IA61x – Smart Microphone

Advance API Guide

Version V1.2.89

External Version
For Customers and Partners

Copyright and Trademark Information

Contact Information

Knowles Electronics, LLC
Corporate Headquarters
1151 Maplewood Drive
Itasca, Illinois 60143
USA

Phone: (630) 250-5100

Fax: (630) 250-0575

sales@knowles.com

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

This document serves as the user guide for the API interfaces available for the IA61x – Smart Microphone. This manual is divided into these sections:

- Chapter 1: provides an introduction to the parts and the manual
- Chapter 2: describes Pin configurations
- Chapter 3: describes the Power Up and Boot Sequence
- Chapter 4: describes the Sleep and Wake-Up Sequence
- Chapter 5: describes the Small Bootloader (SBL) and the process required to download the binary files over each of the four interfaces (UART, I2C, SPI, and SoundWire).
- Chapter 6: provides information on the protocol used to send Commands and receive Responses over each of the four interfaces (UART, I^2C, SPI, and SoundWire).
- Chapter 7: provides expanded information on sending Commands and receiving responses over the SoundWire interface.
- Chapter 8: provides information on setting up audio use cases within the device with both routing rules and routing examples.
- Chapter 9: describes the syntax of the API commands, the response codes
- Chapter 10: provides detailed information on the Boot API Commands.
- Chapter 11: provides detailed information on the System API Commands.
- Chapter 12: provides detailed information Audio Path Commands.
- Chapter 13: provides detailed information on the Diagnostic API Commands.

1.1 Overview

The IA61x is an "always-on" smart microphone featuring Audience® Voice Wake and Voice ID keyword detector and Knowles' proven high performance acoustic SiSonic™ MEMS technology in a single, miniature, bottom-port/top-port package. The IA61x offers flexibility by supporting the command and control interfaces over I²C/UART/SPI/SoundWire and audio data interface over PCM/ I²S/PDM/SoundWire. Its integrated programmable audio-optimized DSP is available to write various kinds of algorithms, which can run the code while being ultra-low power on the bus. It also includes a System Control Unit (SCU) that handles power management states such as sleep mode and generates internal clock signals required to run the IA61x without external clock input. The IA61x is optimized for low power operation in a wide variety of devices including mobile devices, laptops, tablets etc.

Key features of the IA61x processor are:

- Ultra-Low power, Best-in-class Audience Voice Wake: Waits with DSP in sleep mode for acoustic activity before going into Audience Voice Wake Mode
- Audience Voice Wake: Best-in-class Always-on keyword detection in ultra-low power mode. Detection of either pre-programmed (OEM), user-trained keywords or voice ID based keywords.
- Audience Continuous Voice Wake (CVQ) for seamless transition from Audience Voice Wake to a command phrase that follows.
- SoundWire digital audio and control interface support (IA61x SoundWire package)
- SPI Slave interface for fast code download and control (IA61x general package)
- Low power up to 40 MHz available for OpenDSP applications (3rd Party)

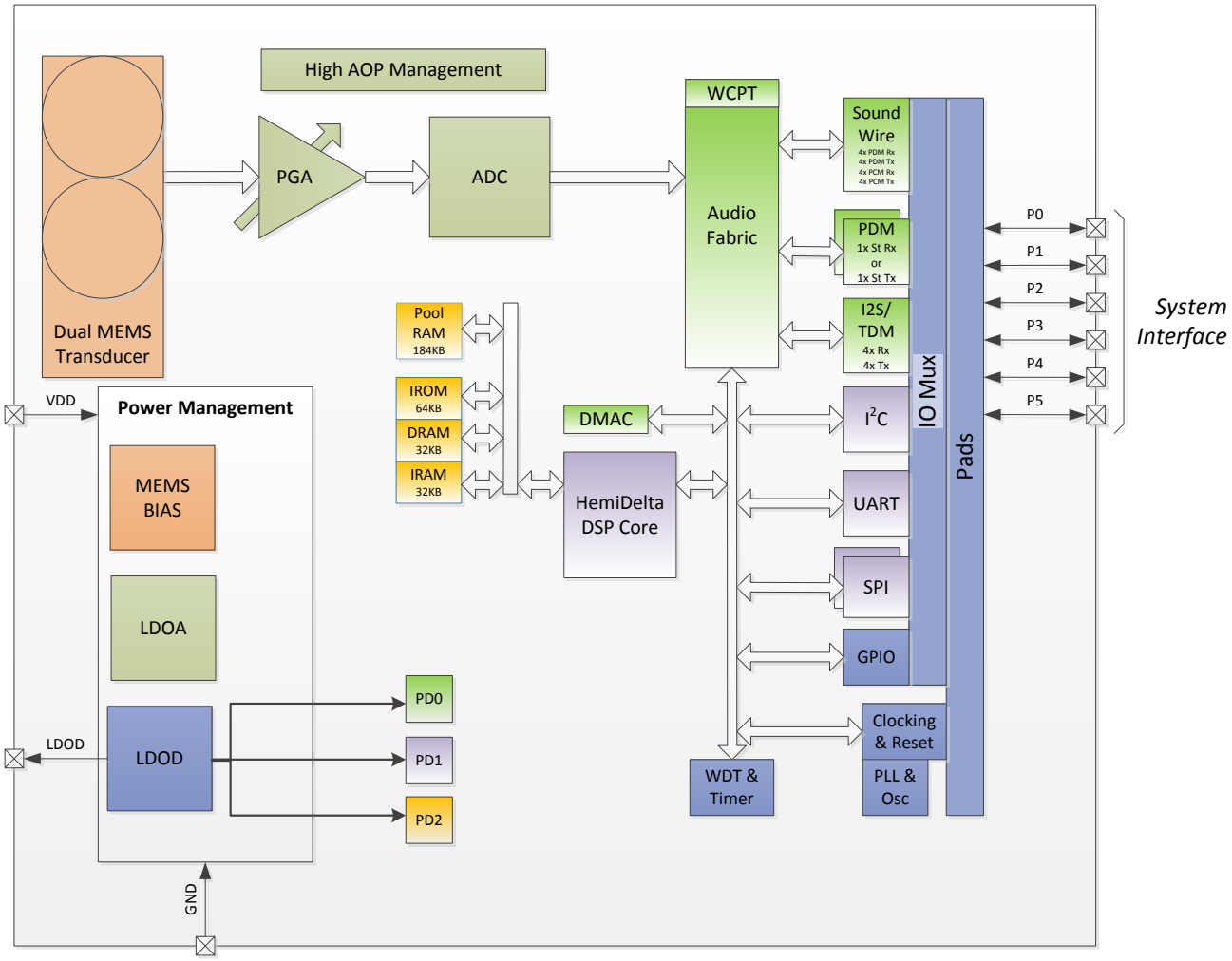

Figure 1 *IA61x System Diagram*

1.2 The IA61x Product Family

The IA61x product family consists of the parts shown in **Error! Reference source not found.**.

Table 1 *IA61x Product Family*

Product	Features
IA610	5th Generation Audio Processor with Voice Wake, PDM and Pass-through, Bottom port
IA611	5th Generation Audio Processor with Voice Wake, PDM and Pass-through, Top port
IA61xW	5th Generation Audio Processor with SoundWire

All of the commands and syntax in this manual apply to all of the parts in the family, except where specifically noted.

These (and any future parts on the family) are referred to as the IA61x in the text of the document, except where they apply to only one part in the family. Then they are referred to by their specific part number. However, the generic number for the part family is IA61x, and this is how the part is referred to in the software listings.

1.3 References

- MIPI Alliance, Specification for SoundWire℠, version 1.0, January 21, 2015.

1.4 Abbreviations

AAD	Acoustic Activity detected
ACK	Acknowledge
BAF	Bosko Algorithm Framework
BRA	Bulk Register Access
API	Application Programming I/F
DHWPT	Digital Hardware Pass-through
DI	Digital Input pin
DO	Digital Output pin
DRAM	Data RAM
DROM	Data ROM
EQ	Equalizer
GPIO	General Purpose Input/output
I2C (I^2C)	Inter-Integrated Circuit Bus
I2S (I^2S)	Inter- Integrated Circuit Sound
ID	Identifier
IRAM	Instruction RAM
IROM	Instruction ROM
KBPS	Kilobits Per Second
LSB	Least Significant Bit (of data word)
MBPS	Megabits Per Second
MISO	Master In Slave Out
MIPI	Mobile Industry Processor Interface
MIPS	Millions of Instructions Per Second
MOSI	Master Out Slave In
OS	Operating System
PC	Program Counter
PCM	Pulse Coded Modulation
PDM	Pulse Density Modulation
PRI	Primary Mic Input
POR	Power-On Reset
RAM	Random-Access Memory
ROM	Read-Only Memory
SBL	Small Boot Loader
SPI	Serial Peripheral Interface
TBD	To Be Determined
UART	Universal Asynchronous Receiver Transmitter
VAD	Voice Activity Detection

Chapter 2: Pin Configuration

The IA61x has four control interfaces: UART, I²C, SPI, and SoundWire. The IA61x also supports four audio interfaces: PDM, I²S, PCM, and SoundWire. It has only six pins to support all of these control/audio interfaces, so the IA61x has pin multiplexing to choose between combinations of the interfaces. The IA61x can be configured in six valid modes as listed in **Error! Reference source not found.**.

Table 2 *The IA61x Pin modes*

No.	Mode	P0	P1	P2	P3	P4	P5	IRQ	Wake
1	PDM + I²C	pdm_clk	pdm_sdo pdm_sdi*	I2c_addr1 wake	I2c_addr2 irq	i2c_sclk	i2c_sda	P3	P2
2	PDM + UART	pdm_clk	pdm_sdo pdm_sdi*	pdm_sdi*	Irq	uart_rx wake	uart_tx irq*	P3 P5*	P4
3	PDM + SPI	pdm_clk	pdm_sdo irq pdm_sdi*	spi_sclk	spi_miso	spi_ss wake	spi_mosi	P1/P3	P4
4	PDM + SoundWire	pdm_clk	pdm_sdi	sw_clk	sw_data	i2c_sclk sw_addr1 wake	i2c_sda sw_addr2 irq	P5	P4
5	I²S + I²C	i2s_ws	i2s_clk	i2s_sdi wake	i2s_sdo irq	i2c_sclk	i2c_sda	P3	P2
6	I²S + UART	i2s_ws	i2s_clk	i2s_sdi	i2s_sdo irq	uart_rx wake pdm_sdo*	uart_tx pdm_clk* irq*	P3 P5*	P4
7	I²S + PDM	pdm_clk	pdm_sdo	i2s_sdi wake	i2s_sdo irq	i2s_clk	i2s_ws	P3	P2

* These are special pad configuration, which can be configured using SysConfig parameters, refer SysConfig API guide for more information.

Note:

1. In PDM+SPI, IRQ line can be configurable from SysConfig variable spiHostIrq. In case of SPI_MISO as IRQ, GPIO IRQ will clear after DSP core receives first command.
2. In I2S+UART or PDM+UART, IRQ line can be configurable from SysConfig variable uartHostIrq. If enabled UART_Tx line will act as IRQ.

Table 3 *IA61x Pin description*

Pin name	Pin Direction	Description
pdm_clk	input	PDM clock
pdm_sdo	output	PDM data out
pdm_sdi	input	PDM data in

Pin name	Pin Direction	Description
i2s_clk	input	I²S bit clock
i2s_ws	input	I²S word select
i2s_sdo	output	I²S data out
i2s_sdi	input	I²S data in
i2c_sclk	input	I²C clock
i2c_sda	input/ output	I²C data
i2c_addr1[1]	input	I²C address bit. This pin is latch on reset.
I2c_addr2[1]	input	I²C address bit. This pin is latch on reset.
uart_rx	input	UART receive data line
uart_tx	output	UART transmit data line
spi_sclk	input	SPI clock
spi_ss	input	SPI slave select
spi_mosi	input	SPI master out slave in
spi_miso	output	SPI master in slave out
sw_clk	input	SoundWire clock
sw_data	input/ output	SoundWire data
sw_addr1[2]	input	SoundWire address bit. This pin is latch on reset.
sw_addr2[2]	input	SoundWire address bit. This pin is latch on reset.
irq	output	Interrupt from the IA61x to Host
wake	input	Interrupt from Host to the IA61x

Note 1: Latch on Reset Pins for I²C addresses.
Note 2: Latch on Reset Pins for SoundWire addresses

2.1 I²C Slave Address

The IA61x supports four 7-bit I²C slave addresses. The I²C slave address is determined based on the states of Latch-on-Reset pins P2 and P3 during power-on reset. **Error! Reference source not found.** depicts all possible I²C slave addresses.

Table 4 *I²C Slave Address*

P3 Pin LOR Value	P2 Pin LOR Value	7 bit Address
0	0	0x3E
0	1	0x3F
1	0	0x38
1	1	0x39

Note: In SoundWire package I²C address is fixed to 0x3E.

2.2 SoundWire Device Address

SoundWire device has 48-bit unique device address. The IA61x SoundWire device address is as per **Error! Reference source not found.**.

Table 5 *SoundWire Device Address*

Register	Device ID bit Position	Content	Value
SCP_DevId_0 [7:4]	47:44	Version	0x1
SCP_DevId_0 [3:0]	43:40	UniqueID	See **Error! Reference source not found.**
SCP_DevId_1	39:32	Manufacturer ID [15:8]	0x01
SCP_DevId_2	31:24	Manufacturer ID [7:0]	0x91
SCP_DevId_3	23:16	Part ID [15:8]	0x06
SCP_DevId_4	15:8	Part ID [7:0]	0x10
SCP_DevId_4	7:0	Class	0x00

Up to four IA61x Smart Microphones can be connected in a system, and their SoundWire UniqueID will be determined by the LOR value on pins P4 and P5. **Error! Reference source not found.** shows all possible UniqueID values.

Table 6 *SoundWire UniqueID*

P5 Pin LOR Value	P4 Pin LOR Value	UniqueID
0	0	0x00
0	1	0x01
1	0	0x02
1	1	0x03

Chapter 3: Power up and Boot Sequence

Error! Reference source not found. shows the full IA61x boot sequence, including power-up and code (firmware image) download. For more information on the power up and boot sequence, see **Error! Reference source not found. Error! Reference source not found.**.

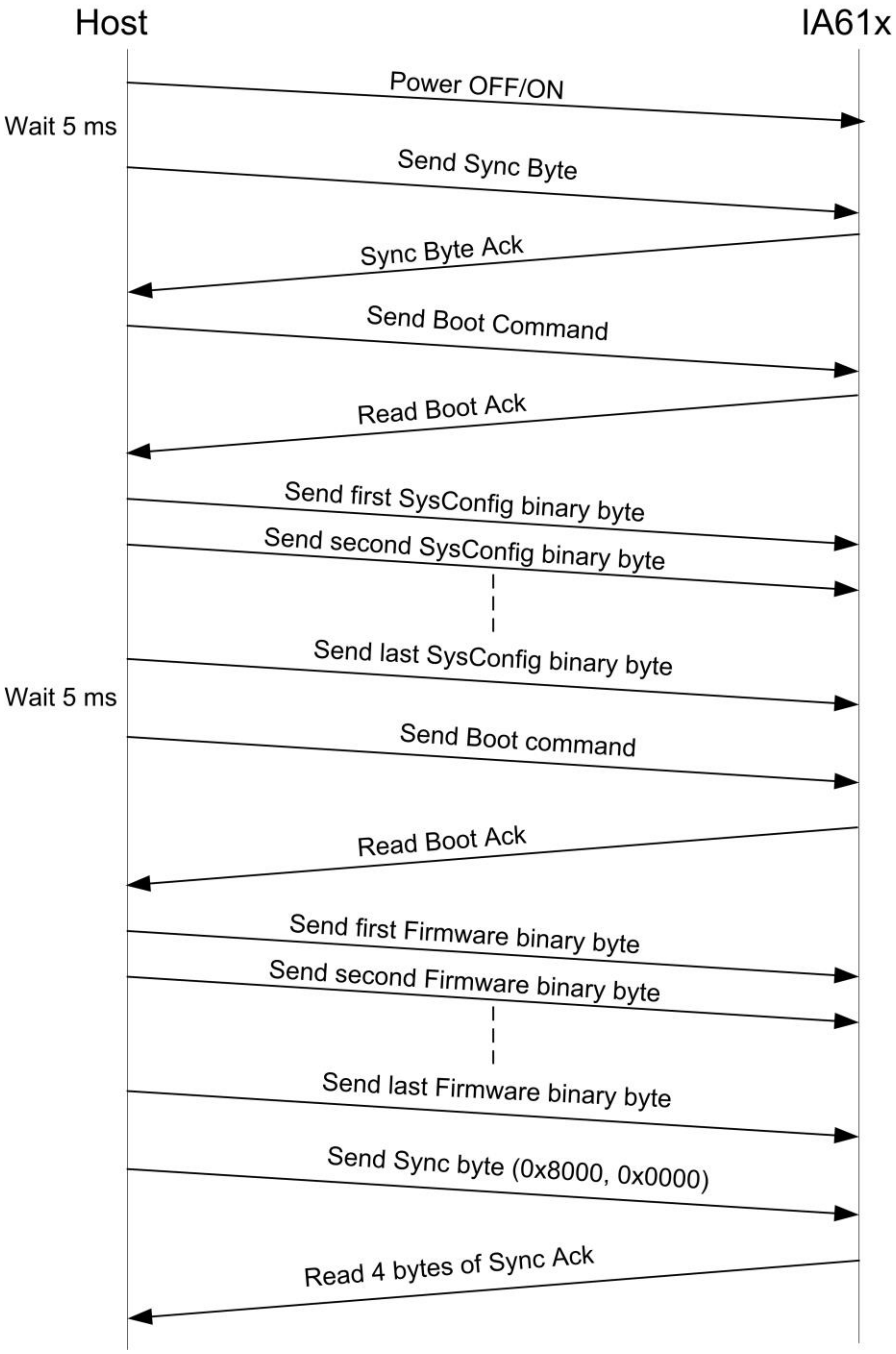

Figure 2 *Power up and Boot Sequence*

The IA61x supports Test Bypass mode at the Small Boot Loader (SBL) level. The IA61x works as dumb microphone in this mode. The SBL monitors the P0 (PDM Clock) pin and activates Test Bypass mode when PDM clock > TBD kHz. The Host needs to follow the sequence in **Error! Reference source not found.** to activate Test Bypass mode.

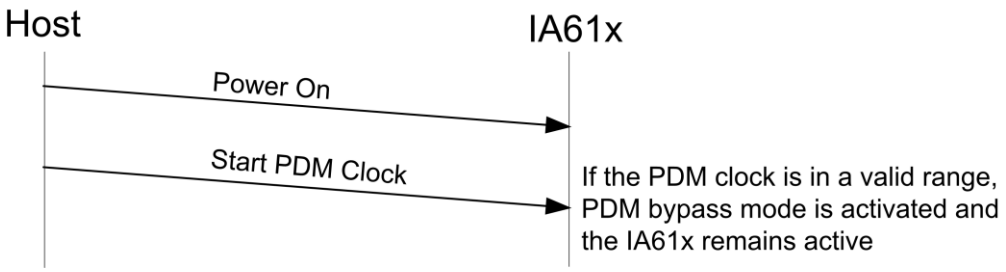

Figure 3 *Test Bypass Mode Sequence*

Chapter 4: Sleep and Wake-up Sequence

Error! Reference source not found. shows the sleep wakeup sequence for the IA61x.

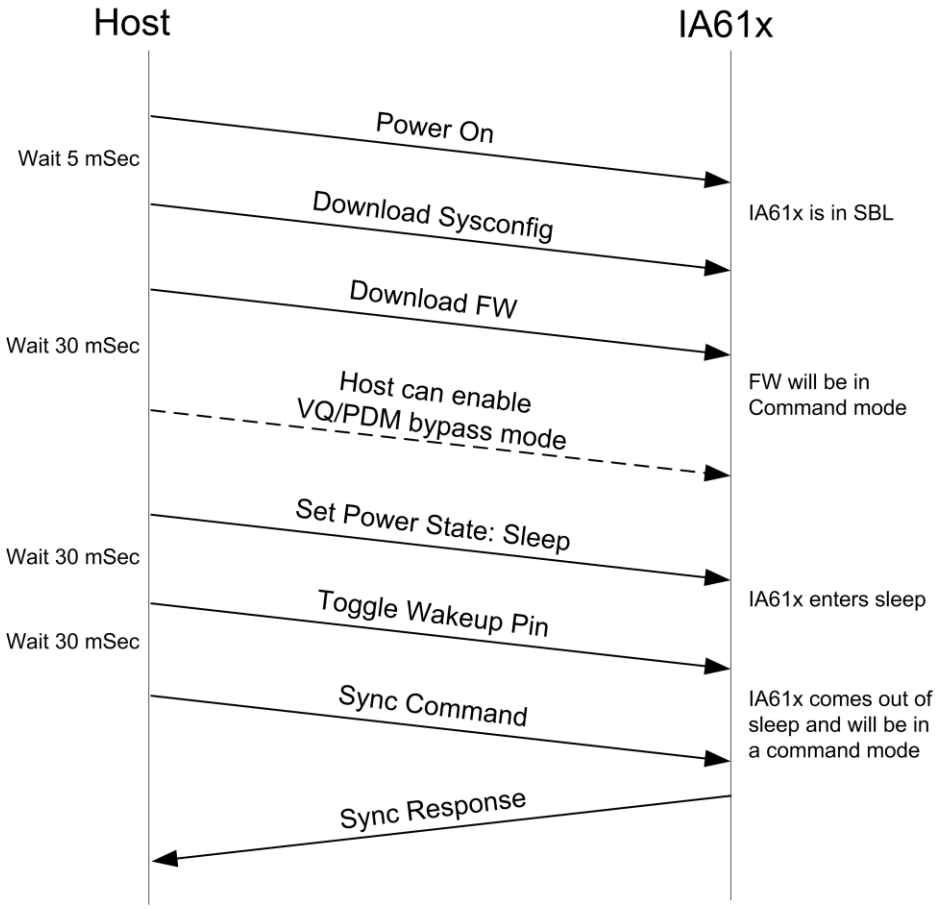

Figure 4 *Sleep Wakeup Sequence*

Note: To wake up the chip, the host needs to toggle the signal on the wake-up pin. The chip wakes up on a high to low transition.

In UART mode, the host can wake up the IA61x by either sending any byte over an interface or by toggling the uart_rx pin P4. The Wakeup pulse low period should be less than 1.5 mSec.

Wakeup from Sleep mode and low power mode will land in FW. However, wakeup from Deep Sleep mode will land in SBL. Therefore, host needs to download the FW after Deep Sleep mode.

PDM bypass mode in SBL will not be available after deep sleep mode unless host brings back the Internal MIC to normal mode. To bring Internal MIC out of low power, the host can send the following commands in SBL, only after deep sleep wakeup. These commands should not be sent after FW download as FW download would automatic bring Internal MIC out of low power mode.

Table 7 *Commands to bring back Internal MIC to Normal mode*

Order of execution	P4 Pin LOR Value
1	0x80360014
2	0x8037000a
3	0x80380002
4	0x80390000

Below figure shows the deep sleep and wakeup sequence for the IA61x.

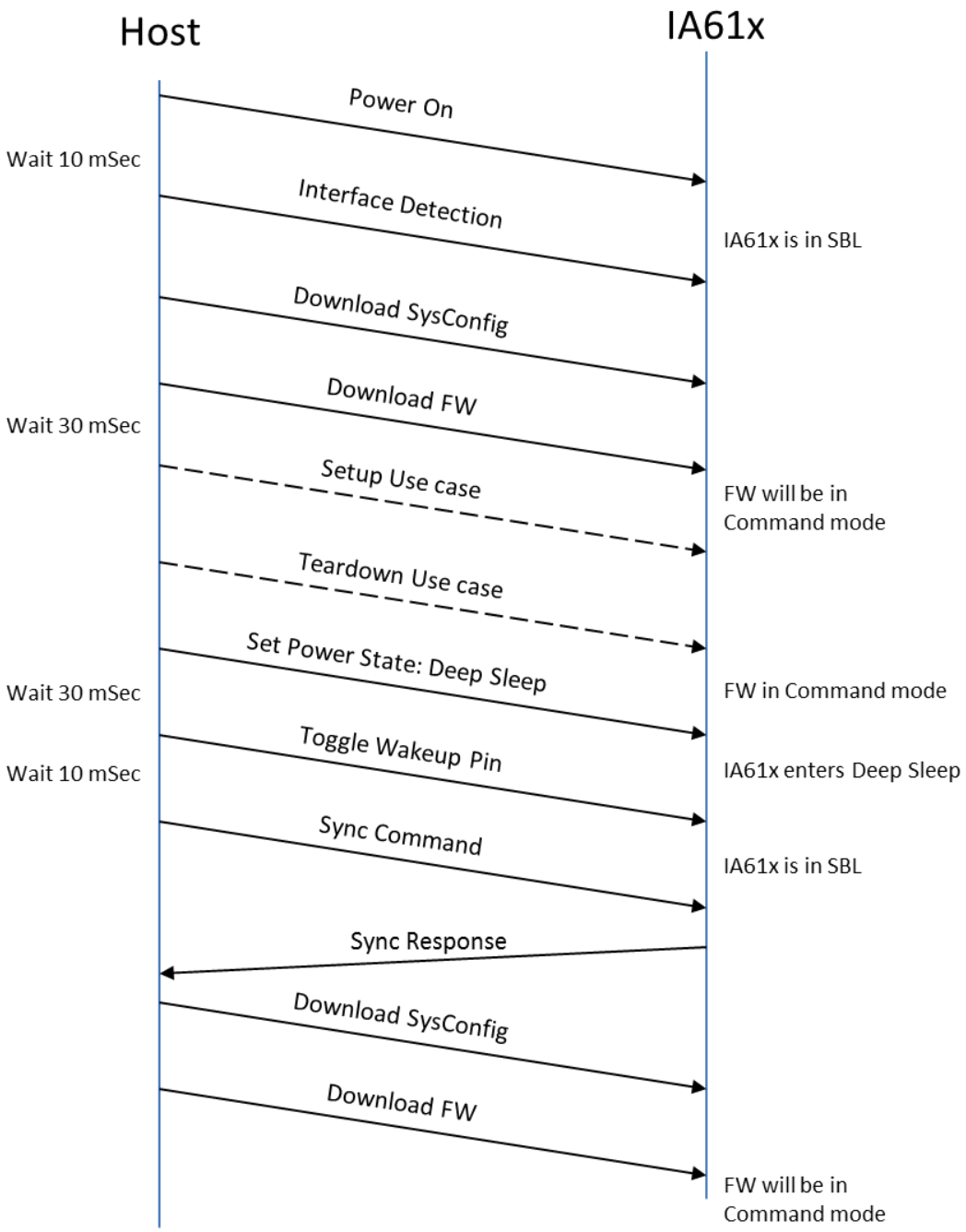

Figure 5 *Deep Sleep-Wake up Sequence*

Chapter 5: Code Download

The Small Boot Loader (SBL) supports firmware download over I^2C, UART, SPI, I^2S, and SoundWire. The IA61x automatically detects the host control interface, which can be I^2C, UART, SPI, or SoundWire.

Note: Firmware images and system configuration images are treated exactly the same from a host perspective. Because of this, subsequent sections of this document will not refer to firmware or system configuration images explicitly. These will be referred to interchangeably as a binary image.

The System configuration image must be sent before the Firmware image.

If the binary contains a start address, the SBL will automatically start running code at that address. If the binary does not contain a start address, then the SBL will simply return to its command loop, awaiting another download (or a boot command).

5.1 Bootloader Auto-detect State

To initiate the bootloader, the Host Controller must place the IA61x – Smart Microphone in reset. This is done by power cycling (off/on) the IA61x. Once powered up, the IA61x bootloader goes into Auto-detect state to detect the host control interface.

The IA61x looks for a Sync Byte after power up. If the control interface is UART then an Autobaud byte should be sent by the Host before the Sync Byte. After the sync byte, the host can send a Boot Command or an API command over the control interface. After a Boot Command is sent to the IA61x, the host must send the firmware binary only, and not any other API command.

In the UART download section **Error! Reference source not found.** **Error! Reference source not found.**, specifies the UART baud rates that are supported by the SBL.

The Autobaud Bytes is a two-byte command (0x00 0x00) for the UART interface.

The Sync byte is a one-byte command (0xB7) for the 8-bit UART interface, a one byte command (0xB7) for the I^2C interface, or a four byte command (0xB7B7B7B7) for the 32-bit SPI interface. A Sync command is not required for the SoundWire interface.

The SPI interface also supports boot commands 0xB8B8B8B8 and 0xB9B9B9B9 that correspond to different SPI sample configurations. Contact your Knowles Applications Engineer for your more information.

The Boot Command is a one-byte command (0x01) for the 8-bit UART interface, a one-byte command (0x01) for the I^2C interface, a four-byte command (0x00000001) for the 32-bit SPI interface, and a four-byte command (0x00000001 for the Control Port, 0x00000004 for Data Port download and 0x00000005 for advanced Bulk Register Access (BRA) download, 0x00000006 for BRA Legacy) for the SoundWire interface.

When the Boot command is sent to one of the IA61x supported interfaces, the IA61x responds with a Boot Ack. Boot Ack is a one-byte response (0x01) on the 8-bit UART interface, a one-byte response (0x01) on the I^2C interface, a four-byte response (0x00000001) on the 32-bit SPI

interface, and a four-byte response (0x00000001/0x00000004/0x00000005/0x00000006) on the 32-bit SoundWire interface.

Once a Boot Ack has been received, the Host can initiate the firmware download procedure. After the firmware download is complete, the IA61x will change to the Active State.

Note that the Host can terminate a firmware download at any time during the firmware download. To terminate the firmware download the host must power cycle (off/on) the IA61x.

5.2 Code Download over SoundWire

The IA61x microphone supports SoundWire as a host interface. The SoundWire interface enables high-speed downloads of the firmware from the Host Controller to the IA61x. This section provides the command sequence and supporting details required for downloading firmware to the IA61x using the SoundWire interface.

There are four different ways to download code over SoundWire:

1. Download over the Control Port
2. Download using Bulk Register Access (BRA) and a legacy binary
3. Download using BRA and an uncompressed binary (Advanced download)
4. Download over the Data port

Note: SoundWire allows command communication only on the Control Port. All commands and responses will be on the Control Port.

5.2.1 Code download over the SoundWire Control Port

The IA61x exposes the SoundWire Control port registers to the host shown in **Error! Reference source not found.**. The Host writes to or reads from these control port register to communicate with the IA61x SBL.

Table 8 *Control Port Addresses*

Byte	Read/Write	Address
Byte 1 (LSB)	Read	0x2004
Byte 2	Read	0x2005
Byte 3	Read	0x2006
Byte 4 (MSB)	Read	0x2007
Byte 1 (LSB)	Write	0x2000
Byte 2	Write	0x2001
Byte 3	Write	0x2002
Byte 4 (MSB)	Write	0x2003

The Host controller must download the system configuration image and then download the IA61x firmware using the SoundWire Control Port. The firmware image stored in the Host memory is already pre-formatted with the necessary flow control, and can be downloaded byte-by-byte by the Host starting with the first memory location and ending with the last. During code download, the firmware is downloaded in chunks of 4 bytes sent to the four register addresses shown in **Error! Reference source not found.**.

At the end of the firmware download, the Host must send a Sync command to verify the IA61x's status. Only if the download was successful does the IA61x acknowledge the Sync. Before sending the sync command, the slave needs to be re-enumerated.

If the IA61x does not acknowledge a Sync command, the Host must decide how many times to re-try before it power cycling the IA61x device.

The Host must start downloading the firmware image by sending a Boot command (0x00000001) to the IA61x on the Control Port, followed by an Ack read from the IA61x. A Boot Ack indicates that the IA61x is ready to receive the firmware image; otherwise, the IA61x must be power cycled and the Boot command should be sent again. These steps are common for SysConfig and BoskoApp binaries. Sync response after SysConfig download should be 0x8000_FFFF and after BoskoApp download should be 0x8000_0000. The process is shown in **Error! Reference source not found.**:

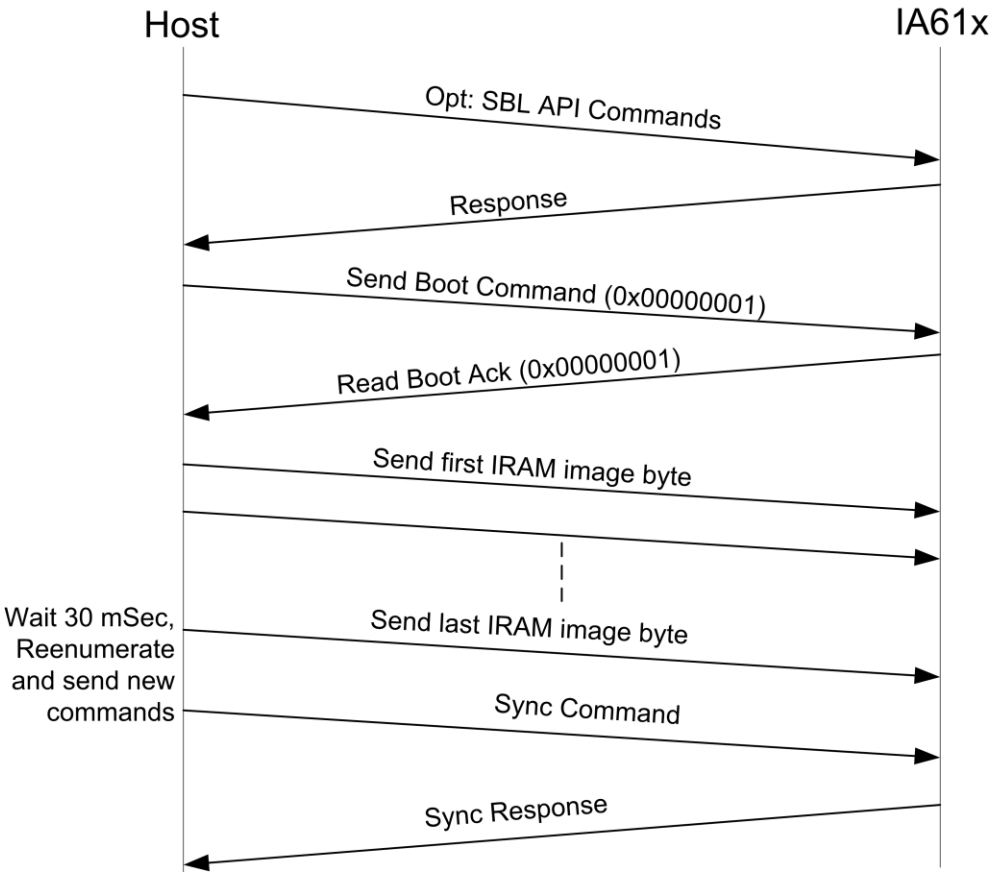

Figure 6 *Code Download Using SoundWire Control Port*

5.2.2 Code Download using BRA

Bulk Register Access (BRA) is a Bulk Transport Protocol (BTP) on Data Port 0 as defined in the MIPI specification. BRA is used to read/write 32-bit system addresses. The host acts as a BRA initiator and specifies the read/write address. For a more detailed description of the BRA protocol, refer to the MIPI specification.

Unlike other interfaces, the boot command for download using BRA needs to come in on the SoundWire Control port. To distinguish the download using BRA from a Control Port download, the boot command is changed to:

0x00000005: Use advanced download
0x00000006: Use Legacy download

5.2.2.1 Using legacy binary

The binary is split into BRA packets and sent to the IA61x on Data Port 0. The BRA packets should be a multiple of 4-bytes, as the packet is read in 4-byte chunks. The 32-bit BRA write address for a legacy binary download is 0x30000000.

Note: this address may change in future.

The Host must start downloading the firmware image by sending a Boot command (0x00000006) to the IA61x on the Control Port, followed by an Ack read from the IA61x on the Control Port. A Boot ACK indicates that the IA61x is ready to receive the firmware image; otherwise, the IA61x must be reset and the Boot command should be sent again. The host should then setup the BRA payload parameters and initiate a BRA write of the entire binary to the above specified address. Before sending any further commands, the slave needs to be re-enumerated. These steps are common for SysConfig and BoskoApp binaries. Sync response after SysConfig download should be 0x8000_FFFF and after BoskoApp download should be 0x8000_0000. The process is described in **Error! Reference source not found.**:

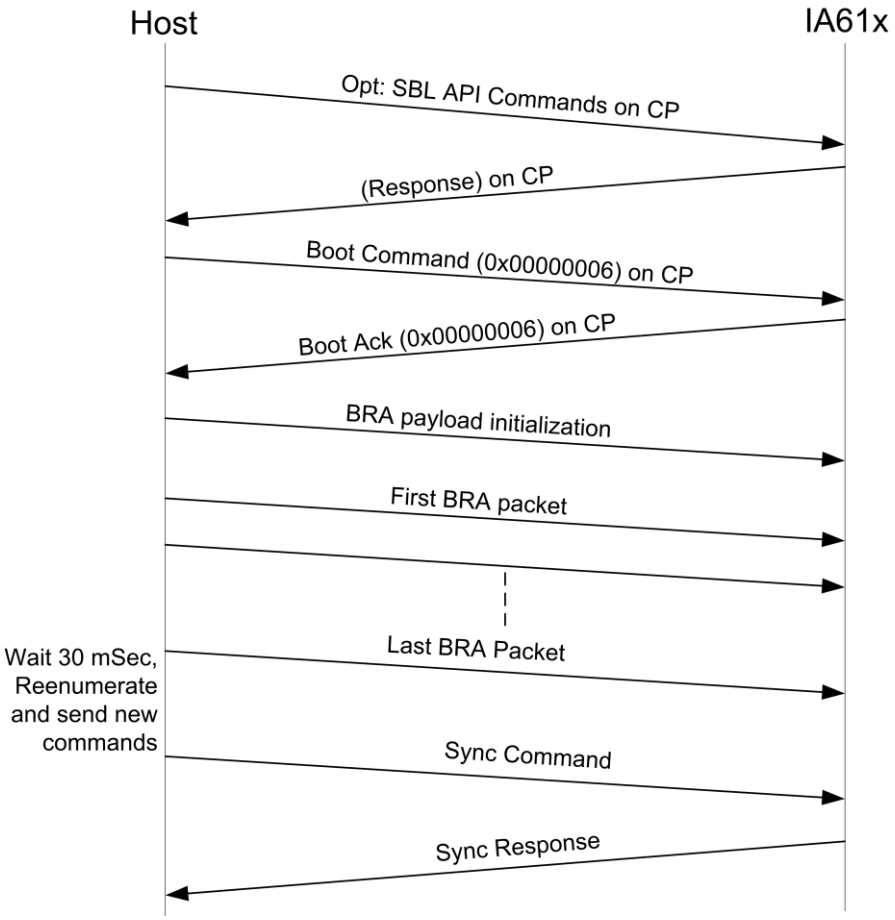

Figure 7 *Code Download Using SoundWire BRA (Legacy)*

While the download is happening, any data sent on the Control Port will be ignored.

