A2B Bridge
API Specification

Version 1.0
September 2024

This document is for information and instruction purposes. Flextech AKT 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 Corporation 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
24613 S 220th St.
Queen Creek, AZ 85142

Website: www.flextechakt.com

Table of Contents

Change Log

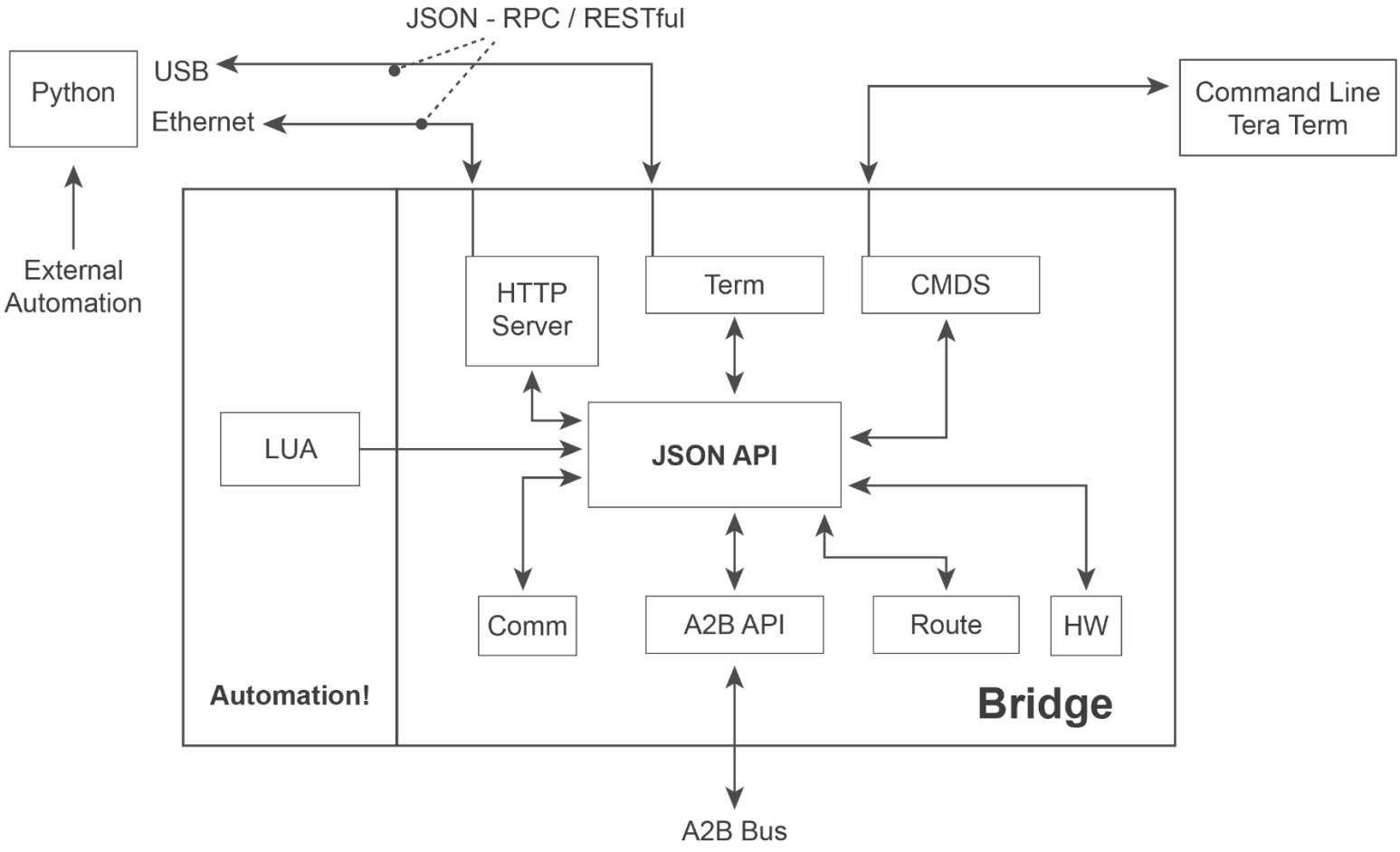
Version	Date	Changes
1.0	30-Sep-2024	Initial release

Introduction

This document describes the A2B Bridge command and control API. This API is available through a RESTful interface over USB, a RESTful interface over Ethernet, Lua scripting modules, or the serial console command-line.

Note: Both the Pocket Bridge and Industrial Bridge connect to a live A2B network via USB interface. While only the Industrial Bridge additionally connects via Ethernet.

This version of the specification is compatible with firmware version 2.2.0 and higher.



USB RESTful API

Transport

The USB RESTful API is the same as the Ethernet RESTful API except for the physical layer interface. RESTful commands over USB are tunneled through the command-line serial console using ANSI escape sequences. RESTful commands and responses are are encapsulated as follows:

<ESC>]0;<RESTful API Request String><BEL>

Where **<ESC> = 0x1B** and **<BEL> = 0x07**

Responses are encoded similarly as follows:

<ESC>]0;<RESTful API Response String><BEL>

Only one request should be in flight at any time. The maximum request size is 64KB.

