Firmware Demo on C5515 EVM

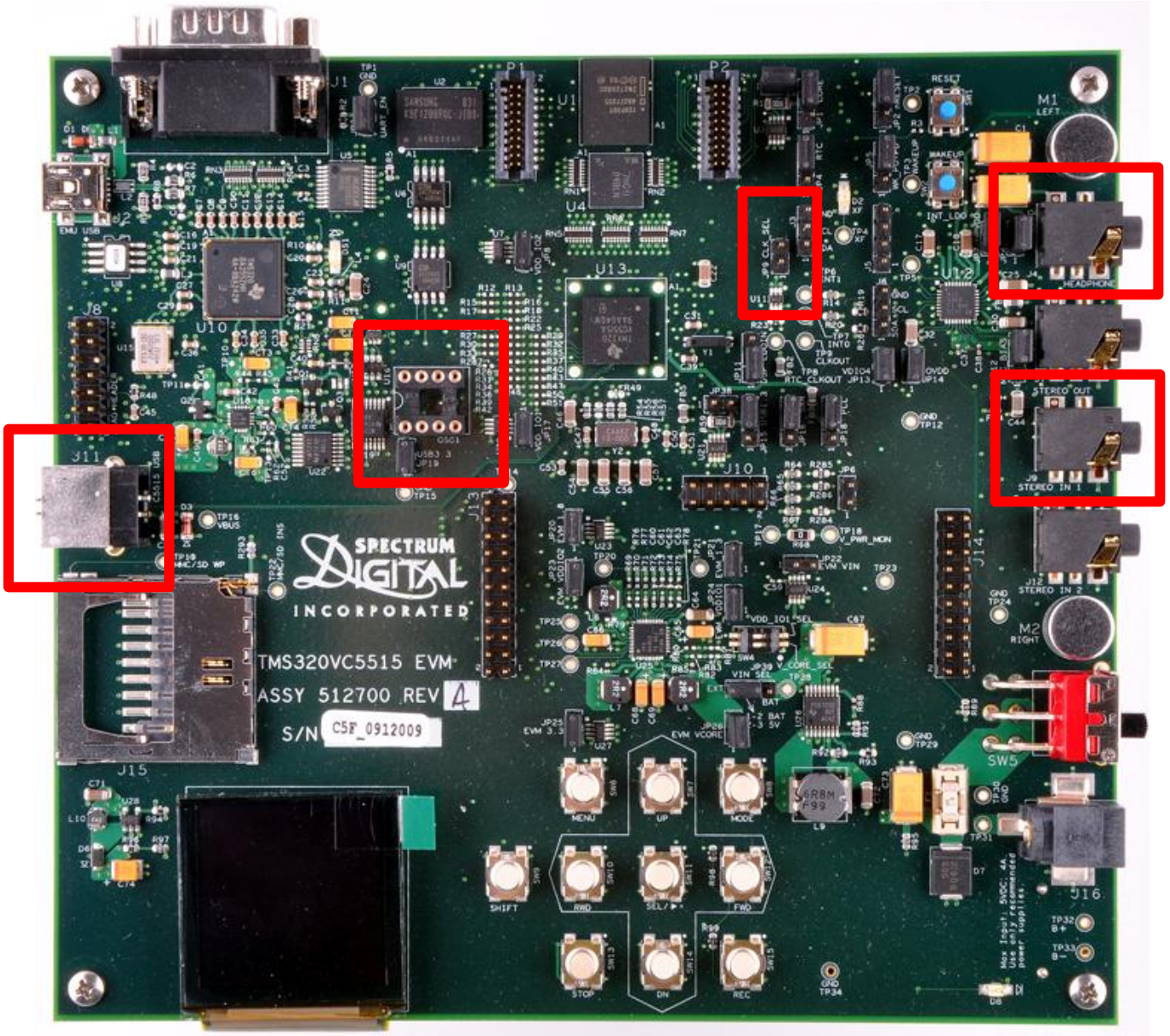
Demo Details – C5515 EVM

- Hardware
  - C5515 EVM
  - Remove jumper from JP9 (CLK_SEL=0)
  - 12 MHz external OSC inserted into OSC1
  - Connect USB cable between Host and USB-B connector (J11)
  - Connect headphones/speakers to HEADPHONE connector (J4)
  - Connect powered audio source (mic, cell phone, C/D player, etc.) to STEREO IN connector (J9)
  - Connect PC with CCS to EVM via JTAG. Load/execute code using emulator.

- Build
  - Tested with CCSv4.2.4. CCSv4 project located in build\CSL_USB_IsoFullSpeedExample_Out
  - Tested with codegen 4.3.8 and 4.3.9
  - Tested with BIOS 5.41.10.36
  - Debug and Release build profiles tested for correct functionality

Demo Details – C5515 EVM (cont.)

- Execute
  - Ensure PLL setup in GEL file uses RTC clock source for DSP (e.g. OnTargetConnect() function in c5505evm_pg20.gel contains call to ProgramPLL_100MHz_clksel0())
  - Load executable to EVM
  - Execute code. EVM should enumerate as “TI C55x USB Audio” (e.g. see Windows Volume Control or Sounds and Audio Device Properties).
  - If problem with execution
    - Scripts→C5505EVM_Configuration→CPU_Reset
    - Scripts→C5505EVM_Configuration→Peripheral_Reset
- Functionality
  - Initiate playback of audio on Host (e.g. Windows Media Player). Audio will play back on EVM.
  - Initiate record of audio on Host (e.g. Windows Sound Recorder). Audio will record to Host.
  - Control playback volume up / volume down / mute using HID controls on EVM
    - SW7: Volume Up
    - SW14: Volume Down
    - SW13: Playback Mute
- Tested on Windows XP SP3, Windows 7. Playback and record tested on Mac OSX.


SPECTRUM DIGITAL INCORPORATED
TMS320VC5515 EVM
ASSY 512700 REV A
S/N C5F_0912009

Firmware Demo on C5535 eZdsp

Demo Details – C5535 eZdsp

- Hardware
  - C55x eZdsp
  - DSP switch SW3 1:OFF, 3:OFF
  - Connect USB cable between Host and USB connector (J1)
  - Connect headphones/speakers to STEREO OUT connector (J4)
  - Connect audio source (mic), to STEREO IN connector (J3)
  - (Option 1) Insert micro-SD card into micro-SD connector (J6). Power eZdsp to bootload code.
  - (Option 2) Connect PC with CCS to XDS100 emulator USB connector (J2). Load/execute code via emulator.

