can be built along with RAM Boot image.

### 6.1 Building and Installing the SDK

Give following commands to build SDK in Yocto workspace.

```
$ cd <YOCTO_DIRECTORY>
$ source setup-environment -m <MACHINE>
$ bitbake adsp-sc5xx-full -c populate_sdk
$ ./poky-glibc-x86_64-adsp-sc5xx-ramdisk-armv7at2hf-neon-toolchain-X.Y.Z.sh
```

Here set the <MACHINE> one of the following based on your platform.

adsp-sc589-mini  
adsp-sc598-som-ezkit

<YOCTO\_DIRECTORY> is the path where Yocto for SC5xx is installed.

**Note:** In case of populate SDK failure, run below commands and then run bitbake command again

```
$ bitbake meta-environment-adsp-<sc5xx>-som-ezkit -cdo_generate_content -Snone
$ bitbake meta-environment-adsp-<sc5xx>-som-ezkit -cdo_generate_content -Sprintdiff
```

Here <sc5xx> refers to platforms – sc589-mini or sc598

For SC589mini platforms, run below command:

```
$ ./tmp/deploy/sdk/poky-glibc-x86_64-adsp-sc5xx-full-aarch64-adsp-sc598-som-ezkit-toolchain-X.Y.Z.sh
```

For full image on SC598, run below command:

```
$ ./tmp/deploy/sdk/poky-glibc-x86_64-adsp-sc5xx-full-aarch64-adsp-sc598-som-ezkit-toolchain-X.Y.Z.sh
```

Here “X.Y.Z” refers to version of poky distribution. SDK version 3.1.8 is used for SC589mini and SC598 platforms. During execution of the script it will provide option to choose root location for the

SDK toolchain. Make sure that the SDK is available at the suggested default directory as the path is used in makefiles to build application. There would be path changes required in makefiles if default path is not selected for SDK installation. For SC589mini, it is “/opt/poky/3.1.8” and SC598, it is “/opt/adi-distro/3.1.8”.

To check the poky version, refer to the Build Configuration section in UART terminal as shown in Figure 3 while populating SDK from the Yocto directory for sc589-mini platform.

```
Build Configuration:
BB_VERSION           = "1.46.0"
BUILD_SYS            = "x86_64-linux"
NATIVELSBSTRING      = "universal"
TARGET_SYS           = "arm-poky-linux-gnueabi"
MACHINE              = "adsp-sc589-mini"
DISTRO               = "adi-distro"
DISTRO_VERSION        = "3.1.8"
TUNE_FEATURES        = "arm armv7a vfp thumb neon callconvention-hard"
TARGET_FPU           = "hard"
meta
meta-poky
meta-yocto-bsp        = "HEAD:6ebb33bdaccaeaff0c85aab27acf35723df00d8"
meta-adi-adsp-sc5xx   = "HEAD:b10a950c031c3d098c468129709910cc3563afe9"
meta-oe
meta-python
meta-networking       = "HEAD:11eae114522a6befa06c7f4021a83bc016133543"

Initialising tasks: 100% |#####|
Sstate summary: Wanted 19 Found 0 Missed 19 Current 1542 (0% match, 98% complete)
NOTE: Executing Tasks
WARNING: linux-adi-5.4.183-r0 do_kernel_metadata: defconfig detected in WORKDIR. sc589-mini_defconfig skipped
NOTE: Tasks Summary: Attempted 3967 tasks of which 3831 didn't need to be rerun and all succeeded.
```

Figure 3 : SDK Poky version

## 6.2 Adding recipe to build application in Yocto

### 6.2.1 Creating recipe

Follow below steps to create a recipe for A2B demo application in Yocto.

1. Create a new folder with name “a2bapp-linux” in YOCTO\_DIRECTORY/sources/meta-adi/meta-adi-adsp-sc5xx/recipes-adi/
2. Copy the Bitbake file “a2bapp-linux.bb” available in the path /opt/analog/a2b-software/X.Y.Z/Target/examples/demo/a2b-linux/<a2b-adsp-sc5XX-linux> to YOCTO\_DIRECTORY/sources/meta-adi/meta-adi-adsp-sc5xx/recipes-adi/a2bapp-linux.
3. Create a folder with name “a2bapp-linux” in YOCTO\_DIRECTORY/sources/meta-adi/meta-adi-adsp-sc5xx/recipes-adi/a2bapp-linux/
4. Copy the “a2bstack” and “examples” folders available in the /opt/analog/a2b-software/X.Y.Z/Target/ to YOCTO\_DIRECTORY/sources/meta-adi/meta-adi-adsp-sc5xx/recipes-adi/a2bapp-linux/a2bapp-linux



## 6.2.2 Adding Recipe to ramboot Image

Follow steps below to add created recipe to ramboot image

1. Append “a2bapp-linux”, “linux-firmware-fastboot”, “linux-firmware-sharc-alsa”, “alsa-utils”, “alsa-lib”, “mplayer-common” and “play” to “IMAGE\_INSTALL” in the file “YOCTO\_DIRECTORY/sources/meta-adi/meta-adi-adsp-sc5xx/recipes-adi/images/adsp-sc5xx-ramdisk.bb”

```
IMAGE_INSTALL = " \
    packagegroup-core-boot \
    packagegroup-base \
    busybox-watchdog-init \
    a2bapp-linux \
    linux-firmware-fastboot \
    linux-firmware-sharc-alsa \
    alsa-utils \
    alsa-lib \
    mplayer-common \
    play \
"
```

2. Remove/Comment below lines from the file “Yocto\_Directory/sources/meta-adi/meta-adi-adsp-sc5xx/recipes-adi/images/adsp-sc5xx-ramdisk.bb”.

```
rm -rf ${IMAGE_ROOTFS}/usr/share/alsa
rm -rf ${IMAGE_ROOTFS}/usr/lib/libasound.so.2.0.0
rm -rf ${IMAGE_ROOTFS}/usr/lib/libasound.so.2
rm -rf ${IMAGE_ROOTFS}/usr/sbin/alsactl
```

3. Open a new terminal and give below commands to build the recipe that is added

```
$ cd <YOCTO_DIRECTORY>
$ source setup-environment -m <MACHINE>
$ bitbake a2bapp-linux
$ bitbake adsp-sc5xx-ramdisk -C compile
```

Here set the <MACHINE> with one of the following based on your platform.

adsp-sc589-mini

adsp-sc598-som-ezkit

<YOCTO\_DIRECTORY> is the path where Yocto for SC5xx is installed.

**Note:** Recipe “a2b\_linux” must be cleaned and rebuilt if there are any code modifications using commands as shown below:

```
$ bitbake -ccleansstate a2bapp-linux
$ bitbake a2bapp-linux
```

4. Since A2b application is added as recipe in the ramdisk image, there is no need to copy the ARM executable from Host machine to target.

## 6.3 Makefile based build for SC5XX

To build the Makefile project execute the following steps.

1. Open a terminal and give following command to set up the environment path

```
$ source /opt/poky/X.Y.Z/environment-setup-armv7at2hf-neon-poky-linux-gnueabi
```

For SC598, use following command

```
$ source /opt/poky/X.Y.Z/environment-setup-aarch64-poky-linux
```

2. Navigate to folder which contains Makefiles for building the stack and demo application

```
$ cd /opt/analog/a2b-software/X.Y.Z/Target/examples/demo/a2b-linux/<a2b-adsp-sc5xx-linux>/Makefiles
```

X.Y.Z refers to the version of Poky distribution. It is 3.1.8 for SC598. Here use following directory <a2b-adsp-sc5xx-linux> based on your platform

a2b-adsp-sc589-mini-linux

a2b-adsp-sc598-linux

3. Execute the following command to build the application.

```
$ sudo make -f <Makefile-SC5XX linux>
```

4. The various build artifacts and temporary folders are created in the folder as explained in Table 4.

**Table 4: Makefile Build Output Folders**

Output folder	Purpose
build	Intermediate build objects placed here
staging/bin	Contains the demo application a2bapp-linux(binary file which should be moved to target to execute application)
staging/include	All Stack include files are placed here
staging/lib	All static libraries built for the Stack can be found here.

