
xCORE-AUDIO configuration guide

This document explains on how to configure an xCORE-AUDIO device. It includes guidance on how to write a configuration file and how to generate a customized firmware binary for the xCORE-AUDIO device (XHRA-2HPA).

The configuration options provide ways to customize device's USB parameters, to add support for a DAC, to control external I2C devices like PLL, etc.

The customized firmware binary is generated using the xConfigurator utility software provided with the document.

Related documents

- xCORE-AUDIO Hi-Res 2 (XHRA-2HPA) datasheet
<http://www.xmos.com/published/xhra-2hpa-tq64-c-datasheet>
- xCORE-AUDIO Hi-Res 2 firmware binary
<http://www.xmos.com/published/xhra-2hpa-firmware-binary>
- xCORE-AUDIO Hi-Res 2 DAC/HPA reference design hardware manual
<https://www.xmos.com/download/XM-008625>

1 Overview

xCORE-AUDIO Hi-Res 2 device - XHRA-2HPA is configured through two options:

- Configuration I/O pins (Straps).
- Flash configuration.

The configuration straps provide options to enable/disable few of the major features of the device, which are discussed in detail in the appendix section of XHRA-2HPA datasheet.

The flash configuration provides complete customization of the device and overwrites all the configuration straps and default behaviour of the device. This document discusses only this flash configuration and the following sections provide detailed information.

2 Flash configuration

Flash configuration is stored in the Quad SPI flash along with the device's firmware binary. Following Figure 1 shows the flash layout structure.

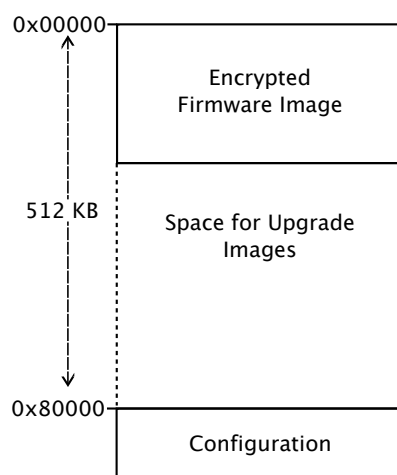


Figure 1: Quad SPI flash storing configuration

The xConfigurator tool helps to generate this entire flash binary image from a configuration file.

2.1 Configuration options

The items that can be configured through the flash configuration are as follows:

- USB identification:
 - Vendor ID
 - Product ID
 - Vendor String
 - Product String
 - Serial String
 - Language ID
- Note: USB strings can be specified in both ASCII or Unicode format.
- USB power options:
 - Maximum power (Max current required)
 - Self/Bus powered
- Other USB options
 - USB audio class 1.0 - Restrict device to UAC 1.0 version + Full speed USB.
 - USB HID inputs - For audio control (play, pause, etc)
- Audio options:
 - Supported sample rates - enable/disable individual sample rates.
 - I2S data alignment format.
- Events and their actions
 - to control GPIO and to send I2C write commands on certain events.

2.2 Write config (.cfg) file

The configuration options mentioned in the above section are specified in a text file (with .cfg extension). This text file is then parsed by xConfigurator tool to convert into the actual byte sequence corresponding to the configuration.

The template of the configuration file is provided in Appendix §A.1 This could be used as the starting point for writing the configuration.

In the config text file any content after '#' symbol in a line is considered as comment and will not be interpreted.

2.2.1 USB identification

An example USB identification configuration is shown below

```
# USB Identification

VENDOR_ID = 0x20B1

PRODUCT_ID = 0x3066

# Format of the given USB strings
#   ASCII = 0
#   UNICODE = 1
# If format is mentioned as 1 (UNICODE), then the following
# USB strings are interpreted as unicode instead of ascii.
STRING_FORMAT = 0

VENDOR_STRING = "XMOS"

PRODUCT_STRING = "xCORE-AUDIO Hi-Res 2"

SERIAL_STRING = ''

LANGUAGE_ID = 0x0409
```

The two byte Vendor ID and Product ID are specified in hexadecimal format.

The three USB strings: Vendor string, Product string and Serial string are specified as normal strings in double or single quotes and the *STRING_FORMAT* must be set to 0 denoting ASCII format. The Language ID is a two byte value denoting English (United States) as specified by USB.org consortium.

To specify the strings in Unicode format, the *STRING_FORMAT* must be set to 1 and the USB strings are provided as byte sequence in UTF-16LE format as shown below:

```
STRING_FORMAT = 1

VENDOR_STRING = "\x9B\x4F\x94\x5E\x46\x55"

PRODUCT_STRING = "\xC2\x81\xB1\x7B"

SERIAL_STRING = ''

LANGUAGE_ID = 0x0C04
```

Each byte of the string is preceded by '\x' denoting that the consecutive two digits are considered as a single byte in hexadecimal format.

In the above example, the Language ID corresponds to 'Chinese', the Vendor string corresponds to 'Vendor' and the Product string corresponds to 'Boom Box'.

2.2.2 USB power options

USB can deliver current upto 500mA for the system. The maximum power drawn from the USB host must be specified in the *MAX_POWER* option in units of 2mA. For example, following configuration shows that the system draws a maximum of 500mA (0xFA x 2).

```
# Power options
MAX_POWER = 0xFA

SELF_POWER_ENABLE = 0
```

The option *SELF_POWER_ENABLE* can be set to 1 if the system is powered by external supply instead of USB; When this option is enabled the *MAX_POWER* will be ignored and set to 0mA.

2.2.3 Other USB options

Following are the other USB configuration options.

```
# USB Audio class 1.0 enable/disable
UAC_1_ENABLE = 0

# USB HID inputs enable/disable
HID_INPUTS_ENABLE = 0
```

Setting the *UAC_1_ENABLE* to 1 will restrict the xCORE-AUDIO device to enumerate in USB full speed mode with Audio class 1.0. This option helps to develop an xCORE-AUDIO based product that works with Windows native drivers but with the trade-off of losing Hi-Res audio and DFU.

xCORE-AUDIO device supports HID inputs over GPI pins for audio control like play, pause, mute, etc. By default, this option is not enabled. It can be enabled by setting the *HID_INPUTS_ENABLE* option to 1. Technically, the xCORE-AUDIO device always enumerates an HID interface, but the HID button states are reported to the host only when this option is enabled.