- Build
  - Tested with CCS 4.2.4. CCSv4 project located in build\CSL_USB_IsoFullSpeedExample_ezdsp_Out
  - Tested with BIOS 5.41.10.36 and code generation tools 4.3.9
  - Debug and Release build profiles tested for correct functionality

Demo Details – C5535 eZdsp (cont.)

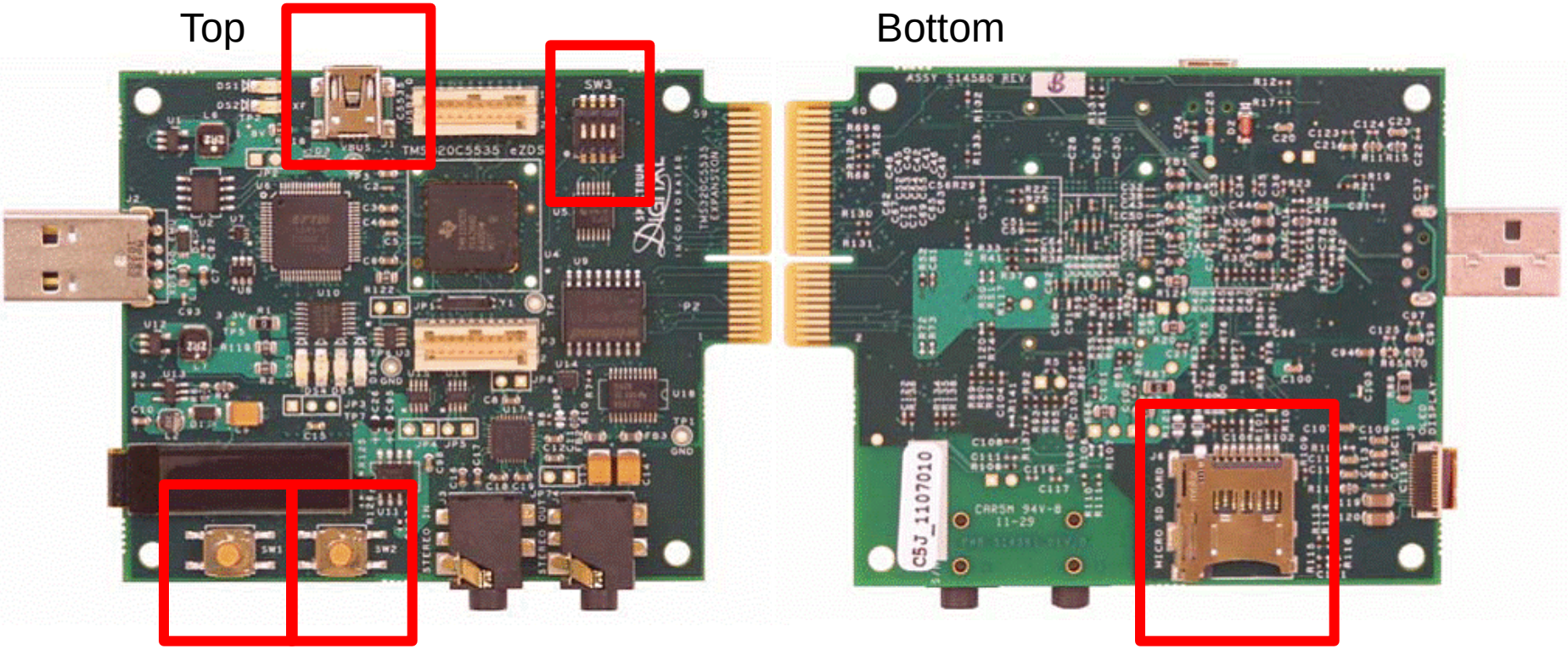
- Execute from CCS
  - Ensure PLL setup in GEL file uses internal clock source (CLK_SEL=0)
  - Load executable to eZdsp
  - Execute code. eZdsp should enumerate and USB Audio Device (see Windows Volume Control or Sounds and Audio Device Properties).
  - If problem with execution
    - Scripts→CPU_Reset
    - Scripts→Peripheral_Reset

- Functionality
  - Initiate playback of audio in Windows (e.g. Media Player). Audio will play back on eZdsp.
  - Initiate record of audio in Windows (e.g. Sound Recorder). Audio will record to Host.
  - Control play back volume up / volume down / mute using HID controls on eZdsp
    - SW1: Volume Up
    - SW2: Volume Down
    - SW1+SW2 (depress simultaneously): Playback Mute
  - Spectrum of playback signal displayed on LCD

- Tested on Windows XP SP3, Windows 7. Playback tested on Mac OSX and Linux (Ubuntu).

Top

Bottom

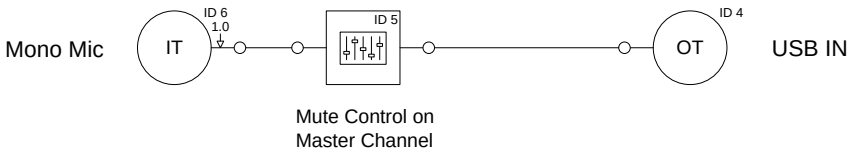


C55xx USB Audio Class Overview

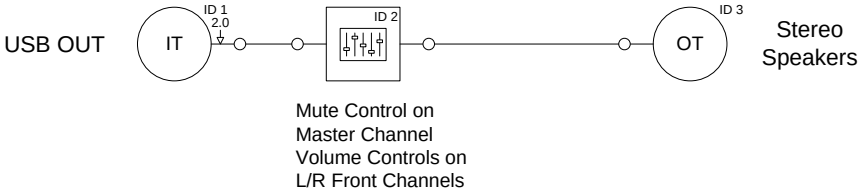
Supported Features

- USB full-speed operation
- Audio Device Class 1.0 compliant
- Audio Record
  - 16-bit PCM
  - 48/16 kHz sampling rate
  - Mono
  - Mute control
  - Asynchronous synchronization
- Audio Playback
  - 16-bit PCM
  - 48/16 kHz sampling rate
  - Stereo/mono
  - Mute and volume control
  - Adaptive synchronization
- HID
  - Playback mute
  - Playback volume up / down
  - Record mute

