A²B® Bridge User Guide

1.3 November 2025

This document is for information and instruction purposes. Flextech AKT LLC reserves the right to make changes in specifications and other information contained in this publication without prior notice, and the reader should, in all cases, consult Flextech AKT to determine whether any changes have been made.

The terms and conditions governing the sale and licensing of Flextech AKT products are set forth in written agreements between Flextech AKT and its customers. No representation or other affirmation of fact contained in this publication shall be deemed to be a warranty or give rise to any liability of Flextech AKT whatsoever.

FLEXTECH AKT MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MATERIAL INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

FLEXTECH AKT SHALL NOT BE LIABLE FOR ANY INCIDENTAL, INDIRECT, SPECIAL, OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING BUT NOT LIMITED TO LOST PROFITS) ARISING OUT OF OR RELATED TO THIS PUBLICATION OR THE INFORMATION CONTAINED IN IT, EVEN IF FLEXTECH AKT HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

U.S. GOVERNMENT LICENSE RIGHTS: The software and documentation were developed entirely at private expense and are commercial computer software and commercial computer software documentation within the meaning of the applicable acquisition regulations. Accordingly, pursuant to FAR 48 CFR 12.212 and DFARS 48 CFR 227.7202, use, duplication and disclosure by or for the U.S. Government or a U.S. Government Subcontractor is Subject solely to the terms and conditions set forth in the license agreement provided with the software, except for provisions which are contrary to applicable mandatory federal laws.

TRADEMARKS: The trademarks, logos and service marks ("Marks") used herein are the property of Flextech AKT LLC or other parties. No one is permitted to use these Marks without the prior written consent of Flextech AKT or the owner of the Mark, as applicable. The use herein of a third- party Mark is not an attempt to indicate Flextech AKT as a source of a product, but is intended to indicate a product from, or associated with, a particular third party.

Flextech AKT LLC
24613 S 220th St.
Queen Creek, AZ 85142

Website: www.flextechakt.com

Contents

Chapter 1. Overview of the A²B Bridge

The A²B Bridge is designed for A²B system evaluation, development testing, and debugging.

The Pocket A²B Bridge is a USB powered, single bus solution targeted towards engineering development and test.

The Industrial A2B Bridge is a multi-port A²B bus tester designed for integrated high volume A²B production and validation testing.

Both products share the same fundamental features and functionality enabling a smooth transition from development to production.

Pocket A²B Bridge Capabilities

The **Pocket A²B Bridge** can be used as a bench tool for developing and testing A²B bus systems:

- Support up to 32 input / 32 output **audio channels** over USB
- Stereo line level analog and optical S/PDIF on ***plus***Audio models
- **Emulate** A²B Main node or Sub node
- Boot commands and optional Lua scripting for **fully autonomous** scripting and start up
- **RESTful command API over USB** for custom test integration
- **Built in** signal generators and audio VU meters
- Mailbox protocol support
- **Guaranteed** deterministic low latency A²B audio
- **AKT Automation** powered by Lua

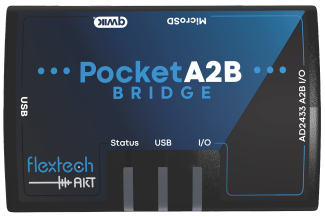
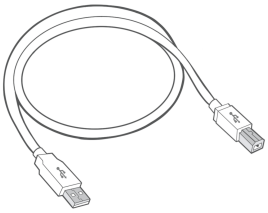
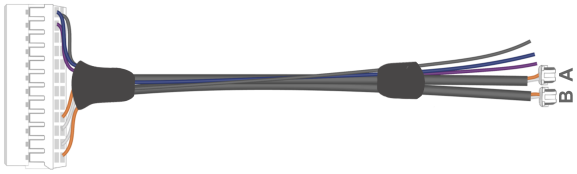
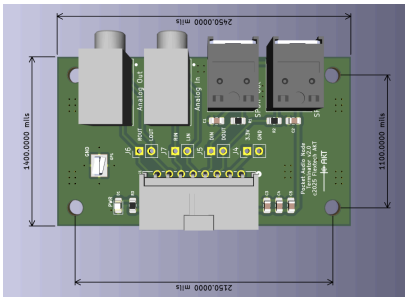
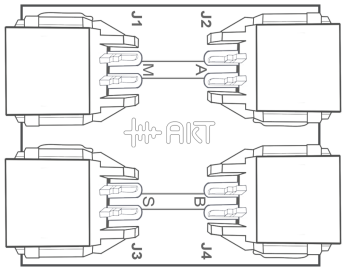
Industrial A²B Bridge Capabilities

The **Industrial A²B Bridge** is designed for high volume integrated test systems. It has the following additional features:

- Independent simultaneous control of **2, 4, or 6 A²B buses**
- **Ethernet** RESTful command API, TFTP and FTP file transfers
- Support up to 32 output audio channels over **Ethernet RTP**
- Support up to 8 output audio channels over **Ethernet VBAN**

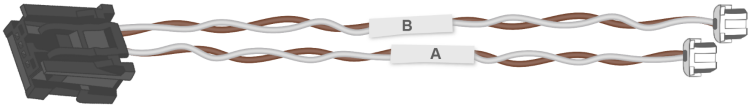
Pocket A²B Bridge Delivery Parts List

The following parts are included and shipped with the Pocket A²B Bridge

1 Pocket A²B Bridge device	
1 USB cable, for connecting the Pocket Bridge to your host PC	
1 Connector Harness with cables for connecting the Pocket Bridge to the A²B network and GPIO pins.	
1 AKT ***plus***Audio I/O Board (***plus***Audio Only) **NOTE:** An additional **audio** I/O header is included in the Connector Harness on ***plus***Audio devices to connect to the AKT ***plus***Audio I/O Board.	
1 Crossover Board	
1 SanDisk Industrial microSD Card	

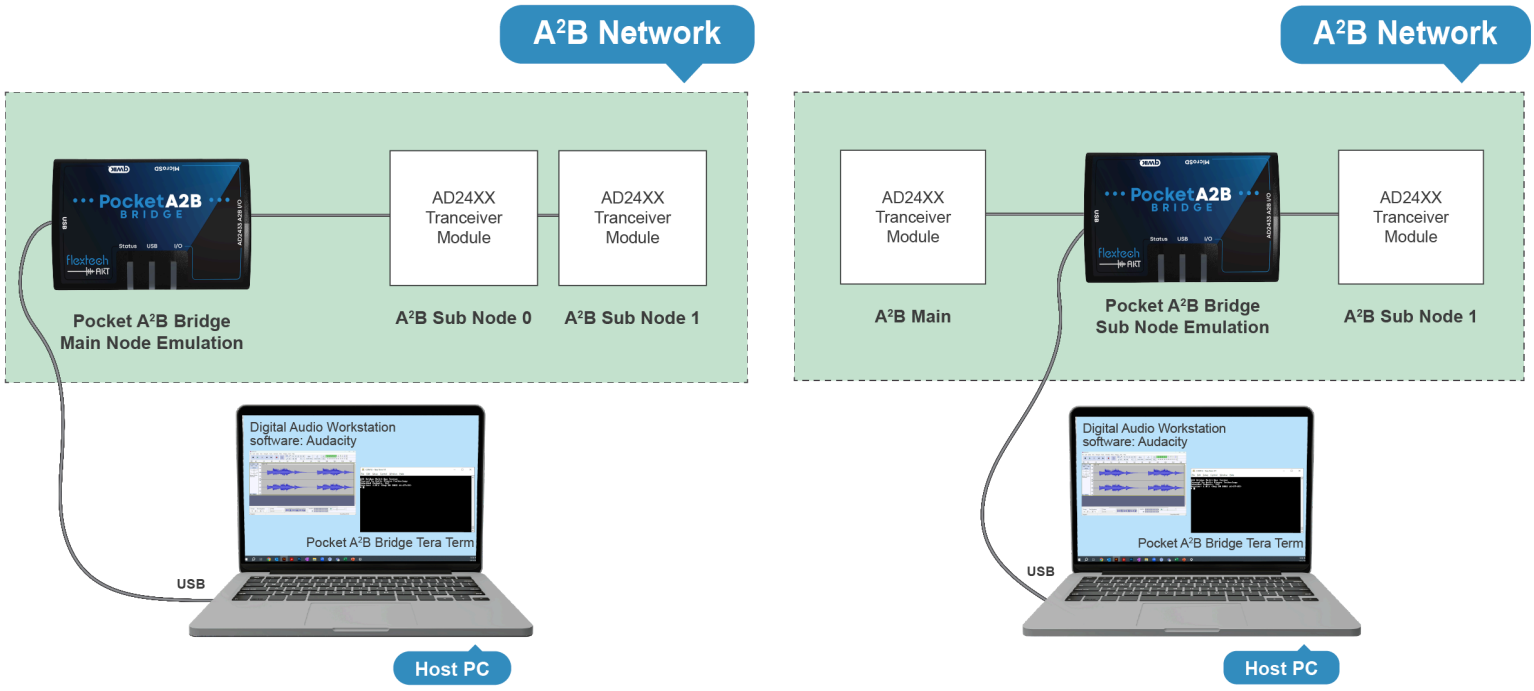
Industrial A²B Bridge Delivery Parts List

The following parts are included and shipped with the Industrial A²B Bridge

1 Industrial Bridge device	
1 Molex Mini50 Harness with cross-over cable for connecting the Industrial Bridge to an A²B network.	
1 SanDisk Industrial Full Size SD Card	
1 12V 3A Power Supply	

A²B Network System Components

The following diagrams show how the A²B Bridge can be integrated into your A²B network as an A²B Main node or Sub node. Place the A²B Bridge at the beginning of the A²B bus when configured as a Main node. Place the A²B Bridge between node segments, or at the end of the bus when configured as a Sub node.



Host PC	System that controls the A²B Bridge via TeraTerm or remote control API. In this setup, the Bridge device is connected to Host PC over USB.
USB	Connection used to configure and control the device. It can also be used to view and capture data from the A²B bridge.
A²B Bridge	Hardware device that lets you emulate an A²B node
A²B network	A²B networks consist of a single Main node and multiple Sub nodes in a daisy-chained configuration.
Main / Sub node	The A²B Bridge acting as a Main or Sub node emulator

Chapter 2. Interfaces

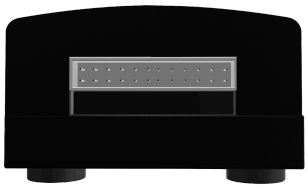
This chapter introduces the ports and LEDs found on the devices.

Pocket A²B Bridge Ports

Front View

1 USB 2.0 high speed type B interface. Main connection to PC.

Note: The Pocket Bridge is powered over the USB connection.



Back View

24 Pin Multi I/O Connector. Includes A-side and B-side A²B connections; two independent, bi-directional general purpose 3.3V logic-level GPIO lines with interrupt capability; access to the A²B transceiver I²C and SPI data pins; analog line level and S/PDIF audio on ***plus***Audio models.

Side View

SD Card Slot for file storage and **AKT Automation**.

[Sparkfun Qwiic Expansion Connector]

Bottom View

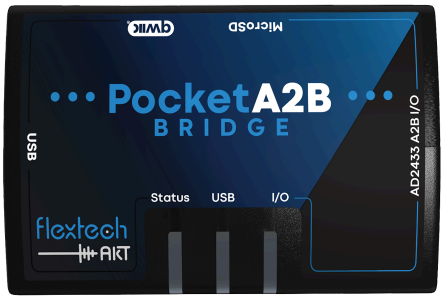
Safe-Boot Recovery button

HW ID and **Serial Number**

Note: For further details on cables and connectors, see Chapter 4. Connectors & Cables

Pocket A²B Bridge LEDs

LEDs on the top indicate the status of the device when it's powered on.