5.2.2.2 Using Advanced BRA Download

This method needs the legacy binary to be decoded in the Host. Each binary block is preceded by a block header that specifies the RAM address for that section. The host needs to initiate BRA writes for each block with the actual RAM address. The bootloader reads the BRA packet and writes to the RAM address directly.

Since, in this method, the bootloader does not know the start address for execution, the Host needs to send the code start address and then indicate to the SBL to start executing from that address at the end of the code download. These instructions are sent as commands on Control Port. Before sending any further commands, the slave needs to be re-enumerated.

The BRA protocol uses CRC for verifying the veracity of the data. Because of this, there is no need for checksum checking in the bootloader. The SoundWire Master driver will do selective repeat of the BRA packets that incurred some kind of error. This all happens at the interface level; the Host and bootloader can be completely oblivious to it. These steps are common for SysConfig and BoskoApp binaries. Sync response after SysConfig download should be 0x8000_FFFF and after BoskoApp download should be 0x8000_0000. The sequence for code download is depicted in **Error! Reference source not found.**:

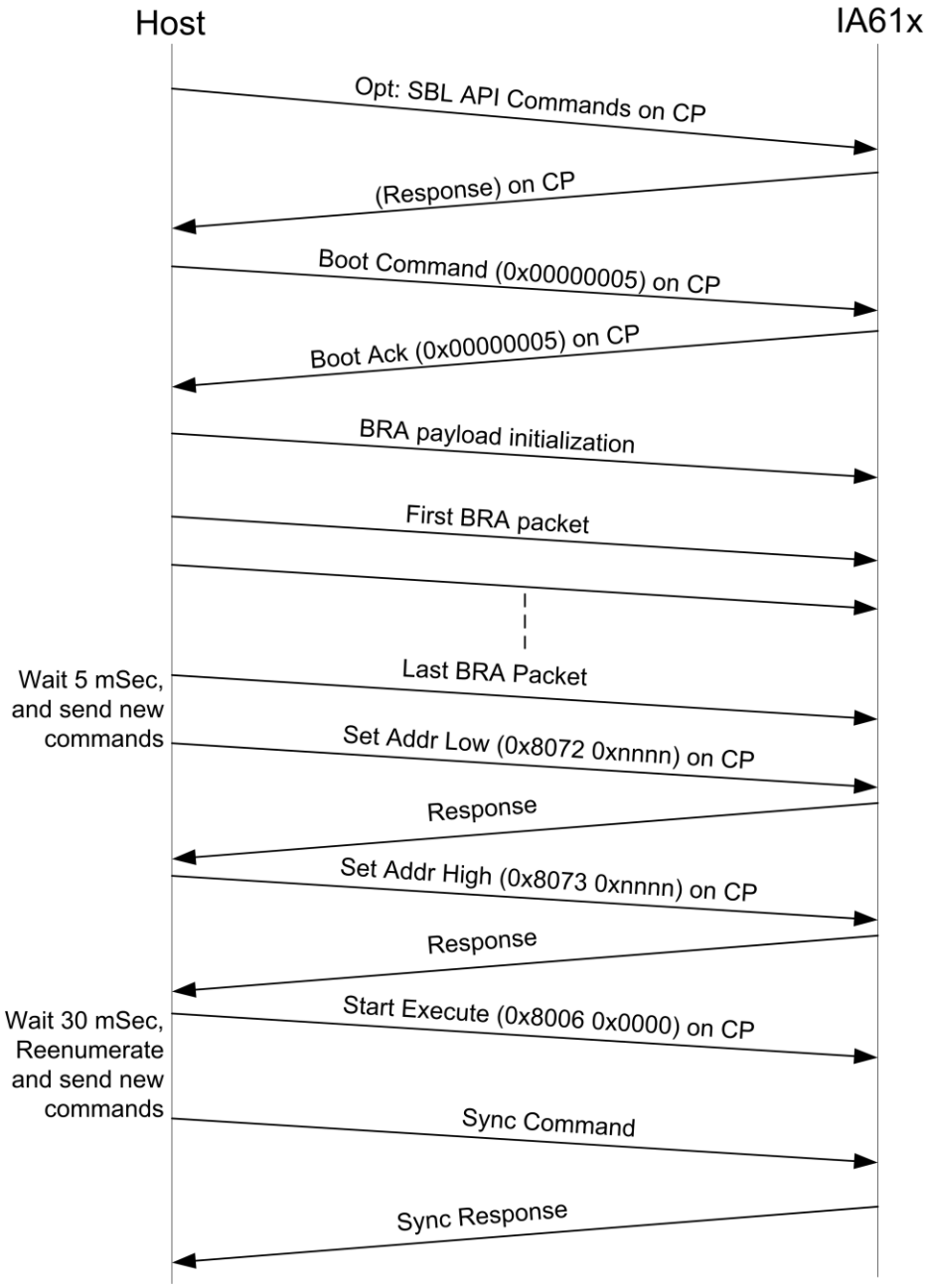

Figure 8 *Code Download Using SoundWire BRA (Advanced)*

5.2.3 Code Download over the SoundWire Data Port

The SoundWire data port for PCM can also be used download the code. The firmware image stored in the Host memory is already pre-formatted with the necessary flow control, and can be downloaded by the Host starting with the first memory location and ending with the last.

The SoundWire PCM Rx port (Data Port 3 Channel 1) is used to send data. The SoundWire PCM Rx port is an audio port with real-time data. If there is no data to send or respond those data elements will be of value 0x00000000. This is why, once the download has begun; there should be continuous data without any unnecessary zero elements in between. These steps are common for SysConfig and BoskoApp binaries. Sync response after SysConfig download should be 0x8000_FFFF and after BoskoApp download should be 0x8000_0000.

The sequence is shown in **Error! Reference source not found.**:

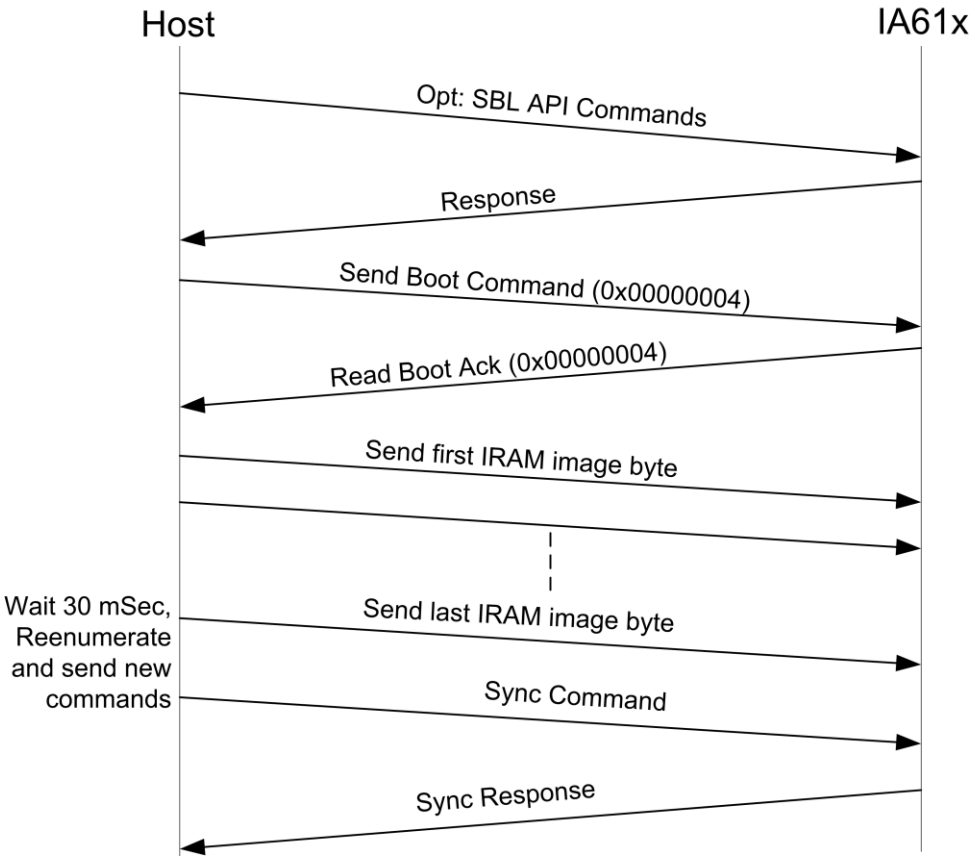

Figure 9 *Code Download Over the SoundWire Data Port*

5.3 Code Download over I²C

The IA61x API allows the Host controller to download the firmware into the IA61x's internal program memory over the I²C interface. The firmware image stored in the Host memory is already pre-formatted with the necessary flow control, and can be downloaded byte by byte by the Host, starting with the first memory location and ending with the last. During code download, the 32-bit API command format is not used; instead, a continuous stream of bytes is sent.

At the end of the firmware download, the Host must send a Sync command to check the status of the IA61x. Only if the download was successful does the IA61x acknowledge the Sync. If the IA61x does not acknowledge a Sync command, the Host must decide how many times to re-try before it power-cycles the IA61x.

Note: The IA61x does not support I²C rates of less than 30 kbits data rate.

Code download is performed after power up (POR) or a system reset. In this case, the Host must start downloading the firmware image by sending a Boot command (0x01) to the IA61x, followed by an ACK read from the IA61x. A Boot ACK indicates that the IA61x is ready to receive the firmware image; otherwise, the IA61x must be power cycle and the Boot command should be sent again.

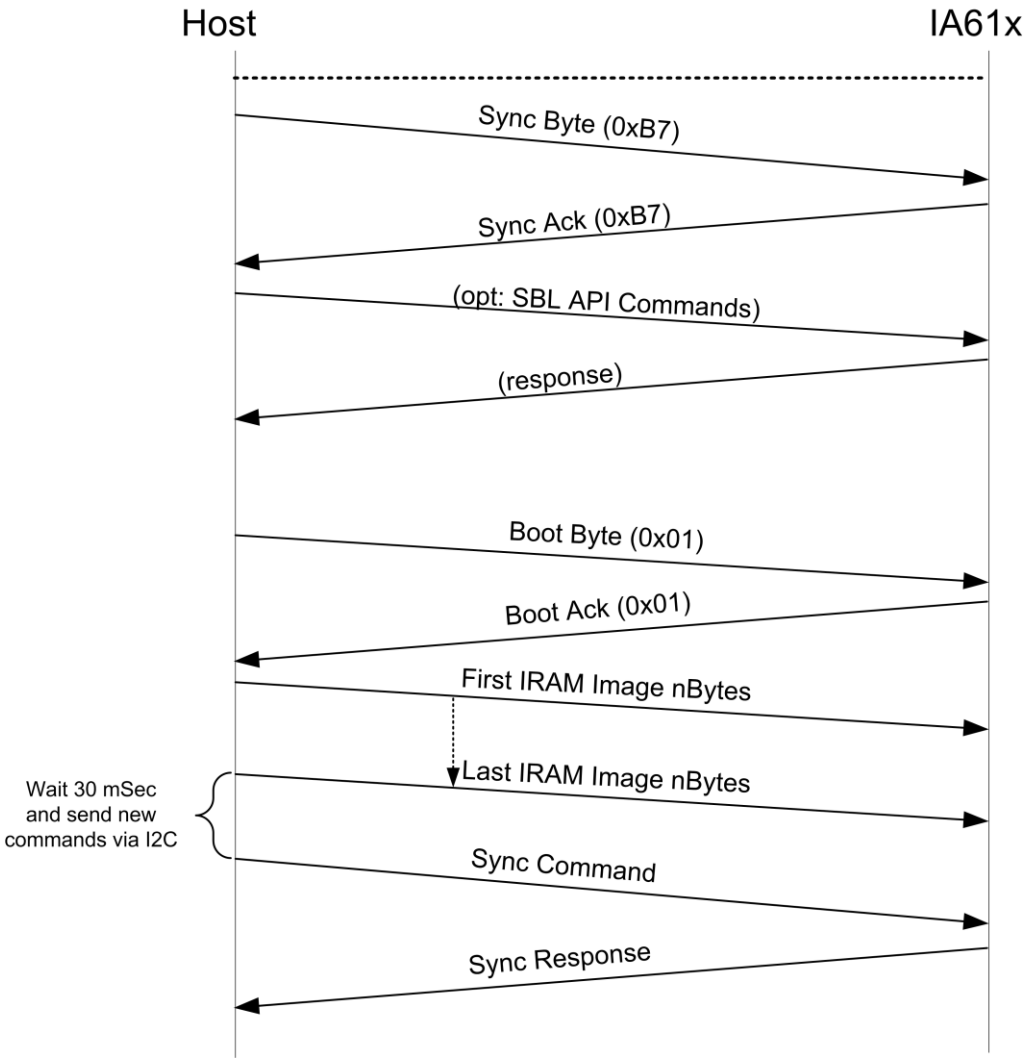

Figure 10 *Code Download Using I²C*

5.4 Code Download over UART

The IA61x host interface includes a two-wire Universal Asynchronous Receiver/Transmitter (UART). The UART interface enables high-speed downloads of the firmware from the Host Controller to the IA61x. This section provides the command sequence and supporting details required for downloading firmware to the IA61x via the UART.

5.4.1 UART Boot Sequence

The UART in the IA61x supports the auto baud rate feature; the Host needs to send Auto baud bytes (0x0000) to lock down baud rate in the SBL after a power cycle with at least 20 ms of delay. **Error! Reference source not found.** lists the supported auto baud rates in the SBL. The Baud rates available in SBL are slow, however the host can send an API command (SetRateRequest, 0x8019) to change the baud rate to a higher speed (between 115.2 kbps and 3.00 Mbps). Subsequent API commands or Image downloads occur at the higher specified baud rate. The Baud rate set in the SBL will remain same after firmware download. The Host cannot change the baud rate after a firmware download.

Table 9 *Baud Rates supported by SBL after POR*

	43.08 MHz	FPGA
19.2 k	√	√
28.8 k	√	√
38.4 k	√	√
57.6 k	√	√
115.2 k	√	√

In **Error! Reference source not found.**, any baud rate that cannot be guaranteed to work at all, or not work reliably, is marked with a dash (–). In some cases, it may work intermittently but it is not to be relied upon.

5.4.2 Firmware Download Procedure

The command/response sequence in **Error! Reference source not found.** shows the complete UART bootload procedure. The sequence is summarized as follows:

1. The Host Controller UART must be configured to operate at a supported baud rate. Refer to **Error! Reference source not found.**.
2. The Host Controller sends the Auto baud bytes (0x00 0x00) to the UART with 20 ms of delay from POR. The Auto baud byte is used primarily for auto baud rate detection.
3. After receiving Auto baud byte acknowledge from the IA61x, the host sends a Sync Byte (0xB7).
4. After receiving the Sync Acknowledge from the IA61x, the Host sends either an SBL API command or a Boot Byte.
5. The Host can send any API command supported by the SBL, as specified in **Error! Reference source not found. Error! Reference source not found.**.
6. If Host sends a RateRequest API command, the SBL responds as shown in **Error! Reference source not found.**, **Error! Reference source not found.**.
7. When the Host is ready to initiate an Image transfer, the Host sends a Boot byte to the IA61x.
8. When the Host receives the Boot ACK response from the IA61x SBL, the Host initiates Image download.

   Note: If the Host terminates image transfer prior to completion, the IA61x will interpret byte transfers (e.g., API commands, boot bytes, sync bytes) as image data, not as the function that Host may be expecting.
9. The Host must wait 30 mSec for the IA61x firmware to initialize. During the UART download, the SBL can return to the Auto-detect state when the Host power cycles the IA61x.

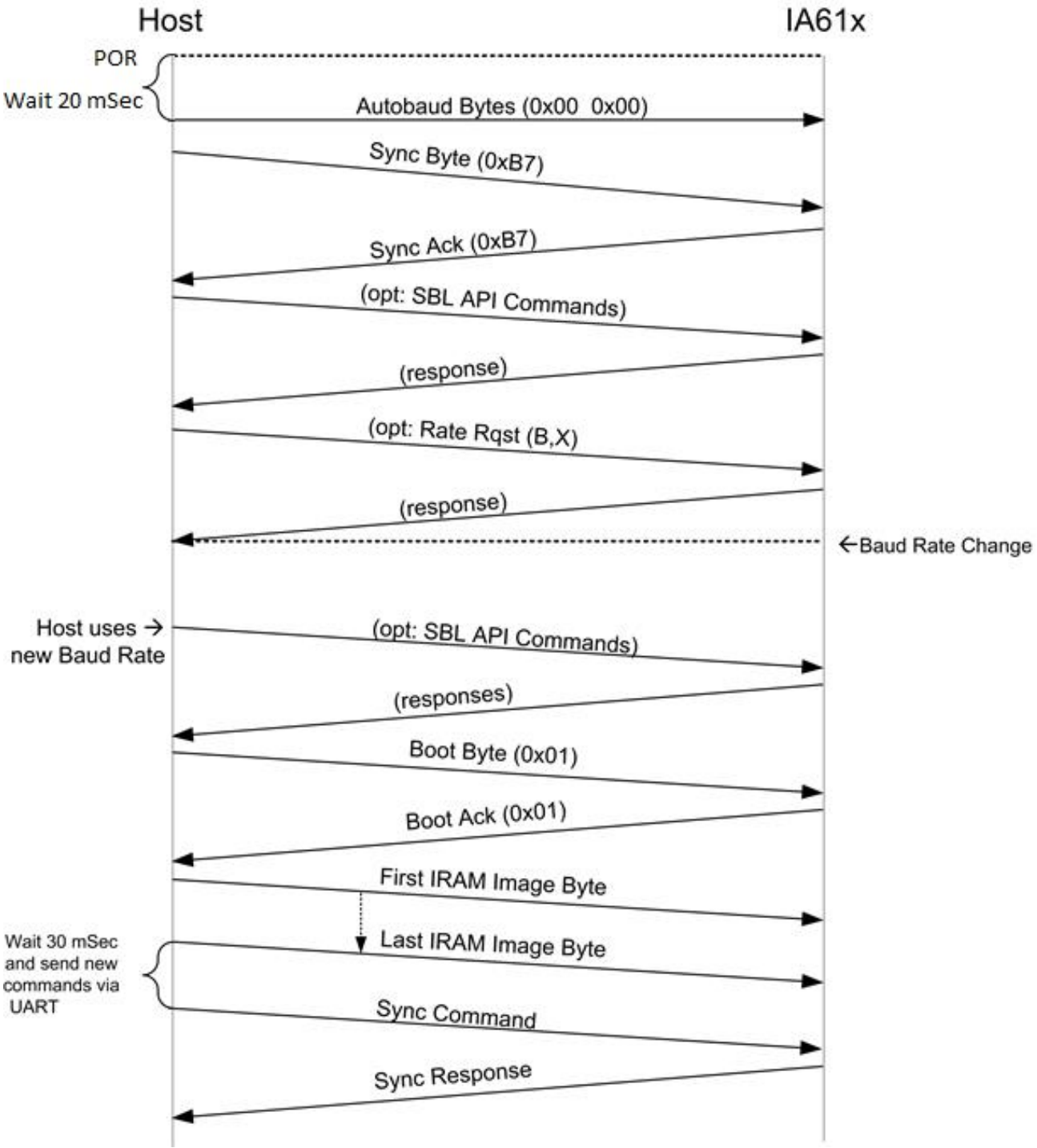

Figure 11 *UART Boot Sequence*

5.4.3 Firmware Download Verification

After the firmware has been downloaded, the UART interface is available for receiving APIs. An API Sync command (0x8000 0x0000) should be used to confirm that the code is running.

If the IA61x responds with a Sync ACK (0x8000 0x0000), the download was successful and the IA61x is active.

The Host should power cycle and retry the bootload procedure for the following cases:

10. If the response is an error (0x8000 0xFFFF), the download failed.
11. If the IA61x sends any other response or no response.

Note: The Host needs to send system configuration image before a firmware download.

5.4.4 UART Status Codes

The status codes returned by the IA61x chip can have the values shown in **Error! Reference source not found.**. These status codes are sent in response to any input, except API commands.

Table 10 *UART Status Codes*

Status Code Value	Code	Description
0xFF	Error	The booting process has stopped or an invalid byte has been written to the bootloader. As a result of this error condition, the bootloader moves back to the Auto-detect state. A re-synchronization of the UART is required to continue the firmware download.
0xB7	Sync	A sequence of sync (0xB7) byte was received. The bootloader locked the control interface as UART. The bootloader is ready to receive API command or Boot command.
0x01	Boot	The booting process has been successfully initiated. The UART interface returns 0x01 once after receiving the Boot Command.
0x02	FW Download Success	This code indicates that the firmware download was successful

5.5 Code Download over SPI

The Host controller can download the IA61x firmware via SPI. The firmware image stored in the Host memory is already pre-formatted with the necessary flow control, and can be downloaded byte by byte by the Host starting with the first memory location and ending with the last. During code download, the 32-bit API command format is not used; instead, a continuous stream of bytes is used.

At the end of the firmware download, the Host must send a Sync command to verify the IA61x's status. Only if the download was successful does the IA61x acknowledge the Sync. If the IA61x does not acknowledge a Sync command, the Host must decide how many times to re-try before it power cycle the IA61x.

Code download is performed after a power up (POR) or system reset. The Host must start downloading the firmware image by sending a Boot command (0x00000001) to the IA61x, followed by an ACK read from the IA61x. A Boot ACK indicates that the IA61x is ready to receive the firmware image; otherwise, the IA61x must be power cycled and the Boot command should be sent again.

After code image download is complete, the max SPI transfer rate depends on the IA61x internal PLL frequency. Check with your Knowles Applications Engineer for your more information.

Note: The Host needs to send a System Configuration image before firmware download.

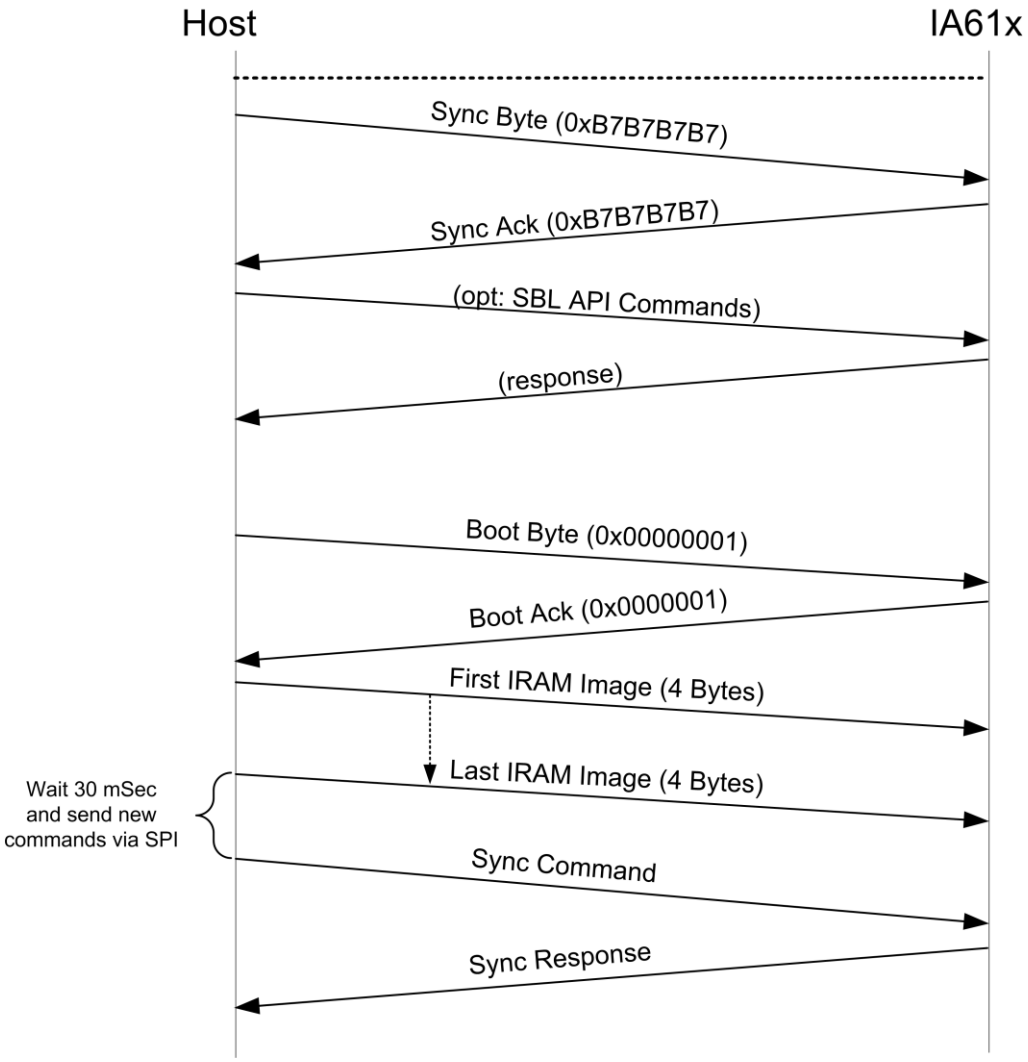

Figure 12 *Code Download Using SPI*

5.6 Code Download over I²S

The Host controller can download the IA61x firmware via I²S. The firmware image stored in the Host memory is already pre-formatted with the necessary flow control, and can be downloaded word-by-word (4 byte) by the Host starting with the first memory location and ending with the last. During code download, the 32-bit API command format is not used; instead, a continuous stream of bytes is used.

The firmware can be sent either on a single channel or on both channels. If firmware is sent on single channel, then the SBL expects data on the left channel only. The SBL expects data in little endian format. The IA61x identifies the mode automatically by looking at the firmware signature words arriving on the left channel only, or both the left and right channels. Downloads over the left or left/right channel have to be selected before the download starts and cannot be switched in between the transfers. I²S firmware download does not work when code download is in big endian format.

At the end of the firmware download, the Host must send a Sync command to verify the IA61x's status. If the firmware download was successful, the firmware acknowledges the Sync command. If the IA61x does not acknowledge a Sync command, the Host must decide how many times to re-try before it power cycle the IA61x.

For downloading a binary over I²S, the boot load initiate sequence has changed. You need a command interface to initiate the Binary download. The I²C/UART control interface is used for boot load to initiate the binary download. This is shown in **Error! Reference source not found.**.

Code download is performed after power up (POR) or system reset. The Host must start downloading the firmware image by sending a Boot command (0x03) either over I²C or UART to the IA61x, followed by an ACK read from the IA61x using the same interface. A Boot ACK indicates that the IA61x is ready to receive the firmware image; otherwise, the IA61x must be power cycled and the Boot command should be sent again.

Note: The Host needs to send a system configuration image before firmware download.

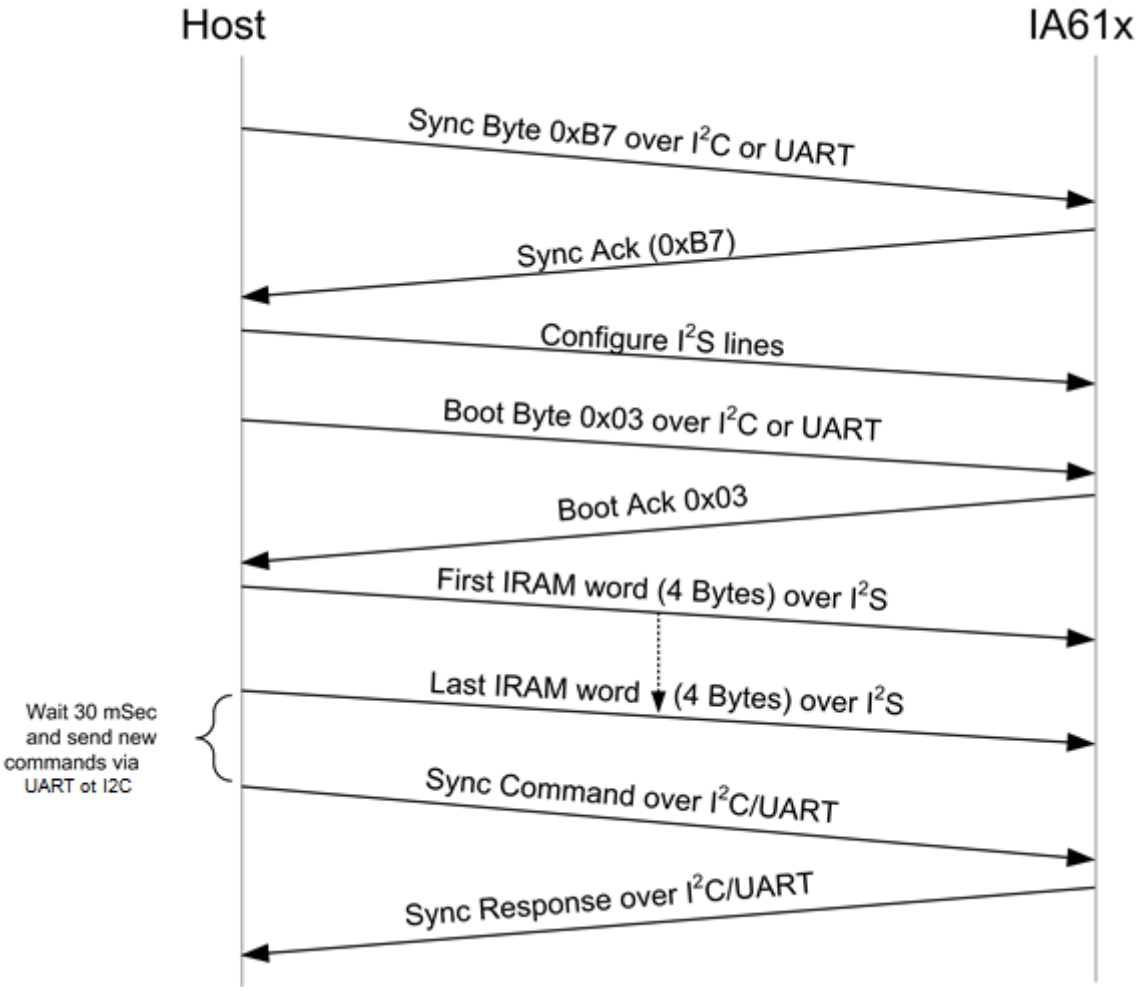

Figure 13 *Code Download Using I²S*

5.6.1 Firmware Download Procedure

The command/response sequence in **Error! Reference source not found.** shows the complete I^2S bootload procedure. The sequence is summarized as follows:

1. The Host I^2S Controller must be configured as:
   - Word length: 16 bits or 32 bit
   - Frame rate: 8 kHz to 192 kHz
   - Clock Sampling: Falling edge
   - Bit Order: MSB (Most Significant Bit)
   - Single Channel (left channel active) or Two channel
2. The default IA61x Port A (I^2S controller) word length is set to 16-bits in the SBL. To change the word length, use the Set Device Parameter APIs. For example:
   - 16 bits: 0x800C 0A00 0x800D 000F (default)
   - 32 bit: 0x800C 0A00 0x800D 001F
3. The Host sends a Sync Byte (0xB7) over the control interface (I^2C or UART) and receives a Sync ACK as (0xB7) on the same control interface.
4. When the Host is ready to initiate an Image transfer, it first configures the I^2S as described in Step **Error! Reference source not found.** and starts the I^2S clock.
5. The Host sends a Boot byte (0x03) over the I^2C or UART interface to download the binary over I^2S. Boot byte 0x03 enables download over I^2S inside the SBL.
6. The Host receives the Boot ACK response (0x03) from the IA61x SBL over the I^2C or UART interface.
7. The Host has to send the system configuration image Binary file over the I^2S interface to the IA61x.
8. The process is the same for firmware image repeat from Step **Error! Reference source not found.** to Step **Error! Reference source not found.**.
9. The Host must wait 5 mSec for the IA61x firmware to initialize.
10. Once the image is sent, the Host can send a sync command over the supported control interface (I^2C or UART) and check for a proper sync command response. If the sync command is not accepted, the Host can power cycle the IA61x and download the firmware again.

5.6.2 Firmware Download over I²S with the IA61x in Master Mode

When performing code download over I²S with the IA61x as the I²S master, the IA61x generates the I²S clock (which will otherwise be supplied by the host). I²S clock generation in SBL can be achieved by performing following steps.

1. Before sending the boot byte(0x03), set Port A into master mode using the Set Device Param API:
   - Master Mode: 0x800C 0A09 0x800D 0001
2. Select the sampling rate for the I2S0 device:
   - Bit Clock 3.072 MHz, 48K, Stereo, 32-bit: 0x800C 1600 0x800D 0000
   - Bit Clock 6.144 MHz, 96K, Stereo, 32-bit: 0x800C 1600 0x800D 0001
   - Bit Clock 1.536 MHz, 48K, Stereo, 16-bit: 0x800C 1600 0x800D 0002
   - Bit Clock 3.072 MHz, 96K, Stereo, 16-bit: 0x800C 1600 0x800D 0003

   In master mode, the IA61x only supports 48K and 96K rates over two channels. The Host I²S controller must be configured to match the I²S clock mode that is selected.
3. The Host sends a Boot Byte (0x03) command over the Control Interface (I²C or UART) and receives a Boot ACK (0x03) back on the same control interface. Once the IA61x receives the boot byte it generates the clock.
4. The Host has to send the system configuration image Binary file over the I²S interface to the IA61x. The Host can send the firmware image over I²S as soon as it receives the clock.
5. Use the same procedure for the firmware image (repeat from Step **Error! Reference source not found.** to Step **Error! Reference source not found.**).
6. The Host must wait 5 mSec for the IA61x firmware to initialize.

   Once the image is sent, the host can send a sync command over the supported control interface (I²C or UART) and check for proper sync command response. If the sync command is not accepted, the host can power cycle the IA61x and download the firmware again.

Note: The I²S interface is used for downloading the firmware only. The command communication should use I²C or UART depending on the IA61x pin configuration.

Chapter 6: Host Interface Bus Protocol

The IA61x is a slave device, and it only responds to commands from the master controller. Most commands have an associated callback message used by the slave to acknowledge receipt of the command and to send the requested data back to the master. The callback command code is the same as that sent by the master in the request message.

6.1 Communication Protocol

In order to handle the varying nature of the different API commands over the I^2C, UART, SPI, or SoundWire, specific communication protocols have to be used by the Host. All communication between Host and the IA61x is done in 4-byte messages. Note that the Host can send commands to the IA61x via the I^2C, UART, SPI, or SoundWire interface. The command response is provided to the Host on the same interface that command was received.

Two general scenarios need to be handled by these protocols:

- Commands from Host to the IA61x that request no data back
- Commands from Host to the IA61x that request data to be transmitted back to the Host from the IA61x.

Most commands are executed by the IA61x before an acknowledgement is sent back to the Host. Commands that request data back from the IA61x send the requested data back with the acknowledgement. Commands that change the chip state, such as Reset, are acknowledged before they are executed.

Each command sent to the IA61x is acknowledged with one of the following three response code types:

Legal command: The same command code is used for acknowledgement if the command is legal.

Example: Host sends 0x801E0005; the IA61x responds with 0x801E0005

Illegal command: The Error response code (0xFFFF) is used if the API command is not supported.

Example: Host sends 0x80590001; the IA61x responds with 0xFFFF8059.

Response not ready: A "zero" response code (0x0000 0000) is sent as long as a reply (either acknowledgment or requested data) is not ready. This response code should be treated by the Host as a Response-Not-Ready indicator.

Example: Host sends 0x801E0005; the IA61x responds with 0x00000000.

In order to ensure a fast command-reply cycle, the Host can periodically poll the IA61x after sending a command to check if a response is ready (a typical recommended polling period is 10 mSec). Polling is done by simply activating the port so the clock runs, and monitoring the data sent back to the Host.

6.1.1 SoundWire Protocol Specifics

The SoundWire Control Port is command based. Each SoundWire command can be used to read or write 1 byte. Since the IA61x API commands are four bytes, the host needs to read out or write in four Control words. Each frame can contain only one control word; therefore, a command write/read takes a minimum of four frames to complete. The host should wait for 20 mSec before trying to read the response using a SoundWire read command.

Commands are sent on the Control Port as a Register Write. For reading response back, a Register Read is used. The register addresses to read/write are shown in **Error! Reference source not found.**:

Table 11 *SoundWire Address Register*

Byte	Read/Write	Address
Byte 1	Read	0x2004
Byte 2	Read	0x2005
Byte 3	Read	0x2006
Byte 4	Read	0x2007
Byte 1	Write	0x2000
Byte 2	Write	0x2001
Byte 3	Write	0x2002
Byte 4	Write	0x2003

The control word specifics for read and write are shown in **Error! Reference source not found.**:

Table 12 *Control Word Format*

Bits/Command	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
Read	X	0	1	0	IA61x Device Address				Register Address [15:8] (0x20)							
Write	X	0	1	1												
Bits/Command	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
Read	Register Address [7:0] (0x04 – 0x07)								1	0	1	1	0	0	0	1
Write	Register Address [7:0] (0x00 – 0x03)															
Bits/Command	**32**	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**	**41**	**42**	**43**	**44**	**45**	**46**	**47**
Read	X	Read data [7:0] (from IA61x)								Dynamic Sync				PARITY	NAK	ACK
Write		Write data [7:0]								Dynamic Sync						

Parity, NAK and ACK are written by each slave in the SoundWire system even if the device addressed was the IA61x.

6.1.2 UART Protocol Specifics

UART communication is very simple. The Host transmits all command bytes to the IA61x chip via the UART pin. When command responses are available, the IA61x immediately transmits all bytes to the Host. No host polling or host interrupt processing is required.

6.1.3 I²C Protocol Specifics

Error! Reference source not found. show the scenario where the Host is sending a command to the IA61x (Master to Slave), and the scenario where the IA61x is sending a reply to the Host in response to a query from the Host. In this example, the IA61x has been configured with a 7-bit Slave Address 0x3E (**Error! Reference source not found.**). Other I²C device addresses are supported; refer to Section **Error! Reference source not found.**, **Error! Reference source not found.**.

Note: When Master Receiving data from IA61x slave, it should not write sub address before reading data. IA61x does not support this type of mechanism.

