AN12762

Audio Player In HiFi4

Rev. 0 — 02 March 2020

Application Note

Contents

1 Introduction..... 1

2 HiFi 4 Development in i.MX RT600..... 1

3 Software Architecture..... 1

4 Implementation..... 2

5 Revision history..... 5

1 Introduction

This application note describes how to develop an audio player with NXP i.MX RT600.

The i.MX RT600 features an Arm Cortex-M33 CPU combined with Cadence Xtensa HiFi4 Audio DSP CPU. The Arm Cortex-M33 is a next generation core based on the ARMv8-M architecture that offers system enhancements, such as Arm TrustZone security, single-cycle digital signal processing, and tight-coupled coprocessor interface.

The Cadence Xtensa HiFi 4 Audio DSP engine is a highly optimized audio processor designed especially for efficient execution of audio and voice codecs and pre- and post-processing modules.See Section “2”

2 HiFi 4 Development in i.MX RT600

The highlights of HiFi 4 DSP are

- Up to 600 MHz
- Four 32x32-bits MACs per cycle
- Some support for 72-bit accumulators
- Limited ability to support eight 32x16-bit MACs
- Dual issue two 64-bit loads per cycle
- Four single-precision IEEE floating point MACs per cycle

3 Software Architecture

The “audio player” software architecture is shown in Figure 1 .

Arm Cortex M33 and Cadence HiFi4 DSP are running each program. At Arm Cortex M33 side, “Audio Player” application is based on i.MX RT600 MCUXpresso SDK, using FreeRTOS; At Cadence HiFi4 DSP side, “Audio Player” DSP handler is designed to handle requests from Arm, using XOS, a RTOS from Cadence. Besides XOS, “Audio Player” demo also uses “Nature DSP”, “Audio Framework” and “MP3 decoder” from Cadence, all software is free for i.MX RT600 users. As to open source software, LibFlac from https://xiph.org/flac/ is used from FLAC decoding. LibFlac is under BSD license.

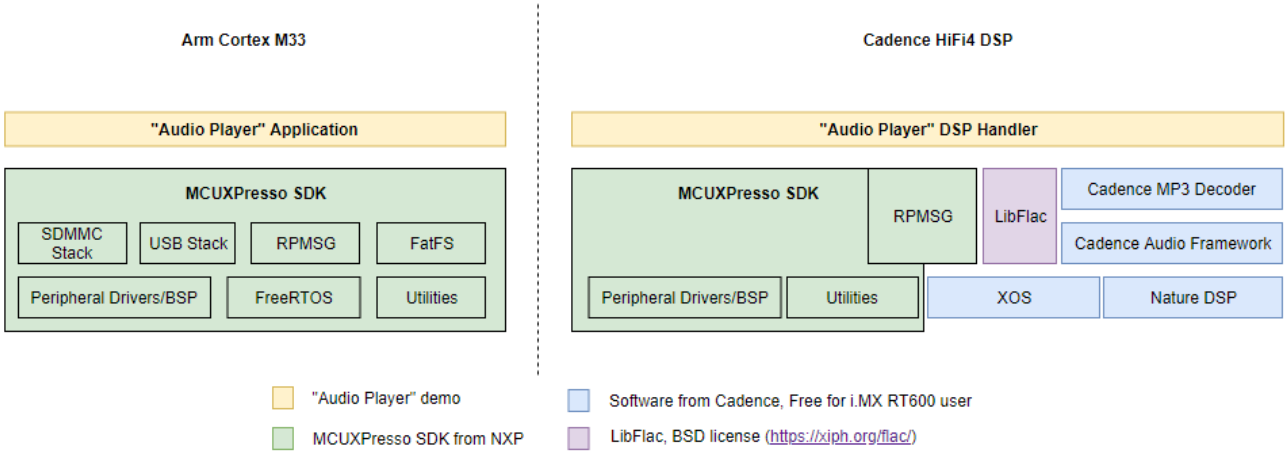

Figure 1. "Audio Player" software archtechture

Figure 1 shows software used in "Audio Player" demo application.

4 Implementation

4.1 Device initialization

In "Audio Player" demo, HS USB, USDHC, and HiFi 4 DSP are used in Arm firmware. HS USB mess storage with file system support is for media file storing; SDCard with file system support is for media retrieving. DSP is for audio decoding and playback. Thus, HS USB, SDCard, and DSP should be configured and initialized when Arm firmware is booted.

Before USB device is configured, HS USB RAM should be powered, peripheral clock should be attached and enabled, and peripheral reset should be released. HS USB PHY uses external OSC as clock source, the internal USB PLL can boost the frequency to 480 MHz.

Similar to HS USB device initialization, SDHC RAM, clock, and reset should be ready before SD host controller is initialized. Aux0 PLL clock@396MHz is attached for SDIO controller. SDMMC stack must be initialized in an RTOS task context. File system must be initialized after SDMMC stack is initialized as file system is using SD card for its physical media.