Since API requests are tunneled through escape sequences, API request and command-line commands can be used simultaneously.

Ethernet RESTful API

Note: Ethernet capabilities only applicable to Industrial Bridge.

Transport

All commands shall be sent as HTTP POST requests to port 4040 on the hard-wired Ethernet port. The request URL shall have the form:

http://<ip>:<port>/<protocol_version>

Where <ip> is the IP address of the A2B Bridge, <port> is fixed to 4040, and <protocol_version> is a single digit number representing the API version. This digit represents a major API version and may be incompatible with other major versions. Minor updates to the protocol will maintain a consistent major version number but may result in default values being applied or deprecated values being ignored.

The major API version described in this document is 1 (one).

Protocol

All requests and responses shall comply with the JSON-RPC version 2.0 specification found here: https://www.jsonrpc.org/specification

JSON-RPC batch requests are not supported but similar capability is provided through the util.batch API.

All strings are case sensitive. The maximum request size is 64KB.

Normal Response

A normal response shall be in the following form:

{ "jsonrpc": "2.0", "id": 1, "response": { } }

"response" will be a command specific JSON object.

Error Response

{ "jsonrpc": "2.0", "id": 1, "error": {"code": -1, "message": "", "data":{} } }

"code" will be a unique negative integer
"message" will be short "friendly" error string
"data" will be a code specific JSON object

Lua Scripting API

Access to the API through the Lua scripting engine is provided through native Lua modules. These modules do not directly utilize the RESTful API but do follow the same internal flow as the RESTful API to achieve identical results.

Command-line API

Access to many API features is also available through the command-line. The command line does not directly utilize the RESTful API but does follow the same internal flow as the RESTful API to achieve identical results.

Atomic API operations

API access through the various interfaces are thread safe. A command executed on one interface will execute completely before any command is allowed to execute on any other interface.

Most commands are bus specific and require a series of commands to operate properly. If more than one interface will be utilized at the same time, it may be necessary to ensure bus operations are performed in an atomic way.

To do this, wrap sequences with api.lock and api.unlock commands. Save the current bus at the beginning of the sequence and restore the current bus at the end.

All command-line commands internally perform this sequence of operations.

The following pseudo-code shows how to safely perform an atomic discovery on A2B0 from any API interface:

1. api.lock()
2. bus = setup.getBus()
3. setup.setBus('a2b0')
4. setup.setMode('master')
5. setup.setNetwork(...)
6. master.discover()
7. setup.setBus(bus)
8. api.unlock()

If only a single interface will be utilized at any given time (including the command-line) then it is not necessary to lock and unlock the API.

File Names

The A2B Bridge has a small internal Flash file system and an SD card file system. Wherever file names are mentioned, prefix the file name with "sf:" to access files on the Flash file system or "sd:" to access files on the SD card. File names with no prefix will default to the SD card.

API Flow

A2B Master Mode

For A2B Master operations, the API should be used as follows:

1. Optionally lock the API
    a. api.lock
2. Optionally reset to power on reset state
    a. setup.reset
3. Perform setup
    a. setup.setBus
    b. setup.setMode
    c. setup.setSigGen (optional)
    d. setup.setRoute (optional)
4. Set the network configuration
    a. setup.setNetwork
5. Attach, configure, and start communication protocol engines (optional)
    a. comm.attach
    b. comm.cmd
6. Discover the network
    a. master.discover
7. Start audio streaming on the network
    a. streaming.start
8. Unlock the API
    a. api.unlock
9. Utilize APIs post-discovery as required (I2C, routing, sig gen, streaming, etc.) locking the API if other API interfaces will be accessed simultaneously.

API Detail

Error Response Codes

In addition to the standard JSON-RPC errors, the A2B Bridge can return the following errors:

Code	Message	Detail
-100	"Generic error"	Generic error
-101	"File not found"	Network configuration file not found
-102	"File error"	Error reading network configuration
-103	"A2B network load error"	Error loading or parsing the network configuration
-104	"A2B network start error"	Error starting the network discovery

-105	"A2B network discover error"	Error discovering the network. The error message will contain additional detail.
-106	"Invalid mode selected"	Invalid mode selected in the setup.setMode API
-107	"Invalid network type"	Invalid network selected while loading the network configuration
-108	"I2C error"	Error during I2C transaction
-109	"Incompatible master SPORT configuration"	The master TDM configuration contained within the A2B configuration cannot be realized on the hardware.
-110	"Invalid reset type"	An invalid reset type was requested in the setup.reset API
-111	"Invalid frequency"	An invalid frequency was requested for a tone signal generator in the setup.setSigGen API
-112	"Invalid amplitude"	An invalid amplitude was requested for a signal generator in the setup.setSigGen API
-113	"Invalid ID"	An invalid route or signal generator ID was requested in the setup.setSigGen or setup.setRoute API
-114	"Invalid source"	An invalid route source was requested in the setup.setRoute API
-115	"Invalid destination"	An invalid route destination was requested in the setup.setRoute API
-116	"Invalid A2B bus selected"	An invalid A2B Bus was requested
-117	"Error setting bus mode"	An error occurred while setting the A2B bus mode
-118	"Invalid bus power mode"	An invalid bus power mode was requested
-119	"Error setting bus power"	An error occurred while setting the bus power mode
-120	"Invalid comm protocol"	An invalid communication protocol was requested in comm.attach()
-121	"Invalid comm version"	An invalid communication version was requested in comm.attach()
-122	"Invalid comm role"	An invalid communication role was requested in

