

A2B SIGMASTUDIO USER GUIDE

Document Status	Approved
Approved by	ASH

ANALOG DEVICES, INC.

www.analog.com



AHEAD OF WHAT'S POSSIBLE™

ANALOG DEVICES, INC.

www.analog.com

Revision List

Table 1: Revision List

Revision	Date	Description
0.1	08.11.2016	Draft Version of 13.0.0
1.0	10.11.2016	Approved and baselined for Rel 13.0.0
1.1	16.01.2017	Updates for Rel14.0.0(Section 2.1)
1.2	18.01.2017	Addressed review comments
2.0	20.01.2017	Approved and baselined for Rel 14.0.0
2.1	21.02.2017	Updates for Rel15.0.0(section 2.2.3.2)
2.2	23.02.2017	Added Section 6 (Programming SigmaDSPover A2B)
3.0	03.03.2017	Baselined for Rel15.0.0
3.1	08.05.2017	Updates for Rel16.0.0(Section 3.2)
3.2	10.05.2017	Incorporating review comments
4.0	12.05.2017	Baselined for Rel16.0.0
4.1	28.09.2017	Updates for Rel17.0.0 (Section 3.3 and 4.1.2 added)
5.0	05.10.2017	Baselined for Rel17.0.0
5.1	15.11.2017	Updates for Rel18.0.0 (Section 2.2.2 and 2.2.3.1.1)
6.0	05.12.2017	Baselined for Rel18.0.0 Beta
6.1	07.05.2018	Updates for Rel19.0.0 (Section 2.1, 6.0, 4.1)
6.2	15.05.2018	Review comments incorporated
7.0	06.06.2018	Baselined for Rel19.0.0
7.1	22.10.2018	Updates for Rel19.1.0
7.2	25.10.2018	Review comments incorporated
8.0	31.10.2018	Baselined for Rel19.1.0
8.1	30.8.2020	Updates for Rel19.4.0
9.0	02.9.2020	Baselined for Rel19.4.0
9.1	27.04.2021	Updates for Rel19.4.2
9.2	02.05.2021	Review comments incorporated
10.0	03.05.2021	Baselined for Rel19.4.2

10.1	20.4.2022	Updated for Rel19.4.3
11.0	25.04.2022	Baselined for Rel19.4.3

Copyright, Disclaimer Statements

Copyright Information

Copyright (c) 2009-2022 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

Revision List.....	3
Copyright, Disclaimer Statements	5
Table of Contents.....	6
List of Figures	8
List of Tables.....	10
1 Introduction.....	11
2 A2B schematics components	12
2.1 Hardware configuration Tab.....	12
2.1.1 AD24xx	12
2.1.2 Communication Channel.....	13
2.2 Schematic Tab.....	13
2.2.1 Target Processor	13
2.2.1.1 Target Processor Properties.....	14
2.2.1.1.1 System Settings Tab	14
2.2.1.1.2 Direct Programming Tab.....	16
2.2.1.1.3 Monitor	16
2.2.2 Transceiver	17
2.2.2.1 Transceiver Node Properties.....	18
2.2.3 Peripheral device	21
2.2.3.1 Peripheral device types	21
2.2.3.1.1 Audio Source.....	21
2.2.3.1.2 Audio Sink	21
2.2.3.1.3 Audio Source and Sink	22
2.2.3.1.4 Generic.....	22
2.2.3.1.5 Audio Host.....	22
2.2.3.2 Peripheral Programming.....	23
2.2.3.2.1 Generic Peripheral register programming	23
2.2.3.2.2 Block Peripheral register programming.....	24
2.2.3.2.3 Generating Programming file (XML) from SigmaDSP schematic	26
2.2.3.2.4 Linking SigmaDSP Schematic with Peripheral programming file.....	27
3 Drawing A2B schematics	28
3.1 Slot Configuration	31

3.1.1 AD241x	31
3.1.2 AD242x	32
3.1.2.1 Upstream	32
3.1.2.2 Downstream	33
3.2 Stream based Network Design	34
3.2.1 Network wide Stream Configuration	35
3.2.1.1 Stream Definition	35
3.2.1.2 Stream Assignment	35
3.2.2 Stream reordering to optimize bandwidth	39
3.2.3 Node specific Stream Information	39
3.3 Custom Node ID based Configuration	41
3.3.1 I2C Device	41
3.3.2 GPIO Pins	42
3.3.3 Mailbox	43
3.3.4 Operability within the A2B Stack	44
3.4 A2B Bus Analyzer as Sub node	45
3.5 Auto Configuration	47
3.5.1 Auto Configuration of Network	47
3.5.1.1 Storing Network Configuration	47
3.5.1.2 Loading Network Configuration	48
3.5.2 Auto Configuration of Slave Nodes	48
3.5.2.1 Storing Module Configuration	49
3.5.2.2 Loading Module Configuration	49
4 Exporting A2B System Configuration files	50
4.1 General View Configuration files	52
4.1.1 Bus Configuration Files (BCF)	53
4.1.1.1 adi_a2b_busconfig.c	53
4.1.1.2 adi_a2b_busconfig.dat	53
4.1.1.2.1 Stream Information	53
4.1.1.3 adi_a2b_busconfig.xml	56
4.1.2 Super Bus Configuration Files (Super BCF)	57
4.1.2.1 adi_a2b_superbusconfig.c	58
4.1.2.2 adi_a2b_superbusconfig.xml	59
4.1.3 Multi Master Bus Configuration File	59
4.1.3.1 adi_a2b_multibusconfig.c	60
4.1.3.2 adi_a2b_multibusconfig.xml	60
4.2 I2C Command List files	60
4.2.1.1 adi_a2b_commandlist.h	60

4.2.1.2 adi_a2b_commandlist.xml	61
4.3 Node Configuration File (NCF)	61
4.4 Working with Configuration XML files	63
4.4.1 Workflow using NCF and BCF files	63
4.4.2 Workflow using BCF and Super BCF Files	64
5 Network Analysis and Debug	66
5.1 Bandwidth Calculation	66
5.2 Bit Error Rate (BERT) Monitoring	68
5.3 Power Calculation	70
5.4 Debug	73
5.4.1 Trace	73
5.4.2 Sequence Chart	74
5.5 Schematic Validation Report	76
6 Tuning SigmaDSP over A2B	79
6.1 Tuning / Programming a SigmaDSP	79
7 Miscellaneous	82
7.1 Fix for USBi Download Issue	82
Terminology	83
References	83

List of Figures

Figure 1: AD24xx in TreeToolBox of SigmaStudio	12
Figure 2: Target Processor in TreeToolBox	13
Figure 3: System settings tab	14
Figure 4: Direct programming tab	16
Figure 5: Monitor tab	17
Figure 6: A2B Transceiver nodes in TreeToolBox	18
Figure 7: General View Tab	19
Figure 8: Register View Tab	20
Figure 9: Swapping Tx pins	21
Figure 10: Peripheral nodes in TreeToolBox	22
Figure 11: Generic peripheral programming window	24
Figure 12: Block peripheral register programming window	25
Figure 13: Sequencer window for creating block programming file	25
Figure 14: Saving SigmaDSP Schematic as a XML file	26

Figure 15: Linking SigmaDSP Schematic with Peripheral Programming file	27
Figure 16: A2B Schematic tab	28
Figure 17: A2B Stream configuration	29
Figure 18: Device properties window	30
Figure 19: Specifying silicon revision	31
Figure 20: Specifying broadcast slots	32
Figure 21: Slot Config – Slave Node Properties Window	34
Figure 22: Stream Definition	35
Figure 23: Stream Assignment	36
Figure 24: Channel to Skip set to 0	37
Figure 25: Slave Node 1 Stream View without skip	37
Figure 26: Channels to Skip set to 1	38
Figure 27: Slave Node 1 Stream with skip	38
Figure 28: Optimized and Default Buttons	39
Figure 29: Stream View	40
Figure 30: Custom Node ID in Memory	41
Figure 31: Writing Custom Node ID in a Memory device	42
Figure 32: Custom Node ID using GPIO pins	43
Figure 33: Custom Node ID using A2B mailbox	44
Figure 34: A2B Analyzer as sub node	46
Figure 35: Saving Schematic to EEPROM	47
Figure 36: Schematic Dump as XML	48
Figure 37: Saving Auto Configuration to EEPROM	49
Figure 38: Exporting system configuration	51
Figure 39: System Configuration File export window	52
Figure 40: EEPROM and .dat export with Stream Information	54
Figure 41: BCF import	57
Figure 42: Super Bus Configuration File export	58
Figure 43: Multi master Bus Configuration File export	59
Figure 44: NCF Export	61
Figure 45: NCF export window	62
Figure 46: NCF import	63
Figure 47: Work flow using NCFs and BCF	64
Figure 48: Work flow using BCFs and SuperBCF	65
Figure 49: Calculate Option	66
Figure 50: Bandwidth Calculation	67
Figure 51: Bandwidth break-up	68
Figure 52: BERT Calculation	69

Figure 53: Graphical representation of bit errors.....	70
Figure 54: Power Source.....	71
Figure 55: Power Calculation.....	72
Figure 56: Power Budget.....	73
Figure 57: Sample Trace file	74
Figure 58: Setting Path in Environment Variables.....	75
Figure 59: Sequence Chart Sample.....	76
Figure 59: Error Message – Sequence Chart Prerequisites not met	76
Figure 61: View Validation Report	77
Figure 61: Schematic Validation Report	78
Figure 63: Consolidated Stream View in Schematic Validation Report	78
Figure 64: Connecting SigmaDSP to A2B-USBi channel.....	79
Figure 65: A2B Network Interface Settings.....	80
Figure 66: Tuning SigmaDSP over A2B	81
Figure 67: Enabling USBi Fix.....	82
Figure 68: USBi Warning Message.....	82

List of Tables

Table 1: Revision List	3
Table 2: Terminology.....	83
Table 3: References	83

1 Introduction

This document provides an overview on how to draw A2B schematics on SigmaStudio. The document describes about various features available in SigmaStudio for A2B network discovery, configuration, debug, and analysis. Using SigmaStudio one can quickly prototype and evaluate an A2B system on a Host PC. Once the system is verified, the network configuration information can be exported in various formats for easy integration into an embedded platform.

2 A2B schematics components

Components required for drawing A2B schematics in SigmaStudio are described here. Make sure that SigmaStudio setup is completed as explained in Section 4 of [1].

2.1 Hardware configuration Tab

2.1.1 AD24xx

The A2B transceiver chip AD24xx is listed under Processors (ICs / DSP) in the TreeToolBox of hardware configuration tab as shown in Figure 1. By dragging an AD24xx icon on the Hardware configuration tab schematics for an A2B network can be drawn in the schematic tab.

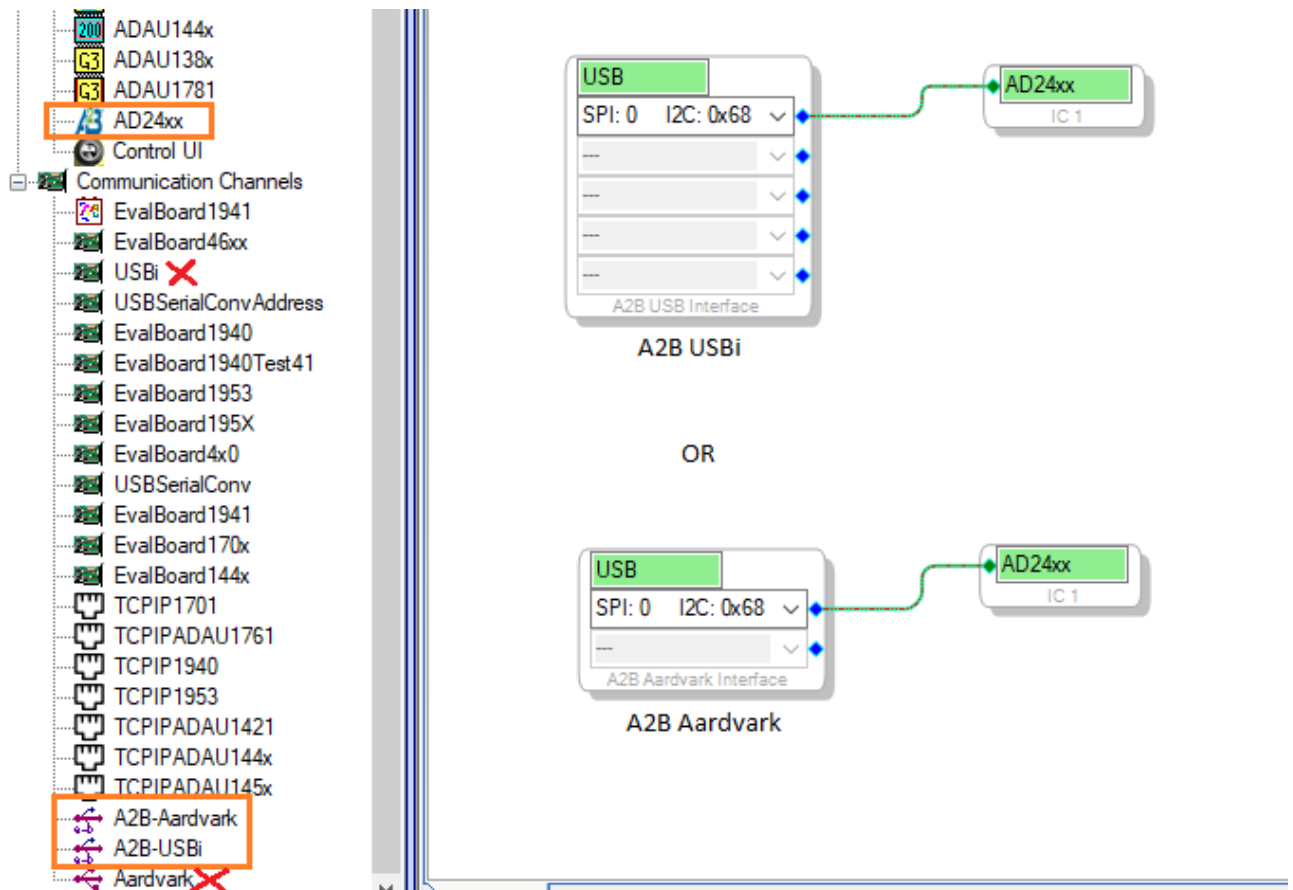


Figure 1: AD24xx in TreeToolBox of SigmaStudio

2.1.2 Communication Channel

Two host adapter options are available for A2B under Communication Channels to interface between SigmaStudio and A2B transceiver chip.

- **A2B-USBi:** Uses Analog Devices USBi I2C/SPI host adapter
- **A2B-Aardvark:** Uses Aardvark (from Total phase) I2C/SPI host adapter

Drag a communication channel block corresponding to your host adapter and connect to AD24xx block by drawing a wire between the two as shown in Figure 1.

Note 1: If no channel is connected, SigmaStudio assumes A2B-USBi by default.

Note 2: Do not connect generic USBi or Aardvark.

2.2 Schematic Tab

2.2.1 Target Processor

The target processor is the controller that connects to the master A2B node in the bus. The target processor can control any A2B/Peripheral node in the bus through the master A2B node. It consists of an I2C port to interface with an A2B transceiver node.

In the schematic, the target processor is represented as shown in the Figure 2. This is the starting node in an A2B schematic and has a single output pin (in brown colour) representing I2C.

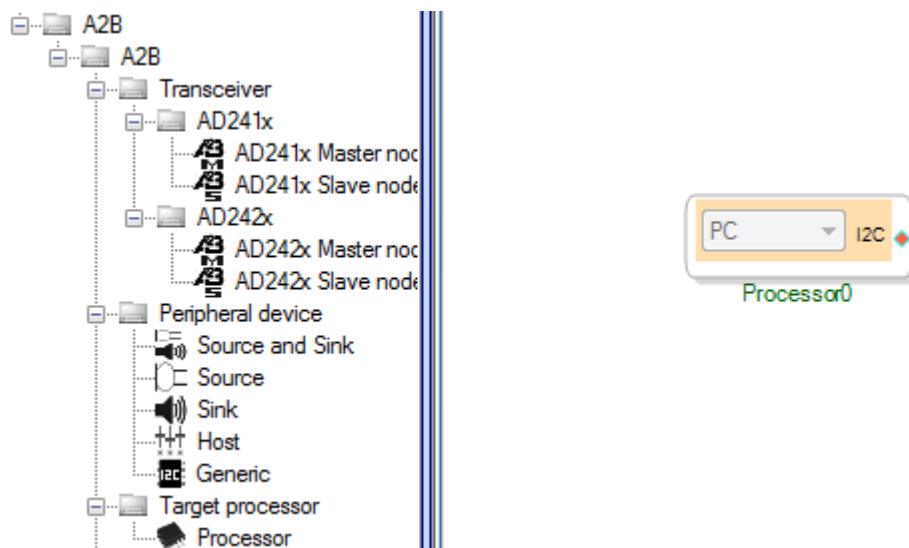


Figure 2: Target Processor in TreeToolBox

2.2.1.1 Target Processor Properties

Users can set properties for the Target processor by accessing the properties window by Right-clicking on the Target processor node and selecting 'Device Properties'. This window offers two tabs. One for setting the system properties and the other for accessing/programming devices connected to the processor.

2.2.1.1.1 System Settings Tab

The system settings tab of target properties window, as shown in Figure 3, offers configurable options for the target processor software. With these settings one can select the node discovery and initialization method to be used and the clock source for the A2B network.

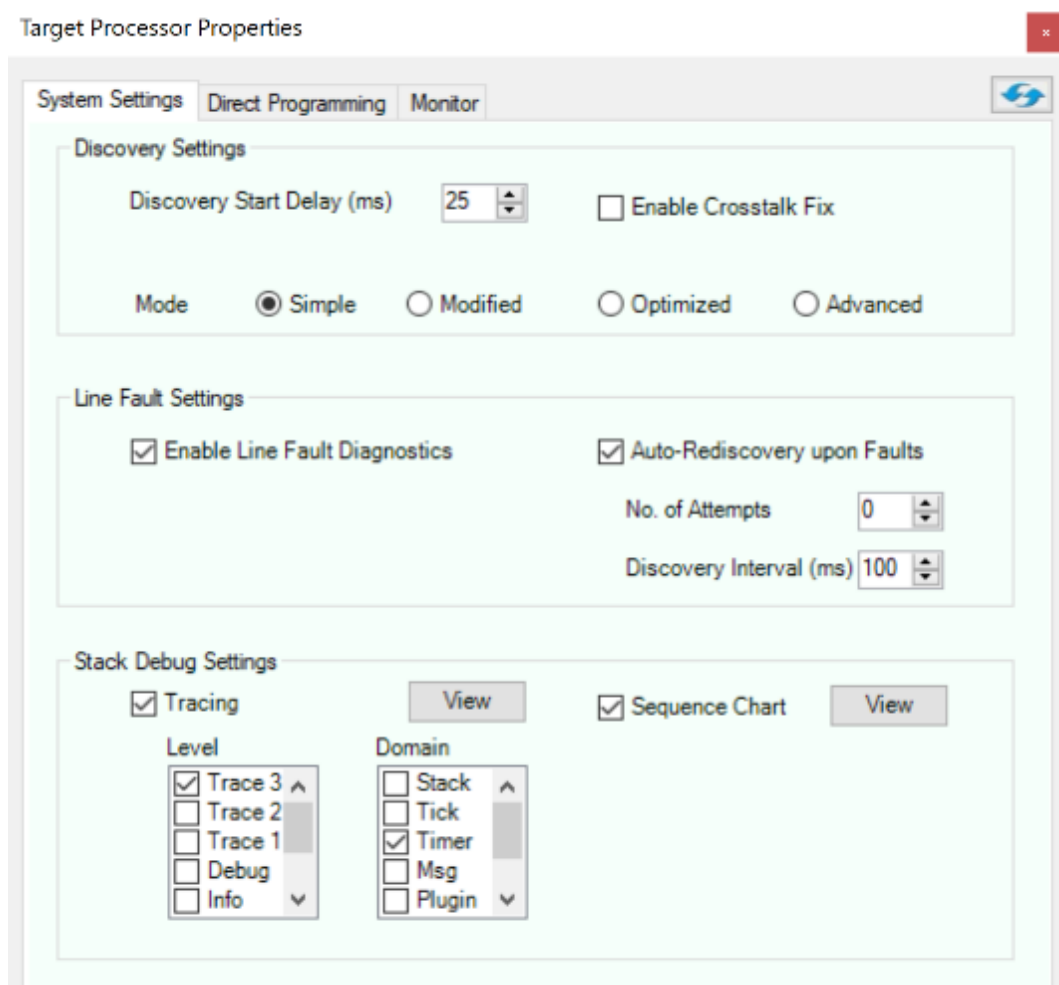


Figure 3: System settings tab

The four possible discovery methods are

Simple: All slave nodes are discovered sequentially from slave 0 to the last available slave in the system. Once all the slaves are discovered, they are initialized for synchronous data exchange.

Modified: All slave nodes are discovered and immediately initialized (for synchronous data exchange) sequentially from slave 0 to the last available slave in the system.

Optimized: Even before a node is initialized the host tries to discover the next node. The time for the next node to be discovered is used to initialize the current node. Synchronous data can start only after all nodes are discovered and initialized.

Advanced: Even before a node is initialized the host tries to discover the next node. The time for the next node to be discovered is used to initialize the current node. Synchronous data exchange can start as soon as a master and slave node 0 is initialized.

Discovery Start Delay: Delay (in milliseconds) to wait after a software reset and before discovery start.

Enable Cross Talk Fix: Enables step wise change of response cycle during discovery and provides more resilience to cross talk from B port to A port. Note that this is applicable only for AD241x and AD2425 variants.

The Line Fault settings allow user to enable fault diagnostic feature of the software. With this any line fault encountered in the system is handled and reported to the user.


Enable Line Fault Diagnostics: Enables line fault diagnostic feature of software. When set, SigmaStudio will continuously monitor the network for faults at 1 second interval. Detected faults will be handled and notified.

Rediscovery upon faults: If checked, automatic network rediscovery will be performed upon detecting post-discovery faults.

- **No. of Attempts:** Specifies the number of rediscoveries attempt to be tried if the fault persists (-1 for infinite retry).
- **Discovery Interval (Ms):** Delay between each rediscovery attempt in milliseconds.

Stack Debug Settings: This feature allows the user to analyse/ understand the system by using the information logged during stack execution. Information can be either through TRACE or sequence chart or both.

- **Tracing:** Logs key information and network events during the stack execution to a text file. User can configure the required 'Level' and 'Domain' for tracing.
- **Sequence Chart:** Sequence Chart captures network transactions in the form of a rich graphical sequence diagram. The sequence chart enables easy understanding of the discovery flow, fault analysis and various other interactions within the system thus enabling quick debugging.

The Refresh  button checks and updates the discovery status of connected A2B Nodes upon a click.

2.2.1.1.2 Direct Programming Tab

Any I2C device connected directly to the target processor or to an A2B node in the network can be accessed using the direct I2C programming feature.

Figure 4 shows the window used for direct I2C programming. Using this feature any I2C device connected in an A2B network can be accessed for read/write operation. This window can be used to program peripherals available on the master board which are directly connected to the target processor. It allows both individual and block register programming (using XML file).

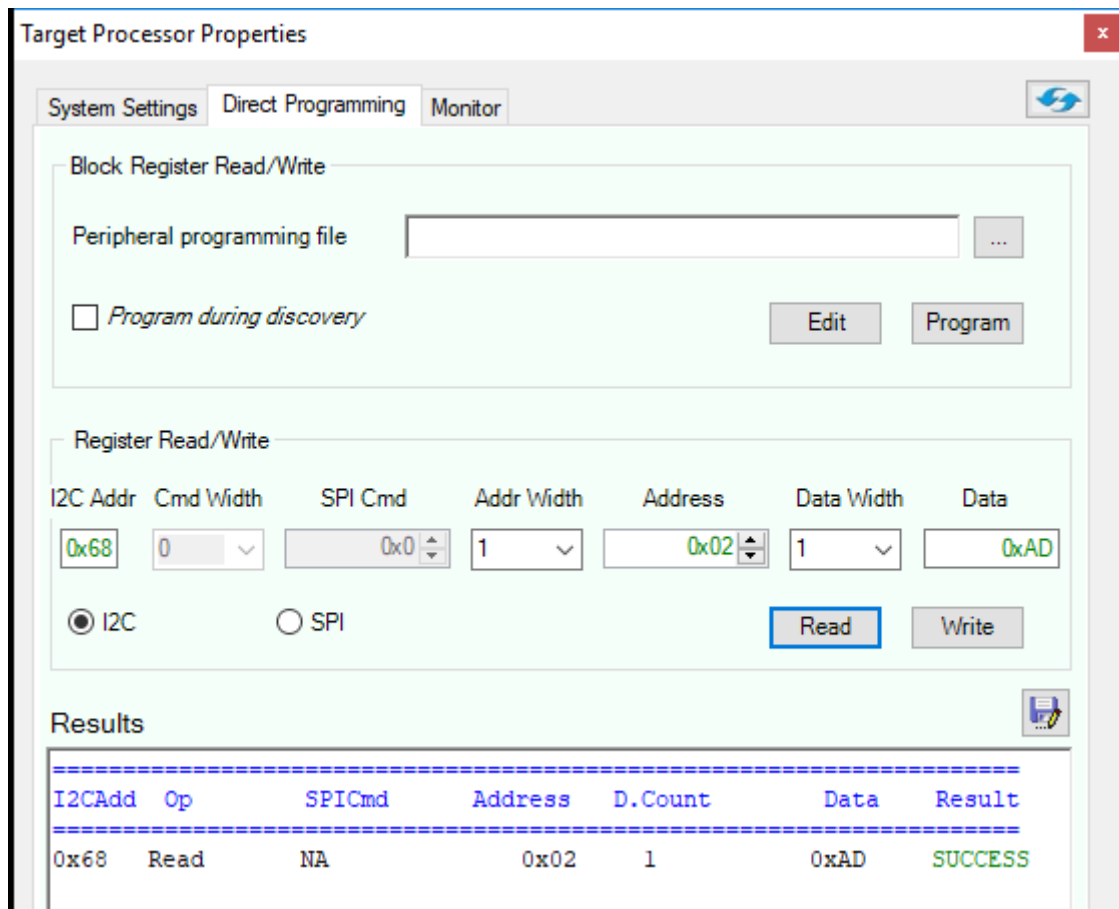


Figure 4: Direct programming tab

2.2.1.1.3 Monitor

Registers of both A2B transceivers and connected peripherals can be periodically polled using the Monitor feature. Option is provided to import & export the fields used for monitoring.