2.2.4 Audio options

Following are the available audio configuration options.

```
# I2S data formats
# I2S_DEFAULT = 0
# I2S_LEFT_ALIGNED = 1
# I2S_RIGHT_ALIGNED = 2
#
I2S_DATA_FORMAT = 0

# Audio sample rates enable/disable
# Bitmap to enable/disable each supported sample rate.
#   | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
#   | 352.8K | 176.4K | 88.2K | 44.1K | 384K | 192K | 96K | 48K |
SAMPLE_RATE_BITMAP = 0b01110111
```

I2S_DATA_FORMAT - It decides the I2S format used by xCORE-AUDIO device to connect to the external DAC.

SAMPLE_RATE_BITMAP - A 8-bit value with each bit acting as a control bit to enable/disable an audio

sample rate. Setting a particular bit enables the corresponding sample rate and the xCORE-AUDIO device will report it as one of the supported sample rates to the USB host.

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
352.8K	176.4K	88.2K	44.1K	384K	192K	96K	48K

Table 1: Sample Rate Bitmap

2.2.5 Events and their actions

In addition to the static configuration options discussed in the previous sections, there are a set of configuration called events and commands that enable to do activity like GPIO toggle, I2C write etc when a particular event happens in the device.

Events are asynchronous changes in the system like sample rate change, USB connected/disconnected, audio stream started etc. Each event can be associated with a sequence of commands that get executed when that event happens.

Following shows an example events configuration:

```
# System initialization event
EVT_SYS_INIT = []

# USB connected event
EVT_USB_CONNECT = [
    GPIO(0, HIGH),
]

# USB disconnected event
EVT_USB_DISCONNECT = [
    GPIO(0, LOW),
]
```

In the above example, *EVT_USB_CONNECT* refers to the event that happens when USB host is connected to the device. It is supplied with a sequence of comma separated commands enclosed in square ([]) brackets. *GPIO(0, HIGH)* is a GPIO command (a function call) that drives GPIO_0 high. This example configuration can enable an LED connected to GPIO_0 to indicate USB connection status.

• Events

The list of supported events are as follows

Event	Description
EVT_SYS_INIT	System initialize event
EVT_USB_CONNECT	USB connected
EVT_USB_DISCONNECT	USB disconnected
EVT_STRM_START	Audio streaming started
EVT_STRM_STOP	Audio streaming stopped
EVT_SEL_48K	48KHz sample rate selected
EVT_SEL_96K	96KHz sample rate selected
EVT_SEL_192K	192KHz sample rate selected
EVT_SEL_384K	384KHz sample rate selected
EVT_SEL_44_1K	44.1KHz sample rate selected
EVT_SEL_88_2K	88.2KHz sample rate selected
EVT_SEL_176_4K	176.4KHz sample rate selected
EVT_SEL_352_8K	352.8KHz sample rate selected
EVT_DSD_ACTIVE	DSD mode active
EVT_PCM_ACTIVE	PCM mode active
EVT_SEL_DSD128	DSD128 format selected
EVT_SEL_DSD64	DSD64 format selected

Table 2: Supported Events

• Commands

Commands that can be used in the events configuration are

– GPIO command

* Used to drive a GPIO high or low:

```
GPIO(number, value)
```

'number' - the GPIO pin number (0-7),
'value' - 'HIGH' or 'LOW'.

– I2C write command

* Used to send a register write command over I2C interface:

```
I2C_WRITE(dev_addr, reg_addr, reg_value)
```

'dev_addr' - the 7-bit I2C device address.
'reg_addr' - the 8-bit internal register address of the I2C device.
'reg_value' - the 8-bit register value to be written.

– Delay command

* Used to include delay in milliseconds:

```
DELAY_MS(value)
```

'value' - delay in the range of (1 - 500) ms.

• Variables

When writing a config file, it may be good to denote certain values with names. For example,

an I2C address of 0x48 can be denoted by an identifier called 'I2C_DAC_ADDRESS'. This will increase the readability of the configuration file. The named value is a variable and it can be defined as follows:

```
I2C_DAC_ADDRESS = 0x48
I2C_DAC_CONTROL_REGISTER_ADDR = 0x01
```

Following are some of the variables used in the config file of Hi-Res 2 DAC/HPA reference platform.

```
# DAC I2C address
SABRE9018Q2C_I2C_ADDR = 0x48
# PLL I2C address
SI5351A_I2C_ADDR = 0x62

# DAC register addresses
SABRE9018Q2C_INPUT_CONFIG = 0x01 # Register 1 - Input Config
SABRE9018Q2C_GEN_SETTING = 0x07 # Register 7 - General Settings
SABRE9018Q2C_MASTER_MODE = 0x0A # Register 10 - Master Mode Control
```

```
#GPIO variables
GPIO_DAC_RST_N = 3 # GPIO_3 is connected to DAC's reset line
GPIO_LED_A = 0 # GPIO_0 is connected to LED A
GPIO_LED_B = 1 # GPIO_1 is connected to LED B
```

2.3 Generate customized firmware binary

xConfigurator is the command line tool that creates the customized firmware binary from the configuration file. We can invoke the tool as follows:

```
xConfigurator -c hi_res_2.cfg -o custom_xhra_firmware.bin
```

Where,

'hi_res_2.cfg' is the configuration file.

'custom_xhra_firmware.bin' is the configured firmware binary and it includes both the device's firmware and the configuration bytes.

This 'custom_xhra_firmware.bin' can be flashed onto the Quad SPI flash memory that is connected to the xCORE-AUDIO device.

Note: The programmer tool that flashes the Quad SPI memory must enable it for the Quad SPI mode.

3 Default behaviour

The xCORE-AUDIO device exhibits a default behaviour if there is no configuration found in the flash. This default behaviour is achieved by hard-coded configuration bytes in the device firmware. The default configuration file can be found in Appendix §A.2.

Following are the default values used:

- Vendor ID is 0x20B1 (XMOS Vendor ID)
- Product ID is 0x3066
- Vendor string is 'XMOS'
- Product string is 'xCORE-AUDIO Hi-Res 2'
- No serial string
- Maximum USB power is 500mA
- Self power disabled
- USB Audio 1.0 disabled
- HID inputs disabled
- I2S data format is default
- Sample rates only upto 192KHz

Following are the default behaviour of GPIO lines.