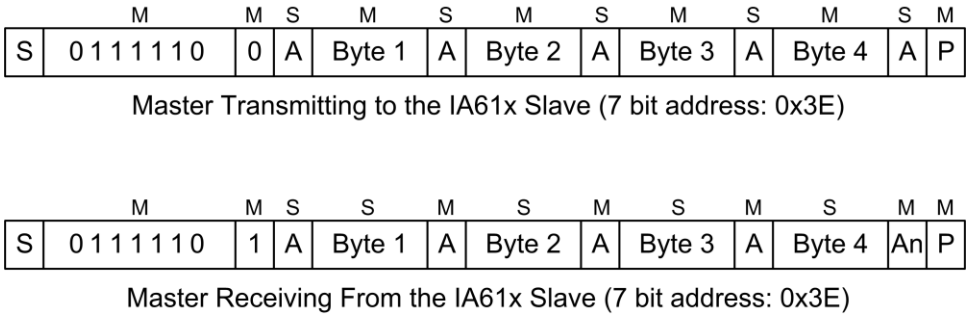

Figure 14 *I²C Protocol Data Flow - 7-bit Address*

Legend:

M	:	bus Master device
S	:	(a) Start condition, or (b) Bus Slave device
P	:	Stop condition
A	:	Acknowledge
An	:	Not Acknowledge (“NAK”)

6.1.4 SPI Protocol Specifics

When the Host asserts the SPI chip select signal, all command bytes **to** the IA61x chip are transmitted on the SPI MOSI (Master Out Slave In) pin. When the host is transmitting data to the IA61x (via the MOSI pin), the host is also receiving data being clocked on the MISO (Master In Slave Out) pin. The return data may contain command responses, if queued, or all zeros if no command responses are queued.

If the host wants to check whether the IA61x Smart Microphone has command response data queued, the host can assert the SPI chip select signal, and read the data from the MISO pin. This will either be command response data, or all zero's. When doing so, the host should send zero bits on the MOSI pin.

The SPI clock polarity is set (SCPOL=0) such that the clock signal idle (base) value is low (0).

The SPI clock phase is set (SCPH=1) such that data is sampled (captured) on clock falling edge and data is propagated on clock rising edge.

Data sent through the MOSI and MISO pins is in 32-bit words.

Chapter 7: SoundWire

7.1 Introduction

The IA61x – Smart Microphone solutions support SoundWire and is compatible with the MIPI SoundWire specification Rev 1.0. SoundWire allows audio data and control information to be transmitted on the same bus. This section has details of IA61x functions and features supported through the SoundWire interface.

7.2 Overview

- The IA61x SoundWire interface provides the following SoundWire capabilities:
  - 32-bit command and response over SoundWire control port.
  - Data Port 4 for SPDM Rx with up to 4 Channels.
  - Data Port 2 for SPDM Tx with up to 4 Channels.
  - Data Port 3 for SPCM Rx with up to 4 Channels and up to 32-bit word length.
  - Data Port 1 and 5 for SPCM Tx with up to 4 Channels each and up to 32-bit word length.
  - Data Port 0 for PCM Port with only 1 Data Channel, up to 32-bit word length used only for BRA.
  - Isochronous transport of data.
  - ClockStopMode 0 and ClockStopMode 1 with WakeUp support.
  - In-band Implementation defined interrupt.
- Following features are supported over SoundWire similar to other interfaces.
  - API commands.
  - Firmware Download. (Control port, BRA, SPCM port).
  - WDB and RDB (Control port and BRA).
  - Streaming and Bursting (Control port and BRA).
  - In-Bound SoundWire Interrupt to Host and WakeUp from Host.
  - Optional GPIO IRQ and GPIO WakeUp.

7.3 Command Protocol

SoundWire control port is the only port supported for sending 32 bit API command and receiving 32-bit API response. Each frame of SoundWire contains 8 bit of control data so it takes four frames to send or receive 32 bit of data. The IA61x provides 4 addresses for writing 32-bit data and another 4 addresses for reading 32-bit data where order must be followed as explained in **Error! Reference source not found.**.

7.4 Audio Route Setup

Steps for configuring routes in case of SoundWire interface are same as other interfaces. Please refer to respective sections for more details. Host must setup the route parameters using API commands for respective routes before configuring data ports and enabling channels.

7.5 SoundWire - GPIO IRQ and Wakeup

SoundWire specification provides a way for slave to generate interrupt to Host at any time. In case of ClockStopModes, the IA61x can interrupt Host by raising data line high. In case when Clock is active, the IA61x can generate alert, which is interrupt to the Host.

SoundWire specification also provides a way to interrupt the IA61x during ClockStopModes by resuming clock.

Apart from in-bound interrupt and wakeup, the IA61x also supports legacy method of sending an interrupt over GPIO and waking up from GPIO signal. For this method, the chip has to be put to sleep using API command and not ClockStopMode1. Here are the details of IRQ and Wakeup behavior specific to SWIRE interface.

As shown in the below figure there are mainly four states (**A**) of device and four possible events which can occur as mentioned (**D - I)**.

GPIO availability (**C**) is based on HW_Config parameter (*SwireGpioConnected)* and should be fixed for a project.

With 'Sleep' command 0x8010, device can wake up by reading control port data (**B14- B17**) if it is enabled in HW_Config (*WakeUpOnSwireRead).* This parameter should be fixed for a project.

WakeUpEnable (**B6-B13**) is a SWIRE feature where Host enables slave to wakeup Host asynchronously during Clock Stop Modes. Host should not change '*WakeUpEnable*' bit after sending '*ClockStopPrepare'* for the very session.

SetEvent API is extended and parameter '5' will refer to SWIRE Implementation Defined Interrupt.

When SetEvent configuration does not match the expected configuration, it will be an **ERROR** case (column **F**)**.** Note that when GPIO is not available but SetEvent API is called to configure GPIO (param 0 to 4); response of API will be an error.

- In these cases 'Override set event' means that FW will handle the case as per correct SetEvent.
- In case of ClockStop cases, '*ClockStop_NotFinished*' bit in '*SCP_Stat*' will be set until clock is resumed so that Host will get to know something is wrong.

Over GPIO. This is the legacy method of sending an interrupt over GPIO. For this method, the chip has to be put to sleep using API command and not ClockStopMode1.

	A	B	C	D	E	F	G	H	I
1	Device current state and capabilities			Key Word detection			Wake-Up GPIO	Clock Resuming	SWIRE Read
2				Correct Set Event	Operation	Set Event Mismatch Handling (ERROR)			
3	Normal State (Already Awake)		GPIO Available	GPIO	Set GPIO_IRQ	Mismatch not possible.	GPIO wakeup interrupt is disabled.	N.A. (Clock is already active)	Normal operation.
4				SWIRE	Set SWIRE Imp. Def. Interrupt				
5			GPIO Not Available	SWIRE	Set SWIRE Imp. Def. Interrupt	Do Not change Set Event. Send API Error.			
6	ClockStopMode_1 (Sleeping)	Wake Up Enabled	GPIO Available	SWIRE	Wake up, set SWIRE Imp. Def. Interrupt and SWIRE Wakeup.	Override set event.NotFinished set to 1.	GPIO wakeup interrupt is disabled.	Normal wake up.	N.A. (Needs Clk Resume)
7			GPIO Not Available			Do Not change Set Event.Send API Error.			
8		Wake Up Disabled	GPIO Available	GPIO	Wake up and set GPIO IRQ.	Override set event. NotFinished set to 1.	Device wakes up. *3		
9			GPIO Not Available	SWIRE	ERROR : Wake up and set SWIRE Imp Def Interrupt only. *1	Do Not change Set Event. Send API Error.	GPIO wakeup interrupt is disabled.		
10	ClockStopMode_0 (Already Awake)	Wake Up Enabled	GPIO Available	SWIRE	Set SWIRE Imp Def Interrupt and WakeUp.	Override set event. NotFinished set to 1.	GPIO wakeup interrupt is disabled.	Normal operation.	N.A. (Needs Clk Resume)
11			GPIO Not Available			Do Not change Set Event. Send API Error.			
12		Wake Up Disabled	GPIO Available	GPIO	Set GPIO IRQ.	Override set event.NotFinished set to 1.			
13			GPIO Not Available	SWIRE	ERROR : Set SWIRE Imp Def Interrupt only. *1	Do Not change Set Event. Send API Error.			
14	Sleep Command Mode (Sleeping)	Wakeup On SWIRE Read Enabled (PD0 ON) *5	GPIO Available	GPIO	Wake up and set GPIO IRQ.	Wakeup only.	Device wakes up.	No wake up.	Device wakes up.
15			GPIO Not Available	SWIRE	ERROR : Wakeup only. *2	Do Not change Set Event. Send API Error.	GPIO wakeup interrupt is disabled.	Clock Stop Handing is in HW Mode.	
16		Wakeup On SWIRE Read Disabled (PD0 OFF)	GPIO Available (PD0 Off) *4	GPIO	Wake up and set GPIO IRQ.	Wakeup only.	Device wakes up.	No wake up (SWIRE Off)	No wake up (SWIRE Off)
17			GPIO Not Available.	ERROR : Bad HW-Configuration. Not Supported.					

*1 There is no way device can intimate Host about keyword detection immediately. Host will see implementation defined interrupt on SWIRE when clock is resumed. *ClockStop_NotFinished*' bit in '*SCP_Stat*' will be set until clock is resumed so that Host will get to know something is wrong.

[*2] There could be a valid case when VQ route is not applicable, GPIO is not available and Host wants to keep the IA61x in sleep. This configuration will be useful in this case. In case of VQ route, device will just wakeup upon keyword but cannot set interrupt, as *IntMaskImpDef1* will get reset during sleep.

[*3] Device simply wakes up but SWIRE clock is not yet there. As device is already awake, there will not be any further delay while resuming the clock later.

[*4] PD0 will be turned off and so will be SWIRE. Host cannot see DSP core on SWIRE bus.

[*5] Device will get detached after Sleep Command is processed and Host needs to re-enumerate to issue Read for waking it up. Device will be detached again after waking up. This is because FW needs to switch between SW and HW mode of Clock Stop Handling, which needs SWIRE to go through reset.

Chapter 8: Use Cases

8.1 Voice Wake & Continuous Voice Wake

Access of the phone (or services) in existing mobile systems is obtained by touching the device to wake it up from a low power state ("sleep mode") where the power consumption is minimized to maximize the battery charge life span. With the introduction of the low power Voice Wake solutions, it is possible to access the devices services by simply saying a predefined command to "wake up" the system. For this to be a viable solution, the detection of the voice command needs to be achieved when keeping the power consumption below TBD mW. This technique is referred to as Voice Wake Up.

8.1.1 Voice Wake

Voice Wake (previously called Voice Sense) is a low-power state in which the IA61x is running a key word detection algorithm. The Host Processor is powered-off, and is depending upon the IA61x to detect a user keyword to cause a wake-up event.

The IA61x is operating in a low-power state, with only enough signal processing activity to monitor in-coming audio signal from a local microphone. The incoming audio is monitored to ascertain whether specific voice utterances ("keywords") have been detected.

8.1.1.1 Voice Wake Sequence

The host must place the IA61x into Keyword Detection state. The procedure is:

1. Download firmware binaries (SysConfig and Firmware)
2. The Host transfers keyword file (OEM, Host, Authentication) using the API command WriteDataBlock (0x802F 0xXXXX), with the command parameter specifying the keyword model length. This needs to be done once after every reset.
3. Set frame size to 16ms: 0x8035 0x0010
4. Set Sample rate to 16 KHz: 0x8030 0x0001
5. Disabled buffering: 0x802A 0x0000.
6. Set up VoiceSense 16k PDM route: 0x8032 0x0008 (Route 8) or 0x8032 0x0006 (Route 6)
7. Enable VS Keyword Detection mode: 0x8017 0x5003 0x8018 0x0000
8. Set power state so that the IA61x enters a low power mode: 0x9010 0x0001 (Stage 0 for Route 8) or 0x9010 0x0002 (Stage 1 for Route 6).
9. Utter keyword at PDM mic. e.g. "Hello VoiceQ".

10. On successful AAD IA61x enters into Stage 1 for Keyword Detection.
11. If Voice Sense Algo detects keyword, the IA61x signals the host by generating an interrupt on HOST_IRQ pin, based on the settings from Get Event Response (rising edge default).
12. The chip at this point switches its power mode back to Stage 2 (Command Mode).
13. The Host wakes up.
14. The Host sends API command GetEvent (0x806D 0x0000) to the IA61x. Response is sent from the IA61x indicating Voice Sense event type:
    - Voice Sense: no keyword detected: 0x0000
    - Voice Sense: keyword detected: Keyword ID
15. This command resets the HOST_IRQ line.

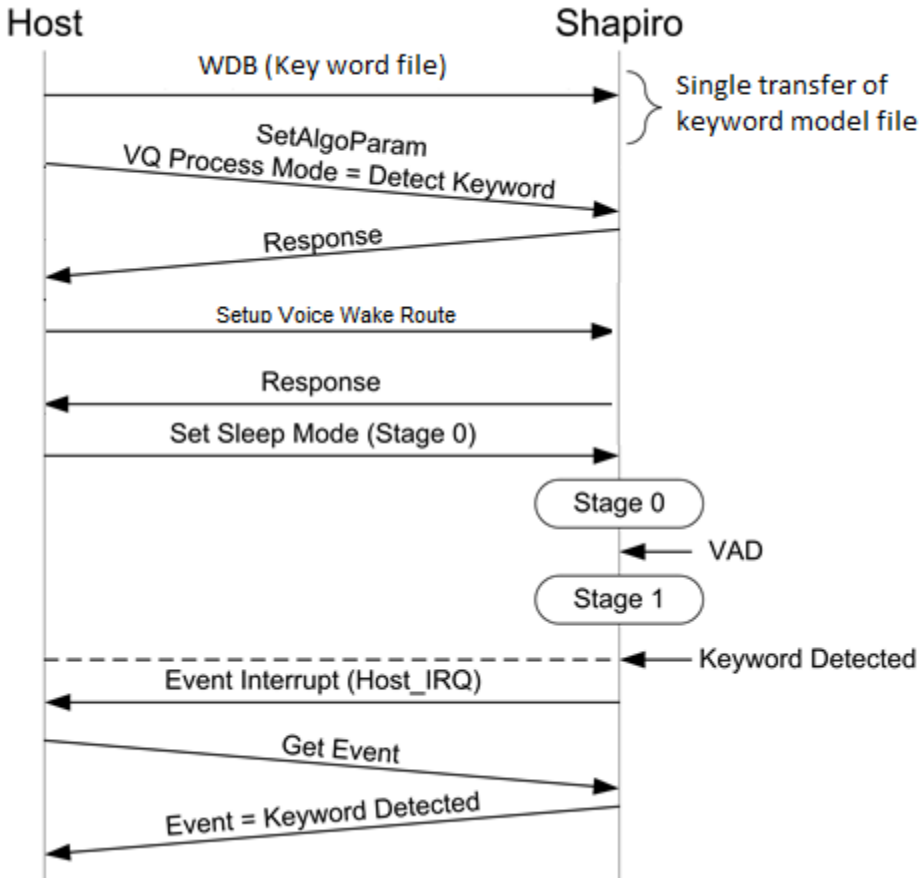

Figure 15 *Voice Wake Sequence*

8.1.1.2 Host Cancellation of Keyword Detection

Host may request cancellation of Voice Wake mode at any time, for any reason. Generally, it is anticipated that an incoming phone call would create this condition, but it could be as simple as the user pushing the phone's "On" button to restore the phone to power-on status. The keyword detection state is only while in low power mode.

The corresponding method is described below:

Voice Wake in Low-Power Mode

1. The Host takes Voice Wake out of low-power mode by causing a falling edge on the Wake Up Input pin, or when using SPI or UART, by sending 4 bytes of 0s (0x00000000).

2. Host can send optional Sync command to verify.

This brings the firmware to idle state. Any use-case can be started at this point.

8.1.2 Continuous Voice Wake

Using voice to access the device services can be extended to any of the active power states of the phone by simply storing the voice command until the phone is able to access the device services. This mode of operation is called Continuous Voice Wake.

The first and most important service of the Continuous Voice Wake mode is the seamless transition from the low power mode to the active power mode. This means that the user should be able to address the device by enunciating the wake up command and without any unnatural pause continue speaking any other possible command. For example "Hello VoiceQ, call Paul's cell phone."

There are two main differences between standard Voice Wake mode and Continuous Voice Wake mode:

1. In Continuous Voice Wake, the firmware is started into active power mode without host intervention

2. Up to three seconds of mic input is stored on the IA61x until it is requested by the host. Once requested it is sent to the host over a high-speed control port (e.g. SPI). This stored data can be keyword + phrase depending on how the firmware was configured prior to detection state.

8.1.2.1 Continuous Voice Wake Sequence

The process of setting up the phone for Continuous Voice Wake is similar to the process of setting up the phone for Voice Wake, but requires some more steps. Host can select to preserve keyword + phrase or only phrase using an API command. The procedure is described as below:

1. Download firmware binaries (SysConfig and Firmware)

2. The Host transfers keyword file (OEM, User Trained or Voice ID) using the API command WriteDataBlock (0x802F 0xXXXX), with the command parameter specifying the keyword model length. This needs to be done once after every reset.
3. Set frame size to 16ms: 0x8035 0x0010
4. Set Sample rate to 16 KHz: 0x8030 0x0001
5. Set Buffer Data Format to Q15 or Q11: 0x8034 0x0002 or 0x8034 0x0001
6. If keyword preservation not required then send: 0x802A 0x0001.
7. Set up VoiceSense 16k PDM route: 0x8032 0x0008 (Route 8) or 0x8032 0x0006 (Route 6)
8. Enable VS Keyword Detection mode: 0x8017 0x5003 0x8018 0x0000
9. Set power state so that the IA61x enters a low power mode: 0x9010 0x0001 (Stage 0 for Route 8) or 0x9010 0x0002 (Stage 1 for Route 6).
10. Utter keyword at PDM mic. e.g. "Hello VoiceQ, call Paul's cell phone".
11. On successful AAD, IA61x enters Stage 1 for Keyword Detection.
12. If Voice Sense Algo detects keyword, the IA61x signals the host by generating an interrupt on HOST_IRQ pin, based on the settings from Get Event Response (rising edge default).
13. The chip at this point switches its power mode back to Stage 2 (Command Mode) and continues to buffer mic input into internal memory.
14. The Host wakes up.
15. The Host sends API command GetEvent (0x806D 0x0000) to the IA61x. Response is sent from the IA61x indicating Voice Sense event type:
    - Voice Wake: no keyword detected: 0x0000
    - Voice Wake: keyword detected: Keyword ID
16. This command resets the HOST_IRQ line.
17. At this stage, the host can choose to capture the buffered audio from the IA61x through bursting, or put the IA61x back to low power, or restart the Voice Wake or run any other use-case like pass-through.

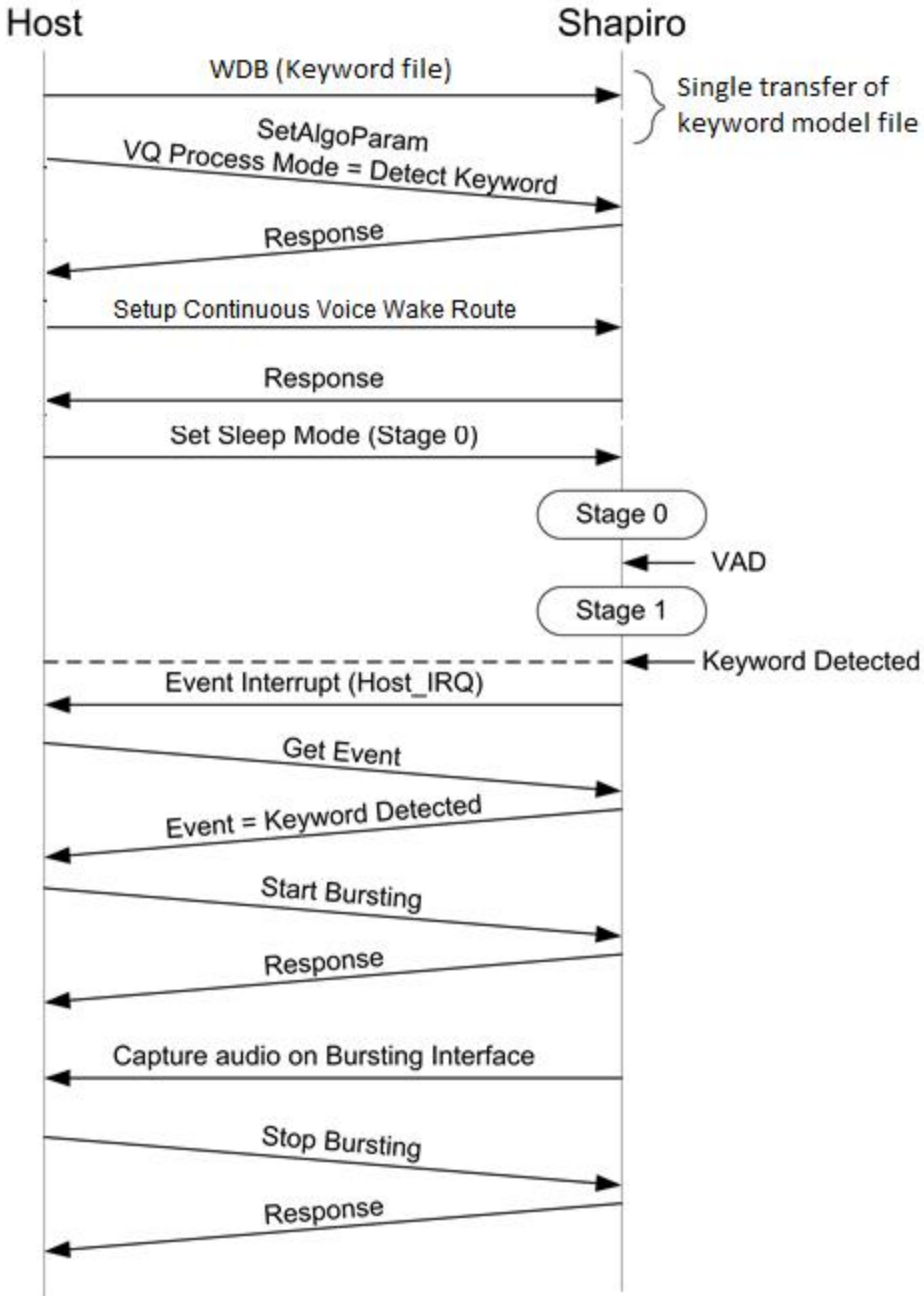

Figure 16 *Continuous Voice Wake Sequence*

8.1.2.2 Bursting buffered audio data

The IA61x will stream the three seconds of stored data over any of the UART, SoundWire, I2C or SPI busses. The host can store the stream and send it to the ASR engine for processing.

The request to get the three seconds worth of data is through the bursting API command. The command to be used is 'Start bursting – 0x8026'.

Once sufficient data is streamed out, the host should stop the streaming using 'Stop bursting – 0x8026' command.

Streaming header is defined as below:

Table 13 *Streaming Header Format*

0x1234	Endpoint&0xFF	Counter&0xFF	Length
Byte 0,1	Byte 2	Byte 3	Bits[31-0]

The length field is in Big Endian format, and has the two MSBs used for different purpose:

Bit [31-30] is used for encoding. Currently following formats are supported:

- 01 – 16-bit PCM

An example header as seen in the stream is:

12 34 14 10 00 02 01 00

Here the length is 0x200 and it is 16-bit PCM, meaning sampling rate of 16 kHz.

8.1.2.3 Host Cancellation of Continuous Voice Wake

Continuous Voice Wake has three states. The procedures to cancel the use case in each of the states are as follows:

1. Low Power keyword detection state: This state is similar to Voice Wake keyword detection state and thus a wake-up will abort the Continuous Voice Wake.
2. High Power data buffering state: In this state, the host has been interrupted but the bursting of data has not yet started. To abort this state and start another use-case, simply starting a new use-case will clean-up the Continuous Voice Wake.
3. High Power data bursting state: In this state, the host has initiated the bursting of data. To abort in this state is to terminate the bursting by sending stop streaming command. This will clear up the Continuous Voice Wake and bring the firmware to idle state.

8.1.3 Voice Wake with Keyword plus Command

Voice Wake with Keyword and Command (previously called Voice Sense) is a low-power state in which the IA61x is running a keyword detection algorithm and command detection after the keyword is detected. The Host Processor is powered-off, and is depending upon the IA61x to detect a user keyword to cause a wake-up event.

The IA61x is operating in a low-power state, with only enough signal processing activity to monitor in-coming audio signal from a local microphone. The incoming audio is monitored to ascertain whether specific voice utterances ("keywords") have been detected.

Note: CVQ Buffering is not supported in case of keyword and command detection.

Note: Routes with multiple input and/or output channels (Routes 15, 17, 18, 19, 20, 21, 22, 23, 24, and 25) are not supported when this feature is enabled in SysConfig.

8.1.3.1 Voice Wake Sequence

The host must place the IA61x into Keyword Detection state. The procedure is:

1. Download firmware binaries (SysConfig and Firmware)
2. The Host transfers keyword file (OEM, Host, Authentication) using the API command WriteDataBlock (0x802F 0xXXXX), with the command parameter specifying the keyword model length. This needs to be done once after every reset.
3. Set frame size to 16ms: 0x8035 0x0010
4. Set Sample rate to 16 KHz: 0x8030 0x0001
5. Disabled buffering: 0x802A 0x0000.
6. Set up VoiceSense 16k PDM route: 0x8032 0x0008 (Route 8) or 0x8032 0x0006 (Route 6)
7. Enable VS Keyword Detection mode: 0x8017 0x5003 0x8018 0x0000
8. Set power state so the IA61x enters a low power mode: 0x9010 0x0001 (Stage 0 for Route 8) or 0x9010 0x0002 (Stage 1 for Route 6).
9. Utter keyword at PDM mic. e.g. "Hello VoiceQ".
10. On successful AAD, the IA61x enters into Stage 1 for Keyword Detection.
11. If Voice Sense Algo detects keyword, the IA61x signals the host by generating an interrupt on HOST_IRQ pin, based on the settings from Get Event Response (rising edge default).
12. The chip at this point switches its power mode back to Stage 2 (Command Mode) and looks for command detection until time out.
13. The Host wakes up.
14. The Host sends API command GetEvent (0x806D 0x0000) to the IA61x. Response is sent from the IA61x indicating Voice Sense event type:
    - Voice Sense Response: no keyword detected: 0x0000
    - Voice Sense Response: keyword detected: Keyword ID

    This command resets the HOST_IRQ line.
15. Utter Command within specified timeout for command detection event to occur.

16. If Voice Sense Algo detects command, the IA61x signals the host by generating an interrupt on HOST_IRQ pin based on the settings from Get Event Response (rising edge default).

17. The Host sends API command GetEvent (0x806D 0x0000) to the IA61x. Response is sent from the IA61x indicating Voice Sense event type:

    - Voice Sense Response: no command detected: 0x0000
    - Voice Sense Response: Command detected: 0x01Command ID

8.1.4 Setting up Voice Wake/Continuous Voice Wake

8.1.4.1 After Chip reset/Power ON

The IA61x supports two routes for setting Voice Wake/Continuous Voice Wake, route 8 and route 6. Route 8 uses AAD functionality in internal mic and route 6 uses AAD in algorithm during stage 0.

The following flow diagram shows the flow for both Voice Wake and Continuous Voice Wake use cases. Note that after a keyword detection, the host can move on to another use case by stopping the Voice Wake route, or send a low power command to continue doing a Voice Wake / Continuous Voice Wake use case. The low power command is necessary to continue Voice Wake/Continuous Voice Wake, as Algo processing will stop the moment the keyword is detected (at the end of the keyword actually, detection might happen a little bit earlier). The only alternative to sending a low power command is to stop the Voice Wake route and set it up again.

8.1.4.1.1 Route 8 - Voice Wake Route (for cases when AAD of Internal Mic is used)

Route 8 uses AAD functionality in internal mic, so that DSP core can be in sleep state and achieve very low power during stage 0.

Note: It is highly recommended to use algorithm without AAD (or AAD disabled) for keyword detection when using route 8, as AAD detection is already done by internal mic. If AAD is enabled in algorithm then keyword detection will happen only if AAD is detected by both internal mic and algorithm, so performance may vary.

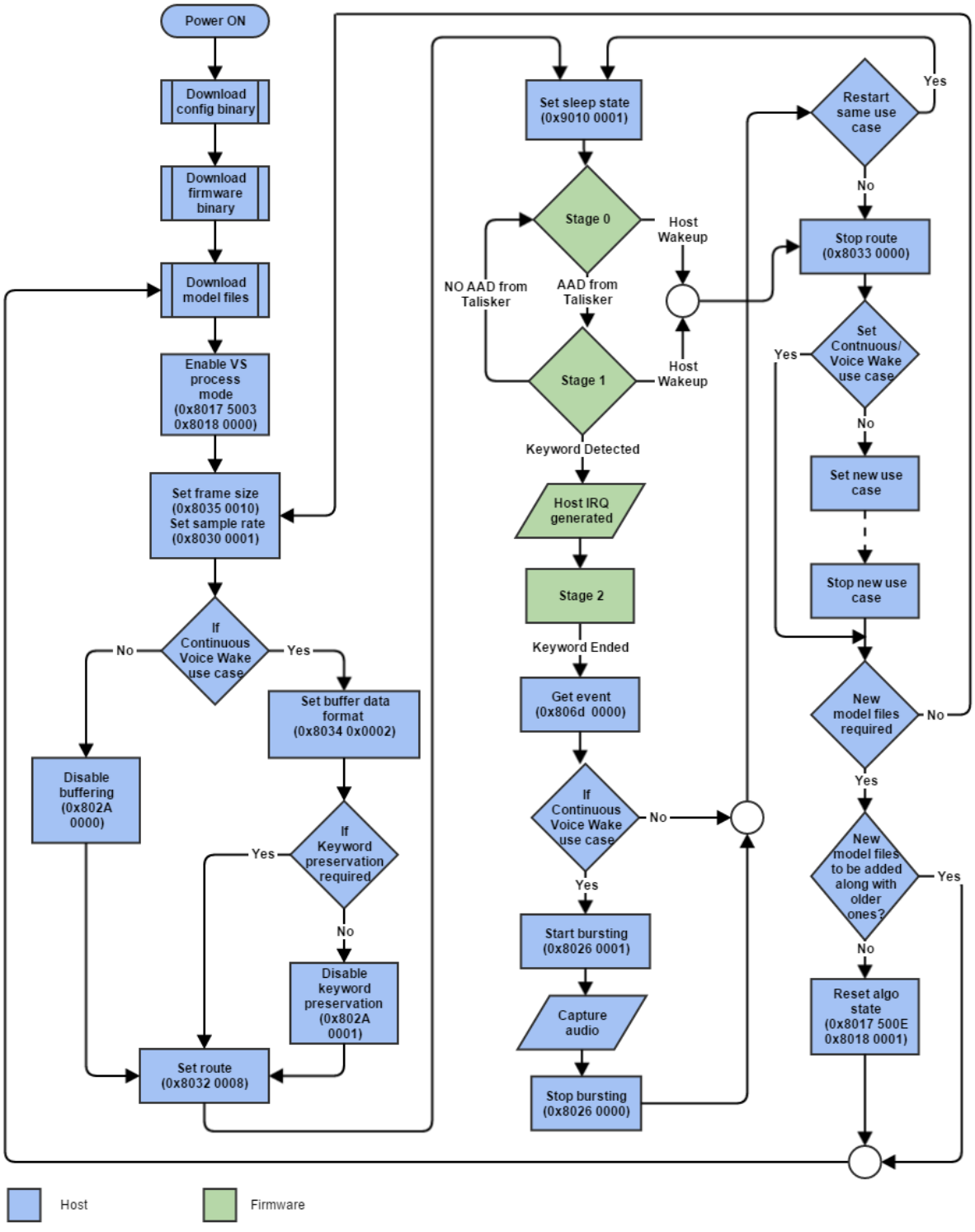

***Figure 17** Route 8*

8.1.4.1.2 Route 6 - Voice Wake Route (for cases when AAD of Internal Mic is not used)

Running Voice Wake/Continuous Voice Wake use cases in Route 6 is similar to Route 8, except the Set Power State command is sent with Low Power as the parameter (0x8010 0x0002), and the route must be torn down after every run. Route 6 is used by algorithms, which need AAD within algorithm.

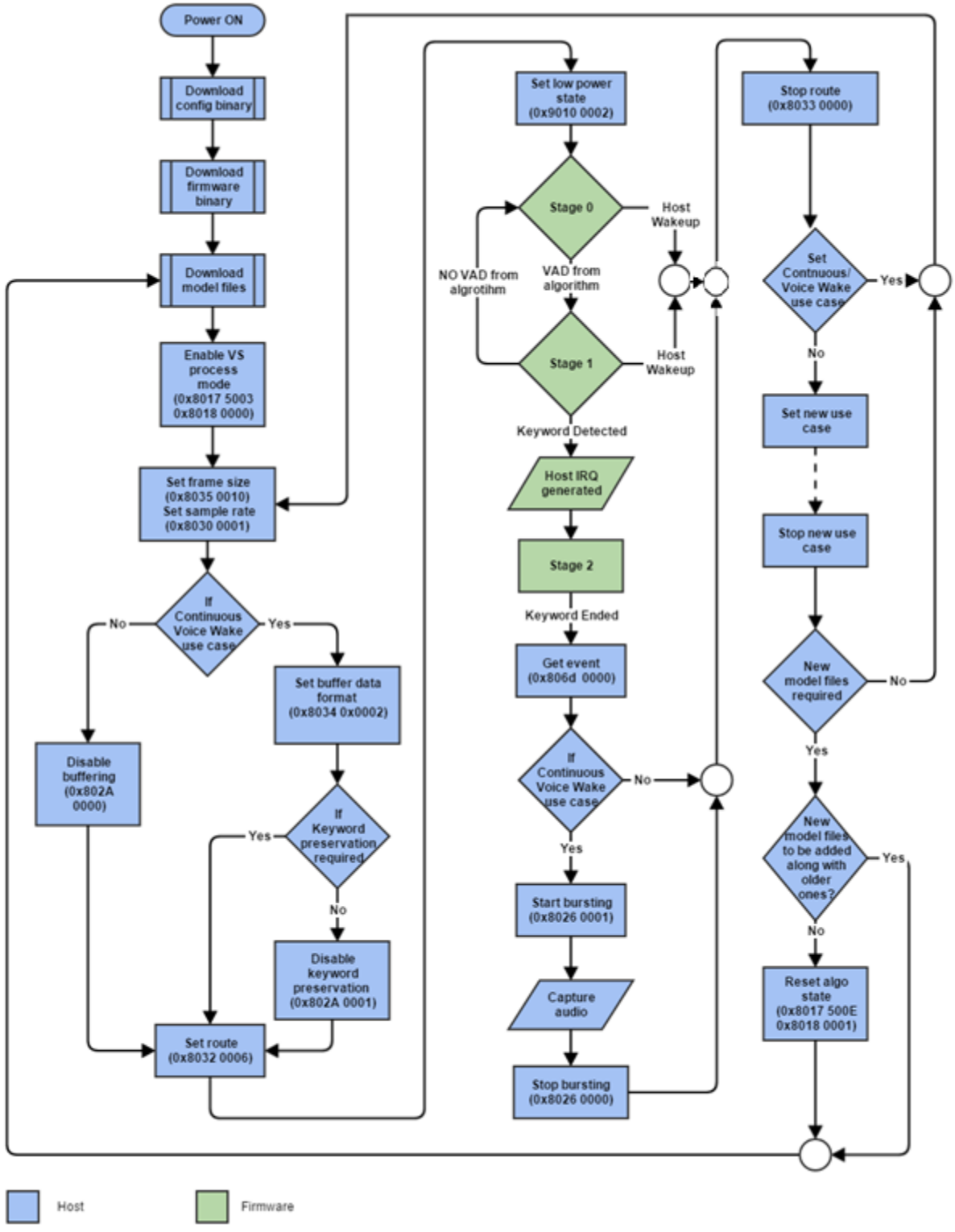

Figure 18 *Route 6 – AAD in algorithm*

8.1.4.1.3 Route 6 - Voice Wake Route (for cases when no AAD is not used)

If algorithm does not support AAD and if route 6 is set, then chip will not enter low power stage 0, instead it will continue to be in stage 1, till keyword detection. The host sequence for setting route 6 remains same irrespective of AAD support in algorithm.

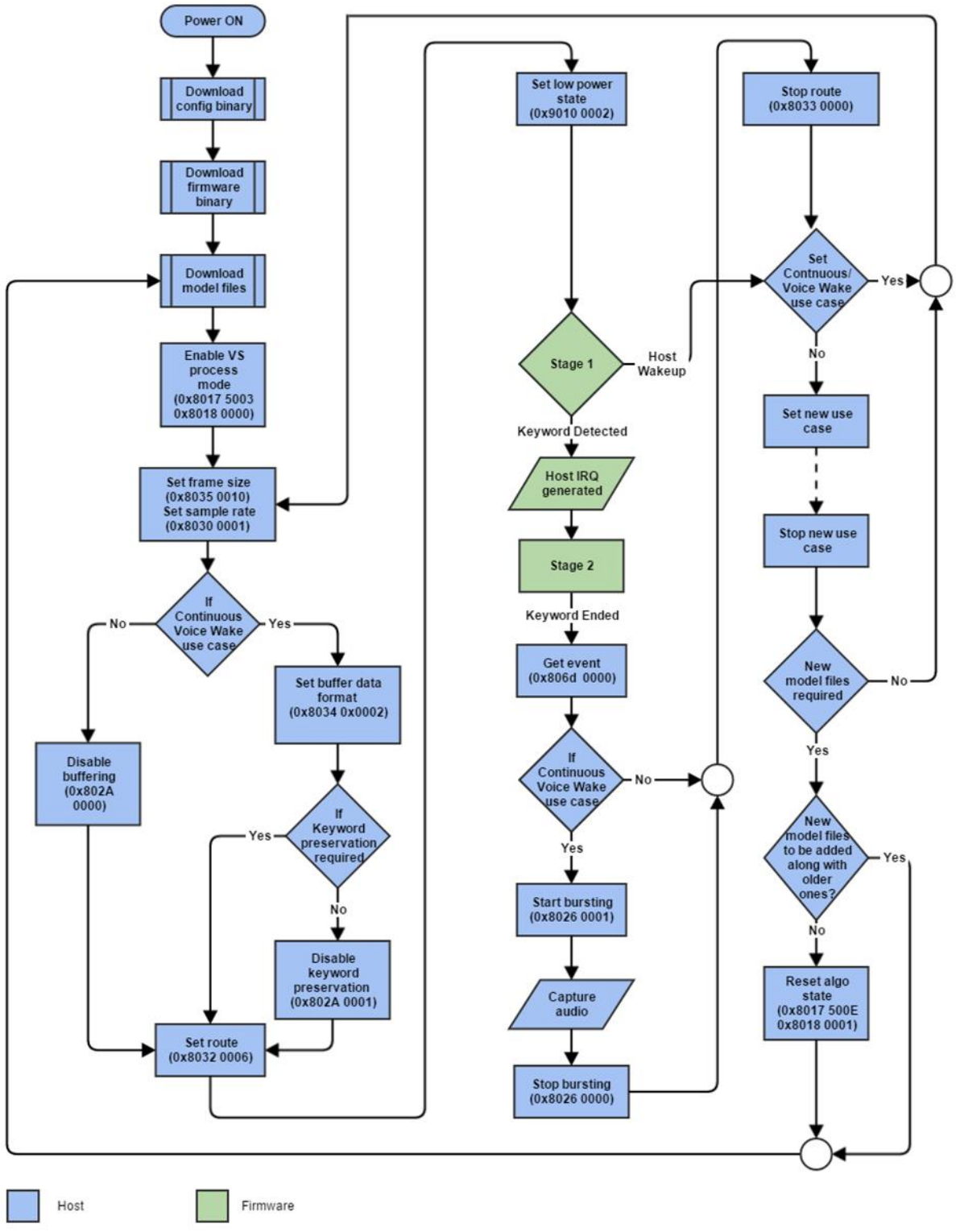

Figure 19 *Route 6 – No AAD*

8.1.4.2 After Keyword detection

If the IA61x is woken up by keyword detection, the keyword models, algorithm parameters, and route are preserved within the firmware. There is no need to take any additional steps after this kind of wake up. The IA61x can be put back to stage 0 by sending the low power command again, and expect it to wake up again after uttering the keyword.