Status LED

LED State	Color	Description
Fast Flash	Green	System OK
	Red	Invalid / missing feature key file. See Troubleshooting.
Slow Blink	Yellow	Bootloader mode active

USB LED

LED State	Color	Description
Blink	Blue	Activity on the USB bus
Steady	Blue	Active USB Audio stream

I/O LED (Input / Output)

LED State	Color	Description
Blink	Blue	A²B active
Steady	Red	A²B error

Industrial A²B Bridge Ports

Ethernet

Ethernet can be used for remote command and control via the HTTP RESTful API. It can also be used to transfer audio and network configuration files to and from the internal filesystem or SD card via TFTP or FTP.

The Ethernet IP address can be configured statically or automatically over DHCP. The mDNS protocol is also supported which requires no configuration and is used when connecting directly to a Windows or Linux PC.

USB

The USB port is used for UAC2 USB audio, a serial command console, and software updates.

USB Audio

By default the device is configured for 20 OUT and 20 IN channels, from the PC to the A²B Bridge. All channels are 16-bit. Linux, Mac, and Windows 10/11 are all compatible with this standard. The IN channels, OUT channels and bit depth can be reconfigured if necessary.

Serial console

A USB serial console is available for command-line configuration, debug, and troubleshooting.

Software Updates

When in bootloader mode, this port is also used for software updates.

SD Card

A standard SD card slot is included for file storage on the A²B Bridge. A²B network configurations exported from Sigma Studio should be stored on the SD card. WAV files and other data files can also be stored on the SD card. The SD card can be freely removed and inserted whenever the A²B Bridge is not actively accessing files on it.

Note: *Always use a minimum Class 10 or UHS Class 1 SD card if the card will be used for WAV file audio. Use a freshly formatted SD card when recording high bit-rate WAV files to reduce the risk of audio drops.*

A²B

The Industrial A²B Bridge supports up to six A²B Buses. The 2-port and 4-port models support Main and Sub Node mode and can utilize both the A-side and B-side A²B connections. The 6-port model only supports Main node mode and utilizes the B-side A²B connection only.

The A²B ports are enumerated starting at zero. A²B port names are A2B0 through A2B5.

> **Note:** For further details on A²B cables and connectors, see Chapter 4. Connectors & Cables

Power

Use the power supply delivered with the unit or any regulated 12V power power supply that can source at least 3A of current.

Industrial A²B Bridge LEDs

Front Panel LEDs

Each A²B port has a dedicated set of Green and Red LEDs.

Green LED	Description
Blink	A²B Bus active and OK
Steady	A²B Bus idle

Red LED	Description
Steady	A²B Bus error
Off	No errors

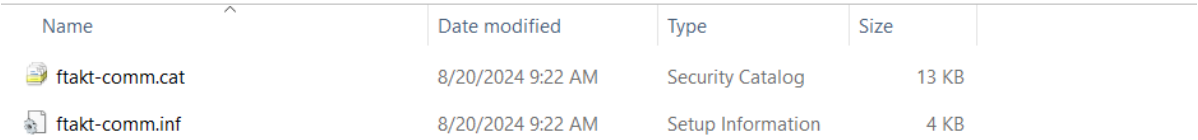
Red / Green LED	Description
Blink	Invalid / missing feature key file. See Troubleshooting.

Initial Setup

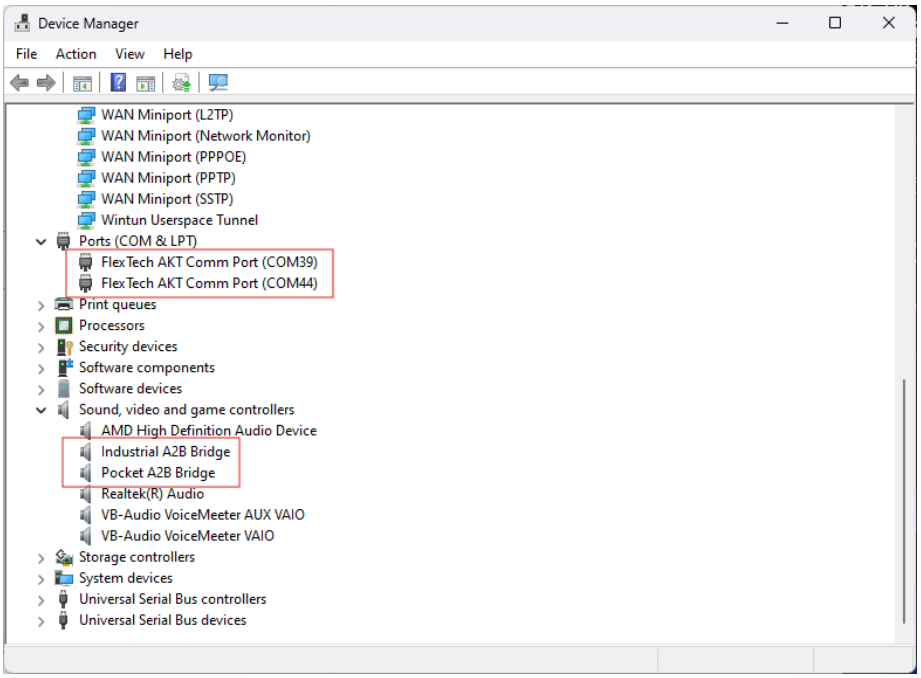
Installing the Flextech AKT USB device driver

To install the USB serial device driver, perform the following steps:

1. Download the ftakt-comm driver from www.flextechakt.com



2. Unzip the supplied zip file
3. Right Click on the *ftakt-comm.inf setup file* and select *install.*
4. After installation you will see the FlexTech AKT Comm Port and sound card listed on Windows Device Manager under Ports (Com & LPT).



Download and Install a Terminal Emulator Utility (Tera Term)

To access the command line interface, you must install a terminal emulator such as Tera Term, Putty, GTKTerm, or screen.

Tera Term under Windows is recommended because it has built in support for X/YMODEM that can be used to transfer files over USB.

Click Here to Access the latest version of **Tera Term**

Hookup

Use the supplied USB Type B cable to attach the A^{2}B Bridge to the PC.

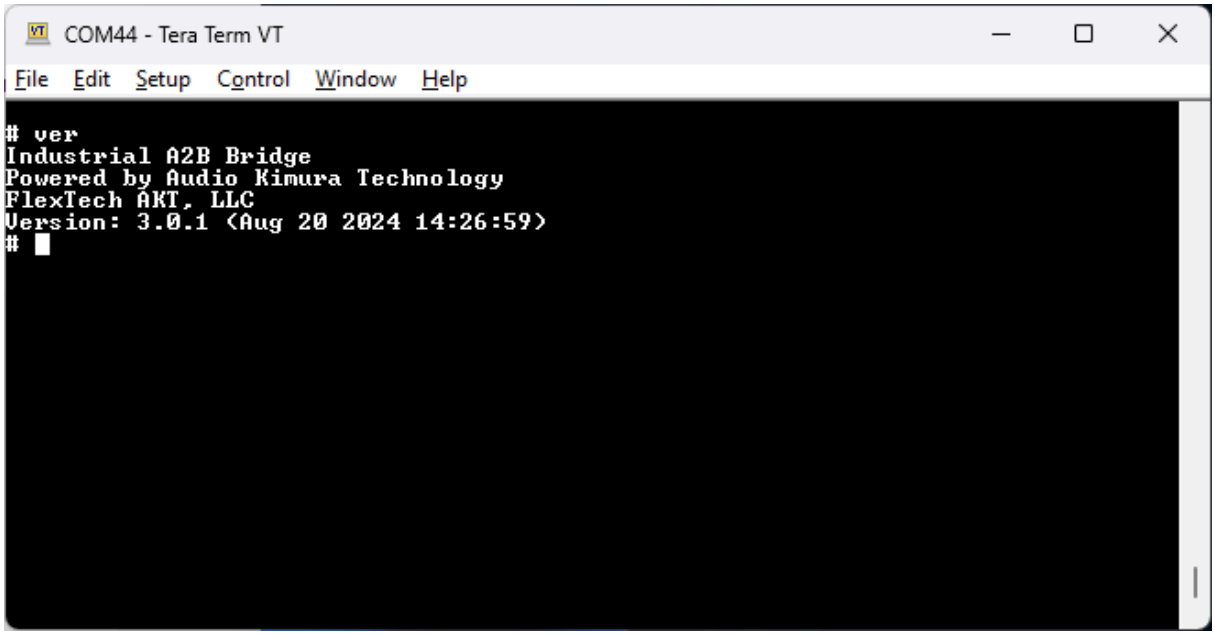
When operating as an A^{2}B Main node, attach the “B” side to the first downstream Sub node. When operating as an A^{2}B Sub node, attach the “A” side to the next upstream node and the “B” side to the next downstream node. Leave the “B” side disconnected if the A^{2}B Bridge is the last node on the network. Review Chapter 4. Connectors & Cables for more information.

For the Industrial Bridge optionally connect the Ethernet port to a local DHCP enabled network or directly to a PC. Attach the supplied +12V power adapter to the Industrial Bridge and switch on.

Start a terminal emulator on the "FlexTech AKT Comm Port" USB UART. Since this is a virtualized USB serial port, the terminal settings are not critical, but suggested to be 115200 Baud, No parity, 8 data bits, 1 stop bit (115200,N,8,1).

Once connected, press `<ENTER>` a few of times until a '`#`' prompt appears. Type '`ver`' and press `<ENTER>`.

A welcome/version message similar to the following will be displayed:



Using the Command Line

The A^{2}B Bridge has a number of useful commands for setup, debug, automation, and maintenance activities.

Type `'help'` at the command prompt to see the full list of available commands. Additional help specific to each command can be accessed by typing `'help <command>'`.

There are many commands available on the A^{2}B Bridge so the contents of the `'help'` command varies depending on the Command Level.

- **Command Level 0** - Available at startup and only shows the most common commands.
- **Command Levels 1 and 2** - Display additional system maintenance and troubleshooting commands.

Note: All commands can be run at all levels to facilitate scripting. Only the help is filtered by the command level.

Use the `'shell'` command to change command levels. For example to switch to command level 2, enter the following command:

```
# shell level 2
```

General Commands

Command	Purpose
`edit`	Simple text editor that can be used to modify small text files directly on the A^{2}B Bridge.
`hwid`	Show device HW ID
`reset`	Resets system components. A “soft” returns the A^{2}B Bridge to its power on reset state.
`resize`	Resize or Sync the terminal window size. Run this command after the Tera Term command window is resized to synchronize the new terminal size or specify a size to resize the terminal window.
`ver`	Show version information
`help`	Shows specific help for commands

For additional help on any command use the built-in help command:

```
# help help
help - shell help
Usage: help [<command>]
  [<command>] - the command to get help on.
Without arguments it shows a summary of all the shell commands.
```

File Management

It is necessary to install files onto the A^{2}B Bridge for a variety of reasons including network discovery, audio playback and recording. The most direct method is to simply copy files to or from a PC on the SD card. The SD card can be safely removed and reinstalled when not in use.

NOTE: Always remember to eject the card from the PC prior to removing it.

In addition to the SD card, the A^{2}B Bridge has a small internal "Flash" file system. This file system is meant for system files that must be maintained even when the SD card is ejected and for files used early at startup. Such files include:

File	Purpose
`XXXX-XXXX-XXXX-XXXX.key`	This file is the product activation key and required for proper operation. ***Never modify or erase this file. It is recommended to make a backup of this file and store it somewhere safe.***
`shell.cmd`	Commands in this file are automatically run at system startup.
`cfg.ini`	Accessed early in the boot process to override system defaults such as USB audio channels or Ethernet settings.