		comm.attach()
-123	"Error attaching comm protocol engine"	An error occurred while attaching the communication protocol engine to the A2B bus
-124	"Invalid comm command"	An invalid command was sent to the communication protocol engine
-125	"Error processing comm command"	An error occurred while the protocol engine was processing a command
-126	"No comm protocol attached to bus"	No communication protocol was assigned to the A2B bus
-127	"Failed to discover mk-messtechnik optoA2B Slave"	Unable to discover the mk-messtechnik optoA2B slave. Ensure the optoA2B slave is powered and properly connected.
-128	"Already enabled"	Selected source or sink already enabled and running
-129	"Invalid channels"	Invalid number of channels selected
-130	"Invalid format"	Invalid audio format selected
-131	"Invalid clock domain"	Invalid clock domain selected
-132	"SPI error"	An error occurred during a SPI over distance transaction.
-133	"Voltage meter not supported"	The voltage meter feature is not supported by the transceiver on the A2B node.
-134	"Stream setup error"	Error starting an Ethernet RTP or VBAN stream (Industrial Bridge only)
-135	"Invalid IP address"	Invalid Ethernet IP address (Industrial Bridge only)

API API

api.lock

{"id":1,"jsonrpc":"2.0","method":"api.lock","params":{ <PARAMS> }}

Use this API to gain an exclusive lock on the protocol engine. The protocol engine must be unlocked before commands are allowed from any other interface. This is a recursive lock and may be called multiple times. An api.unlock command must be called for each call to api.lock.

The api.lock and api.unlock commands ***MUST*** be implemented if the protocol is utilized from multiple interfaces concurrently (i.e. command-line, Ethernet, Lua, or UART).

All command-line commands implement the api.lock and api.unlock operations internally for all bus operations.

Request Parameters

Parameter	JSON Type	Optional	Description
N/A	N/A	N/A	This command requires no parameters.

Response Parameters

Parameter	JSON Type	Optional	Description
N/A	N/A	N/A	An empty object will be returned upon success.

---

api.unlock

{"id":1,"jsonrpc":"2.0","method":"api.unlock","params":{ <PARAMS> }}

Use this API to release the exclusive lock on the protocol engine. If locked, the protocol engine must be unlocked from the same interface before commands are allowed from any other interface. This is a recursive lock and may be called multiple times. An api.unlock command must be called for each call to api.lock.

The api.lock and api.unlock commands ***MUST*** be implemented if the protocol is utilized from multiple interfaces concurrently (i.e. command-line, Ethernet, Lua, or UART).

All command-line commands implement the api.lock and api.unlock operations internally for all bus operations.

Request Parameters

Parameter	JSON Type	Optional	Description
N/A	N/A	N/A	This command requires no parameters.

Response Parameters

Parameter	JSON Type	Optional	Description
N/A	N/A	N/A	An empty object will be returned upon success.

Setup API

setup.setBus

{"id":1,"jsonrpc":"2.0","method":"setup.setBus","params":{ <PARAMS> }}

Subsequent A2B specific API will be applied to the selected A2B bus until this API is called again.

Request Parameters

Parameter	JSON Type	Optional	Description
bus	string	No	Set to ['A2B0', 'A2B1', 'A2B2', …]

Response Parameters

Parameter	JSON Type	Optional	Description
N/A	N/A	N/A	An empty object will be returned upon success.

NOTE: 'A2B0' is the default bus at startup

---

setup.getBus

{"id":1,"jsonrpc":"2.0","method":"setup.getBus","params":{ <PARAMS> }}

Request Parameters

Parameter	JSON Type	Optional	Description
N/A	N/A	N/A	This command requires no parameters.

Response Parameters

Parameter	JSON Type	Optional	Description
bus	string	No	Returns the currently selected bus

setup.setMode

{"id":1,"jsonrpc":"2.0","method":"setup.setMode","params":{ <PARAMS> }}

Request Parameters

<table>
<tr><th>Parameter</th><th>JSON Type</th><th>Optional</th><th>Description</th></tr>
<tr><td>mode</td><td>string</td><td>No</td><td>
<table>
<tr><th>Mode</th><th>Function</th></tr>
<tr><td>'master'</td><td>Sets the bus to master mode</td></tr>
<tr><td>'slave'</td><td>Sets the bus to slave mode.</td></tr>
<tr><td>'off'</td><td>Resets the transceiver in 'master', 'slave', and 'mk-emc' modes but does not otherwise change the underlying mode. It is useful for turning off the bus in 'master' and 'mk-emc' modes.</td></tr>
<tr><td>'mk-emc'</td><td>Sets the bus to mk-messtechnik optoA2B master mode</td></tr>
</table>
</td></tr>
</table>

Response Parameters

Parameter	JSON Type	Optional	Description
N/A	N/A	N/A	An empty object will be returned upon success.

setup.getMode

{"id":1,"jsonrpc":"2.0","method":"setup.getMode","params":{ <PARAMS> }}

Request Parameters

Parameter	JSON Type	Optional	Description
N/A	N/A	N/A	This command requires no parameters.