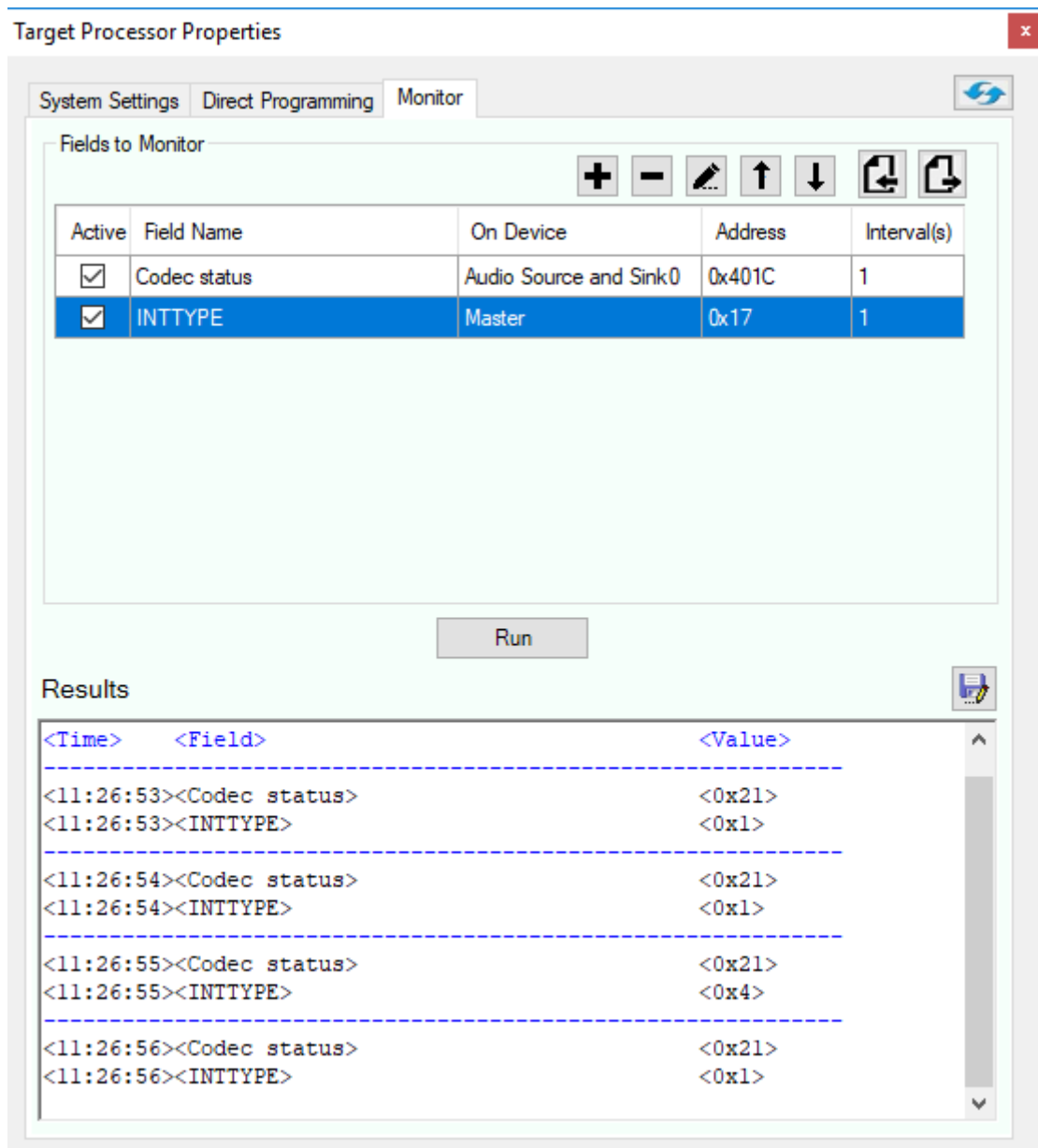


Figure 5: Monitor tab

2.2.2 Transceiver

There are two hierarchies of transceivers- AD241x and AD242x. In each hierarchy, there are A2B transceiver node types namely Master and Slave.

The transceiver nodes are represented as shown in Figure 6. In an A2B schematic, the master transceiver node connects to the target processor over the I2C pin whereas it connects to a slave node over the network output pin "B". The slave transceiver node takes incoming connection on the

network input pin “A” and connects to another slave node on output pin “B”. A2B Transceiver nodes interface with peripherals via I2C/Tx/Rx pins. Rx pin can be used for either PDM or I2S/TDM reception by clicking on the Rx box.

If the output pin B is left open in a slave node, then it is the last device of the A2B bus.

AD241x and AD242x are pin compatible where AD242x has additional features. These two types can be interconnected.

Different variants of transceiver node can be selected using the drop-down box on the node. Depending on the variant selected features/pins become available. Refer A2B Transceiver variant data sheet for more details on the supported features.

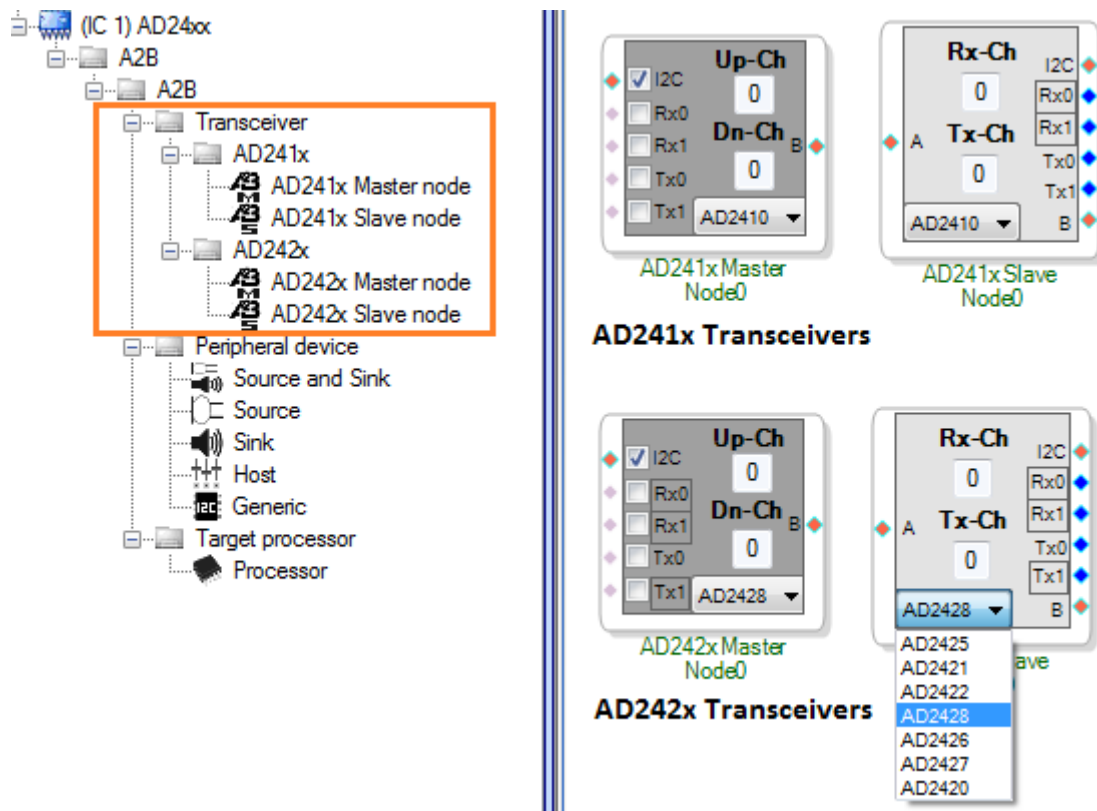


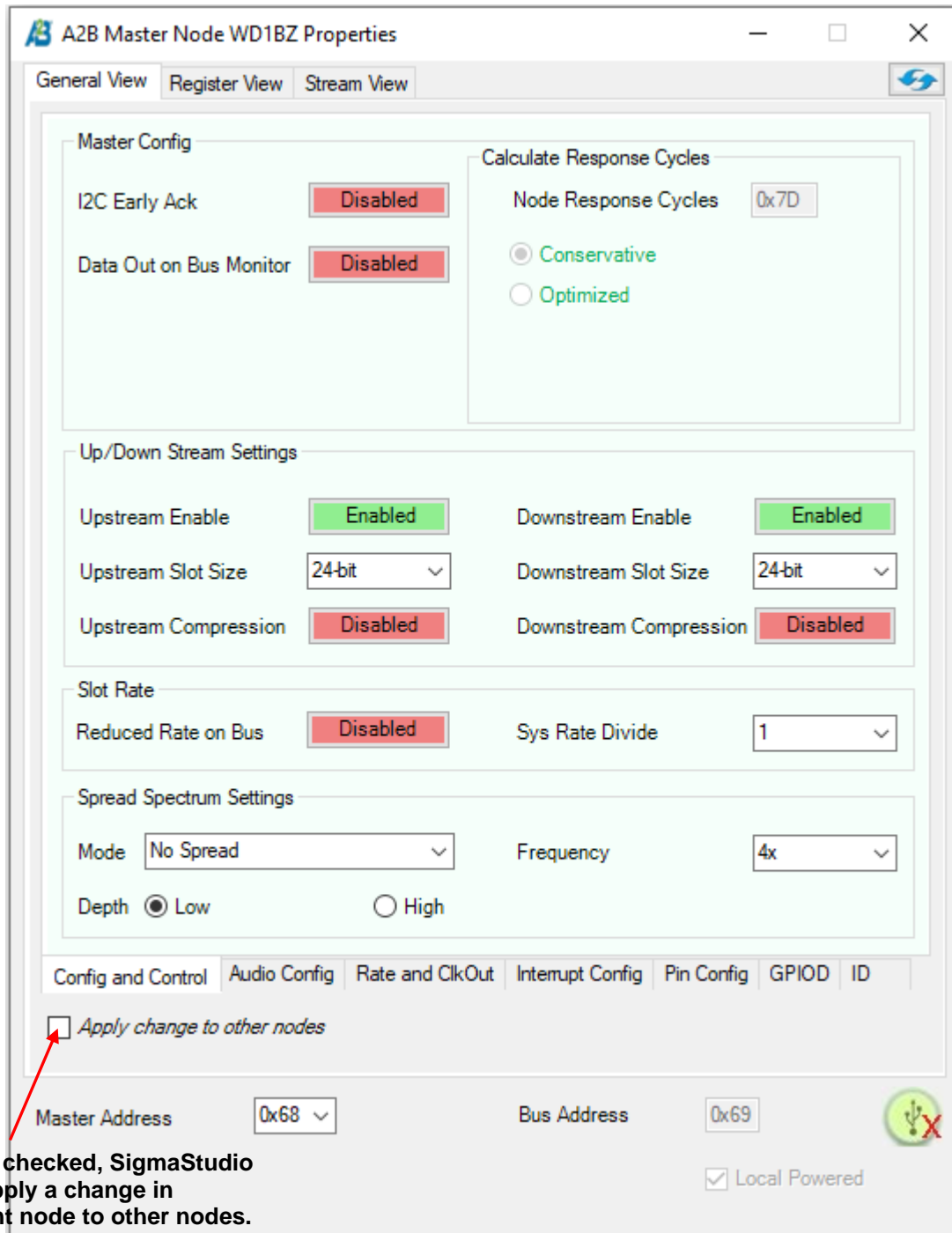
Figure 6: A2B Transceiver nodes in TreeToolBox

2.2.2.1 Transceiver Node Properties

Users can set properties for an A2B transceiver node by accessing the properties window by Right-clicking on the Transceiver node and selecting ‘Device Properties’. This window offers two tabs. The ‘General View’ tab provides functionality-based controls whereas the ‘Register View’ tab provides register fields for configuring the node.

Node configuration can be done from both General and Register Views tabs of Device properties window. Any change in one tab will be reflected in the other.

Figure 7: General View Tab



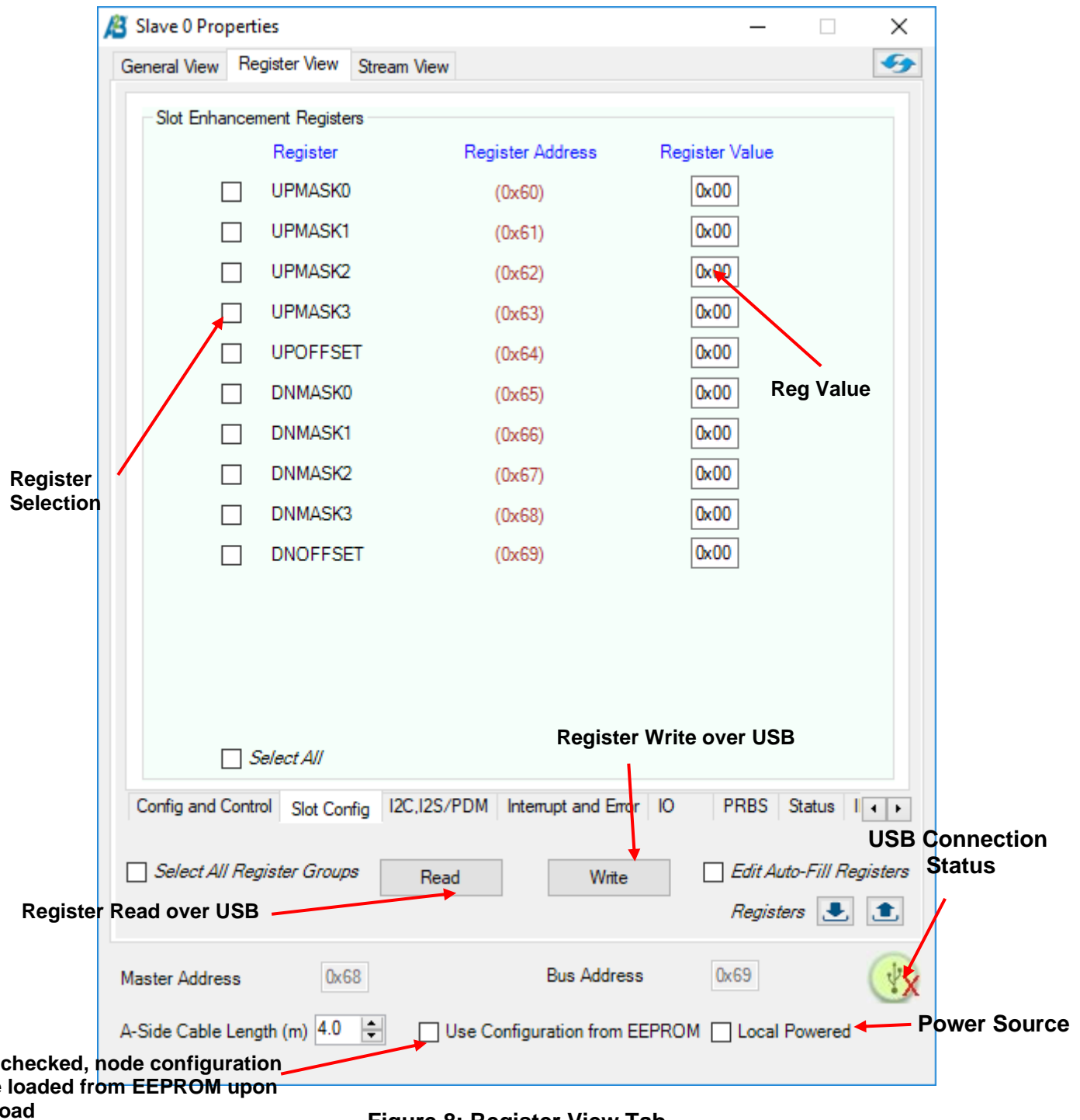


Figure 8: Register View Tab

2.2.3 Peripheral device

Every A2B transceiver node in an A2B network can connect to peripheral devices over the I2C and/or I2S/TDM interface. The peripheral devices can be any device with an I2C and/or I2S interface. Some of the peripherals available on the evaluation boards are SigmaDSP codec, PDM microphones, EEPROM etc.

There are five variants of peripheral devices available as shown in Figure 10. The Tx slots of the peripherals can be connected to the corresponding Rx slots of the A2B nodes and the Rx slots of the peripherals can be connected to the corresponding Tx slots of the A2B nodes. The following section provides a brief description of each peripheral type.

2.2.3.1 Peripheral device types

2.2.3.1.1 Audio Source

Audio source device serves as a source of audio in an A2B network. It connects to A2B slave nodes over I2C and audio receive (Tx) pins. The number of audio channels contributed by the device can be entered separately for each Tx pin. Check boxes can be used to disable or enable pins depending upon the intended use case.

The Tx pins can be swapped in functionality from input to output or vice versa by using the right click option as shown in Figure 9

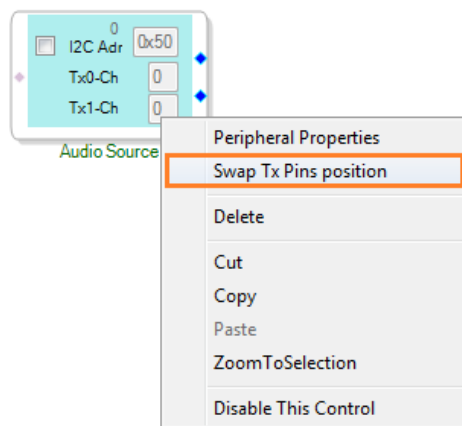


Figure 9: Swapping Tx pins

2.2.3.1.2 Audio Sink

Audio sink device serves as a receiver (sink) of audio in an A2B network. It connects to A2B slave nodes over I2C and audio transmit (Rx) pins. The number of audio channels consumed by the device can be entered separately for each Rx pin. Check boxes can be used to disable or enable pins depending upon the intended use case.

2.2.3.1.3 Audio Source and Sink

Audio Source & Sink device can serve as both source and sink of audio in an A2B network. It connects to A2B slave nodes over I2C, audio transmit (Tx) and receive pins. The number of audio channels contributed and consumed by the device can be entered separately for each Tx and Rx pins. Check boxes can be used to disable or enable pins depending upon the intended use case.

2.2.3.1.4 Generic

This device can be any generic I2C enabled device controlled by the slave transceiver and/or target processor. Control device neither consumes nor contributes audio.

2.2.3.1.5 Audio Host

An Audio Host device interfaces to the A2B master node. Host device routes the audio to and from the A2B network. Two Tx and Rx output pins (Blue coloured pins in Figure 10) connect to the corresponding Rx and Tx pins of the master. Audio Host interfaces to target processor via I2C pin (Brown colour pin in Figure 10).

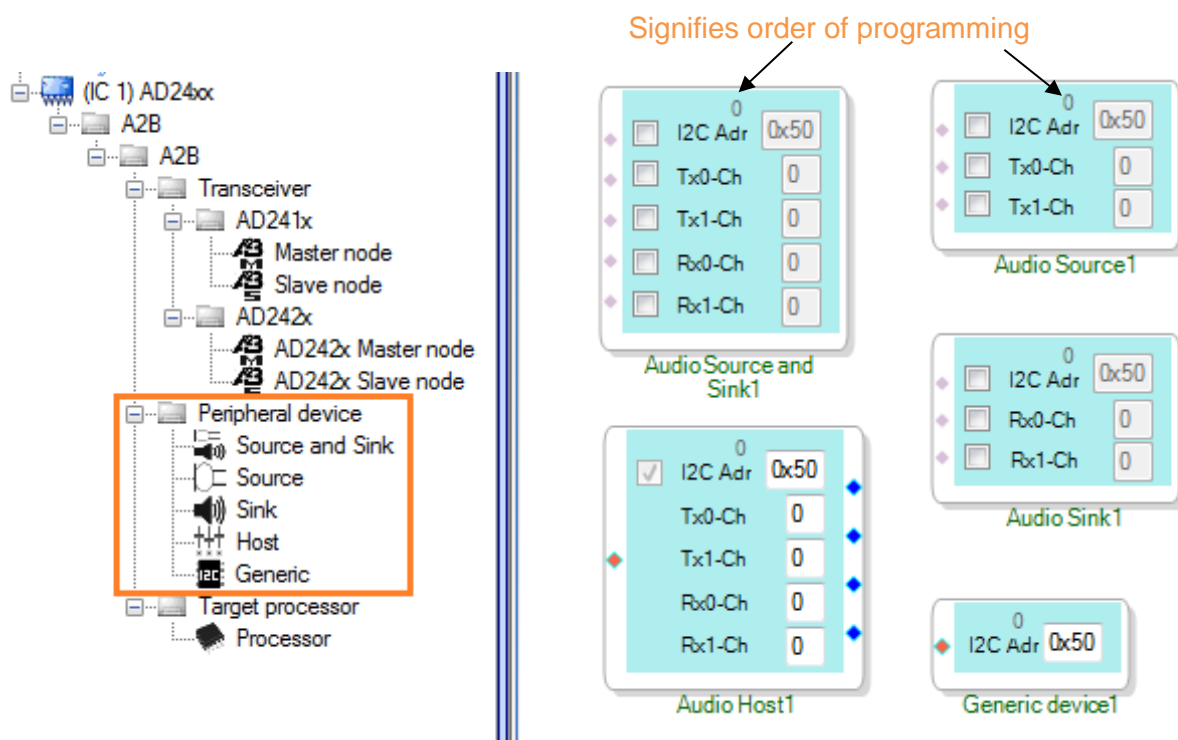


Figure 10: Peripheral nodes in TreeToolBox

2.2.3.2 Peripheral Programming

This section describes the method to be followed for programming peripherals connected to an A2B node. The following general steps are involved in peripheral programming.

1. Create a valid A2B schematic (or open an existing A2B Schematic project) in SigmaStudio. Make sure that the schematic has at least one peripheral connected to a slave node.
2. Enter the relevant peripheral node properties (I2C address, Rx/Tx slots).
3. Download the schematic by clicking '*LinkCompileDownload*' option of SigmaStudio.
4. Open Peripheral device properties window by right clicking on peripheral node

The two options for programming the peripheral device are explained in the following sub sections.

2.2.3.2.1 Generic Peripheral register programming

With the Generic option, user can read or write any individual register (one at a time) by setting required fields as shown in Figure 11.

Note: The Address and Data Widths are in Bytes.

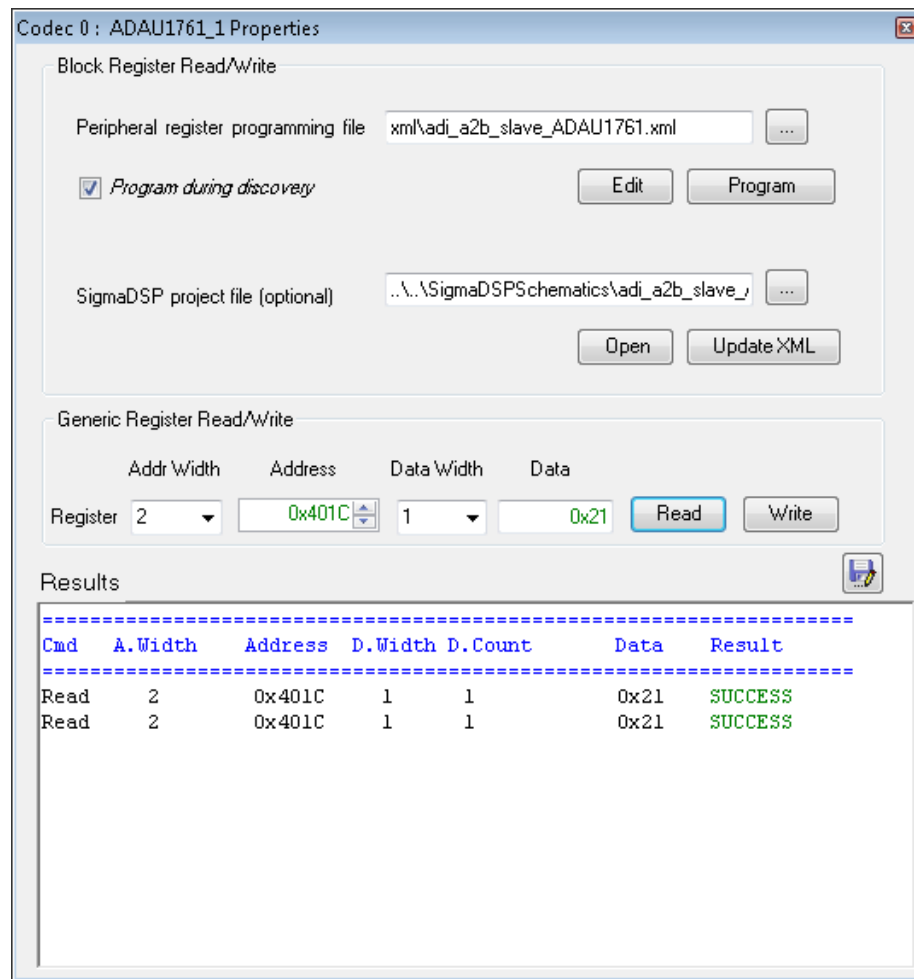


Figure 11: Generic peripheral programming window

2.2.3.2.2 Block Peripheral register programming

With the Block programming option, user can read or write a set of registers as shown in Figure 12. This can be accomplished by providing an xml file containing the instructions to be executed.

By checking 'Program during discovery' option the peripheral will be programmed during the discovery process. Note that the order of peripheral programming depends on the order in which the peripherals were connected to the A2B node. This is indicated by a numeric value at the top of the peripheral device icon as shown in Figure 10.

Note: The Address and Data Widths are in Bytes.