Wherever file names are mentioned in this document, prefix the actual file name with `'sf:'` to access files on the internal Flash file system or `'sd:'` to access files on the SD card. File names with no prefix will default to the SD card.

NOTE: The maximum file name length on the internal Flash filesystem is 30 characters

File Commands

The following commands are available on the A²B Bridge for manipulating files. Type `'help <cmd>'` on the command line for detailed usage instructions.

Command	Purpose
`cat`	Show the contents of a text file. Do not use this command with binary files. Use the `'dump'` command to display binary files.
`cp / copy`	Copy a file
`df`	Show the drive full status
`drive`	Show or set the default drive
`dump`	Show the contents of a file in hex
`format`	Format a drive. ***WARNING:*** *Formatting the internal flash file system will erase the activation key rendering the A²B Bridge inoperable. It should never be necessary to format the internal sf: filesystem.*
`fsck`	Check the integrity of a drive
`edit`	Edit a text file
`ls / dir`	Show a directory listing of a drive
`recv`	Receive a file via XMODEM. If no file name is given, receive multiple files via YMODEM.
`send`	Sends one or more files via YMODEM
`rm / del`	Delete a file
`run`	Run a command script
`tail`	Show the last <n> lines of a text file

File Transfers

The A²B Bridge supports a variety of methods to transfer or create files for system setup.

SDCARD

The most direct method is to simply copy files to or from a PC using the SD card. The SD card can be freely removed and reinstalled in the A²B Bridge when not in use.

NOTE: Always remember to eject the card from the PC prior to removing it.

Edit Command

Simple text files can be created or modified directly from the command line using the `'edit'` command. Press `<CTRL-S>` to save the file. Press `<CTRL-Q>` to quit editing.

X/YMODEM

Files can be downloaded to the A²B Bridge using the XMODEM or YMODEM protocols via the `'recv'` command. XMODEM is used to transfer a single file and used when a file name is provided. YMODEM is used to transfer multiple files when no file name is specified.

TeraTerm supports both XMODEM and YMODEM file transfer protocols. After issuing the `'recv'` command, select `File -> Transfer -> [X][Y]MODEM -> Send` to initiate a transfer.

Files can be uploaded from the A²B Bridge using the YMODEM protocol via the `'send'` command. Multiple files can be sent in a single transfer.

Select the 1k file transfer option for faster transfers.

Ethernet TFTP and FTP

The Industrial A²B Bridge supports Ethernet file transfers using the TFTP or FTP protocols.

Use the `'eth'` command to determine the IP address or Hostname of the A²B Bridge.

To upload a file to the Industrial A²B Bridge using the Windows 10/11 native TFTP client, open a DOS command window and execute the following command:

```
tftp -i <IP/Hostname> PUT <src_file> <dst_file>
```

To download a file from the A²B Bridge:

```
tftp -i <IP/Hostname> GET <src_file> <dst_file>
```

<IP/Hostname> is the IP address or Hostname of the A²B Bridge. The 'eth' command can be used to show the IP address or Hostname. <src_file> is the source file and <dst_file> is the destination file.

Example

Transfer `243x-wbz.xml` from the PC to the internal Flash file system of the A²B Bridge

```
tftp -i <IP/Hostname> PUT "243x-wbz.xml" "sf:243x-wbz.xml"
```

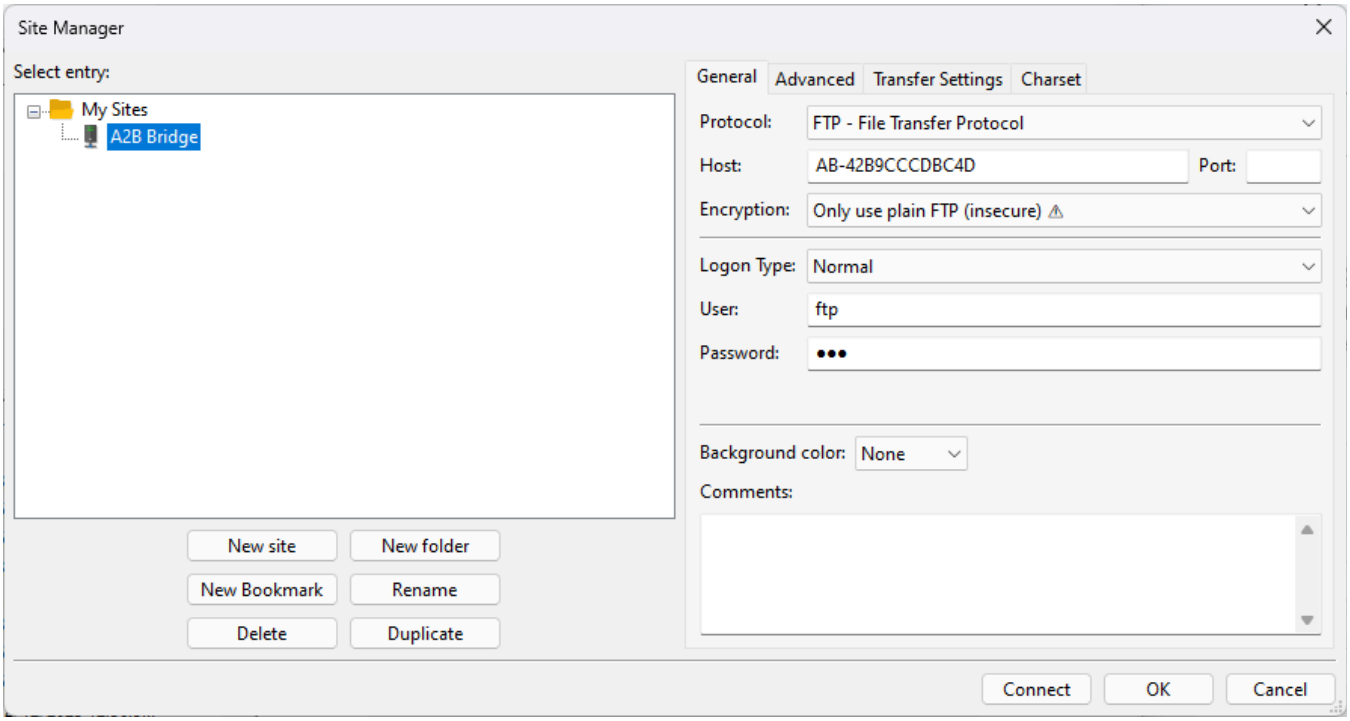
Transfer from the Flash file system to the PC

```
tftp -i <IP/Hostname> GET "sf:243x-wbz.xml" "243x-wbz.xml"
```

NOTE: On Windows, the TFTP client must be explicitly enabled. Go to `"Control Panel" -> "Programs and Features -> "Turn windows features on and off"` and check the box for TFTP Client.

NOTE: Always use the `-i` option to enable binary mode file transfers.

Standard FTP can also be used to transfer files between the PC and A²B Bridge. Any username and password are accepted. FileZilla is an easy to use FTP client that is compatible with the A²B Bridge. Below is an example FileZilla Site Manager configuration for the A²B Bridge:



Changing Default System Settings

A number of default system settings can be modified through the `sf:cfg.ini` configuration file.

To modify the default values, create a text file called `cfg.ini` containing content from the sections below.

Copy this file onto the SD card, insert the SD card into the A²B Bridge, then copy the file from the SD card to the internal flash filesystem with the following command:

```
cp cfg.ini sf:cfg.ini
```

The `sf:cfg.ini` file can also be created or modified directly on the A²B Bridge using the `'edit'` command.

System Sample Rate Re-Configuration

The default sample rate of the A²B Bridge is 48 kHz. To modify the system sample rate in the `cfg.ini` file, add a `[system]` section to it. The example below sets the system sample rate to 44.1 kHz.

```
[system]
sample-rate = 44100
```

For Windows 10/11, you must uninstall the existing A²B Bridge audio driver in the Device Manager after changing any of the USB audio settings. See the USB Sound Card Re-Configuration section for details.

NOTE: *Be very careful when configuring this file. Incorrect settings can result in a boot failure that may require a Safe-Boot Recovery of the A²B Bridge.*

USB Sound Card Re-Configuration

By default the A²B Bridge is configured as a 20 Digital OUT x 20 Digital IN x 16 bit USB sound card.

To modify the default values, create a text file called `cfg.ini` with the contents below. If you already have a `cfg.ini` file, add a `[usb-audio]` section to it.

```
[usb-audio]
out-channels = <2-32 channels>
in-channels = <2-32 channels>
word-size-bits = <16 or 32>
```

Set the values as required for the application. The example below configures 32 OUT channels (from the PC to the A²B Bridge), 2 IN channels (from the A²B Bridge to the PC) with a bit-depth of 16-bits:

```
[usb-audio]
out-channels = 32
in-channels = 2
word-size-bits = 16
```

The USB product string can be overridden by setting a `'product-string'` in the `[usb-audio]` section.

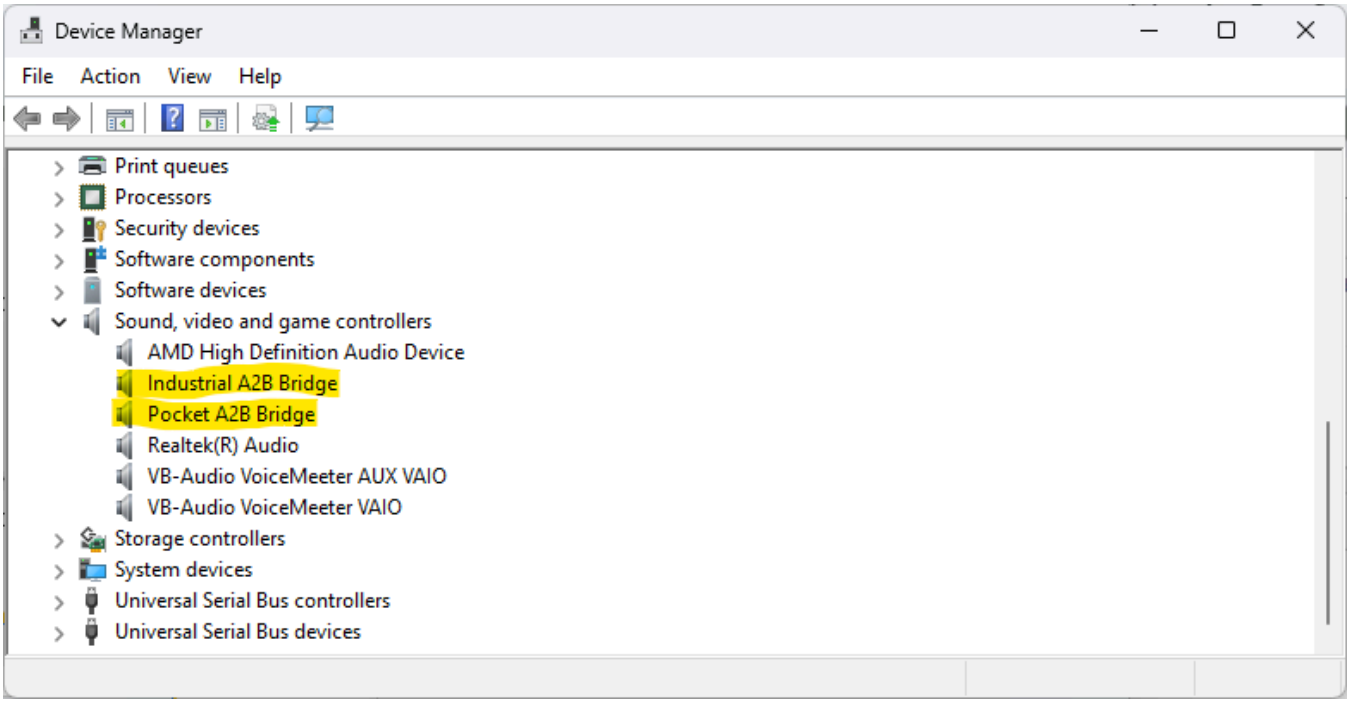
Copy this file onto the SD card, insert the SD card into the A²B Bridge, then copy the file from the SD card to the internal flash filesystem with the following command:

```
cp cfg.ini sf:cfg.ini
```

The `sf:cfg.ini` file can also be created or modified directly on the A²B Bridge using the `'edit'` command.