Response Parameters

Parameter	JSON Type	Optional	Description
mode	string	No	Returns the current mode. See setup.setMode for the full list of modes.

setup.setNetwork

{"id":1,"jsonrpc":"2.0","method":"setup.setNetwork","params":{ <PARAMS> }}

Request Parameters

Parameter	JSON Type	Optional	Description
network	string	No	Set to a Sigma Studio A2B network export XML file or Mentor "Networks" binary export BDD file. This file must have been previously copied to one of the file systems.
peripheral-pkg	string	Yes	Set to a Mentor Network Peripheral Pkg binary file. This is an optional parameter and is only required for A2B slave peripheral I2C initialization. It is ignored for Sigma Studio export files.
type	string	No	<table><tr><th>File Type</th><th>Notes</th></tr><tr><td>'ss-xml'</td><td>Sigma Studio</td></tr></table>

			<table><tr><td></td><td>XML export file</td></tr><tr><td>'mentor-bdd'</td><td>Mentor BDD export file</td></tr></table>

Response Parameters

Parameter	JSON Type	Optional	Description
N/A	N/A	N/A	An empty object will be returned upon success.

---

setup.reset

{"id":1,"jsonrpc":"2.0","method":"setup.reset","params":{ <PARAMS> }}

Request Parameters

Parameter	JSON Type	Optional	Description
type	string	No	Type of reset to perform. Valid values are ['soft', 'hard', 'routes', 'sigGen']. A 'routes' reset clears all routes. A 'sigGen' reset clears all signal generators. A 'soft' reset returns all internal state to power on reset values and resets all A2B buses. A 'hard' reset performs a hardware reset of the device. A 'hard' reset should not be used in normal operation.

Response Parameters

Parameter	JSON Type	Optional	Description
N/A	N/A	N/A	An empty object will be returned upon

			success.

---

setup.setSigGen

{"id":1,"jsonrpc":"2.0","method":"setup.setSigGen","params":{ <PARAMS> }}

Request Parameters

Parameter	JSON Type	Optional	Description
id	number	No	Signal generator ID. Valid values are [0 .. 15]
type	string	No	Signal generator type. Valid values are ['tone', 'pink', 'white', 'hex', 'off']
frequency	number	No	Sets the desired frequency for 'tone' type signal generators. Valid values are 1.0 to 24000.0 Ignored for other types.
amplitude	number	No	Sets the desired amplitude for 'tone', 'pink', and 'white' type signal generators. Valid values are -1.0 to 1.0. Ignored for other types.
value	number string	No	Sets the desired bit pattern for 'hex' type signal generators. Can be either a decimal number or 32-bit 'C' style hex string (i.e. "0xAAAA5555"). Ignored for other types.

Response Parameters

Parameter	JSON Type	Optional	Description
N/A	N/A	N/A	An empty object will be returned upon success.

setup.getSigGen

{"id":1,"jsonrpc":"2.0","method":"setup.getSigGen","params":{ <PARAMS> }}

Request Parameters

Parameter	JSON Type	Optional	Description
N/A	N/A	N/A	This command requires no parameters.

Response Parameters

Parameter	JSON Type	Optional	Description
numGens	number	No	Number of signal generators returned in the sigGens array
sigGens	array	No	Array of objects with each object describing a signal generator. The contents of each object depends on the signal generator type. See setup.setSigGen for a complete list of generator specific parameters.

setup.setWave

{"id":1,"jsonrpc":"2.0","method":"setup.setWave","params":{ <PARAMS> }}

Request Parameters

Parameter	JSON Type	Optional	Description
id	number	No	WAVE file ID. Valid values are [0]
dir	string	No	WAVE file direction. Valid values are ['src', 'sink']
action	string	No	WAVE file action. Valid values are ['on', 'off', 'domain']
domain	string	Yes	Sets the clock domain of the WAVE file.

			Required for the 'domain' action. Valid values for 'domain' are ['SYSTEM', 'A2B0', 'A2B1', 'A2B2', 'A2B3']
filename	string	Yes (once)	Set the WAVE file name. Required for the first 'on' action of any particular ID.
channels	number	Yes	Number of channels for a WAVE file 'sink'. Used for 'on' action. Defaults to 2.
bits	number	Yes	Number of bits per sample for a WAVE file 'sink'. Used for 'on' action. Valid values are [16, 32]. Defaults to 16.

Response Parameters

Parameter	JSON Type	Optional	Description
N/A	N/A	N/A	An empty object will be returned upon success.

