Qualcomm Technologies International, Ltd.

Qualcomm BlueCore Audio API

Design Guide

80-CE527-1 Rev. GA

October 18, 2017

Qualcomm Technologies International, Ltd. (formerly known as Cambridge Silicon Radio Limited) is a company registered in England and Wales with a registered office at: Churchill House, Cambridge Business Park, Cowley Road, Cambridge, CB4 0WZ, United Kingdom. Registered Number: 3665875 | VAT number: GB787433096

Revision history

Revision	Date	Description
1	AUG 2011	Original publication of this document. Alternative document number CS-00209064-DD.
2	DEC 2012	Updates for I²S phase shift
3	DEC 2013	Migrated to CSR™ new type
4	JUN 2014	Unified-27d updates (24-bit audio and sidetone enhancements) and updates for running I²S interface without occasional sample shift
5	AUG 2014	Updates related to 1-sample shift on I²S interface. Settings for `i2_tx/rx_start_sample` added, see .
6	OCT 2014	Updates to `stream_get_source` and `stream_get_sink` , see and stream_get_sink.
7	MAY 2014	Unified-28 updates (PSKEY_CODEC_PIO_SETUP_TIME and `STREAM_AUDIO_SAMPLE_SIZE` added)
8	SEP 2016	Removed `STREAM_PCM_RX_RATE_DELAY`. Added support for 96k and 88.2k on SPDIF Tx Added support for new stream keys: ■ STREAM_CODEC_ADC_DATA_SOURCE_POINT ■ STREAM_DIGITAL_MIC_DATA_SOURCE_POINT ■ STREAM_CODEC_G722_FILTER_ENABLE ■ STREAM_ CODEC _G722_FIR_ENABLE ■ STREAM_DIGITAL_MIC_G722_FILTER_ENABLE ■ STREAM_DIGITAL_MIC_G722_FIR_ENABLE Updated to QTI style guidelines.
9	APR 2017	Description of `stream_sidetone_en` and `StreamEnableSidetone()` added. Added to the Content Management System.
GA	OCT 2017	Document Reference Number updated to use Agile number. No change to technical content.

Contents

Tables

Figures

1 Audio API - overview

This document describes the audio API for devices running BlueCore 5 or later firmware.

NOTE The BlueCore 7820 vA12 IC uses a restricted implementation of the API described in this document. For more information, see the *BlueCore BC7820 Audio API Specification*.

For information on tuning the FM receiver and the FM transmitter, see the *BlueCore FM API*.

For information on BlueCore commands, see the *HQ and BCCMD Commands and Protocols*.

The audio API provides an interface based on the underlying BlueCore stream-based architecture.

Using this API, applications reserve resources by obtaining Source IDs (for input) and Sink IDs (for output). Source and Sink IDs are then configured and connected together to form transforms (a path along which data flows from the Source ID to the Sink ID). The application disconnects a transform and releases the associated resources when it is not required.

In addition, the related Source IDs or Sink IDs can be synchronized to ensure that they are simultaneously enabled. The API supports aliasing of sinks to allow a single input connection to two individual outputs.

The new audio API allows deprecation of several older BlueCore commands. These commands are included in this document and are marked as deprecated. Each affected command includes a brief description about the new command to use instead.

BlueCore 26c and later versions of HCI firmware include the DSPManager feature on ICs with DSP. The audio APIs enhanced to support the DSPManager functionality are:

- stream_get_source
- stream_get_sink
- stream_close_source
- stream_close_sink
- stream_configure
- stream_sync_sid
- stream_connect
- stream_transform_disconnect
- enable_sco_streams

For more information about the API enhancements, see the *Qualcomm BlueCore DSPManager Specification*.

2 BlueCore commands

Current audio BlueCore commands are:

- stream_get_source
- stream_get_sink
- stream_close_source
- stream_close_sink
- stream_configure
- stream_alias_sink
- stream_sync_sid
- stream_connect
- stream_transform_disconnect
- map_sco_audio
- enable_sco_streams
- map_sco_pcm
- mic_bias_ctrl

Deprecated audio BlueCore commands are:

- codec_input_gain
- codec_output_gain
- pcm_attenuation
- pcm2_attenuation
- pcm_clock_rate
- pcm_sync_rate
- pcm_slots_per_frame
- pcm_config32
- digital_audio_rate
- digital_audio_config

NOTE QTIL does not recommend using deprecated commands. The deprecated BlueCore commands are included to help with the transition to use of the new `stream_configure` command.

The BlueCore controller's response indicates success or failure through the `Status` field of the header. A value of `BCCMDPDU_STAT_OK` (`0x0000`) indicates success and any other value indicates failure. For more information on `Status` field values, see the *HQ and BCCMD Commands and Protocols*.

Return details of the commands that return additional information in the message payload are described in the relevant command description.

2.1 Current BlueCore commands

The API BlueCore commands that are currently used to manage audio streams are:

- stream_get_source
- stream_get_sink
- stream_close_source
- stream_close_sink
- stream_configure
- stream_alias_sink
- stream_sync_sid
- stream_connect
- stream_transform_disconnect
- map_sco_audio
- enable_sco_streams
- map_sco_pcm
- mic_bias_ctrl
- stream_sidetone_en

2.1.1 stream_get_source

Varid	Type	Permissions	Intrinsic Permissions
`0x505a`	Complex	WO	WO

This command requests the specified source resource to be reserved.

Message

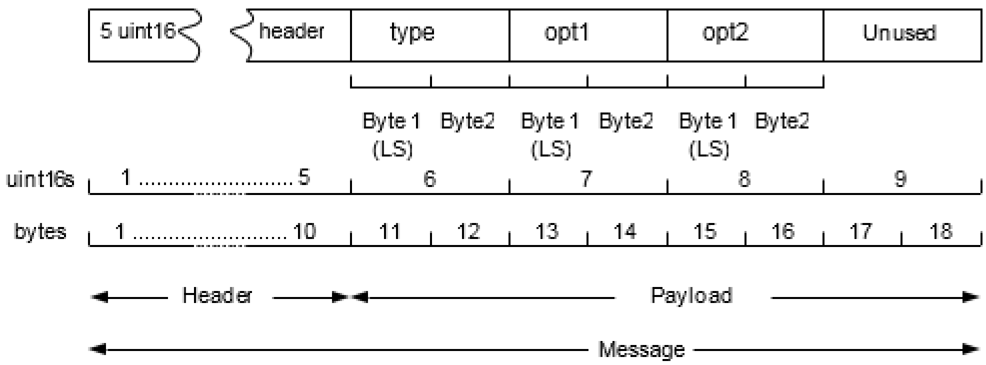

Figure 2-1 stream_get_source command structure

The `type` parameter specifies the resource requested. Table 2-1 shows the type of source specified by each value of the type argument.

Table 2-1 stream_get_source type arguments

Source	Type
PCM	`0x0001`
I^2S	`0x0002`
Codec	`0x0003`
FM	`0x0004`
SPDIF	`0x0005`
Digital mic	`0x0006`
SCO (BlueCore 26 or later firmware)	`0x0009`

Table 2-2 describes the `opt1` and `opt2` arguments for each type of source.

If successful this command returns a source identifier that can be used in subsequent commands.

If the command attempts to reserve a resource that is already reserved, the request succeeds and responds with the original request's Source ID.

An attempt to reserve a resource may fail due to:

- The resource does not exist.
- The resource cannot be reserved because another resource that shares some aspect of its hardware has already been reserved.
- There are insufficient internal resources to support the requested resource.

Table 2-2 stream_get_source and stream_get_sink opt1 and opt2 arguments

Source	opt1	opt2
PCM	0: PCM1 1: PCM2	0: first slot, 1: second slot, 2: third slot, 3: fourth slot. The number of available PCM slots can range from 1 to 4. This can be configured with the command stream_configure or a PS Key.
I²S	0: I²S1 1: I²S2	0: AUDIO_CHANNEL_A, A, or L channel 1: AUDIO_CHANNEL_B, B, or R channel
ADC and DAC	0: ADC or DAC (only one instance)	0: AUDIO_CHANNEL_A, A, or L channel 1: AUDIO_CHANNEL_B, B, or R channel 2: AUDIO_CHANNEL_A_AND_B, A, or L channel output on both output channels, only for stream_get_sink and stereo DAC
FM	0: FM (only one instance)	0: AUDIO_CHANNEL_A, A or L channel 1: AUDIO_CHANNEL_B, B or R channel
SPDIF	0: SPDIF (only one instance)	0: SPDIF_CHANNEL_A, A or L channel 1: SPDIF_CHANNEL_B, B or R channel 3: SPDIF_CHANNEL_A_B_INTERLEAVED A and B (L and R) channels interleaved - left and right channel samples alternatingly (L0, R0, L1, R1, …)
Digital microphone	0: first mic 1: second mic 2: third mic Dual (two channel) digital microphones count as a single instance	0: AUDIO_CHANNEL_A, A or L channel 1: AUDIO_CHANNEL_B, B or R channel
SCO	The HCI handle of the SCO connection.	(Ignored)

Response message

The corresponding `GETRESP` signals success with `BCCMDPDU_STAT_OK` (`0x0000`) in the `Status` field and returns a Source ID for the requested resource in the first two bytes of the payload, LS byte first.

A non-zero value in the `Status` field indicates failure. In the case of failure, the first two bytes of the payload are undefined.

The Source ID returned is an arbitrary value that is used to refer to the resource in the subsequent BlueCore commands.

2.1.2 stream_get_sink

Varid	Type	Permissions	Intrinsic Permissions
`0x505b`	Complex	WO	WO

This command requests the specified sink resource to be reserved.

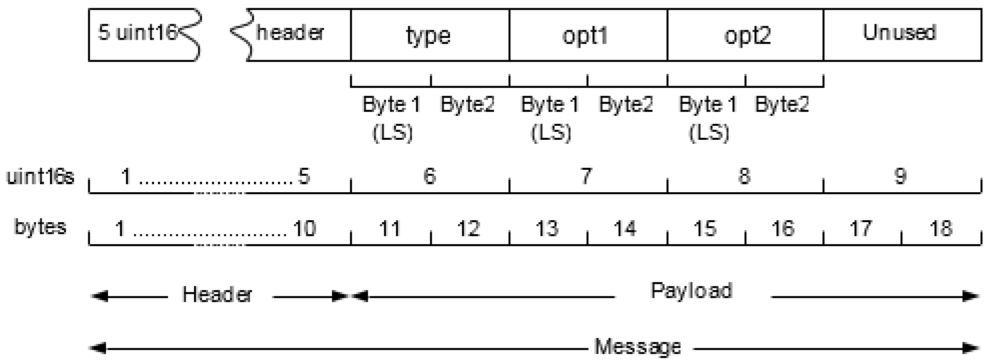

Figure 2-2 stream_get_sink command structure

The `type` parameter specifies the type of resource requested. Table 2-3 shows the type of sink resource specified by the `type` argument.

Table 2-3 stream_get_sink type arguments

Source	type
PCM	`0x0001`
I^2S	`0x0002`
Codec	`0x0003`
FM	`0x0004`
SPDIF	`0x0005`
SCO (BlueCore 26 or later firmware)	`0x0009`

stream_get_source describes the `opt1` and `opt2` arguments for each type of sink.

If successful this command returns a sink identifier that can be used in subsequent commands.

The codec, FM and digital mic channels are logically grouped into pairs, with each pair forming an instance. Therefore, both codec channel A and codec channel B are part of the first instance.

Channel 2 (Channel A and B) is a special case that requests both the 0(A) and 1(B) output channels of a stereo codec instance. If the output channel allocates a sink ID used in a transform, the output of the transform is routed to both of the channels of the codec.

If Channel A and B (Channel 2) is requested for a given codec instance, it is not possible to request Channel A or Channel B without first releasing the sink ID allocated by the Channel A and B request.

If the command attempts to reserve a resource that is already reserved, the request succeeds and responds with the original request's Sink ID.

An attempt to reserve a resource may fail due to one or more of:

- The resource does not exist.
- The resource cannot be reserved because another resource that shares some aspect of its hardware has already been reserved.
- There are insufficient internal resources to support the requested resource.

Response message

The corresponding GETRESP signals success with BCCMDPDU_STAT_OK (0x0000) in the Status field and returns a Sink ID for the requested resource in the first two bytes of the payload, LS byte first.

A non-zero value in the Status field indicates failure. In the case of failure, the first two bytes of the payload are undefined.

The Sink ID returned is an arbitrary value that is used to refer to the resource in the subsequent BlueCore commands.

2.1.3 stream_close_source

Varid	Type	Permissions	Intrinsic Permissions
0x486b	uint16	WO	WO

This command releases the resource currently associated with the specified Source ID. If the resource is currently connected via a transform, the transform is automatically disconnected as part of the command.

When released, the Source ID associated with the resource is invalid and should be discarded. The command fails on specifying an unrecognized Source ID.

Response message

The corresponding GETRESP signals success with BCCMDPDU_STAT_OK (0x0000) in the Status field. A non-zero value in the Status field indicates failure.

2.1.4 stream_close_sink

Varid	Type	Permissions	Intrinsic Permissions
0x486c	uint16	WO	WO

This command releases the resource currently associated with the specified Sink ID. If the resource is currently connected via a transform, the transform is automatically disconnected as part of the command. When released, the Sink ID associated with the resource is invalid and should be discarded. The command fails on specifying an unrecognized Sink ID.

Response message

The corresponding GETRESP signals success with BCCMDPDU_STAT_OK (0x0000) in the Status field. A non-zero value in the Status field indicates failure.

2.1.5 stream_configure

Varid	Type	Permissions	Intrinsic Permissions
0x505c	Complex	WO	WO

This command configures a single property of the specified Source or Sink ID.

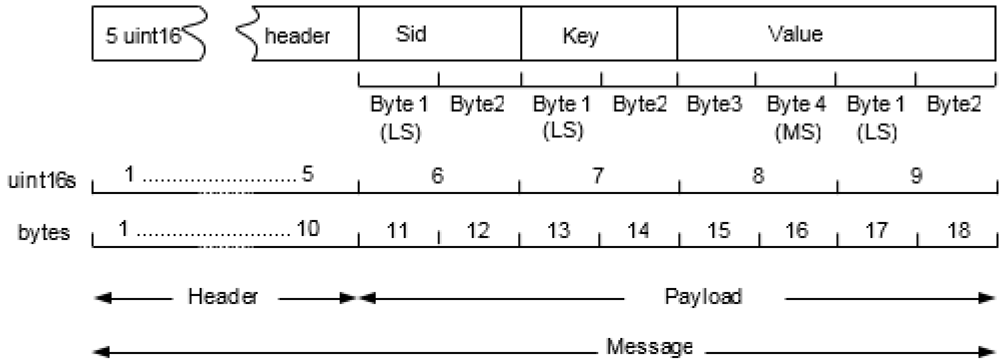

Figure 2-3 stream_configure command structure

The `Sid` parameter specifies the Source or Sink ID to be configured. The `Key` parameter specifies the property of the Source ID or Sink ID to be configured. The `Value` parameter specifies the data value to be assigned to the key.

Table 2-4 to Table 2-10 show the supported keys for each hardware type.