For Windows 10/11, you must uninstall the existing A²B Bridge audio driver in the Device Manager after changing any of the USB audio settings.

With the A²B Bridge powered and connected, right click on the "A²B Bridge" sound card and select uninstall:



Reset the A²B Bridge and Windows will apply the new settings.

NOTE: *Do not set the number of channels or bit-depth higher than required by the application. Unused channels still consume CPU cycles on the A²B Bridge and USB bandwidth on the PC.*

NOTE: *Do not set the number of channels in either direction to zero. A minimum of one channel must be defined in both the IN and OUT directions.*

Ethernet Configuration

The Industrial A^{2}B Bridge supports Ethernet for command, control, and audio.

DHCP

By default the A^{2}B Bridge will attempt to configure the Ethernet interface via DHCP. Use the `'eth'` command to retrieve the DHCP assigned Ethernet IP address when on a DHCP configured network.

Static IP

To configure a static IP address, create a text file called `cfg.ini` with the contents below. If you already have a `cfg.ini file`, add a `[network]` section to it.

```
[network]
ip-addr = <IP>
gateway-addr = <GW>
netmask = <NM>
static-ip = 1
```

Replace IP, GW, and NM with the desired IP address, gateway address, and network netmask. An example is shown below:

```
[network]
ip-addr = 192.168.1.2
gateway-addr = 192.168.1.1
netmask = 255.255.255.0
static-ip = 1
```

Copy this file onto the SD card, insert the SD card into the A^{2}B Bridge, then copy the file from the SD card to the internal filesystem with the following command:

```
cp cfg.ini sf:cfg.ini
```

Reset the A^{2}B Bridge to apply the new settings. Use the `'eth'` command to confirm the new settings have been applied.

NOTE: *Be very careful when configuring this file. Incorrect settings can result in a boot failure that may require a safe-boot recovery of the A^{2}B Bridge.*

mDNS

The A^{2}B Bridge supports the mDNS protocol which requires no configuration and is especially helpful when directly connecting the A^{2}B Bridge to a PC.

Once connected, use the `'eth'` command to retrieve the Hostname and use the Hostname when connecting to the device. The Hostname is unique to each A^{2}B Bridge and does not change.

Emulating a Main Node

Creating an A²B Network Configuration

The A²B Bridge supports Analog Devices' Sigma Studio XML A²B Command List export files.

Refer to the FlexTech AKT Sigma Studio Quick Start Guide for the basics of configuring and exporting an A²B network.

More detailed documentation from ADI can be found in the Docs directory of the A²B plugin installation. **AE_09_A2B_SigmaStudio_UserGuide.pdf** details the A²B specific instructions for Sigma Studio.

Disregard section 4.1 related to BCF files as the A²B Bridge only supports the XML Command List export format.

Copying A²B Network Configuration files

The Sigma Studio XML Command List output can be copied to any drive on the A²B Bridge using any of the file transfer mechanisms detailed in File Transfers. There is no limit to the number of XML files that can be stored on the A²B Bridge.

Setting the A²B mode

The A²B Bridge defaults all buses as Main nodes at power up. To set a bus to Main node mode, use the `'mode'` command as follows.

```
# mode a2b0 main
A2B0 Mode: main
```

Discovering an A²B Network

Use the `'discover'` command to discover an A²B Network. The following example demonstrates how to discover an A²B network on A2B0 using a Sigma Studio Command List XML export.

```
# discover a2b0 243x-spi-test.xml
Discover
  Bus: A2B0
  Type: ADI XML
  Network: 243x-spi-test.xml
Detail: Discovered 1 node(s)
Result: OK
```

In addition to Sigma Studio XML Command List exports, the A²B Bridge also supports legacy Mentor Graphics Analyzer BDD exports.

Interfacing with A²B Sub Nodes

In addition to audio, A²B supports remote A²B Sub node transceiver register access, remote peripheral I²C, remote peripheral SPI, and basic GPIO control.

Most common Sub node interactions are supported directly on the command line. Below are commands used to interface with A²B Sub nodes.

Command	Purpose
`comm`	Configure MBOX communication protocols. Most protocols are proprietary. Contact Flextech AKT directly for more information about this command.
`mboxlog`	Special log used to debug MBOX communication protocols
`i2c`	Perform Main or Sub node register and peripheral reads and writes
`gpio`	Configures local A²B GPIO pins (GPIO0-GPIO7)
`spi_reg`	Performs A²B SPI register transfers (Pocket A²B Bridge and SPI enabled Industrial A²B Bridge only)
`spi_tun`	Performs A²B SPI tunnel transfers (Pocket A²B Bridge and SPI enabled Industrial A²B Bridge only)
`vmtr`	Read VMTR (voltage meter) values. This feature only applies to AD243x transceivers.

For more sophisticated or interactive Sub node transactions use the remote control API over USB / Ethernet or the AKT Autonomous Automation API. Refer to the A²B Bridge API Reference Manual for more information.

A2B 1.0 Examples

The following example reads the Vendor, Product, and Version registers via I²C on A2B0 Sub node 0

```
# i2c a2b0 0 "0x02" 3
0000: ad 33 12
Result: OK
```

The following example reads 16 bytes from a peripheral EEPROM via I²C on A2B0 Sub node 0:

```
# i2c a2b0 0 "0x00,0x00" 16 1 0x50
0000: ab ad 0b 14 00 00 ff ff ff ff ff ff ff ff ff ff
Result: OK
```

The following example reads the Vendor, Product, and Version registers via SPI on the Main node of A2B0

```
# spi_reg A2B0 -1 2 "" 3
0000: ad 33 20
Result: OK
```

The following example reads the Vendor, Product, and Version registers via SPI on A2B0 Sub node 0

```
# spi_reg A2B0 0 2 "" 3
0000: ad 33 12
Result: OK
```

The following example performs a 8 byte Full Duplex synchronous SPI transfer on A2B0 Sub node 0

```
# spi_tun a2b0 0 0x09 "0,1,2,3,4,5,6,7" 8 ADR1 1
0000: 00 01 02 03 04 05 06 07
Result: OK
```

The following example reads the VMTR values on A2B0 Sub node 0

```
# vmtr a2b0 0
 0: 8.437500
 1: 8.250000
 2: 3.328125
 3: 3.328125
 4: 1.906250
 5: 0.014100
 6: 0.000000
```

A2B 2.0 Examples

The following example reads the Vendor, Product, and Version registers via I^2C on the main node

```
# i2c a2b0 -1 "0x00,0x02" 6
0000: 00 ad 24 57 00 10
Result: OK
```

The following example reads the Vendor, Product, and Version registers via SPI on A2B3 main node

```
# spi_reg a2b3 -1 2 "" 6
0000: 00 ad 24 57 00 10
Result: OK
```

NOTE: While the Analog Devices A^2B 2.0 Technical Reference Manual begins A^2B 2.0 node numbering at zero with the main node, the ***A^2B Bridge always addresses the Main node as -1 and the first Sub node as 0*** to maintain API consistency between A^2B 1.0 and A^2B 2.0 networks.

Emulating a Sub Node

Entering Sub Node Mode

The A^{2}B Bridge defaults all buses to a Main node at power up. To set a bus to Sub node mode, use the `'mode'` command.

```
# mode a2b0 sub
A2B0 Mode: sub
```

Sub Node Audio Configuration

Sub node mode requires no configuration for normal I^{2}S or TDM audio. The A^{2}B Bridge configures itself automatically for all supported TDM formats.

The detected settings can be found in the syslog. Below is an example of what appears in syslog when discovered.

```
[  1583.991]  A2B Slave SPORT Start: 02:77
[  1583.991]  AD243x A2B SPORT CFG
[  1583.991]   Direction: TX (AD24xx DRX pins)
[  1583.991]   Size: 32-bit
[  1583.991]   TDM: 8
[  1583.991]   Data Pins: Both
[  1583.991]   Interleave: Yes
[  1583.991]   CLK: Assert falling, Sample rising (I2S)
[  1583.991]   FS: Rising edge
[  1583.991]   FS: Not Early
[  1583.991]   FS: Pulse
[  1583.991]  AD243x A2B SPORT CFG
[  1583.991]   Direction: RX (AD24xx DTX pins)
[  1583.991]   Size: 32-bit
[  1583.991]   TDM: 8
[  1583.991]   Data Pins: Both
[  1583.991]   Interleave: Yes
[  1583.991]   CLK: Assert rising, Sample falling
[  1583.991]   FS: Rising edge
[  1583.991]   FS: Not Early
[  1583.991]   FS: Pulse
```

This detailed I^{2}S / TDM information can be used by developers or system integrators to confirm proper A^{2}B Sub node configuration.

Sub Node Emulation Limitations

The A²B Main Node fully configures all Sub node transceiver registers and peripherals during discovery. Not all peripherals and modes of operation are available on the A²B Bridge when operating in Sub node mode. The items below highlight areas that may require changes to the A²B network configuration to maintain compatibility with the A²B Bridge.

TDM / PDM Audio

When possible, it is recommended to use the following settings for maximum audio compatibility with the A²B Bridge and between A²B transceiver types

A²B 1.0

A²B Register	**Value**
PDMCTL	0x00
I2SGCFG	Any
I2SCFG (AD243x)	RXPINS: 0b000, 0b001, 0b111 TXPINS: 0b000, 0b001, 0b111
I2SCFG (AD242x)	Any

NOTE: *Incompatible AD243x I2SCFG register settings will be automatically overwritten to 0xF7 after discovery.*

A²B 2.0

All SIO channel config registers are overwritten after discovery to ensure network byte order slot mapping.

PDM mode is not supported by the A²B Bridge. When attempting to emulate a PDM Sub node with the A²B Bridge it is necessary to modify the node configuration to use TDM and not PDM.

I²C

The A²B Bridge does not support I²C peripheral emulation. It may be necessary to remove I²C peripheral transactions from the node configuration if the lack of peripheral emulation results in a discovery error.

The Pocket A²B Bridge makes the A²B I²C pins available on the I/O connector. A physical device can be wired to these pins. Be advised, any device attached to the I²C pins may interfere with the proper operation of the A²B Bridge.

SPI

The A²B Bridge does not support SPI peripheral emulation. It may be necessary to remove SPI peripheral transactions from the network configuration if the lack of peripheral emulation results in a discovery error.

The Pocket A²B Bridge makes the A²B SPI pins available on the I/O connector. A physical device can be wired to these pins. Be advised, any device attached to the SPI pins may interfere with the proper operation of the A²B Bridge.

For A²B 1.0 networks, the ADR1/SPISSEL0 pin is exposed as the SPI slave select.

For A²B 2.0 networks, the TS0 pin is exposed as the SPI slave select.

Audio Data Pins

For A²B 1.0 networks, the A²B Bridge supports only 1-pin or 2-pin RX and TX for the AD243x transceiver. All data pin modes are supported for the AD242x transceiver.