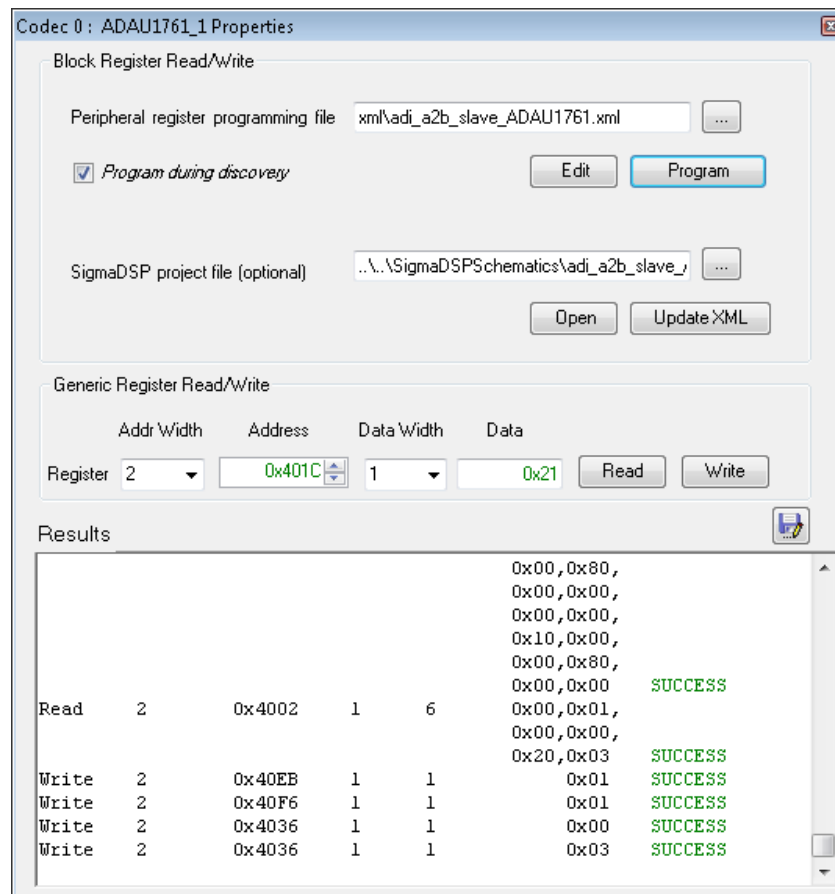


Figure 12: Block peripheral register programming window

An example xml file contents are shown in Figure 13. A new peripheral programming file can be created by clicking on the Edit button. This will open an editor in which the instructions can be added /edited.

Optionally, user can save the SigmaStudio project file for ADI SigmaDSP peripheral corresponding to the XML file provided in the Block Register Read/Write section.

Mode	Ad...	Address	Dat...	Dat...	Data	Parameter Name	Address Incr(in Bytes)	Pro...
Write	2	0x0000	1	4	0x00, 0x00, 0x10, 0x00	NonModRamAlloc	0	I2C
Write	2	0x40EB	1	1	0x7F	IC 1.Sample Rate Setting	0	I2C
Write	2	0x40F6	1	1	0x00	IC 1.DSP Run Register	0	I2C
Write	2	0x4000	1	1	0x0F	IC 1.Clock Control Regis...	0	I2C
Write	2	0x4002	1	6	0x00, 0x01, 0x00, 0x00, 0x20 ...	IC 1.PLL Control Register	0	I2C
DELAY			1		0x00, 0x64	IC 1.Delay		

Figure 13: Sequencer window for creating block programming file

2.2.3.2.3 Generating Programming file (XML) from SigmaDSP schematic

SigmaDSP connected to an A2B node can be programmed by providing an XML file generated from the SigmaDSP schematic. The procedure to generate an XML file is explained here.

1. Open the required SigmDSP project in SigmaStudio. Make sure that 'Capture window' is visible as shown in Figure 14. If not select from SigmaStudio menu View->Capture window.
2. Click on the LinkCompileDownload command.

Note: This step is required only to capture the exact data sequence that SigmaStudio sends to the hardware and hence no real download to hardware is required.

3. Drag and drop Capture window contents to Sequence window (Left to Right).
4. Click on the 'Save' icon in the Sequence window to save as a Sequence XML file.
5. This file can be used for programming SigmaDSP as explain in Section 2.2.3.2.2 .

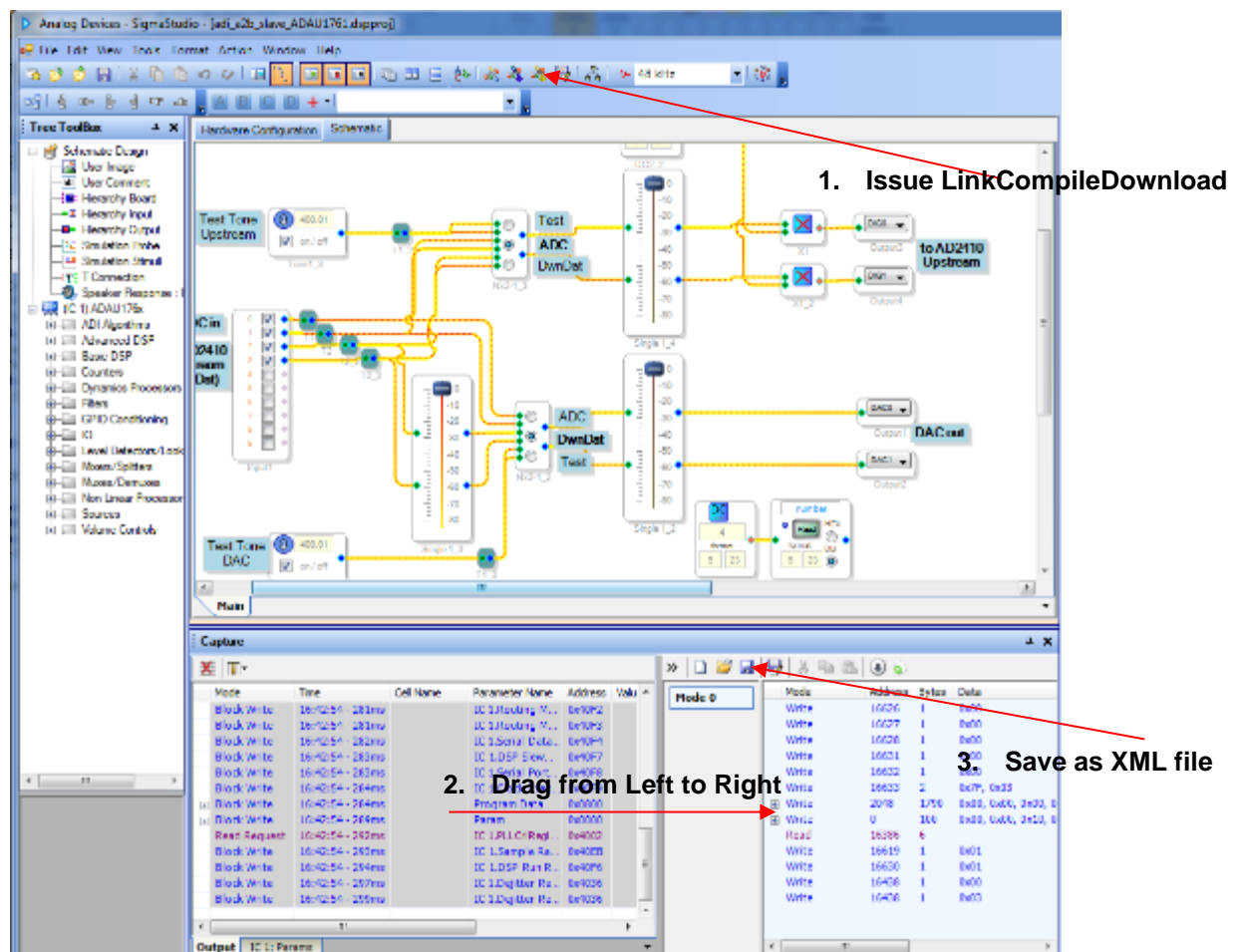


Figure 14: Saving SigmaDSP Schematic as a XML file

2.2.3.2.4 Linking SigmaDSP Schematic with Peripheral programming file

User can link the SigmaDSP schematic and the Peripheral programming file (XML) together by providing the schematic project file as shown in Figure 15. With this, user can directly open the SigmaDSP schematic to make modifications and can update the XML using the 'Update XML' button. Upon successful update, the XML file name turns in to green colour indicating the completion. Note that, this linking is optional and applicable only for SigmaDSP processors.

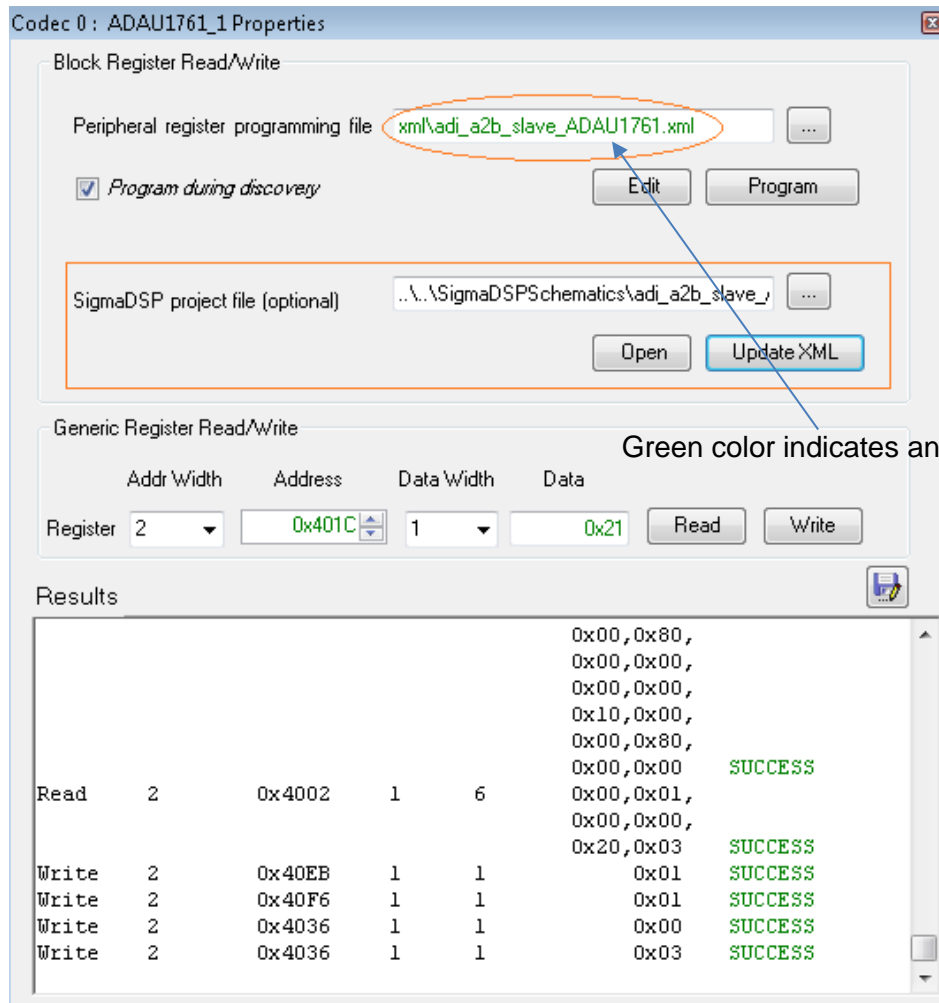


Figure 15: Linking SigmaDSP Schematic with Peripheral Programming file

3 Drawing A2B schematics

The following steps describe the procedure to draw an A2B schematic in SigmaStudio.

1. Open SigmaStudio.
2. Create a new project from File menu.
3. Drag and drop “AD24xx” icon from Tree Toolbox to “Hardware configuration” tab
4. Drag an A2B-USBi or A2B-Aardvark Communication Channel block, depending on the Host I2C adapter used to connect to the transceiver, and wire it to AD24xx block. If no channel is connected, then A2B-USBi is assumed by default.
5. Switch to Schematic tab, click on the A2B icon to list the hierarchies available under A2B. This will list “Peripheral device”, “Transceiver” and “Target processor” icons.
6. Drag icons from Tree Toolbox and connect the blocks to make an A2B schematic as shown in the Figure 16. Select the variant by clicking on the drop-down.

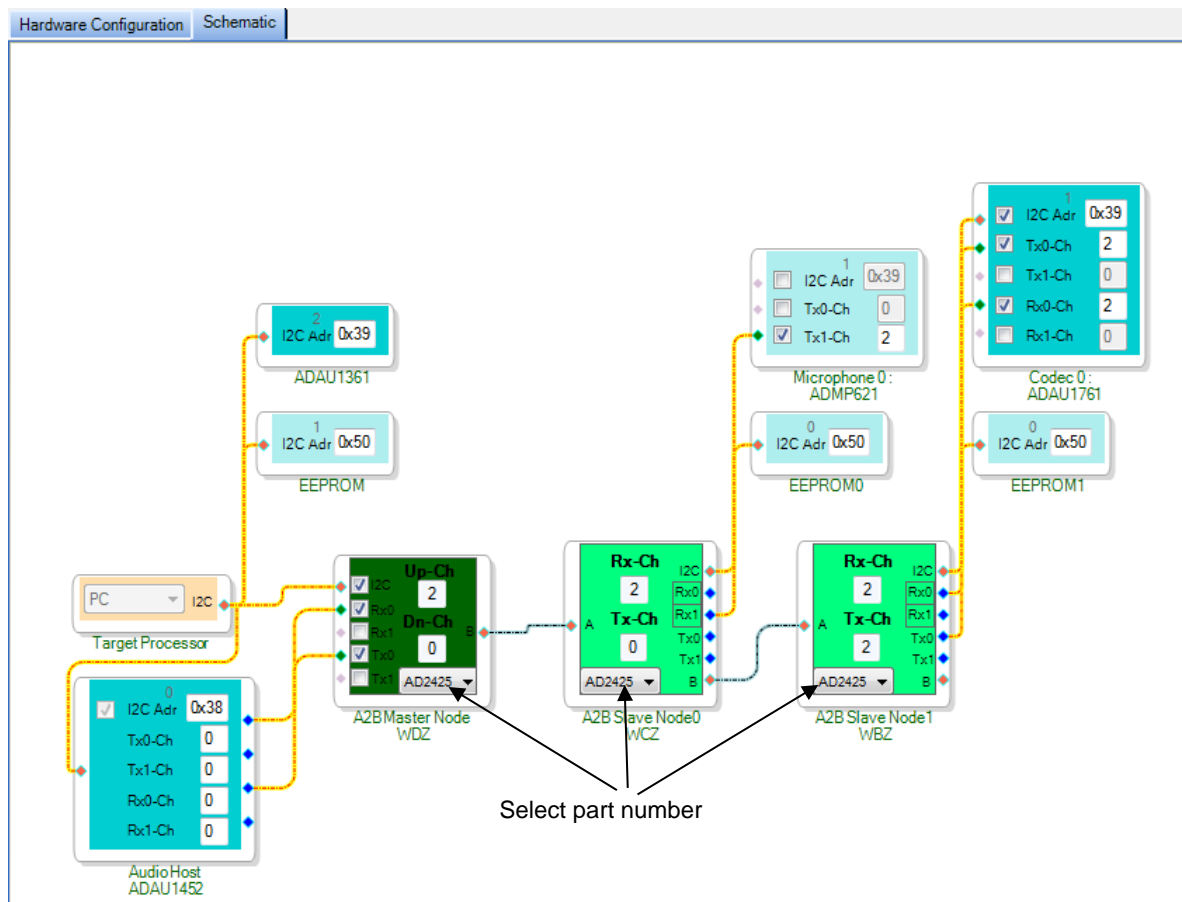


Figure 16: A2B Schematic tab

7. Define and configure streams by right clicking Target Processor. Enable auto-slot configuration (optional) for AD242x network. For details, refer section 3.2

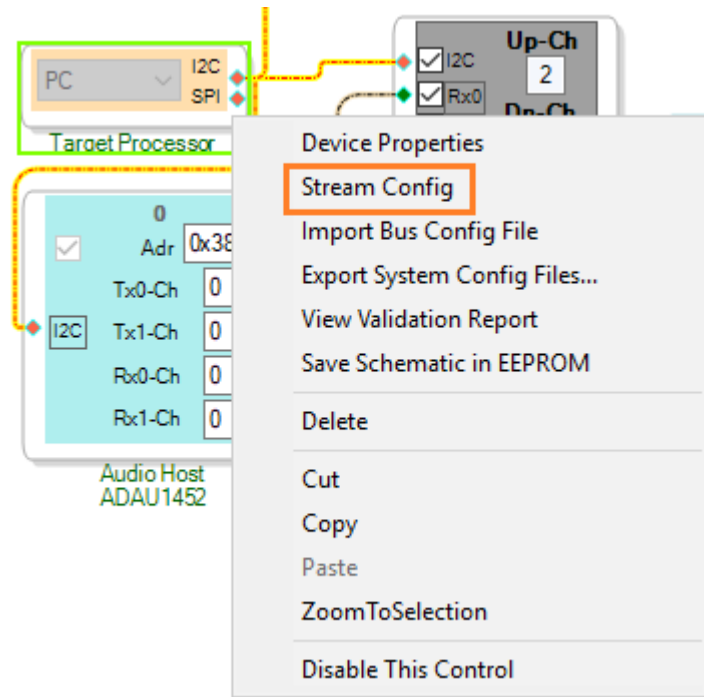


Figure 17: A2B Stream configuration

8. Enter properties for each A2B node by right clicking and selecting the Device properties option. This will open a window as shown in Figure 18.

Note 1: Device properties window of AD241x has two main tabs namely 'General View' and 'Register View'. General View enables intuitive configuration. Any configuration made in General View is reflected in the Register View and vice versa.

Note 2: Device properties window of AD242x has an additional tab - Stream View which provides node level stream usage and bandwidth information.

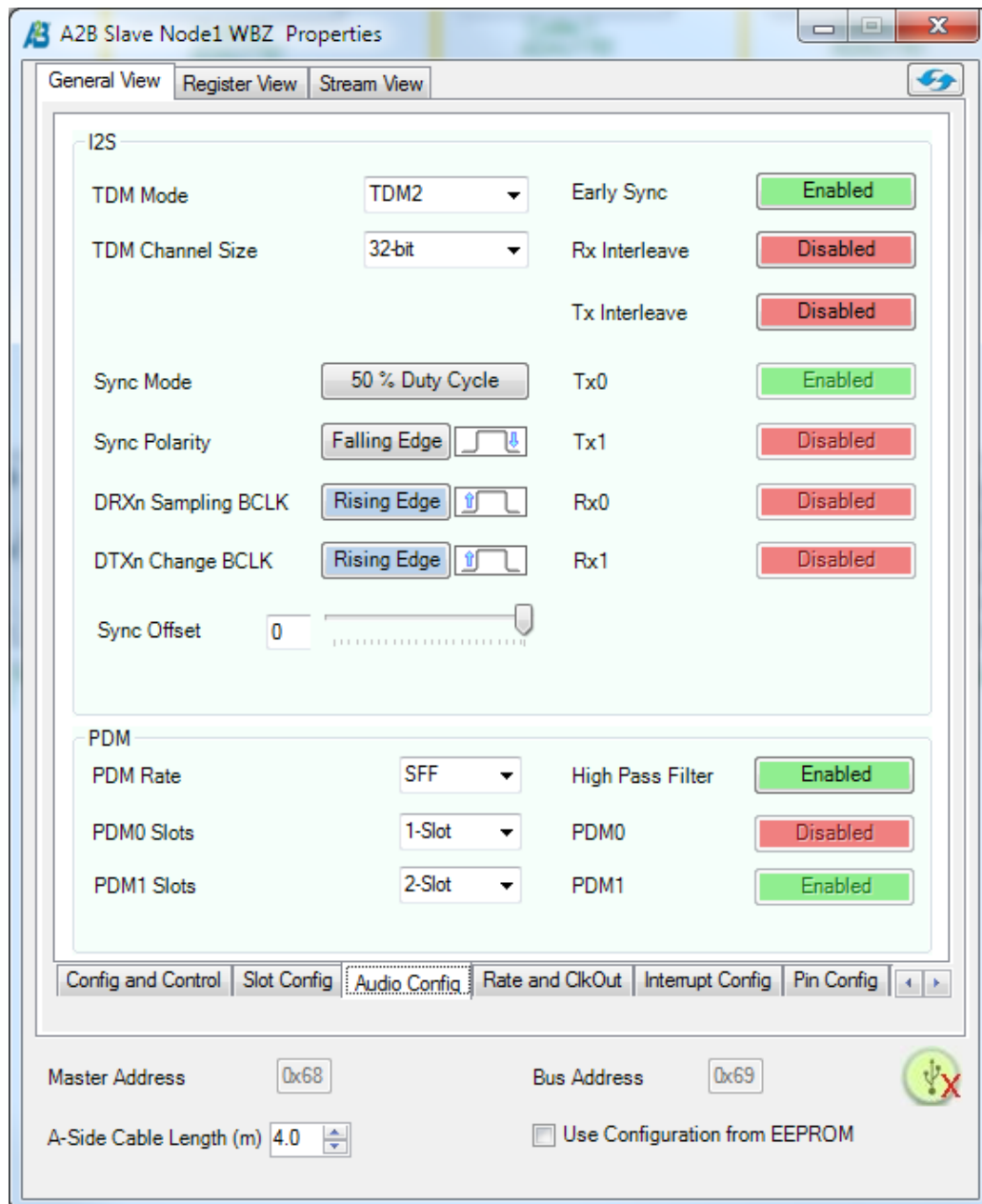


Figure 18: Device properties window

9. Transceiver properties are grouped under different Tabs. Switch to the required Tab and configure as necessary. Configuration of Upstream and Downstream slots may vary depending on the Transceiver type selected. Refer to Section 3.1 for more details.
10. Provide the silicon revision of the A2B Transceiver in the 'ID' tab of General View as shown in Figure 19. Optionally, a custom node identifier can be assigned to a Node by selecting the

“Custom Node Identifier” checkbox. Refer Section 3.3 for more details on Custom Node ID based configuration.

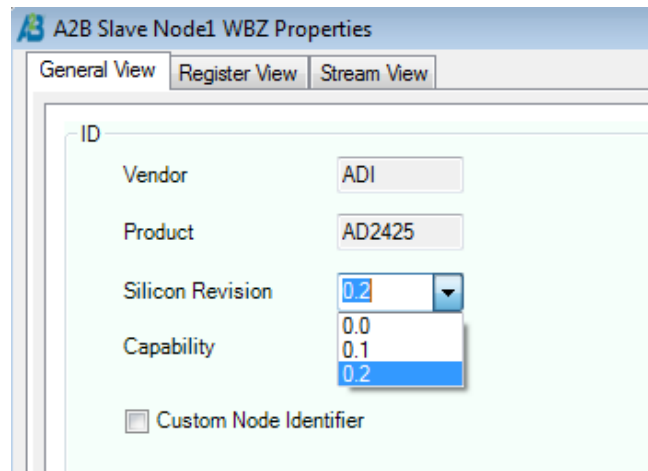


Figure 19: Specifying silicon revision

11. Enter properties for each peripheral node used in the schematic. Refer Section 2.2.3 for details. A peripheral device can be programmed during discovery by providing an XML file having the required configuration information as explained in 2.2.3.2.3
12. Optionally, if the configuration properties for the node and its attached peripherals is to be set by reading from a storage device (EEPROM), user shall set 'Use Configuration from EEPROM' option shown in Figure 18.
13. Once properties for all nodes are entered, check the correctness of the schematic by clicking on the '*Link*' icon in SigmaStudio. Make sure that the drawn schematic has no errors.
14. The schematic is now ready for download.

3.1 Slot Configuration

With slave-to-slave communication possible with AD242x parts, the way upstream and downstream slots are computed in the A2B schematic varies depending on the A2B transceiver type used. This section describes the slot configuration for AD241x and AD242x types.

3.1.1 AD241x

The Upstream and Downstream slots for each A2B slave node are computed using the Rx and Tx channels configured for the peripherals that they are connected to.

If broadcast audio downstream channels are required to be used, then enter Local Down slots + broadcast down slots in '*Rx-Ch*' field of the peripheral and specify broadcast downstream slots BCDNSLOTS in the device properties window as shown in Figure 20.

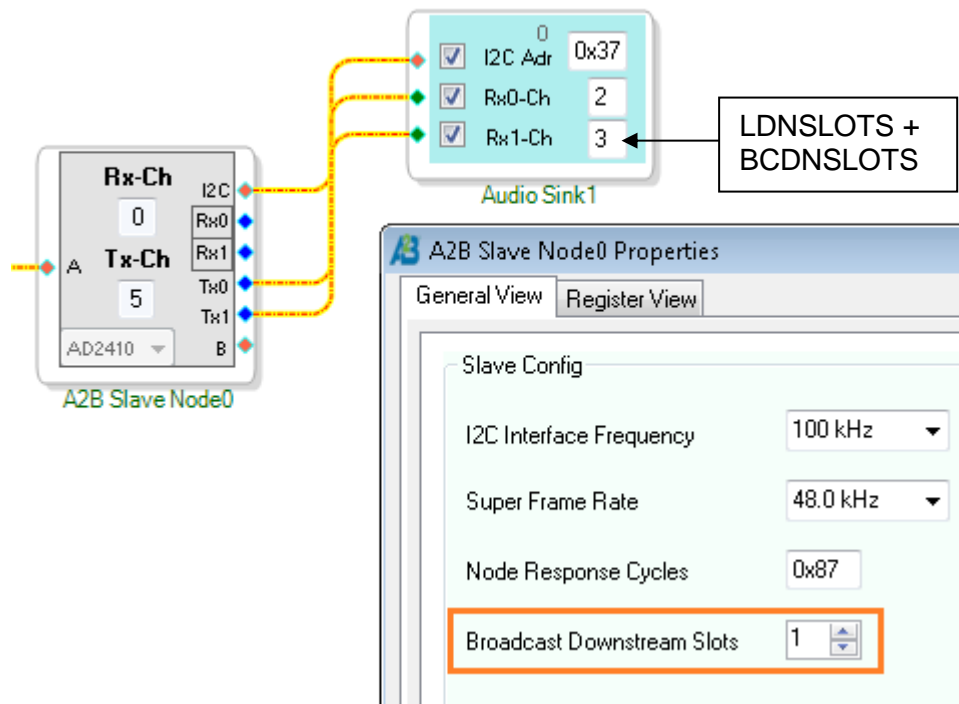


Figure 20: Specifying broadcast slots

3.1.2 AD242x

In A242x, with the ability for a slave node to directly receive/transmit data from/to other slaves without having to move the data all the way up to the master, the Rx and Tx channels configured for the peripherals does not directly translate to the Upstream and Downstream slots.