Table 2-4 Values supported by PCM

PCM	Key	Supported Values
PCM sync rate	`0x0100`	Sync rate in Hz
PCM master clock rate	`0x0101`	0: Autogenerated (see Appendix PCM master clock rate derivation) or Master clock rate in Hz
PCM master mode	`0x0102`	0: Slave 1: Master
PCM slot count	`0x0103`	0: Derived from the clock and the sync rates. See see Appendix PCM master clock rate derivation. 1 – 4: Number of slots.
PCM Manch mode enable (Manchester mode enable)	`0x0104`	0: Disable 1: Enable
PCM short sync enable	`0x0105`	0: Disable 1: Enable
PCM Manchester slave mode	`0x0106`	0: Disable 1: Enable
PCM sign extend mode	`0x0107`	0: Disable 1: Enable
PCM LSB first mode	`0x0108`	0: Disable 1: Enable
PCM TX tri-state mode	`0x0109`	0: Disable 1: Enable
PCM TX tri-state rising edge mode	`0x010a`	0: Disable 1: Enable

Table 2-4 Values supported by PCM (cont.)

PCM	Key	Supported Values
PCM sync suppress enable	`0x010b`	0: Disable 1: Enable
PCM GCI mode	`0x010c`	0: Disable 1: Enable
PCM mute enable	`0x010d`	0: Disable 1: Enable
PCM long length sync	`0x010e`	0: Disable 1: Enable
PCM sample rising edge	`0x010f`	0: Disable 1: Enable
PCM sample format	`0x0114`	This key will be deprecated soon. Please use Audio sample size (`0x0701`) stream key that supports 24-bit format as well. Selects one of the following formats of audio sample on PCM interface(all channels, both directions): 0: 13 bits in a 16-bit slot 1: 16 bits in a 16-bit slot 2: 8 bits in a 16-bit slot 3: 8 bits in an 8-bit slot
PCM Manchester mode RX offset	`0x0115`	0 – 3
PCM audio gain	`0x0116`	0 – 7

Table 2-5 Values supported by I²S

I²S	Key	Supported Values
I²S sync rate	`0x0200`	Sync rate in Hz
I²S master clock rate	`0x0201`	0: Auto-generated (see Appendix I²S Master clock rate derivation) or Master clock rate in Hz
I²S master mode	`0x0202`	0: Slave 1: Master
I²S justify format	`0x0203`	0: Left justified 1: Right justified
I²S left justify delay	`0x0204`	If using left-justified format: 0: MSB of SD data occurs in the first SCLK period following the WS transition 1: MSB of SD data occurs in the second SCLK period
I²S channel polarity	`0x0205`	0: SD data is left channel when WS is high 1: SD data is right channel when WS is high

Table 2-5 Values supported by I²S (cont.)

I²S	Key	Supported Values
I²S audio attenuation enable	`0x0206`	Enables/disables the attenuation applied to incoming (into BlueCore) audio samples in case the PCM format is set to 20/24 bit per sample. Used in conjunction with "I²S audio attenuation". 0: Disable 1: Enable
I²S audio attenuation	`0x0207`	0 – 15 (in 6 dB steps)
I²S justify resolution	`0x0208`	0: 16-bit 1: 20-bit 2: 24-bit
I²S crop enable	`0x0209`	Used to select b/w rounding and cropping (truncation) in I²S RX. If SD_IN carries 24/32 bits per sample, but I²S interface is configured for 16 bits per sample only, then crop enable decides whether the I²S interface will round or truncate incoming 24/32 bits to 16 bits. Additionally, it must be enabled if in 16 bit per sample mode when CLK rate = 32*sample rate. 0: Disable Cropping(or select rounding) 1: Enable Cropping
I²S bits per sample (4)	`0x020a`	16 20 24 See Appendix I²S Master clock rate derivation for more information.
I²S TX start sample (4)	`0x020b`	Selects when to start sampling in TX direction. 0: During low WS phase 1: During high WS phase
I²S RX start sample (4)	`0x020c`	Selects when to start sampling in RX direction. 0: During low WS phase 1: During high WS phase

Table 2-6 Values supported by codec

Codec	Key	Supported Values
Codec input sample rate (1)	`0x0300`	8000 11025 12000(2) 16000 22050 24000 32000 40000(2) 44100 48000(2)
Codec output sample rate (1)	`0x0301`	8000 11025 12000 16000 22050 24000 32000 40000(2) 44100 48000 96000
Codec input gain	`0x0302`	0 – 22
Codec output gain	`0x0303`	0 – 22

Table 2-6 Values supported by codec (cont.)

Codec	Key	Supported Values
Codec raw input gain	`0x0304`	Bit[15] – Select fine Digital gain If Bit[15] = 1 Bits [8:0] – Digital Gain in steps of -30dB 1: Max attenuation 31: Min attenuation 32: Unity 33: Min gain 511: Max gain If Bit[15] = 0 Bits [3:0] – Digital Gain in legacy mode 8: Max attenuation 15: Min attenuation 0: Unity 1: Min gain 7: Max gain Bits [18:16] Analog gain: 0: max attenuation 5: unity 7: max gain
Codec raw output gain	`0x0305`	Bit[15] – Select fine Digital gain If Bit[15] = 1 Bits [8:0] – Digital Gain in steps of -30dB 1: Max attenuation 31: Min attenuation 32: Unity 33: Min gain 511: Max gain If Bit[15] = 0 Bits [3:0] – Digital Gain in legacy mode 8: Max attenuation 15: Min attenuation 0: Unity 1: Min gain 7: Max gain Bits [18:16] Analog gain: 0: Max attenuation 5: Unity 7: Max gain
Codec output gain boost enable	`0x0306`	0 (disable), 1 (enable)

Table 2-6 Values supported by codec (cont.)

Codec	Key	Supported Values
Codec sidetone gain	0x0307	Gain applied to all the sidetone channels. For BlueCore5 and prior: 0 – 7 For BlueCore6 : 0-9 For BlueCore7 and later: 0-15, giving the following respective dB gains: 0: -32.6 1: -30.1 2: -26.6 3: -24.1 4: -20.6 5: -18.1 6: -14.5 7: -12 8: -8.5 9: -6.0 10: -2.5 11: 0.0 12: +3.5 13: +6.0 14: +9.5 15: +12.0
Codec sidetone enable	0x0308	0: Disable 1: Enable
Codec sidetone source point (5)	0x30f	Source point for sidetone data at ADC. 0x00: ADC data is taken before digital gain. 0x01: ADC data is taken after digital gain.
Codec sidetone injection point (5)	0x310	Injection point for sidetone data at DAC. 0x00: Sidetone data is inserted at interpolation stage in DAC 0x01: Sidetone data is inserted at gain stage in DAC
Codec sidetone source mask (5)	0x0311	Mask that selects at most 2 ADC/MIC sources whose sum will be used as sidetone source for a particular DAC channel. 0x00: No sidetone source. As good as sidetone is not enabled. 0x01 : Channel A(ADC A/ DMIC A) is the sidetone source 0x02 : Channel B(ADC B/ DMIC B) is the sidetone source 0x03: (Channel A + Channel B) is the sidetone source -- -- 0x21: (Channel A + Channel F) is the sidetone source

Table 2-6 Values supported by codec (cont.)

Codec	Key	Supported Values
Codec individual sidetone gain (5)	`0x0312`	Gain of a particular sidetone channel. In contrast, when stream key "Codec sidetone gain (`0x0307`)" is used, gain of all sidetone channels is changed simultaneously. dB gain table: 0: -32.6 1: -30.1 2: -26.6 3: -24.1 4: -20.6 5: -18.1 6: -14.5 7: -12 8: -8.5 9: -6.0 10: -2.5 11: 0.0 12: +3.5 13: +6.0 14: +9.5 15: +12.0
Codec individual sidetone enable (5)	`0x0313`	Enable/disable sidetone signal for a particular DAC channel. In contrast, stream key "Codec sidetone enable (`0x0308`) " is used to enable/disable sidetone signal for all DAC channels. 0: Disable 1: Enable
Codec ADC data source point	`0x0314`	ADC data source selection. `0x00`: ADC data is taken from IIR filter out `0x01`: ADC data is taken from Digital gain filter out
Codec sidetone invert (5)	`0x316`	Invert sidetone phase before injecting into DAC chain. 0: Disable (Do not invert) 1: Enable (Invert)
Codec G722 filter enable	`0x317`	Enables optional G722 filter that improves noise performance. 0: Disable 1: Enable
Codec G722 FIR filter enable	`0x318`	Enables optional FIR filter inside G722 filter that droops the response slightly. 0: Disable 1: Enable
Codec mic input gain enable	`0x0309`	0: Disable 1: Enable

Table 2-6 Values supported by codec (cont.)

Codec	Key	Supported Values
Codec low power output stage	`0x030a`	0: Disable 1: Enable
Codec quality mode (2)	`0x030b`	0: Telephony 1: Normal 2: High 3: Bypass in Amp
Codec output interpolation filter mode (2)	`0x030c`	0: Long FIR mode, not available at 96 kHz 1: Short FIR mode 2: Narrow FIR mode
Codec output power mode (2)	`0x030d`	0: 16 Ω, normal power 1: 32 Ω, normal power 2: 32 Ω, low power
Codec sidetone source (4)	`0x030e`	0: High-Quality ADC A/B or Digital Mic instance 0 DMIC channel A goes into DAC A and DMIC channel B goes into DAC B. 1: Digital mic instance 1 DMIC channel C goes into DAC A and DMIC channel D Goes into DAC B. 2: Digital mic instance 2 DMIC channel E goes into DAC A and DMIC channel F goes into DAC B.

Table 2-7 Values supported by FM

FM	Key	Supported Values
FM input sample rate	`0x0400`	8000 11025 12000 16000 22050 24000 32000 40000 44100 48000
FM output sample rate	`0x0401`	8000 11025 12000 16000 22050 24000 32000 40000 44100 48000
FM input gain	`0x0402`	0 – 15 (3): 8: Max attenuation 15: Min attenuation 0: Unity 1: Min gain 7: Max gain
FM output gain	`0x0403`	0 – 15 (3): 8: Max attenuation 15: Min attenuation 0: Unity 1: Min gain 7: Max gain

Table 2-8 Values supported by SPDIF

SPDIF	Key	Supported Values
SPDIF output sample rate	`0x0500`	32000 44100 48000 88200 96000(Tx does not work in 24-bit mode)
SPDIF input channel status report mode	`0x0501`	The key allows Host to configure which channel status will be sent to the DSP in a message. 0: No Channel status 1: Channel status A 2: Channel status B 3: Both channels. (Not supported)
SPDIF output channel status duplicate enable	`0x0502`	0: Channel B carries its own channel status 1: Channel A channel status is duplicated on channel B
SPDIF output channel status word	`0x0503`	The 192-bit output channels status is divided into 12 words of 16 bits each. Each word can be individually set. Bits [31:16]: channel status word index: ■ 0: Min value ■ 11: Max value ■ Any other value: Entire channel status is made 0. Bits [15:0]: value
SPDIF input auto rate detect	`0x0504`	0: SPDIF RX rate is not automatically detected 1: SPDIF RX rate is automatically detected and changed SPDIF RX auto rate detect feature will be disabled if incoming rate is either 96k or 88.2k.

Table 2-9 Values supported by digital mic

Digital Mic	Key	Supported Values
Digital mic input sample rate	`0x0600`	8000 11025 12000(2) 16000 22050 24000 32000 40000(2) 44100 48000(2)
Digital mic input gain (4)	`0x0601`	Bit[15] – Select fine Digital gain If Bit[15] = 1 Bits [8:0] – Digital Gain in steps of -30dB 1: Max attenuation 31: Min attenuation 32: Unity 33: Min gain 511: Max gain If Bit[15] = 0 Bits [3:0] – Digital Gain in legacy mode 8: Max attenuation 15: Min attenuation 0: Unity 1: Min gain 7: Max gain

Table 2-9 Values supported by digital mic (cont.)

Digital Mic	Key	Supported Values
Digital mic sidetone gain (4)	`0x0602`	0-15, giving the following respective dB gains: 0: -32.6 1: -30.1 2: -26.6 3: -24.1 4: -20.6 5: -18.1 6: -14.5 7: -12 8: -8.5 9: -6.0 10: -2.5 11: 0.0 12: +3.5 13: +6.0 14: +9.5 15: +12.0
Digital mic sidetone enable (4)	`0x0603`	0: Disable 1: Enable
Digital mic sidetone source point (5)	`0x605`	Source point for sidetone data at Digital Mic. `0x00`: Data is taken before digital gain. `0x01`: Data is taken after digital gain.

Table 2-9 Values supported by digital mic (cont.)

Digital Mic	Key	Supported Values
Digital mic individual sidetone gain	`0x0606`	Gain of a particular sidetone DMIC channel. **NOTE** Alternatively when stream key, that is, Digital mic sidetone gain (`0x0602`), is used the gain of all sidetone DMIC channels are changed simultaneously. dB gain table: 0: -32.6 1: -30.1 2: -26.6 3: -24.1 4: -20.6 5: -18.1 6: -14.5 7: -12 8: -8.5 9: -6.0 10: -2.5 11: 0.0 12: +3.5 13: +6.0 14: +9.5 15: +12.0
Digital mic data source point	`0x607`	Digital mic data source selection. 0x00: Digital mic data is taken from IIR filter out 0x01: Digital mic data is taken from Digital gain filter out
Digital mic clock rate	`0x0604`	Digital mic clock rate in KHz. 500: 500 KHz 1000: 1 MHz 2000: 2 MHz 4000: 4 MHz
Digital mic G722 filter enable	`0x609`	Enables optional G722 filter that improves noise performance. 0: Disable 1: Enable

Table 2-9 Values supported by digital mic (cont.)

Digital Mic	Key	Supported Values
Digital mic G722 FIR filter enable	`0x60a`	Enables optional FIR filter inside G722 filter that droops the response slightly. 0: Disable 1: Enable
Digital mic amplifier select	`0x60b`	Configure LO_AMP_SEL and HI_AMP_SEL values of DMIC instance. The lower 16 bits sets the LO_AMP_SEL and the higher 16 bits select the HI_AMP_SEL. The two values with a maximum of 7 and minimum of 0, should be set to be symmetrical around the value of 3.5 (that is, 0,7 or 1,6 or 2,5 or 3,4 or 4,3 or 5,2 or 6,1 or 7,0).

Table 2-10 0 Values supported by audio channel

General	Key	Supported Values
Audio channel mute enable (4)	`0x0700`	0: Disable 1: Enable
Audio Sample Size (5)	`0x0701`	Selects the size (width or resolution) of the audio sample on an audio interface. All interfaces except PCM supports the following settings: 16:16-bit sample size 24: 24-bit sample size For PCM interface, following settings are supported: 0: 13 bits in a 16 bit slot 1: 16 bits in a 16 bit slot 2: 8 bits in a 16 bit slot 3: 8 bits in an 8 bit slot 16 : 16 bits(same as setting 1) 24: 24 bits For SPDIF, input channels and output channels can have different sample sizes. All channels in one direction will have same size. All codec input and output channels can have different sample size.

NOTE (1) Devices before BlueCore5 with a mono codec only support an input and output rate of 8 kHz.