The following audio data pin configurations are supported in Sub node mode

AD243x

SIOx Pin	**Supported Function**
SIO0	DRX0
SIO1	Off, DRX1
SIO2	Not supported
SIO3	Off, DTX1
SIO4	DTX0

AD242x

Data Pin	**Supported Function**
RX0	Off, RX0
RX1	Off, RX1
TX0	Off, TX0
TX1	Off, TX1

For A²B 2.0 networks, the A²B Bridge supports 2-pin RX and TX. All SIO channel config registers are overwritten after discovery to ensure network byte order slot mapping. Only Clock Domain 0, SYNC0 / BCLK0, is supported.

ADAA2457

SIOx Pin	**Supported Function**
SIO0	DRX0
SIO1	DRX1
SIO2	Not supported
SIO3	DTX1

SIO4	DTX0
SIO5	Not supported
SIO6	Not supported
SIO7	Not supported

Audio

General

The A^{2}B Bridge processes audio in blocks of 32 samples. Latency through the A^{2}B Bridge for all sources and destinations, except for USB, is 666.67uS. USB audio latency is approximately 960 samples, or 20mS. Latency through USB is generally controlled to within +/- 10 samples of nominal.

Internal audio samples are all 32-bits wide. Audio is up / down converted as required to 32-bits from 16-bit sources / destinations. Samples down-converted from 32-bit to 16-bit are truncated.

Audio routing is “bit perfect” from source to destination except when routed through an ASRC.

Introduction to Audio Clock Domains

All audio on A^{2}B is synchronous to the A^{2}B Main node. In other words, A^{2}B audio is always on the Main node's “Audio Clock Domain”. This is referred to as the A^{2}B clock domain.

Audio internal to the A^{2}B Bridge is on an internal clock domain called the System clock domain.
When an A^{2}B port is emulating a Main node, A^{2}B audio is on the System clock domain.

When an A^{2}B port is emulating a Sub node, inherent differences in the A^{2}B and System clocks cause audio to drift apart over time even if they share the same sampling frequency. If too much audio accumulates between a source and destination, an audio overrun will occur. Similarly, if not enough audio is received an underrun occurs. In either case, the result is an interruption in audio as buffers between the two clock domains are reset.

At a practical level, this means digital audio cannot be simply copied, or routed, between clock domains. The audio has to 1) go through an Asynchronous Sample Rate Converter (ASRC) or 2) the source and destination must be on the same clock domain.

On the Pocket A^{2}B Bridge, all digital audio sources and destinations, except ***plus***Audio line and S/PDIF, are moved to the A^{2}B clock domain when put into Sub node mode and moved back to the System clock domain when placed into Main node mode. This helps ensure audio can be routed between most sources and destinations regardless of the A2B mode. ***Plus***Audio line and S/PDIF remain on the system clock domain at all times.

In general, the Industrial A^{2}B Bridge does not automatically switch clock domains because it has multiple A^{2}B ports. Each A^{2}B port, in Sub node mode, is on its own clock domain and all ports in Main node mode are on the System clock domain. The Industrial A^{2}B Bridge comes equipped with 4 ASRCs that can be used to route audio between all of these clock domains.

NOTE: The *USB clock domain on the Industrial A²B Bridge follows the A2B0 clock domain. When A2B0 is in Main node mode, USB is on the System clock domain. When A2B0 is in Sub node mode, USB is on the A2B0 clock domain. This feature improves system level compatibility with the Pocket A²B Bridge, and ensures a bit-perfect audio path between USB and A2B0 in Subnode mode. The USB clock domain can be overridden at any time using the USB command.*

More detail on the impact of clock domains on each digital source or destination is covered in the Audio section.

Clock Domain Management

It is not possible to route audio directly between clock domains. The following best practices will help with clock domain management in A²B Sub node mode.

1. Route Sub node audio through an ASRC when routing to or from other System clock domain sources.
2. When bit-perfect audio is required, change the clock domain of the source or destination to match the clock domain of the A²B Sub node.
3. Signal generators automatically attach to the clock domain of the source of their first route. Reusing a signal generator in multiple clock domains will result in distorted audio.

A²B

All A²B ports are configured for 32 input slots, 32 output slots, by 32-bits. In Main node mode, the local A²B TDM audio settings are overridden during discovery to comply with these settings.

A²B Audio is on the System clock domain in Main node mode and on the A²B clock domain in Sub node mode.

When creating A²B networks for the A²B Bridge in Main node mode, it is advised to use the register settings in the tables below.

A²B 1.0 Registers

Register	**Value**
I2SGCFG	0x04
I2SCFG	0xF7
PDMCTL	0x00
PDMCTL2 (AD243x)	0x00

A²B 1.0 Slot Mapping

Downstream and upstream A²B 1.0 audio slots are processed and presented as configured by the A²B transceiver xxSLOT register settings. The slot order in the routing table represents the local A²B TDM slot, not the A²B slot.

A²B 2.0 Registers

Register	Value
SIO_SIOGCFG0	0x8100
SIO_SIOCLKCFG0	0x1340
SIO_SIOPINCFG0	0x9088
SIO_SIOPINCFG1	0x0009

A²B 2.0 Slot Mapping

Downstream and upstream A²B 2.0 audio data are processed and presented in network byte order where the first audio slot contains data at the lowest byte offset, the second slot contains data at the next lowest byte offset, and so on. Incoming (source) and outgoing (destination) A²B audio slots start with downstream audio followed by upstream data.

USB Audio

By default the A²B Bridge supports 20 IN (to PC), 20 OUT (from PC), by 16-bit audio. These settings can be modified by the `cfg.ini` file as necessary. USB audio operates on the System clock domain. The `'usb'` command can be used to view USB audio statistics.

View the audio statistics during long-term USB audio recording or playback when testing is sensitive to gaps in audio. Windows 10/11 are not real-time operating systems and can fail to transfer USB audio in a timely manner during periods of high system load.

To listen directly to USB audio from the A²B Bridge on Windows, do the following.

1. "Navigate to Control Panel -> Hardware and Sound -> Sound" or search and launch "mmsys.cpl" on Windows 11.
2. Select the "Recording" tab and locate the Pocket A²B Bridge
3. Select the Pocket A²B Bridge then click Properties
4. Select the "Listen" tab and check "Listen to this device"

A short YouTube video demonstrating this process can be found here.

*plus*Audio

PlusAudio devices support stereo line level analog and digital S/PDIF through an Audio I/O accessory expansion board. The S/PDIF receiver and transmitter can be selectively enabled and disabled using the `'spdif'` command to improve analog audio performance or reduce radiated EMI. The S/PDIF transmitter is disabled by default.

NOTE: Always remove power before connecting or disconnecting the Audio I/O accessory board from the Pocket A^{2}B Bridge.

Internal Signal Generators

The A^{2}B Bridge has built-in signal generators for testing. Sine tones, Pink noise, White noise, and fixed digital HEX values are all supported.

	Pocket A^{2}B Bridge	**Industrial A^{2}B Bridge**
Number of Signal Generators	16	16

Signal generators are not associated with a clock domain. They are automatically tied to the clock domain of the first route destination that uses the signal generator as a source. A signal generator can be used multiple times within a clock domain, but must not be reused across clock domains.

Use the `'gen'` command to configure signal generators.

WAV Files

The A^{2}B Bridge can play and record 16-bit or 32-bit multi-channel WAV files. WAV files up to thirty two 32-bit channels are supported assuming the SD card has sufficient bandwidth. Use the `'sdtest'` command to confirm SD card bandwidth.

A single WAV file can be the source or destination of multiple routes. If an audio route (offset plus channels) extends beyond the number of channels available in a WAV file destination the extra channels are dropped. Empty WAV destination channels are zero filled. If a route extends beyond the channels available in a WAV file source, the missing channels are zero-filled.

Use the `'wav'` command to start and stop WAV file playback or recording.

WAV files default to the System clock domain but can be manually assigned to a different clock domain using the `'wav'` command.

The Pocket A^{2}B Bridge automatically switches WAV files to the A2B0 clock domain when entering Sub node mode and returns WAV files to the System clock domain when A2B0 is set to Main node mode.

NOTE: *Always use a minimum Class 10 or UHS Class 1 SD card if the card will be used for WAV file audio. Use a freshly formatted SD card when recording high bit-rate WAV files to reduce the risk of audio drops. Use the* `'sdtest'` *command to confirm acceptable SD card bandwidth.*

Circular Audio Buffer (CBUF)

The A²B Bridge has a programmable circular memory buffer (CBUF) that can be used to record a rolling window of audio. The audio inside the CBUF can be dumped to a WAV file on the SD card at any time. This feature can be used to capture transient audio events without having to record a continuous WAV file.

Use the `'cbuf'` command to configure and control the CBUF.

VU Meters

The A²B Bridge has a set of simple built in VU meters. The VU meters are very useful for locating upstream audio on A²B. All 32 upstream A²B slots can be routed to the VU meters to observe all upstream A²B audio.

VU meters can be the destination of multiple routes.

Use the `'vu'` command to view the VU meters.

The VU meters default to the System clock domain. VU meters can be viewed or manually assigned to a different clock domain using the `'vu'` command.

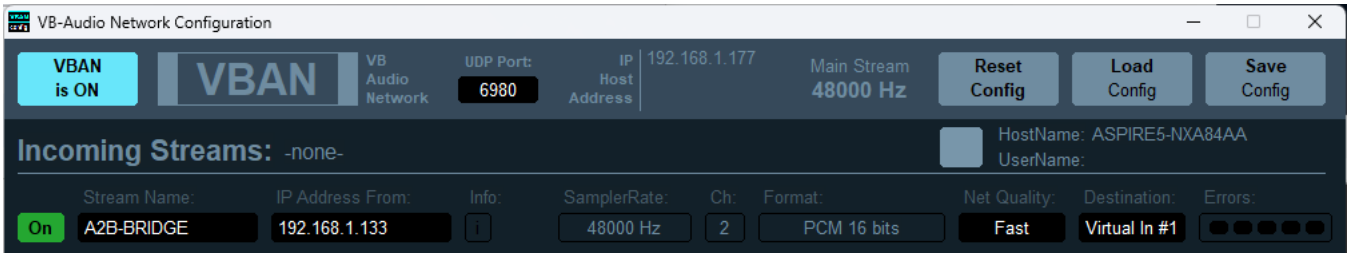
The Pocket A²B Bridge automatically switches VU meters to the A2B0 clock domain when entering Sub node mode and returns the VU meters to the System clock domain when A2B0 is set to Main node mode.

VBAN

The Industrial A²B Bridge supports the VBAN audio transport protocol over Ethernet. VBAN is an easy way to transfer digital audio over Ethernet.

VBAN should only be used as the destination of an audio route. VBAN streams are on the System clock domain. The VBAN stream name is “A²B-BRIDGE”. Use the `'vban'` command to configure VBAN streams.

An example configuration for Voicemeeter Banana, a VBAN streaming and mixing application, is illustrated below



Voicemeeter can be found here https://vb-audio.com/Voicemeeter/index.htm

RTP

The Industrial Bridge supports RTP streaming audio over Ethernet. RTP streaming is supported by gstreamer and other audio applications.

RTP should only be used as the destination of an audio route. RTP streams are on the System clock domain. Use the `'rtp'` command to configure RTP streams.

ASIO

The A^{2}B Bridge supports ASIO on Windows through the FlexASIO driver. Please refer to the FlexASIO Github page for more information.

Audio Routing

The `'route'` command is the key command for transferring audio between audio streams on the A^{2}B Bridge. The A^{2}B Bridge supports simultaneous multi-channel routes enabling very sophisticated audio routing schemes. The routing engine is a full crossbar between any source and any destination within a clock domain.