Event	GPIO_0 (S_AUD_STRM)
Audio stream active	1
Audio stream stopped	0

Table 3: Audio stream status

Event	GPIO_1 (S_USB_HOST)
USB connected	1
USB disconnected	0

Table 4: USB connection status

Event	GPIO_2 (S_DSD_MODE)
DSD mode active	1
PCM mode active	0

Table 5: DSD/PCM mode status

Active Sample rate	GPIO_3 (MCLK_SEL)	GPIO_5 (S_SP_1)	GPIO_4 (S_SP_0)
48 KHz	1	0	0
96 KHz	1	0	1
192 KHz	1	1	0
384 KHz	1	1	1
44.1 KHz	0	0	0
88.2 KHz	0	0	1
176.4 KHz	0	1	0
352.8 KHz	0	1	1
DSD64	0	X	0
DSD128	0	X	1

Table 6: Sample rate encoding

Note: Flash configuration completely overwrites the default behaviour of the device. If some of these default behaviour are required, the configuration file must explicitly specify them.

4 Add support for a DAC

xCORE-AUDIO device can work with almost any DAC that supports I2S and/or DSD interface. DACs that are configurable over I2C can be configured by using the events configuration mechanism. A DAC is supported by storing sequence of commands that could configure the DAC during events like initialization, sample frequency change, PCM/DSD mode change, etc.

The Hi-Res 2 DAC/HPA reference platform uses ESS SABRE9018Q2C DAC¹; its configuration file can be found in Appendix §A.3

Following configuration part shows how the ESS DAC and the clock device (Si5351A) of the Hi-Res 2 platform are initialized

```
# System initialize event
EVT_SYS_INIT = [
    GPIO(GPIO_DAC_RST_N, LOW), #// DAC in reset
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_OE_CTRL, 0xFD), # Disable CLK0 (MCLK_IN) and CLK2 (MCLK_DAC)
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_FANOUT_EN, 0xD0), # Enable Fanout of MS0
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_CLK2_CTRL, 0x69), # Setup CLK2 output
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_MS0_R0_DIV, 0x10), # Change R0 divider 2
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_MS2_R2_DIV, 0x30), # Change R2 divider 8
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_MS0_P2_LOWER, 0x00), # Write lower bits of P2
    DELAY_MS(1), # 1ms delay for MultiSynth output to settle
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_OE_CTRL, 0xF8), # Enable all the clock outputs

    GPIO(GPIO_DAC_RST_N, HIGH), # DAC out of reset
    DELAY_MS(1), # 1ms delay for DAC
    I2C_WRITE(SABRE9018Q2C_I2C_ADDR, SABRE9018Q2C_SOFT_START, 0x0A), # Set soft_start to 0: Ramp the output to
    ↪ ground
    I2C_WRITE(SABRE9018Q2C_I2C_ADDR, SABRE9018Q2C_HP_AMP_CTRL, 0x25), # Set cpw_en_oe and sw_ctrl_oe to 1
    # DAC in "Programming" state now
    I2C_WRITE(SABRE9018Q2C_I2C_ADDR, SABRE9018Q2C_INPUT_CONFIG, 0x80), # DAC input config - I2S, 32 bit
    I2C_WRITE(SABRE9018Q2C_I2C_ADDR, SABRE9018Q2C_MASTER_MODE, 0x12), # DAC in synchronous mode
    # Both channels volumes to full scale -0dBFS.
    I2C_WRITE(SABRE9018Q2C_I2C_ADDR, SABRE9018Q2C_VOLUME1, 0x06),
    I2C_WRITE(SABRE9018Q2C_I2C_ADDR, SABRE9018Q2C_VOLUME2, 0x06),
    # Get DAC to HiFi state
    I2C_WRITE(SABRE9018Q2C_I2C_ADDR, SABRE9018Q2C_HP_AMP_CTRL, 0x65), # Enable the Headphone amp
    I2C_WRITE(SABRE9018Q2C_I2C_ADDR, SABRE9018Q2C_SOFT_START, 0x8A), # Set soft_start to 1
]
```

In the above configuration:

- the DAC reset line is controlled by GPIO commands
- the Si5351A clock chip is configured to generate default clocks for the DAC (MCLK_DAC) and the xCORE-AUDIO device (MCLK_IN) by writing several registers over I2C.
- the ESS DAC is initialized in synchronous mode, input interface set to I2S 32-bit, headphone amplifier enabled and the output is ramped to ground (mute).
- a delay of 1ms is used after getting the DAC out of reset.

Similarly, the commands sequence required to change the MCLK for the DAC are stored in the sample rate change events.

¹ **Disclaimer:** XMOS uses the SABRE9018Q2C DAC chip, an ESS Technology device. "ESS Technology", "ESS Technology, Inc" "SABRE" and "ESS" are trademarks or service marks of ESS. ESS' trademarks may not be used in connection with any product or service that is not ESS', in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits ESS. ESS reserves all intellectual property rights concerning hardware design and software included in the SABRE9018Q2C DAC. Copying, distribution and any other use of hardware design and software is prohibited without written permission of ESS, except and only to the extent otherwise provided in regulations of mandatory law.