(2) Not supported on devices before BlueCore7.

(3) Gain is specified in two's complement format using 4 bits (8: max attenuation, 0: unity, 7: max gain)

(4) Available from BlueCore 26 firmware onwards

(5) Available CSR8675 onwards. Contact QTIL support to confirm that a specific BlueCore device supports this.

The Value parameter is not validated and it is therefore the responsibility of the client to ensure that a supported value is specified. A key value configured using this command persists until the device restarts.

An instance of a digital interface (PCM, I^2S, or SPDIF) has a single set of configuration keys. Therefore, it is unnecessary to set the same keys for each Source ID and Sink ID that relate to a single interface. Applying key values to just one of the Source ID or Sink ID is sufficient.

Response message

The corresponding `GETRESP` signals success with `BCCMDPDU_STAT_OK` (`0x0000`) in the `Status` field.

A non-zero value in the `Status` field indicates failure. The command fails upon specifying an unrecognized `Sid` or `Key` parameter, or if the `Key` parameter is incompatible with the Source ID or Sink ID (for example, using a PCM parameter key with a Codec Source ID).

2.1.6 stream_alias_sink

Varid	Type	Permissions	Intrinsic Permissions
`0x505d`	Complex	WO	WO

This command aliases two specified Sink IDs. The two Sink IDs that are aliased automatically connect to the same source when either of them is connected to a Source ID through `stream_connect` command. This allows `sink2` to automatically output a copy of the data sent to `sink1`.

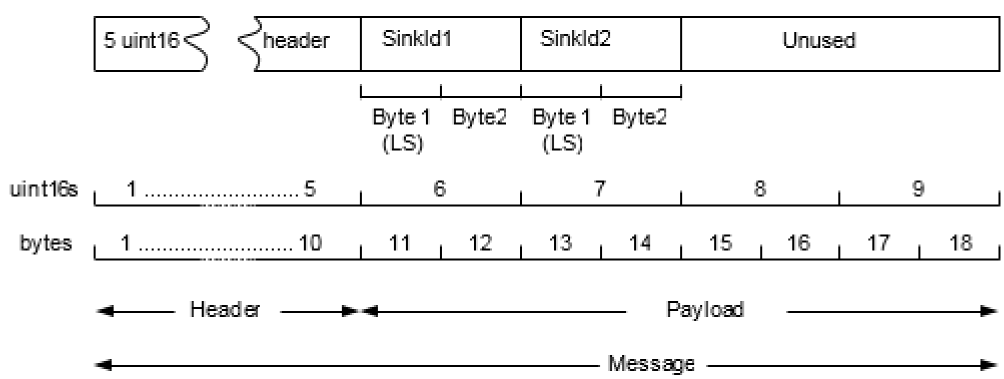

Figure 2-4 stream_alias_sink command structure

The Sink ID aliasing should be performed before they are used in a `stream_connect` command. If the `SinkId2` parameter is `0`, then the Sink ID specified by `SinkId1` is removed from an existing alias association.

When a Source ID or Sink ID is closed using `stream_close_source` command or `stream_close_sink` command, it is automatically removed from an existing alias association.

Restriction

A Sink ID that is currently connected in a transform cannot be aliased.

Response message

The corresponding GETRESP signals success with BCCMDPDU_STAT_OK (0x0000) in the Status field. A nonzero value in the Status field indicates failure.

2.1.7 stream_sync_sid

Varid	Type	Permissions	Intrinsic Permissions
0x5062	Complex	WO	WO

This command marks two specified Source IDs or Sink IDs for synchronization with each other by putting them into the same sync group. All Source IDs or Sink IDs within a particular sync group are enabled simultaneously. This is achieved by automatically deferring stream_connect commands involving synchronized Source IDs or Sink IDs until all associated Source IDs or Sink IDs have a corresponding stream_connect command issued.

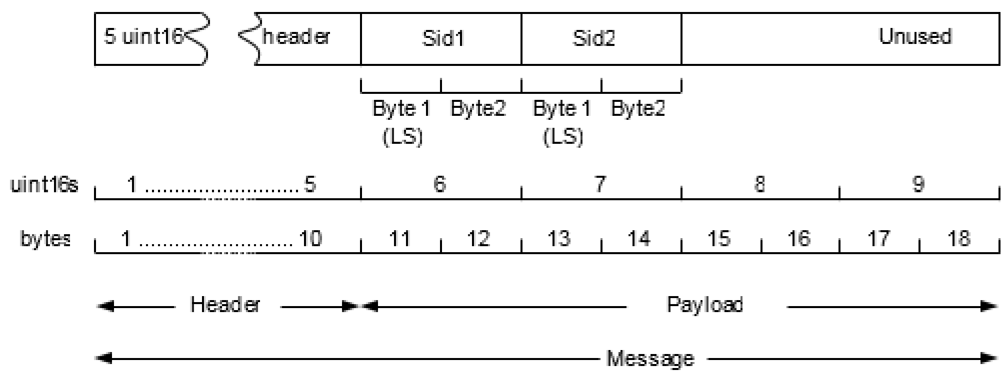

Figure 2-5 stream_sync_sid command structure

If the Sid2 parameter is zero, then the Source ID or Sink ID specified by Sid1 is removed from an existing sync group.

A sync group containing more than two Source IDs or Sink IDs is created using multiple stream_sync_sid BlueCore commands. For example, the following sequence creates a sync group consisting of four Source IDs or Sink IDs a, b, c, and d:

- stream_sync_sid a b
- stream_sync_sid c d
- stream_sync_sid a c

When a Source ID or Sink ID is closed using stream_close_source command or stream_close_sink command, it is automatically removed from an existing sync group.

Restriction

A sync group cannot contain both sources and sinks.

Response message

The corresponding `GETRESP` signals success with `BCCMDPDU_STAT_OK` (`0x0000`) in the `Status` field. A non-zero value in the `Status` field indicates failure.

2.1.8 stream_connect

Varid	Type	Permissions	Intrinsic Permissions
`0x505e`	Complex	WO	WO

This command creates and starts a transform between the specified Source ID and Sink ID (and any other Sink ID aliased to the specified Sink ID). A transform is a route along which data flows. Data enters the transform through the input, identified by the Source ID and leaves through the output, identified by the Sink ID. The format and rate of the input and output data is determined by the configuration of the Source ID and the Sink ID respectively. The direction of data flow through the transform is always from source to sink.

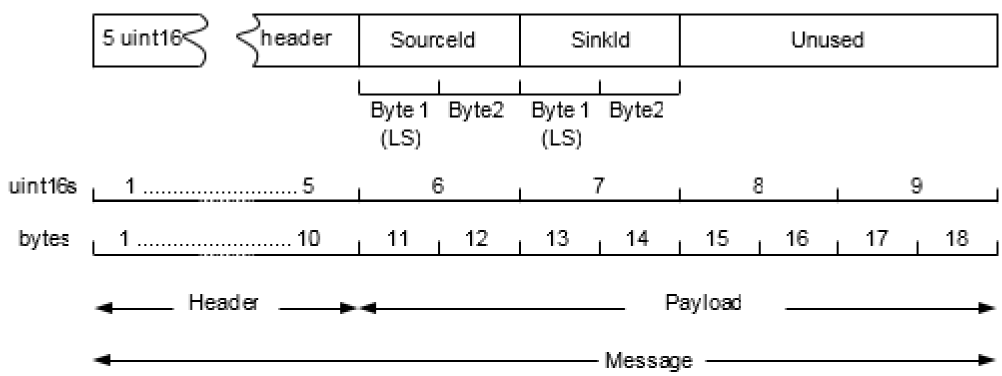

Figure 2-6 stream_connect command structure

Restriction

The Source IDs and Sink IDs that are already not part of the existing transform can be specified.

All Source and Sink IDs within a transform are set to the same sync or sample rate. Failure to ensure this may result in corrupted audio.

Response message

The corresponding `GETRESP` signals success with `BCCMDPDU_STAT_OK` (`0x0000`) in the `Status` field and returns the Transform ID of the newly created connection in the first two bytes of the payload (LS byte first).

A non-zero value in the `Status` field indicates failure. In the case of failure, the first two bytes of the payload are undefined.

The Transform ID returned is an arbitrary value that identifies the transform. It is stored by the client to allow the transform to be disconnected with a subsequent `stream_transform_disconnect` BlueCore command.

2.1.9 stream_transform_disconnect

Varid	Type	Permissions	Intrinsic Permissions
`0x486d`	uint16	WO	WO

This command disconnects the existing transform identified by Transform ID which was formed using the `stream_connect` command.

The Transform ID was returned by a previously successful `stream_connect` command.

Response message

The corresponding `GETRESP` signals success with `BCCMDPDU_STAT_OK` (`0x0000`) in the `Status` field.

A non-zero value in the `Status` field indicates failure. The command fails on specifying an invalid Transform ID.

2.1.10 map_sco_audio

Varid	Type	Permissions	Intrinsic Permissions
`0x506a`	Complex	WO	WO

This command routes the next attempted SCO connection to the specified Source ID and Sink ID.

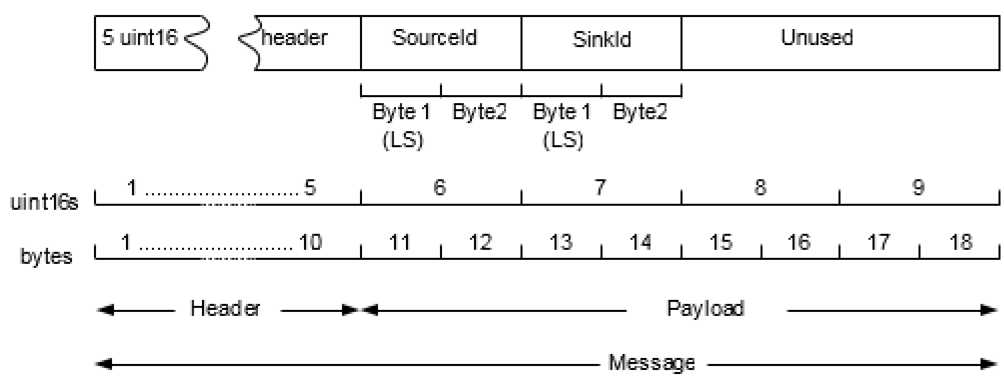

Figure 2-7 map_sco_audio command structure

The SCO source is routed to the `SinkId` and `SourceId` is routed to the SCO sink. The routings specified by the command work only for the next attempted SCO connection and then discarded.

If the `SourceId` and the `SinkId` parameters are specified as `0`, any pending routing is canceled.

For the command to succeed, both the `SourceId` and the `SinkId` parameters must either be `0`, or specify currently valid Source ID and Sink ID of the correct source/sink type.

This command is effective only if PSKEY_HOSTIO_MAP_SCO_PCM is set to `0`, and the `map_sco_pcm` BlueCore command has not been used to create a pending mapping request.

See Appendix SCO routing derivation for a schematic overview of SCO routing derivation.

Response message

The corresponding GETRESP signals success with BCCMDPDU_STAT_OK (0x0000) in the Status field.

A non-zero value in the Status field indicates failure.

2.1.11 enable_sco_streams

Varid	Type	Permissions	Intrinsic Permissions
0x4876	uint16	WO	WO

This command enables or disables the use of streams with future SCO connections. Switch value 1 enables the use of streams; value 0 disables it. When an SCO connection is made:

- If enabled, then the SCO connection has a source ID and a sink ID associated with it. These can then be connected using stream_connect.
- If disabled, then the SCO data is routed to the host directly over HCI.

This command is effective only if PSKEY_HOSTIO_MAP_SCO_PCM is set to 0 and neither of the map_sco_pcm and map_sco_audio BlueCore commands is used to configure a pending routing request. See Appendix SCO routing derivation for a schematic overview of SCO routing derivation.

In BlueCore 25 firmware, a SCO connection triggers a SCO_STREAM_HANDLES HQ event (varid 0x1017) that contains the source ID and the sink ID for that connection.

In BlueCore 26 and later firmware, the source and sink IDs are obtained using the stream_get_source and stream_get_sink commands.

Response message

The corresponding GETRESP signals success with BCCMDPDU_STAT_OK (0x0000) in the Status field.

A non-zero value in the Status field indicates failure.

2.1.12 map_sco_pcm

Varid	Type	Permissions	Intrinsic Permissions
0x481c	uint16	WO	WO

This command specifies which PCM interface and channel (time slot) is used by the next attempted SCO connection. For interface P (0 or 1) and channel C (0 to 3), the switch is calculated 4*P + C + 1. An argument of 0 clears any pending mapping request.

A mapping request set by this command automatically is cleared after the next attempted SCO connection.

This command is effective only if PSKEY_HOSTIO_MAP_SCO_PCM is set to 0.

See Appendix SCO routing derivation for a schematic overview of SCO routing derivation.

Response message

The corresponding `GETRESP` signals success with `BCCMDPDU_STAT_OK` (`0x0000`) in the `Status` field.

A non-zero value in the `Status` field indicates failure.

2.1.13 mic_bias_ctrl

NOTE Only use this BlueCore command in a production test environment.

Varid	Types	Permissions	Intrinsic Permissions
`0x7039`	Complex	RW	RW

This command has two principle functions:

- Get: To get the current mic bias settings.
- Set: To set the mic bias settings in a production test environment.

The Mic Bias system supplies a power source of known output impedance to a microphone used in audio applications. The Mic Bias output pin has four variables and used to setup the pin output.

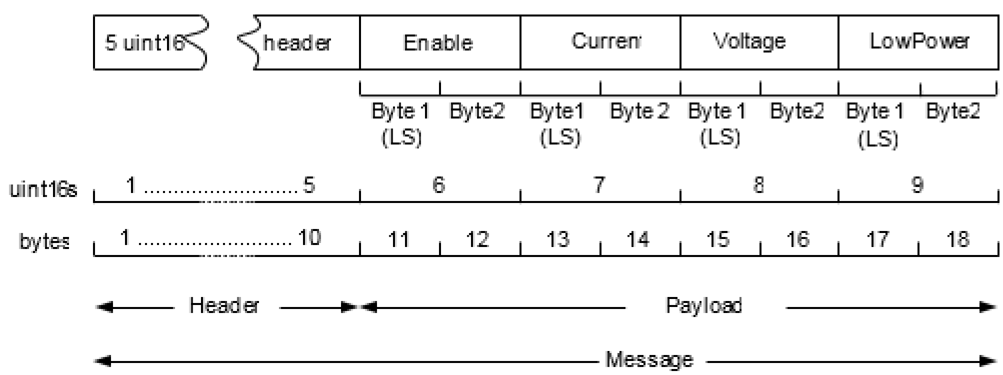

Figure 2-8 mic_bias_ctrl command structure

To set the `SETREQ` (`0x0002`) configuration specified in the Type field (first uint16) of the header, the message payload should be set up as specified in Figure 2-8.

To get the `GETREQ` (`0x0000`) configuration specified in the `Type` field of the header, the response message payload is filled as specified in Figure 2-8.

The `Enable` parameter specifies the mic bias state. A non-zero value enables the mic bias state, 0 disables it.