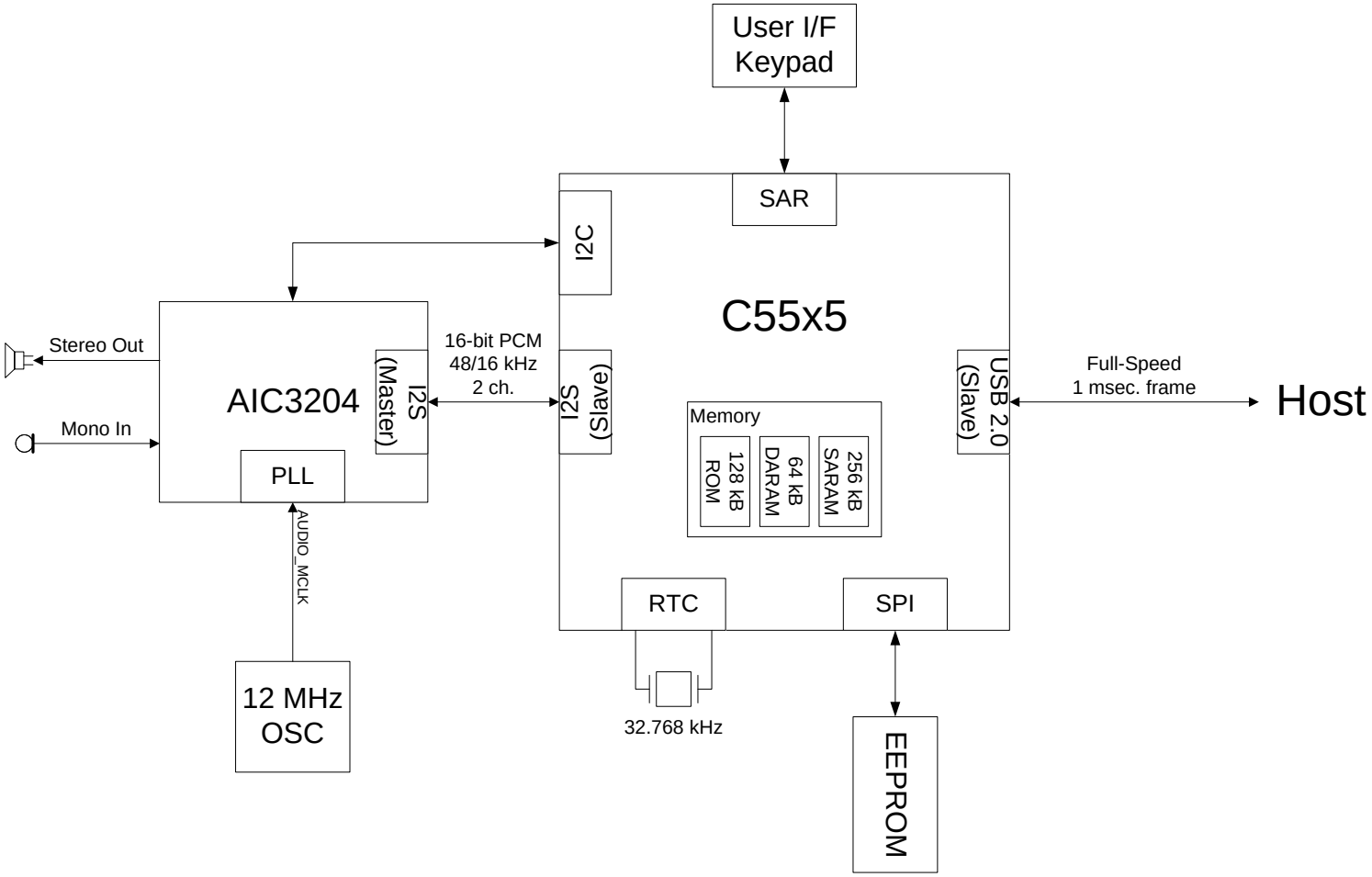
Audio Record



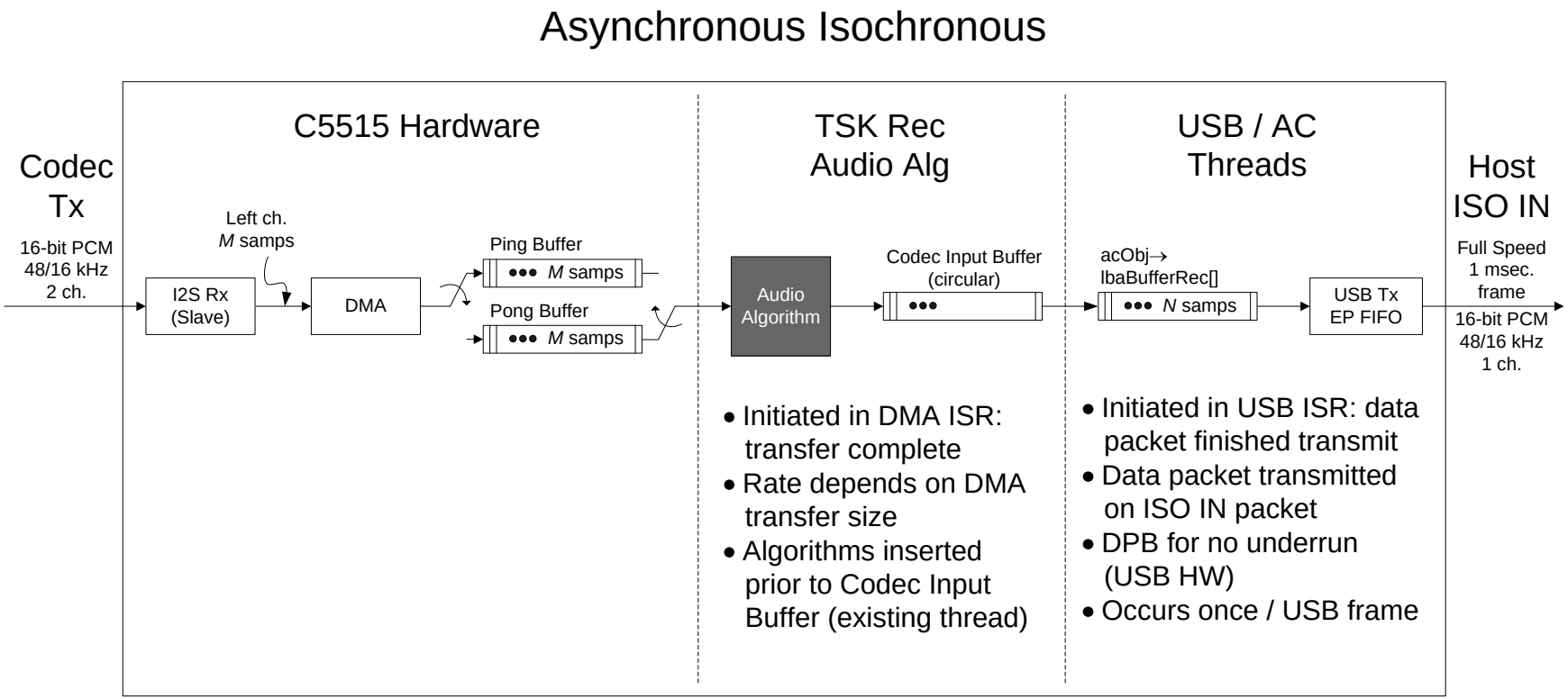
Audio Playback



Hardware Block Diagram

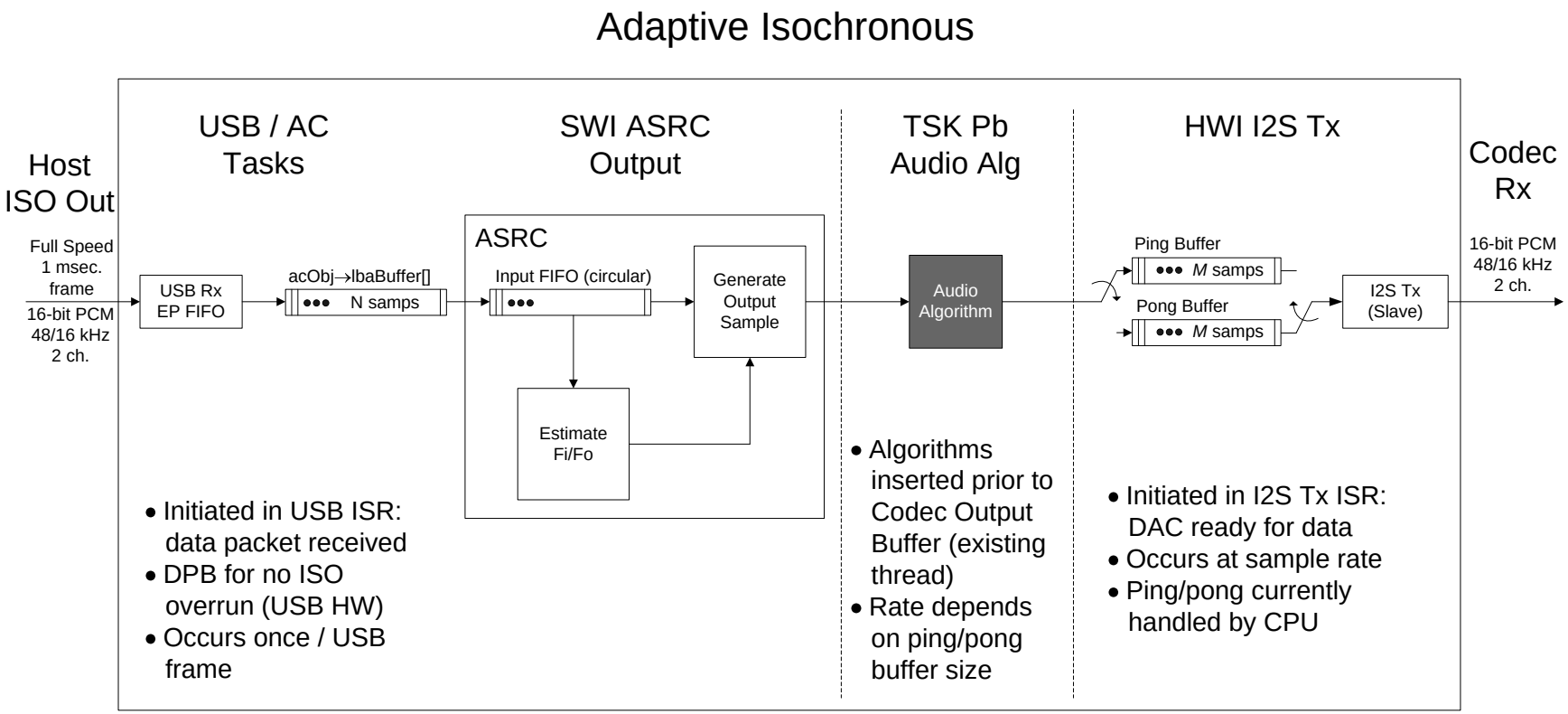


Record Data Flow



- *M*: number of samples in DMA transfer can be changed (default 1 msec.)
- *N*: number of output samples in USB frame varies
- Data rate to USB matches ADC data rate
- SRC performed on Host

Playback Data Flow



- *N*: number of input samples in USB frame varies
- *M*: number of samples in ping/pong buffers can be changed (default 1 msec.)
- Data rate to DAC matches USB data rate
- SRC performed on C55xx

HID Description

- Four binary HID Controls exposed to Host in HID descriptors
  - Playback Mute on Master Channel
  - Playback Volume Up on Left / Right Front channels
  - Playback Volume Down on Left / Right Front channels
  - Record Mute on Master Channel
- Physical controls are buttons in push-button network on C55xx board (EVM or eZdsp)
  - SAR samples network every 64 msec. (configurable) checking for change in control state
  - Report generated if change in state. Reports contain all control states.
- Device sends reports when requested by Host
  - Host polls device on interrupt IN endpoint every 32 msec. (configurable)
  - If no report available, request is NAK'ed
- Host driver interprets report data, changes controls on Host
  - Length of Short, Long, and Double presses determined by Host. Device simply sends state (binary) of HID control when change in control state.
- Host driver interprets report data, translates into control IN messages
  - Messages are setup requests to Audio Controls (e.g Playback Volume Control). Minimum, maximum, and resolution for Audio Controls specified in Audio Control Interface.
  - Message contents interpreted and codec configured as requested

Resource Requirements

MHz

Use Case	MHz
Rec @ 16 kHz	21.17
Pb @ 16 kHz	21.42
Pb @ 48 kHz	41.44
Pb + Rec @ 16 kHz	28.64
Pb @ 48 kHz + Rec @ 16 kHz	48.61