DSP is initialized in below procedures:

1. Enable DSP PLL clock
2. Enable DSP TCM and Cache power, enable DSP clock, release DSP reset
3. Arm copy DSP firmware to DSP TCM RAM and system RAM for execution
4. Initialize message unit for dual-core communication
5. Start DSP execution by clear SYSCTL0->DSPSTALL

Audio playback is implemented at DSP side. DMA1, I2S, and audio codec are initialized in DSP firmware.

Both Arm firmware and DSP firmware are using MCUXpresso SDK for peripheral initialization.

4.2 ARM-DSP communication

The Message Unit (MU), Inter-Processor Interrupt and shared RAM support ARM-DSP communication. AN12749 elaborates the mechanism and implementation of RPMSG, which is software middleware based on MU and Inter-Processor Interrupt for inter-processor communication.

In the demo, RPMSG enables the command channels between Arm and DSP for audio player control. Both Arm and DSP can initial the communication, application messages are designed for audio player creation and control, these commands are listed in Table 1.

Table 1. Audio player creation and control commands

	Direction	Usage
SRTM_REQUEST_ADEV_OPEN	Arm->DSP	Arm sends commands to DSP to set up audio player, see 4.3 DSP Audio Player Creation
SRTM_REQUEST_COMP_CREATE	Arm->DSP	
SRTM_REQUEST_COMP_DELETE	Arm->DSP	
SRTM_REQUEST_ADEV_CLOSE	Arm->DSP	
SRTM_REQUEST_START	Arm->DSP	Arm informs DSP to start audio playback
SRTM_REQUEST_STOP	Arm->DSP	Arm informs DSP to stop audio playback
SRTM_REQUEST_B_MOREDATA	DSP->Arm	DSP asks Arm to transmit new audio encoded data

For data channel, an application-specific share buffer starting at 0x202D0000 is used.

4.3 DSP Audio Player

Figure 2 shows a typical setup for an audio player, which consists of audio source, audio decoder, audio sink, and optional pre-processing modules and post-processing modules. These components can be integrated as a pipeline in an audio framework, and the data flow routes from audio source to audio sink.

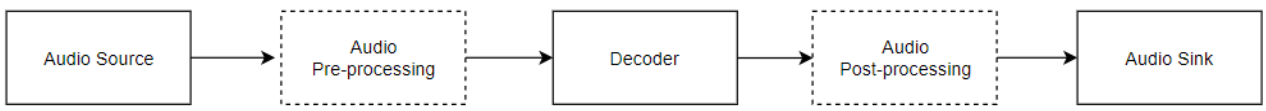

Figure 2. Typical audio player setup

“Audio Player” demo which omits the “audio pre-processing” and “audio post-processing” module implements a minimum audio player which is shown in Figure 3. Audio source, decoder, and audio sink are chained in HiFi4 DSP firmware. The “Audio Player” in under consumer-producer design pattern. There are two consumer-producer pairs in the system. One is audio sink(consumer) and decoder(producer), other is decoder(consumer) and audio source.

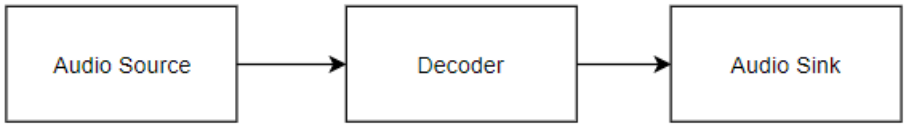

Figure 3. Audio Player" audio chain

The audio source retrieves encoded stream and send it to decoder. Storage and network are typical audio source in audio player. In the demo, Arm reads encoded audio file and write the data to shared RAM between Arm and HiFi4 DSP. From HiFi4 DSP point of view, the encoded audio stream is from Arm, therefore Arm is audio source in “Audio Player” demo.

The demo supports two audio decoders, MP3 and FLAC. For Cadence MP3 decoder, “MP3 Decoder Programmer’s Guide” introduces more information. For LibFlac, https://xiph.org/flac/ has more details.

Decoded audio data is sent to the audio sink for playback. In the demo, the audio DAC WM8904 is the audio sink. DMA1 transmits the decoded audio data to audio DAC via I2S-bus.

Illustrated in Figure 4, Arm sends requests to DSP to create “audio device”, then create “audio source”, “decoder” and “audio sink” components. After DSP creates components, Arm send DSP_REQUEST_START command to start the player. Next, DSP can request encoded data in period until audio data is over.