The `Current` parameter specifies the mic bias current. The value must be in the range 0 to 15. See Table 2-11 for the mapping between the specified value and the current generated.

The `Voltage` parameter specifies the mic bias voltage. The value must be in the range 0 to 15. See Table 2-11 for the mapping between the specified value and the voltage generated.

The `LowPower` parameter specifies the mic bias low power mode. A non-zero value enables, whereas `0` disables it. When operating in low power mode, the hardware cell is noisier than normal operation. As a result, it should only be used in less sensitive test applications.

Table 2-11 1 Mic bias voltage and current

Mic Bias Voltage Table		Mic Bias Current Table	
Parameter	Voltage	Parameter	Current
0	1.72 V	0	0.32 mA
1	1.77 V	1	0.40 mA
2	1.83 V	2	0.48 mA
3	1.89 V	3	0.56 mA
4	1.97 V	4	0.64 mA
5	2.03 V	5	0.72 mA
6	2.12 V	6	0.80 mA
7	2.20 V	7	0.88 mA
8	2.34 V	8	0.97 mA
9	2.44 V	9	1.05 mA
10	2.58 V	10	1.13 mA
11	2.71 V	11	1.21 mA
12	2.92 V	12	1.29 mA
13	3.10 V	13	1.37 mA
14	3.34 V	14	1.45 mA
15	3.60 V	15	1.53 mA

NOTE The successful use of this command is not based on the PSKEY_CODEC_PIO defined.

Response message

The corresponding `GETRESP` signals success with `BCCMDPDU_STAT_OK` (`0x0000`) in the `Status` field. A non-zero value in the `Status` field indicates failure.

If the response relates to a `GETREQ` message, the current mic bias configuration fills the payload. If the response relates to a `SETREQ` message, the payload contents are undefined.

2.1.14 stream_sidetone_en

Varid	Type	Permissions	Intrinsic ermissions
`0x4886`	uint16	WO	WO

This command can be used to either enable or disable the sidetone path once the streams are configured appropriately. A value of `1` enables the sidetone path, a value of `0` disables it.

Response message

The corresponding GETRESP signals success with BCCMDPDU_STAT_OK (0x0000) in the Status field. A non-zero value in the Status field indicates failure.

2.2 Deprecated BlueCore commands

The following BlueCore commands have been deprecated:

- codec_input_gain
- codec_output_gain
- pcm_attenuation
- pcm2_attenuation
- pcm_clock_rate
- pcm_sync_rate
- pcm_slots_per_frame
- pcm_config32
- digital_audio_rate
- digital_audio_config

2.2.1 codec_input_gain

NOTE This BlueCore command is deprecated. Use stream_configure command by specifying the codec_input_gain key (0x0302).

Varid	Type	Permissions	Intrinsic Permissions
0x5058	Complex	WO	WO

This command specifies the input gains for codecs A and B (where present).

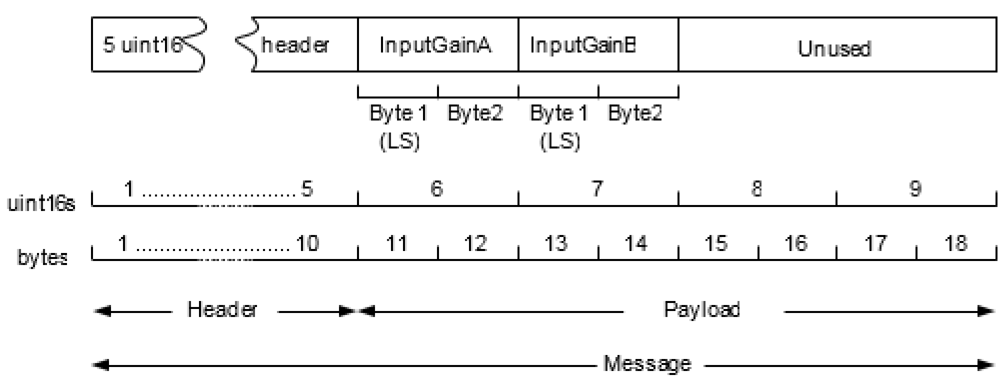

Figure 2-9 codec_input_gain command structure

The InputGainA and InputGainB parameters accept values in the range 0 to 22. A device with a single codec channel ignores the value specified by the InputGainB parameter.

The initial value of the codec input gains is set from PSKEY_CODEC_IN_GAIN. This command allows subsequent changes to the codec input gains on a channel specific basis.

2.2.2 codec_output_gain

This BlueCore command is deprecated. Use `stream_configure` command by specifying the `codec_output_gain` key (`0x0303`).

Varid	Type	Permissions	Intrinsic Permissions
`0x5059`	Complex	WO	WO

This command specifies the output gains for codecs A and B (where present).

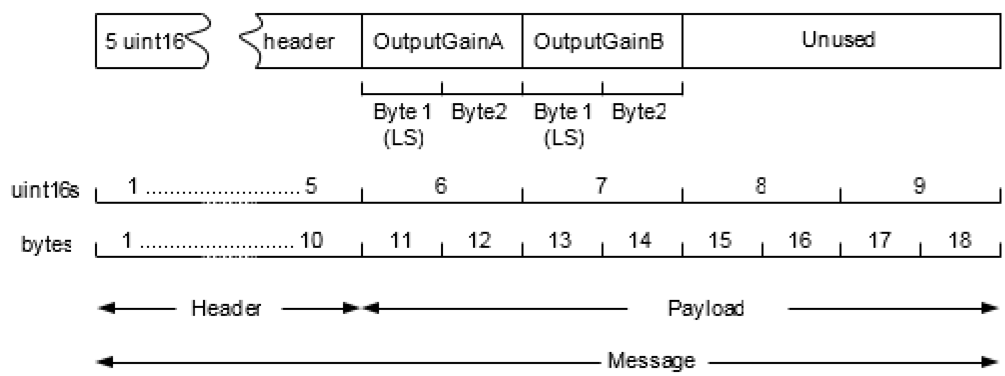

Figure 2-10 0 codec_output_gain command structure

The `OutputGainA` and `OutputGainB` parameters accept values in the range 0 to 22. A device with a single codec channel ignores the value specified by the `OutputGainB` parameter.

The initial value of the codec output gains is set from PSKEY_CODEC_OUT_GAIN. This command allows subsequent changes to the codec output gains on a channel specific basis.

2.2.3 pcm_attenuation

NOTE This BlueCore command is deprecated. Use `stream_configure` command by specifying the `pcm_audio_gain` key (`0x0116`).

Varid	Type	Permissions	Intrinsic Permissions
`0x6832`	uint16	RW	RW

Some codecs allow gain control by the top three bits received at the end of a 13-bit PCM sample in a 16-bit PCM frame. The value of these 3 bits in all such samples sent from BlueCore over the first PCM port initializes from PSKEY_PCM0_ATTENUATION. The 3-bit value (specified using the three least significant bits) changes after using this command.

2.2.4 pcm2_attenuation

NOTE This BlueCore command is deprecated. Use `stream_configure` command by specifying the `pcm_audio_gain` key (`0x0116`).

Varid	Type	Permissions	Intrinsic Permissions
`0x4868`	uint16	WO	WO

This command is the second PCM interface equivalent of `pcm_attenuation`. The value of the three bits in all samples sent from BlueCore over the second PCM port initializes from PSKEY_PCM0_ATTENUATION. The 3-bit value (specified using the three least significant bits) changes subsequently using this command.

2.2.5 pcm_clock_rate

NOTE This BlueCore command is deprecated. Use `stream_configure` command by specifying the `pcm_master_clock_rate` key (`0x0101`).

Varid	Type	Permissions	Intrinsic Permissions
`0x5068`	Complex	WO	WO

This command specifies the master clock rate for the specified interface when operating in PCM mode.

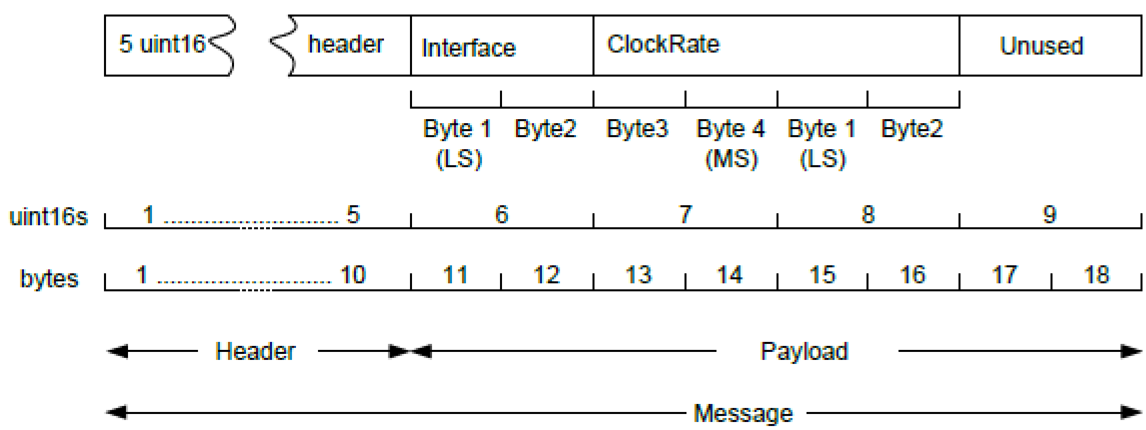

Figure 2-11 1 pcm_clock_rate command structure

The `Interface` parameter values are:

- First PCM interface: `0x0000`
- Second PCM interface: `0x0001`

The `ClockRate` parameter specifies the master clock rate in Hz. If set to `0`, the master clock rate is derived from the slot width, slots per frame and sync rate. See Appendix PCM master clock rate derivation for more details.

The initial values for the PCM master clock rate for the first and second interface are set from PSKEY_PCM_CLOCK_RATE and PSKEY_PCM2_CLOCK_RATE respectively.

2.2.6 pcm_sync_rate

NOTE This BlueCore command is deprecated. Use `stream_configure` command by specifying the `pcm_sync_rate` key (`0x0100`).

Varid	Type	Permissions	Intrinsic Permissions
`0x5069`	Complex	WO	WO

This command specifies the sync rate for the specified interface when operating in PCM mode.

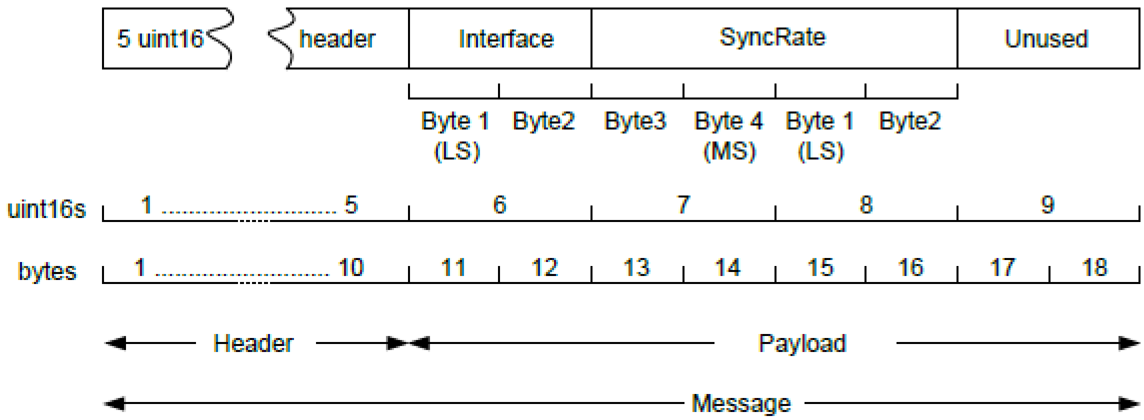

Figure 2-12 2 pcm_sync_rate

The `Interface` parameter values are:

- First PCM interface: `0x0000`
- Second PCM interface: `0x0001`

The `SyncRate` parameter specifies the sync rate in Hz.

The initial values for the PCM sync rates for the first and second interface are set from PSKEY_PCM_SYNC_RATE and PSKEY_PCM2_SYNC_RATE respectively.

2.2.7 pcm_slots_per_frame

NOTE This BlueCore command is deprecated. Use `stream_configure` command by specifying the `pcm_slot_count` key (`0x0103`).

Varid	Type	Permissions	Intrinsic Permissions
`0x5067`	Complex	WO	WO

This command specifies the number of slots between sync pulses for the specified interface when operating in PCM mode.

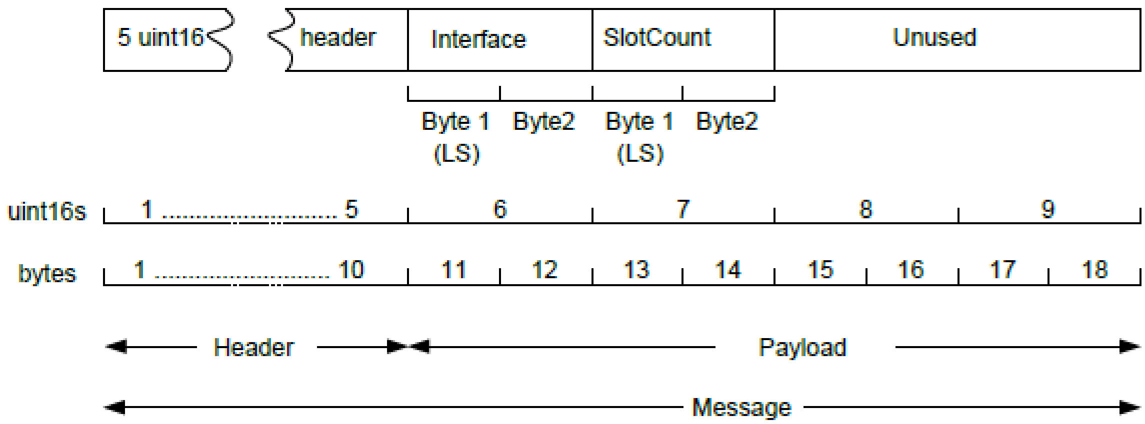

Figure 2-13 3 pcm_slots_per_frame command structure

The `Interface` parameter values are:

- First PCM interface: `0x0000`
- Second PCM interface: `0x0001`

The `SlotCount` parameter should be a value in the range 0 to 4. To specify a specific number of slots, provide a value in the range 1 to 4. To derive the number of slots implicitly from the master clock and sync rate, specify a value of 0.

The initial values for the PCM slot counts for the first and second interface are set from PSKEY_PCM_SLOTS_PER_FRAME and PSKEY_PCM2_ SLOTS_PER_FRAME respectively.

2.2.8 pcm_config32

NOTE This BlueCore command is deprecated. Use `stream_configure` command to set individual PCM interface configuration parameters.

Varid	Type	Permissions	Intrinsic Permissions
`0x502f`	Complex	WO	WO

This command specifies the default settings for the specified digital audio interface when operating in PCM mode.