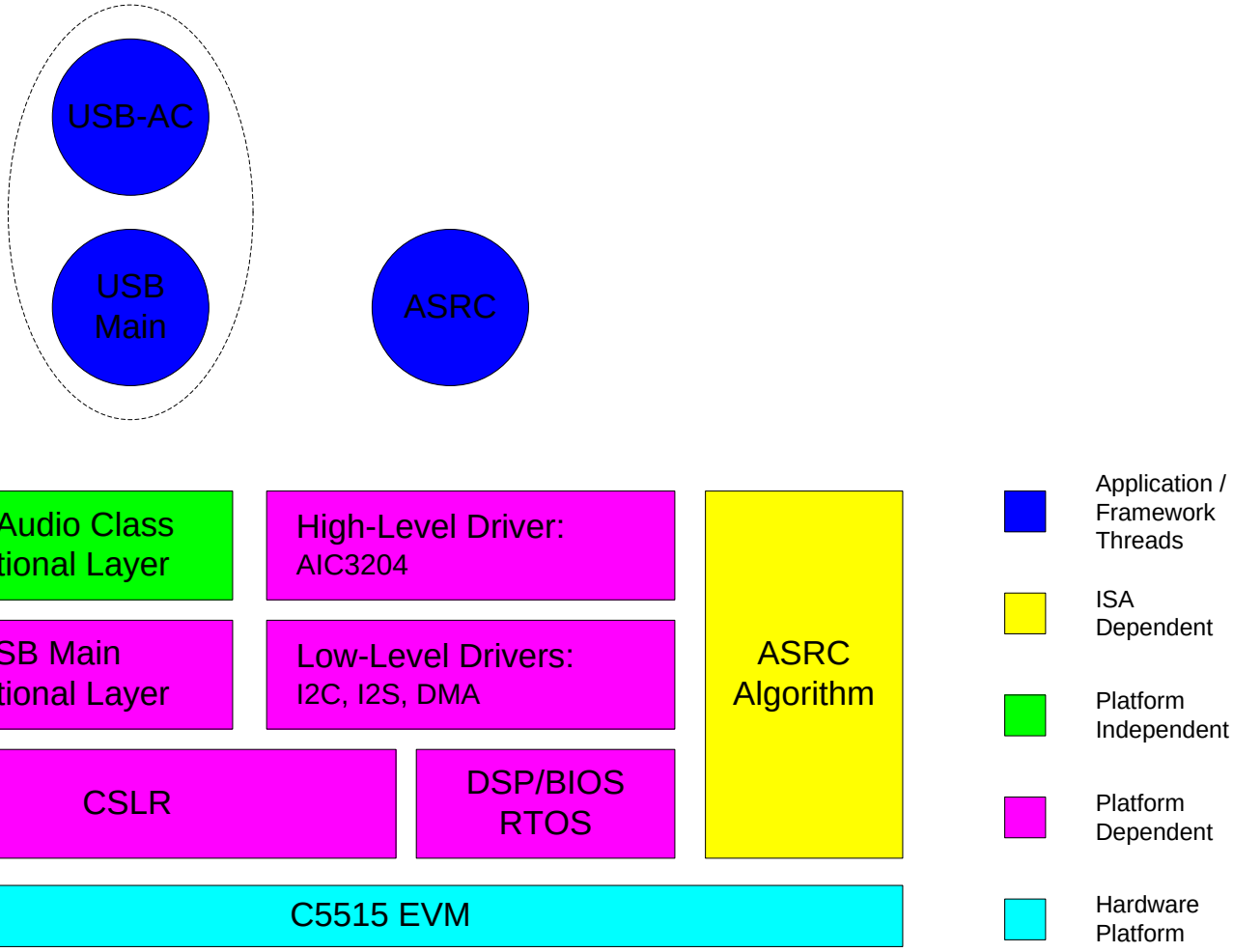
Memory

Memory Type	kB	
Program	49.18	
Initialized Data	9.20	
Uninitialized Data	46.72	31.97
(System Stack)	12	
(Task Stacks)	17.5	

- Optimization for MHz and memory not yet performed. Expect reduction in resource requirements with optimization.

Software Architecture Overview

Software Architecture



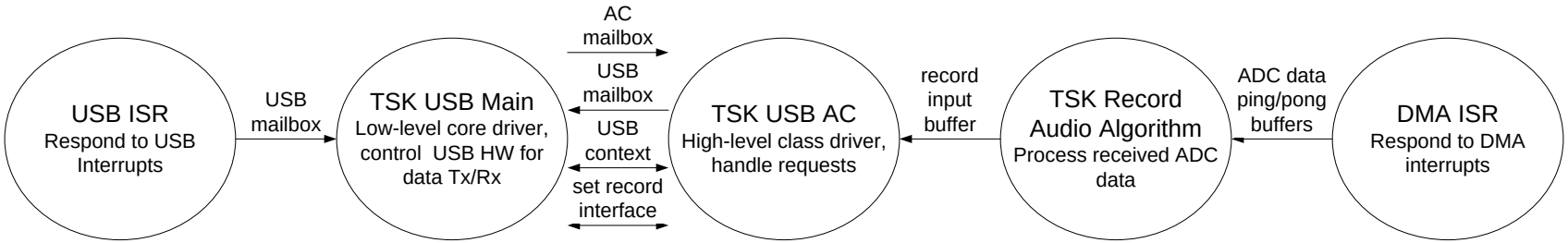
Software Description

- Chip Support Library
  - Set of macros providing simple hardware abstraction
  - Enables developers to configure registers symbolically, eliminating need to calculate bit-maps for each register
- DSP/BIOS Real-Time Operating System
  - Scheduling with different thread types: Hardware Interrupt, Software Interrupt, Task
  - Inter-task synchronization and communication: semaphores and mailboxes
  - Small-footprint instrumentation with logging and statistics objects
- Low-level drivers for I2C, I2S and DMA
  - Support basic resource management, preventing simultaneous assignment of resource to more than one use
- USB driver
  - USB Main functional layer specific to C5515 USB hardware. Controls USB hardware for data Tx/Rx.
  - USB Audio Class functional layer independent of C5515 USB hardware. Interprets/handles USB Standard and Audio Class requests.
- Asynchronous Sample Rate Converter
  - C-callable assembly optimized for C55xx Rev. 3 ISA. C API insulates user from assembly code.
  - Library can be reused on any C55xx Rev. 3 device without modification or rebuild
  - No use of peripheral hardware or OS services
- Framework
  - Threading model allowing data throughput and real-time deadline requirements to be achieved