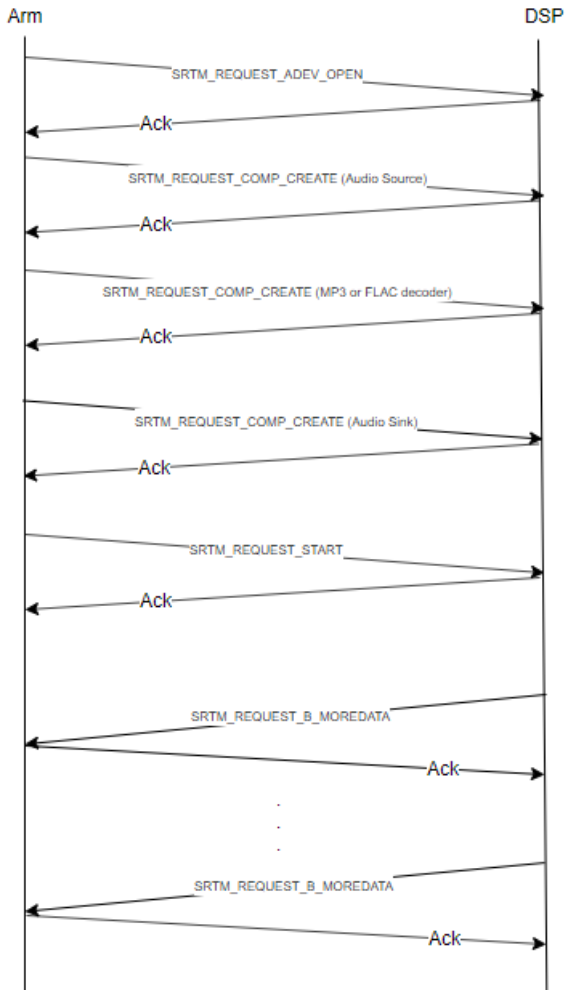

Figure 4. Audio Player” messages

4.4 Playback Control

As shown in Figure 5, “Audio Player” is implemented as a state machine. There are 3 states initialized, prefetching, and playing.

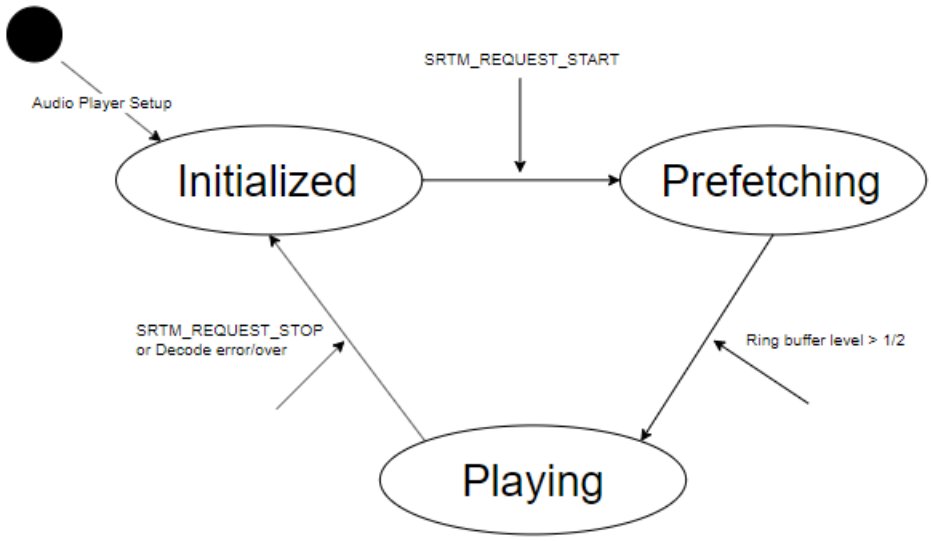

Figure 5. Audio Player" State machine

When “Audio Player” is set up, it is in initialized state, and it goes prefetching state. In prefetching state, decoder is working and keeps putting decoded audio data into a ring buffer, no data consumer for ring buffer at prefetching state. The data level in buffer is monitored. “Audio Player” goes to playing state once data level is higher than ½ of ring buffer size. In playing state, I2S DMA starts, Audio sink is the consumer for ring buffer, the data level in buffer is monitored still, keep the data level is ½ of buffer size, when its level is below ½, decode operation is executed and decoded data are stuffed.

4.5 Audio DMA

There are two DMA controllers in i.MX RT600. In “Audio Player” demo, DMA1 is used at HiFi4 DSP side for I2S DMA. There is no limitation to DMA usage between processors, that meant user can use DMA0 as well if it is not used.

However, there are differences to use DMA and DMA interrupt in HiFi 4 DSP, notes are below.

1. DMA to DSP interrupt must be configured in INPUTMUX
2. DSP interrupt should be registered and enabled in XOS or XTOS
3. DMA descriptor, DMA source address, and DMA destination address should be non-cacheable.

In “Audio Player” demo, DMA1 interrupt is assigned to DSP interrupt slot XCHAL_EXTINT19_NUM, interrupt handler “DMA_IRQHandle” is registered to XOS.

Ping-pong I2S DMA buffer is allocated at 0x20040000 and 0x20050000, which is non-cacheable. DMA device driver in MCUXpresso SDK declared a non-cache section “NonCacheable” for DMA descriptors, this section should be placed in the non-cache memory which address is above 0x20000000.

5 Revision history

Revision number	Date	Substantive changes
0	02/2020	Initial release

.

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

While NXP has implemented advanced security features, all products may be subject to unidentified vulnerabilities. Customers are responsible for the design and operation of their applications and products to reduce the effect of these vulnerabilities on customer’s applications and products, and NXP accepts no liability for any vulnerability that is discovered. Customers should implement appropriate design and operating safeguards to minimize the risks associated with their applications and products.

For more information, please visit: http://www.nxp.com

For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 02 March 2020

Document identifier: AN12762

arm