The table below describes the routing capabilities of the A^{2}B Bridge

	Pocket A^{2}B Bridge	**Industrial A^{2}B Bridge**
Number of Routes	16	16

Stream	**Src / Dest**	**Pocket A^{2}B Bridge**	**Industrial A^{2}B Bridge**	**Notes**
usb	Both	Yes	Yes	
a2b	Both	Yes	Yes	
wav	Both	Yes	Yes	
cbuf	Dest	Yes	Yes	
rtp	Dest	No	Yes	Ethernet link required
vban	Dest	No	Yes	Ethernet link required
asrc	Both	No	Yes	Industrial Bridge only
gen	Src	Yes	Yes	
vu	Dest	Yes	Yes	
line	Both	Yes	No	***plus***Audio only
spdif	Both	Yes	No	***plus***Audio only

An audio route starts with a source stream and source stream index. The source stream is the base stream, like `'a2b'` or `'gen'`. The source stream index indicates which source port to use. For example, A2B0 is the `'a2b'` source stream with index 0. A2B1 is the `'a2b'` source stream with index 1.

A specific channel within a source stream is identified with the source channel offset. For A^{2}B the source channel offset refers directly to the local A^{2}B transceiver TDM slot. For USB audio, the offset refers to the channel offset.

The route source must then connect to a destination. Destination streams are identified by stream, index, and offset exactly like source streams.

A route copies a defined number of channels from the source to the destination. Audio can be optionally attenuated or mixed during this copy.

Route command arguments are :

```
route [ <idx> <src> <index> <offset> <dst> <index> <offset> <channels> [ attenuation ] [ mix|set ] ]
```

Below are some examples:

Route 2 channels from A2B1 TDM slot 0 to A2B2 TDM slot 0 with no attenuation

```
route 0 a2b 1 0 a2b 2 0 2
```

Route 4 channels from A2B0 TDM slot 2 to USB channel offset 0 with 6db of attenuation and mixed

```
route 1 a2b 0 2 usb 0 0 4 6 mix
```

Route 8 channels from both A2B0 and A2B1 to the VU meters

```
route 2 a2b 0 0 vu 0 0 8
route 3 a2b 1 0 vu 0 8 8
```

Route 4 channels from USB to A2B2 TDM slot 4

```
route 4 usb 0 0 a2b 2 4 4
```

A route that exceeds the number of source channels (i.e. source offset + channels > source channels) will zero fill the missing channels. Routes that exceed the number of destination channels (destination offset + channels > destination channels) will terminate after the last destination channel.

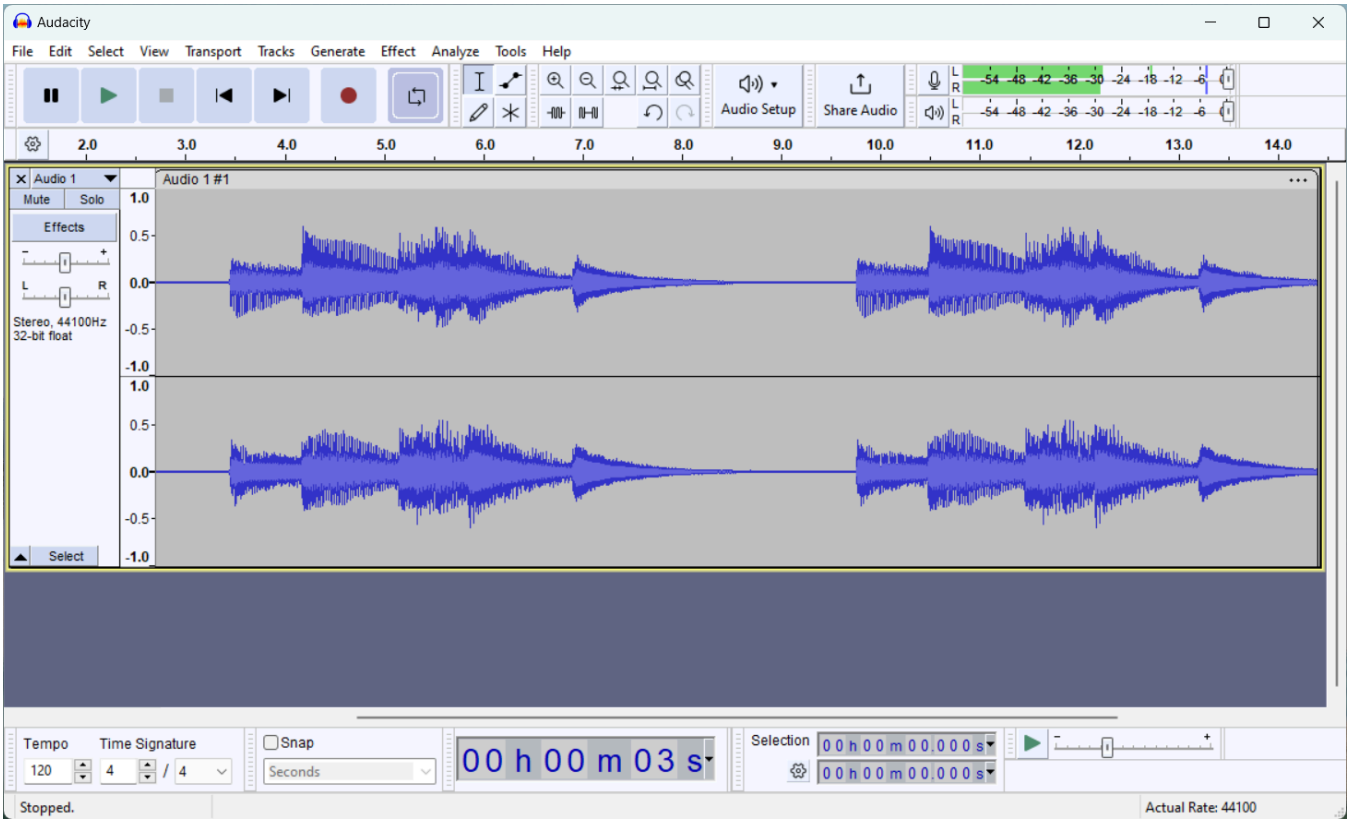
Peak Detectors

The A^{2}B Bridge supports real-time peak detectors on all A^{2}B ports. These peak detectors continuously monitor A^{2}B audio. They can be viewed or cleared using the `'peaks'` command.

Audio Recording and Playback

After A²B audio channels are routed to USB **you can use a Digital Audio Workstation (DAW),** such as Audacity, Reaper, and others to capture, record, analyze and playback any audio stream from the A²B Bridge.

Audacity is a free and easy to use Audio Workstation program.
Click here to access the latest version and user guide.



Qwiic Peripheral Configuration

Qwiic peripheral configuration via I²C on the qwiic connector is supported through the command line and AKT Automation scripts.

I²C Commands

Command	Purpose
`qwiic_i2c`	Performs an I²C write, read, or write/read transaction on the qwiic connector
`qwiic_scan`	Scans the I²C bus on the qwiic connector for active devices

NOTE: *The I²C bus speed is fixed at 400KHz*

Qwiic Examples

Scan the qwiic connector for I²C devices

```
# qwiic_scan
Probing I2C port 2:
 Found device 0x68
```

1 byte write / 16 byte read I²C transaction to device 0x68 on the qwiic connector

```
# qwiic_i2c 0x68 "0" 16
I2C Device 0x68, Read Bytes 16 (0x10)
00000000: 34 07 20 02 06 11 24 00 00 00 00 00 00 00 1c 88
```

Command Scripts

Any series of commands can be grouped together into a command script and executed using the `'run'` command. Command scripts are simple text files containing one command per line. Lines starting with a semicolon or hash mark are ignored and treated as comments. Any valid file name can be used for command scripts.

Some commands are especially useful in command script processing

Command	Purpose
`delay`	Delays script execution for a specified number of milliseconds
`echo`	Displays a line of text
`shell redirect`	Silences or redirects output of a script to the syslog
`reset`	Reset various subsystems to power on reset values. No arguments performs a full system reset.

Be careful scripting interactive commands as the script will not proceed until the interactive command has completed. Interactive commands include `'syslog', 'vu'`, and interactive `'lua'` scripts.

Below is an example command script discovering a node on A2B0 and setting up bi-directional audio routes between USB and A^{2}B. USB audio is shown on the first two VU meters and A^{2}B audio is shown on the next two VU meters:

```
# Perform a soft reset
reset soft
# Delay for 200mS
delay 200
# Route two channels between USB and A2B0
route 0 usb 0 0 a2b 0 0 2
route 1 a2b 0 0 usb 0 0 2
# Route USB and A2B0 to the VU meters
route 2 usb 0 0 vu 0 0 2
route 3 a2b 0 0 vu 0 2 2
# Discover on A2B0
discover a2b0 a2b-test.xml
```

Running commands at startup

If present, the A^{2}B Bridge runs `'sf:shell.cmd'` at startup. This feature allows for autonomous configuration of the A^{2}B Bridge at startup.

NOTE: *The SD card takes some time after startup to initialize. Be sure to add a delay of at least 500ms at the beginning of any startup script that uses files on the SD card.*

NOTE: *Be careful including interactive commands in the startup script. For example, launching a Lua script that never terminates will lock out the command line requiring a Safe Boot Recovery to correct.*

AKT Automation with Lua

Fully Autonomous Automation on the A^{2}B Bridge is made possible through the on-board Lua scripting environment. This feature is standard on the Industrial A^{2}B Bridge and an optional add-on for the Pocket A^{2}B Bridge.

Lua is a fully-featured open source scripting language. More information on Lua, including programmer reference manuals, can be found at https://lua.org/.

The pairing of Lua with the A^{2}B Bridge command and control APIs results in an extremely rich interactive A^{2}B automation environment.

User interfaces can be created using the `'term'` module. A full complement of A^{2}B operations are available through the `'master'` and `'setup'` modules. Low-level operations are possible using the `'freertos'` and `'system'` modules.

For more detailed information on the API, refer to the ***FlexTech AKT A^{2}B Bridge API Guide***. The A^{2}B Bridge also supports many of the Flextech AKT Automation API. For more detailed information on the Automation API, refer to the ***FlexTech AKT Automation API Guide***.

There can be overlapping APIs for certain features like routing, signal generators, etc.. While either API can be used, always prefer the A^{2}B Bridge API as these API operate at a higher level with additional protections for multi-bus and multi-API accesses.

Like the command script in the previous section, the Lua script below discovers a node on A2B0 and sets up bi-directional audio routes between USB and A^{2}B. USB audio is shown on the first two VU meters and A^{2}B audio is shown on the next two VU meters.

This example goes one step further by polling Bit Error Rates on the first Sub node every second and showing the bit error count if the bit error rates are non-zero.