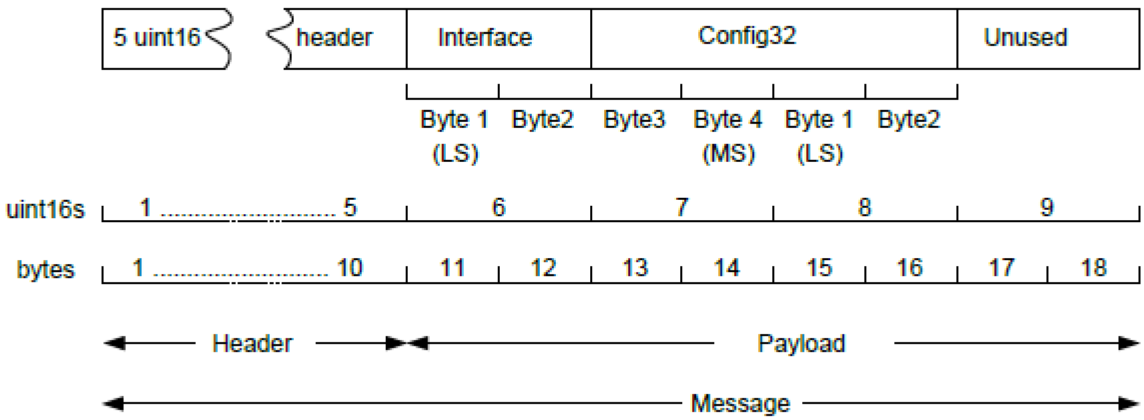

Figure 2-14 4 pcm_config32 command structure

The `Interface` parameter values are:

- First PCM interface: `0x0000`
- Second PCM interface: `0x0001`

The `Config32` parameter bit fields set specify the PCM interface configuration that is identified by the `Interface` parameter. For more information, see BlueCore documentation specific to your device.

The initial configuration for the first and second PCM interface are set from PSKEY_PCM_CONFIG32 and PSKEY_PCM2_ CONFIG32 respectively.

2.2.9 digital_audio_rate

NOTE This BlueCore command is deprecated. Use `stream_configure` command by specifying the `i2s_master_clock_rate` key (`0x0201`) to set the master clock rate or the `i2s_bits_per_sample` key (`0x020a`) to set the number of bits per sample. For devices running BlueCore 25 firmware, if the number of bits per sample required configuration (that is, no clock rate specified), then use the relevant PS Key in preference to this command.

Varid	Type	Permissions	Intrinsic Permissions
`0x5032`	Complex	WO	WO

This command specifies the master clock rate or the number of bits per sample for the specified digital audio interface when operating in I^2S mode.

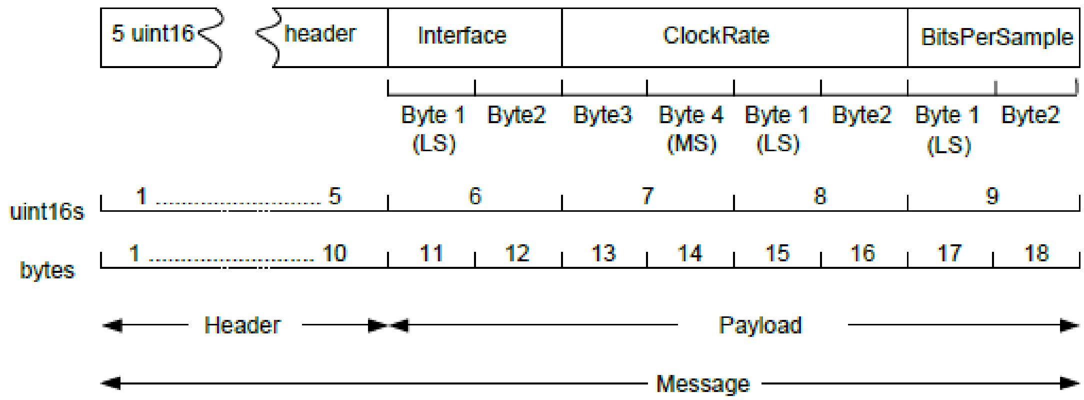

Figure 2-15 5 digital_audio_rate command structure

The `Interface` parameter values are:

- First I^2S interface: `0x0000`
- Second I^2S interface: `0x0001`

The ClockRate parameter specifies the master clock rate (in Hz) that is used by the specified interface when it is operation in I^2S master mode. The clock rate is derived from the sync rate and the number of bits per sample by specifying 0.

The `BitPerSample` parameter specifies the number of bits the specified digital audio interface clocks per sample when operating in I^2S mode. The valid values are 16, 20 and 24. If the number of bits per sample is larger than the internal audio format used by BlueCore, the additional bits are output as 0s in the LSBs. The value of this parameter is ignored if the `ClockRate` parameter specifies a non-zero value.

The initial values for the I^2S master clock rate for the first and second digital interface are set from PSKEY_DIGITAL_AUDIO_RATE and PSKEY_DIGITAL_AUDIO2_RATE respectively.

The initial values for the I^2S bits per sample for the first and second digital interfaces are set from PSKEY_DIGITAL_AUDIO_BITS_PER_SAMPLE and PSKEY_DIGITAL_AUDIO2_BITS_PER_SAMPLE respectively.

2.2.10 digital_audio_config

NOTE This BlueCore command is deprecated. Use `stream_configure` command by specifying the individual I^2S interface configuration parameters.

Varid	Type	Permissions	Intrinsic Permissions
`0x5033`	Complex	WO	WO

This command specifies the default settings for the specified digital audio interface when operating in I²S mode.

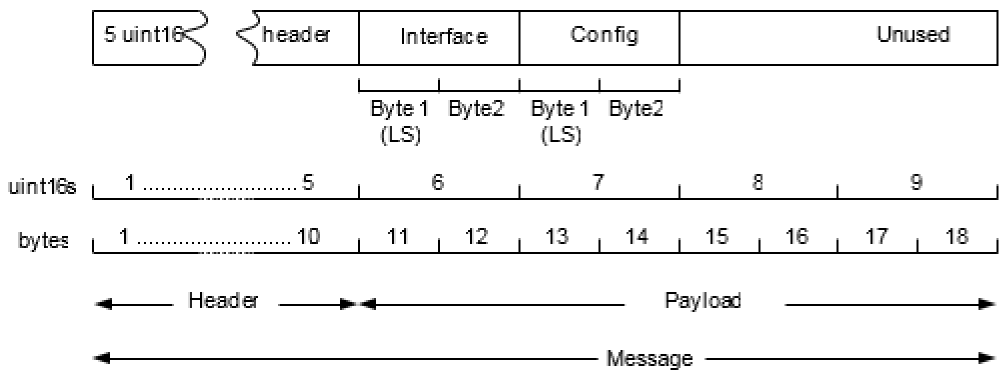

Figure 2-16 6 digital_audio_config command structure

The `Interface` parameter values are:

- First I²S interface: `0x0000`
- Second I²S interface: `0x0001`

The `Config` parameter is a set of bit fields specifying additional configuration details when operating the specified digital interface in I²S mode. For more information, see BlueCore documentation specific to your device.

The initial settings of the bit fields configured by this message for the first and second digital interface are set from PSKEY_DIGITAL_AUDIO_CONFIG and PSKEY_DIGITAL_AUDIO2_CONFIG respectively.

2.3 Audio API examples

The example code provided in this section can be used with the BTCli application on BlueCore7 devices to show the use of the BlueCore commands described in this document.

For information on commands not described in this document, see the *HQ and BCCMD Commands Protocols*.

2.3.1 Removing occasional I²S 1-sample shift on CSR8670

As per HW design of CSR8670, sometimes a 1-sample shift might appear b/w left and right channels on the I²S interface. This shift can be removed by configuring `i2s_tx_start_sample` and `i2s_rx_start_sample` stream keys on the sink and source `sid`s respectively. However, the setting is dependent upon the I²S line format. The setting should be set as described below.

Table 2-12 2 I²S TX configuration

S.No.	Scenario	I2s_tx_start_sample	crop_enable
1	Left justified Audio starts immediately after WS falling edge Clock rate = 32*Fs(frame is fully packed, there are no idle CLK cycles) Audio sample size = 16, Polarity# = 1 #Polarity of I2S Sync	1	1
2	Left justified Audio starts 1 CLK after WS falling edge Clock rate = 32*Fs(frame is fully packed, there are no idle CLK cycles) Audio sample size = 16, Polarity = 1	0	1
3	Left justified Audio starts immediately after WS falling edge Clock rate > 32*Fs(frame is not fully packed, there are idle CLK cycles) Audio sample size = 16, Polarity = 1	1	1
4	Left justified Audio starts 1 CLK after WS falling edge Clock rate > 32*Fs(frame is not fully packed, there are idle CLK cycles) Audio sample size = 16, Polarity = 1	0	1
5	Right justified Clock rate = 32*Fs(frame is fully packed, there are no idle CLK cycles) Audio sample size = 16, Polarity = 1	1	1
6	Right justified Clock rate > 32*Fs(frame is not fully packed, there are idle CLK cycles) Audio sample size = 16, Polarity =1	0	0

S.No.	Scenario	I2s_tx_start_sample	crop_enable
1	Left justified Audio starts immediately after WS falling edge Clock rate = 32*Fs(frame is fully packed, there are no idle CLK cycles) Audio sample size = 16, Polarity =0	0	1
2	Left justified Audio starts 1 CLK after WS falling edge Clock rate = 32*Fs(frame is fully packed, there are no idle CLK cycles) Audio sample size = 16, Polarity =0	1	1
3	Left justified Audio starts immediately after WS falling edge Clock rate > 32*Fs(frame is not fully packed, there are idle CLK cycles) Audio sample size = 16, Polarity =0	1	0
4	Left justified Audio starts 1 CLK after WS falling edge Clock rate > 32*Fs(frame is not fully packed, there are idle CLK cycles) Audio sample size = 16, Polarity =0	1	0
5	Right justified Clock rate = 32*Fs(frame is fully packed, there are no idle CLK cycles) Audio sample size = 16, Polarity =0	1	1
6	Right justified Clock rate > 32*Fs(frame is not fully packed, there are idle CLK cycles) Audio sample size = 16, Polarity =0	1	0

Table 2-13 3 I^{2}S RX configuration

S.No.	Scenario	I2s_rx_start_sample	crop_enable
1	Left justified Audio starts immediately after WS falling edge Clock rate = 32*Fs(frame is fully packed, there are no idle CLK cycles) Audio sample size = 16, Polarity =1	0	1
2	Left justified Audio starts 1 CLK after WS falling edge Clock rate = 32*Fs(frame is fully packed, there are no idle CLK cycles) Audio sample size = 16, Polarity =1	1	1

Table 2-13 3 I^{2}S RX configuration (cont.)

S.No.	Scenario	I2s_rx_start_sample	crop_enable
3	Left justified Audio starts immediately after WS falling edge Clock rate > 32*Fs(frame is not fully packed, there are idle CLK cycles) Audio sample size = 16, Polarity =1	0	0
4	Left justified Audio starts 1 CLK after WS falling edge Clock rate > 32*Fs(frame is not fully packed, there are idle CLK cycles) Audio sample size = 16, Polarity =1	0	0
5	Right justified Clock rate > 32*Fs(frame is fully packed, there are no idle CLK cycles) Audio sample size = 16, Polarity =1	0	X

S.No.	Scenario	I2s_rx_start_sample	crop_enable
1	Left justified Audio starts immediately after WS falling edge Clock rate = 32*Fs(frame is fully packed, there are no idle CLK cycles) Audio sample size = 16, Polarity = 0	1	1
2	Left justified Audio starts 1 CLK after WS falling edge Clock rate = 32*Fs(frame is fully packed, there are no idle CLK cycles) Audio sample size = 16, Polarity = 0	0	1
3	Left justified Audio starts immediately after WS falling edge Clock rate > 32*Fs(frame is not fully packed, there are idle CLK cycles) Audio sample size = 16, Polarity = 0	1	0
4	Left justified Audio starts 1 CLK after WS falling edge Clock rate > 32*Fs(frame is not fully packed, there are idle CLK cycles) Audio sample size = 16, Polarity = 0	1	0
5	Right justified Clock rate > 32*Fs(frame is fully packed, there are no idle CLK cycles) Audio sample size = 16, Polarity = 0	0	X

2.3.2 Removing occasional I²S 1-sample shift on CSR8675

As per HW design of CSR8675, sometimes a 1-sample shift might appear b/w left and right channels on the I²S interface. This shift can be removed by configuring `i2s_tx_start_sample` and `i2s_rx_start_sample` stream keys on the sink and source `sid`s respectively. However, the setting is dependent upon the I²S line format. The setting should be set as described below.

Table 2-14 4 I²S TX Configuration

S.No.	Scenario	I2s_tx_start_sample	crop_enable
1	Left justified Audio starts immediately after WS falling edge Clock rate = 32*Fs(frame is fully packed, there are no idle CLK cycles) Audio sample size = 16, Polarity# = 1 #Polarity of I2S Sync	1	1
2	Left justified Audio starts 1 CLK after WS falling edge Clock rate = 32*Fs(frame is fully packed, there are no idle CLK cycles) Audio sample size = 16, Polarity = 1	0	1
3	Left justified Audio starts immediately after WS falling edge Clock rate > 32*Fs(frame is not fully packed, there are idle CLK cycles) Audio sample size = 16, Polarity = 1	1	0
4	Left justified Audio starts 1 CLK after WS falling edge Clock rate > 32*Fs(frame is not fully packed, there are idle CLK cycles) Audio sample size = 16, Polarity = 1	0	0
5	Right justified Clock rate = 32*Fs(frame is fully packed, there are no idle CLK cycles) Audio sample size = 16, Polarity = 1	1	1
6	Right justified Clock rate > 32*Fs(frame is not fully packed, there are idle CLK cycles) Audio sample size = 16, Polarity =1	1	0

S.No.	Scenario	I2s_tx_start_sample	crop_enable
1	Left justified Audio starts immediately after WS falling edge Clock rate = 32*Fs(frame is fully packed, there are no idle CLK cycles) Audio sample size = 16, Polarity =0	0	1
2	Left justified Audio starts 1 CLK after WS falling edge Clock rate = 32*Fs(frame is fully packed, there are no idle CLK cycles) Audio sample size = 16, Polarity =0	1	1
3	Left justified Audio starts immediately after WS falling edge Clock rate > 32*Fs(frame is not fully packed, there are idle CLK cycles) Audio sample size = 16, Polarity =0	0	0
4	Left justified Audio starts 1 CLK after WS falling edge Clock rate > 32*Fs(frame is not fully packed, there are idle CLK cycles) Audio sample size = 16, Polarity =0	1	0
5	Right justified Clock rate = 32*Fs(frame is fully packed, there are no idle CLK cycles) Audio sample size = 16, Polarity =0	0	1
6	Right justified Clock rate > 32*Fs(frame is not fully packed, there are idle CLK cycles) Audio sample size = 16, Polarity =0	0	0

Table 2-15 5 I^{2}S RX Configuration

S.No.	Scenario	I2s_rx_start_sample	crop_enable
1	Left justified Audio starts immediately after WS falling edge Clock rate = 32*Fs(frame is fully packed, there are no idle CLK cycles) Audio sample size = 16, Polarity =1	1	1
2	Left justified Audio starts 1 CLK after WS falling edge Clock rate = 32*Fs(frame is fully packed, there are no idle CLK cycles) Audio sample size = 16, Polarity =1	0	1

Table 2-15 5 I²S RX Configuration (cont.)