---

setup.setRtp

setup.setVban

{"id":1,"jsonrpc":"2.0","method":"setup.setRtp","params":{ <PARAMS> }}

{"id":1,"jsonrpc":"2.0","method":"setup.setVban","params":{ <PARAMS> }}

Request Parameters

Parameter	JSON Type	Optional	Description
id	number	No	RTP / VBAN stream ID. Valid values are [0]
dir	string	No	RTP / VBAN stream direction. Valid values are ['src', 'sink']
action	string	No	RTP / VBAN stream action. Valid values are ['on', 'off', 'domain']

domain	string	No	Sets the clock domain of the RTP / VBAN stream. Required for the 'domain' action. Valid values are ['SYSTEM', 'A2B0', 'A2B1', 'A2B2', 'A2B3'].
ipAddr	string	Yes (once)	Set the RTP / VBAN stream IP address. Required for the first 'on' action.
channels	number	Yes	Number of channels for the RTP / VBAN stream. Used for 'on' action. Defaults to 2.
bits	number	Yes	Number of bits per sample for the RTP / VBAN stream. Used for 'on' action. Valid values are [16, 32]. Defaults to 16.
port	number	Yes	Port number for the RTP / VBAN stream. Used for the 'on' action. Defaults to 6970 for RTP and 6980 for VBAN.

Response Parameters

Parameter	JSON Type	Optional	Description
N/A	N/A	N/A	An empty object will be returned upon success.

---

setup.setRoute

{"id":1,"jsonrpc":"2.0","method":"setup.setRoute","params":{ <PARAMS> }}

Request Parameters

Parameter	JSON Type	Optional	Description
id	number	No	Route ID. Valid values are [0 .. 15]
channels	number	No	Number of channels to route
src	string	No	Audio source. Valid values are ['a2b', gen, 'usb', 'wav', 'off'] Legacy 'sigGen' == 'gen'

srcId	number	No	Source ID. Sets the ID of the source. Set to zero for the first source instance of a given type (i.e. zero == A2B0).
srcOffset	number	No	Source channel offset. Zero indexed.
dst	string	No	Audio destination. Valid values are ['a2b', 'usb','wav', 'off']
dstId	number	No	Destination ID. Sets the ID of the destination. Set to zero for the first destination instance of a given type.
dstOffset	number	No	Destination channel offset. Zero indexed.
attenuation	number	Yes	Attenuation in dB applied to the source signal. This number must be a positive integer (i.e. 6 = 6dB of attenuation or -6dB of gain). Default 0 dB (no attenuation).

Response Parameters

Parameter	JSON Type	Optional	Description
N/A	N/A	N/A	An empty object will be returned upon success.

---

setup.getRoute

{"id":1,"jsonrpc":"2.0","method":"setup.getRoute","params":{ <PARAMS> }}

Request Parameters

Parameter	JSON Type	Optional	Description
N/A	N/A	N/A	This command requires no parameters.

Response Parameters

Parameter	JSON Type	Optional	Description
numRoutes	number	No	Number of routes returned in the routes array

Parameter	JSON Type	Optional	Description
routes	array	No	Array of objects with each object describing a route. See setup.setRoute for a complete list of route parameters.

NOTE: It is not possible to directly route audio between clock domains. Routes between clock domains must go through an ASRC.

---

setup.setGPIO

{"id":1,"jsonrpc":"2.0","method":"setup.setGPIO","params":{ <PARAMS> }}

Request Parameters

Parameter	JSON Type	Optional	Description
mask	number	No	GPIO mask. Each bit set in 'mask' will be set to its corresponding bit in 'value'. bit 0 (0x01) == GPIO0, bit 1 (0x02) == GPIO1, etc.
value	number	No	GPIO value. Each bit set in 'mask' will have its value set to the corresponding bit in 'value'.
dir	boolean	Yes	Set to 'true' if setting the GPIO direction. Set value bits to '1' for output and '0' for input. Default is 'false'.

Response Parameters

Parameter	JSON Type	Optional	Description
N/A	N/A	N/A	An empty object will be returned upon success.

NOTE: Available GPIO depends on the underlying hardware. Not all A2B GPIO may be available on all hardware platforms.

setup.getGPIO

{"id":1,"jsonrpc":"2.0","method":"setup.getGPIO","params":{ <PARAMS> }}

Request Parameters

Parameter	JSON Type	Optional	Description
mask	number	No	GPIO mask. Each bit set in 'mask' will have its corresponding bit returned in 'value'. bit 0 (0x01) == GPIO0, bit 1 (0x02) == GPIO1, etc.

Response Parameters

Parameter	JSON Type	Optional	Description
value	Number	No	The GPIO value.

NOTE: Available GPIO depends on the underlying hardware. Not all A2B GPIO may be available on all hardware platforms.

setup.setAsrc

{"id":1,"jsonrpc":"2.0","method":"setup.setAsrc","params":{ <PARAMS> }}

Request Parameters

Parameter	JSON Type	Optional	Description
id	number	No	ASRC ID. Valid values are [0 .. 3]
enable	boolean	No	Enable the ASRC
channels	number	No	Number of ASRC channels
quality	number	No	ASRC quality. Valid values are [0 .. 10]
inDomain	string	No	ASRC input clock domain. Valid values

			are ['SYSTEM', 'A2B0', 'A2B1', 'A2B2', 'A2B3']
inFs	number	No	ASRC input sample rate.
outDomain	string	No	ASRC output clock domain. Valid values are ['SYSTEM', 'A2B0', 'A2B1', 'A2B2', 'A2B3']
outFs	number	No	ASRC output sample rate.