8.1.4.3 After Host Wake up

If the Host wakes up the IA61x from stage 0 or stage 1, and Voice Wake/Continuous Voice Wake was set before, then it is not necessary to send the keyword models or algo parameters again. A host wake up will tear down the route, so it will be necessary to set up the route again along with its parameters (frame length, buffered data format). After that, a low power command can be sent, and the IA61x will wake up by saying the keyword.

8.1.4.4 Switching from Voice Wake/Continuous Voice Wake to other use cases and back to Voice Wake/Continuous Voice Wake

Setting the IA61x to other use case (like bypass mode) after a host wake up or keyword detection, then getting out of that use case will have no effect on the keyword models and algo parameters. In this scenario, route and its parameters (frame length, buffered data format) need to be setup again. After that, a low power command can be sent, and the IA61x will wake up by saying the keyword.

8.1.4.5 Switching to a different keyword model

If host wants to switch to a different keyword model, then it needs to first send the algo reset command (0x8017 0x500E 0x8018 0x0001), which will reset the older model files and then download the new model files. While doing this the route should not be active (i.e. route should be stopped).

8.1.4.6 Adding a new keyword model

Host can add new model files (with different keyword IDs) any time before setting up the route. If the new model file need to be added after route is set, then first the active route should be stopped, then algo reset command (0x8017 0x500E 0x8018 0x0001) need to be sent. This will reset the older model files. Download the older and new model files again.

8.2 PDM Bypass Mode

In PDM bypass mode, host/codec can connect to internal mic and bypass the BAF. This mode is useful for customer integration without software support.

In PDM Bypass mode following internal mic pins are routed to host Pins.

- P0 (Input) →MCLK (Output)
- P1 (Output)←MSD (Input)

8.2.1 PDM Bypass mode In SBL

The Host or Codec needs to follow the sequence shown in **Error! Reference source not found.** to enable PDM bypass mode in SBL without using an API command. The PDM Clock on P0 pin should be more than 512 kHz to enable PDM bypass mode1 in SBL without the API command.

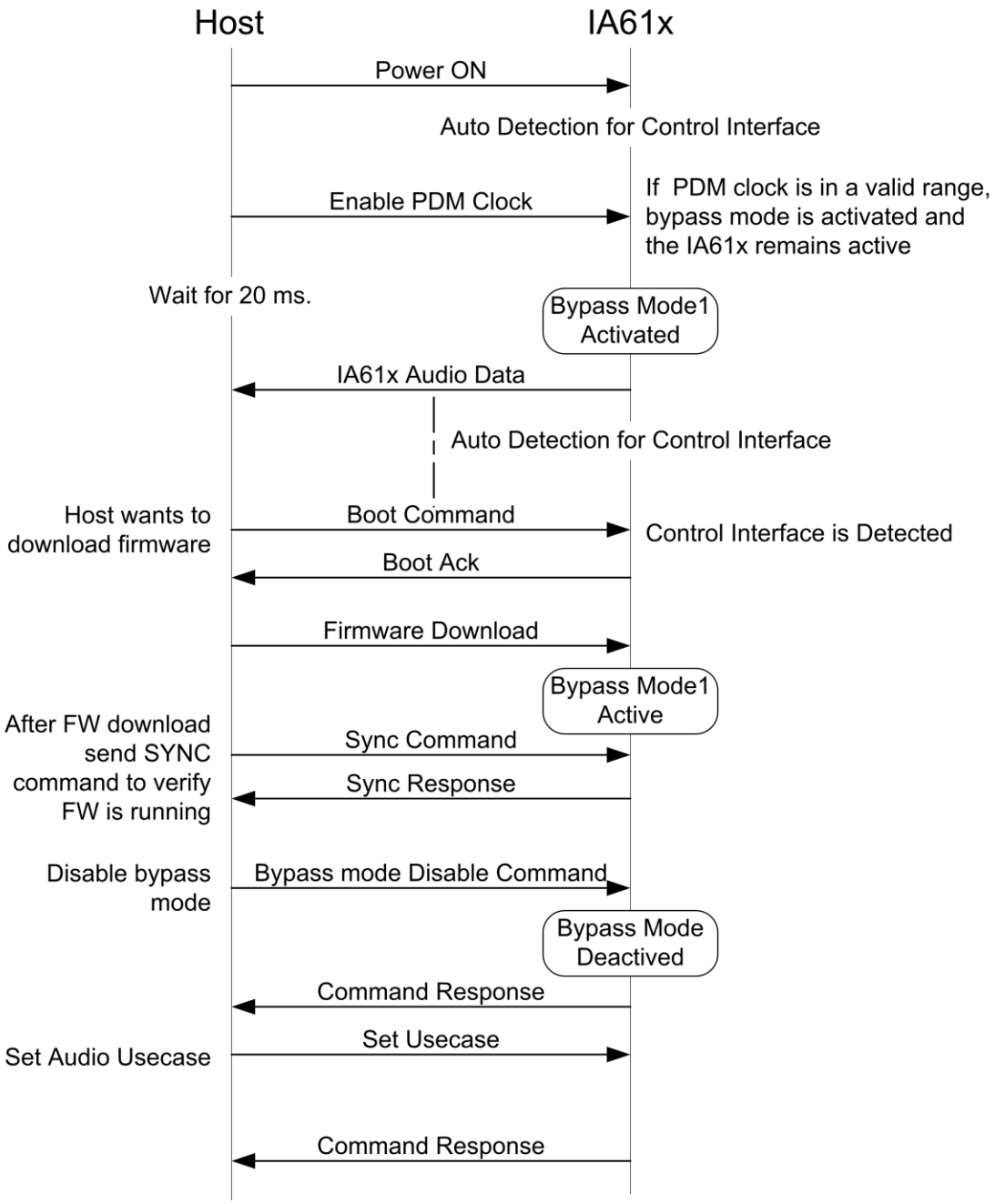

Figure 20 *PDM bypass mode activate sequence in SBL*

PDM bypass mode can also be activated or deactivated using an API command (See Section **Error! Reference source not found.**).

Note: Once the control interface is detected, PDM bypass mode can only be activated using an API command.

Error! Reference source not found. shows the Sleep/Wakeup sequence during Bypass mode.

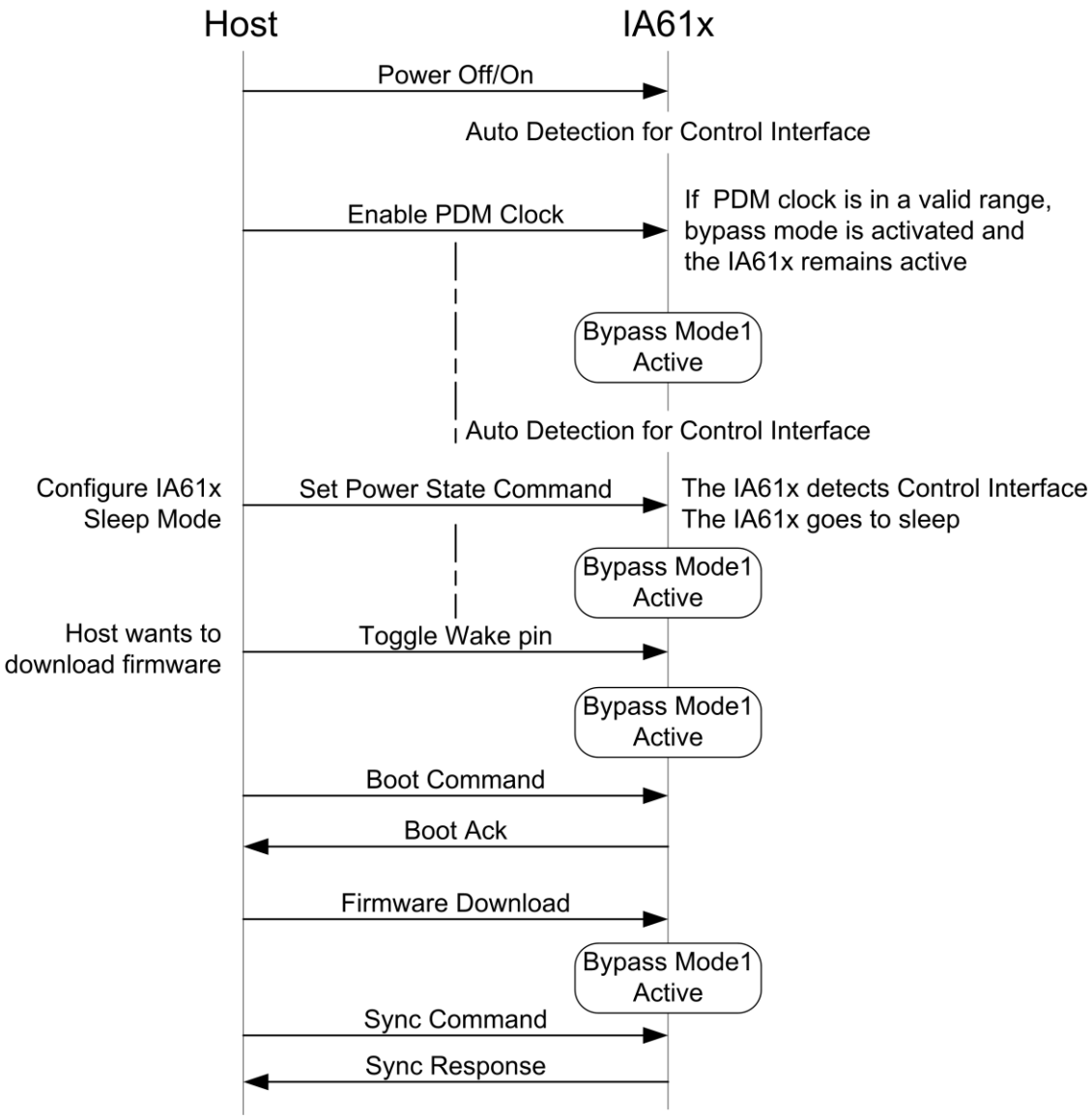

Figure 21 *Sleep/Wakeup during PDM bypass mode*

Error! Reference source not found. shows the Voice Call sequence in SBL during PDM bypass mode.

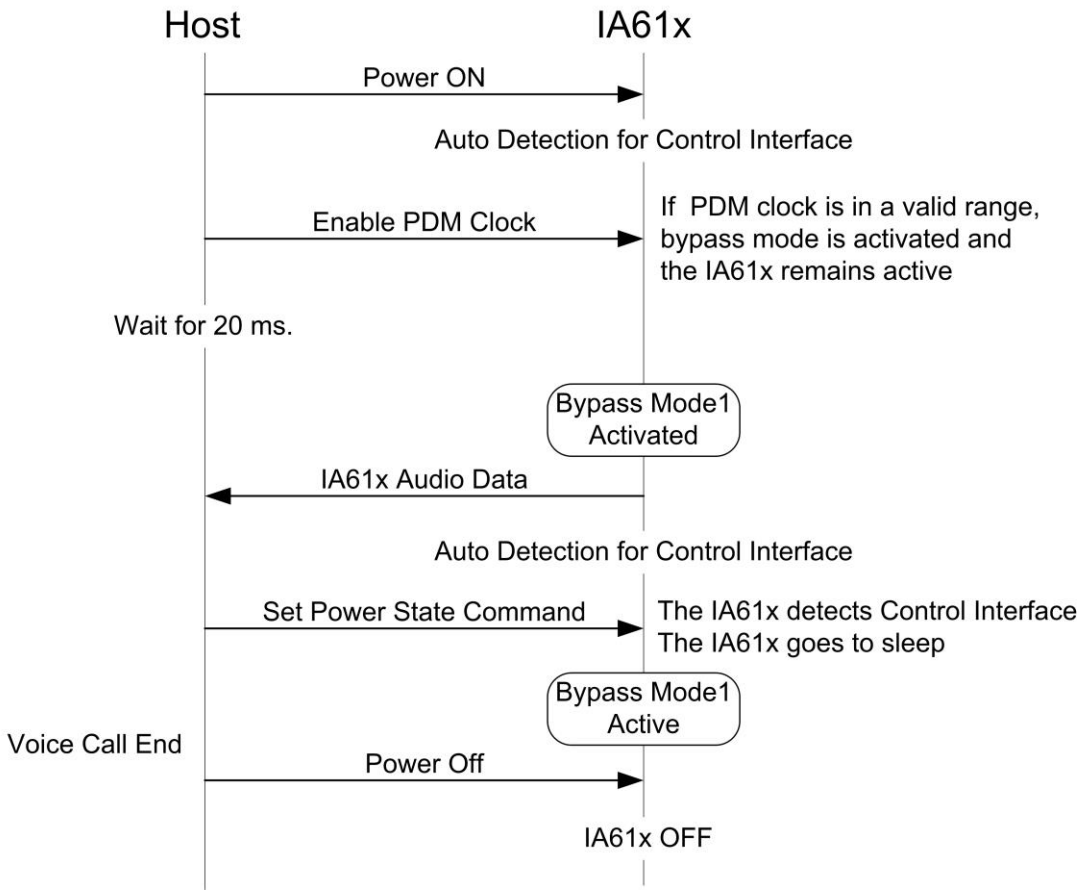

Figure 22 *Voice Call using PDM Bypass Mode*

8.2.2 PDM Bypass mode in BoskoApp

PDM bypass mode can also be enabled or disabled using API commands (See Section **Error! Reference source not found.**). The sequence shown in **Error! Reference source not found.** is for PDM bypass mode enable/disable using API commands in BoskoApp.

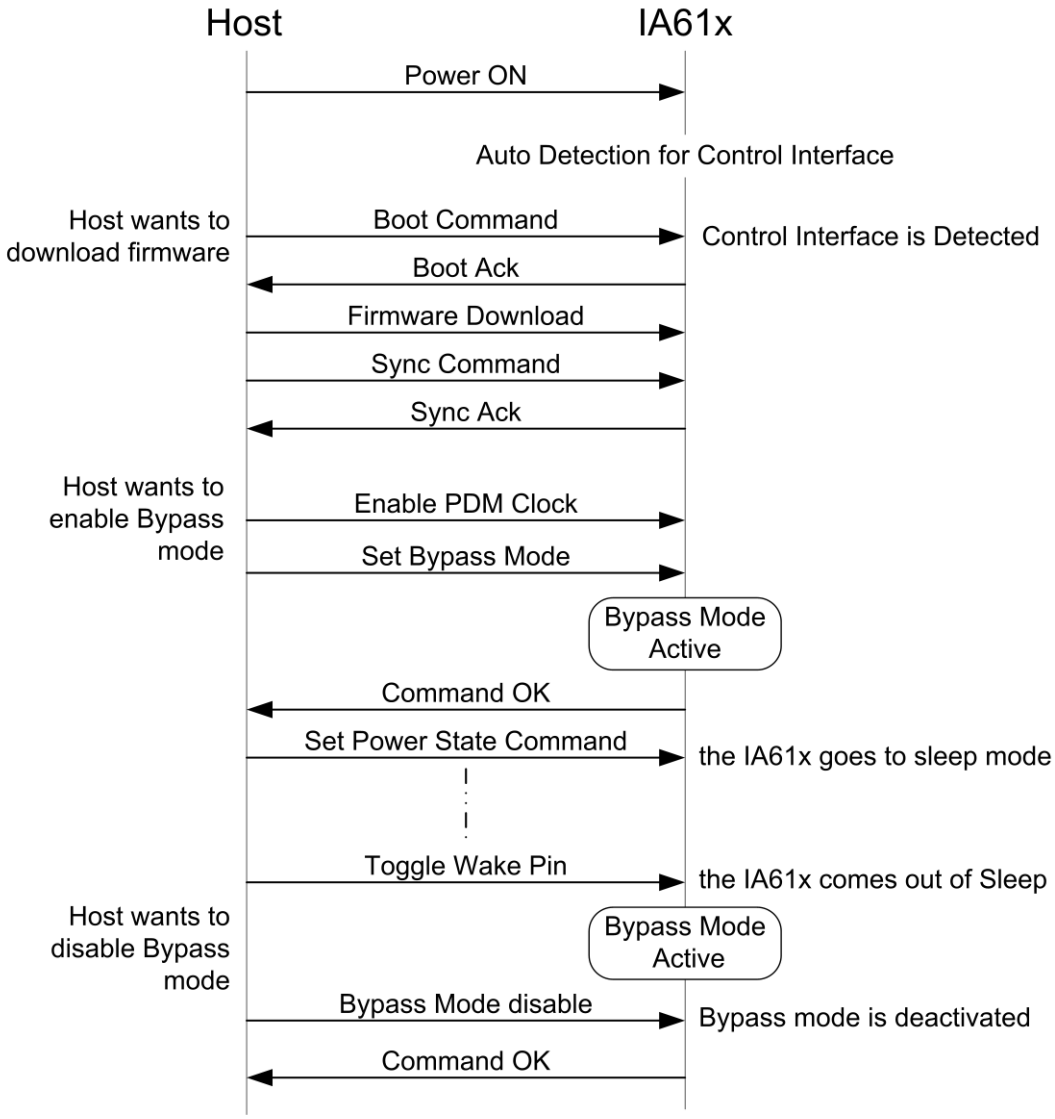

Figure 23 *PDM Bypass Mode enable/disable using API Command in BoskoApp*

Error! Reference source not found. shows the sequence for PDM bypass during Voice Wake.

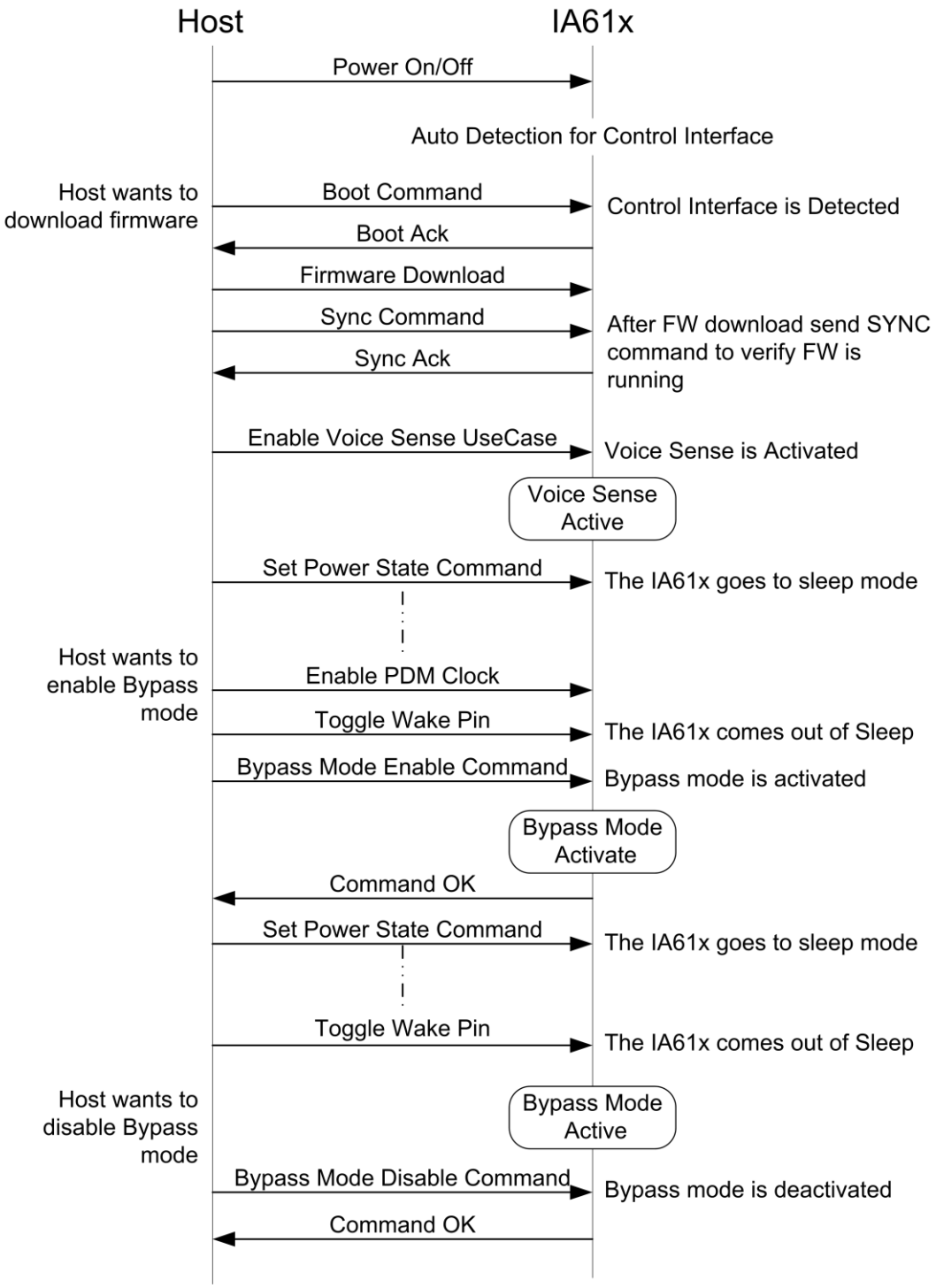

Figure 24 *PDM Bypass Mode during Voice Wake*

Contact your Knowles Applications Engineer for your more information on PDM bypass mode.

8.3 Pass-Through Routes

There are three types of pass-through routes: Hardware Pass-through, Software Pass-through and Low Latency Software Pass-through. For all pass-through routes, Host/Codec is the audio clock master.

18. Hardware Pass-through (PDM Bypass Mode)

- When using hardware pass-through, the BAF of the IA61x can go to sleep mode
  - PDM input from internal mic to PDM out
  - PDM input from internal mic to SoundWire PDM out
  - PCM input (data in) to PCM output (data out). This is like loop back. (This route is not supported during sleep mode)

19. Software Pass-through

- When using software pass-through, the BAF of the IA61x cannot go to sleep mode as it is running system firmware to support the route.
  - PDM input from internal mic to PCM out
  - PDM input from internal mic to SoundWire PCM out
  - SoundWire PCM input to SoundWire PCM out
  - SoundWire PDM input to SoundWire PDM out
  - SoundWire PDM input to SoundWire PCM out

20. Low latency Software Pass-through

- When using Low Latency Pass-through, the firmware uses specialized instructions to transfer the every sample when it arrives immediately, and most of time it is in idle.
  - PDM input from internal mic to PCM out
  - PDM input from internal mic to SoundWire PCM out

8.4 Barge-In feature

Barge-In is a feature in the IA61x that has both AEC and VQ algorithms. AEC internally does echo cancellation and sends clean speech (mic data) to the VQ algorithm for KW detection.

Barge-In build will support the following modes,

- Only AEC
- Only VQ
- Barge-In (AEC + VQ)

8.4.1 Only AEC

By default, AEC is enabled in Barge-In. If the host wants to run AEC only, then it needs to disable VQ by sending the algo params (0x8017 5019, 0x8018 0000).

The AEC algorithm needs two inputs, one input is AEC reference (PCM) and the other input is mic data (PDM). The output of AEC is clean speech and is sent to the PCM port.

Note: Only Route#31 and Route#33 support using only AEC, since these routes have AEC output channels.

8.4.2 Only VQ

By default, VQ is enabled in Barge-In. If the host wants to run VQ only, then it needs to disable AEC by sending the algo params (0x8017 6004, 0x8018 0000).

VQ in this case will work the same as a normal VQ use case.

Note: All Barge-In routes will support this mode.

8.4.3 Barge-In

A Barge-In VoiceQ route is a route that is used to detect a keyword when music playback is happening.

8.4.3.1 Barge-In VoiceQ Sequence

The host must put the IA61x into a Keyword Detection state. The procedure is:

1. Download firmware binaries (SysConfig and Firmware)
2. The Host transfers keyword model file (OEM) using the API command WriteDataBlock (0x802F 0xXXXX), with the command parameter specifying the keyword model length. This needs to be done once after every reset.
3. Set frame size to 16ms: 0x8035 0x0010
4. Set Sample rate to 16 KHz: 0x8030 0x0001

5. Set audio port clock to 1536 KHz: 0x8042 0x0003
6. Config data port to 48KHz: 0x802C 0x4000
7. Set up Barge-In VoiceSense 48K PDM route: 0x8032 0x001F (Route 31) or 0x8032 0x0020 (Route 32) or 0x8032 0x0021 (Route 33) or 0x8032 0x0022 (Route 34).
8. Enable VS Keyword Detection mode: 0x8017 0x5003 0x8018 0x0000
9. Utter keyword at PDM mic.
10. If Voice Sense Algo detects keyword, the IA61x signals the host by generating an interrupt on HOST_IRQ pin, based on the settings from Get Event Response (rising edge default).
11. The Host wakes up.
12. The Host sends API command GetEvent (0x806D 0x0000) to the IA61x. Response is sent from the IA61x indicating Voice Sense event type:
    - Voice Sense: no keyword detected: 0x0000
    - Voice Sense: keyword detected: Keyword ID

Note: Host IRQ line can be configured to either I2S_SDO or UART_Tx line via SysConfig for UART interface. By default, I2S_SDO line is Host IRQ line for both I2C and UART.

Note: Streaming is possible only for one endpoint at a time due to limited MIPS and memory.

8.5 Dual Algo Support

Dual Algo support is a new feature in the IA61x and it will take two algo libraries, which are from either third party or from our Knowles and built in single binary.

EXT_DA_OPT build option is require for building the dual algo support builds. Following are valid value for EXT_DA_OPT.

EXT_DA_OPT= AEC_VQ (BargeIn with KN VQ)

EXT_DA_OPT= AEC_Hotword (BargeIn with Google Hotword)

EXT_DA_OPT= BNE_VQ (KN VQ + KN BNE algo)

EXT_DA_OPT= SRE_VQ (KN VQ + KN SRE algo)

Note: Streaming is possible only one endpoint at a time due to limited MIPS and memory.

8.6 IA61x in Multi-Microphone Array

There are two different ways to use IA61x in mic array

1. PDM Interface Pad (P0/P1)Sharing: PDM interface line (P0/P1) with external PDM mic
2. IA61x External Mic Connection: Get external PDM data in IA61x and IA61x gives two channel PDM data

8.6.1 PDM Interface Pad (P0/P1) Sharing

IA61x can be used as standard PDM mic where it can share PDM interface pins with other mics. PDM bypass mode cannot be used when external mics are sharing PDM pads (P0/P1) with IA61x. IA61x PDM interface is not capable of floating PDM interface pad (P0/P1) when it is in PDM bypass mode. Firmware download is required to share PDM interface pad with external mic

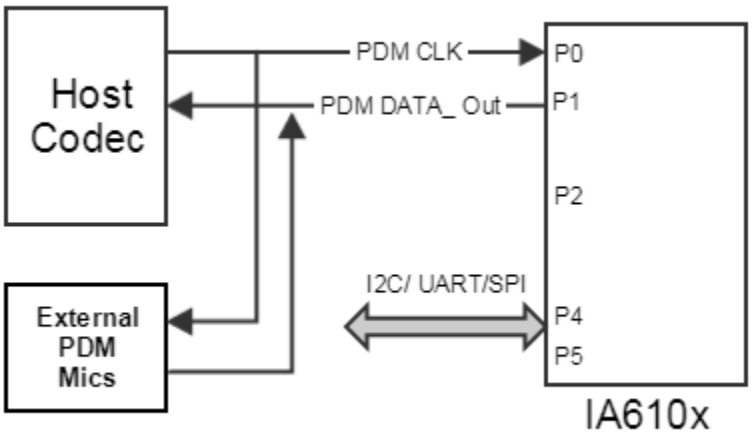

Figure 25 *PDM Interface Pad(P0/P1) shared with external mic*

IA61x can share PDM interface pad with an external mic as shown **Error! Reference source not found.** IA61x can be configured to run 1ch pass-through routes, which pass through internal mic data to PDM interface pad (P0/P1). IA61x drives Left/Right channels based on routes selection.

Table 14 *Routes supported for PDM pad sharing*

Route No.	Input Port	Output Port	Route Type
01	Internal Mic	PDMO0 (Left channel)	Frame-Based
02	Internal Mic	PDMO1 (Right channel)	Frame-Based
41	Internal Mic	PDMO0 (Left channel)	Low Latency
42	Internal Mic	PDMO1 (Right channel)	Low Latency

8.6.2 IA61x External Mic Connection

IA61x can be configured to get external mic data inside and pass-through those audio data on PDM interface pad (P0/P1). P1 or P2 pad can be used to connect external mic as mention in pad configuration table

8.6.2.1 External mic on P2 pin

External mic can be connected to P2 pin as shown in **Error! Reference source not found.**. IA61x sample external mic data and internal mic data, then route them to PDM interface pad (P0/P1).

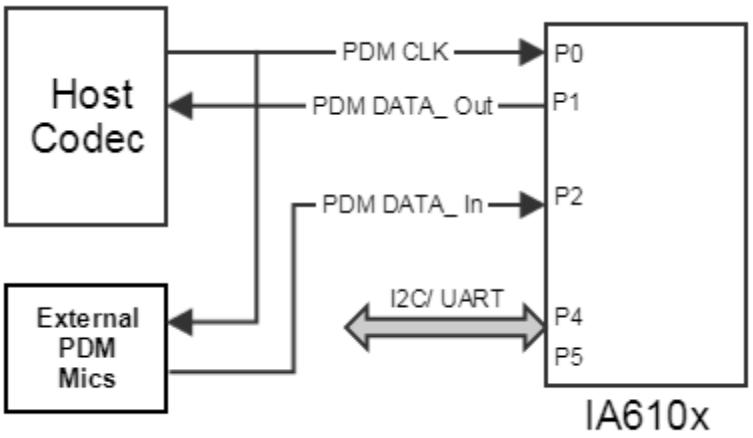

Figure 26 *External mic connects on P2 pad*

Following routes expects external mic on P2 pin.

Table 15 *Routes supported for PDM pad P2*

Route No.	Input Ports		Output Ports		Route Type
26	Internal Mic	External MIC1	PDMO0 (Left channel)	NA	Frame-Based
27	Internal Mic	External MIC1	PDMO0 (Left channel)	PDMO1 (Right channel)	Frame-Based
43	Internal Mic	External MIC1	PDMO0 (Left channel)	PDMO1 (Right channel)	Low Latency
47	Internal Mic	External MIC1	NA	NA	Frame-Based

Note:
Route 26 and Route 27 is not available for SPI interface.
Route 43 only available for UART interface only.

8.6.2.2 External mic on P1 pin

External mic connects to P1 pin as shown in **Error! Reference source not found.**. IA61x sample external mic data and internal mic data, then route one PDM channel to PDM interface pad (P0/P1). In this configuration P1 pin act as PDM data input and output at same time.

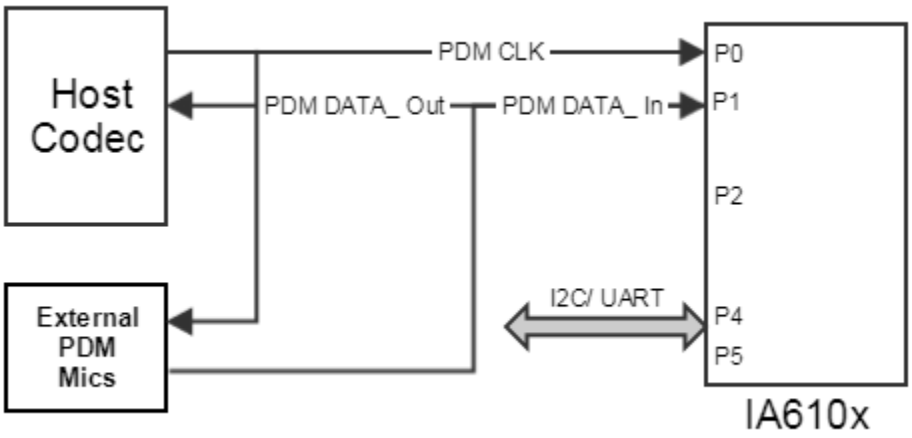

Figure 27 *External mic connects on P1 pad*

Following routes expects external mic on P1 pin.

Table 16 *Routes supported for PDM pad P1*

Route No.	Input Ports		Output Ports		Route Type
28	Internal Mic	External MIC2	PDMO0 (Left channel)	NA	Frame-Based
29	Internal Mic	External MIC1	NA	PDMO1 (Right channel)	Frame-Based
47	Internal Mic	External MIC1	NA	NA	Frame-Based

Note:

Eternal mic2 data should sample on falling edge of clock

External mic1 data should sample on rising edge of clock

SysConfig parameter change is required to run Route 47 on P1 pin

Chapter 9: API Command Syntax

A bit is allocated in the command API field to flag whether the command response is required or not. The syntax is as follows:

Command Syntax

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
10		0	SR	Rsvd				Command ID								Command Value															

Bits 31-29: Always set to a binary value of 100 (0x04)

Bit 28: SR Suppress Response:

0 – Allow response for this command

1 – Do not send a response for this command to Host

Bit 27-24: Reserved for future use

Bit 23-16: Command ID

Valid range is 0x00 – 0x7F

Bit 15-0: Command Value for either Set or Get commands

Valid range is command dependent.

Note: API Commands are always 32-bits long as shown.

9.1 API Response Code Syntax

Each command sent to the IA61x is acknowledged with a response code:

Response Code

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Response Code																															

The three valid response code types are:

Legal command: The same command code is returned for acknowledgement that the command is legal.

Illegal command: Error response code (0xFFFF xxxx) is used if the command is illegal (i.e. unsupported) or if a legitimate command is used with an illegal/unsupported parameter. In the response code (0xFFFF xxxx),"xxxx" represents the unsupported command code.

Response not ready: A "zero" response code (0x0000 0000) is sent when a reply (either acknowledgment or requested data) is not ready. This response code should be treated by the Host as a Response-Not-Ready indicator.

9.2 Suppress Response

The commands listed in this document can either be sent one at a time or can be sent in bulk over the I^2C and/or UART interface. Sending commands in bulk is referred to as "One-Shot" mode.

In this mode, the command over I^2C assumes one of the following patterns:

```
<Address><Cmd_1><Address><Cmd_2>…<Address><Cmd_n>
(Normal)
```

OR

```
<Address><Cmd_1><Cmd_2>…<Cmd_n>                    (One-shot)

Where <Address> is the IA61x I2C device address.
```

It is expected that the Host processor will read acknowledgements for all commands before sending out any other commands during chip integration into the target hardware. However, this could be a significant performance drain.

This version of the API provides the option to suppress the response for each command. Production host software can take advantage of this feature to speed up the host-chip interface processing.

9.3 API Command Summary

A Summary of Boot API commands is shown in **Error! Reference source not found.** and a summary of API commands is shown in **Error! Reference source not found.**.