S.No.	Scenario	I2s_rx_start_sample	crop_enable
3	Left justified Audio starts immediately after WS falling edge Clock rate > 32*Fs(frame is not fully packed, there are idle CLK cycles) Audio sample size = 16, Polarity =1	1	0
4	Left justified Audio starts 1 CLK after WS falling edge Clock rate > 32*Fs(frame is not fully packed, there are idle CLK cycles) Audio sample size = 16, Polarity =1	0	0
5	Right justified Clock rate > 32*Fs(frame is fully packed, there are no idle CLK cycles) Audio sample size = 16, Polarity =1	1	0

S.No.	Scenario	I2s_rx_start_sample	crop_enable
1	Left justified Audio starts immediately after WS falling edge Clock rate = 32*Fs(frame is fully packed, there are no idle CLK cycles) Audio sample size = 16, Polarity = 0	0	1
2	Left justified Audio starts 1 CLK after WS falling edge Clock rate = 32*Fs(frame is fully packed, there are no idle CLK cycles) Audio sample size = 16, Polarity = 0	1	1
3	Left justified Audio starts immediately after WS falling edge Clock rate > 32*Fs(frame is not fully packed, there are idle CLK cycles) Audio sample size = 16, Polarity = 0	0	0
4	Left justified Audio starts 1 CLK after WS falling edge Clock rate > 32*Fs(frame is not fully packed, there are idle CLK cycles) Audio sample size = 16, Polarity = 0	1	0
5	Right justified Clock rate > 32*Fs(frame is fully packed, there are no idle CLK cycles) Audio sample size = 16, Polarity = 0	0	X

2.3.3 Codec RX to I2S TX (audio sample = 16bits) in left justified mode

This command sequence routes codec RX to the first I2S interface in 16-bit format.

The I2S interface operates in following configuration:

- Master mode
- Audio sample width = 16 bits
- Sampling rate = 48 KHz
- Clock rate = 2.304 MHz (implies a slot length of 24CLKs, but audio sample is 16 bits)
- L channel when WS is low and R channel when WS is high
- Left justified with 1 CLK delay b/w 1st bit of audio data and WS edge

```
#Get Codec streams and configure @48k
stream_get_source codec 0 0 <returns source1_id>
stream_get_source codec 0 1 <returns source2_id>
stream_configure source1_id codec_input_rate 48000
stream_configure source2_id codec_input_rate 48000
#Get I2S streams and enable master mode
stream_get_sink i2s 0 0 <returns sink1_id>
stream_get_sink i2s 0 1 <returns sink2_id>
stream_configure sink1_id i2s_master_mode 1
#Configure I2S clock and sample rate
#Here clk rate = 48*sample rate
#Each I2S channel is of 16 bits carried in 24 clks
stream_configure sink1_id i2s_sync_rate 48000
stream_configure sink1_id i2s_master_clock_rate 2304000
#Configure transmission of L channel when WS is low and
#transmission of R channel when WS is high
#Not required if already set using PSKEY
stream_configure sink1_id i2s_chnl_plrty 1
#I2S is left justified with delay of 1 clk from WS edge when 1st bit
is on the line
#Not required if already set using PSKEY
#Defualt PSKEY already sets this setting
stream_configure sink1_id i2s_jstfy_format 0
stream_configure sink2_id i2s_lft_jstfy_dly 1
stream_configure sink2_id i2s_tx_start_sample 0
#Sync streams with each other
stream_sync_sid source1_id source2_id
stream_sync_sid sink1_id sink2_id
#Final step: Connect streams
stream_connect source1_id sink1_id <returns trans1_id>
stream_connect source2_id sink2_id <returns trans2_id>
```

2.3.4 SPDIF RX to I²S TX (audio sample =24bits) in left justified mode

This command sequence routes SPDIF RX to the first I²S interface in 24 bit format.

NOTE This is only supported in CSR8675.

The I²S interface operates in the following configuration:

- Master mode
- Sampling rate = 48 KHz
- Audio sample width = 24 bits
- Clock rate = 3.072 MHz (implies a slot length of 32 CLKs)
- L channel when WS is low and R channel when WS is high
- Left justified with 1 CLK delay b/w 1st bit of audio data and WS edge

```
#Get SPDIF streams and configure @48k in 24bit mode
stream_get_source spdif 0 0 <returns source1_id>
stream_get_source spdif 0 1 <returns source2_id>
stream_configure source1_id spdif_output_rate 48000
stream_configure source2_id spdif_output_rate 48000
stream_configure source1_id audio_sample_size 24
stream_configure source2_id audio_sample_size 24
#Get I2S streams and enable master mode
stream_get_sink i2s 0 0 <returns sink1_id>
stream_get_sink i2s 0 1 <returns sink2_id>
stream_configure sink1_id i2s_master_mode 1
#Configure I2S clock and sample rate
#Here clk rate = 64*sample rate
#Each I2S channel is of 24 bits carried in 32 clks
stream_configure sink1_id i2s_sync_rate 48000
stream_configure sink1_id i2s_master_clock_rate 3072000
#Configure transmission of L channel when WS is low and
#transmission of R channel when WS is high
#Not required if already set using PSKEY
stream_configure sink1_id i2s_chnl_plrty 1
#I2S is left justified with delay of 1 clk from WS edge when 1st bit
is on the line
#Not required if using default PSKEY
stream_configure sink1_id i2s_jstfy_format 0
stream_configure sink1_id i2s_lft_jstfy_dly 1
#Configure I2S in 24 bit mode
stream_configure sink1_id audio_sample_size 24
#Sync streams with each other
stream_sync_sid source1_id source2_id
stream_sync_sid sink1_id sink2_id
#Final step: Connect streams
```

```
stream_connect source1_id sink1_id <returns trans1_id>
stream_connect source2_id sink2_id <returns trans2_id>
```

2.3.5 FM RX to Both I²S and Codec

This command sequence connects FM RX to both I²S and codec:

```
stream_get_source fm 0 0 <returns source1_id>
stream_get_source fm 0 1 <returns source2_id>
stream_get_sink i2s 0 0 <returns sink1_id>
stream_get_sink i2s 0 1 <returns sink2_id>
stream_get_sink codec 0 0 <returns sink3_id>
stream_get_sink codec 0 1 <returns sink4_id>
stream_sync_sid source1_id source2_id
stream_sync_sid sink1_id sink2_id
stream_sync_sid sink3_id sink4_id
stream_configure sink1_id i2s_master_clock_rate 2304000
stream_configure sink1_id i2s_sync_rate 48000
stream_configure sink1_id i2s_master_mode 1
stream_alias_sink sink1_id sink3_id
stream_alias_sink sink2_id sink4_id
stream_connect source1_id sink1_id
stream_connect source2_id sink2_id
```

This command sequence connects FM RX to both I²S and codec with the related FM radio commands:

```
stream_get_source fm 0 0 <returns source1_id>
stream_get_source fm 0 1 <returns source2_id>
stream_get_sink i2s 0 0 <returns sink1_id>
stream_get_sink i2s 0 1 <returns sink2_id>
stream_get_sink codec 0 0 <returns sink3_id>
stream_get_sink codec 0 1 <returns sink4_id>
stream_sync_sid source1_id source2_id
stream_sync_sid sink1_id sink2_id
stream_sync_sid sink3_id sink4_id
stream_configure sink1_id i2s_master_clock_rate 2304000
stream_configure sink1_id i2s_sync_rate 48000
stream_configure sink1_id i2s_master_mode 1
stream_alias_sink sink1_id sink3_id
stream_alias_sink sink2_id sink4_id
bcset fm_reg power 3
bcset fm_reg freq 43000
bcset fm_reg tuner_mode 1
stream_connect source1_id sink1_id
stream_connect source2_id sink2_id
bcset fm_reg mute_state 0
```

The `fm_req` frequency register value is calculated by subtracting 60,000 from the FM frequency in kHz. Therefore, the parameter of 43,000 tunes the receiver to 103.0 MHz.

This command sequence disconnects all audio routings and afterwards cleans up the connection:

```
stream_transform_disconnect trans1_id
stream_transform_disconnect trans2_id
stream_close_source source1_id
stream_close_source source2_id
stream_close_sink sink1_id
stream_close_sink sink2_id
stream_close_sink sink3_id
stream_close_sink sink4_id
```

3 PS Keys

The PS Keys described in this section enable the specification of default hardware configurations. These values are effective only at boot time and are subsequently overridden by the `stream_configure` BlueCore command.

3.1 PCM PS Keys

The following PS Keys are used to configure audio:

- PSKEY_PCM_CLOCK_RATE
- PSKEY_PCM_SLOTS_PER_FRAME
- PSKEY_PCM_SYNC_RATE
- PSKEY_PCM_USE_LOW_JITTER_MODE
- PSKEY_PCM_CONFIG32
- PSKEY_PCM0_ATTENUATION
- PSKEY_PCM2_CLOCK_RATE
- PSKEY_PCM2_SLOTS_PER_FRAME
- PSKEY_PCM2_SYNC_RATE
- PSKEY_PCM2_USE_LOW_JITTER_MODE
- PSKEY_PCM2_CONFIG32

3.1.1 PSKEY_PCM_CLOCK_RATE

Default value (uint32): `0`

This key enables you to specify the exact clock rate (in Hz) when acting as a master on the first digital audio interface in PCM mode. If the value of this key is set to `0`, the clock rate for the interface is calculated from the slot width, number of slots and sync rate. See Appendix PCM master clock rate derivation for more information.

3.1.2 PSKEY_PCM_SLOTS_PER_FRAME

Default value (uint16): `0`

This key specifies the number of PCM slots present between sync pulses for the first PCM interface.

The value of this key is referred only if a master clock rate is not specified. See Appendix PCM master clock rate derivation for more information.

3.1.3 PSKEY_PCM_SYNC_RATE

Default value (uint32): `8000`

This key specifies the sync rate for the first digital interface when operating as a master in PCM mode.

3.1.4 PSKEY_PCM_USE_LOW_JITTER_MODE

Default value (Boolean): `FALSE (0)`

This key specifies if the first digital interface should be configured for low jitter operation when operating at a sync rate of 8 kHz in PCM mode. It replaces PSKEY_PCM_LOW_JITTER_CONFIG. Selecting low jitter mode increases power consumption.

NOTE Uses low jitter mode automatically at all sync rates other than 8 kHz.

3.1.5 PSKEY_PCM_CONFIG32

Default value (uint32): `0x00800000`

This key is a set of bit fields that specify extra configuration details for the first digital interface, used when operating in PCM mode. For more information, see BlueCore documentation specific to your device.

3.1.6 PSKEY_PCM0_ATTENUATION

Default value (uint16): `3`

Some codecs allow their gain to be controlled by three extra bits received at the end of a 13-bit PCM sample. This key allows the value of those three bits to be specified. The value specified by this key applies to both PCM interfaces (if present).

Some codecs allow gain control by the top three bits received at the end of a 13-bit PCM sample. This key allows you to specify those three bits. The value specified by this key applies to both PCM interface (if second interface is present).

3.1.7 PSKEY_PCM2_CLOCK_RATE

Default value (uint32): `0`

This key allows you to specify the exact clock rate generated (in Hz) when acting as a master on the second digital audio interface in PCM mode.

If the value of this key is set to `0`, the clock rate for the interface is calculated from the slot width, number of slots and sync rate. See Appendix PCM master clock rate derivation for more information.

3.1.8 PSKEY_PCM2_SLOTS_PER_FRAME

Default value (uint16): `0`

This key specifies the number of PCM slots present between sync pulses for the second PCM interface.

The value of this key is referred only if a master clock rate is not specified. See Appendix PCM master clock rate derivation for more information.

3.1.9 PSKEY_PCM2_SYNC_RATE

Default value (uint32): `8000`

This key specifies the sync rate for the second digital interface when operating as a master in PCM mode.

3.1.10 PSKEY_PCM2_USE_LOW_JITTER_MODE

Default value (Boolean): `FALSE (0)`

This key specifies if the second digital interface should be configured for low jitter operation when operating at a sync rate of 8 kHz in PCM mode. It replaces PSKEY_PCM2_LOW_JITTER_CONFIG. Selecting low jitter mode increases power consumption.

NOTE Uses low jitter mode automatically at all sync rates other than 8 kHz.

3.1.11 PSKEY_PCM2_CONFIG32

Default value (uint32): `0x00800000`

This key is a set of bit fields that specify extra configuration details for the second digital interface, used when operating in PCM mode. For more information, see BlueCore documentation specific to your device.

3.2 I^{2}S PS Keys

The following PS Keys are used to configure I^{2}S:

- PSKEY_I2S_MASTER_EN
- PSKEY_DIGITAL_AUDIO_RATE
- PSKEY_DIGITAL_AUDIO_BITS_PER_SAMPLE
- PSKEY_I2S_SYNC_RATE
- PSKEY_DIGITAL_AUDIO_CONFIG
- PSKEY_SIDE_TONE_GAIN
- PSKEY_DIGITAL_AUDIO2_RATE
- PSKEY_DIGITAL_AUDIO2_BITS_PER_SAMPLE

- PSKEY_DIGITAL_AUDIO2_BITS_PER_SAMPLE
- PSKEY_I2S2_SYNC_RATE
- PSKEY_DIGITAL_AUDIO2_CONFIG

3.2.1 PSKEY_I2S_MASTER_EN

Default value (Boolean): `FALSE (0)`

This key specifies if the first I^2S interface should operate in slave (`FALSE`) or master (`TRUE`) mode.

3.2.2 PSKEY_DIGITAL_AUDIO_RATE

Default value (uint32): `0`

This key allows the specification (in Hz) of the exact clock rate to be generated when acting as a master on the first digital audio interface in I^2S mode.

If the key is set to `0`, the clock rate is calculated based on the sync rate and the number of bits per sample. See Appendix I^2S Master clock rate derivation for more information.

3.2.3 PSKEY_DIGITAL_AUDIO_BITS_PER_SAMPLE

Default value (uint16): `24`

This key specifies the number of bits the first digital audio interface clocks per sample in I^2S mode. If PSKEY_DIGITAL_AUDIO_RATE is specified (that is, a non-zero value), then the value of this key is ignored. See Appendix I^2S Master clock rate derivation for more information.

3.2.4 PSKEY_I2S_SYNC_RATE

Default value (uint32): `8000`

This key specifies the sync rate for the first digital interface when operating as a master in I^2S mode.

3.2.5 PSKEY_DIGITAL_AUDIO_CONFIG

Default value (uint16): `0x0006`

This key is a set of bit fields that specify more configuration details when operating the first digital interface in I^2S mode. Table 3-1 describes each of the bits in this key:

Table 3-1 PSKEY_DIGITAL_AUDIO_CONFIG bit fields

Bit No.	Description	Permitted Values
0	Justify format	Same purpose as that of "I^2S justify format" stream key. See Table 2-5. 0: left justified 1: right justified
1	Left justify delay	Same purpose as that of "I^2S left justify delay" stream key. See Table 2-5. 0: MSB of SD data occurs in the 1st SCLK period following WS transition. 1: MSB of SD data occurs in the 2nd SCLK period. This should always be set to 1 for standard I^2S format.
2	Channel polarity	Same purpose as that of "I^2S channel polarity" stream key. See Table 2-5. 0: SD data is left channel when WS is high. 1: SD data is right channel when WS is high. This should always be set to 1 for standard I^2S format.
3-7	NA	Must remain 0.
8-9	Resolution	Same purpose as that of "I^2S justify resolution" stream key. See Table 2-5.
10	Crop enables	Same purpose as that of "I^2S crop enable" stream key. See Table 2-5.
11-12	NA	Must remain 0.
13	TX_START_RISING	Same purpose as that of "I^2S TX start sample" stream key. See Table 2-5.
14	RX_START_RISING	Same purpose as that of "I^2S RX start sample" stream key. See Table 2-5.
15	NA	Must remain 0.