To clean previous build artifacts, execute the following command

```
# sudo make -f <Makefile-SC5XX linux> clean
```

Use the make file <Makefile-SC5XX linux> based on your platform

Makefile-SC589-mini linux

Makefile-SC598 linux

**Note:** This method creates application binary (a2bapp-linux) in staging/bin which shall be moved to target by using tftp command (discussed in later sections).

5. Open a terminal in Linux host and transfer the *a2bapp-linux* demo application binary available in “/opt/analog/a2b-software/X.Y.Z/Target/examples/demo/a2b-linux/<a2b-adsp-sc5XX-linux>/Makefiles/staging/bin” to “tftpboot” folder.

```
$ cd /opt/analog/a2b-software/X.Y.Z/Target/examples/demo/a2b-linux/<a2b-adsp-sc5XX-linux>/Makefiles/staging/bin
$ sudo cp a2bapp-linux /tftpboot
```

6. Use tftp command in the Minicom terminal to transfer the *a2bapp-linux* demo application binary over Ethernet to target.

```
# tftp -gr a2bapp-linux <HOST_IPADDR>
```

*<HOST\_IPADDR> is IP address of host PC(Open the terminal and give “ifconfig” command in host PC to find IP Address of Host PC E.g.: 192.168.0.1)*

The above commands copy the application to the target. Verify whether application binary is copied to target by executing “ls” command

7. In the target console run the demo application

```
# chmod 777 a2bapp-linux
# ./a2bapp-linux
```

## 6.4 Makefile based build for a different platform

To build the demo application for a different platform the guidelines to be followed are described here.

1. The top level Makefile *Makefile-SC5xx.linux* and associated component Makefiles are available in the in /opt/analog/a2b-software/X.Y.Z/Target/examples/demo/a2b-linux/a2b-adsp-sc5xx-linux/Makefiles folder from where the make based build is triggered. This top level Makefile is used for building the A2B stack and associated libraries, plugins, and applications. The top level Makefile invokes another component Makefiles namely *Makefile.linux* for every component to be built. The various component Makefiles folders and the sources they build is explained below to facilitate easy porting to a new platform.
  - a2bstack
    - Contains the Makefile to build the generic or target agnostic portions of the A2B Software Stack. The sources for the same are located in /opt/analog/a2b-software/X.Y.Z/Target/a2bstack/a2bstack/src.

- **a2bstack-pal**
    - Contains the Makefile to build the platform adaptation layer (PAL) for the Linux A2B Software Stack Implementation. The sources for the same are located in */opt/analog/a2b-software/X.Y.Z/Target/examples/demo/a2b-linux/a2b-adsp-sc5xx-linux/a2b-app-linux\_Core0/a2bstack-pal*. The PAL for Linux implements the timer, i2c, memory manager, timer and logging functionalities required by stack. These are implemented using Linux system calls & C library calls and is portable across platforms running Linux. Hence user is not expected to modify the same.
  - **a2bstack-protobuf**
    - Contains the Makefile to build the a2bstack-protobuf component containing the source code for the A2B Bus Description Data (BDD) schema and the Google Protobuf implementation called Nanopb. It also contains routines to parse bus configuration format generated from Sigma Studio to BDD structure. The sources for the same are located in */opt/analog/a2b-software/X.Y.Z/Target/a2bstack/a2bstack-protobuf/src*.
  - **a2bplugin-master**
    - Contains the Makefile to build the sources for the A2B Software Stack master node plugin. The A2B network discovery algorithms and line fault diagnostics are encapsulated within these sources. The sources are located in */opt/analog/a2b-software/X.Y.Z/Target/a2bstack/a2bplugin-master/src*.
  - **a2bplugin-slave**
    - Contains the Makefile to build the sources for a simple A2B Software Stack slave node plugin. These sources are a trivial example of a slave plugin for use as a launching pad for developing custom plugins. The sources are located in */opt/analog/a2b-software/X.Y.Z/Target/a2bstack/a2bplugin-slave/src*
  - **a2baplinux**
    - Contains the Makefile to build the sources for a sample A2B application for Linux. This application performs discovery of the A2B network depending on the network configuration provided. This application utilizes both the A2B Stack Software (a2bstack), Linux PAL (a2bstack-platform). The sources for the same are located in */opt/analog/a2b-software/X.Y.Z/Target/examples/demo/a2b-linux/a2b-adsp-sc5xx-linux/a2b-app-linux\_Core0/app/*.
2. By default, the top level Makefile *Makefile-SC5xx.linux* is configured to use the *arm-linux-gnueabi-gcc* cross compiler provided by Linux add-in for SC5xx. Modify the same and provide the compiler for your platform. Also, the Makefiles by default use GNU tools for various commands. The important parameters in the top level Makefile that need to be commonly modified are listed in Table 5.

**Table 5: Top level Makefile parameters**

Parameter	Explanation
INSTALL_TOP	Default: \$(CURDIR)/staging Defines where to install the various build artifacts (libraries, executables, etc.).
STACK_ROOT	Default: \$(CURDIR)/../../../../../a2bstack Defines where the sources for the various components of the core A2B stack are available. By default available in <i>/opt/analog/a2b-software/X.Y.Z/Target/a2bstack/</i>
APP_ROOT	Default: \$(CURDIR)/../a2b-app-linux_Core0 Defines where the sources for the a2b demo application, a2bstack PAL on Linux and platform specific A2B stack configuration for the platform are available. By default for SC5XX available in <i>/opt/analog/a2b-software/X.Y.Z/Target/examples/demo/a2b-linux/a2b-adsp-sc5xx-linux/a2b-app-linux_Core0</i>
BUILD_DIR	Default: \$(CURDIR)/build Defines where various intermediate build objects are placed (objects, libraries, etc.).
CFLAGS	Compiler flags
CC	Target Compiler
AR	Target Compiler
MKDIR	This value should point to an appropriate utility that can build a recursive directory structure similar to the GNU “mkdir -p” command.
RM	This value should point to an appropriate utility that can recursively remove directories and files like the GNU “rm” command.
INSTALL	This value should point to an appropriate utility that can install files like the GNU “install -p” command
INSTALL_DATA	This value should point to an appropriate utility that can install files with write permissions disabled like the GNU “install -p -m 0644” command.
INSTALL_EXEC	This value should point to an appropriate utility that can install files with execute permissions set like the GNU “install -p -m 0755” command.
SUBDIRS	Provides the path to the various components built by this Makefile.

3. The Stack must be built and installed in a particular order to satisfy include dependencies components. The top level Makefile establishes this order. Even if some artefact is to be removed user must ensure the following build order.  
*a2bstack >a2bstackprotobuf>a2bpluginmaster>Slave Plugin or PAL*
4. The platform specific configuration files of the stack are located in */opt/analog/a2b-software/X.Y.Z/Target/examples/demo/a2b-linux/a2b-adsp-sc5xx-linux/a2b-app-*

*linux\_Core0/a2bstack-pal/platform/a2b*. These files are configured by default for ADSP-SC5xx. It should be inspected by user and modified to suit the platform. Specifically, the files to be modified are described below.

- a) The Linux PAL is configurable through the platform configuration file *platform.h*. User should modify this file to suit his platform specifications. Specifically, the Linux PAL uses the `ioctl` kernel API interface for all I2C transactions. The `A2B_CONF_DEFAULT_I2C_DEVICE_PATH` should point to the actual i2c adapter connected to the A2B chip in the platform. These values are set by default for SC5XX platform.
- b) The memory configuration of stack is controlled by the *conf.h* file. Specifically, in this file the `A2B_CONF_MEMORY_ALIGNMENT` and `A2B_CONF_POINTER_SIZE` settings should be set correctly for proper operation of the stack.

## 7 SC589-Mini Linux Demo

### 7.1 Demo Set Up

The sample demo can be run using ADSP-SC589-Mini as the host. In this case the demo application running on Linux in ARM core of host processor controls the discovery and programming of A2B nodes in the system. The audio routing application in SHARC is responsible for initializing clock to the A2B transceiver/codecs and controls the routing of audio communicated from A2B chip and codecs. The block diagram of a 3 node A2B system with ADSP-SC589-Mini as Host is shown in Figure 4.