APPENDIX A - Configuration file listing

A.1 Template configuration file

```
# Copyright (c) 2015, XMOS Ltd, All rights reserved

#####
# xCORE-AUDIO Configuration template file
#####

# USB Identification

VENDOR_ID = 0x20B1

PRODUCT_ID = 0x3066

# Format of the given USB strings
#   ASCII = 0
#   UNICODE = 1
# If format is mentioned as 1 (UNICODE), then the following
# USB strings are interpreted as unicode instead of ascii.
STRING_FORMAT = 0

VENDOR_STRING = "XMOS"

PRODUCT_STRING = "xCORE-AUDIO Hi-Res 2"

SERIAL_STRING = ''

LANGUAGE_ID = 0x0409

# Power options
MAX_POWER = 0xFA

SELF_POWER_ENABLE = 0

# USB Audio class 1.0 enable/disable
UAC_1_ENABLE = 0

# USB HID inputs enable/disable
HID_INPUTS_ENABLE = 0

# I2S data formats
#   I2S_DEFAULT = 0
#   I2S_LEFT_ALIGNED = 1
#   I2S_RIGHT_ALIGNED = 2
#
I2S_DATA_FORMAT = 0

# Audio sample rates enable/disable
# Bitmap to enable/disable each supported sample rate.
#   | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
#   | 352.8K | 176.4K | 88.2K | 44.1K | 384K | 192K | 96K | 48K |

SAMPLE_RATE_BITMAP = 0b11111111

# Any variables required for writing event actions can be declared here
```

```
# Example: I2C address of the DAC can be set as a variable called
# DAC_I2C_ADDR = 0x48; For ease of writing I2C_WRITE command, the register
# addresses can also be set to variables.

# Events and their actions

# System initialize event
EVT_SYS_INIT = []

# USB connected event
EVT_USB_CONNECT = []

# USB disconnected event
EVT_USB_DISCONNECT = []

# Audio streaming started event
EVT_STRM_START = []

# Audio streaming stopped event
EVT_STRM_STOP = []

# Event on host selecting 48KHz sample rate
EVT_SEL_48K = []

# Event on host selecting 96KHz sample rate
EVT_SEL_96K = []

# Event on host selecting 192KHz sample rate
EVT_SEL_192K = []

# Event on host selecting 384KHz sample rate
EVT_SEL_384K = []

# Event on host selecting 44.1KHz sample rate
EVT_SEL_44_1K = []

# Event on host selecting 88.2KHz sample rate
EVT_SEL_88_2K = []

# Event on host selecting 176.4KHz sample rate
EVT_SEL_176_4K = []

# Event on host selecting 352.8KHz sample rate
EVT_SEL_352_8K = []

# DSD active event
EVT_DSD_ACTIVE = []

# PCM active event
EVT_PCM_ACTIVE = []

# Event on host selecting DSD128 format
EVT_SEL_DSD128 = []

# Event on host selecting DSD64 format
EVT_SEL_DSD64 = []
```

A.2 Default configuration file

```
# Copyright (c) 2015, XMOS Ltd, All rights reserved

#####
# xCORE-AUDIO default configuration file
#####

# USB Identification

VENDOR_ID = 0x20B1

PRODUCT_ID = 0x3066

# Format of the given USB strings
#   ASCII = 0
#   UNICODE = 1
# If format is mentioned as 1 (UNICODE), then the following
# USB strings are interpreted as unicode instead of ascii.
STRING_FORMAT = 0

VENDOR_STRING = "XMOS"

PRODUCT_STRING = "xCORE-AUDIO Hi-Res 2"

SERIAL_STRING = ''

LANGUAGE_ID = 0x0409

# Power options
MAX_POWER = 0xFA

SELF_POWER_ENABLE = 0

# USB Audio class 1.0 enable/disable
UAC_1_ENABLE = 0

# USB HID inputs enable/disable
HID_INPUTS_ENABLE = 0

# I2S data formats
#   I2S_DEFAULT = 0
#   I2S_LEFT_ALIGNED = 1
#   I2S_RIGHT_ALIGNED = 2
#
I2S_DATA_FORMAT = 0

# Audio sample rates enable/disable
# Bitmap to enable/disable each supported sample rate.
#   | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
#   | 352.8K | 176.4K | 88.2K | 44.1K | 384K | 192K | 96K | 48K |

SAMPLE_RATE_BITMAP = 0b01110111

# Any variables required for writing event actions can be declared here
# Example: I2C address of the DAC can be set as a variable called
# DAC_I2C_ADDR = 0x48; For ease of writing I2C_WRITE command, the register
# addresses can also be set to variables.
```

```
# Default GPIO pins
GPIO_S_AUD_STRM = 0 # GPIO_0 is S_AUD_STRM
GPIO_S_USB_HOST = 1 # GPIO_1 is S_USB_HOST
GPIO_S_DSD_MODE = 2 # GPIO_2 is S_DSD_MODE
GPIO_MCLK_SEL   = 3 # GPIO_3 is MCLK_SEL
GPIO_S_SP_0     = 4 # GPIO_4 is S_SP_0
GPIO_S_DSD_128  = 4 # GPIO_4 is also acts as S_DSD_128
GPIO_S_SP_1     = 5 # GPIO_5 is S_SP_1

# Events and their actions

# System initialize event
EVT_SYS_INIT = []

# USB connected event
EVT_USB_CONNECT = [
    GPIO(GPIO_S_USB_HOST, HIGH),
]

# USB disconnected event
EVT_USB_DISCONNECT = [
    GPIO(GPIO_S_USB_HOST, LOW),
]

# Audio streaming started event
EVT_STRM_START = [
    GPIO(GPIO_S_AUD_STRM, HIGH),
]

# Audio streaming stopped event
EVT_STRM_STOP = [
    GPIO(GPIO_S_AUD_STRM, LOW),
]

# Event on host selecting 48KHz sample rate
EVT_SEL_48K = [
    GPIO(GPIO_MCLK_SEL, HIGH),
    GPIO(GPIO_S_SP_1, LOW),
    GPIO(GPIO_S_SP_0, LOW),
]

# Event on host selecting 96KHz sample rate
EVT_SEL_96K = [
    GPIO(GPIO_MCLK_SEL, HIGH),
    GPIO(GPIO_S_SP_1, LOW),
    GPIO(GPIO_S_SP_0, HIGH),
]

# Event on host selecting 192KHz sample rate
EVT_SEL_192K = [
    GPIO(GPIO_MCLK_SEL, HIGH),
    GPIO(GPIO_S_SP_1, HIGH),
    GPIO(GPIO_S_SP_0, LOW),
]

# Event on host selecting 384KHz sample rate
```

```
EVT_SEL_384K = [  
    GPIO(GPIO_MCLK_SEL, HIGH),  
    GPIO(GPIO_S_SP_1, HIGH),  
    GPIO(GPIO_S_SP_0, HIGH),  
]  
  
# Event on host selecting 44.1KHz sample rate  
EVT_SEL_44_1K = [  
    GPIO(GPIO_MCLK_SEL, LOW),  
    GPIO(GPIO_S_SP_1, LOW),  
    GPIO(GPIO_S_SP_0, LOW),  
]  
  
# Event on host selecting 88.2KHz sample rate  
EVT_SEL_88_2K = [  
    GPIO(GPIO_MCLK_SEL, LOW),  
    GPIO(GPIO_S_SP_1, LOW),  
    GPIO(GPIO_S_SP_0, HIGH),  
]  
  
# Event on host selecting 176.4KHz sample rate  
EVT_SEL_176_4K = [  
    GPIO(GPIO_MCLK_SEL, LOW),  
    GPIO(GPIO_S_SP_1, HIGH),  
    GPIO(GPIO_S_SP_0, LOW),  
]  
  
# Event on host selecting 352.8KHz sample rate  
EVT_SEL_352_8K = [  
    GPIO(GPIO_MCLK_SEL, LOW),  
    GPIO(GPIO_S_SP_1, HIGH),  
    GPIO(GPIO_S_SP_0, HIGH),  
]  
  
# DSD active event  
EVT_DSD_ACTIVE = [  
    GPIO(GPIO_S_DSD_MODE, HIGH),  
]  
  
# PCM active event  
EVT_PCM_ACTIVE = [  
    GPIO(GPIO_S_DSD_MODE, LOW),  
]  
  
# Event on host selecting DSD128 format  
EVT_SEL_DSD128 = [  
    GPIO(GPIO_S_DSD_128, HIGH),  
]  
  
# Event on host selecting DSD64 format  
EVT_SEL_DSD64 = [  
    GPIO(GPIO_S_DSD_128, LOW),  
]
```