3.2.6 PSKEY_I2S2_MASTER_EN

Default value (Boolean): `FALSE (0)`

This key specifies if the second I^2S interface should operate in slave (`FALSE`) or master (`TRUE`) mode.

3.2.7 PSKEY_DIGITAL_AUDIO2_RATE

Default value (uint32): `0`

This key allows the specification (in Hz) of the exact clock rate to be generated when acting as a master on the second digital audio interface in I^2S mode.

If this key is set to 0, the clock rate is calculated based on the sync rate and number of bits per sample. See Appendix I²S Master clock rate derivation for more information.

3.2.8 PSKEY_DIGITAL_AUDIO2_BITS_PER_SAMPLE

Default value (uint16): 24

This key specifies the number of bits the second digital audio interface clocks per sample in I²S mode. If PSKEY_DIGITAL_AUDIO2_RATE is specified (that is, a non-zero value), then the value of this PS Key is ignored. See Appendix I²S Master clock rate derivation for more information.

3.2.9 PSKEY_I2S2_SYNC_RATE

Default value (uint32): 8000

This key specifies the sync rate for the second digital interface when operating as a master in I²S mode.

3.2.10 PSKEY_DIGITAL_AUDIO2_CONFIG

Default value (uint16): 0x0006

This key is a set of bit fields that specify more configuration details when operating the second digital interface in I²S mode. For more information, see BlueCore documentation specific to your device.

3.3 Codec PS Keys

The following PS Keys are used to configure the codec:

- PSKEY_CODEC_INPUT_RATE
- PSKEY_CODEC_OUTPUT_RATE
- PSKEY_CODEC_IN_GAIN
- PSKEY_CODEC_OUT_GAIN
- PSKEY_SIDE_TONE_ENABLE
- PSKEY_SIDE_TONE_GAIN
- PSKEY_SIDE_TONE_AFTER_ADC
- PSKEY_SIDE_TONE_AFTER_DAC
- PSKEY_CODEC_PIO
- PSKEY_CODEC_PIO_SETUP_TIME
- PSKEY_MIC_BIAS_LOW_POWER_MODE
- PSKEY_MIC_BIAS_PIN_VOLTAGE
- PSKEY_MIC_BIAS_PIN_CURRENT
- PSKEY_AUDIO_OUTPUT_POWER
- PSKEY_CODEC_OUT_DISABLE_WAITING_TIMEOUT

3.3.1 PSKEY_CODEC_INPUT_RATE

Default value (uint32): `8000`

This key specifies the sample rate used by the coded ADCs in Hz. Table 3-2 shows the permitted values.

Table 3-2 Codec input rate permitted values

BlueCore Version	Permitted Values
Pre-BlueCore5	8000
BlueCore5 and BlueCore6	8000 11025 16000 22050 24000 32000 44100
BlueCore7 onwards	8000 11025 12000 16000 22050 24000 32000 40000 44100 48000

3.3.2 PSKEY_CODEC_OUTPUT_RATE

Default value (uint32): `8000`

This key specifies the sample rate used by the codec DACs in Hz. Table 3-3 shows the permitted values.

Table 3-3 Codec output rate permitted values

BlueCore Version	Permitted Values
Pre-BlueCore5	8000
BlueCore5 and BlueCore6	8000 11025 12000 16000 22050 24000 32000 44100 48000
BlueCore7 onwards	8000 11025 12000 16000 22050 24000 32000 40000 44100 48000

3.3.3 PSKEY_CODEC_IN_GAIN

Default value (uint16): `8`

This key specifies the audio gain to be used by the codec ADCs when in use as a codec. Table 3-4 shows the interpretation of the value varies depending on the device.

Table 3-4 Codec input gain values

BlueCore Device	Bit Fields Description
BlueCore7-Multimedia	Bits [3:0]: Analog gain [(2)]: Range: 0-15 0: Max/Min attenuation 1: Unity 2: Min gain 15: Max gain Bits [7:4]: Digital gain: Range: 0-15 8: Max attenuation 15: Min attenuation 0: Unity 1: Min gain 7: Max gain Bit [8]: Enable the input pre-amplifier [(1)] Bits [10:9] [(1)]: Pre-amplifier gain (0 = unity)
Other devices with a codec	Bits [3:0]: Analog gain: Range: 0-15 0: Max attenuation 7: Min attenuation 8: Unity 9: Min gain 15: Max gain Bits [7:4]: Digital gain: Range: 0-15 8: Max attenuation 15: Min attenuation 0: Unity 1: Min gain 7: Max gain Bit [8]: Enable scaling down of DAC inputs [(2)]
NOTE [(1)] Not supported on BlueCore7-FM [(2)] Only supported on BlueCore4	

3.3.4 PSKEY_CODEC_OUT_GAIN

Default value (uint16):

- Pre-BlueCore7: `5`
- BlueCore7 onwards: `7`

This key specifies the audio gain to be used by the codec DACs when in use as a codec. Table 3-5 shows the interpretation of the value varies depending on the device.

Table 3-5 Codec output gain values

BlueCore Device	Bit Fields Description
BlueCore5	Bits [2:0]: Analog gain: Range: 0-7 0: Max attenuation 4: Min attenuation 5: Unity 6: Min gain 7: Max gain Bits [7:4]: Digital gain: Range: 0-15 8: Max attenuation 15: Min attenuation 0: Unity 1: Min gain 7: Max gain Bit [10]: Enable an extra 3 dB gain on DAC A Bit [11]: Enable an extra 3 dB gain on DAC B
BlueCore7	Bits [2:0]: Analog gain: Range: 0-7 0: Max attenuation 6: Min attenuation 7: Unity Bits [7:4]: Digital gain: Range: 0-15 8: Max attenuation 15: Min attenuation n 0: Unity 1: Min gain 7: Max gain **NOTE** In BlueCore7-FM, the codec ADC and DAC share the same analog gain. Therefore, Bits [2:0] set both codec analog input gain and output gain.

Table 3-5 Codec output gain values (cont.)

BlueCore Device	Bit Fields Description
Other devices with a codec	Bits [2:0]: Analog gain: Range: 0-7 0: Max attenuation 4: Min attenuation 5: Unity 6: Min gain 7: Max gain Bits [7:4]: Digital gain: Range: 0-15 8: Max attenuation 15: Min attenuation 0: Unity 1: Min gain 7: Max gain Bits [9:8]: Delta-sigma gain (0 = nominal)

3.3.5 PSKEY_SIDE_TONE_ENABLE

Default value (Boolean): `FALSE (0)`

This key specifies if the sidetone hardware should be enabled or disabled. This key applies only for devices that support sidetone.

3.3.6 PSKEY_SIDE_TONE_GAIN

Default value (uint16): `0`

This key specifies the sidetone gain. The supported values can range between 0 (minimum gain), Table 3-6 show the maximum gain available depending on the device.

Table 3-6 Sidetone gain permitted values

BlueCore Version	Permitted Values
Up to BlueCore5	0: Min gain 7: Max gain
BlueCore6	0: Min gain 9: Max gain
BlueCore7 onwards	0: Min gain 15: Max gain

3.3.7 PSKEY_SIDE_TONE_AFTER_ADC

Default value (Boolean): `FALSE (0)`

This key controls the sidetone source. Setting it to `FALSE` takes the sidetone signal from ADC output before applying the digital gain. Setting it to `TRUE (1)` takes the sidetone signal after applying the digital gain.

3.3.8 PSKEY_SIDE_TONE_AFTER_DAC

Default value (Boolean): `FALSE (0)`

This key controls the sidetone addition. Setting it to `FALSE` adds the sidetone signal before applying the digital DAC gain. Setting it to `TRUE (1)` adds the sidetone signal after applying the digital DAC gain.

3.3.9 PSKEY_CODEC_PIO

Default value (uint16): By default, this key is undefined. Therefore, it has no default value.

If defined, this key specifies the PIO (or alternatively the dedicated mic bias pin if present) that should be enabled when the built in the codec is enabled. The range of permitted values is defined by the `EnumMicBiasPin` enumeration.

3.3.10 PSKEY_CODEC_PIO_SETUP_TIME

Default value (TIME): 0

This key specifies the codec PIO setup time(in microseconds), which is the delay between enabling the audio stream and enabling the codec PIO line.

3.3.11 PSKEY_MIC_BIAS_LOW_POWER_MODE

Default value (Boolean):

- BlueCore7-Multimedia: `TRUE (1)`
- Other devices: `FALSE (0)`

This key controls the low power mode of the mic bias hardware for devices with a dedicated mic bias pin. Setting this key to `TRUE` enables the low power mode.

3.3.12 PSKEY_MIC_BIAS_PIN_VOLTAGE

Default value (uint16): `0`

This key controls the pin voltage level for devices with a dedicated mic bias pin. For more information, see BlueCore documentation specific to your device about the range of values supported and the voltage they specify.

3.3.13 PSKEY_MIC_BIAS_PIN_CURRENT

Default value (uint16): `0`

This key controls the pin current level. For more information, see BlueCore documentation specific to your device about the range of values supported and the current they specify.

3.3.14 PSKEY_AUDIO_ADC_DITHER

Default value (uint8): `0`

This key specifies an override to the default audio ADC dither setting for channels A or B. This key is read at audio initialization and set, if present. Permitted values are:

- 0: Channel A Off Channel B Off
- 1: Channel A On Channel B Off
- 2: Channel A Off Channel B On
- 3: Channel A On Channel B On

3.3.15 PSKEY_AUDIO_OUTPUT_POWER

Default value (uint8): `0`

This key defines the selection of output impedance and low power driver. It sets the default treatment for the codec outputs when they are active. This key is read at audio initialization and set, if present.

The values assigned from the enumeration are:

- 0 - 16 Ω output: Low power output driver disabled
- 1 - 32 Ω output: Low power output driver disabled
- 2 - 32 Ω output: Low power output driver enabled

3.3.16 PSKEY_CODEC_OUT_DISABLE_WAITING_TIMEOUT

Default value (TIME): 0x`1312D00(=20sec)`

This key specifies a DAC hold-on delay (in microseconds) for which the DAC will remain enabled even after the DAC stream or corresponding transform has been disconnected. The DAC is turned off after the timeout period if the Host does not use the same DAC channel again during hold-on period. This PS Key can be used to eliminate audio artifacts generated by common mode voltage changes on the input of external power amplifiers.

3.4 FM PS Keys

The following PS Keys are used to configure FM:

- PSKEY_FM_INPUT_RATE
- PSKEY_FM_OUTPUT_RATE

- PSKEY_FM_INPUT_GAIN
- PSKEY_FM_OUTPUT_GAIN

3.4.1 PSKEY_FM_INPUT_RATE

Default value (uint32): `8000`

This key specifies the sample rate at which the FM input streams operate during FM receive in Hz. Permitted values are:

- 8000
- 11025
- 12000
- 16000
- 22050
- 24000
- 32000
- 40000
- 44100
- 48000

3.4.2 PSKEY_FM_OUTPUT_RATE

Default value (uint32): `8000`

This key specifies the sample rate at which the FM output streams operate during FM transmit in Hz. Permitted values are:

- 8000
- 11025
- 12000
- 16000
- 22050
- 24000
- 32000
- 40000
- 44100
- 48000

3.4.3 PSKEY_FM_INPUT_GAIN

Default value (uint8): `1`

This key specifies the digital gain used by the codec ADC during FM receive.

The gain is specified in Bits [3:0] using 2's complement format with 0 representing unity gain.

3.4.4 PSKEY_FM_OUTPUT_GAIN

Default value (uint8): 0

This key specifies the digital gain used by the codec DAC during FM transmit.

The gain is specified in Bits [3:0] using 2's complement format with 0 representing unity gain.

3.5 SPDIF related PS Keys

The following PS Key holds the SPDIF configuration data:

- PSKEY_SPDIF_OUTPUT_RATE

3.5.1 PSKEY_SPDIF_OUTPUT_RATE

Default value (uint32): 44100

This key specifies the SPDIF interface output sample rate in Hz. Permitted values are:

- 32000
- 44100
- 48000

NOTE If SPDIF is used in both directions simultaneously, the incoming sample rate must match the specified output sample rate.

3.6 Digital Mic PS Keys

The following PS Keys are used to configure the digital mic:

- PSKEY_DIGITAL_MIC_INPUT_RATE
- PSKEY_DIGITAL_MIC_INPUT_GAIN
- PSKEY_CODEC_IN_QUALITY_MODE
- PSKEY_CODEC_OUT_QUALITY_MODE
- PSKEY_DIGITAL_MIC_x_PIOS
- PSKEY_DIGITAL_MIC_x_CHAN_SWAP
- PSKEY_DIGITAL_MIC_x_CLOCK_RATE
- PSKEY_DIGITAL_MIC_x_AMP_SEL

3.6.1 PSKEY_DIGITAL_MIC_INPUT_RATE

Default value (uint32): `8000`

This key specifies the digital mic input sample rate in Hz. Table 3-7 shows the range of supported values dependent on the device.

Table 3-7 Digital mic input rate supported values

BlueCore Version	Supported Values
BlueCore5	8000 11025 16000 22050 24000 32000 44100
BlueCore7 onwards	8000 11025 12000 16000 22050 24000 32000 40000 44100 48000

3.6.2 PSKEY_DIGITAL_MIC_INPUT_GAIN

Default value (uint16): `0x00`

This key specifies the default input gain that is used by the digital mic. The gain is specified in Bits [3:0] using two's complement format with 0 representing unity gain. The gain value applies to all the digital mic instances present. Permitted values for Bits [3:0] digital mic input gain are:

- 8: Max attenuation
- 15: Min attenuation
- 0: Unity
- 1: Min gain
- 7: Max gain

3.6.3 PSKEY_CODEC_IN_QUALITY_MODE

Default value (uint8): `2`

This key specifies the default quality mode for the codec input. Permitted values are:

- 0: Telephony mode
- 1: Normal mode
- 2: High mode
- 3: Bypass in Amp

3.6.4 PSKEY_CODEC_OUT_QUALITY_MODE

Default value (uint8): `2`

This key specifies the default quality mode for the codec output. Permitted values are:

- 0: Telephony mode
- 1: Normal mode
- 2: High mode

3.6.5 PSKEY_DIGITAL_MIC_x_PIOS

Default value (uint16): Replace `x` in this key with 0, 1 or 2 depending upon the digital mic instance to be configured.