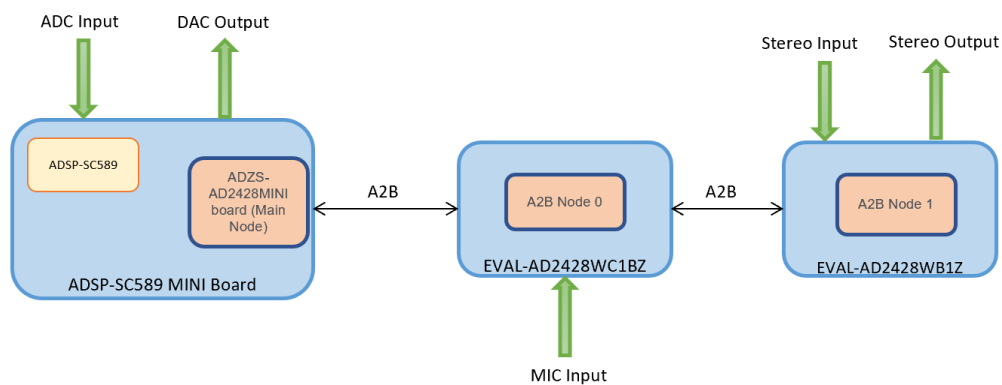


Figure 4: Demo block diagram

### 7.2 Connections

To run the sample demo, the following setup connections are to be made as shown in Figure 5

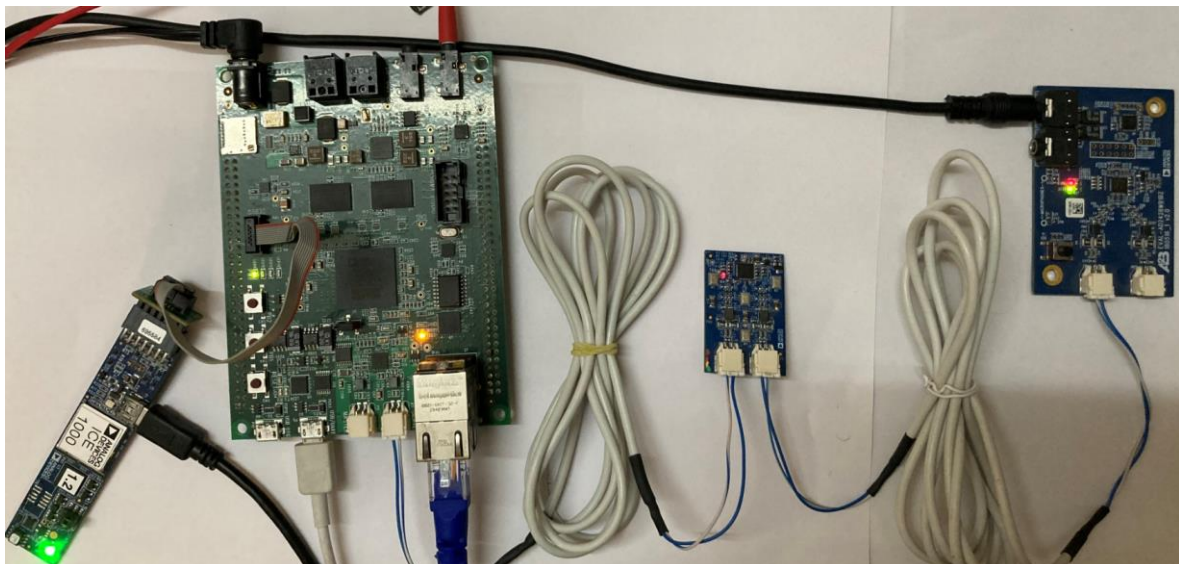


Figure 5: Demo hardware connections

### 7.2.1 Evaluation boards

In this demo ADSP-SC589-Mini acts as A2B Host and AD2428 transceiver available on ADSP-SC589-Mini EZ-Kit acts as master node. EVAL-AD2428WC1BZ and EVAL-AD2428WB1BZ act as Slave nodes. Hence AD2428 master node on ADSP-SC589-Mini board is to be connected to the AD2428 Slave node in EVAL-AD2428WC1BZ board. Connect twisted-pair wire between the “B” slave node connector P7 on the ADSP-SC589-Mini Master board and the “A” connector on the EVAL-AD2428WC1BZ Slave board. Connect the “B” side connector on EVAL-AD2428WC1BZ Slave board to the “A” connector on EVAL-AD2428WB1BZ board.

### 7.2.2 Ethernet and UART Connection to Linux Host

- Connect an Ethernet cable (RJ45) from the J3 connector on ADSP-SC589-Mini EZ-Kit to Ethernet port of the Linux host. The Ethernet connection is required to transfer demo application binary cross-compiled in Linux host to the filesystem of Linux running on ADSP-SC589-Mini EZ-Kit.
- Connect the USB Micro-B to Standard-A cable from the P6 connector on ADSP-SC589-Mini EZ-Kit to the USB connector of Linux host. The UART connection is required to view the console output from u-boot (bootloader) and configure the boot parameters. The Linux console output is also available on UART and the demo application is to be run from this console.

### 7.2.3 Audio In/Out

- Connect an audio source (e.g., output from an iPod) to ‘Stereo Input’ port of AD2428WB1BZ board.
- Connect an audio sink (e.g., active speakers) to ‘Stereo Output’ port of AD2428WB1BZ board.
- Connect an audio sink (e.g., active speakers) to Headphone Connector (J2) port of ADSP-SC589-Mini EZ-Kit.

## 7.3 Installing the Yocto Sources for ADSP-SC589-Mini

To install Yocto sources give below commands. Refer to “Building the Image” section in [9].



```
$ mkdir ~/griffin
$ cd ~/griffin
$ mkdir bin
$ curl http://commodatastorage.googleapis.com/git-repo-downloads/repo > ./bin/repo
$ chmod a+x ./bin/repo
$ ./bin/repo init \
  -u https://github.com/analogdevicesinc/lnxdsp-repo-manifest.git \
  -b release/yocto-2.1.0 \
  -m release-yocto-2.1.0.xml
$ ./bin/repo sync
$ source setup-environment -m adsp-sc589-mini
$ bitbake adsp-sc5xx-ramdisk
```

**Note:** Make sure that branch used is “release/yocto-2.1.0” while Installing Sources from repository as SC589-Mini platform is tested with that source available in that branch.

## 7.4 Booting Linux on SC589-Mini

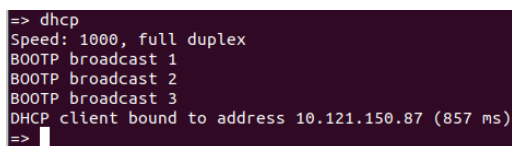
Ensure that the Jumper JP1 is connected to Pin 1 and 2

1. Ensure that serial and Ethernet cable is connected from the board to the Linux host.
2. Open a terminal in Linux and run minicom. For steps on setting up minicom refer to “Installing Required Packages” section in Yocto Linux Getting Started page [9].
3. Copy the U-Boot binary files to the /tftpboot directory of Host PC as below

```
$ cp tmp/deploy/images/adsp-sc589-mini/u-boot-sc589-mini.ldr /tftpboot
```

4. If your network **supports** DHCP, then run

```
=> dhcp
```

A terminal window with a dark background showing the output of the 'dhcp' command. The text is as follows:

```
=> dhcp
Speed: 1000, full duplex
BOOTP broadcast 1
BOOTP broadcast 2
BOOTP broadcast 3
DHCP client bound to address 10.121.150.87 (857 ms)
=> █
```

**Figure 6: Output of DHCP command in terminal**

5. If your network **does NOT** support DHCP, run:

```
=> set ipaddr <ADDR>
```

Where <ADDR> is the IP address you want to assign.