Thread Summary

Name	Purpose	Executes	Priority	Frequency
	Hardware Interrupts			
HWI_INT4 (gpt1Isr())	Initiate processing of user interface (SAR sampling, etc.).	Timer Int	0	64 msec.
HWI_INT8 (DMA_Isr())	Execute callback functions for active DMA channels.	DMA1 Ch0 Int	1	Max frame rate (1 msec)
HWI_INT14 (i2s_txIsr())	Call ASRC function to accumulate Tx samples per USB SOF. Write next L/R output samples to I2S TX. Swap ping/pong buffers.	I2S2 TX Int	2 - highest	PB samp. rate (worst 20.83 usec)
HWI_INT20 (USBisr())	Respond to USB interrupts. Send message to TSK_MUSBMainTask indicating interrupt occurrence. Post SWI_Store_USB_Input for storing USB input for playback.	USB Int	1	1/frame ISO IN 1/frame ISO OUT 1/frame SOF
	Software Interrupts			
SWI_Store_USB_Input (store_USB_input())	Copy frame data from EP2 OUT FIFO to AC object buffer. Prepare EP2 OUT FIFO to receive next frame data. Post SWI for processing USB input (SWI_Process_USB_Input).	Post from USBisr()	5 - highest	Frame rate (1 msec)
SWI_Process_USB_Input	Call ASRC function to update Fi/Fo estimate. Call ASRC function to update ASRC input FIFO. Call ASRC function to update NCO & compute output samples.	Post from store_USB_input()	5	Frame rate (1 msec)
SWI_UserInterface	Check change in state in push-button network. If change, format report and indicate report ready to USB Interrupt In EP handler.	Post from gpt1Isr()	4	64 msec
	Tasks			
mainTsk (CSL_acTest())	System initialization.	Application startup	15 – highest	Run once
TSK_MUSBMainTask	USB Main driver. Control USB HW for Tx/Rx. Adjust EP sample rates and interface mute & volume controls.	USB Main mbx msg (e.g. USB interrupt, USB transaction post from USB AC)	9	2/frame ISO IN 1/frame SOF
TSK_MUSBACTask	USB AC driver. Handle Standard and AC-specific requests on EP0. Handle IN/OUT requests on ISO EPs.	USB AC mbx msg (e.g. ISO EP IN/OUT req., EP0 req., USB reset)	8	2/frame ISO IN
TSK_PbAudioAlg	Process ASRC output with playback audio algorithm. Output data to I2S Tx ping/pong buffer.	Post from i2s_txIsr()	7	Max frame rate (1 msec)
TSK_RecAudioAlg	Process DMA ping/pong input data with record audio algorithm. Output data to Codec Input Circular Buffer.	Post from DMA_Isr()	7	Max frame rate (1 msec)
TSK_CodecConfig	Configure codec via I2C in response to USB commands on EP0.	Codec cfg mbx msg	6	N/A

Record Thread Collaboration



USB Context

USB operating mode (POLLED)
Initialized flag
Setup packet flag
ISO IN packet ready flag
ISO OUT packet ready flag
Interrupt flags
Pointer to EP status array
Bus speed
EP0 state
Cable state
Suspend callback fxn
Wakeup callback fxn
Start transfer callback fxn
Complete transfer callback fxn
...

Endpoint

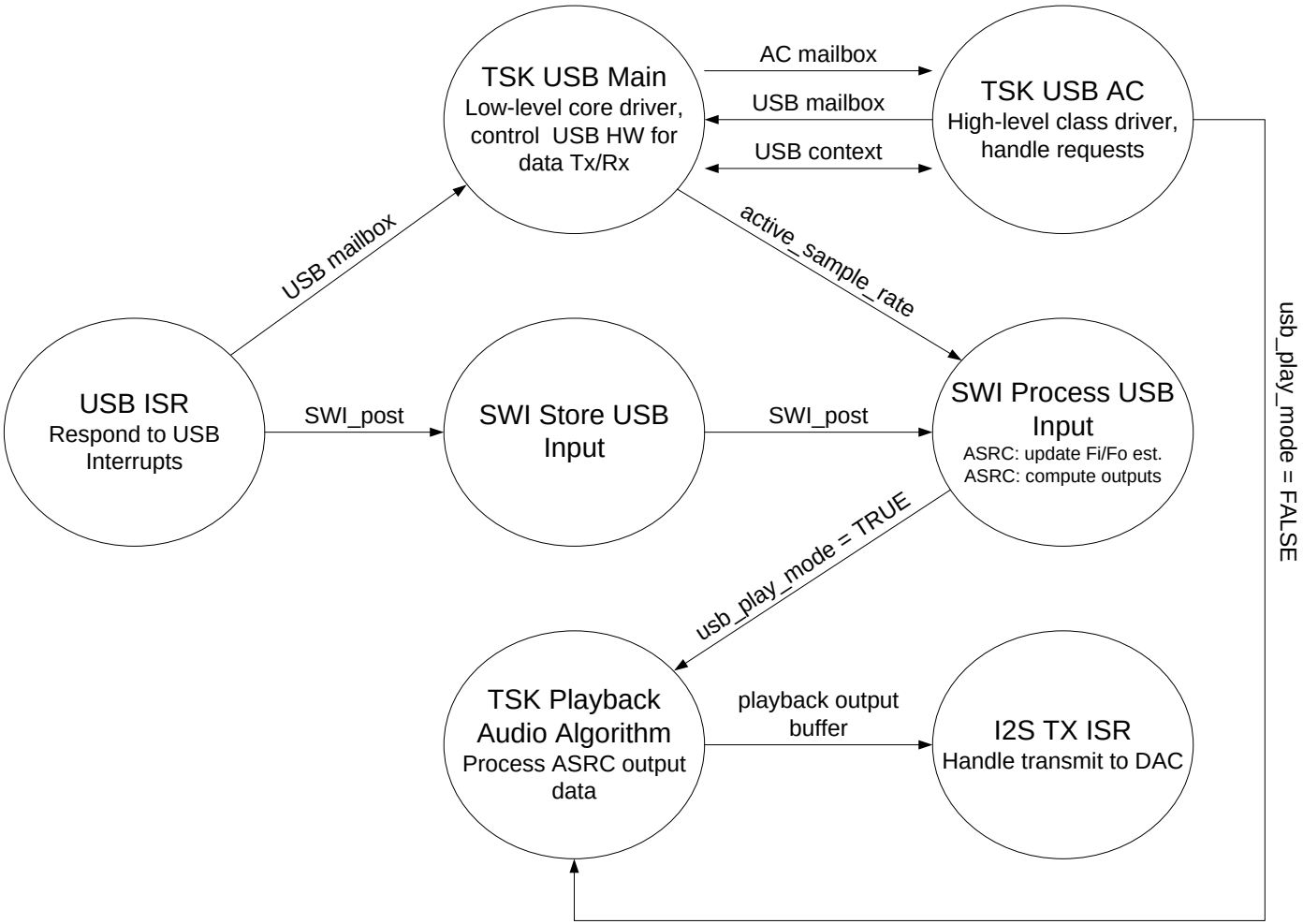
Endpoint number + IN/OUT
Transfer type (CTRL, BULK, ISO, INT)
Maximum Packet size
...