One can configure slots for AD242x slave nodes using

- Stream based network design scheme where the slots are calculated **automatically** based on network wide stream configuration. Refer Section 3.2 for details.
- 'Slot Config' tab of Device properties window one can **manually** configure slots for AD242x slave nodes as shown in **Error! Reference source not found..** The fields are elaborated in the following section.

3.1.2.1 Upstream

Slots Received at Port B: Number of slots received at Port B. Calculated as Maximum of 'UPMask_{max}' and 'Passed up slots from Port B'.

Slots Passed Up from Port B: Number of received slots to be passed up from Port B.

Slots Contributed: Number of slots being contributed to upstream.

Receive Offset: Number of slots which are skipped before transmission

Slots Transmitted at Port A: Number of slots transmitted at Port A. This is the sum of 'Slots Passed Up from Port B' + 'Slots Contributed'.

+Rx Up Offset: Additional offset that needs to be added to Receive offset after auto slot calculation.

3.1.2.2 Downstream

Slots received at Port A: Number of slots received at Port A. Calculated as Maximum of 'DNMask_{max}' and 'Passed down slots from Port A'.

Slots Passed Down from Port A: Number of received slots to be passed down from Port A

Slots Contributed: Number of slots being contributed to downstream.

Slots Consumed: Number of slots being consumed from downstream when the masks are not used.

Receive Offset: Number of slots which are skipped before transmission.

Slots Transmitted at Port B: This is the sum of 'Passed down slots from Port A' + 'Slots Contributed'.

Broadcast Downstream Slots: This is active only when the Select slots to consume is disabled (AD241x style)

+Rx Down Offset: Additional offset that needs to be added to Receive offset after auto slot calculation.

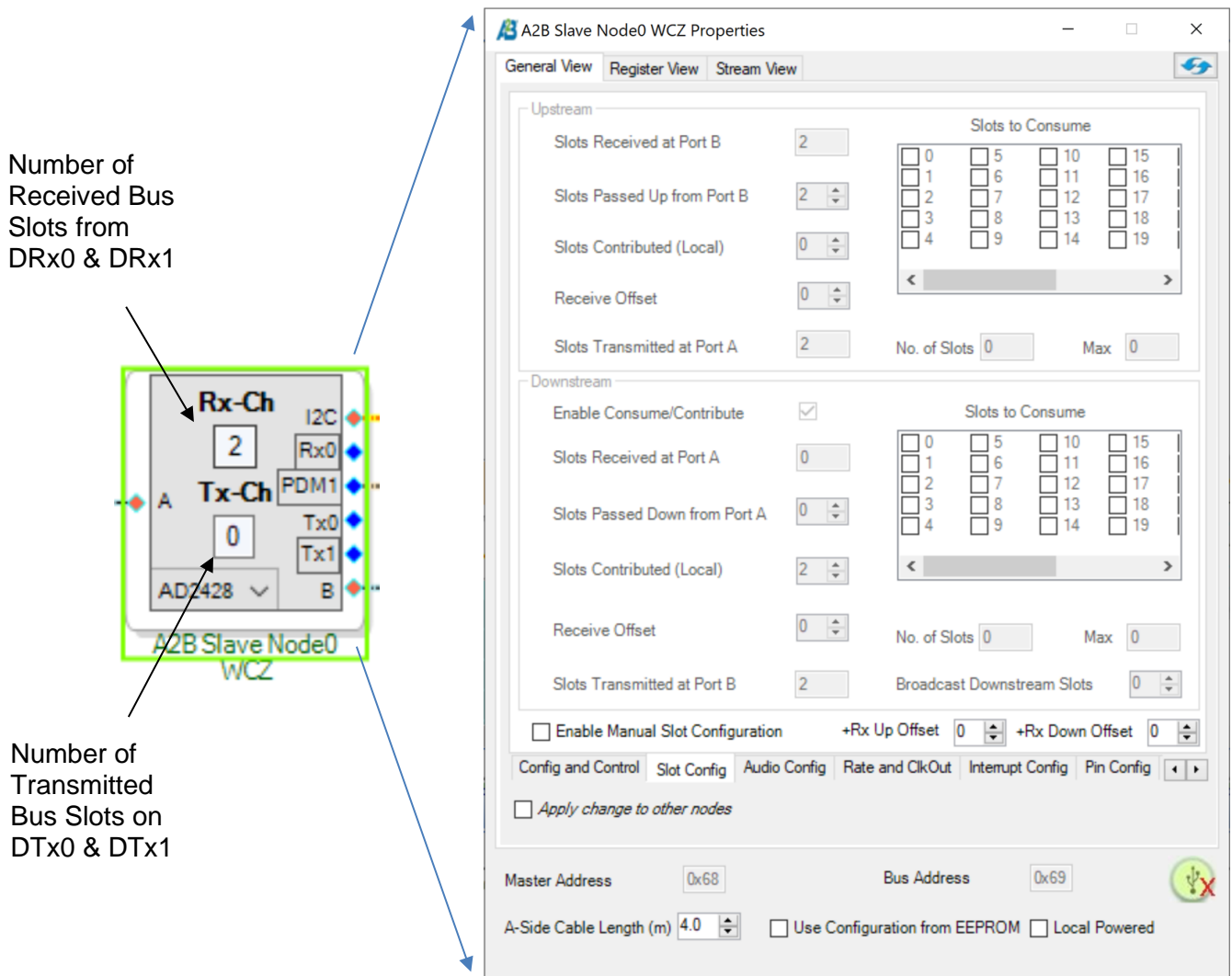


Figure 21: Slot Config – Slave Node Properties Window

Note: Unlike AD241x, the peripheral channels are not automatically used for slot configuration.

3.2 Stream based Network Design

Alternative to the approach of configuring slots in an A2B network using General View/Register view tabs, one could use the Stream based network design approach wherein SigmaStudio does all required A2B slots register settings.

This feature enables auto configuration of A2B slots based on the network wide stream specification. It also facilitates mechanism to store, communicate and import of network wide stream information.

3.2.1 Network wide Stream Configuration

Open Stream configuration window by right clicking Target processor as shown in Figure 17

3.2.1.1 Stream Definition

This window allows the user to define streams along with properties like sampling rate, number of channels per stream etc. Separate option is provided to edit a selected stream properties. User may remove as well as re-order a selected stream using the respective options.

Note: User shall provide a unique name and ID for each stream.

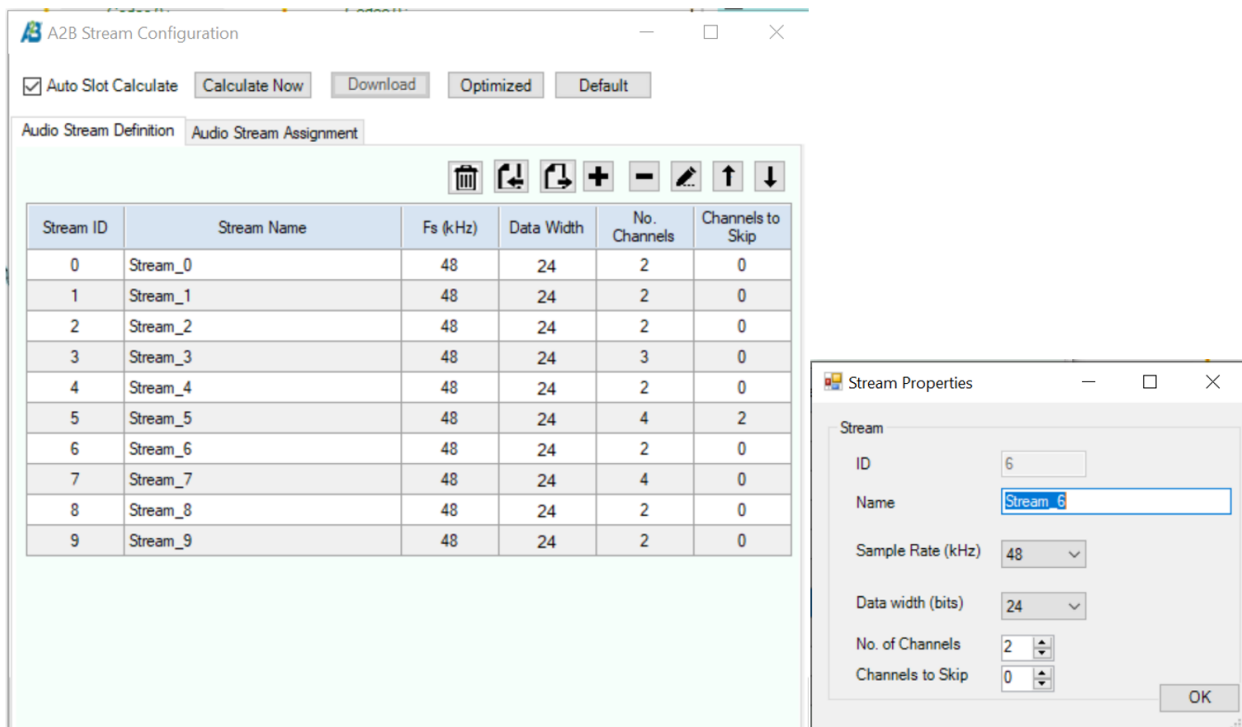


Figure 22: Stream Definition

3.2.1.2 Stream Assignment

All the defined streams are available for assignment. For each stream, user needs to select the source and destination. Each stream can have a single source and one or more destination. The

option “*Auto Slot Calculate*” ensures that slot configuration registers are programmed according to the assignment during schematic Link/download. Optionally, user may trigger slot calculation based on the current assignment by using the button “Calculate Now” and new slots can be dynamically written to a functional network using “Download” button (without rediscovering the network).

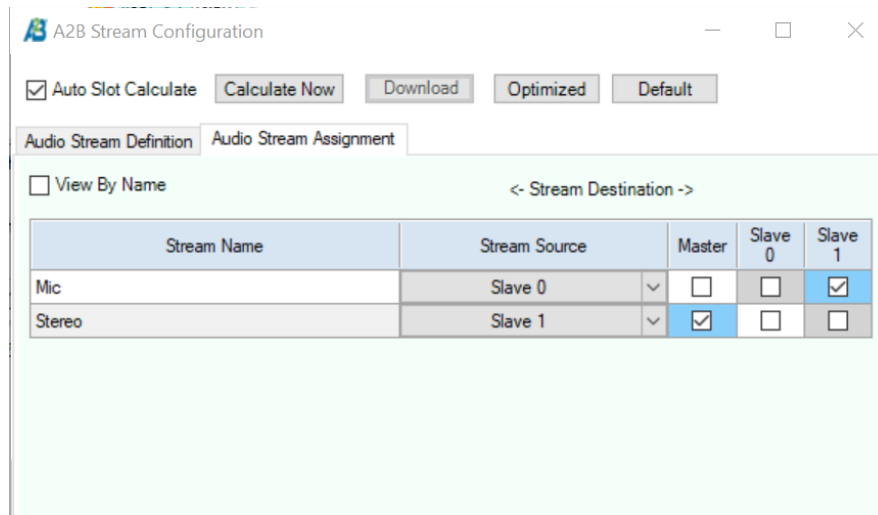


Figure 23: Stream Assignment

Note: The ‘*Auto Slot Calculation*’ option uses slave to slave communication. Hence, it is enabled only when all the nodes support slave to slave communication (Not supported for AD241x).

Note: Channels to Skip field in stream definition specifies the number of slots from the end, that will not be consumed by destination nodes of the stream.

Example: -

With “Channels to Skip” equal to 0 for stream assignment defined in Figure 23.

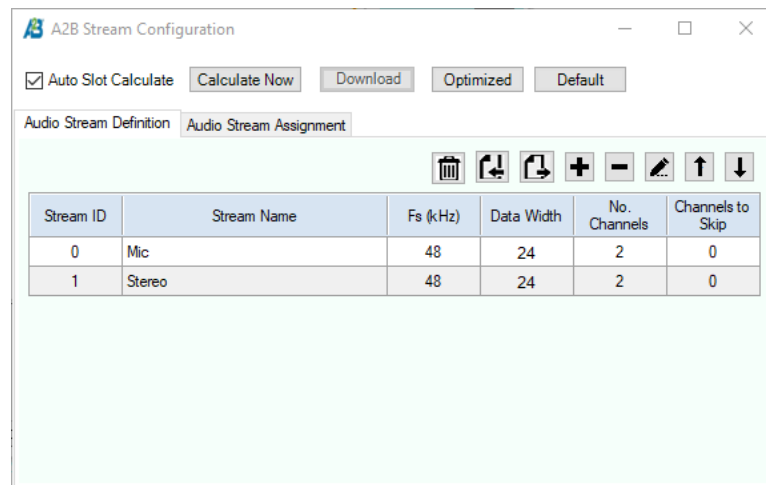


Figure 24: Channel to Skip set to 0

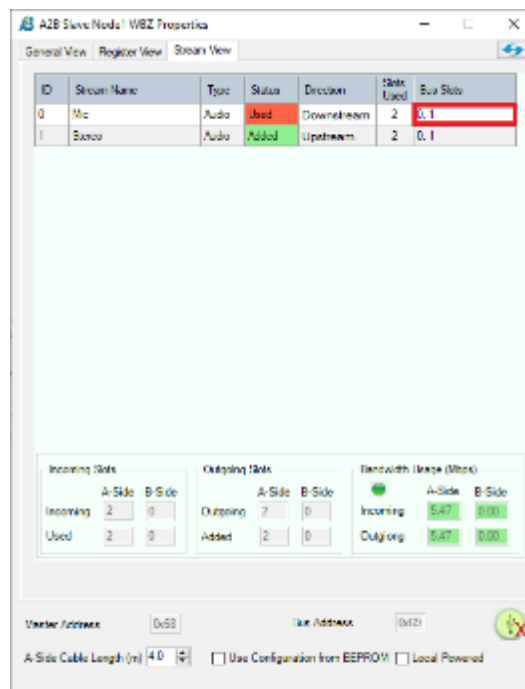


Figure 25: Slave Node 1 Stream View without skip

Both the slots are consumed at the destination (Slave Node 1).

With “Channels to Skip” equal to 1 for stream assignment defined in Figure 23.

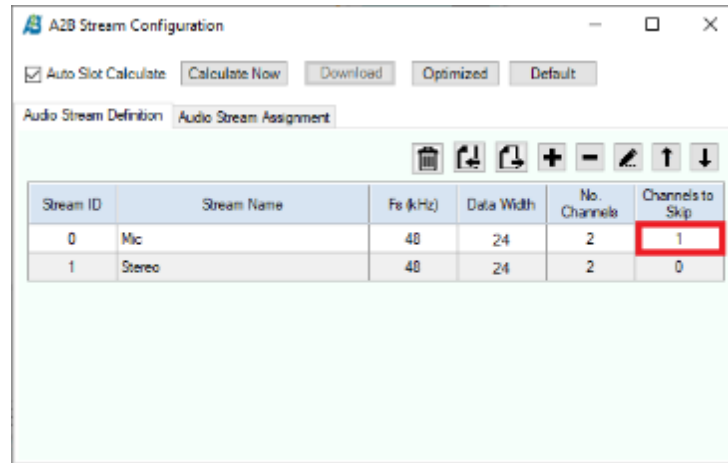


Figure 26: Channels to Skip set to 1

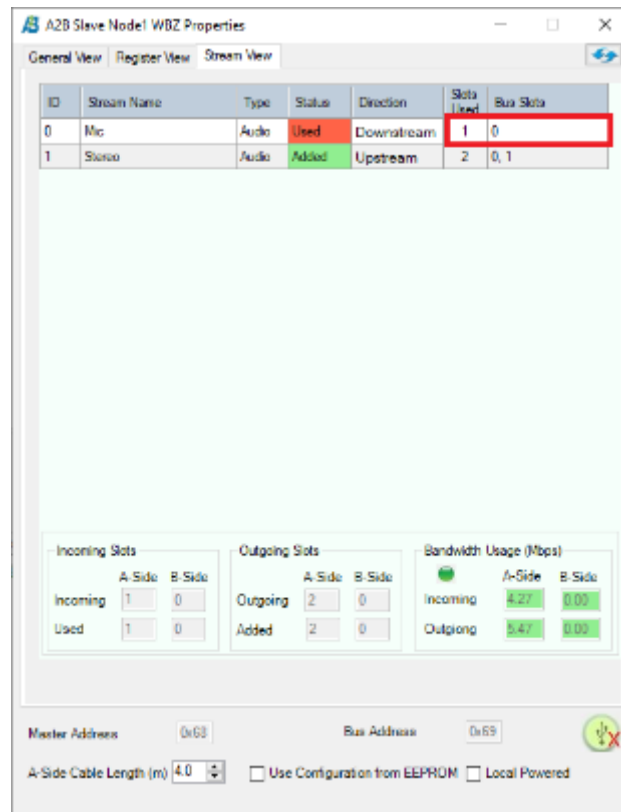


Figure 27: Slave Node 1 Stream with skip.

One slot from end is skipped at destination even though both slots are available at Slave node 1. So only 1 slot is consumed at Destination with both slots available for consumption.

3.2.2 Stream reordering to optimize bandwidth.

The order of streams is changed to save bandwidth on bus upon clicking “Optimized” button.

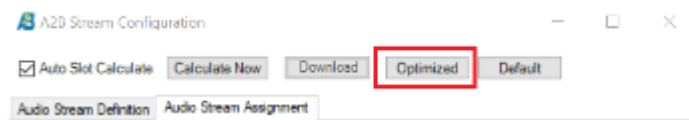


Figure 28: Optimized and Default Buttons

The stream order that is result of optimization would consume least Bandwidth. There can be other stream orders which may have same bandwidth usage. But it will not be less than Optimized stream order.

Note: - Streams having Upstream destination only are placed after streams having Downstream destination (and Upstream). If we have enabled 2 RX/PDM pins user has to make sure that streams are defined in the order RX buffer of A2B is filled (By moving streams up and down).

3.2.3 Node specific Stream Information

The ‘Stream View’ tab of ‘Device Properties’ window as shown in Figure 29, captures the node level information based on the stream assignment. User can view all the streams associated with the current node. Usage of the streams, direction and corresponding bus slots are captured in this view. Note that ‘Stream View’ will be populated only when “Auto Slot Calculate” option is enabled during Stream Assignment. ‘Stream View’ also captures the bandwidth used by the current node.

Note: Stream configuration window and ‘Stream View’ can be viewed concurrently. One can use the “Apply” button (as shown in Figure 23) of stream configuration window to trigger slot calculation. The latest changes with respect to the current node shall be viewed by clicking the “Refresh” button of device properties (as shown in Figure 29).

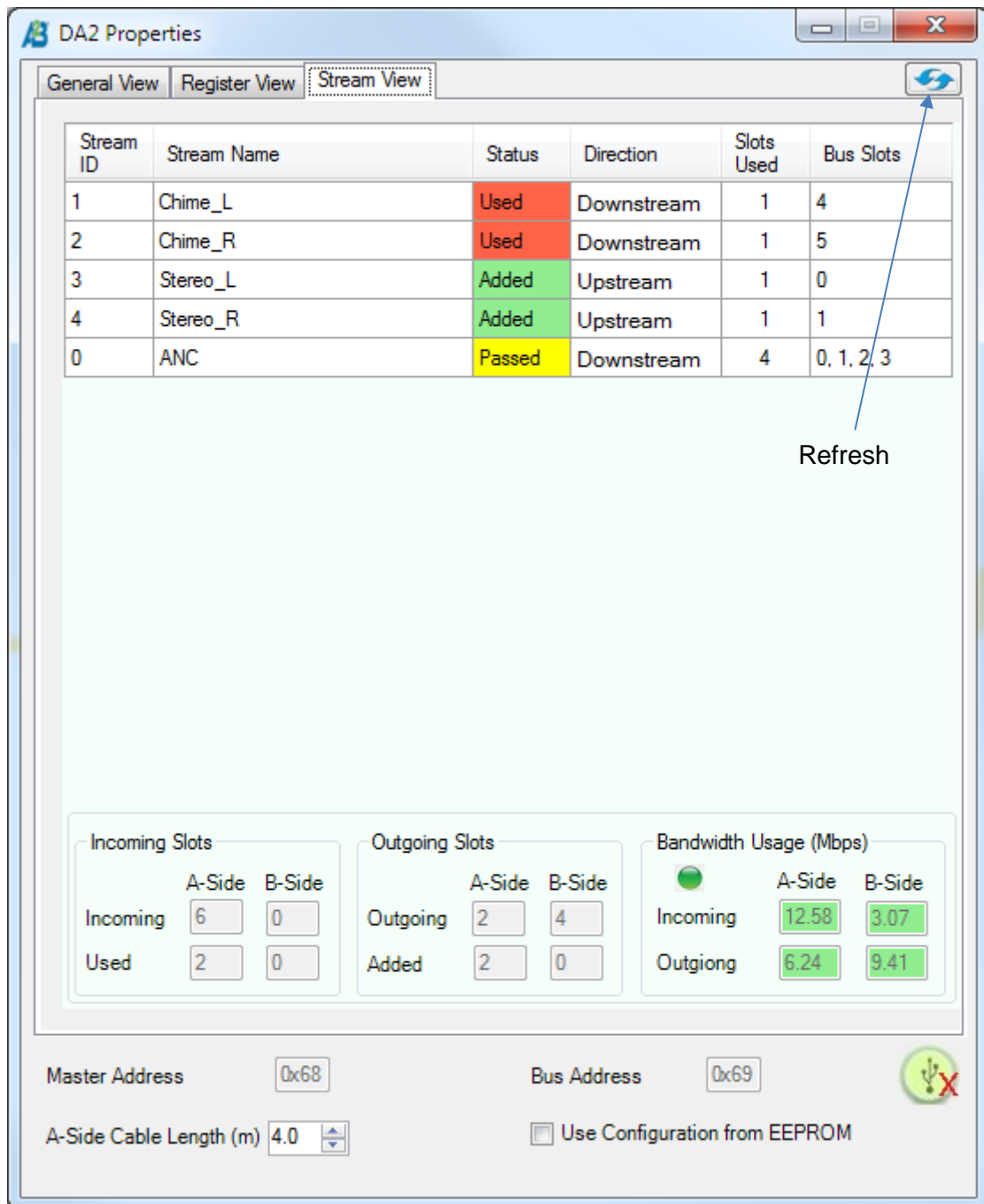


Figure 29: Stream View

3.3 Custom Node ID based Configuration

A Custom Node Identifier allows A2B software stack to determine, at run-time, whether the nodes are physically connected in the expected order so that right configuration is applied to the node. This enables system integrators to uniquely identify and authenticate a Node in an A2B network.

User can set unique ID for a node in an attached I2C enabled Memory device such as an EEPROM attached or by setting unique combination of A2B Transceiver GPIO pin(s) and levels (low/high) or even store in an attached processor memory. The Custom ID information (as set in the hardware) shall be provided in SigmaStudio so that it is used for A2B node validation during discovery process.

Custom Node Identifier can be set from the Node's 'Device Properties->General View->ID' tab.

3.3.1 I2C Device

Use the 'I2C Device' option as shown in Figure 30 when the Custom ID for the node is stored in a Memory device attached to it.

Specify the Custom ID (either as an ASCII string or as a Hexadecimal number) – Max 50 characters, device address, address width and the memory location in the device where the ID is stored.

☒ Custom Node Identifier

Read From:
☒ I2C Device ☐ GPIO Pins ☐ Mailbox

Identifier (ID): A2B Slave Node1 WBZ

☒ String (7-bit ASCII) ☐ Number (Hex)

Device Addr (7-bit): 0x50 Addr Width: 2 Address: 0x100

Number of Retries: 0

Config Audio Config Rate and ClkOut Interrupt Config Pin Config GPIO ID

Figure 30: Custom Node ID in Memory

It is possible to write Custom Node ID into an I2C enabled memory device from SigmaStudio using the 'peripheral properties' window as shown in Figure 31. The data in the XML file represents comma separated ASCII values of Custom Node ID in Figure 30.

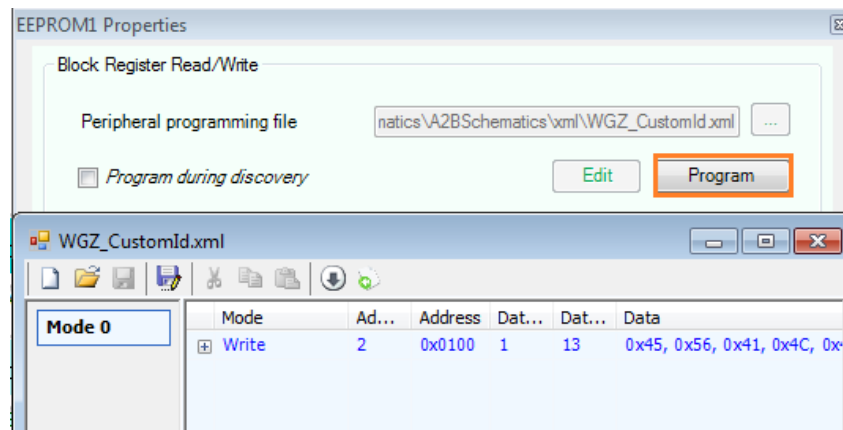


Figure 31: Writing Custom Node ID in a Memory device

3.3.2 GPIO Pins

Use 'GPIO Pins' option as shown in Figure 32 when A2B Transceiver GPIO pins are used for Custom Node Identification. Specify transceiver GPIO Pins and their level (high/low) to be matched on the actual hardware during A2B discovery process. A GPIO pin not used for Node Identification shall be set to 'IGNORE'.