Table 17 *Summary of Boot API Commands*

Cmd name	CmdID	Section	Notes
Sync	0x8000	**Error! Reference source not found.**	Available in BoskoApp
GetChipID	0x800E	**Error! Reference source not found.**	Available in BoskoApp
UART RateRequest	0x8019	**Error! Reference source not found.**	Available in BoskoApp
PDM Bypass mode	0x8040	**Error! Reference source not found.**	Available in BoskoApp
Set Audio Port Clock Frequency	0x8042	**Error! Reference source not found.**	Available in BoskoApp
Set Audio Data port	0x8004	**Error! Reference source not found.**	
Set Clock Source	0x8011	**Error! Reference source not found.**	Available in BoskoApp
Set Digital Hardware Pass-through	0x8052	**Error! Reference source not found.**	Available in BoskoApp
Get Info	0x8022	**Error! Reference source not found.**	Available in BoskoApp
Set SPI Sample Config	0x8023	**Error!**	Available in BoskoApp

		Reference source not found.	
Set UART Stop Bits	0x8012	**Error! Reference source not found.**	Available in BoskoApp
Power down Internal Oscillator	0x8043	10.12	
Enable/Disable JTAG	0x8061	10.13	
Set PAD control Register	0x8062	10.14	

Table 18 *Summary of API Commands*

Cmd name	CmdID	Section	Description
Sync	0x8000	**Error! Reference source not found.**, **Error! Reference source not found.**	Available in SBL
GetChipID	0x800E	**Error! Reference source not found.**	Available in SBL
RateRequest	0x8019	**Error! Reference source not found.**	Available in SBL
GetDeviceParm	0x800B	**Error! Reference source not found.**	
SetDeviceParmID	0x800C	**Error! Reference source not found.**	
SetDeviceParm	0x800D	**Error! Reference source not found.**	
SetPowerState	0x8010	**Error! Reference source not found.**	

GetEvent	0x806D	**Error! Reference source not found.**	
GetAlgorithmParm	0x8016	**Error! Reference source not found.**	
SetAlgorithmParmID	0x8017	**Error! Reference source not found.**	
SetAlgorithmParm	0x8018	**Error! Reference source not found.**	
WriteDataBlock	0x802F	**Error! Reference source not found.**	
ReadDataBlock	0x802E	**Error! Reference source not found.**	
SetPreset	0x8031	**Error! Reference source not found.**	
SetDigitalGain	0x8015	**Error! Reference source not found.**	
GetDigitalGain	0x801D	**Error! Reference source not found.**	
GetSigRMS	0x8013	**Error! Reference source not found.**	
GetSigPeak	0x8014	**Error! Reference source not found.**	
OutputKnownSig	0x801E	**Error! Reference**	

		source not found.	
SelectRoute	0x8032	**Error! Reference source not found.**	
StopRoute	0x8033	**Error! Reference source not found.**	
SetStreaming	0x8025	**Error! Reference source not found.**	
SelectStreaming	0x8028	**Error! Reference source not found.**	
PDM Bypass mode	0x8040	**Error! Reference source not found.**	
Set Audio port clock Frequency	0x8042	**Error! Reference source not found.**	Available in SBL
Set Clock Source	0x8011	**Error! Reference source not found.**	Available in SBL
Set Digital Hardware Pass-through	0x8052	**Error! Reference source not found.**	Available in SBL
Get Info	0x8022	**Error! Reference source not found.**	Available in SBL
Set SPI Sample Config	0x8023	**Error! Reference source not found.**	Available in SBL
Set UART Stop Bits	0x8012	**Error! Reference source not found.**	Available in SBL

Set Diagnostic Config	0x801C	**Error! Reference source not found.**	
Get BAF Info	0x8027	**Error! Reference source not found.**	
Set Sample Rate	0x8030	**Error! Reference source not found.**	
Config Data Port	0x802C	**Error! Reference source not found.**	

9.4 Common Command Parameters

An Endpoint is used in multiple commands to specify a route terminus point. The general definition is:

- Endpoint: A terminal or IA61x intermediary point in a route

9.4.1 Endpoint Parameter Definitions

An Endpoint parameter is an 8-bit field with the following format:

- Reserved: bit: [8]
- Package ID: bits: [6:3]
- Direction: bit: [2]
- Endpoint number: bits:[1:0]

These subfields are defined below:

PackeIDs: refer to: **Error! Reference source not found.**, **Error! Reference source not found.**

Direction: refer to: **Error! Reference source not found.**, **Error! Reference source not found.**

Endpoint Number: refer to: **Error! Reference source not found.**, **Error! Reference source not found.**

9.4.2 Package IDs

Package Id represents each different routing block in the system that is responsible for generating an audio frame, processing audio frames, or forwarding audio frames. The IA61x

includes the package Ids shown in **Error! Reference source not found.** as part of routing design:

Table 19 *Package ID information*

Package Name	ID	Description
PKG_STREAM_MGR	0x1	Stream manager
PKG_BUFFER_MGR	0x2	Buffer Manager
PKG_BAF_MGR	0x3	BAF Manager

9.4.3 Direction

Table 20 *Direction*

Direction	Value	Endpoint direction in terms of firmware	
In	0	StreamManager Tx BufferManager Rx BAF Manager Rx	For Stream manager, Tx is “0” as it is receive point in FW and transmit data to HW For buffer manager and BAF manager. Rx is “0” as it is receive point in FW
Out	1	StreamManager Rx BufferManager Tx BAF Manager Tx	For Stream manager, Rx is “1” as it is transmit point in FW and receive data from HW. For buffer manager and BAF manager. Tx is “1” as it is transmit point in FW

9.4.4 Endpoint Numbers

Endpoint numbers are defined in **Error! Reference source not found.**, **Error! Reference source not found.**:

Table 21 *Endpoint numbers*

Endpoint Type	ID	Description	Rx or Tx	Endpoint Value (hex)	Endpoint Value (Dec)
Stream Manager	0	Rx endpoint 0	RX	0xC	12
	1	Rx endpoint 1	RX	0xD	13
	2	Rx endpoint 2	RX	0xE	14
	3	Rx endpoint 3	RX	0xF	15
	0	Tx endpoint 0	TX	0x8	8
	1	Tx endpoint 1	TX	0x9	9
	2	Tx endpoint 2	TX	0xA	10
	3	Tx endpoint 3	TX	0xB	11
Buffer Manager	0	Rx endpoint 0	RX	0x10	16
	1	Rx endpoint 1	RX	0x11	17

	0	Tx endpoint 0	TX	0x14	20
BAF Manager	0	Rx endpoint 0	RX	0x18	24
	1	Rx endpoint 1	RX	0x19	25
	2	Rx endpoint 2	RX	0x1A	26
	3	Rx endpoint 3	RX	0x1B	27
	0	Tx endpoint 0	TX	0x1C	28
	1	Tx endpoint 1	TX	0x1D	29
	2	Tx endpoint 2	TX	0x1E	30
	3	Tx endpoint 3	TX	0x1F	31

Chapter 10: Boot API Commands

This chapter describes the Boot mode operation API commands that enable the host to control certain Boot mode features.

Note: SBL API commands do not send a response if the no-response bit (bit 28) is set in the API (commands starting with 8). No Response SBL API commands are supported if bit 28 is set (commands starting with 9) and the SBL will not send any response for that particular command.

10.1 Sync

The Sync API command can be used by the Host to ensure that the IA61x is ready to receive commands.

See **Error! Reference source not found.** for allowable Control_Mode values.

Table 22 *Sync Command Values*

Command	Code	Value
Sync	0x8000	0x0000 = Always. Other values are ignored.

Response: Always returns 0x8000 FFFF in Boot mode operation.

10.2 Get CHIP ID

The Get CHIP ID command is used by the Host to verify a chip revision, to facilitate use of chip specific features, and to download the proper revision of chip firmware.

See **Error! Reference source not found.** for allowable Value values.

Table 23 *GetChipID values*

Command	Code	Value
GetChipID	0x800E	0x0610 = IA61x

10.3 Set UART Rate Request

The SetRateRequest API command enables the Host to request a particular UART baud rate for subsequent boot mode API commands and for system configuration image and firmware image downloads. The Baud rate set in SBL will remain same after firmware download and sleep wakeup.

Baud rate information is provided in the baud rate request command; the last two bytes of the command are used to set the baud rate. Once the UART rate request command is sent, the SBL will switch to rate calibration mode and will be waiting for four bytes of 0's over UART with the new baud rate. Follow these steps to change UART Baud rate.

1. Send a 0x8019 0xxxxx command to start calibration (No response for the command)
2. Configure the UART with new baud rate
3. Send 0x00000000 over UART with configured baud rate.
4. Read the response for the 0x8019 0xxxxx.

It is recommended to use Set Rate Request command only once. Whenever the UART command/response does not work, it may require recalibration of the UART rate. UART rate recalibration is likely to be required when the clock source has been switched, internal OSC drift over time, or broken host communication over UART. SBL or FW can start recalibration process if it receives 0xFFFFFFFF over UART from the host system. 0xFFFFFFFF should be sent with the last configured baud rate. The system can adapt to ±5% internal clock change. Note that the UART rate is taken as reference to adapt to the internal clock change.

See **Error! Reference source not found.** for allowable Value values.

Table 24 *UART SetRateRequest Values*

Command	Code	Value
UART SetRateRequest	0x8019	0xXXXX where XX is a 16-bit field. XXXX: requested baud rate /100 0x1200 = 460.8 kHz 0x2400 = 921.6 kHz 0x2710 = 1.000 MHz 0x2800 = 1.024 MHz 0x2D00 = 1.152 MHz 0x4E20 = 2.000 MHz 0x5000 = 2.048 MHz 0x7530 = 3.000 MHz 0x7800 = 3.072 MHz

10.4 PDM Bypass Mode

The PDM Bypass mode allows the host to access the internal mic. In this mode, the IA61x enables digital hardware pass-through for the PDM signal.

Note: PDM bypass mode needs to disabled when moving to any other bypass mode or setup new route.

Note: SoundWire DHWPT modes 04, 06, 07, 08, 09 are only available in BoskoApp.

Note: SoundWire DHWPT features can only enable if SoundWire Bus Frequency is in between 3.5 MHz and 17.5 MHz.

Note: 0xY nibble is only applicable in BoskoApp.

Table 25 *PDM Bypass Command*

Command	Code	Value
PDM Bypass Mode	0x8040	0xAAYX values 0xX values 0x0: PDM Bypass mode Disable 0x1:PDM Bypass mode 1 0x2: Reserved 0x3: Reserved 0x4: SoundWire DHWPT (PDM0) 0x5: Reserved 0x6: SoundWire DHWPT (PDM2) 0x7: SoundWire DHWPT (PDM0/2) 0x8: SoundWire DHWPT (PDM2/3) 0x9: SoundWire DHWPT (PDM0/2/3) 0xY values 0x0: PDM clock is less than 2.1MHz 0x1: PDM clock is greater than 2.1MHz 0xAA values This value is valid for SoundWire DHWPT only. This value is used as the divider of the SoundWire bus frequency to generate a final PDM clock frequency. If nothing is specified it takes the default divider 0x02. 0x00 and 0x01 are invalid; if specified, the IA61x will use the default value. Max. divider value is 0x31 Example: If SoundWire Bus clock is 12.288 MHz. Required PDM Clock is 1.536 MHz then the divider value should be 0x08.

10.5 Set Audio Port Clock Frequency

The Set Audio port clock frequency command allows the host to communicate external audio port clock frequency provided by host. This clock can be either I2S bit clock or PDM clock or SoundWire Clock. Clock value set using this API command is being used to configure internal clock divider while route is being setup. The Host can get the current audio port clock frequency by using the Get info command (0x8022). If the audio port clock frequency is not set then the Get info command response is 0x0008 (Reserved).

It is recommended to use the minimum clock frequency of 768 kHz. As per the Internal Mic design, the max band of audio signal in PDM should be calculated as:

(Port Clock Freq / 96)

For example, with 768 kHz, the max band of recorded audio signal will be 8 kHz.

Table 26 *Set Audio port clock Frequency Command*

Command	Code	Value
Set Audio port clock Frequency	0x8042	0x0000: 512 kHz 0x0001:768 kHz 0x0002:1024 kHz 0x0003:1536 kHz 0x0004:2048 kHz 0x0005:3072 kHz 0x0006:6144 kHz 0x0007:12288 kHz 0x0008:Reserved (Default) 0x0009: 9600Khz 0x000A: 2400Khz 0x000B: 4800Khz 0x000C: 4608Khz

10.6 Set Audio Data port

The Set Audio data port command allows the host to set the audio port to be used in the system. The default host audio data port is PDM. The Host needs to send this command to set audio data port to I^2S, PCM, or SoundWire. This command is valid in SBL only.

Note: If PDM bypass mode is enabled, then it should be disabled before setting the audio data port to I^2S, PCM, or SoundWire.

Table 27 *Set Audio Data Port*

Command	Code	Value
Set Audio Data Port	0x8004(Set)	0x0000:PDM Data port 0x0001:I^2S/PCM Data port 0x0002:SoundWire Data port

10.7 Set Clock Source

The Set Clock source command allows the Host to select the clock source for the processor. The default clock source is REFCLK in SBL and is Internal Oscillator in BoskoApp. The Host can choose one clock source out of three possible clock sources. This command is also valid in BoskoApp.

The Host has to set the audio port clock frequency using the Set Audio Port Clock Frequency command 0x8042 and the Audio Data port using the Set Audio Data Port 0x8004 command before selecting audio port clock as a clock source.

It is recommended to use the Set Clock Source command as a 'no response' command. This is because the system will start running with the new clock source and the response may not appear properly.

Table 28 *Set Clock Source*

Command	Code	Value
Set Clock Source	0x8011(Set)	0x0000: Ref. Clock (768 kHz) 0x0001: Internal Oscillator (43.008 MHz) 0x0002: Audio Port clock

10.8 Set Digital Hardware Pass-through

The Set Digital hardware pass-through allows the host to loopback PCM channel at the pad level. This command is only valid for I^2S + I^2C and I^2S + UART. The default IA61x audio data port is PDM, so the host needs to change the audio port to PCM before sending the Digital Hardware Pass-through command. This command is also valid in BoskoApp.

Table 29 *Set Digital Hardware Pass-through*

Command	Code	Value
Set Digital Hardware pass-through	0x8052	0x0000: DHWPT disable 0x0001: DHWPT enable

10.9 Get Info

The Get Info command allows the host to get various system status flags. The Host sends command 0x8022<ParamId> to the IA61x and the response to the command contains the current status of ParamId passed in the command. This command is also valid in BoskoApp. If BoskoApp supports MIPS measurements, then this command will also return that information.

Table 30 *Get Info*

Command	Code	ParamId		Response Value
		High Byte Value	Low Byte Value	
Get Info	0x8022	0x00	0x01: PDM bypass mode	0x0000: Bypass mode disable
				0x0001: PDM Bypass mode1
				0x0002: PDM Bypass mode2
				0x0003: PDM Bypass mode3
				0x0004: Invalid
		0x00	0x02: Clock Source	0x0000: Ref. Clock (768 kHz)
				0x0001: Internal Oscillator (43.008 MHz)
				0x0002: Audio Port clock
		0x00	0x03: Get Audio Port Clock Frequency	0x0000: 512 kHz
				0x0001: 768 kHz
				0x0002: 1024 kHz
				0x0003: 1536 kHz
				0x0004: 2048 kHz
				0x0005: 3072 kHz
				0x0006: 6144 kHz
				0x0007: 12288kHz
				0x0008: Reserved (Default)
				0x0009: 9600kHz
				0x000A: 2400Khz
				0x000B: 4800Khz
				0x000C: 4608Khz
		0x00	0x04: Hardware Mode	0x0000: ASIC
				0x0001: FPGA
		0x00	0x0500: Control Interface detection OTP bits value. **Note:** if more than one interface is enabled in package then values are OR'ed.	0x0000: Auto detect control interface
				0x0001: I^2C interface
				0x0002: UART interface
				0x0004: SPI interface
				0x0008: SoundWire interface

Command	Code	ParamId		Response Value
		High Byte Value	Low Byte Value	
Get Info	0x8022	Endpoint ID (Example: 0x0C –RX0)	0x06: Endpoint frame count	Get Frame received count per endpoint. Used to check whether the endpoint is active or not. If the count is incrementing, data is flowing into the system and that endpoint is Active.
		0x00	0x07: Get Crash Info	0x0000: No crash 0x0001: Crash happened
		0x00	0x08: Reserved	Reserved
		0x00	0x09: Get ROM version	Invalid response: ROM version V1.0 0x0101: ROM version V1.1

10.10 Set SPI Sample Configuration

The IA61x SPI Hardware supports speeds up to Sampling clock /2.5, however there are some specific systems that require special settings for SPI to achieve higher data rates. To enable these settings, FW provides specific boot commands and APIs. SPI clock speed depends on the Boot command and API used to configure sample configuration.

The IA61x supports three modes, HW, FW - Early Launch, and FW-Normal Launch. **Error! Reference source not found.** provides recommended mode for the type of Host AP.

In **Error! Reference source not found.**, if the sampling clock is 43 MHz, a 5x ratio means 43/5 = 8.6 MHz and a 2.5x ratio means 43/2.5 = 17.2 MHz.

Table 31 *SPI Recommended Sample Config Modes*

AP Type		Speed Limit	Recommended Mode
Capture Point	CS to CLK delay		
Falling Edge	Less than 1 bit clock	Speed up to 5x	HW
		Speed between 5x and 2.5x	HW
Falling Edge	More than 1 bit clock	Speed up to 5x	HW or FW-Normal Launch
		Speed between 5x and 2.5x	FW - Early Launch
Mid of clock	Less than 1 bit clock	Speed up to 5x	HW or FW-Normal Launch
		Speed between 5x and 2.5x	HW or FW-Normal Launch
Mid of clock	More than 1 bit clock	Speed up to 5x	FW-Normal Launch
		Speed between 5x and 2.5x	FW-Normal Launch

Modes can be selected during boot up using the boot command and can be modified using the API at any point in time.

Table 32 *SPI Sample Config Mode Setup*

Mode	Sync Bytes	API Command
HW	0xB8B8B8B8	0x8023 0000
FW - EL	0xB7B7B7B7	0x8023 0003
FW - NL	0xB9B9B9B9	0x8023 0001

After sending an API command, the response should be read after at least 2 ms. The same is true for sending the next command when the response is suppressed (0x9023 0001/0).

Usage of API command is same in SBL and BoskoApp and settings will persist across sleep/wakeup cycle.

10.11 Set UART Stop Bits

The Set UART Stop Bits command allows the host to change the number of stop bits used in UART communication. The default configuration for the UART is to use 2 stop bits. Using this API, the number of stop bits can be switched between 1 and 2. This command is supported in both SBL and BoskoApp. Once the setting is done in SBL, it will remain the same after downloading the firmware.

Table 33 *Set UART Stop Bits*

Command	Code	Value
Set UART Stop Bits	0x8012	0x0000: 1 stop bit 0x0001: 2 stop bits(Default)

10.12 Power down Internal Oscillator

This command can be used to power down the INT_OSC to save power in SBL.

This command is not available in BoskoApp, as it will handle INT_OSC internally.

Table 34 *Power down Internal Oscillator*

Command	Code	Value
Power down INT_OSC	0x8043	0x0000 : INT_OSC powered down 0xFFFF : Cannot power down INT_OSC as boot was from INT_OSC

Note: Host should not send this command If CPU is running from either INT_OSC or AUDIO_PORT_CLK.

10.13 Enable/Disable JTAG

This command can be used to Enable/Disable JTAG in SBL only.

Table 35 *Enable/Disable JTAG*

Command	Code	Value
Enable/Disable JTAG	0x8061	0x0000 : Disable 0x0001 : Enable

10.14 Set PAD control Register

API command 0x8062, which can be used to change PAD control Register in SBL mode.

Table 36 *Set PAD control Register*

Command	Code	ParamID		
		Higher Byte		Lower Byte (Reg Value)
		Higher nibble (Pad Num)	Lower nibble (Reg Field)	
Set PAD control Register	0x8062	0x0: PAD_CMD_A1	0x0: DDST	0x00 – 0x03
		0x1: PAD_MCLK	0x1: RET	0x00 – 0x01
		0x2: PAD_MSD	0x2: SMT	0x00 – 0x01
		0x3: PAD_MSEL	0x3: SR	0x00 – 0x01
		0x4: PAD_P0	0x4: I/O PAD Drive Level Strength	0x00 – 0x03 0x01(Default)
		0x5: PAD_P1		
		0x6: PAD_P2		
		0x7: PAD_P3		
		0x8: PAD_P4		
		0x9: PAD_P5		
		0xA: PAD_SWR_CLK		
		0xB: PAD_SWR_DATA		

Limitations:

1. Register values are not preserved over sleep/wakeup sequence.
2. Register values may change after I2S firmware download.
3. SetPadmode() function has higher priority over this 0x8062 API command.

Chapter 11: System API Commands

API commands that enable the host to control and setup specific IA61x hardware features as described in **Error! Reference source not found.**.

Note: Refer section 11.2.1 for Bit clock frequency.

Table 37 *Summary of System API commands*

Cmd name	Cmd ID	Sub-Cmd(1)	Sub-Cmd(2)	Description
Sync	0x8000	n/a	n/a	
Get/Set Device Parameter	0x800C (Set Param ID) 0x800B (Get Param) 0x800D (Set param)	0x0A=PCM0	0x00	PCM Word Length
			0x01	Timeslots Per Frame
			0x02	PCM Tx Delay from Frame Sync
			0x03	PCM Rx Delay from Frame Sync
			0x04	PCM latch Edge
			0x05	PCM endian-ness
			0x06	PCM Tri-State enable
			0x07	PCM Audio Port Mode (PCM or I^2S)
			0x08	Reserved
			0x09	PCM Clock Control
			0x0A	Data Justification
			0x0B	Reserved
			0x0C	Frame Sync Polarity
		0x10=PDMIN0 (Internal mic, Left)	0x02	Bit Clock Frequency
		0x11= PDMIN1 (Internal mic, Right)	0x02	Bit Clock Frequency
		0x12= PDMIN2	0x02	Bit Clock Frequency
		0x13= PDMIN3	0x02	Bit Clock Frequency
		0x14= PDMOUT0	0x02	Bit Clock Frequency

Get/Set Device Parameter	0x800C (Set Param ID) 0x800B (Get Param) 0x800D (Set param)	0x15= PDMOUT1	0x02	Bit Clock Frequency
		0x16	0x00	I2SM0 Config mode
		0x17	0x00	I2SM1 Config mode
		0x18 = SoundWire PDMI0	0x02	Bit Clock Frequency
		0x19= SoundWire PDMI1	0x02	Bit Clock Frequency
		0x1A = SoundWire PDMI2	0x02	Bit Clock Frequency
		0x1B = SoundWire PDMI3	0x02	Bit Clock Frequency
		0x1C= SoundWire PDMO0	0x02	Bit Clock Frequency
		0x1D= SoundWire PDMO1	0x02	Bit Clock Frequency
Set Power State	0x8010	n/a	n/a	Set Sleep state
Write Data Block	0x802F	n/a	n/a	
GetEvent	0x806D	0000		Get Keyword ID
		0x0001 – 0xFFFF		Reserved
Set Diagnostic Config	0x801C	Section Opt	Section ID	Set Diagnostic Configuration (See Section 11.10)

11.1 Sync

The Sync command is used by the Host to ensure that the IA61x is ready to receive commands. The Host should wait for a Sync Ack before sending any additional commands.

A 10 mSec timeout timer should be started by the Host when the Sync command is sent to the IA61x. If this timer expires before a Sync Ack is received by the Host from the IA61x, another Sync command should be sent and the timer should be started again. If the timer expires three consecutive times, the Host should assert the Master power cycle if possible; otherwise, the IA61x should be declared unusable.

Table 38 *Sync Command Control_Mode Values*

Command	Code	Control_Mode Value
Sync	0x8000	0x0000 = Polling

The Polling control mode is used to send the reply back to the Host. In this mode, the Host periodically reads four bytes from the IA61x (the typical polling period should be 20 ms) until non-zero data is received. The IA61x will send zeros as long as the reply is not ready.

The IA61x supports commands over I^2C, UART, SPI, and SoundWire. The Host will receive the response in the same interface where it sent the command.

11.2 Get/Set Device Parameter and Parameter ID

This command allows the host to read or write a device parameter specified by Parameter ID and specify the value to be written into that parameter in the case of Set. Parameter ID consists of a 16-bit value that is divided into a high byte and a low byte. The high byte specifies the device of interest, while the low byte specifies the particular device parameter to be read or written.

Sequence for setting a parameter is:

- SetDeviceParmID
- SetDeviceParam

Table 39 *Get/Set Device Parameter Command Codes*

Command	Code
GetDeviceParm	0x800B
SetDeviceParmID	0x800C
SetDeviceParm	0x800D

11.2.1 Bit Clock Frequencies

Param values for bit clock frequencies as follows.

Table 40 *Bit clock frequency*

	Value
Bit Clock Frequency	0x0000 – 512k 0x0001 – 768K 0x0002 – 1024K 0x0003 – 1536K 0x0004 – 3072K 0x0005 – 6144K 0x0006 - 4608K 0x0007 – 2048K

11.2.2 Port A Device Parameters

Table 41 *Get/Set Port A Device Parameter ID Values*

Device	Device Parameter	Parameter ID		Value	Remarks
		High Byte Value	Low Byte Value		
PORTA	Word Length	0x0A	0x00	16 bits = 0x000F (Default) 20 bits = 0x0013 24 bits = 0x0017 32 bits = 0x001F	Length of data word for Tx & Rx [bits]
	TDM time slots Per Frame		0x01	0x000X: 0…8 (Default=1)	Total number of timeslots per Frame, minus 1 (0=1 slot, 1=2 slots, etc.). This field is only valid in Master mode.
	Del From Fs Tx		0x02	Min = 0x0000 (Default) … Max = 0x001F	Number of cycles after frame sync to start Tx
	Del From Fs Rx		0x03	Min = 0x0000 … Max = 0x001F	Number of cycles after frame sync to start Rx Default Value: 0x0000, if Audio Port Mode = PCM 0x0001, if Audio Port Mode = I^2S
	Latch Edge		0x04	Tx on Rising Edge, Rx on	Default Value:

Device	Device Parameter	Parameter ID		Value	Remarks
		High Byte Value	Low Byte Value		
				Rising Edge = 0x0000 Tx on Falling Edge, Rx on Rising Edge = 0x0001 Tx on Rising Edge, Rx on Falling Edge = 0x0002 Tx on Falling Edge, Rx on Falling Edge = 0x0003	0x0002, if Audio Port Mode = PCM 0x0001, if Audio Port Mode = I^2S
	Endianness		0x05	Little Endian(LSB first) = 0x0000 Big Endian(MSB first) = 0x0001 (Default)	Endianness
	Tristate Enable		0x06	Disable = 0x0000 Enable = 0x0001 (Default)	When enabled, PORTA_DO is tri-stated when idle
PORTA	Audio Port Mode	0x0A	0x07	PCM = 0x0000 I^2S = 0x0001 (Default)	Configure the Audio Port to either I^2S or PCM. Setting the port audio mode adjusts the following registers: - Word length (it is set to 16 bits) - Latch edge polarity (I^2S => Rx+/Tx-, PCM => Rx-/Tx+) - Frame sync duration (I^2S => 16 clocks, PCM => 1 clock) - Rx delay (I^2S => 1 clock, PCM => 0 clocks) - TDM (I^2S => disabled, PCM => enabled)
	TDM Enabled		0x08	Reserved	This is reserved for future use.
	Clock Control		0x09	Slave = 0x0000 (Default) Master = 0x0001	Output Frequency: For Output frequency, please see I^2S Master device configuration.
	Data Justification		0x0A	0x00 for right justified data 0x01 for left justified data (default)	When the transmitted data is narrow than the frame it is transmitted in (for example, 24-bit data in a 32-bit frame), this parameter selects how the data is aligned.

Device	Device Parameter	Parameter ID		Value	Remarks
		High Byte Value	Low Byte Value		
	Frame sync duration		0x0B	Reserved	This is reserved for future use.
	Frame sync polarity		0x0C	High : 0 (default) Low : 1	Frame sync polarity

11.2.3 I²S Master 0 (I2SM0) Device Parameters

(This device is dedicated to provide MCLK to the IA61x for PDM0 Left/Right. (Internal mic)

Device	Device Parameter	Parameter ID		Value
		High Byte Value	Low Byte Value	
I2SM0	Config Mode	0x16	0x00	I2SM_MODE00_08K_16B_1CH = 0 I2SM_MODE01_08K_16B_2CH, = 1 I2SM_MODE02_08K_16B_4CH, = 2 I2SM_MODE03_08K_24B_1CH, = 3 I2SM_MODE04_08K_24B_2CH, = 4 I2SM_MODE05_08K_24B_4CH, = 5 I2SM_MODE06_08K_32B_1CH, = 6 I2SM_MODE07_08K_32B_2CH, = 7 I2SM_MODE08_08K_32B_4CH, = 8 I2SM_MODE09_16K_16B_1CH, = 9 I2SM_MODE10_16K_16B_2CH, = 10 I2SM_MODE11_16K_16B_4CH, = 11 I2SM_MODE12_16K_24B_1CH, = 12 I2SM_MODE13_16K_24B_2CH, = 13 (Default) I2SM_MODE14_16K_24B_4CH, = 14 I2SM_MODE15_16K_32B_1CH, = 15 I2SM_MODE16_16K_32B_2CH, = 16 I2SM_MODE17_16K_32B_4CH, = 17 I2SM_MODE18_48K_16B_1CH, = 18 I2SM_MODE19_48K_16B_2CH, = 19 I2SM_MODE20_48K_16B_4CH, = 20 I2SM_MODE21_48K_24B_1CH, = 21 I2SM_MODE22_48K_24B_2CH, = 22 I2SM_MODE23_48K_24B_4CH, = 23 I2SM_MODE24_48K_32B_1CH, = 24 I2SM_MODE25_48K_32B_2CH, = 25 I2SM_MODE26_48K_32B_4CH, = 26 I2SM_MODE27_96K_16B_1CH, = 27 I2SM_MODE28_96K_16B_2CH, = 28 I2SM_MODE29_96K_16B_4CH, = 29 I2SM_MODE30_96K_24B_1CH, = 30 I2SM_MODE31_96K_24B_2CH, = 31 I2SM_MODE32_96K_24B_4CH, = 32 I2SM_MODE33_96K_32B_1CH, = 33 I2SM_MODE34_96K_32B_2CH, = 34 I2SM_MODE35_192K_16B_1CH, = 35 I2SM_MODE36_192K_16B_2CH, = 36 I2SM_MODE37_192K_24B_1CH, = 37 I2SM_MODE38_192K_24B_2CH, = 38 I2SM_MODE39_192K_32B_1CH, = 39 Where Field after Mode is Sample rate and word length

11.2.4 I²S Master 1 (I2SM1) Device Parameters

This device is default dedicated for PCM/I²S master mode. It can be used to generate a clock for PDM1 Left/Right PDMIN2/PDMIN3 or for PDMO Left/Right. (PDMO0/PDMO1).

Device	Device Parameter	Parameter ID		Value
		High Byte Value	Low Byte Value	
I2SM1	Config Mode	0x17	0x00	I2SM_MODE00_08K_16B_1CH = 0 I2SM_MODE01_08K_16B_2CH, = 1 I2SM_MODE02_08K_16B_4CH, = 2 I2SM_MODE03_08K_24B_1CH, = 3 I2SM_MODE04_08K_24B_2CH, = 4 I2SM_MODE05_08K_24B_4CH, = 5 I2SM_MODE06_08K_32B_1CH, = 6 I2SM_MODE07_08K_32B_2CH, = 7 I2SM_MODE08_08K_32B_4CH, = 8 I2SM_MODE09_16K_16B_1CH, = 9 I2SM_MODE10_16K_16B_2CH, = 10 I2SM_MODE11_16K_16B_4CH, = 11 I2SM_MODE12_16K_24B_1CH, = 12 I2SM_MODE13_16K_24B_2CH, = 13 (Default) I2SM_MODE14_16K_24B_4CH, = 14 I2SM_MODE15_16K_32B_1CH, = 15 I2SM_MODE16_16K_32B_2CH, = 16 I2SM_MODE17_16K_32B_4CH, = 17 I2SM_MODE18_48K_16B_1CH, = 18 I2SM_MODE19_48K_16B_2CH, = 19 I2SM_MODE20_48K_16B_4CH, = 20 I2SM_MODE21_48K_24B_1CH, = 21 I2SM_MODE22_48K_24B_2CH, = 22 I2SM_MODE23_48K_24B_4CH, = 23 I2SM_MODE24_48K_32B_1CH, = 24 I2SM_MODE25_48K_32B_2CH, = 25 I2SM_MODE26_48K_32B_4CH, = 26 I2SM_MODE27_96K_16B_1CH, = 27 I2SM_MODE28_96K_16B_2CH, = 28 I2SM_MODE29_96K_16B_4CH, = 29 I2SM_MODE30_96K_24B_1CH, = 30 I2SM_MODE31_96K_24B_2CH, = 31 I2SM_MODE32_96K_24B_4CH, = 32 I2SM_MODE33_96K_32B_1CH, = 33 I2SM_MODE34_96K_32B_2CH, = 34 I2SM_MODE35_192K_16B_1CH, = 35 I2SM_MODE36_192K_16B_2CH, = 36 I2SM_MODE37_192K_24B_1CH, = 37 I2SM_MODE38_192K_24B_2CH, = 38 I2SM_MODE39_192K_32B_1CH, = 39 Where Field after Mode is sample rate and word length

To calculate the clock using Config mode, the value is shown here:

Example: Mode = I2SM_MODE01_08K_16B_2CH

So the clock generated would be 8*16*2 = 256 KHz.

For PCM and I²S the mode clock and FS would be same.

Note: If the Audio Port Mode is configured in I²S mode (See Section **Error! Reference source not found.**), then Config Mode can only be *_2CH. The I²S Protocol does not allow anything other than two channels. You can use the entire Config mode for PCM.

11.2.5 PDM Device Parameters

Table 42 *Get/Set PDM Device Parameter ID Values*

Device	Device Parameter	Parameter ID		Value	Remarks
		High Byte Value	Low Byte Value		
PDM	PortClockFrequency	0x10	0x02	0.512 MHz: 0x00 0.768 MHz: 0x01 (Default) 1.024 MHz: 0x02 1.536 MHz: 0x03 3.072 MHz: 0x04 N/A: 0x05 4.608 Mhz: 0x06 2.048 Mhz: 0x07 2.400 Mhz: 0x08 4.800 Mhz: 0x09	PDM port frequency

Note: The default value for this device parameter is defined in the default system configuration, which can be overridden by the product specific Config.

Note: PDMI0 clock rate must be the same as the external PDMOx port clock rate.

Maximum supported PDM clock for Internal Mic Internal Mic is 4.8MHz, so 6.144MHz PDM clock is not supported.

These commands are valid for the following PDM ports as well. Note that you will have to change the port IDs ("High Byte Value" in "Parameter ID") of the commands in order to address the required PDM port.

Table 43 *Default values for PDM parameters*

Device	High Byte Value	Device Parameter	Default Value
Internal Mic	0x10	PortClockFrequency (0x02)	0x01: 0.768 MHz
PDMI2	0x12	PortClockFrequency (0x02)	0x01: 0.768 MHz
PDMI3	0x13	PortClockFrequency (0x02)	0x01: 0.768 MHz
PDMOUT0	0x14	PortClockFrequency (0x02)	0x01: 0.768 MHz

PDMOUT1	0x15	PortClockFrequency (0x02)	0x01: 0.768 MHz

Internal Mic clock is given to IA61x internal mic, default value is 768 KHz. In general, it is same as port clock, however, audio port clock (refer section 10.5) and Internal Mic clocks can differ too. For example,

PDM Audio port clock set to 1536 KHz and Internal Mic clock set to 3072 KHz is valid configuration.

PCM audio port clock set to 6144 KHz (I2S 192Khz) and Internal Mic clock set to 4608 KHz is valid configuration.

11.2.6 SoundWire PDM Port Configuration

Table 44 *Get/Set PDM Device Parameter ID Values*

Device	Device Parameter	Parameter ID		Value	Remarks
		High Byte Value	Low Byte Value		
SoundWire PDMI0	PortClockFrequency	0x18	0x02	0.512 MHz: 0x00 0.768 MHz: 0x01 1.024 MHz: 0x02 1.536 MHz: 0x03 3.072 MHz: 0x04 6.144 MHz: 0x05 4.608 Mhz: 0x06 2.048 Mhz: 0x07	SoundWire PDM port frequency **Note:** If SoundWire bus Frequency is set at 9.6 MHz then 3.072 MHz and 6.144 MHz is not supported.

Table 45 *Default Values for SoundWire PDM Ports*

Device	High Byte Value	Device Parameter	Default Value
SoundWire PDMI0	0x18	PortClockFrequency (0x02)	0x01: 0.768 MHz
SoundWire PDMI1	0x19	PortClockFrequency (0x02)	0x01: 0.768 MHz
SoundWire PDMI2	0x1A	PortClockFrequency (0x02)	0x01: 0.768 MHz
SoundWire PDMI3	0x1B	PortClockFrequency (0x02)	0x01: 0.768 MHz
SoundWire PDMO0	0x1C	PortClockFrequency (0x02)	0x01: 0.768 MHz
SoundWire PDMO1	0x1D	PortClockFrequency (0x02)	0x01: 0.768 MHz

11.3 Set Power State

The SetPowerState command allows the host to set the IA61x into one of the following low power modes:

5. Deep Sleep mode (Power down mode when the IA61X is not in use).
6. Sleep mode (Stage 0: Power down mode with internal power regulators).
7. Low Power mode (Stage 1: Reduce power to only what is needed to run Voice Wake)

Table 46 *Set Power State Command Codes*

Command	Code	Value
SetPowerState	0x8010	0x0000 = DEEP SLEEP 0x0001 = SLEEP (Stage 0) 0x0002 = Low Power (Stage 1) 0x0003 = Binary Retention Sleep

Note 1: All System and Feature API settings will revert back to their default states after Sleep Mode is initiated. Upon wakeup from Sleep Mode, all custom settings previously set may need to be reapplied.

Note 2: The SetPowerState command should be sent with "No Response." e.g. 0x9010_0001

Note 3: The IA61x normally transitions to Low Power mode after AAD when in Sleep. However when using this command with parameter 2, the IA61x can go to Low Power mode directly.

Note 4: Setting the IA61x to Low Power mode will only work if a 1Ch Voice Wake Buffered route (Route 6) is set.

Note 5: Deep sleep mode should be used only when IA61x operation is not required. It is the lowest power mode of the IA61x.

Note 6: Deep Sleep and Low power mode is supported only after FW download.

Note 7: Binary Retention Sleep is applicable after FW download and puts the IA61x in low power state similar to SLEEP but power consumption is lesser than SLEEP. No audio feature such as keyword detection, PDM bypass etc will function during this sleep. Host wakeup method is same as other sleeps.

To restore the IA61x into normal operating mode, refer to **Error! Reference source not found.** **Error! Reference source not found.**.

11.4 Write Data Block (WDB)

The Write Data Block API provides the Voice Wake Keyword detection process with keyword data.

The Host transfers the Keyword Data in one continuous stream after sending the WriteDataBlock API command, with the command parameter specifying the data block length and the data as the block data.

Table 47 *Write Data Block*

Command	Code	Value
WriteDataBlock	0x802F	16-bit value for the data block length (excluding headers appended by Host and zero padding)

The model file has a 4-byte header organized as follows:

Bits 31 – 24	Bits 23 – 8	Bits 7 – 0
Algo ID	Algo specific	Chunk ID

Figure 28 *Model file header*

- Algo ID: Specifies the algorithm ID.
- Algo specific: Information specific to algorithm, used to identify the data block.
- Chunk ID: Count for the nth 512-byte block of the keyword model. This count must end at 0xFF, as the WDB sequence will be aborted by having 0xFE written into this field

The Write Data Block command utilizes a communication protocol that is different from other API commands. After the command is issued, the model file must be sent in one continuous stream with a header (as described in **Error! Reference source not found.**) at every 512 bytes (4 bytes of header + 508 bytes of data).

The number of bytes sent through WDB has to be a multiple of 512. If the keyword model size (plus headers) does not meet this requirement, then the Host must pad with zeros.

Note: 512 bytes is the default block size, but this size is actually configurable in SysConfig.

Note: The model files themselves will only contain the header at the beginning, and the remaining headers must be appended by the host.

After all WDB bytes have been sent, the IA61x will respond to the Write Data Block command with 0x802F 0x0000. If this response isn't received, then WDB failed.

The host must provide one command response delay (TBD) between the WDB command and the WDB data and also between the end of WDB data and before reading the response. If proper delays are not provided, then the WDB will fail.

The sequence diagram for this protocol is shown in **Error! Reference source not found.**: **Error! Reference source not found.**

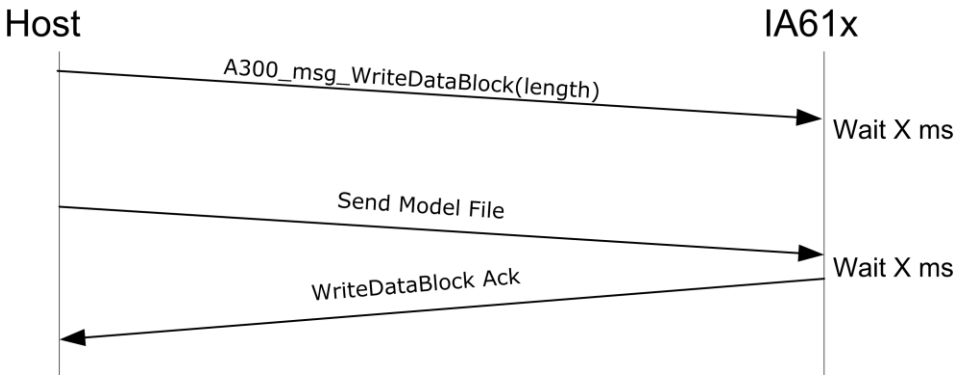

Figure 29 *Write Data Block Sequence Diagram*

Note: Model file header in the first chunk will vary for Knowles voice wake algorithm and also there should be strict check for OEM models due to this host should send the 12-bytes header in the first chunk for OEM models. For more details, please refer Section: 14.3.3.