Endpoint Status

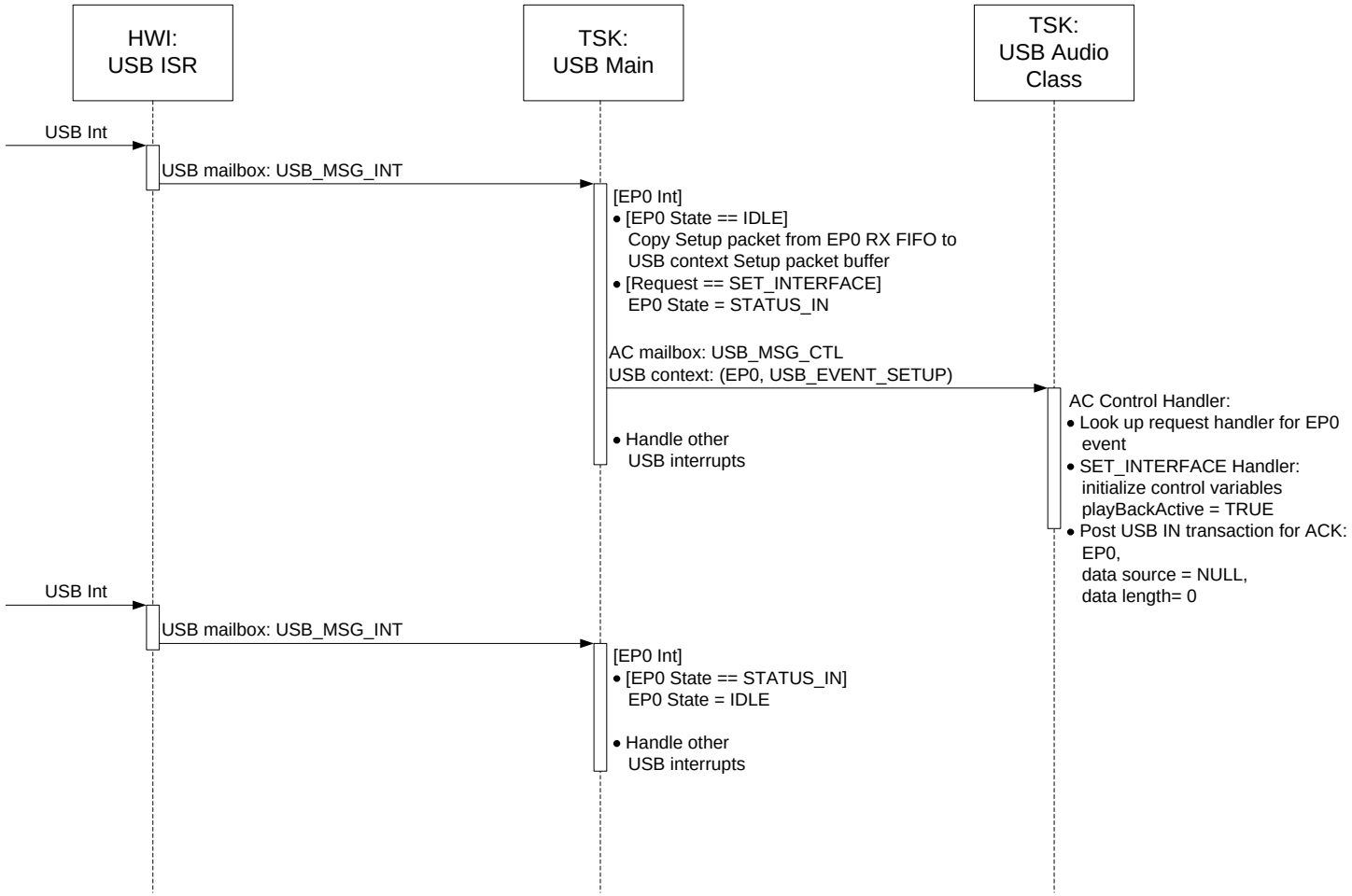
Endpoint number
Transfer type (CTRL, BULK, ISO, INT)
Packet size
Initialized flag
Pointer to current transfer
Events bitfield
Stalled flag
...

EP0 IN
EP0 OUT
EP1 IN (ISO)