A.3 xCORE-AUDIO Hi-Res 2 DAC/HPA reference platform configuration file

```
#####
# Copyright (c) 2015, XMOS Ltd, All rights reserved

# Disclaimer:
# XMOS uses the SABRE9018Q2C DAC chip, an ESS Technology device.
# "ESS Technology", "ESS Technology, Inc" "SABRE" and "ESS" are
# trademarks or service marks of ESS. ESS' trademarks may not be
# used in connection with any product or service that is not ESS',
# in any manner that is likely to cause confusion among customers,
# or in any manner that disparages or discredits ESS. ESS reserves
# all intellectual property rights concerning hardware design and
# software included in the SABRE9018Q2C DAC. Copying, distribution
# and any other use of hardware design and software is prohibited
# without written permission of ESS, except and only to the extent
# otherwise provided in regulations of mandatory law.

#####
# xCORE-AUDIO Hi-Res 2 DAC/HPA reference platform configuration file
#####

# USB Identification

VENDOR_ID = 0x20B1

PRODUCT_ID = 0x3066

# Format of the given USB strings
#   ASCII = 0
#   UNICODE = 1
# If format is mentioned as 1 (UNICODE), then the following
# USB strings are interpreted as unicode instead of ascii.
STRING_FORMAT = 0

VENDOR_STRING = "XMOS"

PRODUCT_STRING = "xCORE-AUDIO Hi-Res 2"

SERIAL_STRING = ''

LANGUAGE_ID = 0x0409

# Power options

MAX_POWER = 0xFA

SELF_POWER_ENABLE = 0

# USB Audio class 1.0 enable/disable

UAC_1_ENABLE = 0

# USB HID inputs enable/disable

HID_INPUTS_ENABLE = 0

# I2S data formats
#   I2S_DEFAULT = 0
#   I2S_LEFT_ALIGNED = 1
#   I2S_RIGHT_ALIGNED = 2
#
I2S_DATA_FORMAT = 0

# Audio sample rates enable/disable
# Bitmap to enable/disable each supported sample rate.
#   | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
#   | 352.8K | 176.4K | 88.2K | 44.1K | 384K | 192K | 96K | 48K |

SAMPLE_RATE_BITMAP = 0b11111111

# Any variables required for writing event actions can be declared here
# Example: I2C address of the DAC can be set as a variable called
# DAC_I2C_ADDR = 0x48; For ease of writing I2C_WRITE command, the register
# addresses can also be set to variables.
```

```
# DAC I2C address
SABRE9018Q2C_I2C_ADDR = 0x48
# PLL I2C address
SI5351A_I2C_ADDR = 0x62

# DAC register addresses
SABRE9018Q2C_INPUT_CONFIG = 0x01 # Register 1 - Input Config
SABRE9018Q2C_GEN_SETTING = 0x07 # Register 7 - General Settings
SABRE9018Q2C_MASTER_MODE = 0x0A # Register 10 - Master Mode Control

SABRE9018Q2C_SOFT_START = 0x0E # Register 14 - Soft Start Settings
SABRE9018Q2C_VOLUME1 = 0x0F # Register 15 - Volume 1
SABRE9018Q2C_VOLUME2 = 0x10 # Register 16 - Volume 2

SABRE9018Q2C_CHIP_STATUS = 0x40 # Register 64 - Chip Status
SABRE9018Q2C_HP_AMP_CTRL = 0x2A # Register 42 - Headphone Amp Control

# PLL register addresses
SI5351A_OE_CTRL = 0x03 # Register 3 - Output Enable Control
SI5351A_FANOUT_EN = 0xBB # Register 187 - Fanout Enable Control

SI5351A_MS0_R0_DIV = 0x2C # Register 44 - Multisynth0 Parameters - R0_DIV[2:0], MS0_DIVBY4[1:0] and MS0_P1
↳ [17:16];
SI5351A_MS2_R2_DIV = 0x3C # Register 60 - Multisynth2 Parameters - R2_DIV[2:0], MS2_DIVBY4[1:0] and MS2_P1
↳ [17:16];

SI5351A_CLK0_CTRL = 0x10 # Register 16 - CLK0 Control
SI5351A_MS0_P1_UPPER = 0x2D # Register 45 - Multisynth0 Parameters - MS0_P1[15:8]
SI5351A_MS0_P2_LOWER = 0x31 # Register 49 - Multisynth0 Parameters - MS0_P2[7:0]

SI5351A_CLK2_CTRL = 0x12 # Register 18 - CLK2 Control
SI5351A_MS2_P1_UPPER = 0x3D # Register 61 - Multisynth2 Parameters - MS2_P1[15:8]
SI5351A_MS2_P2_LOWER = 0x41 # Register 65 - Multisynth2 Parameters - MS2_P2[7:0]

#GPIO variables
GPIO_DAC_RST_N = 3 # GPIO_3 is connected to DAC's reset line
GPIO_LED_A = 0 # GPIO_0 is connected to LED A
GPIO_LED_B = 1 # GPIO_1 is connected to LED B

# Events and their actions

# System initialize event
EVT_SYS_INIT = [
    GPIO(GPIO_DAC_RST_N, LOW), #// DAC in reset
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_OE_CTRL, 0xFD), # Disable CLK0 (MCLK_IN) and CLK2 (MCLK_DAC)
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_FANOUT_EN, 0xD0), # Enable Fanout of MS0
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_CLK2_CTRL, 0x69), # Setup CLK2 output
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_MS0_R0_DIV, 0x10), # Change R0 divider 2
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_MS2_R2_DIV, 0x30), # Change R2 divider 8
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_MS0_P2_LOWER, 0x00), # Write lower bits of P2
    DELAY_MS(1), # 1ms delay for MultiSynth output to settle