11.4.1 Write Data Block Guidelines

The Host software needs to consider the following guidelines for WDB:

1. Do not change header byte 1, which is reserved for firmware internal use.
2. Do not change header bytes 2 and 3, which are algorithm specific.
3. Update the sequence number in header byte 4 as per sequence number. The last block must have a sequence number of 0xFF. Use 0xFE to abort the transfer.
4. The abort chunk must be chunk-size aligned. For example, the abort header should follow (Chunk size – header size) number of zeros or data before the transfer is aborted.
5. The abort sequence will have a success response.
6. WDB returns the error codes shown in **Error! Reference source not found.** (0 indicates success; non-zero indicates error). These codes apply to both Algo and Voice Wake WDB functions.

Table 48 *Write Data Block Error Codes*

Return Value	Constant	Description
0	PB_SUCCESS	Successful.
1	PB_INVALID_ALGO_ID	Header algo ID invalid.
2	PB_INVALID_BLOCK_TYPE	Header type invalid, must be: • PB_TYPE_VS_USERKEYWORD = 6 • PB_TYPE_VS_OEMKWORD = 7 • PB_TYPE_VS_BKGDATA = 8
3	PB_INVALID_SIZE	Size of the model file is invalid.
4 - 9	Reserved	Reserved for future generic WDB error codes.
>= 10	Algorithm specific	Algorithm specific error codes.

11.5 Read Data Block (RDB)

11.5.1 Diagnostic Log Read Data Block

The Read Data Block ("RDB") command allows the Host to read variable length blocks of results from the target. The data block read is determined by the Block ID, where the upper 8 bits represent the algo ID and the lower 8 bits represent the block type.

Command	Code	Value(Block ID)
Read Data Block	0x802E	0xABCD Where, 0xAB is the algo ID 0xCD is the block type

Algo ID '0x00' is used for firmware and other IDs are used for algorithms (Knowles and 3rd party).

The IA61x returns the status of the command. If it returns No RDB data found, then the command failed due to a wrong RDB ID. If the block ID is fine, then it will ask the location to save the output `.bin` file in AuViD.

Note that the RDB command only reads four byte multiples, so the size must be rounded up to the next integral multiple of four bytes.

11.6 GetEvent

The Host uses this API to get event status.

Table 49 *GetEvent Request*

Command	Code	Value
GetEvent	0x806D	0x0000 => Get Event ID Others => Reserved

When this API is received, the Event status is returned, and the Event status is subsequently cleared, which includes clearing the status of HOST_IRQ pin-out signal (aka, HOST_IRQ signal).

The Event value will be re-set if and when the Event re-occurs.

Table 50 *GetEvent Response for Get Keyword ID*

Command	Code	Event type	Event ID
GetEvent	0x806D	0x00 => Keyword	0x00 No keyword detected Keyword ID => If keyword detected
		0x01 => Command	0x00 No keyword detected Command ID => If keyword detected 0xff => Command Timeout
		0x02 => Noise Level decrease	0xXX: These are don't care bits
		0x04 => Noise Level increase	0xXX: These are don't care bits
		0xFFFF => Exception has occurred	

When this API responds with 0x806D 0xFFFF, it means the firmware in the IA61x has crashed. In this scenario, the HOST_IRQ signal will not be cleared as a result of this command, but will instead be cleared after firmware is re-downloaded.

11.7 Get Overflow Status

When running CVQ, it is useful for the Host to know whether the CVQ buffer has overflowed or not. When audio data from a burst is expected, but missing, the Host can use this command to find out if that missing data is due to a CVQ buffer overflow (usually happens when the Host takes too long to start bursting), or a bug in the bursting process.

Table 51 *GetOverflowStatus*

Command	Code	Value
GetOverflowStatus	0x806E	0x0000 => Overflow did not occur 0x0001 => CVQ buffer overflowed

Note that when Keyword Preservation is enabled (see Set Buffering State), this command will return an overflow status if the keyword is overwritten up to the point of a keyword detection. However, if Keyword Preservation is disabled, then an overflow status will be returned if the write pointer overwrites the point of a keyword end.

11.8 Set Event Response

The Host uses this API to set the type of interrupt generated on the HOST_IRQ line after an event (eg. Keyword Detection, Command Detection, or Exception). This command can be sent at any time after firmware download.

Table 52 *SetEventResponse*

Command	Code	Value	
		Higher byte	Lower byte
SetEventResponse	0x801A	Refer Figure27	0x00 => Disable 0x01 => Low Level 0x02 => High Level 0x03 => Falling Edge 0x04 => Rising Edge 0x05 => SOUNDWIRE Trigger

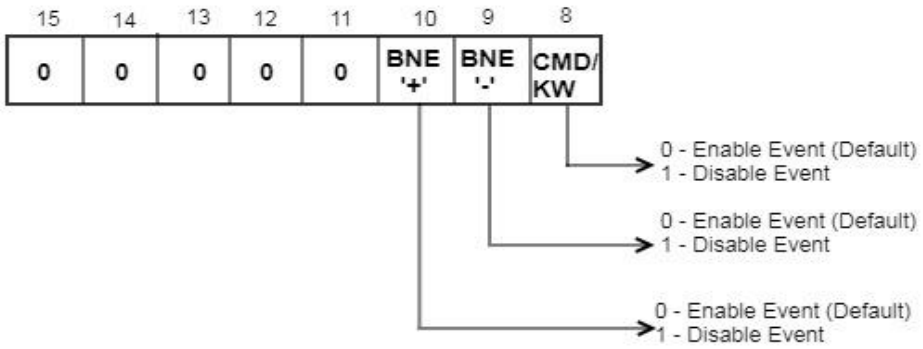

Figure 30 *Configure Event mask*

Low Level –HOST_IRQ will be driven low after an event. On a Get Event command, HOST_IRQ will either be let go if the pin is not dedicated (see **Error! Reference source not found.**), or be driven high if it is. The host must ensure that HOST_IRQ is pulled high by default in order for this setting to work.

High Level –HOST_IRQ will be driven high after an event. On a Get Event command, HOST_IRQ will either be let go if the pin is not dedicated, or be driven

low if it is. The host must ensure that HOST_IRQ is pulled low by default in order for this setting to work.

Falling Edge –A falling edge will always be generated after an event, regardless of the default level of HOST_IRQ. This is done driving HOST_IRQ high for some time (configurable via Sys Config, default is 100 cycles), then driving the line low. Sending a Get Event command will make firmware let go of HOST_IRQ.

Rising Edge – A rising edge will always be generated after an event, regardless of the default level of HOST_IRQ. This is done by driving HOST_IRQ low for some time, then driving the line low. Sending a Get Event command will make firmware let go of HOST_IRQ.

SoundWire Trigger: When Host configures SoundWire as the Trigger, firmware will generate a 'SoundWire Implementation Defined Interrupt' on keyword detection. SoundWire will also generate a wake-up signal on the SoundWire bus in case of clockStopModes when it is enabled. SoundWire interrupt and wakeup will be visible to Host only if it is unmasked in SCP_IntMask_1 and SCP_SystemCtrl registers respectively. Note that SetEventResponse parameter value '0' (Disabled) is not applicable for SoundWire. Refer to IRQ and WakeUp with SoundWire for more details.

This command will not have an effect when the event interrupt is generated due to a crash. In that case, a pulse is generated which contains low level, rising edge, high level, and falling edge interrupts. The diagram below illustrates this. Note that if the HOST_IRQ line is being pulled high, there will be two rising edges 1290us apart, and two falling edges 1000us apart. In such a scenario, the Host must be able to handle the second "false" interrupt.

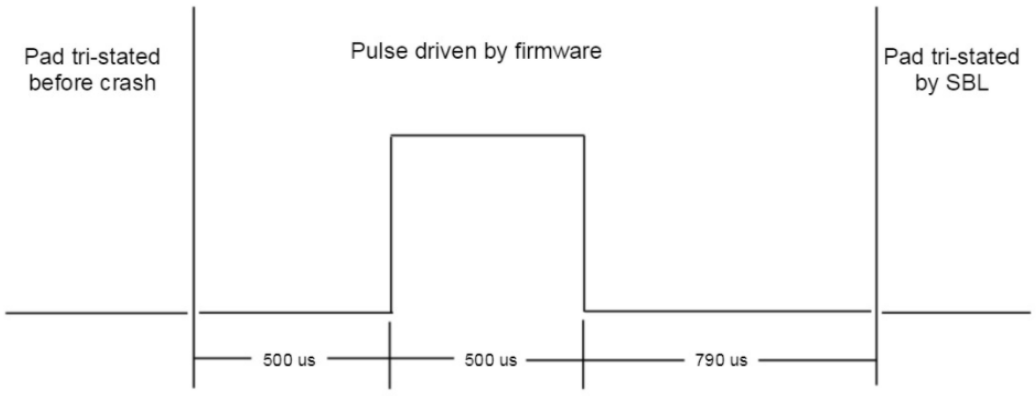

Figure 31 *Generic pulse generated on crash event*

11.9 Streaming Data

11.9.1 Set Streaming

This command instructs the IA61x to send Streaming Data over the control interface. Inputs and outputs can be sent over I^2C, UART, SoundWire, or SPI interfaces. Streaming is supported for 8 K and 16 K sample rate routes only.

Table 53 *SetStreaming Command Code Values*

Command	Code	Value
SetStreaming	0x8025	0x0000 => Disable streaming 0x0001 => Enable Q15 data Streaming 0x0002 => Enable AFLOAT data Streaming

Note that Streaming requires an active route. You can only select Streaming for valid Endpoints. Streaming is enabled over the active control interface. The active control interface is the one that is used for loading firmware. It can be I^2C, SPI, UART, or SoundWire.

Note: Streaming at smaller audio frame-sizes (0.5ms/1ms/2ms) is NOT recommended due to not enough MIPS. Also available streaming MIPS varies based on algorithm MIPS requirement, which is different for each algorithm.

11.9.2 Stop Streaming Special Handling

SPI Interface

The Stop Streaming command over SPI must be sent with the response enabled: (0x8025 0x0000). Here are the steps and details.

- Send a Stop command 0x80250000.
- Send 60 bytes (15 words) of zeros.

(The Above two steps can be combined in one and host can send total of 64 bytes; the first four as '0x80250000' and next sixty as '0x00000000')

- Wait for at least 2 mS.
- Read next 50 words by sending '0x00000000' with read interval of 2ms.

Important Note: A Response of 0x80250000 could be anywhere in these steps (after sending 0x80250000) and once a response is detected there is no need of sending extra zeros. However, there is no harm in sending extra zeros.

As an alternative, after sending 0x80250000, the Host can keep reading responses until it gets the response 0x80250000 followed by few zeros to make sure it is not part of the streaming data.

SoundWire Interface

Stop Streaming needs special handling when the SoundWire Control Port is used for IO Streaming. After sending the 'Stop Streaming' command (0x80250000), there has to be two extra 4-byte reads before reading the response for 'Stop Streaming.' The first two reads will contain streaming data and the third read will be the response to "Stop Streaming." The sequence is:

- Send 0x80250000 to 0x2000-0x2003
- Read from 0x2004-0x2007: streaming data
- Read from 0x2004-0x2007: streaming data
- Read from 0x2004-0x2007: 0x80250000

When using a command with response disabled, the third read should be omitted.

11.9.3 Select Streaming

The Select Streaming command is used to enable or disable streaming of audio samples for a particular Endpoint. Sending stream data from specific audio paths is either enabled or disabled. Note that streaming is only valid for Active routes, and only for active Endpoints in the active routes.

Endpoint ID: See Section **Error! Reference source not found.**, **Error! Reference source not found.**.

Table 54 *SelectStreaming Command Code Values*

Command	Code	Value	
		Bit 15	Bits 14:0
SelectStreaming	0x8028	0 => Disabled 1 => Enabled	Endpoint parameter

By default, streaming for all Endpoints is disabled, and you need to enable the required Endpoints for streaming.

Note: Due to memory limitation, maximum two endpoint(s) can be streamed at a time. If more than two are selected, unexpected behavior will be observed.

For guidance on Stream data packet format, see: Section **Error! Reference source not found.**, **Error! Reference source not found.**.

11.9.4 Streaming Data Format

Streaming data is transferred as data packets over the designated interface. Each packet has an 8-byte header, followed by an n-byte payload.

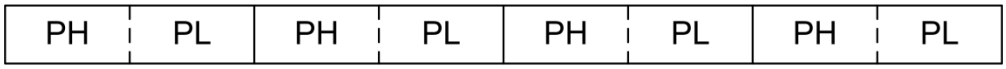

Figure 32 *Streaming Data Format*

Where PH = packet header and PL = Packet payload. These are always in pairs.

The Payload header is always 8-bytes as shown in **Error! Reference source not found.**

Byte 7	Byte 6	Byte 5	Byte 4		Byte 3	Byte 2	Byte 1	Byte 0
Marker 0x12	0x34	ID: p_1p_2	Y	n	Length 0xAo	0x00	0xFF	0x00

Figure 33 *Streaming Data – Packet Header*

Note that the Marker field (two bytes) has the numeric sequence as shown.

If multiple channels (e.g., endpoint 1, endpoint 2) are requested, their respective packets are interleaved.

The packet "ID" field is two bytes, and has two subfields in "p_1p_2 n" format:

(1) Where "p_1p_2" is a 8-bit field containing an Endpoint Parameter Definition (See Section **Error! Reference source not found.**), and

(2) Where "n" is a 4-bit field containing a packet counter field (right justified).

Example:

If the fields are:

```
p1p2  = 0x08
n     = 0x2
```

Then:

```
P (Endpoint)        = 0x08
N (packet counter)          = 0x02
```

(3) Where "Y" is a 4-bit field containing a Sample rate Index. The various values are as shown in **Error! Reference source not found.**.

Table 55 *Sample Rate Index Values*

Description	Index value
Sample Rate -8K	0
Sample Rate -16K	1
Sample Rate -24K	2
Sample Rate -32K	3
Sample Rate -48K	4
Sample Rate -96K	5
Sample Rate -192K	6

(4) The Length field is two bytes in little-endian ("LE") format, so 0xA000 is 160_{10} bytes.

The Payload length value depends on these two parameters of any use case,

1. Audio sample rate
2. Audio frame size

This is generic equation for payload length:

Payload length = (sample rate Hz/1000) * (frame size in mSec) * (bytes per sample)

For 16 kHz sample rate and 16 mSec frame size and 16-bit sample format:

Payload length = (16000/1000) * 16 * 2 = 512 bytes

For 16 kHz sample rate and 16 mSec frame size and AFLOAT sample format:

Payload length = (16000/1000) * 16 * 4 = 1024 bytes

(5) The last two bytes indicate the data format of the Payload.

1. 0x01 indicates Payload data is in 16-bit signed data, in little endian format.
2. 0x02 indicates Payload data is in AFLOAT, in little endian format.

Note: For Streaming packets over SPI interface, the data is transferred in 4-byte words. The payload words need to be swapped (i.e., endianness reversal).

11.9.5 Streaming Example

Error! Reference source not found. shows an example of the usage of streaming for a simple 1-Ch pass-through route 0.

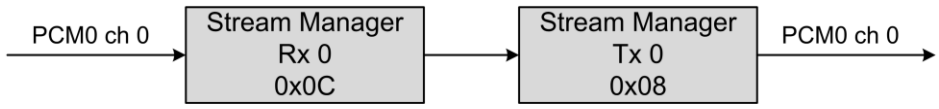

Figure 34 *Streaming Example*

In this example, the command sequence to enable Streaming is.

```
0x8028 0x800C;    //  Select  stream  manager  rx  endpoint  0  for
streaming
0x8028 0x8008;    //  Select  stream  manager  tx  endpoint  0  for
streaming

0x8025 0x0001;    // Enable streaming
0x8025 0x0000;    // Disable streaming
```

11.9.6 Continuous Voice Wake buffer bursting

Continuous Voice Wake buffer upload happens through bursting, which is based on same data/packet format as streaming except that Bursting only supports Q15 format of data, but has a separate API command. The Continuous Voice Wake buffer bursting can be done through the Control interface or Data interface.

Table 56 *Bursting Command Code Values*

Command	Code	Value
Start/Stop Bursting	0x8026	0x0000 => stop bursting 0x0001 => start bursting 0x0011 => start bursting over I^2S 0x0010 => stop bursting over I^2S

Stopping Burst mode is similar to stopping streaming. Please refer to Section **Error! Reference source not found.**, **Error! Reference source not found.** and replace command 8025 with 8026.

Note that this command will return an error if buffering is not enabled (see Section **Error! Reference source not found.**, **Error! Reference source not found.**, 0x802A)

11.9.7 Set Buffering State

By default, the IA61x will buffer up to three seconds of audio, and the burst will contain the keyword that woke up the IA61x as well as any audio that came after it (as long as bursting starts before the three second circular buffer is overwritten). If different buffering options are

wanted, the Host can send a Set Buffering State command with the options described in the following table.

Table 57 *Buffering State Command Code Values*

Command	Code	Value
Set Buffering State	0x802A	0x0000 => Disable buffering 0x0001 => Disable keyword preservation 0x0002 => Enable full buffering (default)

With buffering disabled, only the minimal amount of buffering needed to get keyword detection is done. Any audio coming in after that will be ignored, saving power and MIPS. A burst command will fail if it is attempted during this state.

If keyword preservation is disabled, a burst will contain any audio that came right after keyword detection occurred.

11.9.8 Continuous Voice Wake buffer bursting over I²S

Continuous Voice Wake buffer upload over I²S happens through bursting, which is based on same data/packet format as streaming. Continuous Voice Wake buffer upload to host can be done over I²S in mono or stereo mode with 16/24/32 bit word length configurations. When bursting over I²S, whether you send 1 or 2 channels depends on the one-time configuration command 0x802C. See Section **Error! Reference source not found.** for configuring the I²S port.

Command sequence for enabling the I²S bursting:

```
0x802C 0x4200  // 4 indicates sample rate and 2 indicates
               // the number of channels used for data transfer
0x800C 0x0A00  // Select Device Param PCM port settings
0x800D 0x001F  // word length 16/24/32 bit (default 16bit)
0x8026 0x0011  // Enable bursting over I2S
0x8026 0x0010  // Disable Bursting over I2S
```

11.10 Set SoundWire Stream Mode

This command is used to set the mode of bulk data transfer to either the Control Port or the Bulk Register Access (BRA) Port. As firmware retains the configuration across sleep wakeup cycles, the Host need not send this API again unless the mode of transfer needs to change. This command is applicable for WDB, RDB, Bursting and Streaming. The default setting of this mode is based on the System Config parameter SWIRE_DFLT_STREAM_MODE.

Table 58 *Buffering State Command Code Values*

Command	Code	Value
Set SoundWire Stream Mode	0x8029	0x0000: Control Port 0x0001: BRA Port

11.11 Set Diagnostic Configuration

The Set Diagnostic Config command (0x801C) can be used in BoskoApp to configure the Diagnostic Logging. Using this command you can start, stop, and clear the diagnostic logging memory operations by either section-wise or all sections at a time, except for the Exception register section since the logging only happens after a firmware crash. When you clear the log, all of the data that was logged previously (except the header information for the log) is cleared. When you start logging again, you re-start logging from the beginning.

Table 59 *Set Diagnostic Config*

Command	Code	Parameter ID	
		High Byte Value	Low Byte Value
Set Diagnostic Config	0x801C	0x00	0x01 – Start Debug counters log 0x02 – Start Command history logs 0x03 – Start Route flow log 0x04 – Start BAF log 0x05 – Start Power states Log 0xFF – Start All sections log 0x0E – Start Trace
		0x01	0x01 – Stop Debug counters log 0x02 – Stop Command history logs 0x03 – Stop Route flow log 0x04 – Stop BAF log 0x05 – Stop Power states Log 0xFF – Stop All sections log 0x0E – Set PC stop start range address
		0x02	0x01 – Clear Debug counters log 0x02 – Clear Command history logs 0x03 – Clear Route flow log 0x04 – Clear BAF log 0x05 – Clear Power states Log 0xFF – Clear All sections log 0x0E – Set PC stop end range address
		0xYY	0x1E – Set Lower byte of post size 0x2E – Set Higher byte of post size

11.12 Algorithm Configuration

The Algorithm Configuration set of API commands control the algorithm functions within the IA61x software. The host can control specific features of the IA61x by sending a command and its associated parameter via the host interface bus. The IA61x software executes the specific command upon receipt.

11.12.1 Get/Set Algorithm Parameter and Parameter ID

The Get/Set Algorithm Parameter command allows the host to read and write the desired feature parameter specified by the Parameter ID. Setting a parameter is a two-step procedure that should follow the following sequence:

1. The Host sends `SetAlgorithmParmID` with the Parameter ID
2. The Host reads the 4-byte acknowledge
3. The Host sends `SetAlgorithmParm` with the Value
4. The Host reads the 4-byte acknowledge

The Parameter ID and its 16-bit values need to be available from the Algorithm developers.

Table 60 *Get/Set Algorithm Parameter Command Codes*

Command	Code
GetAlgorithmParm	0x8016
SetAlgorithmParmID	0x8017
SetAlgorithmParm	0x8018

11.12.2 External VQ algorithm API's

These APIs can be configured using GetParameter and SetParameter functions.

Note: These API's are more specific to Knowles VQ algorithm.

Table 61 *Knowles VoiceQ APIs*

ID	Command Name	Values	Remarks
0x5003	VS Processing Mode eVsParam_processMode	*0x0000 Keyword detection*	This command allows the host to select the operating mode of the VoiceQ system
0x5008	VS OEM Detection Sensitivity Threshold eVsParam_detectionSensitivityOEM	0x000A: 10(Max) … 0x0000: 0(min) 5 is the default value	This command allows the host to set the sensitivity threshold for OEM keyword detection (lower makes keyword detection easier)
0x5009	VS USR Detection Sensitivity Threshold eVsParam_detectionSensitivityUSR	0x000A: 10(Max) … 0x0000: 0(min) 5 is the default value	This command allows the host to set the sensitivity threshold for USR keyword detection (lower makes keyword detection easier)
0x500D	VS Auth(VoiceID) Detection Sensitivity Threshold eVsParam_detectionSensitivityAUT	0x000A: 10(Max) … 0x0000: 0(min) 5 is the default value	This command allows the host to set the sensitivity threshold for Authentication keyword detection (lower makes keyword detection easier)
0x500E	VoiceQ: Reset the algorithm state eVsParam_reset	0x0000: 0 (No Reset) 0x0001: 1 (Perform Reset)	This command allows the host to reset the VoiceQ algorithm state so that it can restart keyword detection from scratch

11.13 Algo Reset

The Host can send the Algo Reset command to reset an algorithm with the specified Algo ID. This is useful when an algorithm needs to be reconfigured with new bulk parameter data, such as keyword models. Algo Reset will do the following:

1. Call algorithm Destroy API
2. Clear the algorithm heap memory
3. Call algorithm Create API

If all algorithms need to be reset simultaneously, then the Host can send this command with Algo ID 0xFF.

Table 62 *Reset Algo Command*

Command	Code	Parameter	
		Higher Byte	Lower Byte
Algo Reset	0x806C	Algo ID	0x00

Table 63 *Knowles Algo IDs*

Knowles Library	Algo ID
VQ or BargeIn	0x50
Dummy	0xE0

Note that in order for this command to succeed, a route must not be running when this command is sent.

11.14 Get BAF Info

The Get BAF Info command (0x8027) is used in BoskoApp for getting Bosko Algorithm Framework (BAF) information. Currently, you can get consumed persistent memory and scratch memory usage, as well MIPS information, provided that firmware is compiled with support for it. The possible ParamId values are shown in **Error! Reference source not found.**.

Table 64 *Get BAF information*

<table>
<tr><th rowspan="3">Command</th><th rowspan="3">Code</th><th colspan="3">ParamId</th></tr>
<tr><th rowspan="2">Higher Byte Value</th><th colspan="2">Lower Byte Value</th></tr>
<tr><th>Higher Nibble</th><th>Lower nibble</th></tr>
<tr><td rowspan="7">Get
BAF Info</td><td rowspan="7">0x8027</td><td rowspan="4">AlgoId</td><td>0x0: Persistent memory currently used by algorithm (in words)</td><td rowspan="4">0x0: Get Memory</td></tr>
<tr><td>0x1: Scratch memory currently used by algorithm (in words)</td></tr>
<tr><td>0x2: Staging memory currently used by algorithm (in words)</td></tr>
<tr><td>0x3: Maximum available algo heap section (in words)</td></tr>
<tr><td rowspan="3">0x00</td><td>0x0: Average MIPS</td><td rowspan="2">0x1: Get MIPS (Must be compiled into BoskoApp)</td></tr>
<tr><td>0x1: Peak MIPS</td></tr>
<tr><td colspan="2">0x02: Get BAF status</td></tr>
</table>

The response of Get BAF Status (0x80270002) is 0 or 1. If the response is 1 then BAF is Active otherwise it is Inactive.

11.15 Set Presets

Presets are meant for collecting commonly used parameter / value pairs, and keeping them within the Firmware release. These values depend on the choices being made by the customers and can vary between their products. Presets can include any supported API commands. However, it is not meaningful to include commands that do not change any settings, such as Get and Sync commands. Presets are usually defined per operational mode. For example, Voice Sense mode, Normal mode, etc.

Presets allow you to send a minimal number of commands to set many parameters. Delay due to sending one command at a time from the Host processor is completely avoided in this case. By far, this is the fastest method of sending a large set of parameter updates.

Clearly, this is meant for production stage software. If the Host processor is still in the process of tuning the parameters, Presets can't be used without creating multiple builds; which can lead to a lot of confusion. Once the Presets are created using Knowles tuning tool AuViD, these can be integrated into the customer binary.

This API allows the Host processor to select a particular Preset to be applied at any given instance in time. A Preset number is in the range 0…65535. The actual number of presets that can be defined in a binary will depend on the amount of free memory available.

Table 65 *SetPreset*

Command	Code	Value
SetPreset	0x8031	Preset number

Note: Since the Preset values are based on tuning results, they are subject to change on consecutive binary builds.

11.15.1 Voice Wake Presets

Voice Wake presets can be set to run a set of commands when waking up from Stage 0 to Stage 1. This is useful when you want run a route in Stage1 that might be different from the one that was running before setting the IA61x to sleep.

Table 66 *Set VS Preset*

Command	Code	Value
Set VS Preset	0x806F	Preset number

11.16 Get Firmware/Boot ROM/Algorithm Build String

This command enables the Host to read the build string. Two API commands are provided:

- GetFirstBuildLabelChar

  This API command reads the first character of the Build string.

- GetNextBuildLabelChar

This API command is invoked iteratively to retrieve successive Build string characters, which terminates when a 0 binary value is retrieved.

Notes:

Following POR the IA61x is in the SBL state. These API commands return the Boot ROM build string. The Param ID has no effect in SBL mode.

Following a firmware download, the IA61x is in normal Application operating state. These API commands return the build string for the downloaded firmware, if algo ID is 0 and return the build string of SysConfig, if algo ID is 0xFFFF. Otherwise, they return the algorithm label of the algorithm corresponding to the algo ID.

Table 67 *GetFirstBuildLabelChar and GetNextBuildLabelChar Values*

Command	Code	ParamId
GetFirstBuildLabelChar	0x8020	0: Firmware build string
		0xFFFF: SysConfig build string.
		Others: Algorithm label if ParamId matches the algo ID.

Command	Code	
GetNextBuildLabelChar	0x8021	Retrieves next char from “build string.” Zero terminated

11.16 Soft Rest

The IA61x can be reset to start from SBL. Host must do interface detection once soft reset was done.

Note 1: Soft reset command should be a no response command.

Note 2: Once soft reset command was sent, host must wait for 30 mSec to perform interface detection.

Table 68 *Soft Reset*

Command	Code	Value
Soft Rest	0x8003	0x0001

Below figure shows the soft reset sequence for the IA61x

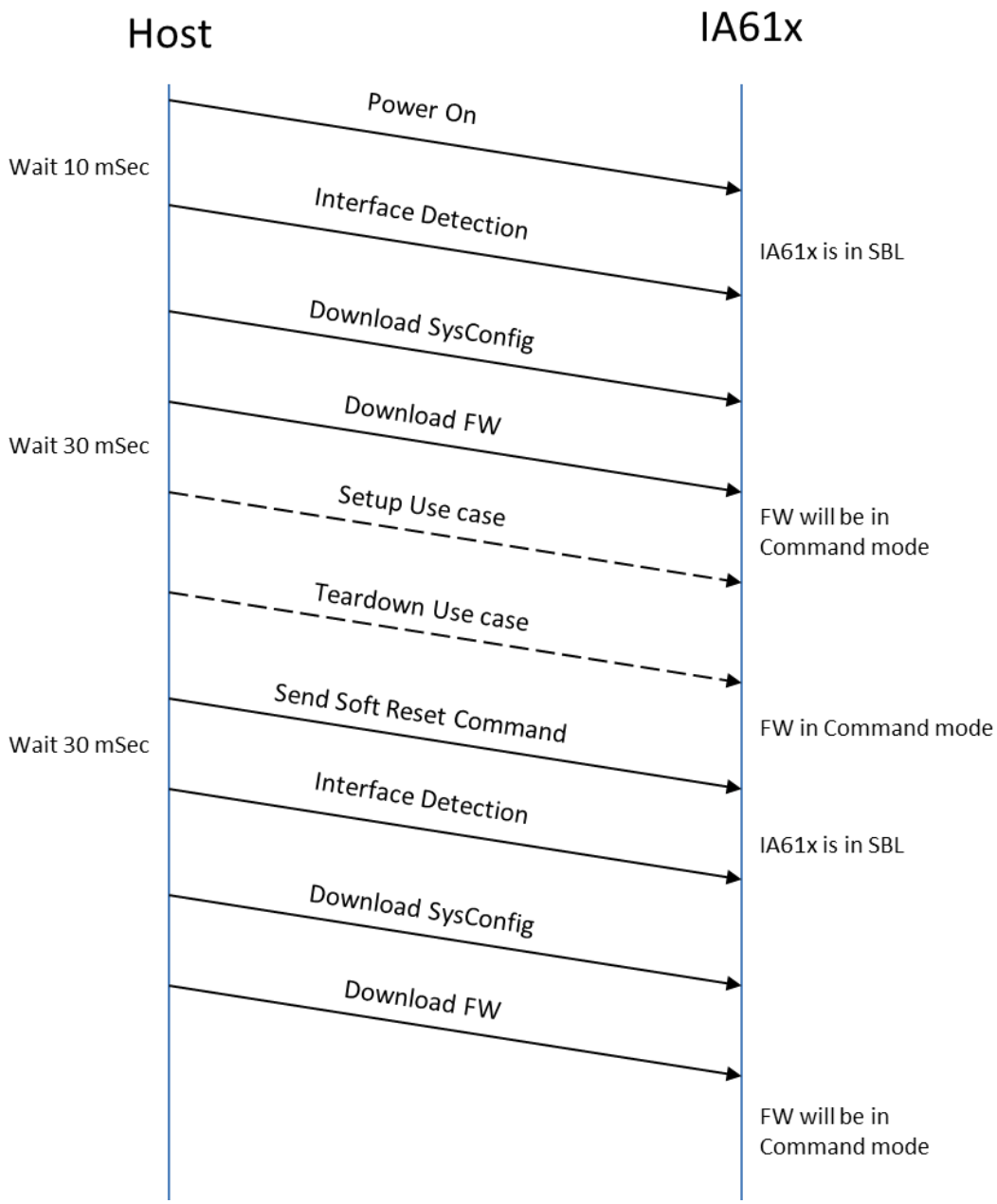

Figure 35 *Soft Reset Sequence*

Chapter 12: Audio Path Commands

The IA61x has a route configuration that is different from previous Knowles chips.

12.1 Get/Set Digital Output Gain

The IA61x provides API commands to set the digital gain for any endpoint. These gains are typically set to values that optimize the performance of the algorithms during the development phase of the project. Changing these gain settings after tuning of the system has been completed is strongly discouraged.

Gains are not meant to be used to adjust the overall transmit level to any particular value. Their only function is to set the transmit levels in the IA61x to the optimal value for Knowles algorithms. The digital gain can also be used to adjust volume.

The Host uses the Get/Set Digital Gain command to read or set the Digital Gain in 1 dB increments from -128 dB to +127 db. Get Digital Gain returns a signed 16-bit value.

Set Digital Gain sets and Get Digital Gain gets the gain value on an Endpoint. The gain is applied to, or the gain value is read from the Endpoint specified in the command.

Endpoint format is described in Section **Error! Reference source not found.**, **Error! Reference source not found.**.

Note: Set Digital Gain and Get Digital Gain are only supported on PDM Input and Output and only on Stream Manager endpoints.

Note: For low latency use cases, Set Digital Gain and Get Digital Gain are only supported on Transmit (Tx) Stream Manager endpoints.

Table 69 *Set Digital Gain*

Command	Code	High Value	Low Value: Gain
SetDigitalGain	0x8015	Endpoint ID as per section 9.4.1	0x7F Max Value(+127dB) … 0x00 Default Value (0dB) 0xFF 1st negative Value (-1dB) … 0x80 Min Value (-128dB)

Table 70 *Get Digital Gain*

Command	Code	High Value	Low Value
GetDigitalGain	0x801D	Endpoint ID as per Section **Error! Reference source not found.**	Return the gain

12.2 Set Internal Sample Rate

The IA61x microphone provides the Set Sample Rate command to set the sample rate for a particular route. The value for each supported sample rates are defined in **Error! Reference source not found.**.

For PDM Input or Output, the actual sample rate is double the internal sample rate.

Refer Section PDM clock Decimation Ratio to find valid sample rate for given PDM Audio port clock.

Example: If a route is PDM Pass-through and the Internal Sample rate is set to 48 kHz, the actual PDM port would operate at 96 kHz.

Notes:

1. Setting sample Rate is strictly enforced. Before setting a route, the Host must set the sample rate for that route. If the sample rate is not set, Firmware will give an error.

Table 71 *Set Sample Rate*

Command	Code	Value
SetSampleRate	0x8030	Where Value is: SRATE_8K = 0, SRATE_16K = 1, SRATE_24K = 2, SRATE_32K = 3, SRATE_48K = 4, SRATE_96K = 5, SRATE_192K = 6,

12.3 SelectRoute

The IA61x has an internal routing table where routes are pre-defined and each entry in the table describes one possible unique route. A detailed description of each route and its supported sample rate and frame size are provided in **Error! Reference source not found.**

Table 72 *SelectRoute*

Command	Code	Value
SelectRoute	0x8032	Route number

Notes:

1. Setting the sample rate is strictly enforced. Before setting a route, the Host must set the sample rate for the route using the command 0x8030. If the sample rate is not set, Firmware will give an error.

2. By default, the Frame Size is 8 ms for any route; you need to explicitly set the Frame Size as per use case.

3. 48 kHz and 96 kHz only give proper Outputs on a Frame Size of 1 mSec or 2 mSec

4. Route 4 and Route 5 are special Low latency pass-through routes. The routes operate on a sample-by-sample basis rather than a frame basis. Setting of Route 4 and Route 5 should be done with the suppress response option. Once the route is set, Firmware will not respond to any commands and will only answer the Stop Route command. Therefore, it is mandatory to set Routes 4 or 5 using 0x90320004 or 0x90320005.

5. Single clock source means that the clock coming from the host audio port clock is used to generate the IA61x MCLK. "Set Audio port clock frequency" (0x8042 0xXXXX) is a required command for all single clock source routes; otherwise firmware generates an error on route configuration.

Following host audio port clock frequencies are supported: 512 kHz, 768 kHz, 1.024 MHz, 1.536 MHz, 2.048 MHz, 3.072 MHz, 6.144 MHz, 9.6 MHz and 12.288 MHz.

In case of single clock source route(s), host need to make sure that PDM clock frequency(s) for all the PDMs (input or output) or Swire PDMs (input or output) audio ports part of a route, should be set same as host audio port frequency to get better audio performance.

If a single clock source route involves more than one PDMIN audio port(s), host has to make sure that all PDMIN port(s) are configured for same PDM clock, which can be externally configured using SetDevParam (PDMINx, PDM_Clk_Freq) command. In case of failure to configure for one of PDMIN port(s), unexpected behavior i.e. distorted audio may occur.

Example:

For Route#19, there are total three PDMIN ports (Internal Mic, PDMIN2 and PDMIN3) are used. In case host wants to generate 3.072MHz clock, following commands MUST be sent before SelectRoute command.

```
0x800C 0x1002 0x800D 0x00004 (Internal MIC, 3.072MHz)
0x800C 0x1202 0x800D 0x00004 (PDMIN2 MIC, 3.072MHz)
0x800C 0x1302 0x800D 0x00004 (PDMIN3 MIC, 3.072MHz)
```

6. For Internal Mic to host PDM (PDMO or SPDMO) routes, host need to configure o/p PDM port(s) for generated host audio clock.

Example:

For Route#1, if host generates 1.536MHz audio clock on PDMO0 port, below command need to send before SelectRoute command.

```
0x800C 0x1402 0x800D 0x00003 (PDMO0, 1.536MHz)
```

Error! Reference source not found. lists the currently supported route numbers and definitions:

Table 73 *Currently Supported Route Numbers and Definitions*

Route Number	Route Description	Input Ports	Output Ports	Remarks
0	1CH Pass-Through	PCM00	PCM00	
1	1CH Pass-Through	Internal mic	PDMO0	Single clock source: runs on host PDMO0 clock
2	1CH Pass-Through	Internal mic	PDMO1	Single clock source: runs on host PDMO1 clock
3	1CH Pass-Through	SoundWire PCM	SoundWire PCM	
4	1CH Low Latency Pass Through	PDMI0	PCM	Special Low latency Pass-through, Single clock source: runs on host PCM bit clock
5	Voice Wake with external PCM input	PCMI0	N/A	Buffering and bursting is not support in this route
6	Voice Wake /Continuous Voice Wake without AAD from internal mic	Internal mic	N/A	
7	1CH Voice Wake	PDMIN2	N/A	Host side PDM Input
8	Voice Wake /Continuous Voice Wake with AAD from internal mic	Internal mic	N/A	
9	1CH Pass Through	PDMI0	PCM	Single clock source: runs on host PCM bit clock
10	1CH Pass Through	PDMI0	SoundWire PCM	SoundWire hub route. Single clock source: runs on SoundWire bus clock (12.288 MHz/9.6 MHz)
11	1CH Pass-Through	SPDMI0	SoundWire PCM	
12	1CH Pass-Through	Internal mic	SPDM	SoundWire hub route. Single clock source: runs on SoundWire bus clock (12.288 MHz/9.6 MHz)
13	1CH Pass-Through	SPDMI0	SPDM	
14	1CH Pass-Through	SoundWire PCM	SPDMO0	
15	2CH Pass-Through	PCM00 PCM01	PCM00 PCM01	
16	Voice Wake with SoundWire PCM input	SoundWire PCM	N/A	SoundWire PCM Voice Wake (Internal Usage only).