Response Parameters

Parameter	JSON Type	Optional	Description
N/A	N/A	N/A	An empty object will be returned upon success.

---

setup.getAsrc

{"id":1,"jsonrpc":"2.0","method":"setup.getAsrc","params":{ <PARAMS> }}

Request Parameters

Parameter	JSON Type	Optional	Description
N/A	N/A	N/A	This command requires no parameters.

Response Parameters

Parameter	JSON Type	Optional	Description
numAsrc	Number	No	Number if ASRCs
asrcs	Array	No	Array of ASRC parameters as detailed in setup.setAsrc()

Master API

master.discover

{"id":1,"jsonrpc":"2.0","method":"master.discover","params":{ <PARAMS> }}

Request Parameters

Parameter	JSON Type	Optional	Description
retry	number	Yes	Number of discovery retries. Default is zero retries if not specified.
filename	string	Yes	Discovery log filename

Response Parameters

Parameter	JSON Type	Optional	Description
numNodes	number	No	Returns the number of slave nodes discovered.
retries	number	No	Returns the number of discover retries

---

master.i2cPeripheralRead

master.i2cRead

{"id":1,"jsonrpc":"2.0","method":"master.i2cPeripheralRead","params":{ <PARAMS> }}

{"id":1,"jsonrpc":"2.0","method":"master.i2cRead","params":{ <PARAMS> }}

Request Parameters

Parameter	JSON Type	Optional	Description
nodeAddr	number	No	The A2B node address. The first slave is at address 0. The master is at address -1.
i2cAddr	number	No	The 7-bit I2C peripheral address of the

			device on the slave node. This parameter is only valid for I2C peripheral reads on slave nodes.
nRead	number	No	The number of bytes to read. This value can be zero.

Response Parameters

Parameter	JSON Type	Optional	Description
bytes	number array	No	Array of bytes read.

NOTE: This API requires a successful A2B discovery to function properly for slave node reads.

---

master.i2cPeripheralWriteRead

master.i2cWriteRead

{"id":1,"jsonrpc":"2.0","method":"master.i2cPeripheralWriteRead","params":{ <PARAMS> }}

{"id":1,"jsonrpc":"2.0","method":"master.i2cWriteRead","params":{ <PARAMS> }}

Request Parameters

Parameter	JSON Type	Optional	Description
nodeAddr	number	No	The A2B node address. The first slave is at address 0. The master is at address -1.
i2cAddr	number	No	The 7-bit I2C peripheral address of the device on the slave node. This parameter is only valid for I2C peripheral write/reads on slave nodes.
wBuf	number array	No	The array of bytes to write
nRead	number	No	The number of bytes to read. This value can be zero.

Response Parameters

Parameter	JSON Type	Optional	Description
bytes	number array	No	Array of bytes read.

NOTE: This API requires a successful A2B discovery to function properly for slave node write/reads.

---

master.cmdlistPlay

{"id":1,"jsonrpc":"2.0","method":"master.cmdlistPlay","params":{ <PARAMS> }}

Request Parameters

Parameter	JSON Type	Optional	Description
filename	string	No	The filename of the XML command list to play.

Response Parameters

Parameter	JSON Type	Optional	Description
N/A	N/A	N/A	An empty object will be returned upon success.

NOTE: This API can interfere with the operation of the A2B Bridge if the command list overwrites configuration values of internal components. Use with caution!

---

master.vmtr

{"id":1,"jsonrpc":"2.0","method":"master.vmtr","params":{ <PARAMS> }}

Request Parameters

Parameter	JSON Type	Optional	Description
nodeAddr	number	Yes	Node from which to read the voltage meter (vmtr) values. The first slave is at address 0. The master is at address -1 (default).

Response Parameters

Parameter	JSON Type	Optional	Description
voltages	array of numbers	No	Returns node voltage levels in the order indicated below: 0 VIN – GND 1 VBUS – GND 2 IOVDD – GND 3 TRXVDD – GND 4 DVDD – GND 5 ISENSEN – VSENSEN 6 VBUS – ISENSEP

NOTE: This feature only works on AD243x nodes. See the Hardware Reference manual for additional details on the voltage levels.

Streaming API

streaming.start

{"id":1,"jsonrpc":"2.0","method":"streaming.start","params":{ <PARAMS> }}

Request Parameters

Parameter	JSON Type	Optional	Description
all	boolean	Yes	Enables global streaming if true. Otherwise, enables streaming only on the current A2B bus if omitted (see setup.setBus).

Response Parameters

Parameter	JSON Type	Optional	Description
N/A	N/A	N/A	An empty object will be returned upon success.

NOTE: Audio does not stream to or from an A2B bus until streaming is started on that bus. Both global streaming and bus streaming must be enabled for audio to stream on the selected A2B bus.

---

streaming.stop

{"id":1,"jsonrpc":"2.0","method":"streaming.stop","params":{ <PARAMS> }}

Request Parameters

Parameter	JSON Type	Optional	Description
all	boolean	Yes	Disables global streaming if true. Otherwise, disables streaming only on the current A2B bus if omitted (see setup.setBus).