As most GPIO pins are multiplexed with other functionality, user shall select pin(s) that are free for Node ID. Note that a GPIO pin will assume a multiplexed functionality only after the A2B Transceiver is discovered and configured. So, with additional on-board circuitry (pull-up/pull-downs) one should be able to use A2B GPIO pins (including multiplexed GPIO) for Custom Node ID. If multiple nodes in a network use GPIO pins for Identification, then each node shall have unique GPIO pin/level settings.

The screenshot shows the 'Custom Node Identifier' configuration window. At the top, the checkbox 'Custom Node Identifier' is checked. Below it, the 'Read From' section has three radio buttons: 'I2C Device' (unselected), 'GPIO Pins' (selected), and 'Mailbox' (unselected). A table of GPIO pin configurations is shown below, with 'GPIO1' set to 'LOW' and others to 'IGNORE' or 'HIGH'. The 'Number of Retries' is set to 0. The bottom of the window shows a tabbed interface with 'GPIOID' selected.

Read From		
<input type="radio"/> I2C Device <input checked="" type="radio"/> GPIO Pins <input type="radio"/> Mailbox		
GPIO0	HIGH	GPIO1
GPIO3	IGNORE	GPIO4
GPIO6	IGNORE	GPIO7

Number of Retries: 0

Config Audio Config Rate and ClkOut Interrupt Config Pin Config **GPIOID** ID

Figure 32: Custom Node ID using GPIO pins

3.3.3 Mailbox

Use 'Mailbox' option as shown in Figure 33 when Custom Node Identifier for the slave node is to be requested from a connected processor.

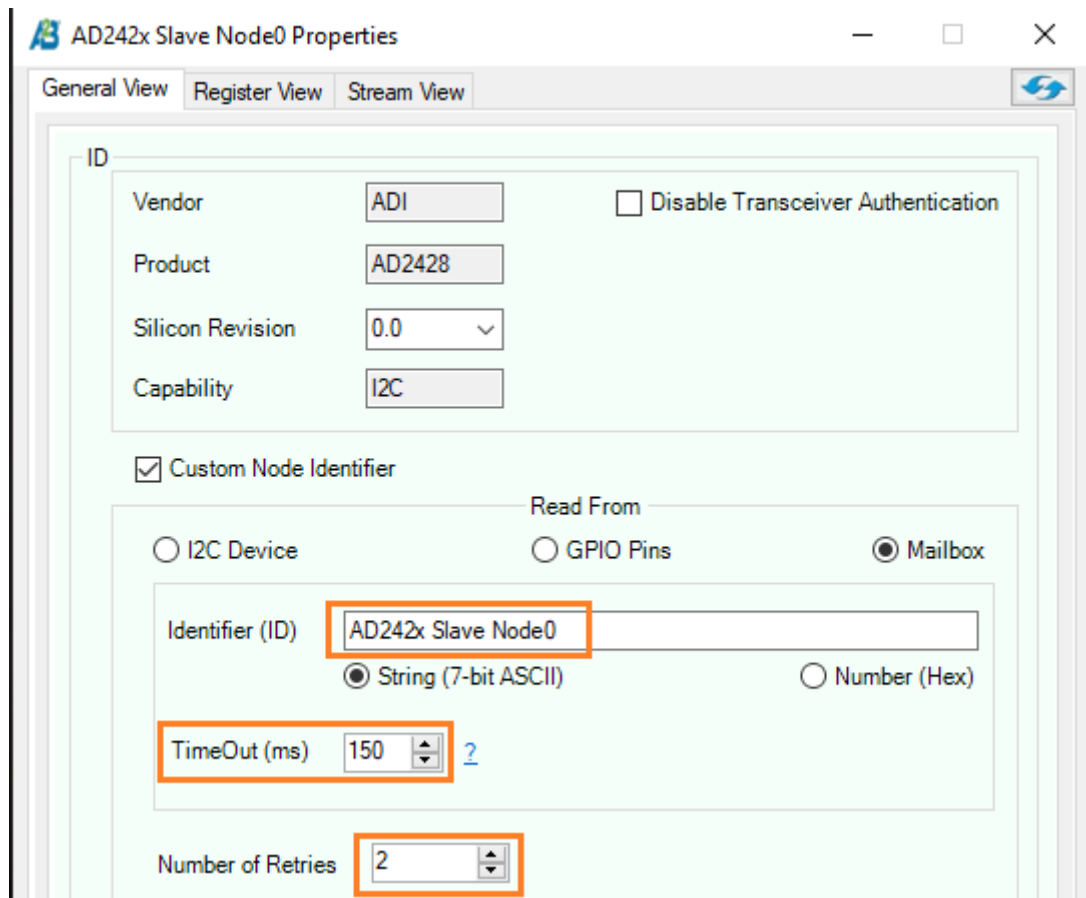


Figure 33: Custom Node ID using A2B mailbox

When a schematic is downloaded with this option set, SigmaStudio sends Custom Node Identifier request message to the slave node over A2B Mailbox, using A2B Stack's Communication channel module. The slave node processor, running the Communication channel instance, is expected to handle the request message and respond with its Custom ID as set in SigmaStudio before the 'TimeOut' period. The 'TimeOut' field specifies the time in milliseconds before which the slave node must respond with its custom Node ID. If the slave node fails to respond after 'number of Retries' or responds with an invalid ID, then SigmaStudio aborts discovery and a discovery fail message is provided to the user. For more details on communication channel to be run on slave and application integration details refer to the Communication channel integration guide [4].

3.3.4 Operability within the A2B Stack

During A2B discovery process, the Software stack will read the Custom Identifier value from the remote memory device via I2C at the specified address or will read levels on the specified GPIO lines and validate it against the assigned value. In case of mailbox authentication, the Software stack will request for the ID via the communication channel over A2B mailbox. The slave controller is expected to run the communication channel and the slave application is required to handle the

request message and respond with the custom node ID as set in SigmaStudio. In all cases if there is a match then the node will be successfully configured otherwise the discovery will be aborted. Additionally, user may register a call back after custom node authentication check, where authentication status can be overridden.

Note that a node may not have a Custom ID set, in which case the stack discovery and initialization process will not perform any Custom ID authentication and hence simply applies the configuration available for that node position.

3.4 A2B Bus Analyzer as Sub node

A2B Bus Analyzer can be used as a sub node in A2B network to monitor A2B Bus. Enable “Enable Analyzer Node” checkbox to use A2B Analyzer as a sub node. During A2B discovery process, the Software stack will allow the discovery of A2B Analyzer node, if this field is checked.

Note: A2B Analyzer as a Sub node is not supported in uncompressed bus configuration file (busconfig.c),

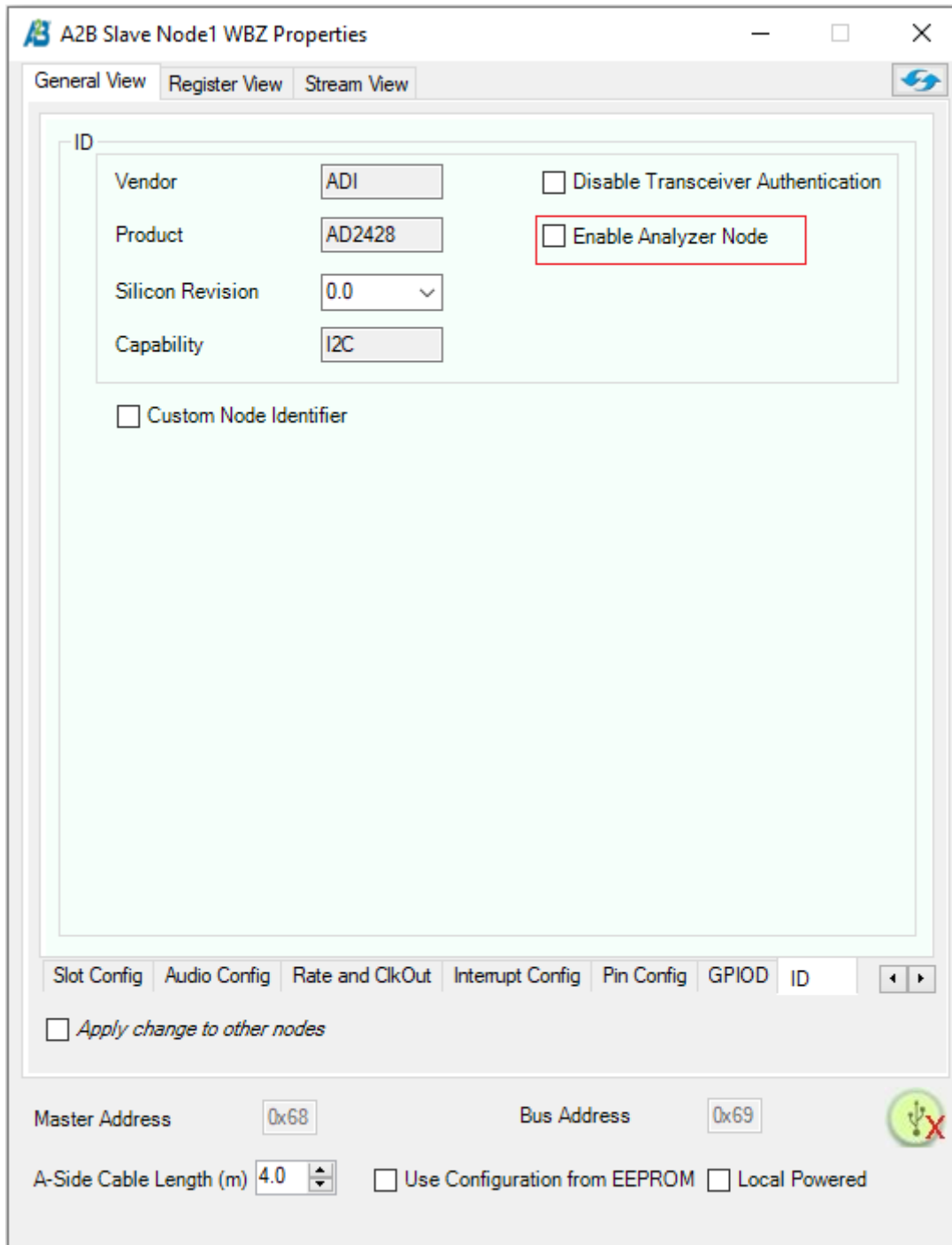


Figure 34: A2B Analyzer as sub node

3.5 Auto Configuration

In the Auto configuration scheme, a host processor can automatically configure a slave A2B node and its peripherals or a complete A2B network without prior knowledge of the topology. Typically, a non-volatile memory device like EEPROM is used to store the configuration information. Following sections describe the Auto configuration options in detail.

3.5.1 Auto Configuration of Network

A complete A2B network can be configured using the information stored in a memory device that is directly connected to the Host processor via I2C. Such memory device shall use device address 0x50 (7-bit) and accessible directly to the Host processor over I2C. Upon boot, the host processor can read this information to automatically configure the complete A2B network i.e., master and slave A2B Transceivers, network and slave node peripheral devices without any prior knowledge of the network.

3.5.1.1 Storing Network Configuration

In SigmaStudio, one can store the network configuration information into a memory device by saving the Schematic into an EEPROM connected to ECU. This can be done by right clicking on the Target processor and selecting 'Save Schematic in EEPROM'.

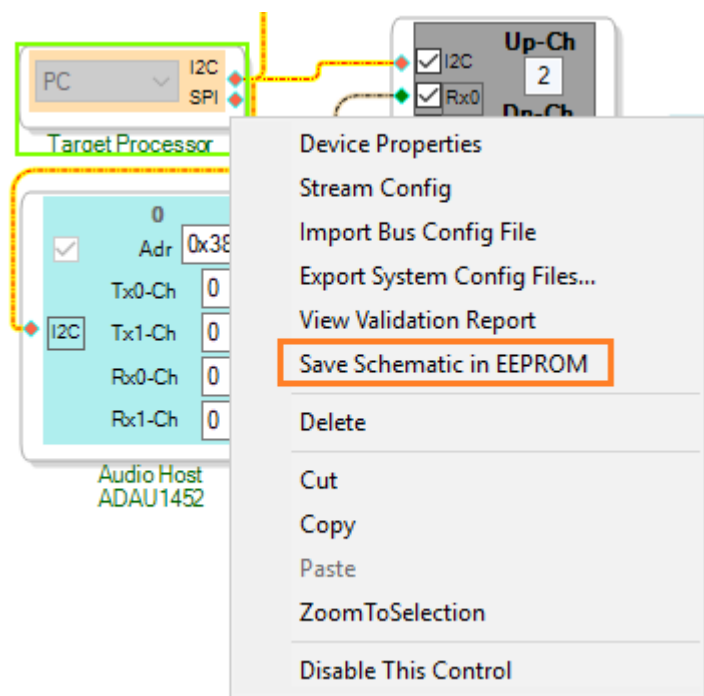


Figure 35: Saving Schematic to EEPROM

Also, there is a provision to export the information into an XML/ Dat file as shown in Figure 36. These files can be input to custom EEPROM programming utilities, specifically .dat file can be converted to hex/.s37 format for ease of flashing.

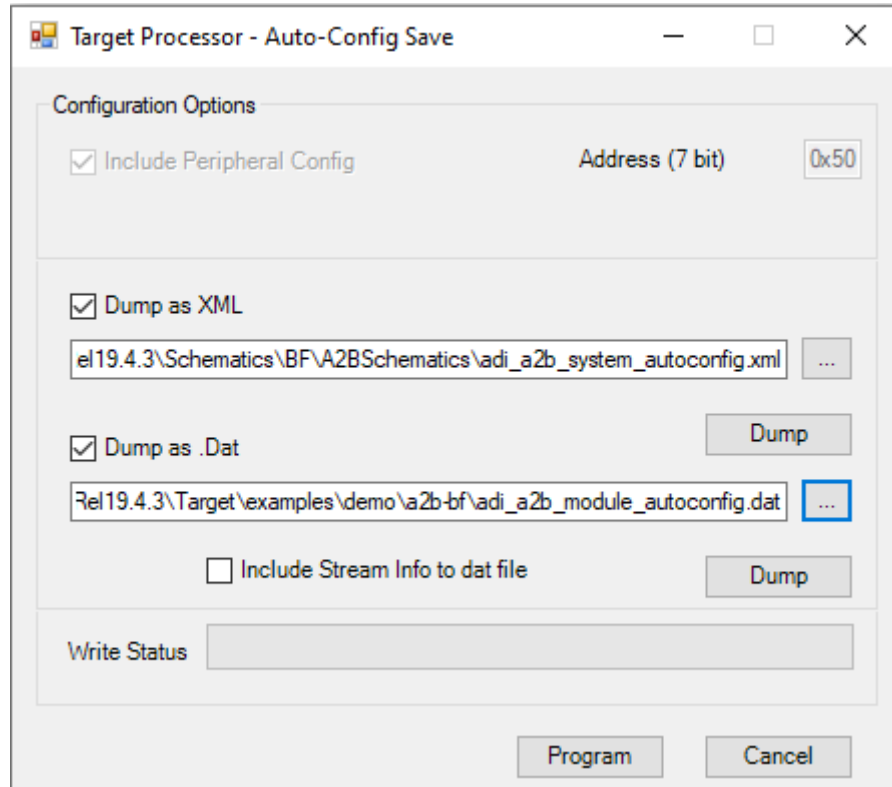


Figure 36: Schematic Dump as XML

Enable the “Include Stream info to dat file” option to export the stream configuration of network.

3.5.1.2 Loading Network Configuration

To specify A2B stack running on the Target processor can load the network wide configuration from EEPROM ensure to define the macro ‘A2B_FEATURE_EEPROM_OR_FILE_PROCESSING’ and ‘A2B_BCF_FROM_SOC_EEPROM’ and undefined ‘ADI_SIGMASTUDIO_BCF’ in .\Target\examples\demo\A2B-xx\A2Bstack-pal\platform\A2B\features.h.

Alternatively, network wide configuration can be sourced to target software from .dat file via the local file system. Refer to 4.1.1.2 for details. (.dat file from EEPROM window and adi_a2b_busconfig.dat file is essentially same)

3.5.2 Auto Configuration of Slave Nodes

An A2B Slave module can be configured using the information stored in a memory device that is directly connected to an A2B transceiver via I2C. Such memory device shall use device address

0x50 (7-bit) and accessible over the A2B bus as a peripheral. During discovery, the host processor can read this information to automatically configure the A2B node and its attached peripherals without any prior knowledge of the slave module.

3.5.2.1 Storing Module Configuration

The configuration information shall be stored in the memory device as per the format described in Appendix C of the AD242x Programming Reference Manual [2].

Using SigmaStudio, one can store an A2B slave module specific information in the attached memory device by right-clicking on the slave node and selecting 'Save Auto-config registers in EEPROM' as shown in Figure 37. This will store the A2B register information, from the node's 'Device properties' window, in the memory device. Attached peripheral configuration information, from the .XML file, is also stored if 'Program during discovery' option is set for the peripheral. This option becomes accessible only if the node is successfully discovered.

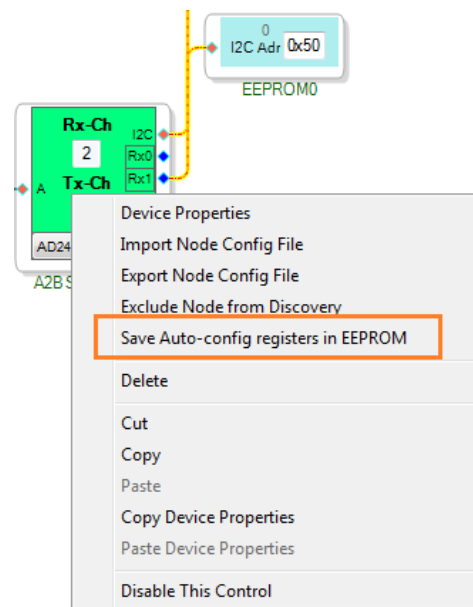


Figure 37: Saving Auto Configuration to EEPROM

3.5.2.2 Loading Module Configuration

To specify an A2B slave module to use configuration from a memory device, set the option 'Use Configuration from EEPROM' in the slave node's 'Device properties' window as shown in Figure 8. In this case user need not have to set any A2B properties/registers for the node/peripheral. If the peripheral configuration is also stored in the EEPROM then ensure that "Program during discovery" option is de-selected for the attached peripheral(s).

4 Exporting A2B System Configuration files

A2B system configuration can be exported to files so that they can be used for configuring A2B system using a microcontroller or from different development environment.

A valid error free A2B schematic can be exported into C programming language and XML formats. The C format configuration files are used in Target software for A2B network configuration. As the configuration is stored in generic C format this file can also be taken to other environments supporting C programming language.

The XML format file is more readable and thus enables easy understanding of A2B system configuration flow.

The following options are available for the user to export A2B system configuration. Each option is explained in more detail in Section 4.1 and 4.2 . These files can be used to configure A2B network.

- General View Configuration Files (C and XML format)
- I2C command list Files (XML and C format)

The following steps describe the procedure for exporting A2B system configuration files

1. Create a valid A2B schematic as explained in section 3 (or open an existing A2B Schematic project) in SigmaStudio.
2. Save and Link the schematics.
3. Right click on Target processor and select 'Export System Configuration Files...' option as shown in Figure 38.

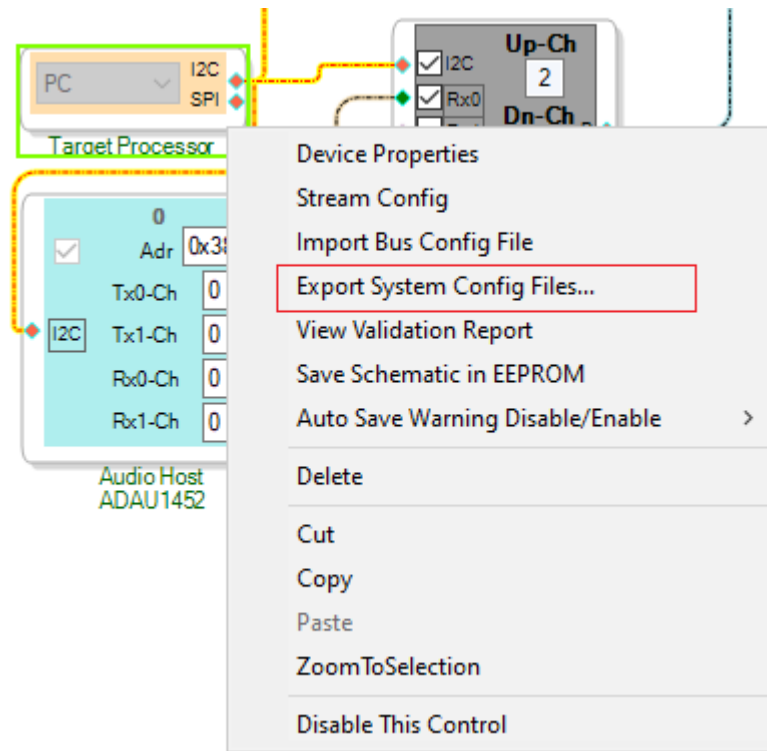


Figure 38: Exporting system configuration

4. A dialog box appears as shown in Figure 39. Export required configuration file using appropriate tab.

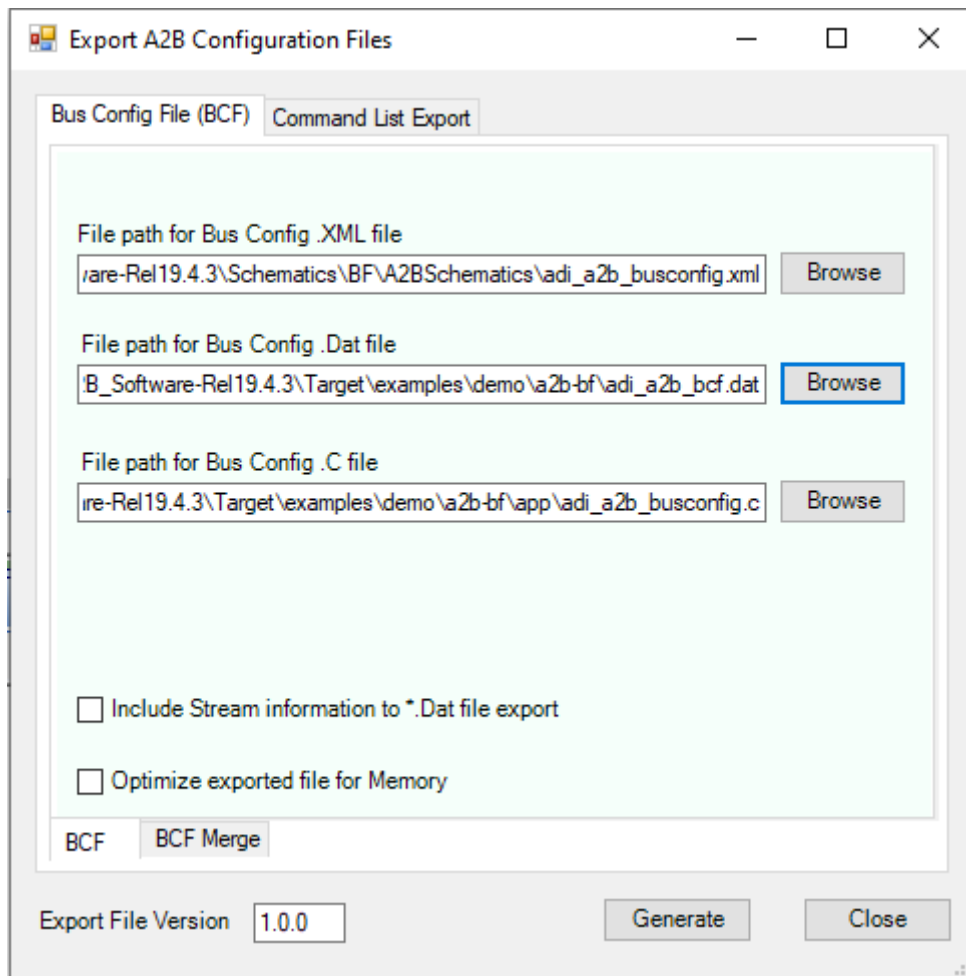


Figure 39: System Configuration File export window

Note: 1) Manual editing of exported XML is not recommended.

2) Changes done to the schematic after last Link will not be reflected in the export. Hence schematic shall be always saved and Linked before exporting.

4.1 General View Configuration files

The general view configuration files capture A2B system configuration information in more readable form and more functionality based rather than just register values. The information contained in these files correspond to the fields of 'General View' tab in the 'Device Properties' Window for all the nodes in the system.

Three files *adi_a2b_busconfig.c*, *adi_a2b_busconfig.dat*, *adi_a2b_busconfig.xml* are generated as part of BCF export.

A2B target software uses information in these files to discover, configure nodes and peripherals in the A2B system when using BF527/SHARC as the target processor or on a custom platform implementing A2B software Stack from ADI.

4.1.1 Bus Configuration Files (BCF)

4.1.1.1 adi_a2b_busconfig.c

This file stores A2B schematic as function-based settings instead of register values. The target software internally parses this file to get the register settings for each node in an A2B network. BCF also contains node level stream information (up to 32 streams).

This file shall be exported to `.\\Target\\examples\\demo\\a2b-xx\\app` path. If the compressed BCF is exported then, ensure that the macro 'ADI_A2B_BCF_COMPRESSED' is defined (`.\\Target\\examples\\demo\\a2b-xx\\app\\a2bapp_defs.h`)

4.1.1.2 adi_a2b_busconfig.dat

This file stores network configuration as binary file. The order of bytes in the file is same as the format specified for 'Network Save in EEPROM'.

Follow the following steps to use this binary file as input to A2B target software.

- Define path for the binary file - `A2B_CONF_BINARY_BCF_FILE_URL` in `a2bapp_defs.h` (`.\\Target\\examples\\demo\\a2b-xx\\app\\a2bapp_defs.h`)
- Define macro 'A2B_FEATURE_EEPROM_OR_FILE_PROCESSING' and 'A2B_BCF_FROM_FILE_IO' in `feature.h` (`.\\Target\\examples\\demo\\a2b-xx\\a2bstack-pal\\a2b\\feature.h`)
- Ensure that `ADI_SIGMASTUDIO_BCF` is **not** defined (in `feature.h`)

The advantage of binary file is that network configuration doesn't have compile time dependency with the target software.