Route Number	Route Description	Input Ports	Output Ports	Remarks
17	2CH Pass-Through	Internal mic PDMIN2	SoundWire PCM SoundWire PCM	SoundWire hub route. Single clock source: runs on SoundWire bus clock (12.288 MHz/9.6 MHz only)
18	2CH Pass-Through	PDMIN2 PDMIN3	SoundWire PCM SoundWire PCM	SoundWire hub route. Single clock source: runs on the SoundWire bus clock (12.288 MHz/9.6 MHz only)
19	3CH Pass-Through	Internal mic PDMIN2 PDMIN3	SoundWire PCM SoundWire PCM SoundWire PCM	SoundWire hub route. Single clock source: runs on SoundWire bus clock (12.288 MHz/9.6 MHz only)
20	2CH Pass-Through	SPCMIN0 SPCMIN2	SoundWire PCM SoundWire PCM	
21	2CH Pass-Through	SPDMIN0 SPDMIN2	SPDMO0 SPDMO1	
22	2CH Pass-Through	Internal mic PDMIN2	SPDMO0 SPDMO1	SoundWire hub route. Single clock source: runs on SoundWire bus clock (12.288MHz/9.6Mhz only)
23	2CH Pass-Through	PDMIN2 PDMIN3	SPDMO0 SPDMO1	SoundWire hub route. Single clock source: runs on SoundWire bus clock (12.288MHz/9.6Mhz only)
24	3CH Pass-Through	SPCMIN0 SPCMIN2 SPCMIN3	SoundWire PCM SoundWire PCM SoundWire PCM	
25	4CH Pass-Through	SPCMIN0 SPCMIN2 SPCMIN3 SPCMIN4	SoundWire PCM SoundWire PCM SoundWire PCM SoundWire PCM	

Note that NOT all routes are supported in each build. Routes are supported based on the control interface build type. **Error! Reference source not found.** shows the supported route table for each control interface.

Table 74 *Supported Routes based on Control Interface*

Control Interface	Supported Routes	Supported Audio Ports
I^2C	0, 1, 2, 4, 6, 7, 8, 9, 15, ,	IA61x PDM, PCM, PDM
UART	0, 1, 2, 4, 6, 7, 8, 9, 15,	IA61x PDM, PCM, PDM
SPI	1, 2, 6, 7, 8	IA61x PDM, PDM
SoundWire	3, 5, 6, 7, 8, 10, 11, 12, 13, 14, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25	IA61x PDM, SoundWire PCM, SoundWire PDM, PDM In only

12.3.1 Route: 0

Route 0 is a 1-channel PCM pass through route. It takes its input from a PCM port and sends its output to a PCM Port.

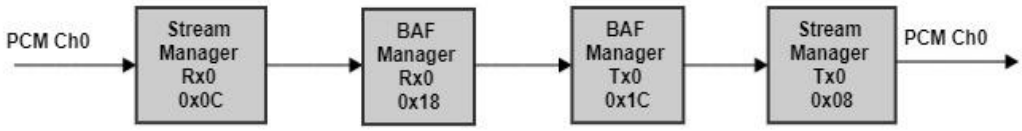

Figure 36 *Route 0*

Table 75 *Data Rate and Frame Size for Route 0*

Route Number	Route Description	Input Ports	Output Ports	Supported Sample Rate	Supported Frame Sizes
0	1CH Pass-Through	PCM00	PCM00	8K	0.5, 1, 2, 8, 10, 15
				16K	0.5, 1, 2, 8, 10, 15
				24K	0.5, 1, 2
				32K	0.5, 1, 2
				48K	0.5, 1, 2
				96K	0.5, 1
				192K	0.5

Note:

1 For LA binaries, please refer to Section**Error! Reference source not found.** for supported sample rates and frame sizes.

2 BAF endpoints are applicable only for LA builds.

Example:

1. Commands to set up 1Ch Pass-Through, 16K, PCM00 -> PCM00, 8Ms frame Slave

```
0x80300001 (16k sample rate)
0x8032 0x0000
```

2. Commands to set up 1Ch Pass-Through, 8K, PCM00 -> PCM00, I2S Master

```
0x800C 0x0A09 0x800D 0x00001 (Master mode)
0x800C 0x1700 0x800D 0x0000F (16bit word Length,  default)
0x800C 0x1701 0x800D 0x00001 (2 words,  default)
0x800C 0x1702 0x800D 0x00008 (8000 Rate,  default)
0x803 00000 (8k sample rate)
0x8032 0x0000
```

3. Commands to set up 1Ch Pass-Through, 16K, PCM00 -> PCM00, I2S Master

```
0x800C 0x0A09 0x800D 0x00001 (Master mode)
0x800C 0x1700 0x800D 0x0000F (16bit word length,  default)
0x800C 0x1701 0x800D 0x00001 (2 words,  default)
0x800C 0x1702 0x800D 0x00010 (16000 Rate,  default)
0x80300001 (16k sample rate)
0x8032 0x0000
```

12.3.2 Route: 1

Route 1 is a 1-channel PDM pass through route. It takes its input from a PDM Mic port and sends its output to a PDM Output Port (Left channel).

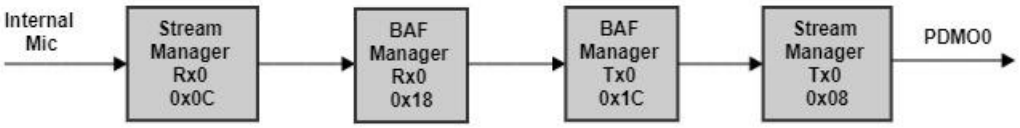

Figure 37 *Route 1*

Note: Decimator and Interpolator are present as part of the route. The actual sample rate is twice the internal sample rate.

Note:

1 In this route the IA61x must be PDM slave, so do not configure the IA61x as PDM master.

2 For LA binaries, please refer to Section**Error! Reference source not found.** for supported sample rates and frame sizes.

3 For select audio port clock frequencies refer section 12.10.

4 BAF endpoints are applicable only for LA builds.

Table 76 *Data Rate and Frame Size for Route 1*

Route Number	Route Description	Input Ports	Output Ports	Supported Sample Rate	Supported Frame Sizes
1	1CH Pass-Through	Internal mic	PDMO0	8K	0.5, 1, 2, 8, 10, 15
				16K	0.5, 1, 2, 8, 10, 15
				24K	0.5, 1, 2
				32K	0.5, 1, 2
				48K	0.5, 1, 2
				96K	0.5
				192K	0.5

Example:

Commands to setup 1-ch Pass-Through route, 8K, 10ms, 512KHz host audio clock

```
0x8030 0000 (8k sample rate)
0x800C 0x1002 0x800D 0x0000 (Internal MIC clock to 512 KHz)
0x800C 0x1402 0x800D 0x0000 (PDMO0 clock to 512 KHz)
0x8042 0000 (Set Audio port clock to 512 kHz)
0x8035 000A (10 msec frame size)
```

```
0x8032 0001 ( 1CH Internal Mic->PDMO0)
```

12.3.3 Route: 2

Route 2 is a 1-channel PDM pass-through route. It takes its input from PDM Mic port and sends its output to the PDM Output Port 1 (Right channel)

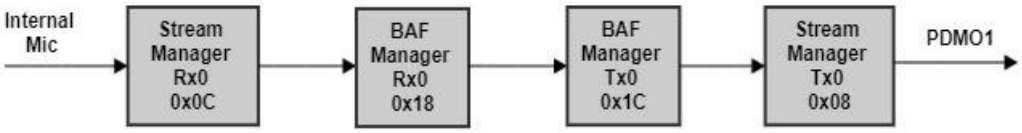

Figure 38 *Route 2*

Note:

1 Decimator and Interpolator are present as part of the route. The actual sample rate is twice the internal sample rate.

2 In this route the IA61x must be PDM slave so do not configure the IA61x as PDM master.

3 For LA binaries, please refer to Section**Error! Reference source not found.** for supported sample rates and frame sizes.

4 For select audio port clock frequencies refer section 12.10.

5 BAF endpoints are applicable only for LA builds.

6 PDM dual mono output is not supported in Route 2

Table 77 *Data Rate and Frame Size for Route 2*

Route Number	Route Description	Input Ports	Output Ports	Supported Sample Rate	Supported Frame Sizes
2	1CH Pass-Through	Internal mic	PDMO1	8K	0.5, 1, 2, 8, 10, 15
				16K	0.5, 1, 2, 8, 10, 15
				24K	0.5, 1, 2
				32K	0.5, 1, 2
				48K	0.5, 1, 2
				96K	0.5
				192K	0.5

Example:

Commands to setup 1-ch Pass-Through route, 8K, 10ms, 512KHz host audio clock

```
0x8030 0000 (8k sample rate)
0x800C 0x1002 0x800D 0x0000 (Internal MIC clock to 512 KHz)
0x800C 0x1502 0x800D 0x0000 (PDMO1 clock to 512 KHz)
```

```
0x8042 0x0000 (Set Audio port clock to 512 kHz)
0x8035 0x000A (10 msec frame size)
0x8032 0x0002 (1CH PT Internal Mic->PDMO1)
```

12.3.4 Route: 3

Route 3 is a 1-channel SoundWire pass-through route. It takes its input from the SoundWire PCM port and sends its output on SoundWire PCM.

Error! Reference source not found. shows the currently supported rates and frame sizes.

Table 78 *Data Rate and Frame Size for Route 3*

Route Number	Route Description	Input Ports	Output Ports	Supported Sample Rate	Supported Frame Sizes
3	1CH Pass-Through	SPCM00	SPCM0	8K	0.5, 1, 2, 8, 10, 15
				16K	0.5, 1, 2, 8, 10, 15
				24K	0.5, 1, 2
				32K	0.5, 1, 2
				48K	0.5, 1, 2
				96K	0.5, 1
				192K	0.5

Example:

Commands to set up a 1-Channel Pass-Through, 32K, SPCM0 -> SPCM0, Master

In SBL:

```
0x8004 0001 (Set Audio Data Port as I2S/PCM)
```

In BoskoApp

```
0x8030 0000 (8k sample rate)
0x8032 0003 (Route 3)
```

12.3.5 Route: 4

Route 4 is a Low latency 1-channel PDM to PCM pass-through route. It takes its input from a PDM Mic port and sends its output to a PCM Output Port

Notes:

1. Half-band is present as part of the route. The actual sample rate is twice the internal sample rate.
2. Route has to be setup with command with response suppressed bit set.
3. Once the route is set, the IA61x goes to low power mode and to stop route host needs to toggle wake up pin
4. TIEQ routes works on a sample-by-sample basis, not on frame size boundary; so there is no need to send a Frame-Size configuration command (0x8035 0xYYYY)
5. In this route the IA61x must be I2S slave so do not configure the IA61x as I2S master

Table 79 *Data Rate and Frame Size for Route 4*

Route Number	Route Description	Input Ports	Output Ports	Supported Sample Rate
4	1CH Low LatencyPass-Through	Internal Mic	PCM0	8K
				16K
				24K
				48K

Example:

Commands to set up 1Ch Pass-Through, 48K, Internal Mic -> PCM0, 1.536MHz host audio clock

In SBL:

```
0x8004 0001 (Set Audio Data Port as I²S/PCM)
```

In BoskoApp:

```
0x8030 0x0004 (48k sample rate)
0x800C 0x1002 0x800D 0x0003 (Internal MIC clock to 1.536MHz)
0x8042 0x0003 (Set Audio port clock to 1.536MHz)
0x9032 0x0004 (1CH PT Low Latency Internal Mic->PCM0)
```

12.3.6 Route: 5

Route 5 is a Voice Wake route with an external PCM input on left. It takes its input from a PCM port and feeds it to the Voice Wake algo.

Figure 39 *Route 5*

Notes:

1. Buffering and Bursting are support in this route.
2. This is a Voice Wake route, so if the ALGO supports the desired sample rate and frame size this route should work.
3. Low power mode is only supported for UART+I2S configuration.
4. I2S data port rate API command is mandatory for this route.

Table 80 *Data Rate and Frame Size for Route 5*

Route Number	Route Description	Input Ports	Output Ports	Supported Sample Rate	Supported Frame Sizes
5	Voice Wake Route (external PCM input)	PCMIN0	NA	16K	8, 10, 15, 16

Example:

Commands to set up Voice Wake route with external PCM input:

In SBL:

```
0x8004 0001 (Set Audio Data Port as I²S/PCM)
```

In BoskoApp:

```
0x8030 0001 (16 KHz sample rate)
0x802C 1000 (16 Khz Port rate)
0x8035 0010 (16 msec frame size)
0x8032 0005 (Route 5)
```

12.3.7 Route: 6

Route 6 is a Voice Wake/Continuous Voice Wake route that does not use AAD in the internal mic. It takes its input from the Internal Mic Internal Mic port and feeds it to Voice Wake Algo.

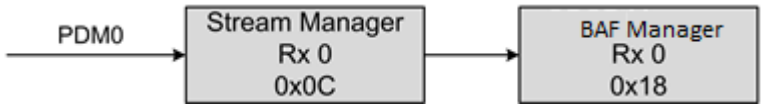

Figure 40 *Route 6*

Notes:

1. Half-band is present as part of route; the actual sample rate is twice the internal sample rate.
2. This is a Voice Wake route, so if the ALGO supports the desired sample rate and frame size this route should work.

Table 81 *Data Rate and Frame Size for Route 6*

Route Number	Route Description	Input Ports	Output Ports	Supported Sample Rate	Supported Frame Sizes
6	Voice Wake Route (for cases when AAD of Internal Mic is not used)	Internal mic	NA	8K, 16K	8, 10, 15, 16

Examples

1. Commands to set up 16 K sample rate, 16 mSec frame size, Voice Wake Route 6.

```
0x8035 0010 (16 msec frame size)
0x8030 0001 (16 KHz sample rate)
0x802A 0000 (Disable buffering)
0x8032 0006 (Route 6)
```

2. Commands to set up 16 K sample rate, 16 mSec frame size, Continuous Voice Wake Route 6.

```
0x8035 0010 (16 msec frame size)
0x8030 0001 (16 KHz sample rate)
0x8034 0002 (16 bits Buffer Data Format)
0x8032 0006 (Route 6)
```

3. Commands to set up 16 K sample rate, 16 mSec frame size, Continuous Voice Wake Route 6 without keyword preservation.

```
0x8035 0010 (16 msec frame size)
```

```
0x8030 0001 (16 KHz sample rate)
0x8034 0002 (16 bits Buffer Data Format)
0x802A 0001 (Disable keyword preservation)
0x8032 0006 (Route 6)
```

12.3.8 Route: 7

Route 7 is a PDM Voice Wake route that takes its input from the Host side PDM Input Mic port and feeds it to the Voice Wake Algo.

Notes:

1. Half-band is present as part of the route. The actual sample rate is twice the internal sample rate.
2. This is a Voice Wake route, so if the ALGO supports the desired sample rate and frame size this route should work.
3. PDMIN2 is the host side PDM input interface. This route is used to test host side PDM. There is no need to do keyword detection, as verification of audio through streaming is good enough.
4. Low power mode is not supported in this route.

Table 82 *Data Rate and Frame Size for Route 7*

Route Number	Route Description	Input Ports	Output Ports	Supported Sample Rate	Supported Frame Sizes
7	Voice Wake (with external PDM input)	PDMIN2	NA	8K, 16K	8, 10, 15, 16

Example:

Commands to setup 2CH Buffered Voice Wake route, 16K, 16ms

```
0x8034 0x0002 (keyword uploading will be in 16 bits data format)
0x8035 0x0010 (Audio Frame Buffer Size, 16ms, depends on 3rd party algorithm)
0x80300001 (16k sample rate)
0x8032 0x0007 (2CH Buffered Voice Wake route)
```

12.3.9 Route: 8

Route 8 is a Voice Wake/Continuous Voice Wake route with AAD from the internal microphone. It takes its inputs from the Internal Mic Internal Mic and feeds it to the Voice Wake Algo.

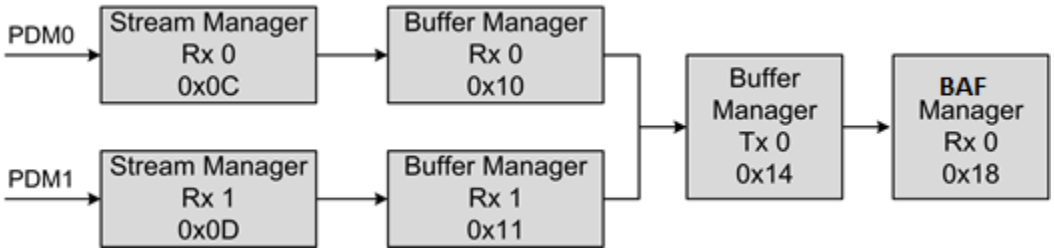

Figure 41 *Route 8*

Notes:

1. Half-band is present as part of route. The actual sample rate is twice the internal sample rate.
2. This is a Voice Wake route, so if the ALGO supports the desired sample rate and frame size this route should work.

Table 83 *Data Rate and Frame Size for Route 8*

Route Number	Route Description	Input Ports	Output Ports	Supported Sample Rate	Supported Frame Sizes
8	Voice Wake Route (for cases when AAD of Internal Mic is used)	Internal mic	NA	8K, 16K	8, 10, 15, 16

Example:

3. Commands to set up 16 K sample rate, 16 mSec frame size, Voice Wake Route 8.

```
0x8035 0010 (16 msec frame size)
0x8030 0001 (16 KHz sample rate)
0x802A 0000 (Disable buffering)
0x8032 0008 (Route 8)
```

4. Commands to set up 16 K sample rate, 16 mSec frame size, Continuous Voice Wake Route 8.

```
0x8035 0010 (16 msec frame size)
0x8030 0001 (16 KHz sample rate)
0x8034 0002 (16 bits Buffer Data Format)
0x8032 0008 (Route 8)
```

5. Commands to set up 16 K sample rate, 16 mSec frame size, Continuous Voice Wake Route 8 without keyword preservation.

```
0x8035 0010 (16 msec frame size)
0x8030 0001 (16 KHz sample rate)
0x8034 0002 (16 bits Buffer Data Format)
0x802A 0001 (Disable keyword preservation)
0x8032 0008 (Route 8)
```

12.3.10 Route: 9

Route 9 is a 1-channel PDM to PCM pass through route. It takes its Input from the PDM Mic port and sends its output to a PCM Output Port.

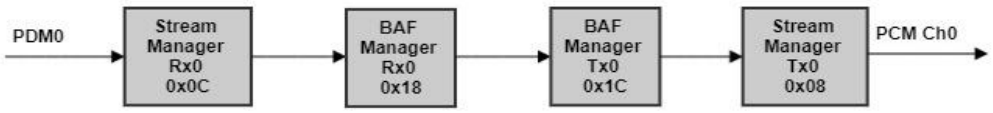

Figure 42 *Route 9*

Note: Half-band is present as part of the route. The actual sample rate is twice the internal sample rate.

Note: In this route, the IA61x must be PDM slave so do not configure the IA61x as PDM master.

1 For LA binaries, please refer to Section**Error! Reference source not found.** for supported sample rates and frame sizes.

2 For select audio port clock frequencies refer section 12.10.

3 For 8K sample rate only 32bit word length is supported, refer section 11.2.2

4 BAF endpoints are applicable only for LA builds.

Table 84 *Data Rate and Frame Size for Route 9*

Route Number	Route Description	Input Ports	Output Ports	Supported Sample Rate	Supported Frame Sizes
9	1CH Low Through	Internal mic	PCM0	8K	0.5, 1, 2, 8, 10, 15
				16K	0.5, 1, 2, 8, 10, 15
				24K	0.5, 1, 2
				32K	0.5, 1, 2
				48K	0.5, 1, 2
				96K	0.5
				192K	0.5

Example:

To setup the internal mic to PCM, 48K, 1ms, Single Clock Source with 1.536 MHz port clock

```
0x8030 0x0004 (48K route sample rate)
0x800C 0x1002 0x800D 0x0003 (Internal MIC clock to 1.536MHz)
0x8042 0x0003 (1.536MHz host audio port clock frequency)
0x8035 0x0001 (Audio Frame Buffer Size, 1ms)
0x8032 0x0009 (1CH PT Internal mic ->PCM)
```

12.3.11 Route: 10

Route 10 is a 1-channel PDM to SoundWire PCM pass-through route. It takes its input from the PDM Mic port and sends its output to the SoundWire PCM Output Port. It is a single clock source route that uses the SoundWire bus clock to drive the audio data in hardware. It supports 12.288 MHz and 9.6 MHz as audio clock frequency configurations for the 0x8042 command, which is required.

Note: Half-band is present as part of the route. The actual sample rate is twice the internal sample rate.

Table 85 *Data Rate and Frame Size for Route 10*

Route Number	Route Description	Input Ports	Output Ports	Supported Sample Rate	Supported Frame Sizes
10	1CH Pass-through Through	Internal mic	SoundWire PCM	8K	0.5, 1, 2, 8, 10, 15
				16K	0.5, 1, 2, 8, 10, 15
				24K	0.5, 1, 2
				48K	0.5, 1, 2
				96K	0.5
				192K	0.5

Example:

To setup the internal Mic to SPCM, 48K, 1ms, Single Clock Source with 1.536 MHz port clock

```
0x8030 0x0004 (48K route sample rate)
0x800C 0x1002 0x800D 0x0003 (Internal MIC clock to 1.536MHz)
0x8042 0x0003 (1.536MHz host audio port clock frequency)
0x8035 0x0001 (Audio Frame Buffer Size, 1ms)
0x8032 0x000A (1CH PT Internal mic ->SPCM)
```

12.3.12 Route: 11

Route 11 is a 1-channel SoundWire PDM to SoundWire PCM pass-through route. It takes its input from the SoundWire PDM port and sends its output to the SoundWire PCM Output Port.

Note: Half-band is present as part of route. The actual sample rate is twice the internal sample rate.

Table 86 *Data Rate and Frame Size for Route 11*

Route Number	Route Description	Input Ports	Output Ports	Supported Sample Rate	Supported Frame Sizes
11	1CH Pass through	SPDMIN	SoundWire PCM	8K	0.5, 1, 2, 8, 10, 15
				16K	0.5, 1, 2, 8, 10, 15
				24K	0.5, 1, 2
				48K	0.5, 1, 2
				96K	0.5
				192K	0.5

12.3.13 Route: 12

Route 12 is a 1-channel PDM pass-through route. It takes its input from the PDM Mic port and sends its output to the SoundWire PDM Output Port. It is single clock source route, that uses the SoundWire bus clock to drive the complete audio data in hardware. It supports 12.288 MHz and 9.6 MHz as audio clock frequency configurations for the 0x8042 command, which is required.

Note: Decimator and Interpolator are present as part of the route. The actual sample rate is twice the internal sample rate.

Table 87 *Data Rate and Frame Size for Route 12*

Route Number	Route Description	Input Ports	Output Ports	Supported Sample Rate	Supported Frame Sizes
12	1CH Pass-Through	Internal mic	SPDMO0	8K	0.5, 1, 2, 8, 10, 15
				16K	0.5, 1, 2, 8, 10, 15
				24K	0.5, 1, 2
				48K	0.5, 1, 2
				96K	0.5
				192K	0.5

Example:

To setup the internal Mic to SPDMO0, 48K, 1ms, Single Clock Source with 1.536 MHz port clock

```
0x8030 0x0004 (48K route sample rate)
0x800C 0x1002 0x800D 0x0003 (Internal MIC clock to 1.536MHz)
0x800C 0x1C02 0x800D 0x0003 (SPDMO0 clock to 1.536MHz)
0x8042 0x0003 (1.536MHz host audio port clock frequency)
0x8035 0x0001 (Audio Frame Buffer Size, 1ms)
0x8032 0x000C (1CH PT Internal mic ->SPDMO0)
```

12.3.14 Route: 13

Route 13 is a 1-channel PDM pass-through route. It takes its input from the SoundWire PDM port and sends its output to the SoundWire PDM Output Port.

Note: Decimator and Interpolator are present as part of the route. The actual sample rate is twice the internal sample rate.

Table 88 *Data Rate and Frame Size for Route 13*

Route Number	Route Description	Input Ports	Output Ports	Supported Sample Rate	Supported Frame Sizes
13	1CH Pass-Through	SPDMIN0	SPDMO0	8K	0.5, 1, 2, 8, 10, 15
				16K	0.5, 1, 2, 8, 10, 15
				24K	0.5, 1, 2
				48K	0.5, 1, 2
				96K	0.5
				192K	0.5

12.3.15 Route: 14

Route 14 is a 1-channel pass-through route. It takes its input from the SoundWire PCM port and sends its output to the SoundWire PDM Output Port.

Note: Decimator and Interpolator are present as part of the route. The actual sample rate is twice the internal sample rate.

Table 89 *Data Rate and Frame Size for Route 14*

Route Number	Route Description	Input Ports	Output Ports	Supported Sample Rate	Supported Frame Sizes
14	1CH Pass-Through	SoundWire PCM	SPDMO0	8K	0.5, 1, 2, 8, 10, 15
				16K	0.5, 1, 2, 8, 10, 15
				24K	0.5, 1, 2
				48K	0.5, 1, 2
				96K	0.5
				192K	0.5

12.3.16 Route: 15

Route 15 is a 2-channel PCM pass-through route. It takes its input from a PCM port and sends its output to a PCM Port.

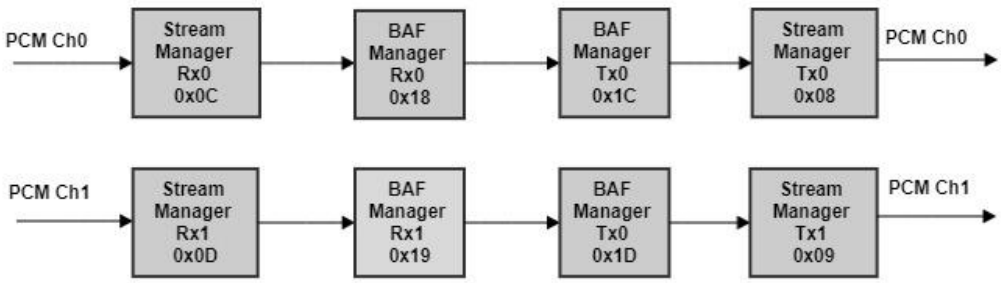

Figure 43 *Route 15*

Note:

1 For LA binaries, please refer to Section**Error! Reference source not found.** for supported sample rates and frame sizes.

2 BAF endpoints are applicable only for LA builds.

Table 90 *Data Rate and Frame Size for Route 15*

Route Number	Route Description	Input Ports	Output Ports	Supported Sample Rate	Supported Frame Sizes
15	2CH Pass-Through	PCM00 PCM01	PCM00 PCM01	8K	0.5, 1, 2, 8, 10
				16K	0.5, 1, 2, 8, 10
				24K	0.5, 1, 2
				32K	0.5, 1, 2
				48K	0.5, 1, 2
				96K	0.5, 1
				192K	0.5

12.3.17 Route: 16

Route 16 is a 1-channel SoundWire PCM Voice Wake route. It takes its input from the SoundWire PCM port and sends output to Voice Wake Algo. This route is for internal use only.

Note: Low power mode is not supported in this route.

Table 91 *Data Rate and Frame Size for Route 16*

Route Number	Route Description	Input Ports	Output Ports	Supported Sample Rate	Supported Frame Sizes
16	Voice Wake Route (SoundWire PCM input)	SoundWire PCM	NA	8K, 16K	8, 10, 15, 16

12.3.18 Route: 17

Route 17 is a 2-channel PDM to SoundWire PCM pass-through route. It is designed for SoundWire hub use cases where the input is received from the IA61x MIC and the Host PDM i.e. dumb MIC (Left only) and transmitted to the SoundWire PCM as a stereo output. It is a single clock source route, that uses the SoundWire bus clock to drive the complete audio data in hardware. This route only supports 12.288 MHz as the audio clock frequency configuration for the 0x8042 command, which is required.

Note: Half-band is present as part of the route. The actual sample rate is twice the internal sample rate.

Table 92 *Data Rate and Frame Size for Route 17*

Route Number	Route Description	Input Ports	Output Ports	Supported Sample Rate	Supported Frame Sizes
17	2CH Pass-Through (SoundWire Hub Use Case)	Internal mic PDMIN2	SPCM00 SPCM01	8K	0.5, 1, 2, 8, 10
				16K	0.5, 1, 2
				24K	0.5, 1, 2
				48K	0.5, 1,2
				96K	0.5
				192K	0.5

Example:

To setup the internal Mic/PDMIN2 to SPCM00/SPCM01, 48K, 1ms, Single Clock Source with 1.536 MHz port clock

```
0x8030 0x0004 (48K route sample rate)
0x800C 0x1002 0x800D 0x0003 (Internal MIC clock to 1.536MHz)
0x800C 0x1202 0x800D 0x0003 (PDMIN2 clock to 1.536MHz)
0x8042 0x0003 (1.536MHz host audio port clock frequency) (see
section 12.10)
0x8035 0x0001 (Audio Frame Buffer Size,  1ms)
0x8032 0x0011 (2CH PT Internal mic/PDMIN2 ->SPCM00/SPCM01)
```

12.3.19 Route: 18

Route 18 is a 2-channel PDM to SoundWire PCM pass-through route. It is designed for SoundWire hub use cases where input is received from Host PDM Left/Right channel i.e. dumb MICs and transmitted to SoundWire PCM as stereo output. It is a single clock source route that uses the SoundWire bus clock to driver complete audio data in hardware. This route supports 12.288 MHz and 9.6 MHz as audio clock frequency configurations for the 0x8042 command, which is required.

Note: Half-band is present as part of the route. The actual sample rate is twice the internal sample rate.

Table 93 *Data Rate and Frame Size for Route 18*

Route Number	Route Description	Input Ports	Output Ports	Supported Sample Rate	Supported Frame Sizes
18	2CH Pass-Through (SoundWire Hub Use Case)	PDMIN2 PDMIN3	SPCM00 SPCM01	8K	0.5, 1, 2, 8, 10
				16K	0.5, 1, 2
				24K	0.5, 1, 2
				48K	0.5, 1,2
				96K	0.5
				192K	0.5

Example:

To setup the PDMIN2/PDMIN3 to SPCM00/SPCM01, 48K, 1ms, Single Clock Source with 1.536 MHz port clock

```
0x8030 0x0004 (48K route sample rate)
0x800C 0x1202 0x800D 0x0003 (PDMIN2 clock to 1.536MHz)
0x800C 0x1302 0x800D 0x0003 (PDMIN3 clock to 1.536MHz)
0x8042 0x0003 (1.536MHz host audio port clock frequency) (see
section 12.10)
0x8035 0x0001 (Audio Frame Buffer Size,  1ms)
0x8032 0x0012 (2CH PT PDMIN2/PDMIN3 ->SPCM00/SPCM01)
```

12.3.20 Route: 19

Route 19 is a 3-channel PDM to SoundWire PCM pass-through route. It is designed for SoundWire hub use cases where one input channel comes from the IA61x and two other input channels come from the Host PDM Left/Right channels (PDMIN2/PDMIN3). These channels are transmitted to SoundWire PCM as three-channel output. This is single clock source route, that uses the SoundWire bus clock to drive the complete audio data in hardware. This route only supports 12.288 MHz as the audio clock frequency configuration for the 0x8042 command, which is required.

Note: Half-band is present as part of the route. The actual sample rate is twice the internal sample rate.

Table 94 *Data Rate and Frame Size for Route 19*

Route Number	Route Description	Input Ports	Output Ports	Supported Sample Rate	Supported Frame Sizes
19	3CH Pass-Through (SoundWire Hub Use Case)	Internal mic PDMIN2 PDMIN3	SPCM00 SPCM01 SPCM02	8K	0.5, 1, 2, 8, 10
				16K	0.5, 1, 2
				24K	0.5, 1, 2
				48K	0.5, 1,2
				96K	0.5
				192K	0.5

Example:

To setup the Internal Mic/ PDMIN2/PDMIN3 to SPCM00/SPCM01/SPCM02, 48K, 1ms, Single Clock Source with 1.536 MHz port clock

```
0x8030 0x0004 (48K route sample rate)
0x800C 0x1002 0x800D 0x0003 (Internal Mic clock to 1.536MHz)
0x800C 0x1202 0x800D 0x0003 (PDMIN2 clock to 1.536MHz)
0x800C 0x1302 0x800D 0x0003 (PDMIN3 clock to 1.536MHz)
0x8042 0x0003 (1.536MHz host audio port clock frequency) (see
section 12.10)
0x8035 0x0001 (Audio Frame Buffer Size,  1ms)
0x8032 0x0013 (3CH PT Internal Mic/PDMIN2/PDMIN3 -
>SPCM00/SPCM01/SPCM02)
```

12.3.21 Route: 20

Route 20 is a 2-channel SoundWire PCM to SoundWire PCM pass-through route.

Table 95 *Data Rate and Frame Size for Route 20*

Route Number	Route Description	Input Ports	Output Ports	Supported Sample Rate	Supported Frame Sizes
20	2CH SoundWire PCM Pass-Through ()	SPCMIN0 SPCMIN1	SPCMO0 SPCMO1	8K	0.5, 1, 2, 8, 10
				16K	0.5, 1, 2, 8, 10
				24K	0.5, 1, 2
				48K	0.5, 1, 2
				96K	0.5, 1
				192K	0.5

12.3.22 Route: 21

Route 21 is a 2-channel SoundWire PDM to SoundWire PDM pass-through route.

Table 96 *Data Rate and Frame Size for Route 21*

Route Number	Route Description	Input Ports	Output Ports	Supported Sample Rate	Supported Frame Sizes
21	2CH SPDM Pass-Through	SPDMIN0 SPDMIN1	SPDMO0 SPDMO1	8K	0.5, 1, 2, 8, 10
				16K	0.5, 1, 2
				24K	0.5, 1, 2
				48K	0.5, 1,2
				96K	0.5
				192K	0.5

12.3.23 Route: 22

Route 22 is a 2-channel PDM to SoundWire PDM pass-through route. It is designed for SoundWire hub use cases where one input channel comes from the IA61x and other input channels comes from the Host PDM Left/Right channel PDMIN2. These channels are transmitted to the SoundWire PDM as two-channel output. It is single clock source route, that uses SoundWire bus clock to driver complete audio data in hardware. It supports 12.288 MHz and 9.6 MHz as audio clock frequency configuration for 0x8042 commands, which is required.

Table 97 *Data Rate and Frame Size for Route 22*

Route Number	Route Description	Input Ports	Output Ports	Supported Sample Rate	Supported Frame Sizes
22	2CH PDM-SPDM Pass-Through (SoundWire Hub Case)	Internal mic PDMIN2	SPDMO0 SPDMO1	8K	0.5, 1, 2, 8, 10
				16K	0.5, 1, 2
				24K	0.5, 1, 2
				48K	0.5, 1,2
				96K	0.5
				192K	0.5

Example:

To setup the Internal Mic/ PDMIN2 to SPDM00/SPDM01, 48K, 1ms, Single Clock Source with 1.536 MHz port clock

```
0x8030 0x0004 (48K route sample rate)
0x800C 0x1002 0x800D 0x0003 (Internal Mic clock to 1.536MHz)
0x800C 0x1202 0x800D 0x0003 (PDMIN2 clock to 1.536MHz)
0x800C 0x1C02 0x800D 0x0003 (SPDMO0 clock to 1.536MHz)
0x800C 0x1D02 0x800D 0x0003 (SPDMO1 clock to 1.536MHz)
0x8042 0x0003 (1.536MHz host audio port clock frequency) (see
section 12.10)
0x8035 0x0001 (Audio Frame Buffer Size,  1ms)
0x8032 0x0016 (2CH PT Internal Mic/PDMIN2->SPDM00/SPDM01)
```

12.3.24 Route: 23

Route 23 is a 2-channel PDM to SoundWire PDM pass-through route. It is designed for SoundWire hub use cases where both input channels are coming from the Host PDM Left/Right channels (PDMIN2/PDMIN3). These channels are then transmitted to SoundWire PDM as two-channel output. This route is a single clock source route that uses the SoundWire bus clock to drive the complete audio data in hardware. It supports 12.288 MHz and 9.6 MHz as audio clock frequency configuration for the 0x8042 command, which is required.

Table 98 *Data Rate and Frame Size for Route 23*

Route Number	Route Description	Input Ports	Output Ports	Supported Sample Rate	Supported Frame Sizes
23	2CH PDM-SPDM Pass-Through (SoundWire Hub Case)	PDMIN2 PDMIN3	SPDMO0 SPDMO1	8K	0.5,1, 2, 8, 10
				16K	0.5,1, 2
				24K	0.5,1, 2
				48K	0.5,1, 2
				96K	0.5
				192K	0.5

Example:

To setup the PDMIN2/PDMIN3 to SPDM00/SPDM01, 48K, 1ms, Single Clock Source with 1.536 MHz port clock

```
0x8030 0x0004 (48K route sample rate)
0x800C 0x1202 0x800D 0x0003 (PDMIN2 clock to 1.536MHz
0x800C 0x1302 0x800D 0x0003 (PDMIN3 clock to 1.536MHz)
0x800C 0x1C02 0x800D 0x0003 (SPDMO0 clock to 1.536MHz)
0x800C 0x1D02 0x800D 0x0003 (SPDMO1 clock to 1.536MHz)
0x8042 0x0003 (1.536MHz host audio port clock frequency) (see
section 12.10)
0x8035 0x0001 (Audio Frame Buffer Size,  1ms)
0x8032 0x0017 (2CH PT PDMIN2/PDMIN3->SPDM00/SPDM01)
```

12.3.25 Route: 24

Route 24 is a 3-channel SoundWire PCM to SoundWire PCM pass-through route.

Table 99 *Data Rate and Frame Size for Route 24*

Route Number	Route Description	Input Ports	Output Ports	Supported Sample Rate	Supported Frame Sizes
24	3CH SoundWire PCM Pass-Through ()	SPCMIN0 SPCMIN1 SPCMIN2	SPCMO0 SPCMO1 SPCMO2	8K	0.5,1, 2, 8, 10
				16K	0.5,1, 2, 8, 10
				24K	0.5,1, 2
				48K	0.5,1, 2
				96K	0.5,1
				192K	0.5

12.3.26 Route: 25

Route 25 is a 3-channel SoundWire PCM to SoundWire PCM pass-through route.