Response Parameters

Parameter	JSON Type	Optional	Description
N/A	N/A	N/A	An empty object will be returned upon success.

---

streaming.getPeaks

{"id":1,"jsonrpc":"2.0","method":"streaming.getPeaks","params":{ <PARAMS> }}

Request Parameters

Parameter	JSON Type	Optional	Description
N/A	N/A	N/A	This command requires no parameters.

Response Parameters

Parameter	JSON Type	Optional	Description
peaks	array of numbers	No	Returns the peak levels of the received audio on the A2B bus. The size of the array depends on the audio TDM settings. In master mode this will always be 32. In slave mode, the size will vary depending on the audio TDM settings. The bit-width of the array depends on the audio TDM settings. In master mode, this will always be signed 32-bit values. In slave mode, the bit-width will vary depending on the audio TDM settings. The peak levels are automatically reset each time this API is called.

---

streaming.getStatus

{"id":1,"jsonrpc":"2.0","method":"streaming.getStatus","params":{ <PARAMS> }}

Request Parameters

Parameter	JSON Type	Optional	Description
N/A	N/A	N/A	This command requires no parameters.

Response Parameters

Parameter	JSON Type	Optional	Description
bus	boolean	No	Returns the streaming status of the currently selected bus.
all	boolean	No	Returns the global streaming status

COMM API

comm.attach

{"id":1,"jsonrpc":"2.0","method":"comm.attach","params":{ <PARAMS> }}

Attach a communication protocol engine to an A2B bus. Any previous communication protocols will be automatically detached.

Request Parameters

Parameter	JSON Type	Optional	Description
protocol	string	No	Communication protocol identifier. See available COMM protocols at the end of the document.
version	string	No	Communication protocol version. See available COMM protocol versions at the end of the document.
role	string	No	Communication protocol role. See available COMM protocol roles at the end of the document.

Response Parameters

Parameter	JSON Type	Optional	Description
N/A	N/A	N/A	An empty object will be returned upon success.

---

comm.detach

{"id":1,"jsonrpc":"2.0","method":"comm.detach","params":{ <PARAMS> }}

Detaches a communication protocol engine from an A2B bus. All protocol support is removed once a protocol engine is detached from the bus.

Request Parameters

Parameter	JSON Type	Optional	Description
N/A	N/A	N/A	This command requires no parameters.

Response Parameters

Parameter	JSON Type	Optional	Description
N/A	N/A	N/A	An empty object will be returned upon success.

---

comm.cmd

{"id":1,"jsonrpc":"2.0","method":"comm.cmd","params":{ <PARAMS> }}

Sends a command to the attached communication protocol engine.

Request Parameters

Parameter	JSON Type	Optional	Description
cmd	string	No	Command string. Valid commands depend on the protocol engine. See protocol engine specific detail at the end of this document.
params	object	No	Object containing command parameters. The object contents depend on the specific command. See protocol engine specific detail at the end of this document.

Response Parameters

Parameter	JSON Type	Optional	Description
N/A	N/A	N/A	The return object contents depend on the specific command. See protocol engine specific detail at the end of this document.

Util API

util.batch

{"id":1,"jsonrpc":"2.0","method":"util.batch","params":{ <PARAMS> }}

Request Parameters

Parameter	JSON Type	Optional	Description
cmds	array of objects	No	Contains an array of fully formatted JSON-RPC subcommands. Each subcommand can optionally contain a 'delay' parameter, in milliseconds, to tightly control timing. The delay is relative to the beginning of the previous command. The 'jsonrpc' parameter can be omitted from the subcommands, but a valid 'id' must be present.

Example Request

The following request will issue an invalid subcommand then stream two 20mS cycles of audio on the currently selected A2B bus.

```
{
    "jsonrpc": "2.0",  "method": "util.batch", "id": 1,
    "params": {
        "cmds": [
            { "delay": 0,  "cmd": {"id": -1, "method": "bad.command" } },
            { "delay": 0,  "cmd": {"id": 0, "method": "streaming.start" } },
            { "delay": 20, "cmd": {"id": 1, "method": "streaming.stop" } },
            { "delay": 10, "cmd": {"id": 2, "method": "streaming.start" } },
            { "delay": 20, "cmd": {"id": 3, "method": "streaming.stop" } }
        ]
    }
}
```

Response Parameters

Parameter	JSON Type	Optional	Description
resps	array of objects	No	Contains an array of fully formatted JSON-RPC responses. Each response contains a beginning and ending timestamp, in milliseconds, associated with the subcommand execution relative to the beginning of batch

			execution.