4.1.1.2.1 Stream Information

Stream information can be made part of `.dat` and **EEPROM** at the end by selecting check box as shown below.

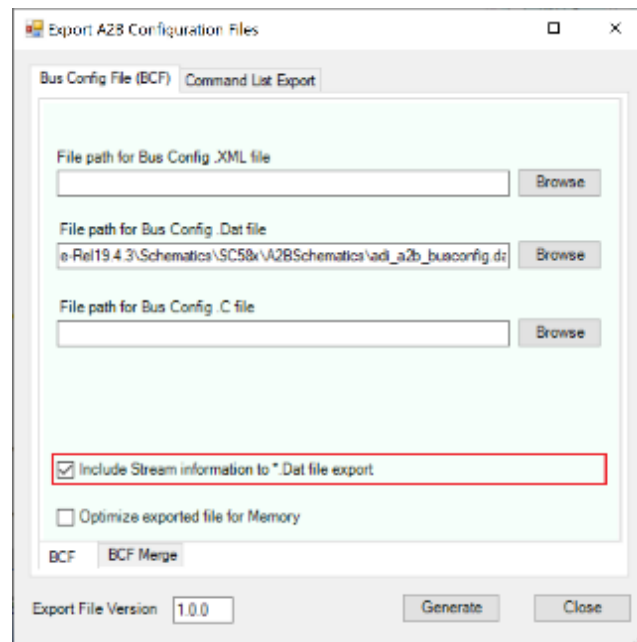


Figure 40: EEPROM and .dat export with Stream Information

The Stream information content of .dat/EEPROM is as follows.

Level3	
Stream Information Config marker - 0xAB	1
Reserved byte	1
Stream Information Length	2
Pointer for Stream Definition	2
Pointer for Data tunnel Definition(Demeter)	2
Number of nodes	1
Main Node Pointer (2 Bytes)	2
Sub Node0 Pointer (2 bytes)	2
Sub Node1 Pointer (2 Bytes)	2
.....	2
.....	2
Stream Definition marker - 0xAB	1
Number of Streams	1
Stream marker - 0xAB	1
Stream ID	1
Stream Name	64
Sampling Rate of Stream	3
Data Width	1

Number of channels	1
Channels to Skip	1
Stream Source Node ID	1
Stream Destination count	1
Stream Destination Node ID	1
Stream Destination Node ID	1
.....	1
Stream marker - 0xAB	
Stream ID	1
Stream Name	64
Sampling Rate of Stream	3
Data Width	1
Number of channels	1
Channels to Skip	1
Stream Source Node ID	1
Stream Destination count	1
Stream Destination Node ID	1
Stream Destination Node ID	1
.....	1
.....	64

Level4

Node Marker - 0xAB	1
Data tunnel present	1
NodeID	1
Node Name	64
Number of Sink streams	1
Pointer to corresponding stream	2
Bus slots index	1
Direction	1
Pointer to corresponding stream	2
Bus slots index	1
Direction	1
.....	N - times
Number of Source streams	1
Pointer to corresponding stream	2
Bus slots index	1
Direction	1

Pointer to corresponding stream	2
Bus slots index	1
Direction	1
.....	N - Times
Number of Passed streams	1
Pointer to corresponding stream	2
Bus slots index	1
Direction	1
Pointer to corresponding stream	2
Bus slots index	1
Direction	1

Note: Above levels will be available in .dat/EEPROM only when stream information is included.

4.1.1.3 adi_a2b_busconfig.xml

This file has schematic information in XML format which can be imported back. This allows user to apply different configurations to A2B system and verify the system behavior.

Note: Manual editing of exported XML file is not recommended.

The BCF file in XML format can be imported back into A2B schematic as shown in Figure 41. Note that BCF import feature can auto draw the nodes/peripherals and import the settings. Schematic is auto saved after importing XML.

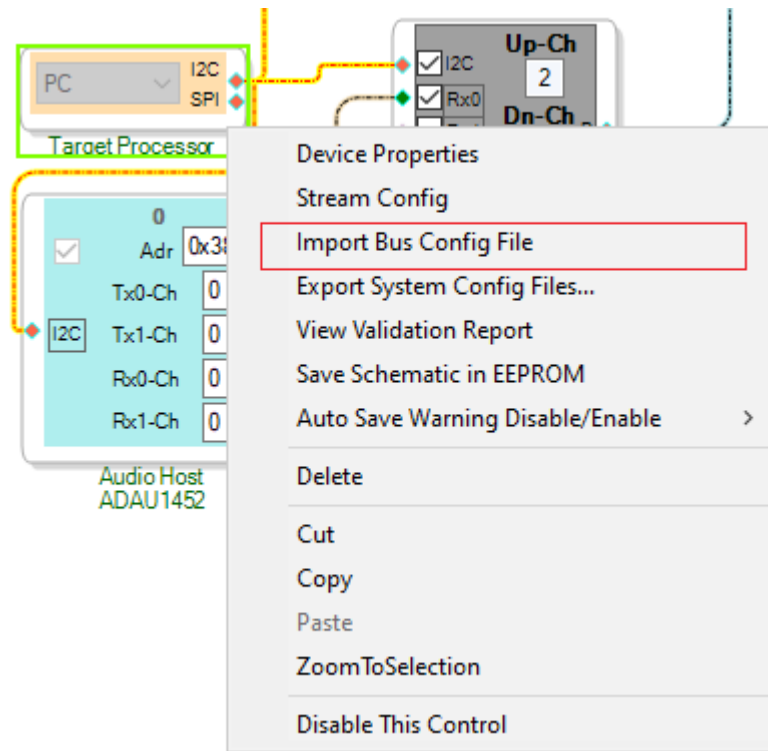


Figure 41: BCF import

Note: Save and Click on the 'Link' icon in SigmaStudio if BCF import option is disabled.

4.1.2 Super Bus Configuration Files (Super BCF)

A Super Bus Configuration file is a super-set file containing information of multiple A2B configuration. This file is generated by providing individual BCF XML files, corresponding to different A2B configuration, as shown in Figure 42. The Super Bus Configuration file is generated in XML and C formats. The .C format file is intended for use in the Target software application.

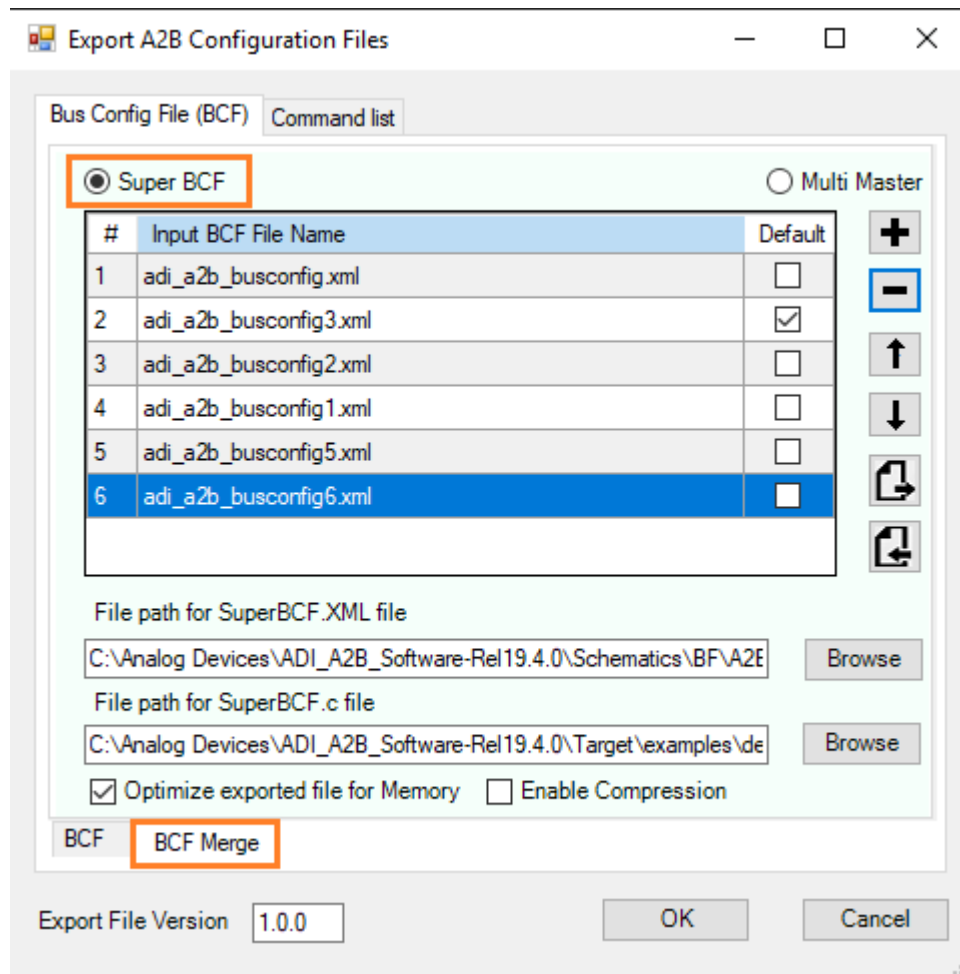


Figure 42: Super Bus Configuration File export

Note that there is an option to set one of the input BCF file as 'Default'. This means the file set as default will be used to configure the A2B network when the order of the physically connected A2B nodes does not match with any of the other input BCF files. The Default BCF file defines the 'Limp-home' mode behaviour of the network when the nodes in the network are out of order. Additionally, this window allows the export and import of BCF file list (File path is stored relative to project path).

4.1.2.1 adi_a2b_superbusconfig.c

This file contains multiple A2B configurations in .C format where each configuration is accessible over an Index. Using this file, the Target software application, at run-time, can switch between different network configurations (including different node order when used with Custom Node IDs) without having to change or update the software on the Master (Variant support)

If including this file in the Target software (at .\Target\examples\demo\a2b-xx\app) ensure to enable Macro 'ENABLE_SUPERBCF' in the file .\Target\examples\demo\a2b-xx\app\a2bapp_defs.h.

Note: Maximum number of BCFs in superBCF is controlled by the macro `A2B_CONF_MAX_NUM_BCD` (default value 40) in 'adi_a2b_busconfig.h'

4.1.2.2 adi_a2b_superbusconfig.xml

This file contains multiple A2B configuration in a readable XML format. The XML version has no significance expect that it is more readable and sharable.

4.1.3 Multi Master Bus Configuration File

The multi master Bus configuration file is like BCF except that multiple A2B masters (parallel A2B Buses) are represented in the single BCF structure. If single Target processors controls multiple A2B network, this option can be used where individual master-slave chains are aggregated to form single structure. Unlike Super BCF, all the configurations are active once the network is set up.

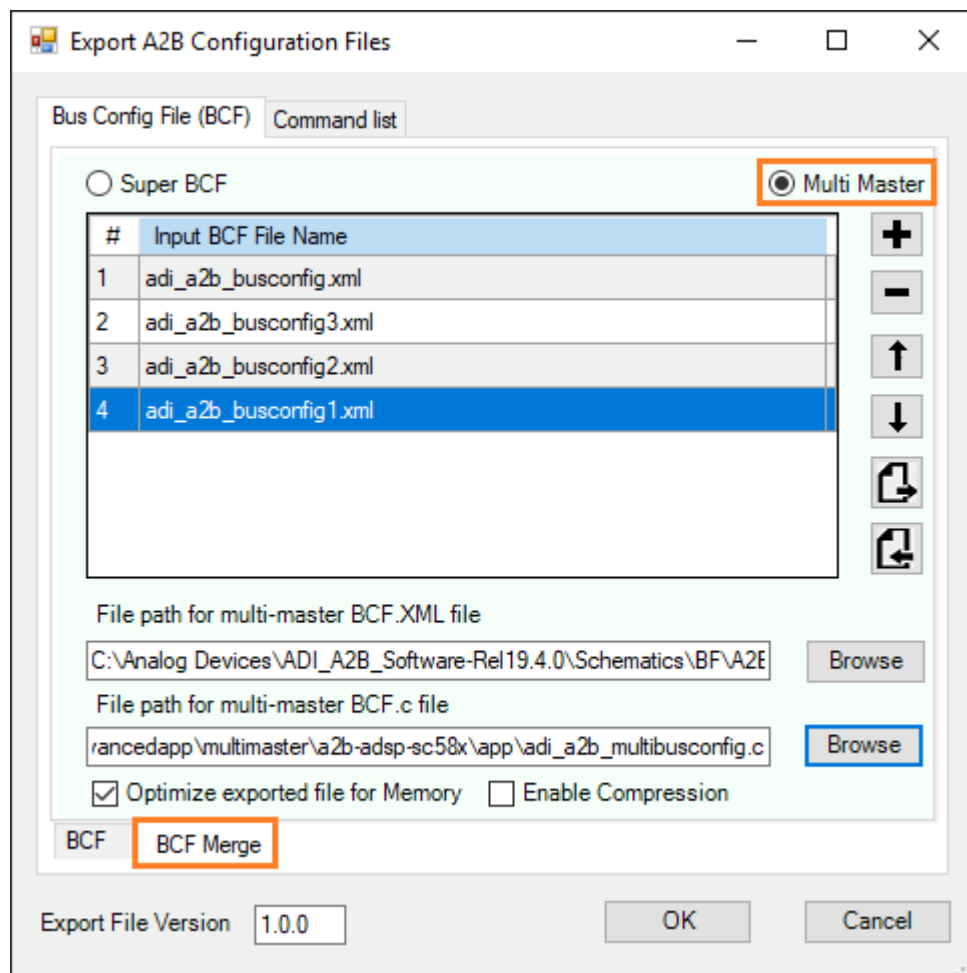


Figure 43: Multi master Bus Configuration File export

4.1.3.1 adi_a2b_multibusconfig.c

This file contains A2B configurations in .C format corresponding to individual daisy chain of master and slave nodes. Using this file, the Target software application can configure and control multiple, (parallel) A2B network simultaneously.

If including this file in the Target software (like .\Target\examples\advancedapp\multimaster\adsp-sc58x\app) ensure that the macro `A2B_CONF_MAX_NUM_MASTER_NODES` (.Target\examples\advancedapp\multimaster\adsp-sc58x\adbstack-pal\platform\adsp-sc58x\conf.h) matches (macro can be larger) with the multi master BCF file. For more details on multi master bus set up, refer A2B Stack user guide [3]

Note: Target properties (like discovery mode, line fault settings) from the first bus config file (XML) is considered for export. Configuration from the other bus config files are ignored.

4.1.3.2 adi_a2b_multibusconfig.xml

This file contains multi master Bus configuration in a readable XML format. Significance of this file is limited to configuration sharing.

4.2 I2C Command List files

I2C command list files enable users to bring up A2B system without having to use the Target software provided with the release package. All that is required to bring up an A2B system using I2C command list is to just have an I2C driver specific to the controller used for programming A2B Transceivers.

The I2C command list can be exported in C and XML formats and are explained in the following sections.

4.2.1.1 adi_a2b_commandlist.h

This file contains sequence of I2C commands (Write/Read/Delay) to be programmed for discovering and configuring A2B transceiver nodes and peripherals as per the drawn schematic. This file is saved in the form of a C header file so that this can be included and used in any custom project using a different microcontroller/DSP platform for quick evaluation.

An example project using I2C command list for A2B network discovery and configuration is available at .\Target\adsp-sc58x\commandlist.

Note: I2C command list file does not provide A2B interrupt or event/fault handling routines. The software running on the controller shall be responsible for handling such events/faults.

4.2.1.2 adi_a2b_commandlist.xml

The sequence of I2C commands are also generated in XML format for easy readability and understanding. The generated XML file can be opened from 'Direct I2C programming window'. Refer Section 2.2.1.1.2 for details on the usage of this window.

4.3 Node Configuration File (NCF)

Like exporting entire A2B system configuration into files even an A2B node configuration can also be exported into a file. A node configuration XML file will contain all information related to the exported node along with the information of the peripherals connected to it.

Node configuration file can be used to apply different configurations for a node or for applying same configuration to similar nodes using the import feature.

The following steps describe NCF generation

1. Draw A2B schematics as described in Section 3 . Save and Link the schematics
2. Right click on the node to be exported and select '*Export Node Config File*' option as shown in Figure 44.

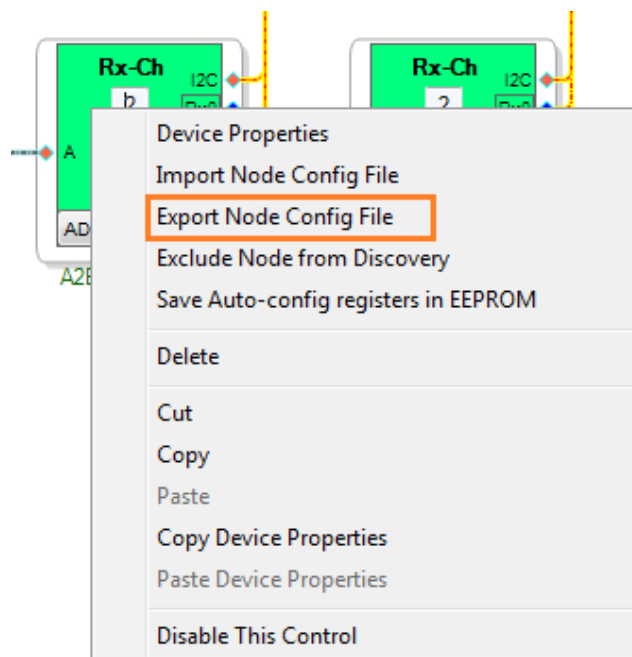


Figure 44: NCF Export

3. Check the '*Node Configuration File (XML)*' option and provide the file name & location in the Dialog box as shown in Figure 45. Enter the version number for the exported file and then press OK.

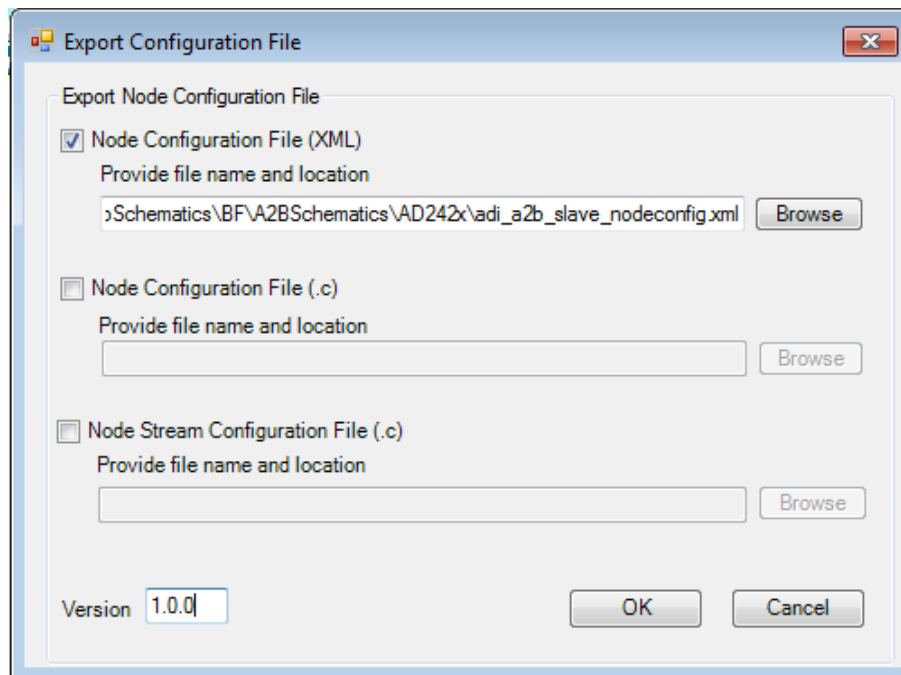


Figure 45: NCF export window

Note 1: Manual editing of exported XML is not recommended.

Note 2: Changes done to the schematic after last Link will not be reflected in the export. Hence Schematic shall be always Saved and Linked before exporting an NCF.

Node configuration file in XML format can be imported to an A2B node as shown in Figure 46. This allows user to apply different configurations to a node and verify the system behavior.

Note that NCF import feature can (19.4.0 onwards) auto draw the connected peripherals and apply the settings accordingly.

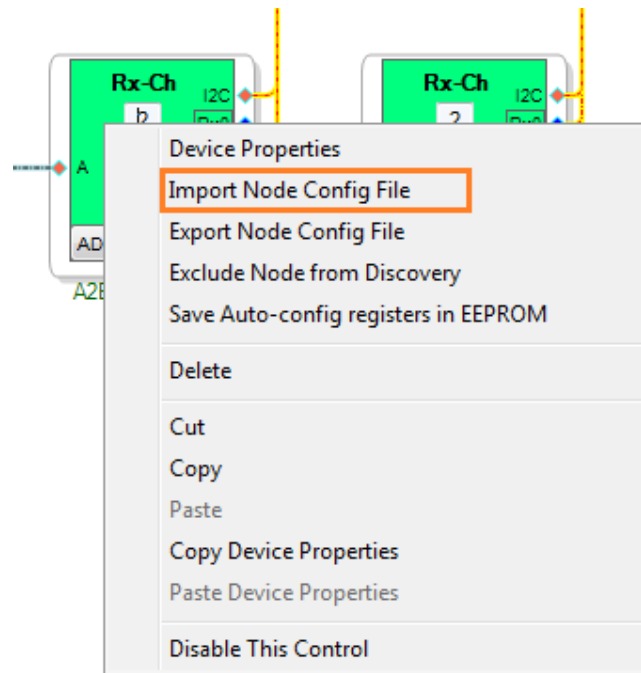


Figure 46: NCF import

4.4 Working with Configuration XML files

The following sub-sections describe the workflow associated with using different configuration XML files, exported from SigmaStudio, when building an A2B System.

4.4.1 Workflow using NCF and BCF files

Figure 47 shows a typical workflow between A2B node suppliers and system architects/integrators when building an A2B system using NCF and BCF.

A node supplier is responsible for designing the node's functionality/TDM interfaces and verifying the design on SigmaStudio. A verified node's configuration, exported as NCF XML, shall be provided to A2B system integrator who would import individual node configuration files to build the final A2B system schematic on SigmaStudio. A BCF .C file exported from such a verified schematic shall be used in the Target software.

Note that Custom Node Identifier needs to be set for nodes if node configuration is to be applied only upon Custom Node ID authentication as explained in Section 3.3 .

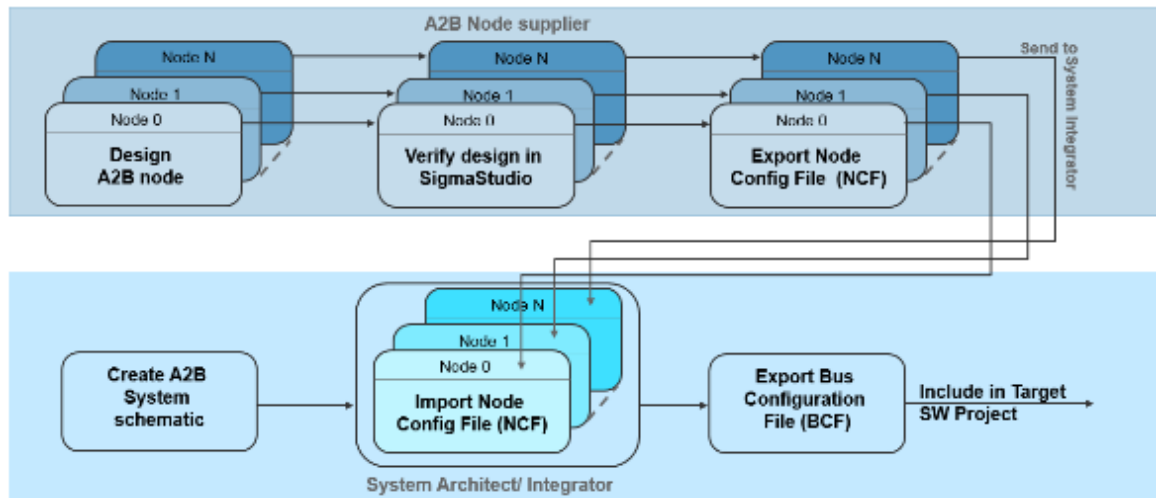


Figure 47: Work flow using NCFs and BCF

4.4.2 Workflow using BCF and Super BCF Files

A Super BCF file provides the ability to shift the order of nodes in an A2B network for different systems/configurations without the need to change the Target software. This requires Custom Node IDs to be set in each node as explained in Section 3.3 .

To develop such a flexible system, a system architect shall determine all possible network combinations that the final system should support. A schematic corresponding to each network combination shall be built on SigmaStudio by importing individual NCF XML files from each node supplier. For each verified A2B schematic a BCF XML file shall be exported which then can be merged into a single Super BCF using the option shown in Figure 42. The generated Super BCF .C file shall be used in the Target software. At run time, Target processor can switch between different network configurations. This work flow is shown in Figure 48.