Table 100 *Data Rate and Frame Size for Route 25*

Route Number	Route Description	Input Ports	Output Ports	Supported Sample Rate	Supported Frame Sizes
25	4CH SoundWire PCM Pass-Through ()	SPCMIN0 SPCMIN1 SPCMIN2 SPCMIN3	SPCMO0 SPCMO1 SPCMO2 SPCMO3	8K	0.5,1, 2, 8, 10
				16K	0.5,1, 2, 8, 10
				24K	0.5,1, 2
				48K	0.5,1, 2
				96K	0.5,1
				192K	0.5

12.3.27 Route: 26

Route 26 is a 2-channel PDM In to 1-channel PDM out route. It takes its 1 input from internal mic and 1 input from external mic. This route can be used for two mic NS algo.

Note: Half-band is present as part of the route. The actual sample rate is twice the internal sample rate.

Note: In this route the IA61x must be PDM slave so do not configure the IA61x as PDM master.

Note: In LA build if algorithm is not present then Internal Mic data are bypassed to output. External mic connected to data is not routed on output port.

Note: This route is not supported in VQ or VW build.

Note: External Mic should be connected on P2 pin.

Table 101 *Data Rate and Frame Size for Route 26*

Route Number	Route Description	Input Ports	Output Ports	Supported Sample Rate	Supported Frame Sizes
26	2-CHIN PDM In	Internal mic PDMIN2	PDMO0	8K	0.5, 1, 2, 8, 10
				16K	0.5, 1, 2, 8, 10
				24K	0.5, 1, 2
				48K	0.5, 1, 2
				96K	0.5
				192K	0.5

Example:

```
0x8035 0002 (2 mSec frame size)
0x8030 0000 (8 KHz sample rate)
0x8042 0001 (0.768MHz host audio port clock frequency)
0x8032 001A (Route 26)
```

12.3.28 Route: 27

Route 27 is a 2-channel PDM In to 2-channel PDM out route. It takes its 1 input from internal mic and 1 input from external mic. This route can be used for two mic NS algo.

Note: Half-band is present as part of the route. The actual sample rate is twice the internal sample rate.

Note: In this route the IA61x must be PDM slave so do not configure the IA61x as PDM master.

Note: In LA build if algorithm is not present then Internal Mic data are bypassed to output. External mic connected to data is not routed on output port.

Note: This route is not supported in VQ or VW build.

Note: External Mic is connected on P2 pin.

Table 102 *Data Rate and Frame Size for Route 26*

Route Number	Route Description	Input Ports	Output Ports	Supported Sample Rate	Supported Frame Sizes
27	2-CHIN PDM In	Internal mic PDMIN2	PDMO0 PDMO1	8K	0.5, 1, 2, 8, 10,16
				16K	0.5, 1, 2, 8, 10,16
				24K	0.5, 1, 2
				48K	0.5, 1, 2
				96K	0.5
				192K	0.5

Example:

```
0x8035 0002 (2 mSec frame size)
0x8030 0000 (8 KHz sample rate)
0x8042 0001 (0.768MHz host audio port clock frequency)
0x8032 001B (Route 27)
```

12.3.29 Route: 28

Route 28 is a 2-channel PDM In to 1-channel PDM out route. It takes its one input from internal mic and one input from external mic. This route is used for two mic NS algo.

Note: Half-band is present as part of the route. The actual sample rate is twice the internal sample rate.

Note: In this route the IA61x must be PDM slave so do not configure the IA61x as PDM master.

Note: In LA build if algorithm is not present then Internal Mic data are bypassed to output. External mic connected to data is not routed on output port.

Note: This route is not supported in VQ or VW build.

Note: External mic should be connected on P1 pin. IA61x sample external mic data on falling edge and IA61x assert data on low clock phase.

Table 103 *Data Rate and Frame Size for Route 28*

Route Number	Route Description	Input Ports	Output Ports	Supported Sample Rate	Supported Frame Sizes
28	2-CHIN PDM In	Internal mic PDMIN3	PDMO0	8K	0.5, 1, 2, 8, 10
				16K	0.5, 1, 2, 8, 10
				24K	0.5, 1, 2
				48K	0.5, 1, 2
				96K	0.5
				192K	0.5

Example:

```
0x8035 0002 (2 mSec frame size)
0x8030 0004 (48 KHz sample rate)
0x8042 0005 (3.072MHz host audio port clock frequency)
0x800C 0x1002 0x800D 0x0004 (Internal MIC clock to 3072 KHz)
0x800C 0x1302 0x800D 0x0004 (PDMIN3 clock to 3072 KHz)
0x800C 0x1402 0x800D 0x0004 (PDMO0 clock to 3072 KHz)
0x8032 001C (Route 28)
```

12.3.30 Route: 29

Route 29 is a 2-channel PDM In to 1-channel PDM out route. It takes its one input from internal mic and one input from external mic. This route can be used for two mic NS algo.

Note: Half-band is present as part of the route. The actual sample rate is twice the internal sample rate.

Note: In this route the IA61x must be PDM slave so do not configure the IA61x as PDM master.

Note: In LA build if algorithm is not present then Internal Mic data are bypassed to output. External mic connected to data is not routed on output port.

Note: This route is not supported in VQ or VW build.

Note: External mic should be connected on P1 pin. IA61x sample external mic data on rising edge and IA61x assert data on high clock phase.

Table 104 *Data Rate and Frame Size for Route 29*

Route Number	Route Description	Input Ports	Output Ports	Supported Sample Rate	Supported Frame Sizes
29	2-CHIN PDM In	Internal mic PDMIN2	PDMO1	8K	0.5, 1, 2, 8, 10
				16K	0.5, 1, 2, 8, 10
				24K	0.5, 1, 2
				48K	0.5, 1, 2
				96K	0.5
				192K	0.5

Example:

```
0x8035 0002 (2 mSec frame size)
0x8030 0004 (48 KHz sample rate)
0x8042 0005 (3.072MHz host audio port clock frequency)
0x800C 0x1002 0x800D 0x0004 (Internal MIC clock to 3072 KHz)
0x800C 0x1302 0x800D 0x0004 (PDMIN3 clock to 3072 KHz)
0x800C 0x1402 0x800D 0x0004 (PDMO0 clock to 3072 KHz)
0x8032 001D (Route 29)
```

12.3.31 Route: 31

Route 31 is a 2-channel Dual Algo Barge-In voice wake route. It takes its one input from a PCM port and another it's input from internal mic and sends AEC out to PCM port.

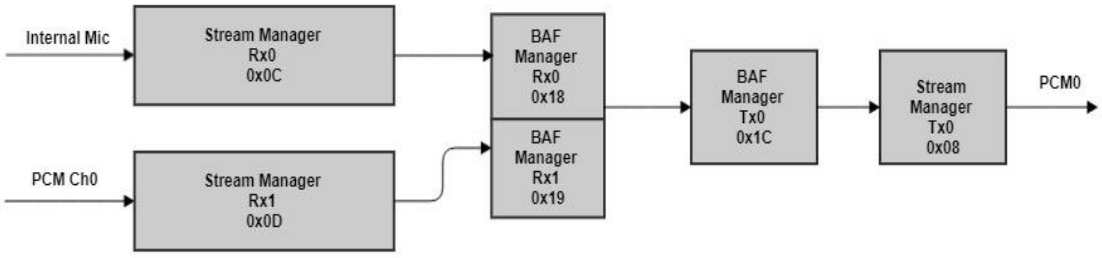

Figure 44 *Route 31*

Note: Half-band is present as part of the route. The actual sample rate is twice the internal sample rate.

Note: In this route the IA61x must be PDM slave so do not configure the IA61x as PDM master.

Note: Based on the input files either 16 KHz or 48 KHz host needs to set the port clock.

Table 105 *Data Rate and Frame Size for Route 31*

Route Number	Route Description	Input Ports	Output Ports	Supported Sample Rate	Supported Frame Sizes
31	2-CHIN, 1-CHOUT Dual algo Voice Wake Route	Internal mic PCM00	PCM0	16K	16 mSec

Example:

```
0x8035 0010 (16 mSec frame size)
0x8030 0001 (16 KHz sample rate)
0x8042 0003 (1.536MHz host audio port clock frequency)
0x802c 4000 (48 KHz port rate)
0x8032 001F (Route 31)
```

12.3.32 Route: 32

Route 32 is a 2-channel Dual Algo Barge-In voice wake route. It takes its one input from a PCM port and another input from internal mic.

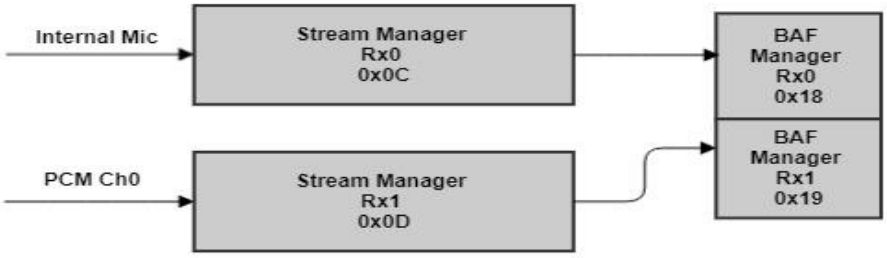

Figure 45 *Route 32*

Note: Half-band is present as part of the route. The actual sample rate is twice the internal sample rate.

Note: In this route the IA61x must be PDM slave so do not configure the IA61x as PDM master.

Note: Based on the input files either 16 KHz or 48 KHz host needs to set the port clock.

Table 106 *Data Rate and Frame Size for Route 32*

Route Number	Route Description	Input Ports	Output Ports	Supported Sample Rate	Supported Frame Sizes
32	2-CHIN Dual algo Voice Wake Route	Internal mic PCM00	NA	16K	16 mSec

Example:

```
0x8035 0010 (16 mSec frame size)
0x8030 0001 (16 KHz sample rate)
0x8042 0003 (1.536MHz host audio port clock frequency)
0x802c 4000 (48 KHz port rate)
0x8032 0020 (Route 32)
```

12.3.33 Route: 33

Route 33 is a 3-channel Dual Algo Barge-In voice wake route. It takes its 2 input from a PCM port and another input from internal mic and sends AEC out to PCM port.

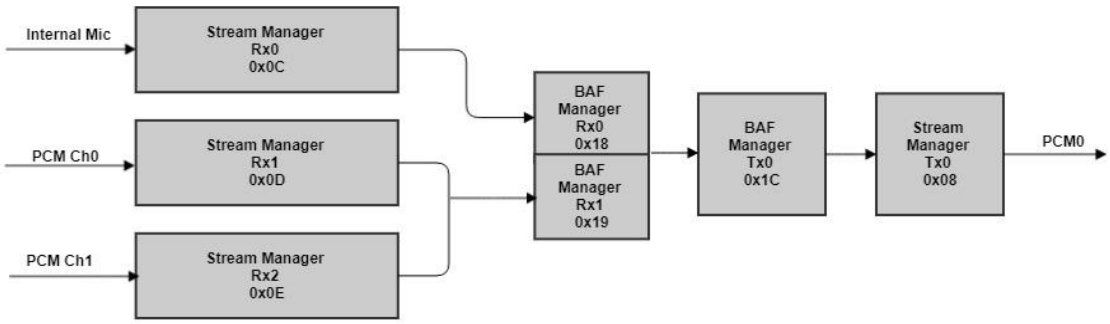

Figure 46 *Route 33*

Note: Half-band is present as part of the route. The actual sample rate is twice the internal sample rate.

Note: In this route the IA61x must be PDM slave so do not configure the IA61x as PDM master.

Note: Based on the input files either 16 KHz or 48 KHz host needs to set the port clock.

Table 107 *Data Rate and Frame Size for Route 33*

Route Number	Route Description	Input Ports	Output Ports	Supported Sample Rate	Supported Frame Sizes
33	3-CHIN, 1-CHOUT Dual algoVoice Wake Route	Internal mic PCM00 PCM01	PCM0	16K	16 ms

Example:

```
0x8035 0010 (16 mSec frame size)
0x8030 0001 (16 KHz sample rate)
0x8042 0003 (1.536MHz host audio port clock frequency)
0x802c 4000 (48 KHz port rate)
0x8032 0021 (Route 33)
```

12.3.34 Route: 34

Route 34 is a 3-channel Dual Algo Barge-In voice wake route. It takes its 2 input from a PCM port and another input from internal mic.

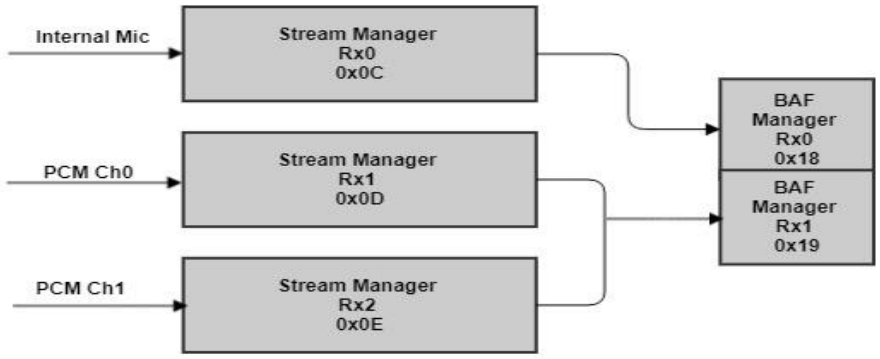

Figure 47 *Route 34*

Note: Half-band is present as part of the route. The actual sample rate is twice the internal sample rate.

Note: In this route the IA61x must be PDM slave so do not configure the IA61x as PDM master.

Note: Based on the input files either 16 KHz or 48 KHz host needs to set the port clock.

Table 108 *Data Rate and Frame Size for Route 34*

Route Number	Route Description	Input Ports	Output Ports	Supported Sample Rate	Supported Frame Sizes
34	3-CHIN Dual algoVoice Wake Route	Internal mic PCM00 PCM01	NA	16K	16 mSec

Example:

```
0x8035 0010 (16 mSec frame size)
0x8030 0001 (16 KHz sample rate)
0x8042 0003 (1.536MHz host audio port clock frequency)
0x802c 4000 (48 KHz port rate)
0x8032 0022 (Route 34)
```

Routes 35 to Route 39 are reserved for future use.

12.3.35 Route: 40

Route 40 is a Voice Wake route with an external PCM input on right channel. It takes its input from a PCM port and feeds it to the Voice Wake algo.

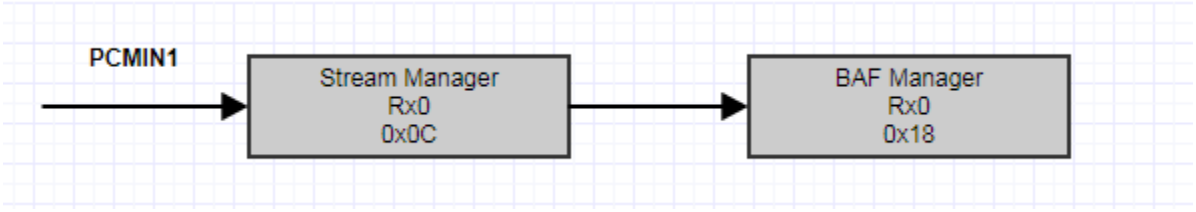

Figure 48 *Route 40*

Notes:

1. Buffering and Bursting are support in this route.
2. This is a Voice Wake route, so if the ALGO supports the desired sample rate and frame size this route should work.
3. Low power mode is only supported for UART+I2S configuration.
4. I2S data port rate API command is mandatory for this route.

Table 109 *Data Rate and Frame Size for Route 40*

Route Number	Route Description	Input Ports	Output Ports	Supported Sample Rate	Supported Frame Sizes
40	Voice Wake Route (external PCM input)	PCMIN1	NA	16K	8, 10, 15, 16

Example:

Commands to set up Voice Wake route with external PCM input:

In SBL:

```
0x8004 0001 (Set Audio Data Port as I²S/PCM)
```

In BoskoApp:

```
0x8030 0001 (16 KHz sample rate)
0x802C 1000 (16 Khz Port rate)
0x8035 0010 (16 msec frame size)
0x8032 0028 (Route 40)
```

12.3.36 Route: 41

Route 41 is a Low latency 1-channel PDM to PDM pass-through route. It takes its input from a PDM Mic port and sends its output to a PDM Output Port (Left Channel)

Notes:

1. Half-band is not present as part of the route.
2. Route has to be setup with command with response suppressed bit set.
3. Once the route is set, the IA61x goes to low power mode and to stop route host needs to toggle wake up pin
4. TIEQ routes works on a sample-by-sample basis, not on frame size boundary; so there is no need to send a Frame-Size configuration command (0x8035 0xYYYY)

Table 110 *Data Rate and Frame Size for Route 41*

Route Number	Route Description	Input Ports	Output Ports	Supported Sample Rate
41	1CH Low LatencyPass-Through	Internal Mic	PDMO0	8K
				16K
				24K
				32K
				48K
				96K

Example:

Commands to set up 1Ch Pass-Through, 48K, Internal Mic -> PDMO0, 1.536MHz host audio clock

In BoskoApp:

```
0x8030 0x0004 (48k sample rate)
0x800C 0x1002 0x800D 0x0003 (Internal MIC clock to 1.536MHz)
0x800C 0x1402 0x800D 0x0003 (PDMO0 clock to 1.536MHz)
0x8042 0x0003 (Set Audio port clock to 1.536MHz)
0x9032 0x0029 (1CH PT Low Latency Internal Mic->PDMO0)
```

12.3.37 Route: 42

Route 42 is a Low latency 1-channel PDM to PDM pass-through route. It takes its input from an Internal Mic port and sends its output to a PDM Output Port (Right Channel)

Notes:

1. Half-band is not present as part of the route.
2. Route has to be setup with command with response suppressed bit set.
3. Once the route is set, the IA61x goes to low power mode and to stop route host needs to toggle wake up pin
4. TIEQ routes works on a sample-by-sample basis, not on frame size boundary; so there is no need to send a Frame-Size configuration command (0x8035 0xYYYY)
5. PDM dual mono output is not supported in Route 42

Table 111 *Data Rate and Frame Size for Route 42*

Route Number	Route Description	Input Ports	Output Ports	Supported Sample Rate
42	1CH Low LatencyPass-Through	Internal Mic	PDMO1	8K
				16K
				24K
				32K
				48K
				96K

Example:

Commands to set up 1Ch Pass-Through, 48K, Internal Mic -> PDMO1, 3.072MHz host audio clock

In BoskoApp:

```
0x8030 0x0004 (48k sample rate)
0x800C 0x1002 0x800D 0x0003 (Internal MIC clock to 1.536MHz)
0x800C 0x1502 0x800D 0x0004 (PDMO1 clock to 3.072MHz)
0x8042 0x0005 (Set Audio port clock to 3.072MHz)
0x9032 0x002A (1CH PT Low Latency Internal Mic->PDMO1)
```

12.3.38 Route: 43

Route 43 is a Low latency 2-channel PDM to PDM pass-through route. It takes its input from an Internal Mic Internal Mic and external PDM Mic and sends both outputs to a PDM Output Port (Left Channel/Right Channel)

Notes:

1. External Mic should be connect on P2 pin
2. Half-band is not present as part of the route.
3. Route has to be setup with command with response suppressed bit set.
4. Once the route is set, the IA61x goes to low power mode and to stop route host needs to toggle wake up pin
5. TIEQ routes works on a sample-by-sample basis, not on frame size boundary; so there is no need to send a Frame-Size configuration command (0x8035 0xYYYY)
6. This route does not support I2C and SPI control interface.

Table 112 *Data Rate and Frame Size for Route 43*

Route Number	Route Description	Input Ports	Output Ports	Supported Sample Rate
43	2CH Low Latency Pass-Through	Internal Mic PDMIN2	PDMO0 PDMO1	8K
				16K
				24K
				32K
				48K
				96K

Example:

Commands to set up 2Ch Pass-Through, 48K, Internal Mic, External Mic -> PDMO0/1, 3.072MHz host audio clock

In BoskoApp:

```
0x8030 0x0004 (48k sample rate)
0x800C 0x1002 0x800D 0x0003 (Internal MIC clock to 1.536MHz)
0x800C 0x1202 0x800D 0x0003 (External MIC clock to 1.536MHz)
0x800C 0x1402 0x800D 0x0004 (PDMO0 clock to 3.072MHz)
0x800C 0x1502 0x800D 0x0004 (PDMO1 clock to 3.072MHz)
0x8042 0x0005 (Set Audio port clock to 3.072MHz)
0x9032 0x002B (2CH PT Low Latency Internal Mic->PDMO0,PDMIN2-
>PDMO1)
```

12.3.39 Route: 44

Route 44 is a Low latency 1-channel PDM to 2-channels PDM route. It designs for Ultrasound band split use case where input is received from Internal-Mic. Input audio signal is mixed audio band signal having low band (0 - 24 KHz, voice or music) and high band (24 KHz – 48 KHz, Ultrasound signal). It split mixed audio band into two different audio o/p(s):

- PDM left channel with low band signal.
- PDM right channel with high band, high band -mirrored to low band i.e. 39KHz signal is shifted to 9KHz (48KHz – 39KHz).

Notes:

1. This route only support 96 KHz sample rate due to Ultrasound, high freq. signal.
2. Half-band is not present as part of the route.
3. Route has to be setup with command with response suppressed bit set.
4. Once the route is set, the IA61x goes to low power mode and to stop route host needs to toggle wake up pin
5. TIEQ routes works on a sample-by-sample basis, not on frame size boundary; so there is no need to send a Frame-Size configuration command (0x8035 0xYYYY)

Table 113 *Data Rate and Frame Size for Route 44*

Route Number	Route Description	Input Ports	Output Ports	Supported Sample Rate
44	1CH Ultrasound Band Split	Internal Mic	PDMO0 PDMO1	96K

Example:

Commands to set up Ultrasound Band Split, 96K, Internal Mic-> PDMO0/1, 3.072MHz host audio clock

In BoskoApp:

```
0x8030 0x0005 (96k sample rate)
0x800C 0x1002 0x800D 0x0004 (Internal MIC clock to 3.072MHz)
0x800C 0x1402 0x800D 0x0004 (PDMO0 clock to 3.072MHz)
```

```
0x800C 0x1502 0x800D 0x0004 (PDMO1 clock to 3.072MHz)
0x8042 0x0005 (Set Audio port clock to 3.072MHz)
0x9032 0x002C (Ultrasound Low Latency Band Split, Internal Mic-
>PDMO0/PDMO1)
```

12.3.40 Route: 45

Route 45 is a BNE (Background Noise Estimator) route. It takes its input from a PDM Mic port and estimates background noise.

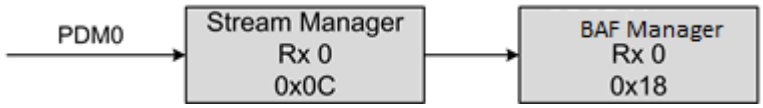

Figure 49 *Route 45*

Notes:

1. Half-band is present as part of route; the actual sample rate is twice the internal sample rate.
2. This is a BNE route, so if the ALGO supports the desired sample rate and frame size this route should work.

Table 114 *Data Rate and Frame Size for Route 45*

Route Number	Route Description	Input Ports	Output Ports	Supported Sample Rate	Supported Frame size
45	Background Noise Estimator Route	Internal Mic	NA	8K	16ms
				16K	16ms

Example:

Commands to set up BNE route, 16K Internal Mic

In BoskoApp:

```
0x8030 0x0001 (16k sample rate)

0x8035 0x0010 (Set Audio frame size to 16ms)
0x8032 0x002D (BNE route)
```

12.3.41 Route: 46

Route 46 is a dual algorithm BNE (Background Noise Estimator) and VoiceQ (VQ) route. It takes its input from a PDM Mic port, estimates background noise and does KW detection.

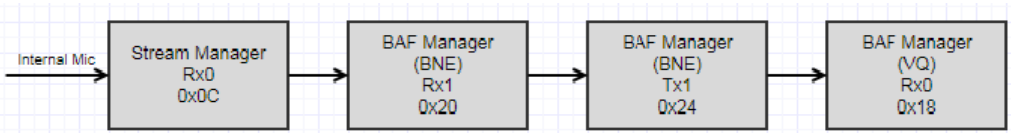

Figure 50 *Route 46*

Notes:

1. Half-band is present as part of route; the actual sample rate is twice the internal sample rate.
2. This is a BNE+VQ route, so if the ALGO supports the desired sample rate and frame size this route should work.

Table 115 *Data Rate and Frame Size for Route 46*

Route Number	Route Description	Input Ports	Output Ports	Supported Sample Rate	Supported Frame size
46	Dual Algorithm (Background Noise Estimator + VoiceQ) Route	Internal Mic	NA	8K	16ms
				16K	16ms

Example:

Commands to set up BNE+VQ route, 16K Internal Mic

In BoskoApp:

```
0x8030 0x0001 (16k sample rate)
0x8035 0x0010 (Set Audio frame size to 16ms)
0x8032 0x002E (BNE + VQ route)
```

12.3.42 Route: 47

Route 47 is a dual algorithm route. This route has two algorithms one is VQ and other algo could be beam former, which takes multimic audio samples. It takes two PDM Mic inputs, one is internal PDM mic and other is external mic which goes to beam former algo and processed audio samples are feeds to VQ algo.

Notes:

1. Half-band is present as part of route; the actual sample rate is twice the internal sample rate.
2. This route expects external mic data on rising edge (Left channel) of PDM clock.
3. External mic connects to either P1 or P2 pin, which can be configurable using SysConfig parameter. Default configuration is set for P2 pin.
4. This route is only available for UART interface.

Table 116 *Data Rate and Frame Size for Route 47*

Route Number	Route Description	Input Ports	Output Ports	Supported Sample Rate	Supported Frame size
47	Dual Algorithm (Beam Former + VoiceQ) Route	Internal Mic PDMIN2	NA	16K	16ms

Example:

Commands to set up route, 16K Internal Mic

In BoskoApp:

```
0x8030 0x0001 (16k sample rate)
0x8035 0x0010 (Set Audio frame size to 16ms)
0x8032 0x002F (2CH PDM VQ route)
```

12.3.43 Route: 48

Route 48 is a dual algorithm route. This route has two algorithms one is VQ and other algo could be beam former, which takes multimic audio samples. It takes two PDM Mic inputs, one is internal PDM mic and other is external mic which goes to beam former algo and processed audio samples are feeds to VQ algo.

Notes:

1. Half-band is present as part of route; the actual sample rate is twice the internal sample rate.
2. This Route expects external mic data on falling edge (Right channel) of PDM clock.
3. External mic connects to either P1 or P2 pin, which can be configurable using SysConfig parameter. Default configuration is set for P2 pin.
4. This route is only available for UART interface.

Table 117 *Data Rate and Frame Size for Route 48*

Route Number	Route Description	Input Ports	Output Ports	Supported Sample Rate	Supported Frame size
48	Dual Algorithm (Beam Former + VoiceQ) Route	Internal Mic PDMIN3	NA	16K	16ms

Example:

Commands to set up route, 16K Internal Mic

In BoskoApp:

```
0x8030 0x0001 (16k sample rate)
0x8035 0x0010 (Set Audio frame size to 16ms)
0x8032 0x0030 (2CH PDM VQ route)
```

12.4 Low Latency Routes

The Low latency route uses PDM as an input and PCM/PDM as an output. Firmware takes each sample from the Input Port, applies the necessary conversions and sends the data to the transmit port. As processor process every sample, the latency of this type of route is greatly reduced. In the normal case, firmware waits for the whole frame and then copies the data from PDM to PCM.

- Route 4, 41, 42 and 43 are special Low latency pass-through routes.
- Route 44 is low latency route for Ultrasound band split use case only.
- Once the host sets the low latency route, the IA61x goes to low power mode, the host should toggle wakeup pad to stop route. No other communication is allowed or responded to in this special low latency route.
- In order to stop low latency route it is recommended to follow below approach.
    - Send one dummy word of 32 bit for SPI.
    - Send one byte of 0x00 for UART.
    - Toggle wakeup pin Low-High as the IA61x wakes up on rising edge.
- It is mandatory to set the low latency route using the suppress response option. Ex: 0x90320004
- The latency of such routes would 2x sample interval, which is much faster than Route 9 or Route 10.

12.5 Stop Route

The StopRoute command stops audio data flow in the system. This is not applicable for any Low Latency routes.

Table 118 *StopRoute*

Command	Code	Value
StopRoute	0x8033	0x0000

Note that you must stop the current route before starting any new route.

12.6 Set Buffer Data Format

The IA61x supports 12-bit or 16-bit data formats to store audio samples (keyword and command buffering) into the Continuous Voice Wake buffer, and to burst the same data over either the control or the data port interface. The burst of data will always be in 16-bit format. When using 12-bit storage, the firmware will internally convert the data into 16-bits before bursting the data over either the control or the data port interface. By default, the IA61x is configured in 16-bit format, but that can be changed with this command.

Table 119 *BufferDataFormat*

Command	Code	Value
Set Buffer Data Format	0x8034	0x0001 – 12 bits 0x0002 – 16 bits (default)

12.7 Set Frame Size

The IA61x firmware allows a different audio frame size configuration for each route. For example, single clock source routes at higher rates i.e. 48K/96K/192K need 0.5ms/1ms/2ms configurations while Voice Wake routes need 8 mSec, 10 mSec, 15 mSec, or 16 mSec based on third-party algorithm requirements.

Table 120 *SetFrameSize*

Command	Code	Value
Set Frame Size	0x8035	0x0000 – 0.5ms 0x0001 – 1ms 0x0002 – 2ms 0x0008 – 8ms (default) 0x000A – 10ms 0x000F – 15ms 0x0010 – 16ms

12.8 Data Port Configuration

The IA61x Firmware allows the Host to do Bursting the Data interface. This command holds the information of I^2S configuration used for Continuous Voice Wake bursting.

This is a onetime configurable command. The Host can configure the data port immediately after binary download. The I^2S configuration set based on this command will be used for Continuous Voice Wake bursting over I^2S.

In addition, this command is added in SBL to set I2S data length for binary download over I2S. In case of different length Host has to indicate the data length using 0x802C command.

Note: This command is mandatory for Barge-In routes.

Table 121 *Config Data Port*

<table>
<tr><th>Command</th><th>Code</th><th colspan="2">Higher Value</th><th>Lower Value</th></tr>
<tr><td rowspan="3">Data Port Configuration</td><td rowspan="3">0x802C</td><td>Sample Rate Index</td><td>Number of channels</td><td>00 Default</td></tr>
<tr><td>0 = 8K
1 = 16k
2 = 24k
3 = 32k
4 = 48k
5 = 96k
6 = 192k</td><td>1 = 1channel
2 = 2 channel</td><td></td></tr>
<tr><td colspan="2">0x00</td><td>0x0F: 16 bit I2S data length
0x17: 24 bit I2S data length</td></tr>
</table>

12.9 PDM Dual Mono Output Enable

The IA61x Firmware allows the Host to enable PDM dual mono mode. When this mode is enabled, the IA61x drives both rising and falling edge PDM channels. When this mode is disabled, the IA61x only drives rising or falling edge PDM channels. This API command only valid if route has PDM output port enable.

1. Only available for PDM output port
2. This API is only valid for 1ch PDM routes
3. This API is only valid for routes which has PDMO0 port

Table 122 *PDM Dual mono output Enable*

Command	Code	Value
PDM Dual mono Output Enable	0x8044	0x0000: Dual mono output disable (Default) 0x0001: Dual mono output enable

12.10 Supported Route Config for LA builds

LA pass through routes will support the following sample rates with corresponding frame size.

Table 123 *Supported route configuration for LA builds*

Sample rate	Supported Frame Size (ms)					Max. Input Channels	Max. Output Channels
	0.5	1	2	8	10		
8000	YES	YES	YES	YES	YES	4	4
16000	YES	YES	YES	NO	NO	4	4
24000	YES	YES	YES	NO	NO	4	4
32000	YES	YES	YES	NO	NO	4	4
48000	YES	YES	YES	NO	NO	4	4
96000	YES	YES	NO	NO	NO	1	1
192000	YES	NO	NO	NO	NO	1	1

Here, YES – supported and NO – not supported

12.11 PDM Clock Decimation Ratio

Set audio port clock supports with below sample rates. PDM audio port rate is same as route configuration sample rate (0x8030 0xYYYY) for all low latency PDM->PDM configuration routes. For rest of route configurations, PDM audio port rate is always double than route configuration sample rate (0x8030 0xYYYY).

Table 124 *Supported Decimation Ratio in LA / VQ builds*

Route Sample rate	Port Clock									
	512K	768K	1024K	1536K	2048K	2400K	3072K	4608K	4800K	6144K
8000	YES	YES	YES	YES	YES	NO	NO	NO	NO	NO
16000	YES	YES	YES	YES	YES	NO	YES	NO	NO	NO
24000	NO	YES	NO	YES	NO	YES	YES	YES	YES	NO
32000	YES	YES	YES	YES	YES	NO	YES	YES	NO	NO
48000	NO	YES	NO	YES	NO	YES	YES	YES	YES	NO
96000	NO	YES	NO	YES	NO	NO	YES	YES	NO	NO
192000	NO	YES	NO	YES	NO	NO	YES	YES	NO	NO

Here, YES – supported and NO – not supported

Table 125 *Supported Decimation Ratio in LA / VQ builds for PDM Low latency route*

Route Sample rate	Port Clock									
	512K	768K	1024K	1536K	2048K	2400K	3072K	4608K	4800K	6144K
8000	YES	YES	YES	NO	NO	NO	NO	NO	NO	NO
16000	YES	YES	YES	YES	YES	NO	NO	NO	NO	NO
24000	NO	YES	NO	YES	NO	NO	YES	NO	NO	NO
32000	YES	YES	YES	YES	YES	NO	YES	NO	NO	NO
48000	NO	YES	NO	YES	NO	YES	YES	YES	YES	NO
96000	NO	YES	NO	YES	NO	YES	YES	YES	YES	NO
192000	NO	YES	NO	YES	NO	NO	YES	YES	NO	NO

Here, YES – supported and NO – not supported

Route Sample Rate: This parameter can be set by SetSampleRate (0x8030) API. Refer Section 12.2

Port Clock: There are two types of audio clocks needs to be setup.

1. Audio Port Clock: This parameter can be set by Set Audio port clock Frequency (0x8042) API section 10.5. Clock value set using this API command is being used to configure internal clock divider while route is being setup.

2. PDM Port Clock: This parameter can be set by Get/Set Device Parameter API. Refer section 11.2 and 11.2.5. PDM port clock should be same as Audio port clock. PDM port clock should be set for all ports involved in route. Number of ports involved in each route is different; refer 12.3 for port involved in specific route. Route 43 (section 12.3.36) is 2ch pass-through route has four PDM port involved, one is internal mic, one is external mic PDMIN2 and two output port PDMO0/1, so there is four port clocks needs to be configure when Route 43 is being setup.

Example 1:

Route: 1Ch PDM Passthrough Frame based route. (Internal mic to PDMO0 port)

Host Audio Port: PDM

Host Audio Clock: 1536 KHz

Host Sample Rate: 24000

In frame-based route host clock 1536 KHz and sample rate 24000 is supported, refer Table 118

Following command needs add in route setup.

```
0x8030 0002 (24k sample rate)
0x800C 0x1002 0x800D 0x0003 (Internal MIC clock to 1536 KHz)
0x800C 0x1402 0x800D 0x0003 (PDMO0 clock to 1536 KHz)
0x8042 0003 (Set Audio port clock to 1536 kHz)
```

Example 2:

Route: 1Ch PDM Passthrough Low latency route. (Internal mic to PDMO0 port)

Host Audio Port: PDM

Host Audio Clock: 1536 KHz

Host Sample Rate: 8000

In low latency route Host clock 1536 KHz and sample rate 8000 is not supported, refer Table 119.

Chapter 13: Diagnostic API Commands

These APIs provide diagnostic information. Although use of these APIs is optional, it is highly recommended that you implement them as they provide valuable information that is useful for testing and diagnosing problems related to hardware and audio.

Table 126 *Diagnostic API Commands*

Cmd name	Cmd ID	Description
Signal RMS, Get	0x8013	Get Signal RMS value on specified port(s)
Signal Peak, Get	0x8014	Get Signal peak value on specified port(s)
Output Known Signal	0x801E	Output a 1 kHz signal on specified port(s)

13.1 Get Signal RMS

The Get Signal RMS command allows the host to receive the RMS value of the specified audio/voice signal. The RMS value is computed based on the last time the Host asked for the RMS value; therefore, the first request for an RMS value will return a zero. If an RMS request is not received by the IA61x after four seconds, then the RMS system is turned off. A subsequent request for an RMS value will return a zero and re-enable the RMS system. The RMS value returned is a linear value (not dB) between 0x0000 and 0x7FFF. To convert to dBFS, apply the following formula:

$$Level\_dBFS = 20 * \log10 (Level\_linear/32768)$$

Table 127 *Get Signal RMS*

Command	Code	Value (Endpoint / SigID)
GetSigRMS	0x8013	Per Endpoint

13.2 Get Signal Peak Value

The Get Signal Peak Value command allows the host to receive the peak sample value of the specified audio/voice. The peak value is computed based on the last time the Host asked for the peak value for that mic; therefore, the first request for a peak value will return a zero. If a peak value request is not received by the IA61x after four seconds, then the peak value system is turned off. A subsequent request for a peak value will return a zero and re-enable the peak value system. The peak value returned is a linear value (not dB) between 0x0000 and 0x7FFF.

Table 128 *Get Signal Peak*

Command	Code	Value (Endpoint / SigID)
GetSigPeak	0x8014	Per Endpoint parameter

13.3 Output Known Signal

This command instructs the IA61x to continuously output a known waveform. The characteristics of the known output signal are described in **Error! Reference source not found.**. The port(s) that output the known signal is/are determined by the Audio Routing selected prior to issuing this API command.

Table 129 *Output Known Signal*

Command	Code	Value
OutputKnownSig	0x801E	0x0000 = No Signal (Default) 0x0005 = 1 kHz Sinusoid, 0 dBFS peak level

When the OutputKnownSignal is enabled, the specified tone is provided on all outputs specified in the current route. When the OutputKnownSignal is subsequently disabled, the output returns to the normal output, as specified in the route set-up.

This page intentionally left blank.

Knowles Electronics, LLC
Corporate Headquarters
1151 Maplewood Drive
Itasca, Illinois 60143
USA
Phone: (630) 250-5100
Fax: (630) 250-0575
sales@knowles.com

Rev. 4/16/18