6. Next, run the U-Boot update command to copy the U-Boot loader file from the host PC to the target board, and write it into flash:

```
=> run update
```

7. U-boot should now load, and the prints should be seen on serial console and Press any key for interrupting U-Boot autoboot process as shown in Figure 7.

```
U-Boot 2020.10 (Sep 29 2022 - 07:08:35 +0000)

CPU:   ADSP ADSP-SC589-0.1 (Detected Rev: 1.1) (spi flash boot)
VCO: 450 MHz, Cclk0: 450 MHz, Sclk0: 112.500 MHz, Sclk1: 112.500 MHz, DCLK: 450 MHz
OCLK: 150 MHz
Model: ADI sc589-mini
I2C:   ready
DRAM:  224 MiB
MMC:   SC5XX SDH: 0
Loading Environment from SPIFlash...
SF: Detected is25lp512 with page size 256 Bytes, erase size 64 KiB, total 64 MiB
*** Warning - bad CRC, using default environment

In:    serial@0x31003000
Out:   serial@0x31003000
Err:   serial@0x31003000
other init
Net:   dwmac.3100c000
Hit any key to stop autoboot:  0
=> 
```

Figure 7: u-boot console output

8. To instruct Linux kernel not to perform SEC initialization, as it is done in Core1, execute the following command to modify the ramargs environment variable

```
=> editenv ramargs
```

And add “enable\_sec=no” at the end to modify *ramargs*, also print the ramargs environment variable to verify the change as shown in Figure 8.

```
=> pri ramargs
ramargs=setenv bootargs root=/dev/mtdblock2 rw rootfstype=jffs2 earlyprintk=serial,uart0,115200 console=ttySC0,115200
=> editenv ramargs
edit: setenv bootargs root=/dev/mtdblock2 rw rootfstype=jffs2 earlyprintk=serial,uart0,115200 console=ttySC0,115200 enable_sec=no
=> pri ramargs
ramargs=setenv bootargs root=/dev/mtdblock2 rw rootfstype=jffs2 earlyprintk=serial,uart0,115200 console=ttySC0,115200 enable_sec=no
=> 
```

Figure 8: Editing boot arguments in u-boot

9. Follow the steps mentioned in Section 5 to enable Remote Processor on corresponding Yocto build.
10. Follow the steps described in Section 6.1 to enable platform specific SDK installation and Section 6.2 to build A2B Demo application as part of Yocto build’s root file system.

11. Copy the *fitImage* and *rootfs* files from Yocto build directory to '/tftpboot' folder of TFTP Server machine.

```
$ cp tmp/deploy/images/adsp-sc589-mini/fitImage /tftpboot
$ cp tmp/deploy/images/adsp-sc589-mini/adsp-sc5xx-ramdisk-adsp-sc589-mini.cpio.xz.
u-boot /tftpboot/ramdisk.cpio.xz.u-boot
```

12. Load the *fitImage* file kernel image (includes Kernel, DTBs and overlays) over Ethernet from the Linux host to RAM and boot by giving below command.

```
=>run ramboot
```

Once Linux kernel has booted up login with 'root' as username and "adi" as password.

## 7.5 Running the A2B SHARC(Core1) application

Follow below commands to run the SHARC application on Core1.

1. The firmware components are stored in **/lib/firmware/** folder. In the minicom terminal enter following command to set the firmware name through sysfs

```
root@adsp-sc589-ezkit:~# cd /lib/firmware
root@adsp-sc589-ezkit:/lib/firmware# echo a2baudiorouter-app-sc589-mini_Core1.ldr >
/sys/class/remoteproc/remoteproc0/firmware
```

2. Start the SHARC Application by below command

```
root@adsp-sc589-ezkit:~# echo start > /sys/class/remoteproc/remoteproc0/state
```

//This command will bring Core1 out of reset and load ldr in Core1. The application in Core1 will start running.

After setting remoteproc0 state to start will bring Core1 out of reset and load ldr in Core1. The application in Core1 will start running as shown in Figure 9

```
root@adsp-sc589-mini:/lib/firmware# echo a2baudiorouter-app-sc589-mini_Core1.ldr > /sys/class/remoteproc/remoteproc0/firmware
root@adsp-sc589-mini:/lib/firmware# echo start > /sys/class/remoteproc/remoteproc0/state
remoteproc remoteproc0: powering up core1-rproc
remoteproc remoteproc0: Booting fw image a2baudiorouter-app-sc589-mini_Core1.ldr, size 89008
remoteproc remoteproc0: remote processor core1-rproc is now up
root@adsp-sc589-mini:/lib/firmware#
```

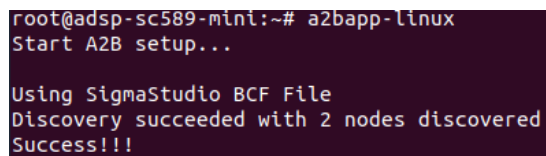
Figure 9: Load Core1 application by remote proc (on SC589-Mini)

## 7.6 Running the A2B ARM(Core0) demo application

Give “a2bapp-linux” command in the minicom terminal to run the Core0 application.

```
# a2bapp-linux
```

The output on console should show “Discovery Succeeded with 2 nodes discovered” indicating successful discovery as below:



```
root@adsp-sc589-mini:~# a2bapp-linux
Start A2B setup...

Using SigmaStudio BCF File
Discovery succeeded with 2 nodes discovered
Success!!!
```

**Figure 10: A2B Linux demo application on ADSP-SC589 Mini**

After running the demo, the Mic input from Slave node 0 should play at Slave node1’s HP OUT. Audio Input provided at Slave Node 1’s LINE IN should play at Target board’s DAC (HEADPHONES).

## 8 ADSP-SC598 Linux Demo

### 8.1 Demo Set Up

The sample demo can be run using ADSP-SC598 as the host processor. In this case the host processor controls the discovery and programming of A2B nodes in the system. The block diagram of a 3 node A2B system with SC598 as Host is shown as in Figure 11

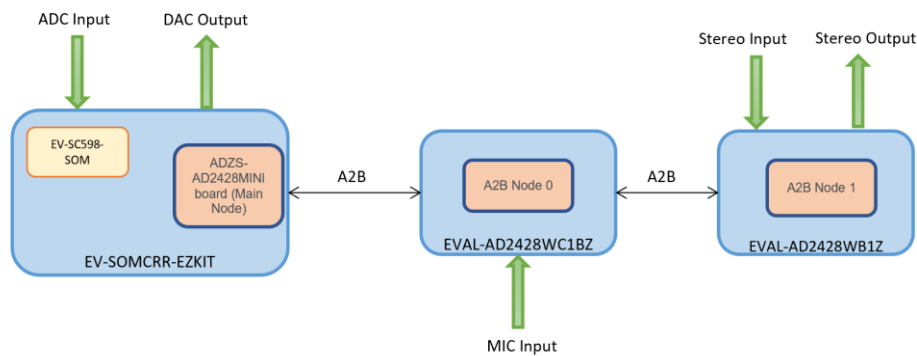


Figure 11: A2B system with ADSP-SC598 as Host

### 8.2 Connections

Mount ADZS-AD2428MINI Board with I2C address 0x68 on Connector J10 of EV-SOMCRR-EZKIT.