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_OE_CTRL, 0xF8), # Enable all the clock outputs

    GPIO(GPIO_DAC_RST_N, HIGH), # DAC out of reset
    DELAY_MS(1), # 1ms delay for DAC
    I2C_WRITE(SABRE9018Q2C_I2C_ADDR, SABRE9018Q2C_SOFT_START, 0x0A), # Set soft_start to 0: Ramp the output to
    ↳ ground
    I2C_WRITE(SABRE9018Q2C_I2C_ADDR, SABRE9018Q2C_HP_AMP_CTRL, 0x25), # Set cpw_en_oe and sw_ctrl_oe to 1
    # DAC in "Programming" state now
    I2C_WRITE(SABRE9018Q2C_I2C_ADDR, SABRE9018Q2C_INPUT_CONFIG, 0x80), # DAC input config - I2S, 32 bit
    I2C_WRITE(SABRE9018Q2C_I2C_ADDR, SABRE9018Q2C_MASTER_MODE, 0x12), # DAC in synchronous mode
    # Both channels volumes to full scale -0dBFS.
    I2C_WRITE(SABRE9018Q2C_I2C_ADDR, SABRE9018Q2C_VOLUME1, 0x06),
    I2C_WRITE(SABRE9018Q2C_I2C_ADDR, SABRE9018Q2C_VOLUME2, 0x06),
    # Get DAC to HiFi state
    I2C_WRITE(SABRE9018Q2C_I2C_ADDR, SABRE9018Q2C_HP_AMP_CTRL, 0x65), # Enable the Headphone amp
    I2C_WRITE(SABRE9018Q2C_I2C_ADDR, SABRE9018Q2C_SOFT_START, 0x8A), # Set soft_start to 1
]

# USB connected event
EVT_USB_CONNECT = [
    GPIO(GPIO_LED_A, HIGH),
]
```

```
# USB disconnected event
EVT_USB_DISCONNECT = [
    GPIO(GPIO_LED_A, LOW),
]

# Audio streaming started event
EVT_STRM_START = [
    GPIO(GPIO_LED_B, HIGH),
]

# Audio streaming stopped event
EVT_STRM_STOP = [
    GPIO(GPIO_LED_B, LOW),
]

# Event on host selecting 48KHz sample rate
EVT_SEL_48K = [
    I2C_WRITE(SABRE9018Q2C_I2C_ADDR, SABRE9018Q2C_SOFT_START, 0x0A), # Get DAC to "Programming" mode
    DELAY_MS(100),
    I2C_WRITE(SABRE9018Q2C_I2C_ADDR, SABRE9018Q2C_HP_AMP_CTRL, 0x25), # Set amp_en to '0'
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_OE_CTRL, 0xFD), # Disable the CLK0 and CLK2 outputs
    # MCLK = 24.576MHz (48,96,192KHz)
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_CLK0_CTRL, 0x4D), # PLLA - source for MS0, MS0 - source of CLK0, 4mA
    ↪ drive strength
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_MS0_P1_UPPER, 0x05), # Set P1 divider
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_MS0_P2_LOWER, 0x00), # Write lower bits of P2 divider
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_MS2_R2_DIV, 0x30), # Change R2 divider to 8
    DELAY_MS(1), # Delay for MultiSynth output to settle
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_OE_CTRL, 0xF8), # Enable all the clock outputs
    I2C_WRITE(SABRE9018Q2C_I2C_ADDR, SABRE9018Q2C_INPUT_CONFIG, 0x80), # DAC input config - I2S - 32 bit
    # Get DAC to HiFi state
    I2C_WRITE(SABRE9018Q2C_I2C_ADDR, SABRE9018Q2C_HP_AMP_CTRL, 0x65), # Enable the Headphone amp
    I2C_WRITE(SABRE9018Q2C_I2C_ADDR, SABRE9018Q2C_SOFT_START, 0x8A), # Set soft_start to 1
]

# Event on host selecting 96KHz sample rate
EVT_SEL_96K = [
    I2C_WRITE(SABRE9018Q2C_I2C_ADDR, SABRE9018Q2C_SOFT_START, 0x0A), # Get DAC to "Programming" mode
    DELAY_MS(100),
    I2C_WRITE(SABRE9018Q2C_I2C_ADDR, SABRE9018Q2C_HP_AMP_CTRL, 0x25), # Set amp_en to '0'
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_OE_CTRL, 0xFD), # Disable the CLK0 and CLK2 outputs
    # MCLK = 24.576MHz (48,96,192KHz)
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_CLK0_CTRL, 0x4D), # PLLA - source for MS0, MS0 - source of CLK0, 4mA
    ↪ drive strength
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_MS0_P1_UPPER, 0x05), # Set P1 divider
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_MS0_P2_LOWER, 0x00), # Write lower bits of P2 divider
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_MS2_R2_DIV, 0x20), # Change R2 divider to 4
    DELAY_MS(1), # Delay for MultiSynth output to settle
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_OE_CTRL, 0xF8), # Enable all the clock outputs
    I2C_WRITE(SABRE9018Q2C_I2C_ADDR, SABRE9018Q2C_INPUT_CONFIG, 0x80), # DAC input config - I2S - 32 bit
    # Get DAC to HiFi state
    I2C_WRITE(SABRE9018Q2C_I2C_ADDR, SABRE9018Q2C_HP_AMP_CTRL, 0x65), # Enable the Headphone amp
    I2C_WRITE(SABRE9018Q2C_I2C_ADDR, SABRE9018Q2C_SOFT_START, 0x8A), # Set soft_start to 1
]

# Event on host selecting 192KHz sample rate
EVT_SEL_192K = [
    I2C_WRITE(SABRE9018Q2C_I2C_ADDR, SABRE9018Q2C_SOFT_START, 0x0A), # Get DAC to "Programming" mode
    DELAY_MS(100),
    I2C_WRITE(SABRE9018Q2C_I2C_ADDR, SABRE9018Q2C_HP_AMP_CTRL, 0x25), # Set amp_en to '0'
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_OE_CTRL, 0xFD), # Disable the CLK0 and CLK2 outputs
    # MCLK = 24.576MHz (48,96,192KHz)