NOTE: *This script is simplified for clarity and does not have the API locking required for multi-bus or multi-API operation.*

```
--[[
;
; AKT Automation
; Discovery and bit error example
;
--]]
master = require("master")
setup = require("setup")
rtos = require("freertos")
term = require("term")

-- Perform a soft reset
print("Soft reset...")
ok, msg = setup.reset("soft")

-- Delay 200mS
print("Delay...")
ok, msg = rtos.delay(0.200)

-- Route two channels between USB and A2B0
print("USB/A2B0 Routes...")
ok, msg = setup.setRoute(0, 'usb', 0, 0, 'a2b', 0, 0, 2)
ok, msg = setup.setRoute(1, 'a2b', 0, 0, 'usb', 0, 0, 2)
-- Route USB and A2B0 to the VU meters
print("VU Routes...")
ok, msg = setup.setRoute(2, 'usb', 0, 0, 'vu', 0, 0, 2)
ok, msg = setup.setRoute(3, 'a2b', 0, 0, 'vu', 0, 2, 2)

-- Discover on A2B0
print("Discover...")
ok, msg = setup.setBus("a2b0")
ok, msg = setup.setNetwork("a2b-test.xml")
if not ok then
    print("setup.setNetwork() error")
    print(msg)
    return
end
ok, msg, nodes, retries = master.discover()
if not ok then
    print("master.discover() error")
    print(msg)
    return
end

print("Monitoring Bit Errors...")
print("Press any key to exit...")

A2B_BECCTL_REG = 0x1E
A2B_BECCTL_ALL = 0x1F
A2B_BECNT_REG = 0x1F
A2B_SUB_NODE = 0
```

```
-- Enable monitoring of all bit errors on Sub node 0
ok = master.i2cWriteRead(A2B_SUB_NODE, { A2B_BECCTL_REG, A2B_BECCTL_ALL })
if not ok then
    print("Error enabling BER monitoring")
    return
end

-- Check for bit errors every second
-- Exit on error or keypress
repeat
    ok, BER = master.i2cWriteRead(0, { A2B_BECNT_REG }, 1)
    if ok and BER[1] > 0 then
        print(string.format("%u bit errors detected", BER[1]))
        master.i2cWriteRead(0, { A2B_BECNT_REG, 0 }) -- Clear
    else
        if not ok then
            print(BER)
        end
    end
    if ok then
        rtos.sleep(1)
    end
    key = term.getchar(term.NOWAIT)
until not ok or key ~= -1

-- Exit
print("Goodbye")
```

When developing Lua scripts, the following techniques can significantly speed up script development:

1. On the Industrial A^{2}B Bridge, use Ethernet and TFTP or FTP to transfer the script from the PC to the A^{2}B Bridge
2. Use XMODEM to transfer scripts to the A^{2}B Bridge instead of the SD card
3. Running Lua with no arguments starts an interactive Lua interpreter. Lua code "chunks" can be copied and pasted from the PC into the interpreter for quick prototyping of logic or code blocks.
4. Use the on-board 'edit' command for quick bug fixes or script modifications.

Remote AKT Automation with Python

The A^{2}B Bridge supports remote command and control over USB. The Industrial A^{2}B Bridge also supports command and control over Ethernet.

Python contains all of the necessary components to easily communicate with the A^{2}B Bridge on both Windows and Linux.

For more detailed information on the API, refer to the FlexTech AKT A^{2}B Bridge API Guide.

```
import requests
import json
import serial
```

```
import time
import signal
import sys

from jsonrpcclient import request, parse, Ok, Error, parse_json

QUIT = False

def sig_int(sig, frame):
    global QUIT
    QUIT = True

signal.signal(signal.SIGINT, sig_int)

# Helper class for USB RESTful API
class A2BBridgeAPI():

    def __init__(self, com_port):
        self.ser = serial.Serial(com_port, 115200, timeout = 5)

    def postRequest(self, api):
        if self.ser is not None:
            req = '\x1B]0;' + json.dumps(api) + '\x07'
            self.ser.write(req.encode())
            resp = self.ser.read_until(expected=b'\x07')
            if len(resp) > 7:
                resp = parse_json(resp[4:-3].decode())
                ok = isinstance(resp, Ok)
                if not ok:
                    print(f'Request: {api}')
                    print(f'Error: {resp.code}, {resp.message}')
                    return ok, resp.message
                else:
                    return ok, resp.result
        return False, None

    def execute(self, method, p):
        ok, resp = self.postRequest(request(method, params = p))
        return ok, resp

api = A2BBridgeAPI("COM39")

# Perform a soft reset
ok, resp = api.execute('setup.reset', {'type':'soft'})

# Delay 200mS
print("Delay...")
time.sleep(0.200)

# Route two channels between USB and A2B0
print("USB/A2B0 Routes...")
ok, resp = api.execute('setup.setRoute',
    { 'id':0, 'src':'usb', 'srcId':0, 'srcOffset':0,
              'dst':'a2b', 'dstId':0, 'dstOffset':0, 'channels':2})
ok, resp = api.execute('setup.setRoute',
    { 'id':1, 'src':'a2b', 'srcId':0, 'srcOffset':0,
```

```
                'dst':'usb', 'dstId':0, 'dstOffset':0, 'channels':2})

    # Route USB and A2B0 to the VU meters
    print("VU Routes...")
    ok, resp = api.execute('setup.setRoute',
        { 'id':0, 'src':'usb', 'srcId':0, 'srcOffset':0,
                'dst':'vu', 'dstId':0, 'dstOffset':0, 'channels':2})
    ok, resp = api.execute('setup.setRoute',
        { 'id':1, 'src':'a2b', 'srcId':0, 'srcOffset':0,
                'dst':'vu', 'dstId':0, 'dstOffset':2, 'channels':2})

    # Discover on A2B0
    print("Discover...")
    ok, resp = api.execute('setup.setBus', {'bus':'a2b0'})
    ok, resp = api.execute('setup.setNetwork',
        {'network':'a2b-test.xml', 'type':'ss-xml'})
    if not ok:
        print("setup.setNetwork() error")
        print(resp)
        exit()
    ok, resp = api.execute('master.discover', {})
    if not ok:
        print("master.discover() error")
        print(resp)
        exit()

    print("Monitoring Bit Errors...")
    print("Press <ctrl-c> to exit...")

    A2B_BECCTL_REG = 0x1E
    A2B_BECCTL_ALL = 0x1F
    A2B_BECNT_REG = 0x1F
    A2B_SUB_NODE = 0

    # Enable monitoring of all bit errors on Sub node 0
    ok, resp = api.execute('master.i2cWriteRead',
        { 'nodeAddr':A2B_SUB_NODE,
          'wBuf':[A2B_BECCTL_REG, A2B_BECCTL_ALL ], 'nRead':0})
    if not ok:
        print("Error enabling BER monitoring")
        exit()


    # Check for bit errors every second
    # Exit on error or CTRL-C

    while True:
        ok, BER = api.execute('master.i2cWriteRead',
            { 'nodeAddr':A2B_SUB_NODE, 'wBuf':[A2B_BECNT_REG], 'nRead':1}
        )
        if ok and BER['bytes'][0] > 0:
            print(f"{BER['bytes'][0]} bit errors detected")
            ok, BER = api.execute('master.i2cWriteRead',
                { 'nodeAddr':A2B_SUB_NODE,
                  'wBuf':[A2B_BECNT_REG,0], 'nRead':0}) # Clear
        else:
```

```
            if not ok:
                print(BER)
        if ok:
            time.sleep(1)
        if not ok or QUIT:
            break

    # Exit
    print("Goodbye")
```

AKT Automation Github Repository

The latest versions of the examples shown here, along with many other examples, are available in the AKT Automation Github repository.

Updating the Firmware

***WARNING*:** *Prior to updating the firmware, always be sure to disable any custom **cfg.ini** and **sf:shell.cmd** startup files. These may interfere with the proper startup of the new firmware causing the update to fail unexpectedly.*

Download the AKT Flasher Utility from www.flextechakt.com and install.

Follow one of the methods below to update the firmware.

Methods to Update the Firmware

Command-line Initiated AKT Flasher Over USB.

To start this update, issue the following commands from the Tera Term command line:

```
# bootmode 1
# reset
```

Proceed to update with AKT Flasher utility. Once the `'reset'` command is issued, the unit must be updated using the AKT Flasher.

Command-line via SD card file

To initiate this update, the desired firmware binary must be on the SD card. One can simply copy it from the PC or use the `'recv'` command to transfer it through TeraTerm via XMODEM over USB.

Once the file is on the SD card, issue the following command:

```
# update <file>
```

Where `<file>` is the firmware binary. Reset the unit once the update is complete using the `'reset'` command or cycling power.

Bootloader initiated AKT Flasher over USB

Press and hold the "Boot Recovery" button while powering on the unit. The unit will immediately enter the bootloader mode and can be updated using the AKT Flasher utility. If the unit is reset before updating the firmware, it will boot normally. Once an update is initiated with AKT Flasher, the update must complete.

Bootloader mode is indicated by a slow yellow blink of the Status LED on the Pocket products.

NOTE: The "Boot Recovery" button is located under the small hole on the underside of the Pocket A^2B Bridge and behind the hole on the rear panel of the Industrial A^2B Bridge.

Chapter 4. Connectors & Cables

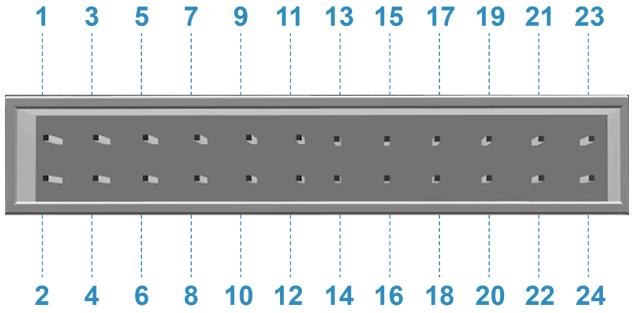
Pocket Bridge Connectors

USB

The USB connector is a standard USB 2.0 Type B receptacle.

24 Pin Multi I/O Connector

JST S24B-PHDSS



Pin	1	3	5	7	9	11	13	15	17	19	21	23
Description	GND	GPIO0		Left In[1][2]	Left Out[1][2]	A²B IO7	SPDIF Out[1][3]	A²B SCLK	A-	A+	B+	B-
Color	Black	Blue		Blue	Yellow		Purple		Brown			Brown

Pin	2	4	6	8	10	12	14	16	18	20	22	24
Description	3.3V	GPIO1		Right In[1][2]	Right Out[1][2]		SPDIF In[1][3]	A²B MISO[4]	A²B MOSI[4]	CS	A²B SDA	A²B SCL
Color	Red	Purple		Green	Orange		Gray					

Note: Color shown matches the provided harness for each signal. Gray sections represent unterminated pins.
Note[1]: Available on ***plus****Audio* Pocket Products Only
Note[2]: Single Ended Line Level Analog I/O
Note[3]: 3.3V Logic Level SPDIF I/O
Note[4]: A2B 2.0 devices present MISO on pin 18 and MOSI on pin 16

The I/O Header on the Pocket Bridge accepts the JST PHDR-24VS Socket Housing. Click here for the full data sheet for the JST PHD connector series.

Industrial Bridge Connectors

The 2-port and 4-port Industrial A^{2}B Bridges can emulate both an A^{2}B Main node and Sub node. The 6-port Industrial Bridge can only emulate an A^{2}B Main node.

As a Main node, connect the “B” side to the first downstream Sub node.

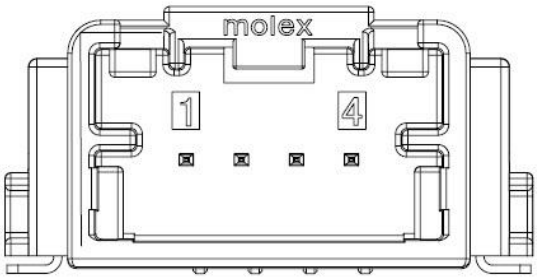
As a Sub Node, connect the “A” side to the next upstream node (Sub Node or Main Node) and the “B” side to the next downstream Sub node. Leave the “B” side disconnected if the Bridge is the last Sub node.

Always connect plus to plus and minus to minus.

The Molex Mini50 connector, pinout, and mating part numbers are indicated below.

A^{2}B Bridge Header Type: Molex: 34912-8040 Black
Connector Pinout: Pin 1: AN(-) Brown* Pin 2: AP(+) White* Pin 3: BP(+) White Pin 4: BN(-) Brown
Connector Receptacle Housing: Molex: 34791-0040 Black
Terminal: 560023-0448

* A-Side connections are only available on Sub node capable devices



Analog Devices Evaluation Platform Connectors

The A-side and B-side on most Analog Devices evaluation boards use Molex DuraClik connectors (Molex part number 502352-0200).