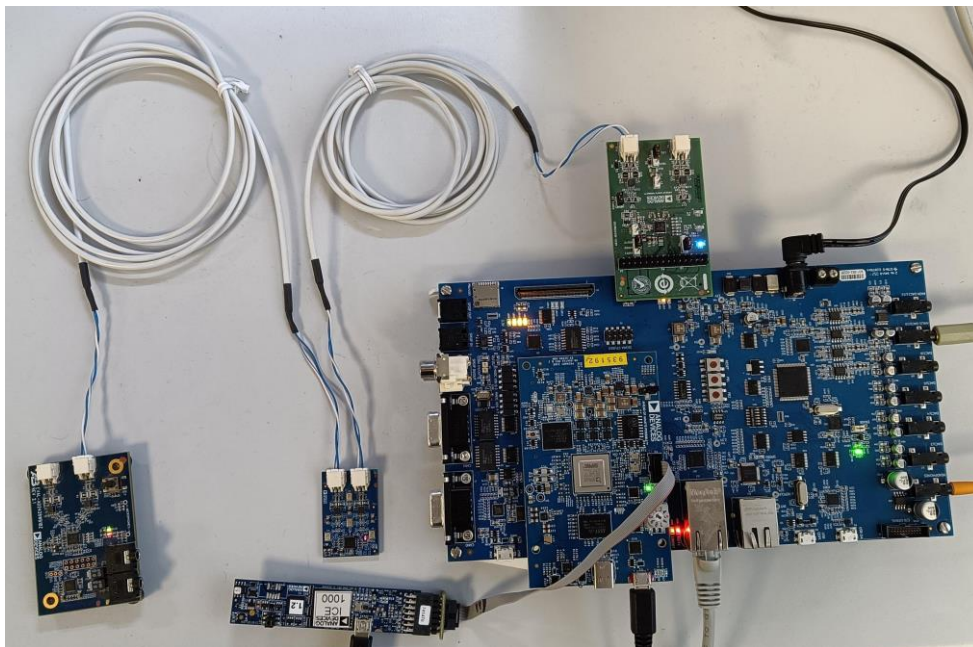


Figure 12: ADSP-SC598 Demo Hardware Connections

### 8.2.1 Evaluation boards

In this demo ADSP-SC598 acts as Target board and AD2428 Mini transceiver connected to EV-SOMCRR-EZKIT Board on J10 interface acts as master node. EVAL-AD2428WC1BZ and EVAL-AD2428WB1BZ act as Slave nodes and are connected to P10 interface on AD2428MINI Node. Hence AD2428 master node (P10) is to be connected to EVAL-AD2428WC1BZ board (J8) using twisted pair cable and EVAL-AD2428WC1BZ board (J7) is connected to EVAL-AD2428WB1BZ (J8) as shown in Figure 12.

### 8.2.2 Ethernet and UART Connection to Linux Host

- Connect an Ethernet cable (RJ45) from the J13 connector on EV-SOMCRR-EZKIT Board to Ethernet port of the Linux host. The Ethernet connection is required to transfer demo application binary cross-compiled in Linux host to the filesystem of Linux running on ADSP-SC598 EZ-Kit.
- Connect the USB Micro-B to Standard-A cable from the P6 connector on ADSP-SC598 EZ-Kit to the USB connector of Linux host. The UART connection is required to view the console output from u-boot (bootloader) and configure the boot parameters. The Linux console output is also available on UART and the demo application is to be run from this console.

### 8.2.3 Audio In/Out

- Connect an audio source (e.g., output from an iPod) to 'Stereo Input' port of AD2428WB1BZ board.
- Connect an audio sink (e.g., active speakers) to 'HP OUT' port of AD2428WB1BZ board.
- Connect an audio sink (e.g., active speakers) to Headphone Connector (J4) port of EV-SOMCRR-EZKIT Board.

## 8.3 Installing the Yocto Sources for ADSP-SC598

To install Yocto sources give below commands. Refer to "Building the Image" section in [10].

```

$ mkdir ~/gxp2
$ cd ~/gxp2
$ mkdir bin
$ curl http://commodatastorage.googleapis.com/git-repo-downloads/repo > ./bin/repo
$ chmod a+x ./bin/repo
$ ./bin/repo init \
  -u https://github.com/analogdevicesinc/lnxdsp-repo-manifest.git \
  -b release/yocto-2.1.0 \
  -m release-yocto-2.1.0.xml
$ ./bin/repo sync
$ source setup-environment -m adsp-sc598-som-ezkit
$ bitbake adsp-sc5xx-ramdisk

```

**Note:** Make sure that branch used is “release/yocto-2.1.0” while Installing Sources from repository as SC598 platform is tested with that source available in that branch.

## 8.4 Booting Linux on SC598

To Load Linux on ADSP-SC598 from RAM follow the below steps

1. Ensure that serial and Ethernet cable is connected from the board to the Linux host.
2. Open a terminal in Linux and run minicom. For steps on setting up minicom refer to “Configuring Minicom” section in Yocto Linux Getting Started guide [10].
3. To install U-boot in target board, copy below files into /tftpboot folder as referred in [10].

```

$ cp tmp/deploy/images/adsp-sc598-som-ezkit/stage1-boot.ldr /tftpboot/
$ cp tmp/deploy/images/adsp-sc598-som-ezkit/stage2-boot.ldr /tftpboot/
$ cp tmp/deploy/images/adsp-sc598-som-ezkit/fitImage /tftpboot/
$ cp tmp/deploy/images/adsp-sc598-som-ezkit/adsp-sc5xx-minimal-adsp-sc598-som-ezkit.jffs2 /tftpboot
$ cp tmp/deploy/images/adsp-sc598-som-ezkit/adsp-sc5xx-ramdisk-adsp-sc598-som-ezkit.cpio.xz.u-boot /tftpboot/ramdisk.cpio.xz.u-boot

```

4. In the U-Boot console, set the TFTP Server IP

```

=> setenv serverip <SERVERIP>
=> dhcp

```

```

=> dhcp
Speed: 1000, full duplex
BOOTP broadcast 1
BOOTP broadcast 2
BOOTP broadcast 3
DHCP client bound to address 10.121.150.87 (857 ms)
=>

```

Figure 13: ADSP-SC598 u-boot console output

- If the network **does NOT** support DHCP, in the U-Boot console configure the board IP address and remove the “run init\_ethernet;” from the “update\_spi\_sc598” command.

```
=> setenv ipaddr <IPADDR>
=> edit update_spi_sc598
=> edit: <remove "run init_ethernet;" from here> sf probe ${sfdev}; sf erase 0 ${sfsize};
run update_spi_uboot; run update_spi_fit; run update_spi_rfs; sleep 3; saveenv
=> run update_qspi_sc598
```

- The U-Boot console is used to copy U-Boot (SPL and Proper), the minimal root filesystem image and the fitImage (which contains the kernel image and dtb file) into RAM and then write them to Flash by running below command

```
=> run update_qspi_sc598
```

- U-boot should now load, and the prints should be seen on serial console as shown in Figure 14. Hit any key for interrupting U-Boot autoboot process.

```
U-Boot SPL 2020.10 (Sep 29 2022 - 07:08:35 +0000)
ADI Boot Mode: 1 (QSPI Master)
Trying to boot from BOOTROM

U-Boot 2020.10 (Sep 29 2022 - 07:08:35 +0000)

Model: ADI sc598-som-ezkit
        Watchdog enabled
I2C:    ready
DRAM:   224 MiB
MMC:    mmc@310C7000: 0
Loading Environment from SPIFlash...
SF: Detected is25lp512 with page size 256 Bytes, erase size 64 KiB, total 64 MiB
OK
In:     serial@0x31003000
Out:    serial@0x31003000
Err:    serial@0x31003000
Net:    eth0: eth@0x31040000
Hit any key to stop autoboot:  0
=>
```

Figure 14: ADSP-SC598 U-Boot console output

- To instruct Linux kernel not to perform SEC initialization execute the following command to modify the ramargs environment variable.

```
=>editenv ramargs
```

And add “*enable\_sec=no*” at the end to modify *ramargs* and also print ramargs environment variable to verify the change as shown in Figure 15.

```
=>set bootargs root=/dev/mtdblock2 rw rootfstype=jffs2 clk_in_hz=(25000000) console=
ttySC0,57600 mem=112M enable_sec=no
```



```
=> editenv ramargs
edit: setenv bootargs root=/dev/mtdblock2 rw rootfstype=jffs2 earlycon=adi_uart,0x31003000 console=ttySC0,115200 enable_sec=no
=> pri ramargs
ramargs=setenv bootargs root=/dev/mtdblock2 rw rootfstype=jffs2 earlycon=adi_uart,0x31003000 console=ttySC0,115200 enable_sec=no
=>
```

Figure 15: Editing boot arguments in u-boot

9. Follow the steps mentioned in Section 5 to enable Remote Processor on corresponding Yocto build.
10. Follow the steps described in Section 6.1 to enable platform specific SDK installation and Section 6.2 to build A2B Demo application as part of Yocto build's root file system.
11. Copy the *fitImage* and *rootfs* files from Yocto build directory to '/tftpboot' folder of TFTP Server machine.

```
$ cp tmp/deploy/images/adsp-sc598-som-ezkit/fitImage /tftpboot
$ cp tmp/deploy/images/adsp-sc598-som-ezkit/adsp-sc5xx-ramdisk-adsp-sc598-som-ezki
t.cpio.xz.u-boot /tftpboot/ramdisk.cpio.xz.u-boot
```

12. To boot Linux from TFTP server, run the below command.

```
=>run ramboot
```

13. Once Linux kernel has booted up login with '*root*' as username and "adi" as password.

## 8.5 Running the A2B SHARC(Core1) application

Follow commands to run the SHARC application on Core1.