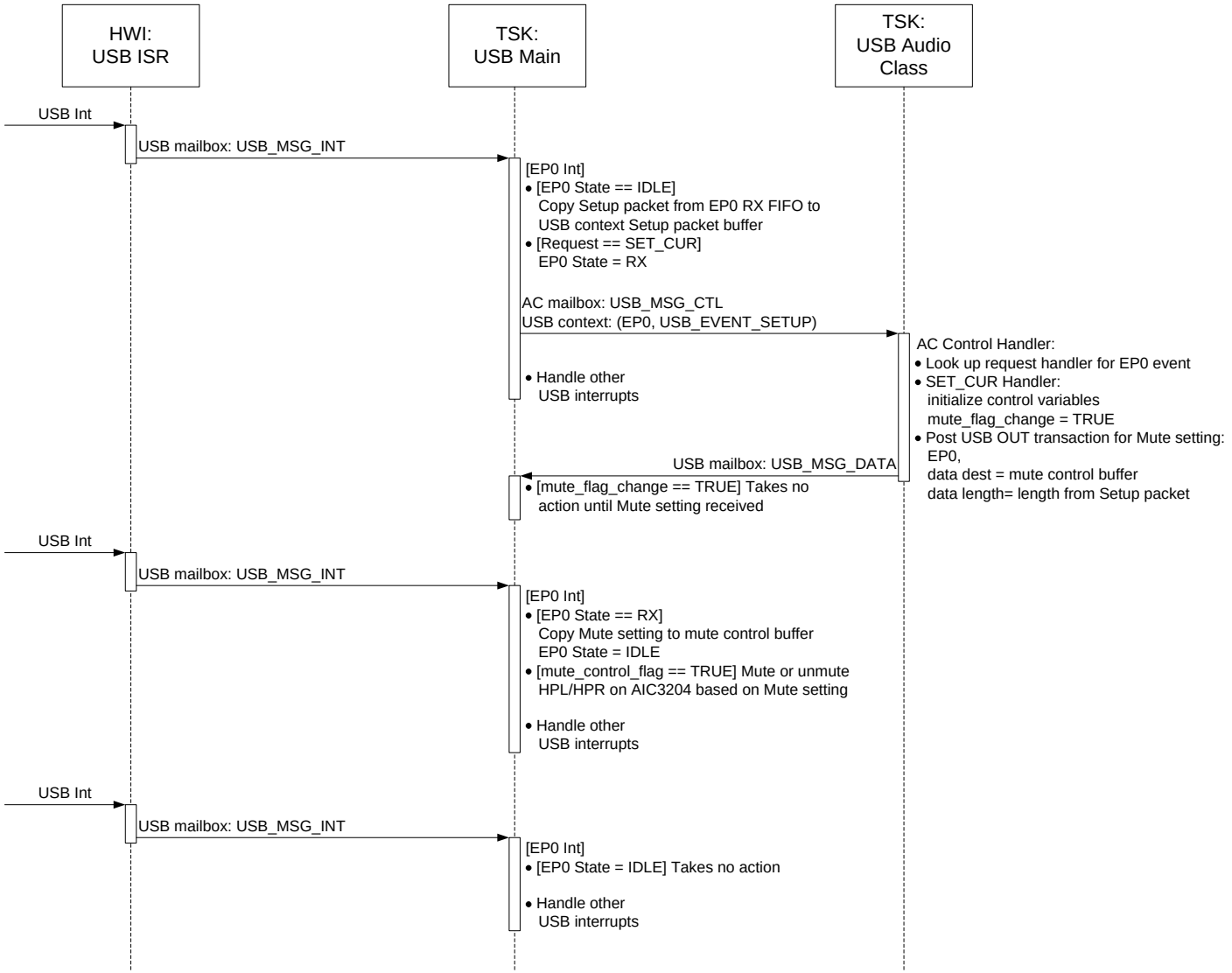
Playback Thread Collaboration



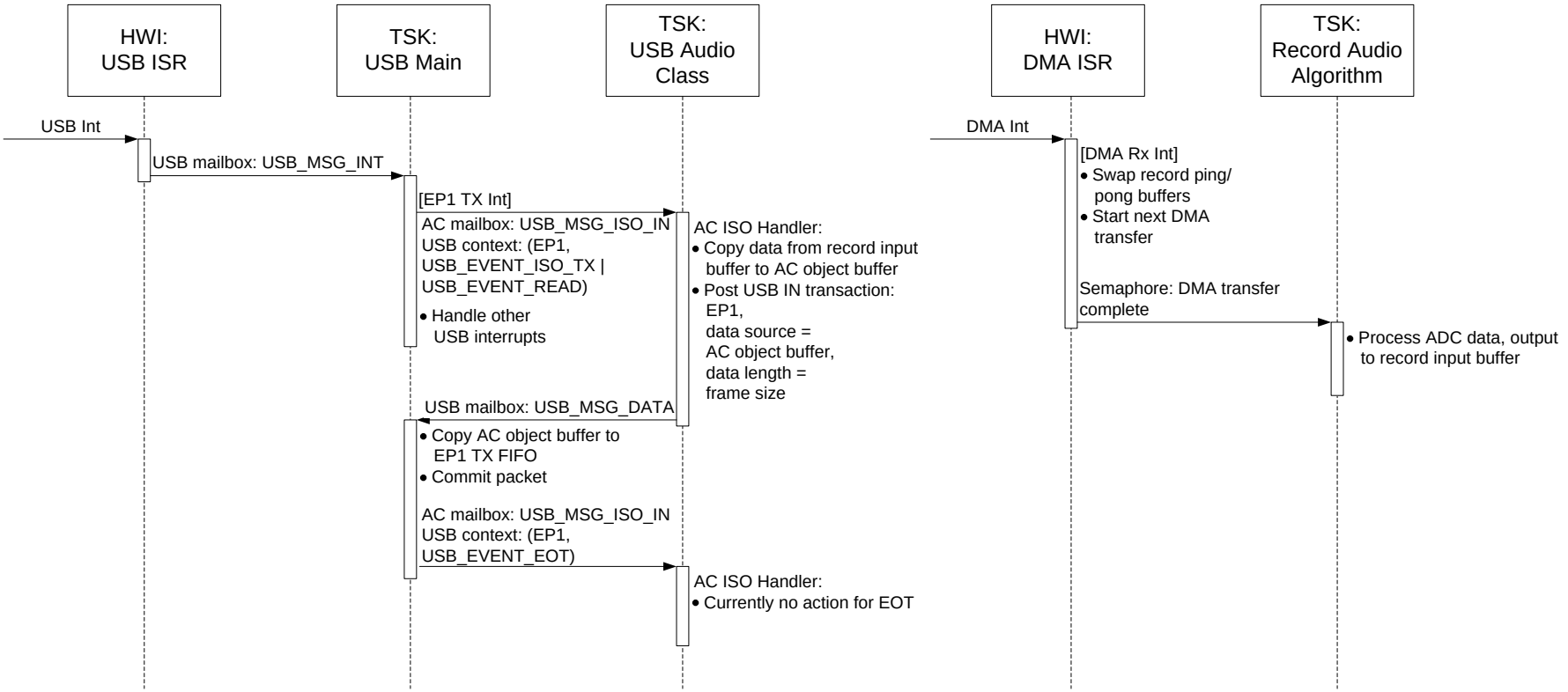
Set Interface Sequence Diagram



Set Mute Control Sequence Diagram



Record Sequence Diagram



Playback Sequence Diagram