- When looking into the DuraClik headers on ADI evaluation boards, pin 1 is on the left side and pin 2 is on the right side.
- The polarity of the A^2B signals is inverted between A-side and the B-side.
- On the A-side, which faces the Main node, pin 1 is positive and pin 2 is negative.
- On the B-side, which faces the last Sub node, pin 1 is negative and pin 2 is positive.

Fabricating Pocket A^{2}B Bridge Cables

After completing the steps in this procedure, you will have A-side and B-side cables that connect the Pocket Bridge to an Analog Devices evaluation platform.

You might need to craft your own custom cables to connect the Pocket Bridge to your A^{2}B network. This section describes how to fabricate a Pocket A^{2}B cable harness with a JST socket housing on the Pocket Bridge side and DuraClik housings, compatible with Analog Devices evaluation platforms, on the other.

Note: Adjust these instructions as required for your own hardware.

Note: When connecting the A-side or B-side of an A^{2}B Pocket Bridge to an ADI evaluation platform, you can start with the provided cable harness with DuraClik connectors.

Prerequisites

Item	Qty	Mfg.	Mfg. P/N	Dist.	Dist. P/N
DuraClik plug	2	Molex	502351-0200	Mouser	538-502351-020
Duraclik Crimp Terminal	4	Molex	50212-8100	Mouser	538-50212-8100
Duraclik Crimp Tool		Molex	63823-5100	Mouser	538-63823-5100
JST Socket Housing	1	JST	JST PHDR-24VS	DigiKey	455-1177-ND
JST Terminal	4	JST	SPHD-001T-P0.5	DigiKey	455-1325-1-ND
JST Crimp Tool	1	JST	WC-240	DigiKey	455-1128-ND
Alternate Crimp Tool		Engineer	PA-09	Amazon	PA-09
Wire, Brown, 24AWG		Any	UL1061 24AWG		
Wire, White, 24 AWG		Any	UL106124AWG		
Cable, 2 Cond, 24 AWG, Black		Belden	1353A 010	Digikey	BEL1253-1000-ND

Procedure

1. If using discrete wires, cut the white and brown wires to the desired length considering that the wires must be twisted pairs. If using the Belden cable, cut to the desired length. Ensure no copper is exposed while separating the conductors.

2. If using discrete wires for the A^{2}B cables, twist the wires at 0.4 twist per cm or 1 twist per inch.

3. Strip and crimp DuraClik terminals onto one end of the wires or cable using a Molex compatible crimp tool.

4. Insert the terminals into the DuraClik plug, carefully noting the polarity of the cables and the placement of the conductors.

5. For the A-side cable, insert the white wire into pin 1 (left side) of the DuraClik plug. Insert brown wire into pin 2. For the B-side cable, insert the brown wire into pin 1 (left side) of the DuraClik plug. Insert the white wire into pin 2.

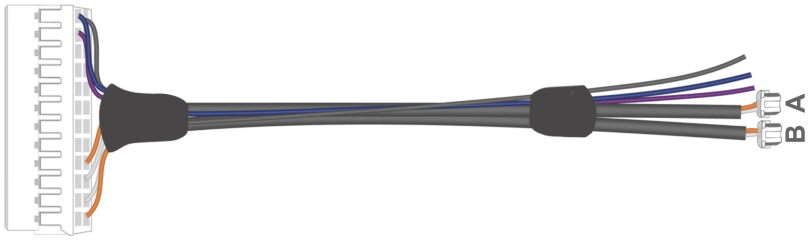
6. (Optional) Apply shrink tubing to dress the end of the cable.

7. Label your A-side and B-side cables.

8. Crimp JST terminals onto the opposite end using a JST compatible crimp tool.

9. Insert terminals into the JST plug, carefully selecting the terminal slots with polarity as shown in the JST 24 Pin I/O Header Pinout.

10. The following figure shows the finished cables.



Fabricating Industrial A^{2}B Bridge Cables

Bill of Materials

Item	Qty	Mfg.	Mfg. P/N	Dist.	Dist. P/N
DuraClik plug	2	Molex	502351-0200	Mouser	538-502351-020
Duraclik Crimp Terminal	4	Molex	50212-8100	Mouser	538-50212-8100
Duraclik Crimp Tool	1	Molex	63823-5100	Mouser	538-63823-5100
Molex 4pos Mini50 Housing	1	Molex	34791-0040 Black	Digikey	WM9381-ND
Molex CTX50 Terminal	4	Molex	560023-0448	Digikey	WM16315CT-ND
Molex Crimp Tool	1	Molex	63811-1000	Digikey	WM9999-ND
Wire, Brown, 24AWG		Any	UL1061 24AWG		
Wire, White, 24 AWG		Any	UL106124AWG		
Cable, 2 Cond, 24 AWG, Black		Belden	1353A 010	Digikey	BEL1253-1000-ND

Procedure

1. If using discrete wires, cut the white and brown wires to the desired length considering that the wires must be twisted pairs. If using the Belden cable, cut to the desired length. Ensure no conductor is exposed while separating the conductors.

2. If using discrete wires for the A^{2}B cables, twist the wires at 0.4 twist per cm or 1 twist per inch.

3. Strip and crimp DuraClik terminals onto one end of the wires or cable using a Molex compatible crimp tool. Do not overstrip the insulation. Ensure conductor does not protrude past the conductor crimp barrel.

4. Insert the terminals into the DuraClik plug, carefully noting the polarity of the cables and the placement of the brown and white wires.

5. For the A-side cable, insert the white wire into pin 1 (left side) of the DuraClik plug. Insert brown wire into pin 2. For the B-side cable, insert the brown wire into pin 1 (left side) of the DuraClik plug. Insert the white wire into pin 2.

6. (Optional) Apply shrink tubing to dress the end of the cable.

7. Label your A-side and B-side cables.

8. Crimp Molex CTX50 terminals onto the opposite wire ends using a Molex compatible crimp tool.

9. Insert terminals into Molex housing, carefully selecting the terminal connector position with polarity as shown in the Industrial Bridge Header Pinout.

Chapter 5. Specifications

This chapter provides technical specifications for the Pocket and Industrial A²B Bridge.

Environmental

Pocket A²B Bridge	Industrial A²B Bridge
DC Characteristics Target Power: USB +5V, 400mA max GPIO Signal: 3.3V, 10 mA **Dimensions (W x D x L)** 68 x 43 x 25 mm (2.7” x 1.8 x 1”) **Weight** 64 g (0.14 lbs) **Operating Temperature** 0° C to 70° C (32° C to 158° F)	**DC Power Characteristics** 12V 3 A **Dimensions (W x D x L)** 19” x 13” x 1.75” (with rack mounting brackets) **Weight** 5 lbs **Operating Temperature** 0° C to 70° C (32° C to 158° F)

Ordering Information

Pocket A²B Bridge	Industrial A²B Bridge
Part Number: AKT-1000-X(L) • X= A AD2428 Transceiver | AKT-1000-A OR • X= B AD2433 Transceiver | AKT-1000-B L = Optional LUA Scripting License **Country of Origin**: USA **HTS:** 8473.30.1180 **ECCN**: EAR99	Part Numbers: AKT-7076-XXXXXX AKT-7074-XXXX AKT-7072-XX X= A2B Transceiver Installed in Any Particular Slot Options for X include: None, A-2428, B-2433, C-2435 **Country of Origin**: USA **HTS:** 8473.30.1180 **ECCN**: EAR99

Chapter 6. Troubleshooting

Audio

- Confirm your device is running the latest firmware
- Basic USB audio can be confirmed by routing it back to itself, starting a recording in Audacity, then playing audio from another application.

```
route 0 usb 0 0 usb 0 0 2
```

- Use a Pocket A2B Bus Monitor to confirm audio is actually on A²B in the expected slots.
- To check all possible A²B slots for audio, route all 32 slots to the VU meters

```
route 1 a2b 0 0 vu 0 0 32
```

System Log

The A²B Bridge has an internal system log that contains useful system information. Review the system log using the `'syslog'` command. Technical support may also ask for the system log when diagnosing problems.

Safe-Boot

An incorrect `sf:cfg.ini or sf:shell.cmd` boot script may cause the system to not boot properly and require a Safe-Boot Recovery to fix. The Safe-Boot Recovery bypasses all custom configurations and boots the unit in its default factory state.

A Safe-Boot Firmware Update skips the application boot phase and directly launches the Bootloader for firmware updates.

On the Pocket A²B Bridge, the Safe-Boot button is located behind the small hole on the underside of the unit. On the Industrial A²B Bridge, the Safe-Boot button is located behind the small hole on the rear panel.

Safe-Boot Recovery

To initiate a Safe-Boot Recovery, press and hold the Safe Boot Button during the "Two Blink" boot phase.

On the Pocket A^{2}B Bridge, this is when the Status LED blinks Yellow twice following power up. On the Industrial Bridge, this is when the front panel LEDs blink twice. Do not press the Safe-Boot button prior to power up otherwise a Safe-Boot Firmware Update will be initiated. If Safe-Boot Firmware Update mode is entered accidentally, and no firmware update is needed, simply power cycle the unit to return to normal operation.

Safe-Boot Recovery can be confirmed by viewing the syslog. This line will be present at the top of the syslog following a Safe-Boot Recovery.

```
[     0.000]  Safe boot mode
```

Following the Safe-Boot, correct whatever caused the boot failure and reset the device to resume normal operation.

Safe-Boot Firmware Updates

A Safe-Boot Firmware Update requires the AKT Flasher utility. To enter Safe-Boot Firmware Update mode, press and hold the Safe-Boot button during power up.

On the Pocket A^{2}B Bridge, the Status LED will slow blink yellow indicating bootloader mode. The Industrial A^{2}B Bridge will slow blink the front panel LEDs.

At this point, use the AKT Flasher to update the firmware and power cycle the unit to complete the update.
If Safe-Boot Firmware Update mode was entered accidentally, and no firmware update is needed, simply power cycle the unit to return to normal operation.

Common Issues

Issue	Possible Cause	Solution
No USB Audio input in Windows	Windows Audio Enhancement is On for the A^{2}B Bridge	Open Windows Sound Settings, Select the A^{2}B Bridge audio device and confirm Audio Enhancements is Off.
No USB Audio in Windows	Modified USB audio settings in cfg.ini	Windows requires removing and reinstalling the device in the device manager when the audio settings are changed.
No Audio	Attempting to route audio across audio clock domains	It is not possible to route directly between audio clock domains. Insert an ASRC in the audio path to route audio across audio clock domain boundaries.
Status LED Flashing Red	The device has an invalid or missing feature license key	Check to see if you have a copy of the license file and re-install it on the internal sf: filesystem. If not, contact support at flextechakt.com
Pocket A^{2}B Bridge resets during A^{2}B discovery	USB cable is bad or too long causing power problems.	Use a shorter USB cable with at least 24 AWG power wires. The Monoprice Product #5437 3' USB cable supplied with the device meets these requirements.
A^{2}B Bridge not showing up as an audio device	Windows versions prior to Windows 10, release 1703 do not support the USB UAC2 audio protocol	Use a newer release of Windows
USB Audio playback dropouts	There is a rate-feedback bug in early versions of Windows 10 that can result in audio dropouts during playback.	The latest version of Windows 10 or 11 is required for proper operation. The `'usb'` command can be used to troubleshoot USB audio problems.
USB Audio dropouts	Windows 10/11 are not real-time operating systems and can fail to transfer USB audio in a timely manner.	Use a different host PC platform, like Linux or Mac, if USB audio is critical.