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_CLK0_CTRL, 0x4D), # PLLA - source for MS0, MS0 - source of CLK0, 4mA
    ↪ drive strength
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_MS0_P1_UPPER, 0x05), # Set P1 divider
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_MS0_P2_LOWER, 0x00), # Write lower bits of P2 divider
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_MS2_R2_DIV, 0x10), # Change R2 divider to 2
    DELAY_MS(1), # Delay for MultiSynth output to settle
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_OE_CTRL, 0xF8), # Enable all the clock outputs
    I2C_WRITE(SABRE9018Q2C_I2C_ADDR, SABRE9018Q2C_INPUT_CONFIG, 0x80), # DAC input config - I2S - 32 bit
    # Get DAC to HiFi state
    I2C_WRITE(SABRE9018Q2C_I2C_ADDR, SABRE9018Q2C_HP_AMP_CTRL, 0x65), # Enable the Headphone amp
    I2C_WRITE(SABRE9018Q2C_I2C_ADDR, SABRE9018Q2C_SOFT_START, 0x8A), # Set soft_start to 1
]
```

```
# Event on host selecting 384KHz sample rate
EVT_SEL_384K = [
    I2C_WRITE(SABRE9018Q2C_I2C_ADDR, SABRE9018Q2C_SOFT_START, 0x0A), # Get DAC to "Programming" mode
    DELAY_MS(100),
    I2C_WRITE(SABRE9018Q2C_I2C_ADDR, SABRE9018Q2C_HP_AMP_CTRL, 0x25), # Set amp_en to '0'
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_OE_CTRL, 0xFD), # Disable the CLK0 and CLK2 outputs
    # MCLK = 24.576MHz (48,96,192KHz)
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_CLK0_CTRL, 0x4D), # PLLA - source for MS0, MS0 - source of CLK0, 4mA
    ↪ drive strength
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_MS0_P1_UPPER, 0x05), # Set P1 divider
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_MS0_P2_LOWER, 0x00), # Write lower bits of P2 divider
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_MS2_R2_DIV, 0x00), # Change R2 divider to 1
    DELAY_MS(1), # Delay for MultiSynth output to settle
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_OE_CTRL, 0xF8), # Enable all the clock outputs
    I2C_WRITE(SABRE9018Q2C_I2C_ADDR, SABRE9018Q2C_INPUT_CONFIG, 0x80), # DAC input config - I2S - 32 bit
    # Get DAC to HiFi state
    I2C_WRITE(SABRE9018Q2C_I2C_ADDR, SABRE9018Q2C_HP_AMP_CTRL, 0x65), # Enable the Headphone amp
    I2C_WRITE(SABRE9018Q2C_I2C_ADDR, SABRE9018Q2C_SOFT_START, 0x8A), # Set soft_start to 1
]

# Event on host selecting 44.1KHz sample rate
EVT_SEL_44_1K = [
    I2C_WRITE(SABRE9018Q2C_I2C_ADDR, SABRE9018Q2C_SOFT_START, 0x0A), # Get DAC to "Programming" mode
    DELAY_MS(100),
    I2C_WRITE(SABRE9018Q2C_I2C_ADDR, SABRE9018Q2C_HP_AMP_CTRL, 0x25), # Set amp_en to '0'
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_OE_CTRL, 0xFD), # Disable the CLK0 and CLK2 outputs
    # MCLK = 22.5792MHz (44.1,88.2,176.4kHz)
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_CLK0_CTRL, 0x6D), # PLLB - source for MS0, MS0 - source of CLK0, 4mA
    ↪ drive strength
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_MS0_P1_UPPER, 0x07), # Set P1 divider
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_MS0_P2_LOWER, 0x00), # Write lower bits of P2 divider
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_MS2_R2_DIV, 0x30), # Change R2 divider to 8
    DELAY_MS(1), # Delay for MultiSynth output to settle
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_OE_CTRL, 0xF8), # Enable all the clock outputs
    I2C_WRITE(SABRE9018Q2C_I2C_ADDR, SABRE9018Q2C_INPUT_CONFIG, 0x80), # DAC input config - I2S - 32 bit
    # Get DAC to HiFi state
    I2C_WRITE(SABRE9018Q2C_I2C_ADDR, SABRE9018Q2C_HP_AMP_CTRL, 0x65), # Enable the Headphone amp
    I2C_WRITE(SABRE9018Q2C_I2C_ADDR, SABRE9018Q2C_SOFT_START, 0x8A), # Set soft_start to 1
]

# Event on host selecting 88.2KHz sample rate
EVT_SEL_88_2K = [
    I2C_WRITE(SABRE9018Q2C_I2C_ADDR, SABRE9018Q2C_SOFT_START, 0x0A), # Get DAC to "Programming" mode
    DELAY_MS(100),
    I2C_WRITE(SABRE9018Q2C_I2C_ADDR, SABRE9018Q2C_HP_AMP_CTRL, 0x25), # Set amp_en to '0'
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_OE_CTRL, 0xFD), # Disable the CLK0 and CLK2 outputs
    # MCLK = 24.576MHz (48,96,192KHz)
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_CLK0_CTRL, 0x6D), # PLLB - source for MS0, MS0 - source of CLK0, 4mA
    ↪ drive strength
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_MS0_P1_UPPER, 0x07), # Set P1 divider
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_MS0_P2_LOWER, 0x00), # Write lower bits of P2 divider
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_MS2_R2_DIV, 0x20), # Change R2 divider to 4
    DELAY_MS(1), # Delay for MultiSynth output to settle