1. The firmware components are stored in **/lib/firmware/** folder. In the minicom terminal enter following command to set the firmware name through sysfs

```
root@adsp-sc598-ezkit:~# cd /lib/firmware
root@adsp-sc598-ezkit:/lib/firmware# echo a2baudiorouter-app-sc598_Core1.ldr > /sys/
class/remoteproc/remoteproc0/firmware
```

2. Start the SHARC Application by below command

```
root@adsp-sc598-ezkit:~# echo start > /sys/class/remoteproc/remoteproc0/state
```

//This command will bring Core1 out of reset and load ldr in Core1. The application in Core1 will start running.

After setting remoteproc0 state to start will bring Core1 out of reset and load ldr in Core1. The application in Core1 will start running as shown below.

```

root@adsp-sc598-som-ezkit:~# cd /lib/firmware/
root@adsp-sc598-som-ezkit:/lib/firmware# echo a2baudiorouter-app-sc598_Core1.ldr > /sys/class/remoteproc/remoteproc0/firmware
root@adsp-sc598-som-ezkit:/lib/firmware# echo start > /sys/class/remoteproc/remoteproc0/state
[ 42.581460] remoteproc remoteproc0: powering up core1-rproc
[ 42.587305] remoteproc remoteproc0: Booting fw image a2baudiorouter-app-sc598_Core1.ldr, size 102644
[ 45.405199] adi_remoteproc 28240000.core1-rproc: Core1 rpmsg init timeout, probably not supported.
[ 45.414015] virtio_rpmsg_bus virtio0: rpmsg host is online
[ 45.424549] remoteproc0#vdev0buffer: registered virtio0 (type 7)
[ 45.431191] remoteproc remoteproc0: remote processor core1-rproc is now up
root@adsp-sc598-som-ezkit:/lib/firmware#

```

Figure 16: Load Core1 application by remoteproc (on SC598)

## 8.6 Running the A2B ARM(Core0) demo application

Give “a2bapp-linux” command in the minicom terminal to run the Core0 application as shown in Figure 17. The output on console should show “Discovery Succeeded with 2 nodes discovered” message indicating successful discovery as below:

```

root@adsp-sc598-som-ezkit:~#
root@adsp-sc598-som-ezkit:~# a2bapp-linux
Start A2B setup...

Using SigmaStudio BCF File
NODE DISCOVERY: nodeType=2 nodeAddr=1 discoveryCompleteCode=0
NODE DISCOVERY: nodeType=2 nodeAddr=2 discoveryCompleteCode=0
Discovery succeeded with 2 nodes discovered
Success!!!

```

Figure 17: A2B Linux demo application on ADSP-SC598

After running the demo, the Mic input from Slave node 0 should play at Slave node1’s HP OUT. Audio Input provided at Slave Node 1’s LINE IN should play at Target board’s DAC (HEADPHONES).

## 9 Demo Application and A2B stack features on Linux

Demo application *a2bapp-linux* and a2b stack in Linux supports several features that can be invoked by the user using the various command line and run time options. Also, several other features can be configured during build time.

### 9.1 Build Time options

#### 9.1.1 Utilizing a custom schematic

The default demo application uses the bus configuration file *adi\_a2b\_busconfig.c* to perform bus discovery. This file corresponds to the 3-node sample node schematic in */opt/analog/a2b-software/X.Y.Z/Target/Schematics/SC58x/A2BSchematics/adi\_a2b\_3NodeSampleDemoConfig.dsproj*. To use a different schematic user has to first export the bus configuration C file from Sigma Studio. Refer to Sigma Studio User Guide [5] for more details on creating and exporting a custom schematic from Sigma Studio. The default *adi\_a2b\_busconfig.c* should be replaced with this file in the */opt/analog/a2b-software/X.Y.Z/Target/examples/demo/a2b-linux/a2b-adsp-sc5xx-linux/a2b-app-linux\_Core0/app/* and application rebuilt.

#### 9.1.2 Trace file and Level

By default, trace feature is disabled in Linux. The steps to log trace messages from A2B Stack Software are described below.

1. To enable trace feature, define the macro *A2B\_FEATURE\_TRACE* in the file *features.h*. The trace file and trace debug level can be set in the file *platform.h* with the macros *A2B\_CONF\_DEFAULT\_TRACE\_CHAN\_URL* and *A2B\_CONF\_DEFAULT\_TRACE\_LVL*. The *features.h* and *platform.h* files are to be modified in */opt/analog/a2b-software/X.Y.Z/Target/examples/demo/a2b-linux/a2b-adsp-sc5xx-linux/a2b-app-linux\_Core0/a2bstack-pal/platform/a2b*.
2. *A2B\_CONF\_DEFAULT\_TRACE\_LVL* specifies the default level of trace messages to be captured. The levels are defined based on the severity of messages. This also specifies the domains for e.g.: stack, plugin etc from which to capture messages. The default value is set to *(A2B\_TRC\_DOM\_ALL | A2B\_TRC\_LVL\_DEBUG | A2B\_TRC\_LVL\_INFO |*

*A2B\_TRC\_LVL\_DEFAULT* | *A2B\_TRC\_LVL\_ALL*). This causes all levels of messages from all domains to be captured. For a description of the different trace domains and levels refer to the A2B Software Stack API reference document.

3. The *A2B\_CONF\_DEFAULT\_TRACE\_CHAN\_URL* specifies the channel where the trace output is logged in terms of an URL. By default, it is set to *"file://TraceFile.txt"* which causes the trace messages to be stored in the file *TraceFile.txt* in the same folder. Build and run the demo application on the ADSP-SC5XX Ez-Boards. To view the log of trace messages, stop the application by typing 'q' on the console. The trace file namely *"TraceFile.txt"* is created in the folder from where the application is run. Open the file in a text editor to analyze the messages. Note that the next time the application is run old trace messages are removed in the existing file and written with the new messages. To view the trace messages on console where the application is running instead of logging it to a file, the URL should be modified to *"stdio://stdout"*.

### 9.1.3 Sequence chart feature

By default, sequence chart feature is disabled in Linux. To enable define the macro *A2B\_FEATURE\_SEQ\_CHART* in *features.h*. The *features.h* file is to be modified in */opt/analog/a2b-software/X.Y.Z/Target/examples/demo/a2b-linux/a2b-adsp-sc5xx-linux/a2b-app-linux\_Core0/a2bstack-pal/platform/a2b* folder. The sequence chart output is logged in a file. The file name can be specified as a command line argument to the demo application. Refer to section 9.2.1 for the detailed steps to generate and view the sequence chart file.

## 9.2 Command Line options

The various command line options and usage is listed when the application is run with the command line option *"-h"*. The following important command line options are supported.

### 9.2.1 Specifying Sequence chart file and generating UML diagram

The sequence chart output file can be specified with the *"-s"* command line option. The sequence chart generated can be visualized as a UML diagram using the *plantuml* utility. The steps for the same are given below.

1. Generate the sequence chart by running the application specifying the sequence file name as SequenceFile.txt. The file is created in the same directory from which the application is run on the target.

```
# ./a2baplinux -s file://SequenceFile.txt
```

2. Transfer the file from target to Linux host over Ethernet using the “scp” command .For ex: the Sequence file is at ‘/’ path in target then run the following command in Linux host to transfer file to the /opt/analog/a2b-software/X.Y.Z/Target/examples/demo/a2b-linux/a2b-adsp-sc5xx-linux folder in the host from target.

```
$ scp root@target_ip:/SequenceFile.txt /opt/analog/a2b-software/X.Y.Z/Target/examples/demo/a2b-linux/a2b-adsp-sc5xx-linux
```

3. In Linux host install JAVA sdk and graphviz utility by executing the following commands.

```
$ sudo apt-get install default-jre  
$ sudo apt-get install graphviz
```

4. Next to generate the plantuml diagram run the SeqChartProcess\_linux.sh script from the tools folder.

```
$ cd /opt/analog/a2b-software/X.Y.Z/Target/tools  
$ sudo sh SeqChartProcess_linux.sh
```

The sequence chart uml diagram namely *SequenceFile.detailed.png* is created in the /opt/analog/a2b-software/X.Y.Z/Target/examples/demo/a2b-linux/a2b-adsp-sc5xx-linux folder.