This key defines the PIOs for the clock output and the data output for digital mic instance. Bits [7:0] specify the PIO used for the clock output and Bits [15:8] specify the PIO used for the data input. The PIO selected for the clock output must be an even-numbered PIO whereas the PIO selected for the data input must be an odd-numbered PIO. The value `NOT_MAPPED` (`0xff`) implies that the pin is not required whereas setting the key to `0xffff` is equivalent to not having the PS Key defined (that is, no PIOs are reserved for this mic instance).

3.6.6 PSKEY_DIGITAL_MIC_x_CHAN_SWAP

Default value (Boolean): `FALSE (0)`

This key swaps the rising and falling edge data for digital mic instance `x`. Replace `x` in this key with 0, 1 or 2 depending upon the digital mic instance to be configured.

3.6.7 PSKEY_DIGITAL_MIC_x_CLOCK_RATE

Default value (uint16): `2000` (that is, 2 MHz)

Replace `x` in this key with 0, 1 or 2 depending upon the digital mic instance to be configured. This key selects the clock rate for the digital mic instance `x`. This key supports four rates:

- 4000 (4 MHz)
- 2000 (2 MHz)
- 1000 (1 MHz)
- 500 (500 KHz)

3.6.8 PSKEY_DIGITAL_MIC_x_AMP_SEL

Default value (uint16): `0x0700`

Replace `x` in this key with 0, 1 or 2 depending upon the digital mic instance to be configured. The Bits [2:0] select the `LO_AMP_SEL` and the Bits [10:8] select the `HI_AMP_SEL`. The two values must be set symmetrical around the value of 3.5 (that is, 0,7 or 1,6 or 2,5 or 3,4 or 4,3 or 5,2 or 6,1 or 7,0). If the two values do not add to 7, DC appears at the codec output.

3.7 SCO Routing PS Keys

The following PS Keys are use to configure SCO routing:

- PSKEY_HOSTIO_MAP_SCO_PCM
- PSKEY_HOSTIO_MAP_SCO_CODEC
- PSKEY_HOSTIO_MAP_SCO_PCM_SLOT
- PSKEY_ENABLE_SCO_STREAMS

3.7.1 PSKEY_HOSTIO_MAP_SCO_PCM

Default value (Boolean): `FALSE` (except for devices that allow transport to be set from PIO state)

If the value of this key is `TRUE` (`1`), all SCO connections are routed over one of the PCM interfaces (if PSKEY_HOSTIO_MAP_SCO_CODEC is `FALSE`). The interface and the slot used are determined by PSKEY_HOSTIO_MAP_SCO_PCM_SLOT.

If the value of this key is `FALSE` (`0`), SCO routing is determined by the value of PSKEY_ENABLE_SCO_STREAM or one of the SCO related BlueCore commands described earlier in this document.

3.7.2 PSKEY_HOSTIO_MAP_SCO_CODEC

Default value (Boolean): `FALSE (0)`

If the value of this key is `TRUE` (`1`), and the value of PSKEY_HOSTIO_MAP_SCO_PCM is `TRUE`, all SCO connections are routed through the built in audio codec instead of the PCM interface.

3.7.3 PSKEY_HOSTIO_MAP_SCO_PCM_SLOT

Default value (uint16): `0`

This key specifies the PCM interface and the slot used when the first SCO connection opens over a device's PCM interface. When the first SCO connection is still active, all subsequent attempts to open additional SCO connections fail.

The values 0, 1, 2 and 3 refer to the four slots in the first PCM interface. Similarly, the values 4, 5, 6 and 7 refer to the four slots in the second PCM interface (if present).

It is important to configure the slot specified for this key appropriately. For example, if the key is set to `3`, ensure that the clock and the sync for the first PCM interface is configured in such a way that all four slots are present.

3.7.4 PSKEY_ENABLE_SCO_STREAMS

Default value (Boolean); `FALSE (0)`

This key specifies the initial value of SCO streams handling. It gives the initial value of the flag that `enable_sco_streams` command changes. When this key is set to `FALSE`, a SCO connection with no routing details is routed to HCI. When this key is set to `TRUE` (`1`), the SCO's stream handles are given to the host.

NOTE See Appendix SCO routing derivation for a schematic overview of SCO routing derivation.

A PCM master clock rate derivation

NOTE The information in this appendix relates to configuring the first digital interface. References to any PS Keys should be changed accordingly to configure the second digital interface.

Figure A-1 shows the master clock rate derivation when a transform is created, which contains a PCM source ID or a sink ID configured as master:

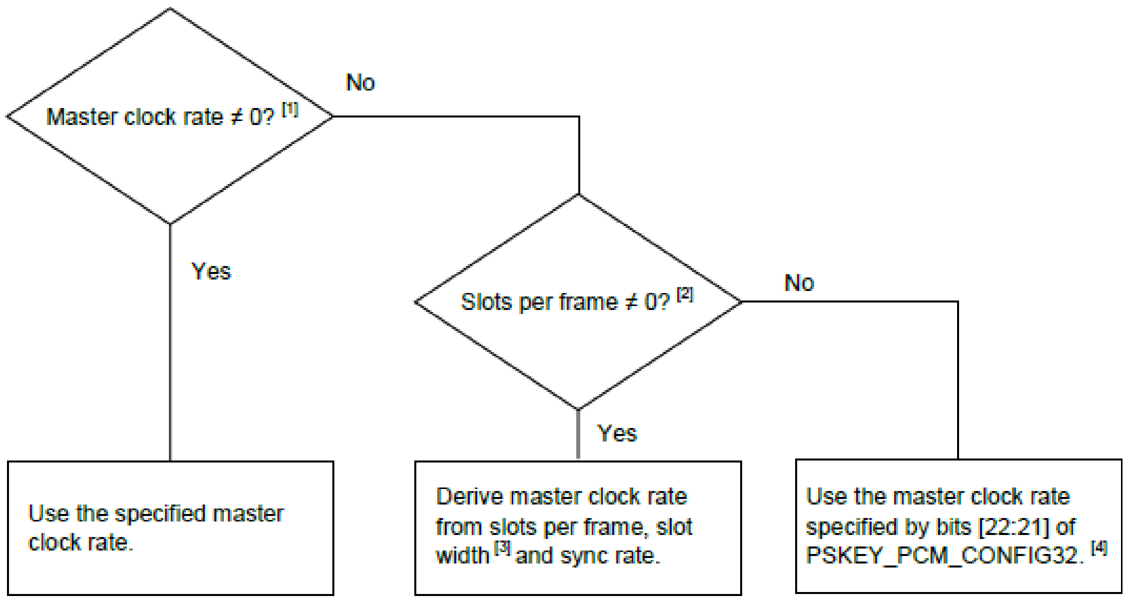

Figure A-1 PCM master clock rate derivation

[1] Non-zero master clock rate specified by the PSKEY_PCM_CLOCK_RATE or the `stream_configure` BlueCore command using the key `0x0101` (PCM master clock rate).

[2] Non-zero number of slots specified by the PSKEY_PCM_SLOTS_PER_FRAME or the `stream_configure` BlueCore command using the key `0x0103` (PCM slot count).

[3] Slot width is part of the PCM sample format by Bits [28:27] of PSKEY_PCM_CONFIG32.

Bit [28]	Bit [27]	PCM Sample Format
0	0	13 bits in a 16-bit slot
0	1	16 bits in a 16-bit slot
1	0	8 bits in a 16-bit slot
1	1	8 bits in an 8-bit slot

Slot width is specified with the `stream_configure` BlueCore command using the key `0x0114` (PCM sample format).

[4] When no specific master clock rate or slot count has been specified, the master clock rate specified by Bits [22:21] of PSKEY_PCM_CONFIG32 are used to determine the master clock rate.

The master clock rate specified by Bits [22:21] of PSKEY_PCM_CONFIG32 determines the master clock rate in the absence of a specific master clock rate or count.

Bit 22	Bit 21	Master Clock Rate
0	1	128 kHz
0	0	256 kHz
1	0	512 kHz

B I^{2}S Master clock rate derivation

The information in this appendix relates to configuring the first digital interface. References to any PS Keys should be changed accordingly to configure the second digital interface.

Figure B-1 shows the master clock rate derivation when a transform is created, which contains an I^{2}S source ID or a sink ID configured as master:

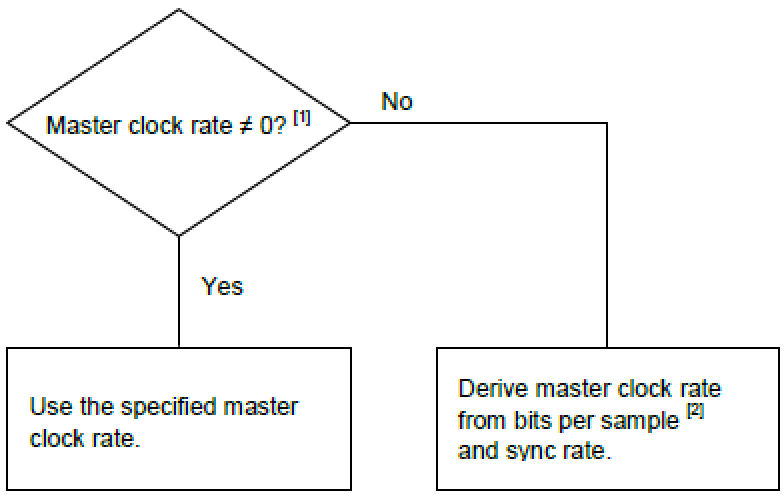

Figure B-1 I^{2}S Master clock rate derivation

[1] Non-zero master clock rate specified by the PSKEY_DIGITAL_AUDIO_RATE or the `stream_configure` BlueCore command using the key `0x0201` (I^{2}S master clock rate).

[2] The number of bits per sample is specified by the PSKEY_DIGITAL_AUDIO_BITS_PER_SAMPLE. It indicates the number of bits clocked per sample. If the number of bits per sample is larger than the internal audio format used by BlueCore, the additional bits are output as 0s in the least significant bits or ignored in an input.

C SCO routing derivation

Figure C-1 shows the SCO routing derivation.

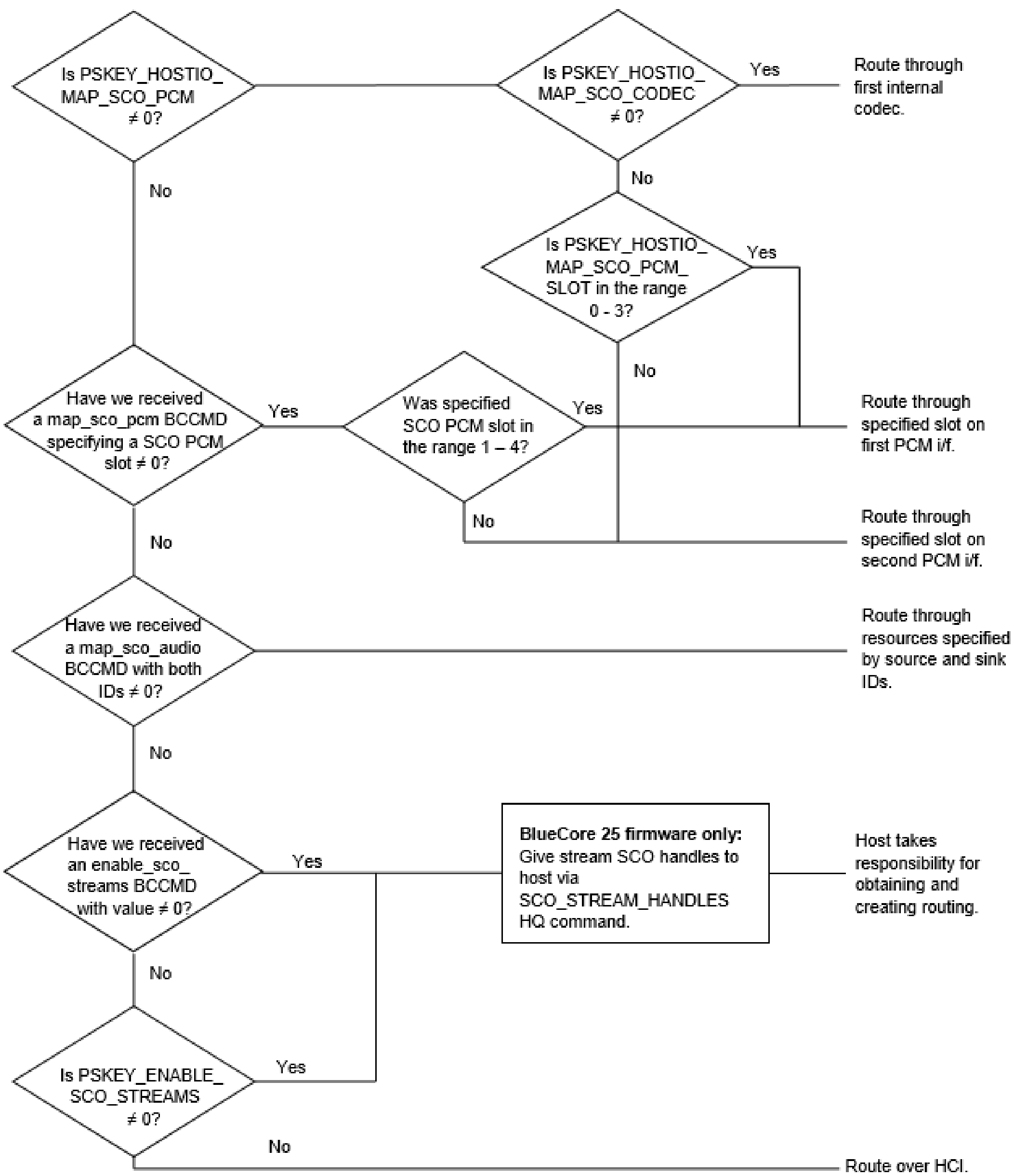

Figure C-1 SCO routing derivation

Document references

Document	Reference
HQ and BCCMD Commands and Protocols	80-CT714-1/CS-00227432-SP
BlueCore-FM API	CS-00101761-SP
DSPManager Specification	80-CT670-1/CS-00208512-SP

Terms and definitions

Term	Definition
ADC	Analog-to-digital converter
AMP	Amplifier
API	Application Programming Interface
BCCMD	BlueCore Command
BlueCore	Group term for the QTIL range of Bluetooth wireless technology chips
Bluetooth	Set of technologies providing audio and data transfer over short-range radio connections
BTCli	Bluetooth Command Line Interface
Client	Software task
CLK	CLocK or Clock cycle
CODEC	COder DECoder
DAC	Digital-to-analog Converter
DC	Direct Current
DSP	Digital Signal Processor
FIR	Finite Impulse Response (filter)
FM	Frequency Modulation
GCI	General Circuit Interface
HCI	Host Controller Interface
I^2S	Inter-Integrated circuit Sound
ID	Identifier
LS	Least Significant
LSB	Least significant Bit
Mic	Microphone
MSB	Most significant bit
PCM	Pulse Code Modulation
PIO	Programmable Input Output
PS Key	Persistent Store Key
QTIL	Qualcomm Technologies International, Ltd.
RX	Receive or Receiver
SCO	Synchronous Connection-Oriented

Term	Definition
SPDIF	Sony / Philips Digital Interface
TX	Transmit or Transmitter