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_OE_CTRL, 0xF8), # Enable all the clock outputs
    I2C_WRITE(SABRE9018Q2C_I2C_ADDR, SABRE9018Q2C_INPUT_CONFIG, 0x80), # DAC input config - I2S - 32 bit
    # Get DAC to HiFi state
    I2C_WRITE(SABRE9018Q2C_I2C_ADDR, SABRE9018Q2C_HP_AMP_CTRL, 0x65), # Enable the Headphone amp
    I2C_WRITE(SABRE9018Q2C_I2C_ADDR, SABRE9018Q2C_SOFT_START, 0x8A), # Set soft_start to 1
]

# Event on host selecting 176.4KHz sample rate
EVT_SEL_176_4K = [
    I2C_WRITE(SABRE9018Q2C_I2C_ADDR, SABRE9018Q2C_SOFT_START, 0x0A), # Get DAC to "Programming" mode
    DELAY_MS(100),
    I2C_WRITE(SABRE9018Q2C_I2C_ADDR, SABRE9018Q2C_HP_AMP_CTRL, 0x25), # Set amp_en to '0'
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_OE_CTRL, 0xFD), # Disable the CLK0 and CLK2 outputs
    # MCLK = 24.576MHz (48,96,192KHz)
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_CLK0_CTRL, 0x6D), # PLLB - source for MS0, MS0 - source of CLK0, 4mA
    ↪ drive strength
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_MS0_P1_UPPER, 0x07), # Set P1 divider
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_MS0_P2_LOWER, 0x00), # Write lower bits of P2 divider
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_MS2_R2_DIV, 0x10), # Change R2 divider to 2
    DELAY_MS(1), # Delay for MultiSynth output to settle
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_OE_CTRL, 0xF8), # Enable all the clock outputs
    I2C_WRITE(SABRE9018Q2C_I2C_ADDR, SABRE9018Q2C_INPUT_CONFIG, 0x80), # DAC input config - I2S - 32 bit
]
```

```
# Get DAC to HiFi state
I2C_WRITE(SABRE9018Q2C_I2C_ADDR, SABRE9018Q2C_HP_AMP_CTRL, 0x65), # Enable the Headphone amp
I2C_WRITE(SABRE9018Q2C_I2C_ADDR, SABRE9018Q2C_SOFT_START, 0x8A), # Set soft_start to 1
]

# Event on host selecting 352.8KHz sample rate
EVT_SEL_352_8K = [
    I2C_WRITE(SABRE9018Q2C_I2C_ADDR, SABRE9018Q2C_SOFT_START, 0x0A), # Get DAC to "Programming" mode
    DELAY_MS(100),
    I2C_WRITE(SABRE9018Q2C_I2C_ADDR, SABRE9018Q2C_HP_AMP_CTRL, 0x25), # Set amp_en to '0'
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_OE_CTRL, 0xFD), # Disable the CLK0 and CLK2 outputs
    # MCLK = 24.576MHz (48,96,192KHz)
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_CLK0_CTRL, 0x6D), # PLLB - source for MS0, MS0 - source of CLK0, 4mA
    ↪ drive strength
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_MS0_P1_UPPER, 0x07), # Set P1 divider
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_MS0_P2_LOWER, 0x00), # Write lower bits of P2 divider
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_MS2_R2_DIV, 0x00), # Change R2 divider to 1
    DELAY_MS(1), # Delay for MultiSynth output to settle
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_OE_CTRL, 0xF8), # Enable all the clock outputs
    I2C_WRITE(SABRE9018Q2C_I2C_ADDR, SABRE9018Q2C_INPUT_CONFIG, 0x80), # DAC input config - I2S - 32 bit
    # Get DAC to HiFi state
    I2C_WRITE(SABRE9018Q2C_I2C_ADDR, SABRE9018Q2C_HP_AMP_CTRL, 0x65), # Enable the Headphone amp
    I2C_WRITE(SABRE9018Q2C_I2C_ADDR, SABRE9018Q2C_SOFT_START, 0x8A), # Set soft_start to 1
]

# DSD active event
EVT_DSD_ACTIVE = [
    I2C_WRITE(SABRE9018Q2C_I2C_ADDR, SABRE9018Q2C_SOFT_START, 0x0A), # Get DAC to "Programming" mode
    DELAY_MS(100),
    I2C_WRITE(SABRE9018Q2C_I2C_ADDR, SABRE9018Q2C_HP_AMP_CTRL, 0x25), # Set amp_en to '0'
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_OE_CTRL, 0xFD), # Disable the CLK0 and CLK2 outputs
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_CLK0_CTRL, 0x6D), # PLLB - source for MS0, MS0 - source of CLK0, 4mA
    ↪ drive strength
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_MS0_P1_UPPER, 0x07), # Set P1 divider
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_MS0_P2_LOWER, 0x00), # Write lower bits of P2 divider
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_MS2_R2_DIV, 0x00), # Change R2 divider to 1
    DELAY_MS(1), # Delay for MultiSynth output to settle
    I2C_WRITE(SI5351A_I2C_ADDR, SI5351A_OE_CTRL, 0xF8), # Enable all the clock outputs
    I2C_WRITE(SABRE9018Q2C_I2C_ADDR, SABRE9018Q2C_INPUT_CONFIG, 0x83), # DAC input config - DSD
    # Get DAC to HiFi state
    I2C_WRITE(SABRE9018Q2C_I2C_ADDR, SABRE9018Q2C_HP_AMP_CTRL, 0x65), # Enable the Headphone amp
    I2C_WRITE(SABRE9018Q2C_I2C_ADDR, SABRE9018Q2C_SOFT_START, 0x8A), # Set soft_start to 1
]

# PCM active event
EVT_PCM_ACTIVE = []

# Event on host selecting DSD128 format
EVT_SEL_DSD128 = []

# Event on host selecting DSD64 format
EVT_SEL_DSD64 = []
```