Example Response

The following is the response to the example request above:

```
{
    "jsonrpc": "2.0", "id": 1, "result": {
        "resps": [
            { "begin": 0, "end": 1,
              "resp": {
                "jsonrpc": "2.0", "id": -1,
                "error": { "code": -32601, "message": "method not found" }
              }
            },
            { "begin": 1, "end": 1,
              "resp": {
                "jsonrpc": "2.0", "id": 0,
                "result": {}
              }
            },
            { "begin": 21, "end": 21,
              "resp": {
                "jsonrpc": "2.0", "id": 1,
                "result": {}
              }
            },
            { "begin": 31, "end": 31,
              "resp": {
                "jsonrpc": "2.0", "id": 2,
                "result": {}
              }
            },
            { "begin": 51, "end": 51,
              "resp": {
                "jsonrpc": "2.0", "id": 3,
                "result": {}
              }
            }
        ]
    }
}
```

Command Line API

The serial command line provides convenient access to a number features available through the RESTful API described in this document.

Commands placed into a text file named 'shell.cmd' on the internal Flash filesystem ('sf:shell.cmd') will be executed at startup.

There should be one command per line. Blank lines are ignored as are comment lines beginning with a '#' character.

The 'run' command can be used to execute command scripts as well.

Type 'help' for a full list of commands. Type 'help <command>' at the command line for detailed usage. A list of useful commands with references to the related API are included below:

Command	Nearest RESTful API	Notes
discover	master.discover()	Used to discover an A2B network on an A2B bus.
route	setup.setRoute() setup.getRoute()	Used to get/set audio route parameters
gen	setup.setSigGen() setup.getSigGen()	Used to get/set signal generator parameters
reset	setup.reset()	Performs various types of resets
mode	setup.setMode() setup.getMode()	Used to get/set the mode of an A2B bus
cmdlist	master.cmdlistPlay()	Plays a raw ADI command list.
comm	COMM Module API	Provides access to the COMM API
streaming	streaming.start() streaming.stop()	Streaming start/stop and status
gpio	setup.setGPIO() setup.getGPIO()	Used to manipulate A2B GPIO
i2c	master.i2cRead() master.i2cPeripheralRead() master.i2cWriteRead()	Used to perform A2B I2C operations.

	master.i2cPeripheralWriteRead()	
wav	setup.setWave()	Used to get/set WAVE file src/sink parameters
rtp	setup.setRtp()	Used to set/get RTP src/sink parameters
vban	setup.setVban()	Used to set/get VBAN src/sink parameters
vmtr	master.vmtr()	Get AD243x voltage meter readings
peaks	streaming.getPeaks()	Get A2B audio peak levels

Lua Scripting API

The advanced Lua scripting engine provides access to the API via Lua built-in module functions. Most Lua API function parameters mirror those found in the RESTful API. Exceptions are noted below, otherwise refer to the RESTful API for details.

All Lua API modules support a brief 'help()' method to assist script development.

'api' module

Require

api=require('api')

Help

api.help()

Functions

lock()
unlock()

'master' module

Require

master=require('master')

Help

master.help()

Functions

discover([retry])
i2cPeripheralRead(node,i2cAddr,nRead)
i2cPeripheralWriteRead(nodeAddr,i2cAddr,wBuf[,nRead])
i2cRead(nodeAddr,nRead)
i2cWriteRead(nodeAddr,wBuf[,nRead])
cmdlistPlay(filename)
vmtr(node)

'setup' module

Require

setup=require('setup')

Help

setup.help()

Functions

setBus(bus)
ok,bus=getBus()
setMode(mode)
ok,mode=getMode()
setNetwork(network[,type][,peripheral-pkg])
setSigGen(id,type[,typeargs])
ok,genTabl=getSigGen()
reset(type)
setRoute(id,src,srcId,srcOffset,dst,dstId,dstOffset,channels[,attenuation])
ok,routeTable=getRoute()

setWave(dir,id,action[,actionargs])
setRtp(dir,id,action[,actionargs])
setVban(dir,id,action[,actionargs])
ok,msg=setAsrc(idx,'enable'/'disable'[,in_domain][,in_fs][,out_domain][,out_fs][,channels]
ok,asrcTable=getAsrc()
setGPIO(mask,value[,'dir'])
ok,value=getGPIO(mask)

'comm' module

Require

comm = require('comm')

Help

comm.help()

Functions

attach(protocol,version,role)
detach()
cmd(cmd,params)

'streaming' module

Require

streaming = require('streaming')

Help

streaming.help()

Functions

start(['all'])
stop(['all'])
ok,peaksTable=getPeaks()
ok,bus,all=getStatus()

'freertos' module

Require

freertos = require('freertos')

Help

freertos.help()

Functions

sleep(sec) - Suspends execution for 'sec' seconds (real number, i.e. 0.1 = 100mS)
delay(sec) - Suspends execution for 'sec' seconds (real number, i.e. 0.1 = 100mS)
time() - Returns time elapsed since power on in seconds
clock() - Returns time elapsed since power on in seconds

'term' module

The 'term' module can be used to build basic terminal based interactive user interfaces.

Require

term = require('term')

Help

Refer to the following for a detailed API description of the term module:

http://www.eluaproject.net/doc/master/en_refman_gen_term.html

COMM Protocol Engines

Contact Embedded Gadgets, LLC or Flextech Solutions, LLC for communication protocol detail.

"mk-emc" Mode

Contact Embedded Gadgets, LLC or Flextech Solutions, LLC for details interfacing with MK-Messtechnik optoA2B EMC interface boxes..