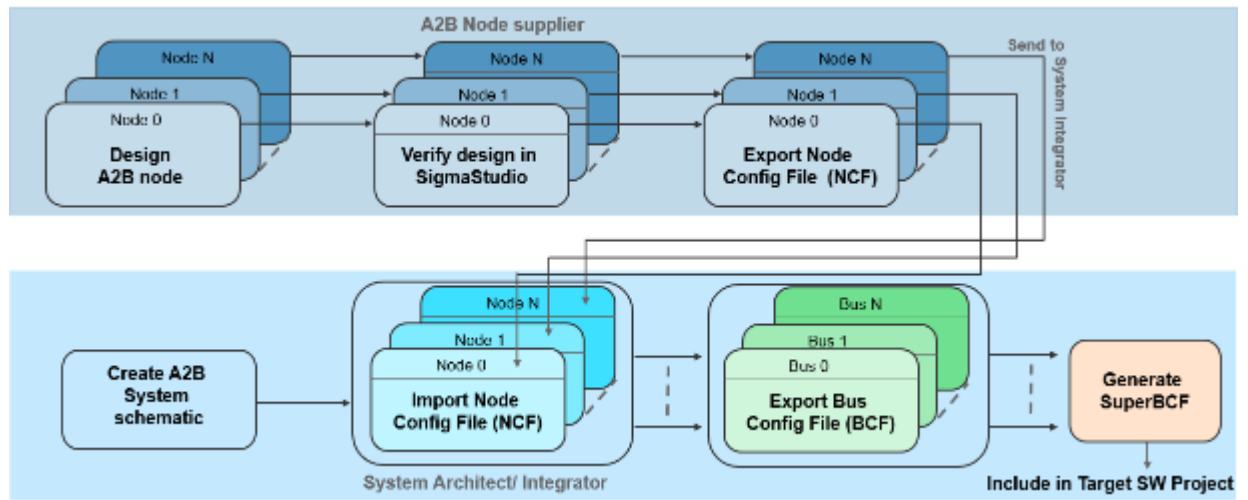


Figure 48: Work flow using BCFs and SuperBCF

5 Network Analysis and Debug

SigmaStudio allows user to estimate important network parameters like Bandwidth and power consumption. It also allows user to monitor real-time bit errors in the system using pseudo or actual bit stream. This option can be accessed using the Right-Click menu of the Master Transceiver node as shown in Figure 49. Note that *calculate...* option is accessible only if the A2B schematic is 'Link'ed after the last schematic modification. Save schematic and click on the 'Link' icon if the option is disabled.

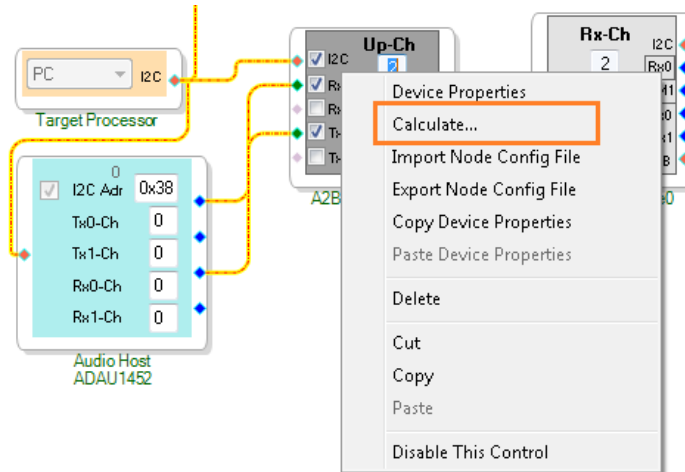


Figure 49: Calculate Option

5.1 Bandwidth Calculation

The Bandwidth calculation tab allows user to estimate the total bandwidth used in the A2B network. Bandwidth calculation is useful to design an efficient A2B network.

The following steps are involved in Bandwidth calculation

1. Create an A2B schematic (or open an existing A2B Schematic project) in SigmaStudio.
2. Right click on master transceiver node and choose *calculate...* option. This will open *Calculate A2B network parameters* window as shown in Figure 50.
3. Select the Bandwidth Tab.
4. Enter required details (*Cable delay and Sampling Rate* parameters) for calculation. Other values required for Bandwidth calculation like Slot format, data width and number of upstream /downstream slots etc. will be picked up from the device properties window settings.
5. Press *Run* button to get the bandwidth information as shown in Figure 50.

Calculate A2B network parameters

Bit Error Rate Bandwidth Power

Settings

SCF Length (bits) 64

SRF Length (bits) 64

Super Frame Length (bits) 1024

Delay from A to B port (bits) 9

Delay from B to A port (bits) 11

Cable Delay (ns/m) 6.5

Sampling Rate 48 kHz

Response cycle

☒ Edit RESPCYCS

Master Response Cycles 0x93

☒ Auto-Calculate

☐ Conservative

☒ Accurate

Update Slave RESPCYCS Apply

Run Reset

Results

SUCCESS:

Maximum Activity	:398 bits	38.87%
Required Idle Time	:124 bits	12.11%
Overall Idle Time	:626 bits	61.13%
Bandwidth utilized	:522 bits	50.98%

Figure 50: Bandwidth Calculation

- Press Save button to store the results in a text file. Detailed break-up of bandwidth usage can be visualized by clicking on the *Graph* icon as shown in Figure 51.

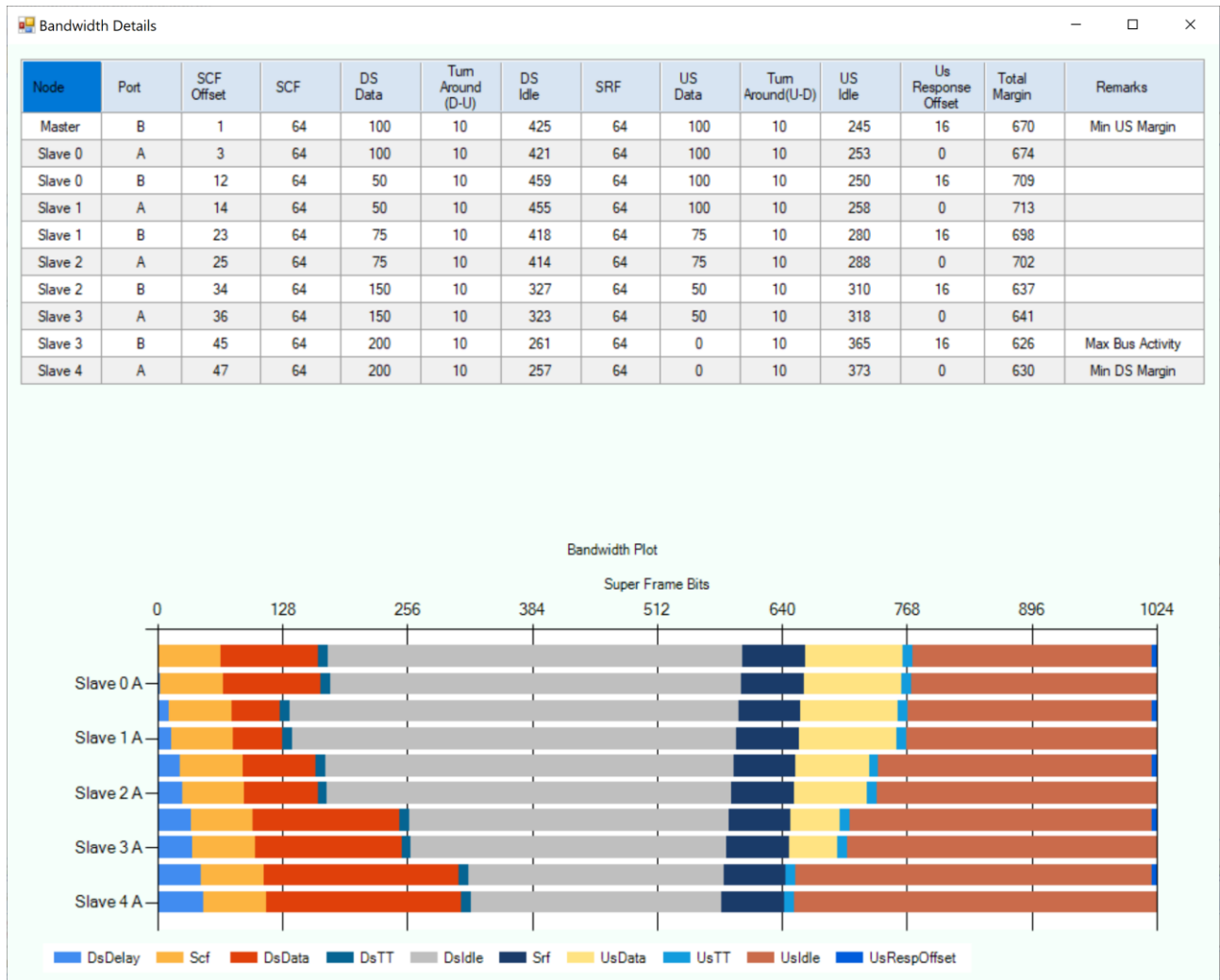


Figure 51: Bandwidth break-up

- Optionally, user can edit Master node response cycles (RESPCYCS) to better utilize the overall bandwidth.
- Reset* button can be pressed to clear the results and rerun the calculation.

5.2 Bit Error Rate (BERT) Monitoring

Bit errors generated over an A2B network can be monitored in the GUI. BERT calculation is useful to design an efficient error-free A2B network.

The following steps are involved in Bandwidth calculation

- Create an A2B schematic (or open an existing A2B Schematic project) in SigmaStudio.

2. Right click on master transceiver node and choose *calculate...* option. This will open *Calculate A2B network parameters* window as shown in Figure 52.
3. Select the Bit Error Rate Tab.
4. Select the *Bit Stream* type to be monitored for errors. Select '*Pseudo Random*' option to monitor errors on pseudo random bit stream else select '*Audio*' option to monitor errors on actual audio bit stream.
5. Select required errors to be monitored and make required display settings. Refer to the tool tip for more information by hovering the mouse pointer over the fields.

Display Update Rate (s) defines the rate at which error counts are read and updated on the display.

Graph Display Mode sets the BERT graph display mode to either Time vs. Bit error count or Time vs. Bit error ratio.

6. Press *Run* button to get the bit error information as shown in Figure 52.

Calculate A2B network parameters

Bit Error Rate | Bandwidth | Power

Settings

Bit Stream Type

☐ Pseudo Random ☒ Audio

☐ Node-to-Node Checking

☒ DRCERR ☒ ICRCERR

☒ DDERR ☐ DPERR

Display | Generate Error

Display Update Rate (s) 1.0

☒ Auto-Reset Interval (s) 5.0

☐ Show Error difference

Graph Display Mode Error Count

☐ Beep on Bit Error

Frequency (Hz) 1000

Duration (s) 1.0

Results

<Time>	<Node>	<Bits Checked>	<Bit Error Count>	<BER/Ratio>
<3:04:03>	<Master>	<0004800000>	<0000000000>	<0.00>
<3:04:03>	<Slave 0>	<0007200000>	<0000000000>	<0.00>
<3:04:03>	<Slave 1>	<0005280000>	<0000000000>	<0.00>
<3:04:04>	<Master>	<0004800000>	<0000000000>	<0.00>
<3:04:04>	<Slave 0>	<0007200000>	<0000000000>	<0.00>
<3:04:04>	<Slave 1>	<0005280000>	<0000000000>	<0.00>
<3:04:05>	<Master>	<0004800000>	<0000000000>	<0.00>
<3:04:05>	<Slave 0>	<0007200000>	<0000000000>	<0.00>
<3:04:05>	<Slave 1>	<0005280000>	<0000000000>	<0.00>

Figure 52: BERT Calculation

7. Optionally, Bit errors can be generated from *Generate Error* Tab.
8. Bit errors can be visualized in graphical form by clicking on the *Graph* icon as shown in Figure 53.

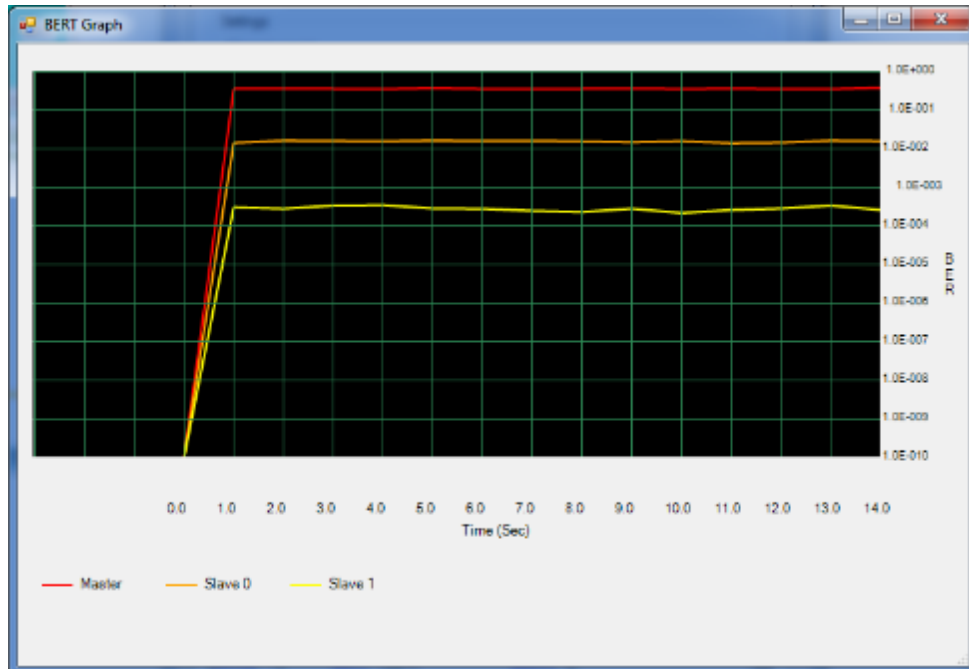


Figure 53: Graphical representation of bit errors

9. Press *Save* button to store the results in a text file.
10. Optionally, *Reset* button can be pressed to clear the results and restart the calculation.

5.3 Power Calculation

Overall power consumption of the A2B network can be estimated using the Power Calculation tab. Power calculation can assist board designers in estimating power requirements for power supply design.

The following steps are involved in Power calculation

1. Create an A2B schematic (or open an existing A2B Schematic project) in SigmaStudio.
2. Set power source and cable length in the Node 'Device Properties' window as shown in Figure 54

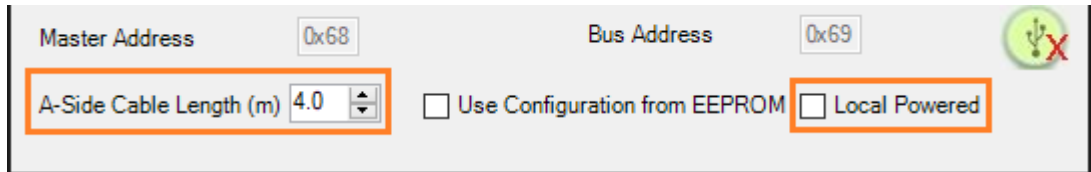


Figure 54: Power Source

3. Right click on master transceiver node and choose *calculate...* option. This will open *Calculate A2B network parameters* window as shown in Figure 50.
4. Select the Power Tab.
5. Provide values for power calculation parameters of each A2B node in the system. Select the desired power calculation Mode.
6. Press *Run* button to get the power consumption information as shown in Figure 55

The screenshot shows the 'Calculate A2B network parameters' window with the 'Power' tab selected. The 'Settings' section includes tabs for Master, Slave 0, Slave 1, Slave 2, Slave 3, and Slave 4. The 'Local Powered' button is active. The 'A-side Cable Length' is set to 'N/A'. The 'Current (mA)' section has input fields for Peri Supply 1 (0.00), Peri Supply 2 (1.00), IOVDD 1 (0.00), IOVDD 2 (2.50), and Ext Reg Peri (0.00). The 'Resistance (Ohm)' section has input fields for Inductor DC (0.38), Current Protection (1.13), Bias+ Switch On (0.16), Bias- Switch On (1.23), Connections (0.03), and Cable (0.20). The 'Voltage (V)' section has input fields for Reg Bias (9.00), Diode1 Drop (0.40), and Diode3 Drop (0.45). The 'Modes' section has radio buttons for Normal (selected) and Standby. There are 'Run' and 'Reset' buttons. The 'Results' section shows a green 'SUCCESS' message and a 'SUMMARY' table.

SUMMARY	
Max Node Dissipation	: 244.40mW
Thermal Power Margin	: 388.60mW ($T_a=105^{\circ}\text{C}$, $T_j=125^{\circ}\text{C}$)
Minimum Vin	: 006.40V
Max Bus Power	: 002.99W

Figure 55: Power Calculation

7. Press Save button to store the results in a text file. Detailed break-up of power usage can be visualized by clicking on the *Graph* icon as shown in

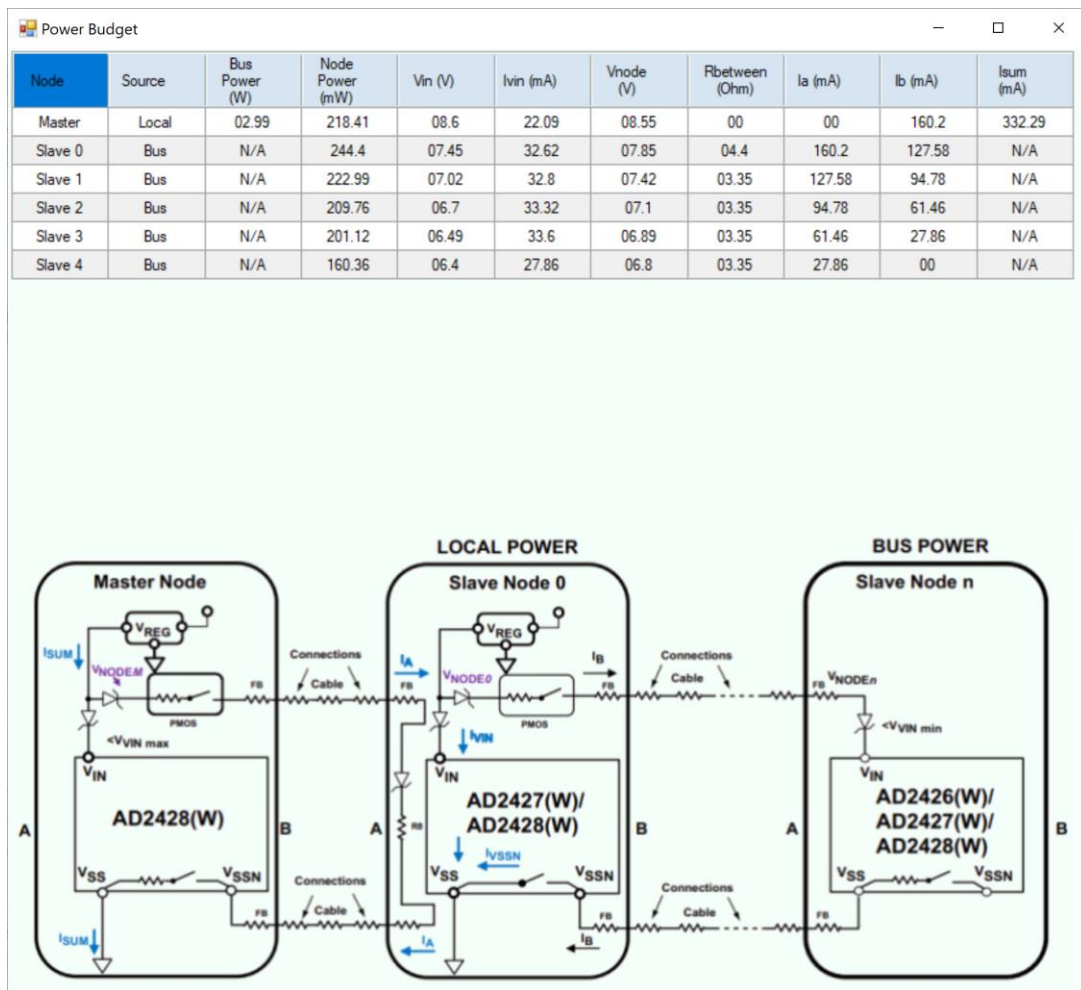


Figure 56: Power Budget

- Optionally, *reset* button can be pressed to clear the results and restart the calculation.

Note: GUI is displaying only customizable parameters. For calculation, constant/recommended parameters are also considered. For details on power calculation concepts, refer A2B transceiver data sheet.

5.4 Debug

5.4.1 Trace

The Trace option in the Target processor properties window show in Figure 3 enables user to log various transactions and network events into a .txt file. Upon enabling this feature the A2B stack running inside SigmaStudio captures all transactions as per the 'Level' and 'Domain' set.

Figure 57 shows a sample Trace file generated for selected Level (Debug, Info, Trace3) and Domain (Plugin, Message, I2C). The Trace file gets saved inside the 'settings' folder corresponding to the A2B schematic project file location.

```

0000001440 stack\stack\stack\src\discovery.c(3615) [DEBUG] A2B Master Plugin dscvryReset(): Starting DiscoveryMode 0
0000001460 stack\stack\stack\src\discovery.c(3659) [DEBUG] A2B Master Plugin dscvryReset(): ...Reset Delay...
0000001470 stack\stack\stack\src\msgRtr.c(851) [DEBUG] a2b_msgRtrNotify(m: 0x9bf590, cmd: 3, ud: 0x9bc600)
0000001490 stack\stack\stack\src\discovery.c(2918) [INFO] A2B Master Plugin Master Node: Silicon vid/pid/ver: AD/10/21
0000001490 stack\stack\stack\src\discovery.c(1238) [TRACE3] A2B Master Plugin NodeInterruptInit(): setIntrMask(0xFFEFF)
0000001520 stack\stack\stack\src\discovery.c(3027) [DEBUG] A2B Master Plugin dscvryPreMasterInit(): ...Waiting for INTTYPE.DSCDONE...
0000001530 stack\stack\stack\src\msgRtr.c(851) [DEBUG] a2b_msgRtrNotify(m: 0x9bf590, cmd: 3, ud: 0x9bc600)
0000001540 stack\stack\stack\src\discovery.c(3253) [INFO] A2B Master Plugin nodeDiscovered(): Silicon node/vid/pid/ver/cap: 00/AD/25/00/01
0000001540 stack\stack\stack\src\discovery.c(3439) [INFO] A2B Master Plugin nodeDiscovered(): BDD node/vid/pid/ver: 00/AD/25/00
0000001550 stack\stack\stack\src\discovery.c(2839) [DEBUG] A2B Master Plugin dscvryPreSlaveInit(): ...Waiting for INTTYPE.DSCDONE...
0000001570 stack\stack\stack\src\msgRtr.c(851) [DEBUG] a2b_msgRtrNotify(m: 0x9bf590, cmd: 3, ud: 0x9bc600)
0000001580 stack\stack\stack\src\discovery.c(3253) [INFO] A2B Master Plugin nodeDiscovered(): Silicon node/vid/pid/ver/cap: 01/AD/25/00/01
0000001580 stack\stack\stack\src\discovery.c(3439) [INFO] A2B Master Plugin nodeDiscovered(): BDD node/vid/pid/ver: 01/AD/25/00
0000001580 stack\stack\stack\src\discovery.c(2728) [INFO] A2B Master Plugin PreSlaveInit(): No more BDD slave nodes
0000001620 stack\stack\stack\src\discovery.c(1238) [TRACE3] A2B Master Plugin NodeInterruptInit(): setIntrMask(0x7F6F)
0000001630 stack\stack\stack\src\plugin_slave.c(387) [INFO] a2b_pluginOpen: opening slave plugin for nodeAddr = 1
0000002020 stack\stack\stack\src\plugin_slave.c(588) [INFO] Slave1 Plugin: A2B_MSGREQ_PLUGIN_PERIPH_INIT processed (rate: 0)
0000002070 stack\stack\stack\src\discovery.c(1238) [TRACE3] A2B Master Plugin NodeInterruptInit(): setIntrMask(0x7F7F)
0000002070 stack\stack\stack\src\plugin_slave.c(387) [INFO] a2b_pluginOpen: opening slave plugin for nodeAddr = 0
0000002100 stack\stack\stack\src\plugin_slave.c(588) [INFO] Slave0 Plugin: A2B_MSGREQ_PLUGIN_PERIPH_INIT processed (rate: 0)
0000002140 stack\stack\stack\src\discovery.c(1238) [TRACE3] A2B Master Plugin NodeInterruptInit(): setIntrMask(0xFFEFF)
0000002160 stack\stack\stack\src\discovery.c(971) [DEBUG] A2B Master Plugin dscvryEnd(): == Discovery Ended ==

```

Figure 57: Sample Trace file

5.4.2 Sequence Chart

The Sequence Chart option in the Target processor properties window show in Figure 3 enables user to view various transactions and network events as a rich graphical sequence diagram (.png).

The sequence diagram shows interactions between different stack modules arranged in time sequence and the messages exchanged between the stack modules in order to discover, configure and handle various network events.

The Sequence Chart file gets saved inside the 'settings' folder corresponding to the A2B schematic project file location.

Following prerequisites shall be ensured to generate the sequence chart upon discovery completion.

Prerequisites:

- Copy 'postProcessUML.exe' and 'plantuml.jar' from A2B release package (.ADI_A2B_Software_RelX.Y.Z\GUI) to the SigmaStudio installation directory.
- Set the PATH environment variable for running 'java.exe' (Most of the times they are set by default)
 - C:\Program Files (x86)\Java\jre<<xx>>\bin

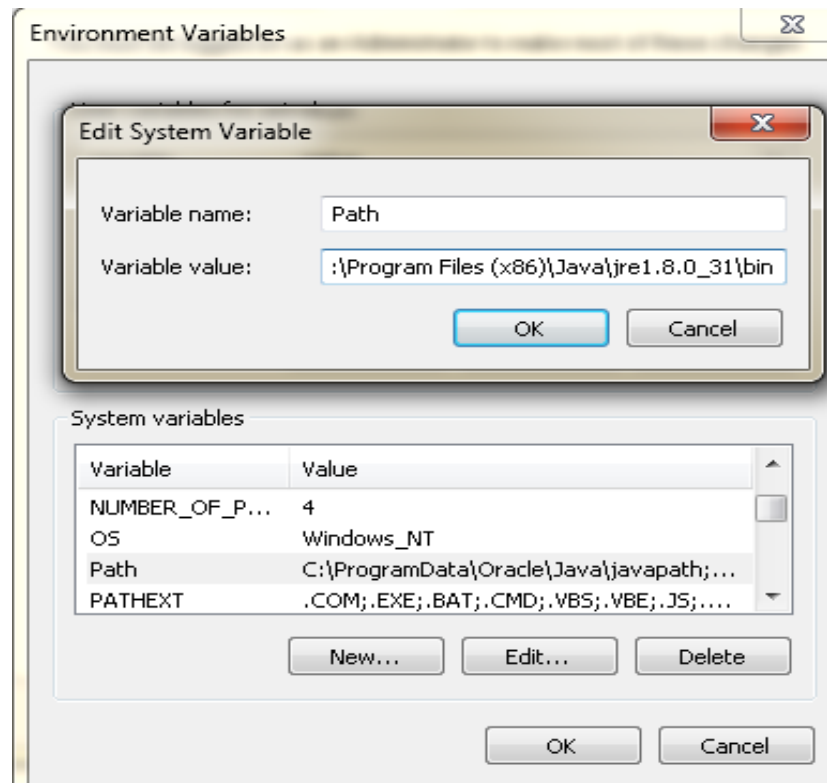


Figure 58: Setting Path in Environment Variables

If the prerequisites are met and sequence chart option is enabled in the Target processor properties, then upon successful discovery of A2B network, the Sequence chart will be generated which can be viewed by clicking on the View button. A Sequence chart generated for a 3-node sample demo schematic is shown in Figure 59.