### 9.2.2 Enabling debug mode

To enable the debug mode the “-d” command line option has to be used. This causes additional information pertaining to the a2b interrupts to be displayed on the console as shown below.

```
# ./a2bapp-linux -d

Using Sigma Studio BCF file

Parsing BCF Done
Initializing master plugin
Initializing slave plugins for 2 slave nodes

Passing Peripheral configuration through discovery message
Hit 'h' for keypress help.
INTERRUPT: intrType=255 nodeAddr=-1
INTERRUPT: intrType=24 nodeAddr=-1
INTERRUPT: intrType=24 nodeAddr=-1
Discovery succeeded with 2 nodes discovered
```

Figure 18: Enabling Debug Mode in application

### 9.2.3 Running as a daemon

The application can be run in background as a daemon by specifying “-D” as a command line option as shown below.

```
# ./a2bapp-linux -D
#
```

Figure 19: Running application as a daemon

### 9.2.4 Enabling audio using A2B sound card

The application can configure and enable audio using A2B sound card by specifying the “-k” as a command line option as shown below. For this option to work, the A2B network should already be discovered successfully before executing the A2B Linux example demo application.

```
# ./a2bapp-linux -k
```

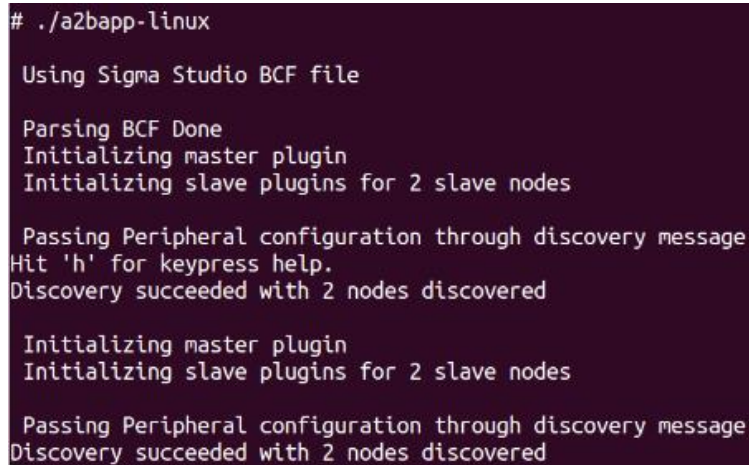
Figure 20: Enabling audio using A2B sound card

## 9.3 Run Time options

This option can be exercised when the application is running through commands typed in the console.

### 9.3.1 Rediscovery

A rediscovery of the network can be initiated by typing “d” on the console. This causes the discovery message to be sent to the stack once again from the application and the entire discovery of the network to be carried. The output on the console is as shown in below figure. This is useful when debugging line faults.



```
# ./a2bapp-linux

Using Sigma Studio BCF file

Parsing BCF Done
Initializing master plugin
Initializing slave plugins for 2 slave nodes

Passing Peripheral configuration through discovery message
Hit 'h' for keypress help.
Discovery succeeded with 2 nodes discovered

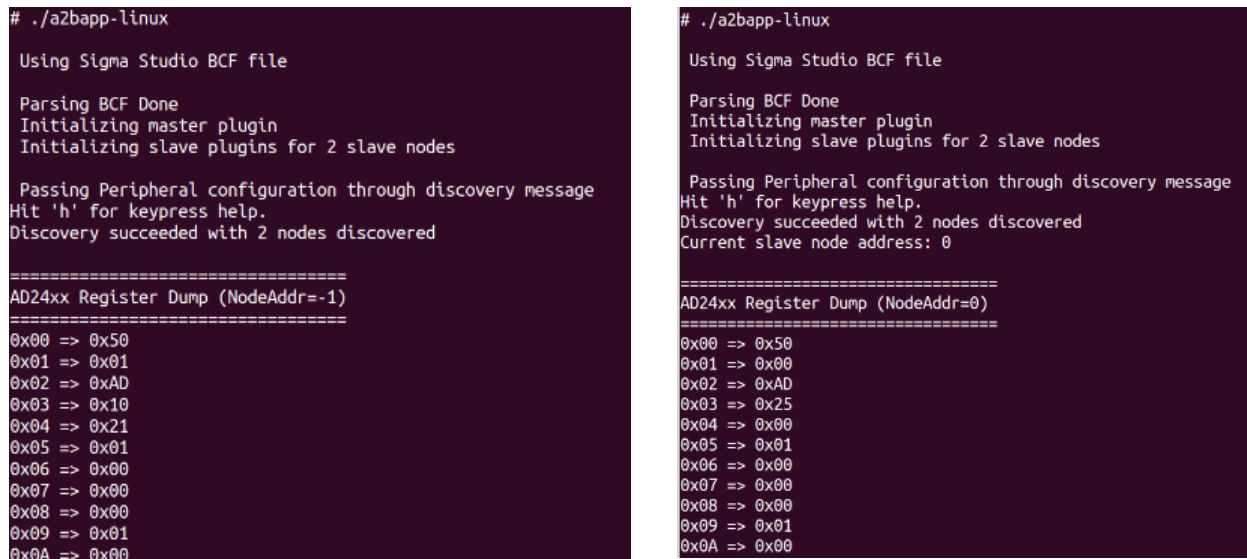
Initializing master plugin
Initializing slave plugins for 2 slave nodes

Passing Peripheral configuration through discovery message
Discovery succeeded with 2 nodes discovered
```

Figure 21: Initiating Rediscovery

### 9.3.2 Register dump of nodes

The applications support dumping of all the AD24x register values of a node on the console. The master node registers can be dumped by typing “R” key on the console. The sample output with limited register values is shown in below figure. To dump the registers of a slave node first the slave node has to be selected by pressing the node number on the console. Then based on the node selected pressing the “r” key dumps the registers of the node. A sample output with limited register values is shown below.



```
# ./a2bapp-linux

Using Sigma Studio BCF file

Parsing BCF Done
Initializing master plugin
Initializing slave plugins for 2 slave nodes

Passing Peripheral configuration through discovery message
Hit 'h' for keypress help.
Discovery succeeded with 2 nodes discovered

=====
AD24xx Register Dump (NodeAddr=-1)
=====
0x00 => 0x50
0x01 => 0x01
0x02 => 0xAD
0x03 => 0x10
0x04 => 0x21
0x05 => 0x01
0x06 => 0x00
0x07 => 0x00
0x08 => 0x00
0x09 => 0x01
0x0A => 0x00

# ./a2bapp-linux

Using Sigma Studio BCF file

Parsing BCF Done
Initializing master plugin
Initializing slave plugins for 2 slave nodes

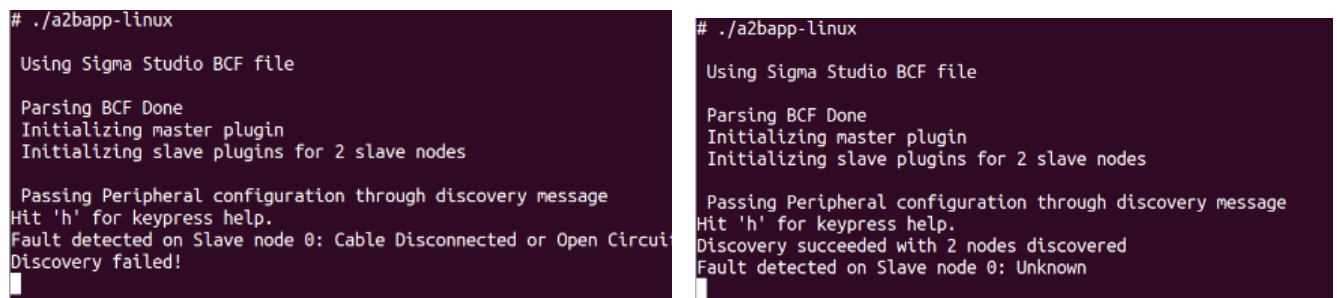
Passing Peripheral configuration through discovery message
Hit 'h' for keypress help.
Discovery succeeded with 2 nodes discovered
Current slave node address: 0

=====
AD24xx Register Dump (NodeAddr=0)
=====
0x00 => 0x50
0x01 => 0x00
0x02 => 0xAD
0x03 => 0x25
0x04 => 0x00
0x05 => 0x01
0x06 => 0x00
0x07 => 0x00
0x08 => 0x00
0x09 => 0x01
0x0A => 0x00
```