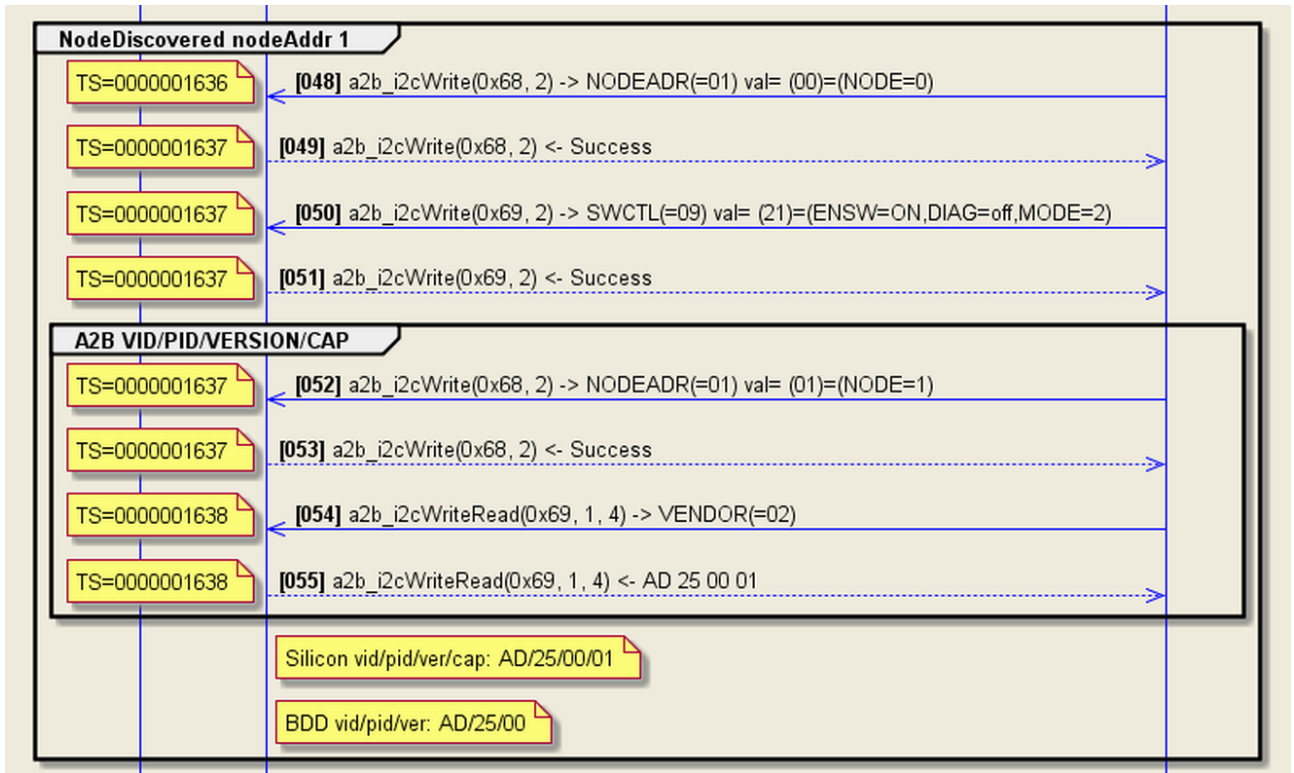


Figure 59: Sequence Chart Sample

If the prerequisites are not met, then a pop up window will display the error message as shown in Figure 60.

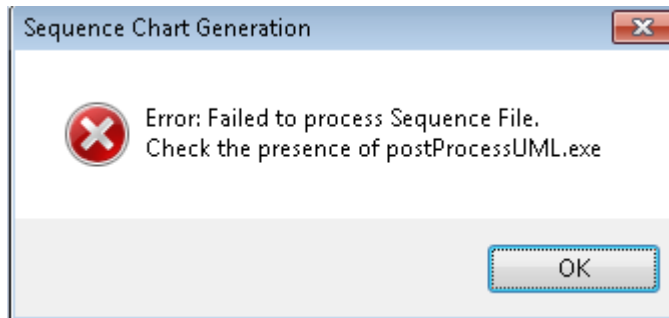


Figure 60: Error Message – Sequence Chart Prerequisites not met

5.5 Schematic Validation Report

SigmaStudio inspects user configuration of schematic and consolidates the validation results into a report. The report can be viewed as context menu option in Target processor as shown in figure

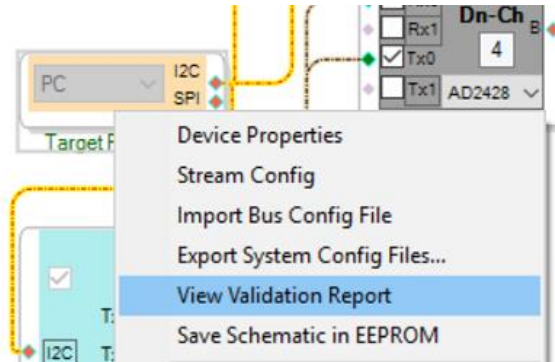


Figure 61: View Validation Report

The validation scheme runs automatically while linking the schematic and report is popped up in case of failure at any node. Report also contains the details on node level A2B bandwidth consumption and consolidated stream information. The report can be saved as word or PDF file for offline analysis.

Validation Report

Choose Node: Slave 0 ☐ View By Name ☒ Show only failed cases ☐ Do not show upon link

1 of 1 100% Find | Next

A2B Schematic Validation Report 8/30/2020 2:41:58 PM

Network Validation Summary: *adi_a2b_3NodeSampleDemoConfig.dspproj* ✖

Node	Product	# Cases	# Violation	BW Usage
Master	AD2428	10	0	32.03%
Slave0	AD2425	10	3	32.03%
Slave1	AD2425	10	3	29.59%
Slave2	AD2428	10	2	34.47%
Slave3	AD2428	10	3	36.91%
Slave4	AD2428	6	2	32.03%

Node Validation: *Slave 0* ✖

ID	Category	Name	Detailed Description	Result
2.1	Stream	<input type="checkbox"/> Rx pin check	At least one Rx/PDM pin should be enabled to source audio stream	FAIL
2.3	Stream	<input type="checkbox"/> Number of Rx channels	Combination of TDM mode and no of pins should be sufficient to receive all the source streams	FAIL
2.4	Stream	<input type="checkbox"/> Number of Tx channels	Combination of TDM mode and no of pins should be sufficient to transmit all the sink streams	FAIL

Bandwidth Usage: *Slave 0* ✔

Figure 62: Schematic Validation Report

The validation report contains the stream definition, stream assignment and stream view configurations done in section 3.2

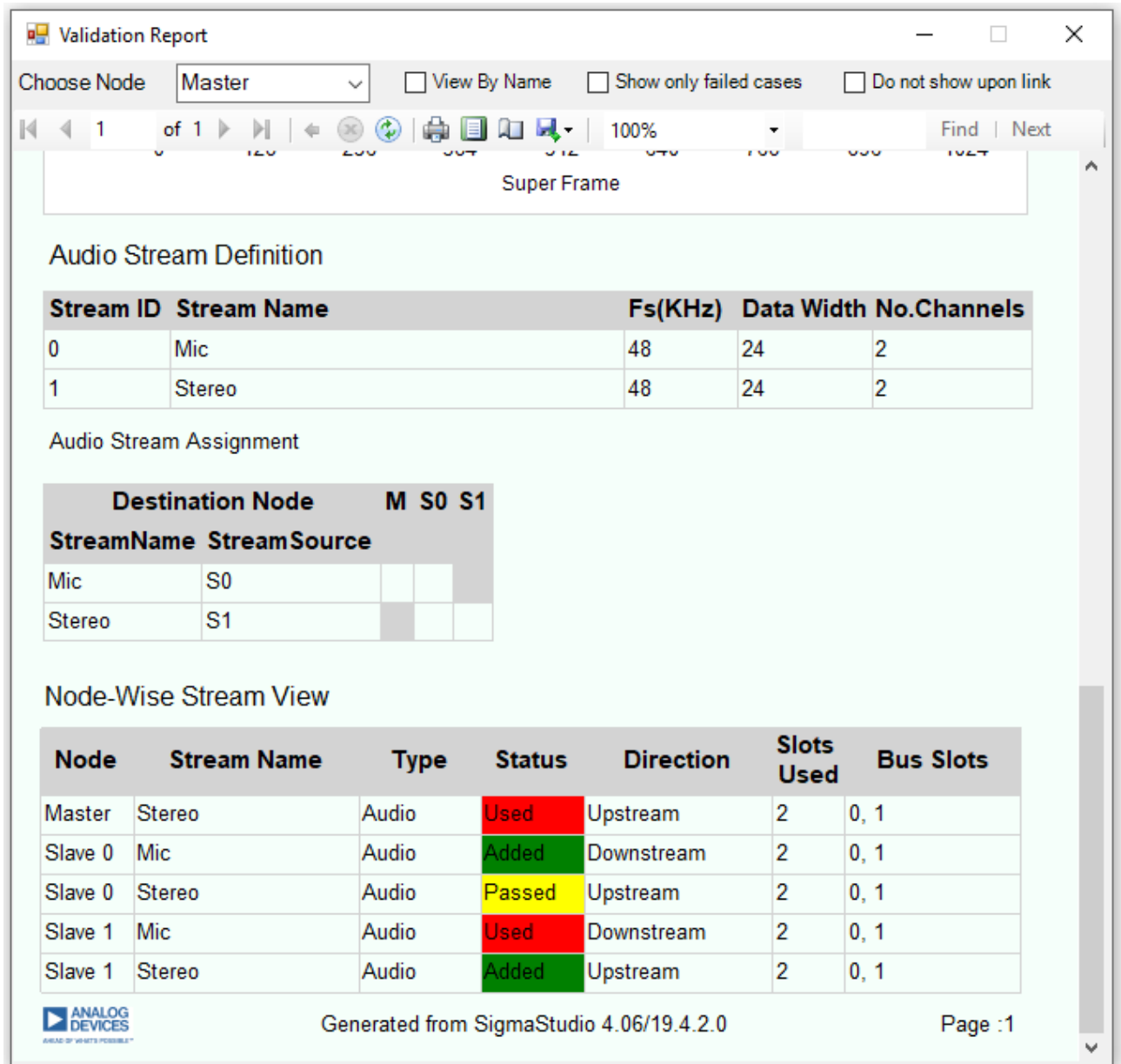


Figure 63: Consolidated Stream View in Schematic Validation Report

6 Tuning SigmaDSP over A2B

SigmaStudio allows user to directly tune/program one or more SigmaDSP peripherals connected to A2B slaves using A2B-USBi and A2B-Aardvark communication channel.

6.1 Tuning / Programming a SigmaDSP

The following steps are involved in tuning or reprogramming a SigmaDSP over A2B

1. Create a valid A2B schematic (or open an existing A2B Schematic project) in SigmaStudio. Make sure that the schematic has at least one SigmaDSP peripheral connected to a slave node.
2. Download A2B schematic and ensure all nodes are discovered.
3. Open the SigmaDSP schematic corresponding to the peripheral connected to the A2B slave which is to be tuned/ reprogrammed. Schematic can be directly opened from the A2B Peripheral Properties window as shown in Figure 15.
4. Switch to the Hardware Configuration Tab of the schematic.
5. Replace the USBi communication channel with an A2B-USBi channel in the HW Tab as shown in Figure 64. Note down the I2C address in the USBi which is currently connected to the SigmaDSP IC before replacing.

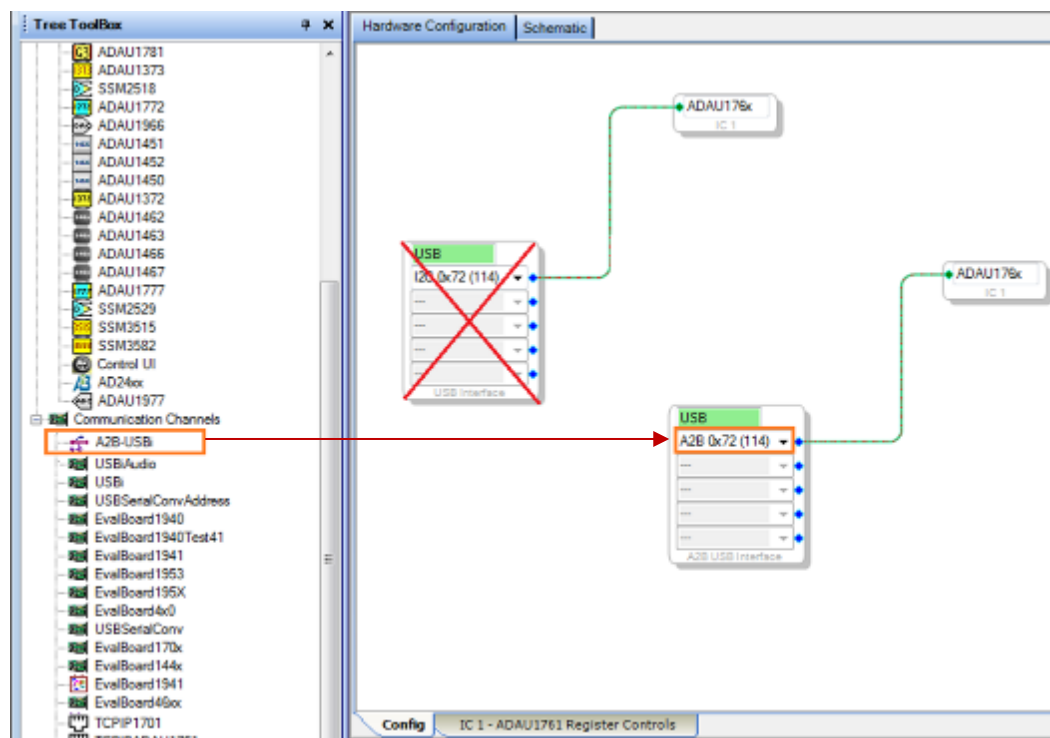


Figure 64: Connecting SigmaDSP to A2B-USBi channel

6. Change the drop-down address to the same Address noted in Step-3 but with a Prefix of A2B instead of I2C (e.g., if the address was I2C 0x72 in the combo box of USB channel, then change the address to A2B 0x72 in the newly connected A2B-USBi channel).
7. Right-click on the 'A2B-USBi' channel and select 'A2B Network Settings'. This will open a window as shown in Figure 65.

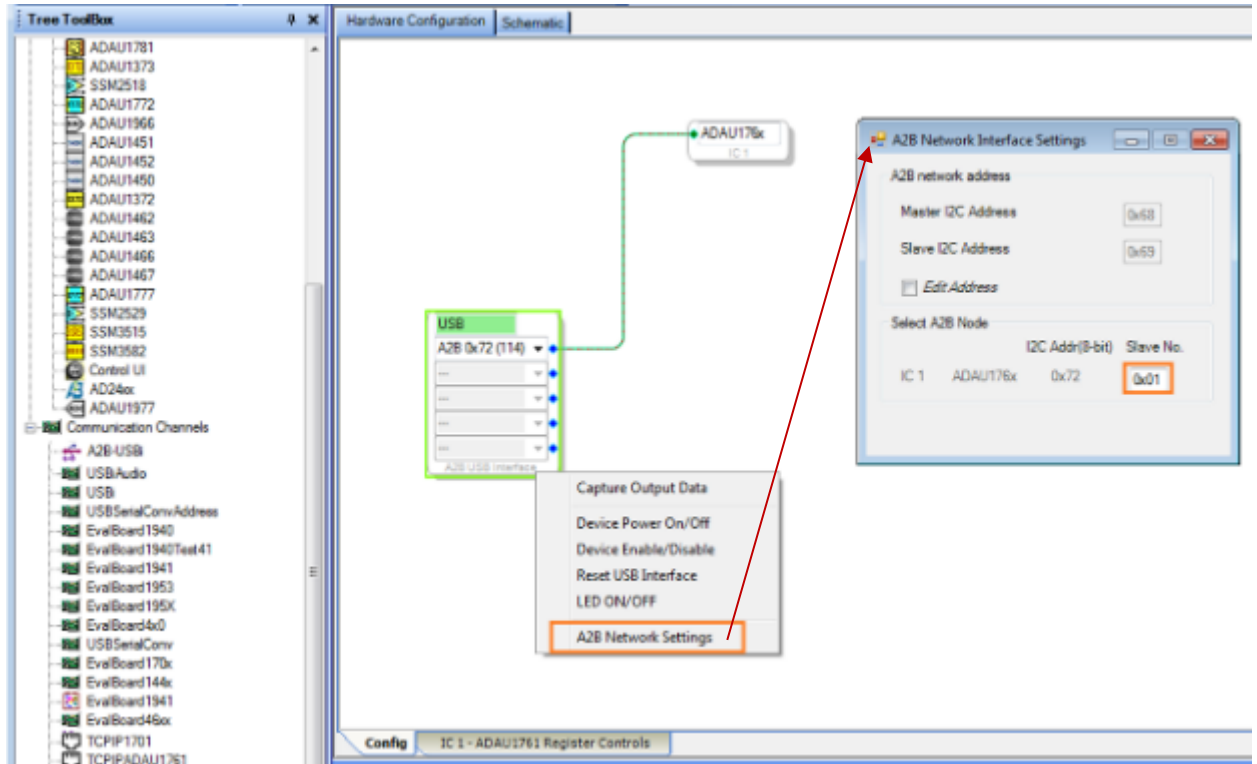


Figure 65: A2B Network Interface Settings

8. Provide the slave node number to which the SigmaDSP/peripheral to be tuned is connected. E.g., if the SigmaDSP is connected to the 1st slave (i.e., node 0) then enter 0x0, if connected to 2nd slave (i.e., node 1) then enter 0x01 and so on.
9. Switch to the Schematic tab and start tuning/reprogramming the peripheral as shown in Figure 66. **Note:** Ensure that A2B schematic is downloaded and nodes are discovered before tuning.

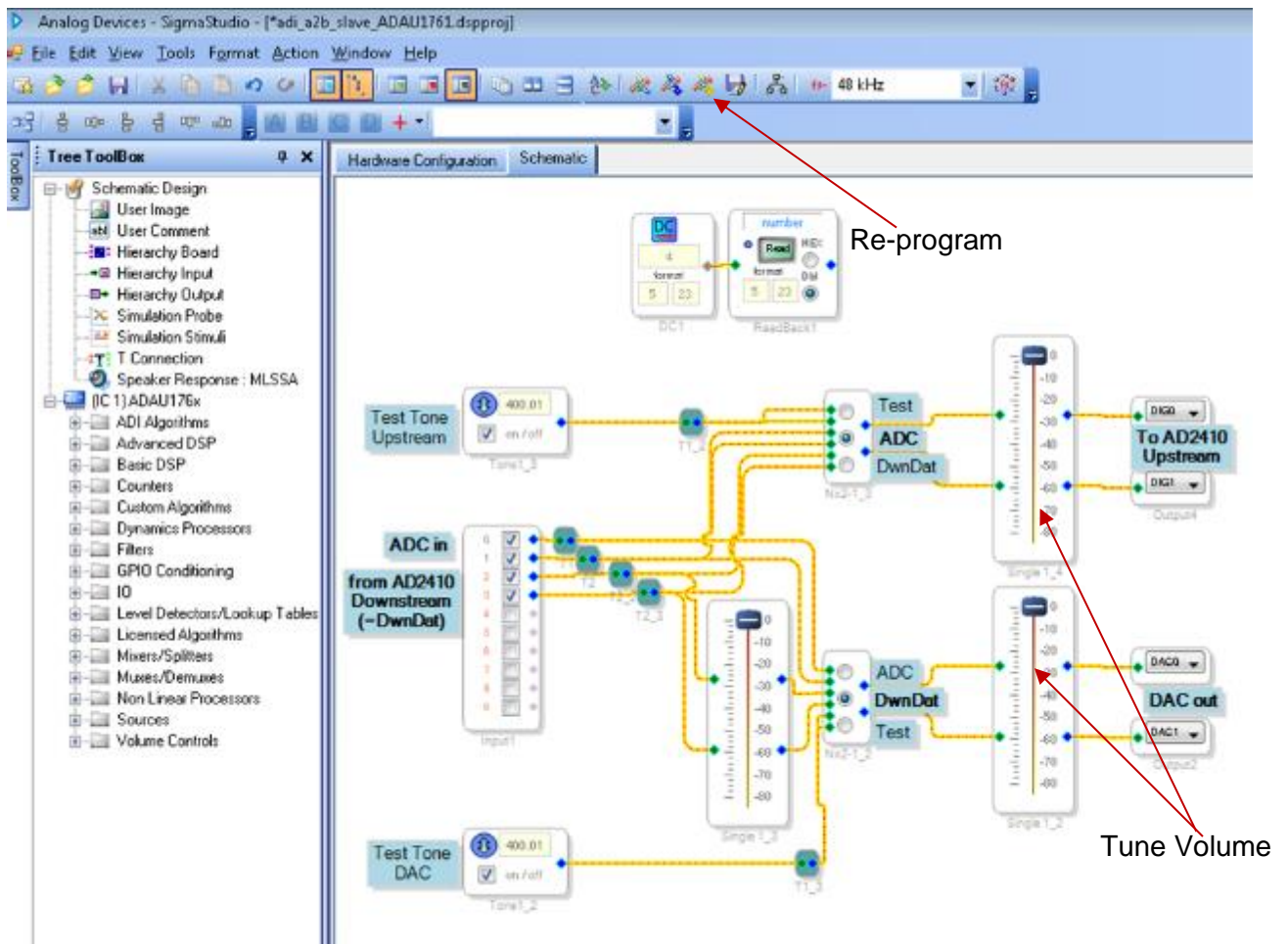


Figure 66: Tuning SigmaDSP over A2B

10. After SigmaDSP schematic is tuned, the corresponding programming file can be updated in A2B schematic by clicking 'Update XML' button as shown in Figure 15.

7 Miscellaneous

7.1 Fix for USBi Download Issue

It is found that on few PCs USBi gets hanged and download is intermittent. The fix for this problem is included in Rel19.0.0 onwards. The fix can be enabled/disabled as shown in Figure 67. Note that the fix enabled by default (from 19.4.0 onwards).

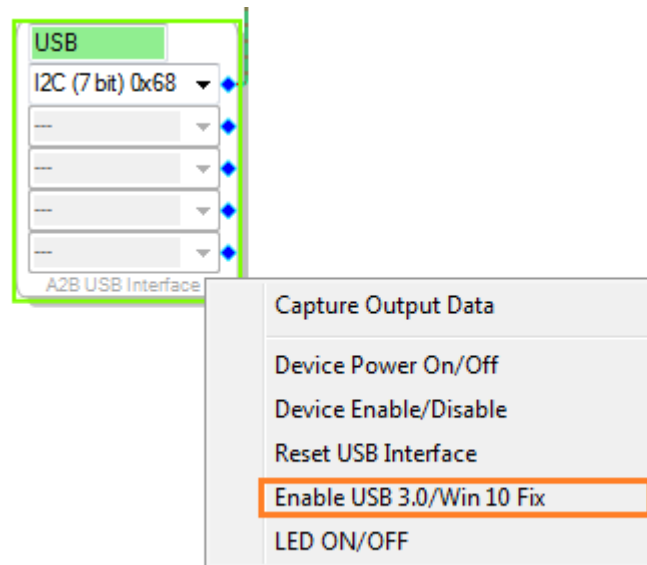


Figure 67: Enabling USBi Fix

Please note that once the fix is enabled, the peripheral (SigmaDSP) configuration files generated from SigmaStudio 4.0 or below won't be compatible. A warning message is displayed while downloading the schematic as shown in Figure 68.

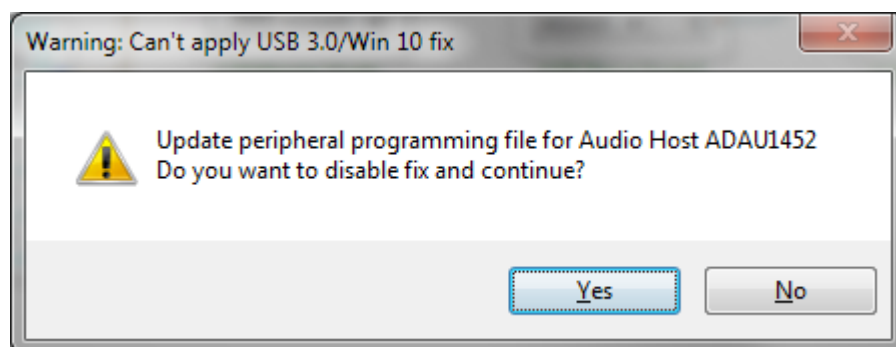


Figure 68: USBi Warning Message

To apply the fix, freshly create/update the XML files for peripherals (SigmaDSP) using SigmaStudio 4.1 or above as explained in section 2.2.3.2.3 and 2.2.3.2.4

Terminology

Table 2: Terminology

Term	Description
A2B	Automotive Audio Bus
A2B node	Refers to AD241x/AD242x.
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.

References

Table 3: References

Reference No.	Description
[1]	AE_09_A2B_QuickStartGuide.pdf
[2]	AD242x Automotive Audio Bus A2B Transceiver Programming Reference
[3]	AE_09_A2B_Stack_UserGuide.pdf
[4]	AE_09_A2B_CommChannel_IntegrationGuide.pdf