Figure 22: Master node register dump and Slave node register dump

### 9.3.3 Line Fault messages

The application supports notifying the user of any line faults that occur both during discovery and post discovery. Whenever a line fault occurs a message is printed on the console specifying the fault type and the node at which the fault occurred. Below figure shows the output when the “B” side cable is disconnected at Node 0 before running the application. As expected discovery fails and error message indicating open or disconnected cable at Node 0 is flashed. Below Figure shows the output when the cable is removed post successful discovery at Node0.



```
# ./a2bapp-linux

Using Sigma Studio BCF file

Parsing BCF Done
Initializing master plugin
Initializing slave plugins for 2 slave nodes

Passing Peripheral configuration through discovery message
Hit 'h' for keypress help.
Fault detected on Slave node 0: Cable Disconnected or Open Circuit
Discovery failed!

# ./a2bapp-linux

Using Sigma Studio BCF file

Parsing BCF Done
Initializing master plugin
Initializing slave plugins for 2 slave nodes

Passing Peripheral configuration through discovery message
Hit 'h' for keypress help.
Discovery succeeded with 2 nodes discovered
Fault detected on Slave node 0: Unknown
```

Figure 23: Line fault during discovery and Line fault post discovery



## 10 Appendix A

### 10.1 A2B Linux Command Line Tools

Using A2B Linux command line tools the A2B system can be directly controlled from a connected Linux PC without involving a microcontroller or a DSP. This mode is helpful for quick testing of the capabilities of AD24xx without the need for a microcontroller in the system.

The A2B command line tools help the user to do the following:

- Discover the A2B network using the A2B command list which is exported from SigmaStudio. Refer to **[5]** for the procedure to export the I2C command list from SigmaStudio.
- Read/write the A2B master/slave registers
- Read/Write the peripheral registers which is connected to the slave nodes.

The command line tools are available in `/opt/analog/a2b-software/X.Y.Z/Utils/linux-a2b-pc-app`. Refer to the Readme available in the `/opt/analog/a2b-software/X.Y.Z/Utils/linux-a2b-pc-app` folder to install the A2B command line tools and its usage. A block diagram of a 3 node A2B system with Linux PC as Host is shown below.

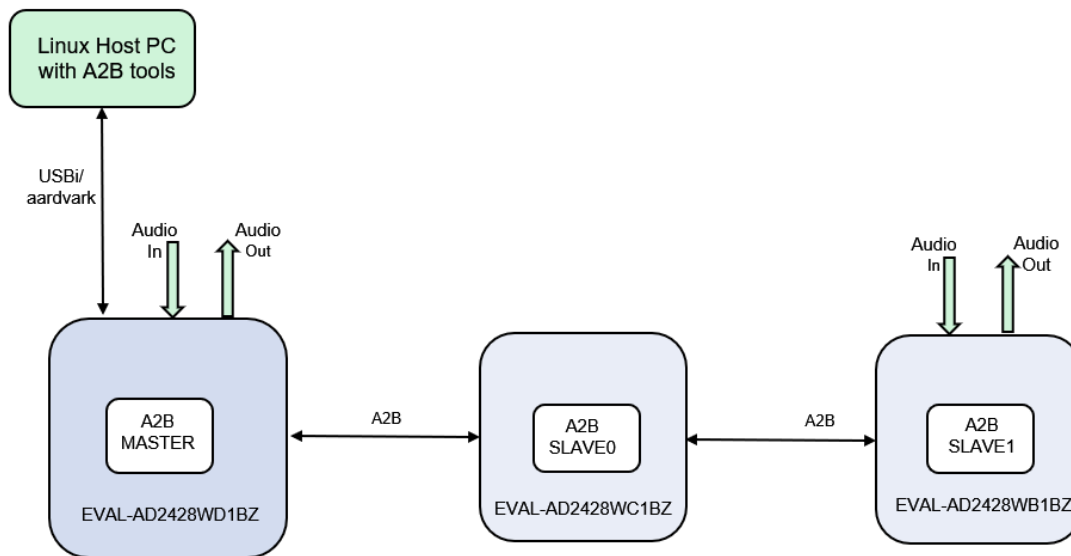


Figure 24: A2B System with Linux PC as Host

**Table 6: Terminology**

Term	Description
A2B	Automotive Audio Bus
A2B node	Refers to AD241x/AD242x.
ADI	Analog Devices, Inc.
ADC	Analog to Digital Converter
DAC	Digital to Analog Converter
DTB	Device Tree Blob. A compiled device tree binary file.
ELF	Executable and Linkable file format. Common file format for executable and object code.
FAE	Field Application Engineer
GNU	GNU's not UNIX
JTAG	Joint Test Action Group
Master Node	A2B transceiver that is connected to the host processor is considered as the master A2B node.
Slave Node	A2B Slave Transceiver with local peripherals such as speakers and microphones.
I2C	Is a multi-master single-ended serial bus used for attaching low-speed peripherals to a processor. In TWI / I2C protocol the serial data transmission is done in asynchronous mode. This protocol uses only two wires named <i>SDA</i> (serial data) and <i>SCL</i> (serial clock) for communicating between two or more ICs.
PAL	Platform Abstraction Layer. The code below this layer is platform specific.
SPORT	Synchronous Serial Peripheral Port. A serial port that supports I2S and TDM protocols.
TFTP	Trivial File Transfer Protocol. File transfer protocol that uses UDP/IP.
UART	Universal Asynchronous Receiver-Transmitter
U-boot	Universal Boot Loader. It is an open source bootloader used primarily in embedded devices.
USB	Universal Serial Bus

**Table 7: References**

Reference No.	Description
[1]	/opt/analog/a2b-software/X.Y.Z/Docs/AE_09_A2B_QuickStartGuide.pdf
[2]	<a href="http://www.analog.com/en/design-center/processors-and-dsp/evaluation-and-development-software/adswt-cces.html#dsp-overview">http://www.analog.com/en/design-center/processors-and-dsp/evaluation-and-development-software/adswt-cces.html#dsp-overview</a>
[3]	CrossCoreEmbeddedStudio_X.Y.Z_Release_Notes.pdf
[4]	<a href="https://wiki.analog.com/resources/tools-software/linuxdsp#getting_started">https://wiki.analog.com/resources/tools-software/linuxdsp#getting_started</a>
[5]	/opt/analog/a2b-software/X.Y.Z/Docs/AE_09_A2B_SigmaStudio_UserGuide.pdf
[6]	/opt/analog/a2b-software/X.Y.Z/Docs/AE_09_A2B_Stack_API_Reference.chm
[7]	<a href="http://www.analog.com/en/design-center/processors-and-dsp/evaluation-and-development-software/linuxaddin.html#dsp-overview">http://www.analog.com/en/design-center/processors-and-dsp/evaluation-and-development-software/linuxaddin.html#dsp-overview</a>
[8]	<a href="https://wiki.analog.com/resources/tools-software/linuxdsp/docs/linux-kernel-and-drivers/remoteproc/remoteproc_ldr_generate">https://wiki.analog.com/resources/tools-software/linuxdsp/docs/linux-kernel-and-drivers/remoteproc/remoteproc_ldr_generate</a>
[9]	<a href="https://wiki.analog.com/resources/tools-software/linuxdsp/docs/quickstartguide/quickstart_sc589">https://wiki.analog.com/resources/tools-software/linuxdsp/docs/quickstartguide/quickstart_sc589</a>
[10]	<a href="https://wiki.analog.com/resources/tools-software/linuxdsp/docs/quickstartguide/quickstart_sc598">https://wiki.analog.com/resources/tools-software/linuxdsp/docs/quickstartguide/quickstart_sc598</a>