Hello, and welcome to this presentation of the STM32 Serial Audio Interface.
I will present the features of this interface, which is used to connect external audio devices.
The Serial Audio Interface (SAI), is integrated in STM32 products to provide an interface, for communicating with external audio devices such as amplifiers, ADCs, DACs, audio codecs, and audio processors. This interface is fully configurable, supporting the most digital audio standards, allowing easy connection to any existing audio devices. Due to the internal synchronization features of the SAI, the required IO pins are reduced to a minimum.
The SAI can be programmed in three different modes:
The Free protocol mode allows the SAI to support standards such as I2S, PCM, TDM... Due to its flexibility, it is possible to customize the serial interface if needed.
The SPDIF protocol mode, allows the SAI to transmit audio samples using the IEC60958 standard.
The AC97 protocol is also supported by the SAI.
The SAI supports all the standard audio sampling rates, depending on the crystal frequency used for the application. In addition, the SAI supports MASTER and SLAVE mode, half-duplex or full-duplex communication. It is also possible to synchronize several SAIIs together. The SAI also provides a FIFO buffer of 8 samples, and up to two interrupt and DMA interfaces.
The SAI is composed of two independent sub-blocks (A and B). Each sub-block has:
Its own APB interface, clock generator, FIFO buffer, DMA interface, and Interrupt interface.

Each sub-block can be configured in receive or transmit, master or slave, with its own protocol.
Internal and external synchronization allows two sub-blocks to be synchronized, or two SAI to be synchronized.

Each sub-block can handle up to four IOs:
FS_x is the frame synchronization signal
SCK_x is the bitclock
SD_x is the serial data line
MCLK_x is the MASTER clock.
The STM32L4xx, embeds 2 SAI s.
Each SAI can receive a kernel clock (SAIn_CK_x) from one of the three internal PLL or from the PADs (SAInEXTCLK).
The kernel clock is used by the SAI in order to generate the timing of the serial audio interface when programmed in MASTER mode.
The free protocol mode, the flexible programming interface facilitates the configuration of most common audio standard interfaces.
The following example shows some of the possibilities of the interface, for I2S-Like protocols:

In I2S-Like protocol, each edge of the frame synchronization signal (FS) is used to align the slots start position.
The frame length, the duty cycle, and polarity can be adjusted.
The clock strobing edge can be selected.
The position of the slots with respect to the frame edges can be selected.
The number of slots per frame: needs to be an even number in I2S-Like Protocol.
The following example shows some of the possibilities of the interface, for the TDM-Like protocols:
In TDM-Like protocol, only one edge of the frame synchronization (rising or falling) is used to align the slots position.
The frame length, the duty cycle, and polarity can be adjusted.
The clock strobing edge can be selected.
The position of the slots with respect to the frame active edge can be selected.
The number of slots per frame (up to 16) can be selected.
The slot size is always bigger than or equal to the data size. The SAI allows control of the position of the data inside each slot, and setting of the un-used slots to HiZ if needed. This function can be useful when the data line is shared between several devices.
In MASTER mode the SAI can generate the master clock (MCLK), or use an external master clock via SAIxEXTCLK PADs. The master clock can be used to provide a reference clock to the external audio codecs.

In SLAVE mode, the MCLK signal is not used.
In MASTER mode it is up to the SAI to generate the timing, in order to provide the correct sampling rate.
In SLAVE mode, the sampling rate is provided by the external audio device.
The clock generator is needed for MASTER mode communications, it is used to adjust the sampling rate of the serial audio interface. The clock generator provides the root frequency for the MCLK_x, SCK_x and the FS_x.

When the master clock (MCLK) needs to be generated, the frame length must be a power of two. The ratio between the FS_x frequency and the MCLK_x frequency is fixed to 256.
When the MCLK_x does not need to be generated, the frame length can take any value from 8 to 256.
Free protocol modes

- SAI Synchronization:
  - The SAI can synchronize the two sub-blocks (internal synchronization).
  - The SAI can synchronize sub-blocks of different SAI s (external synchronization).
  - If the synchronization is not used, each sub-block is independent.
    Some examples:
      - SAI_A in I2S Philips Master, SAI_B in SPDIF
      - SAI_A in TDM SLAVE, SAI_B in AC97
  - If internal or external synchronization is used, the following limitations must be respected:
    - It is not possible to synchronize 2 SAI Sub-Blocks using different protocols.
    - It is not possible to synchronize 2 SAI s using different protocols.

The internal synchronization can be used for communication requesting two data lanes, such as full-duplex I2S. The external synchronization can be used for communication requesting more than 2 data lanes (up to 4). For example when interfacing HDMI ICS.

All the sub-blocks synchronized together must use the same protocol characteristics.
In order to reduce the data size, it is possible to insert in the data path, a A-law or u-law compander.
Note that A-law and u-law are not lossless compressors. Companding modes are generally used in telephony:
The small data are amplified and the big data are attenuated. The SNR tends to be identical for a strong and for a weak signal.
The SAI also provides a MUTE function.
Free protocol modes

- **Anticipated/Late Frame Error:**
  - This function can be used to detect glitches on the SCK Clock/FS due to a noisy environment.
  - In SLAVE mode, the SAI can detect if the frame synchronization occurs at the expected time: not too late, not too early.
  - A Status flag is available, and an interrupt can be generated as well.
  - After an anticipated or late frame detection error, the application software can re-start the SAI.

The Anticipated/Late frame error detection increases the interface reliability by detecting unexpected frame synchronization misalignment.
Free protocol modes

- The SAI guarantees the data alignment even if underrun/overrun occurs

- **Overrun/Underrun Handling:**
  - Overrun occurs when the RX-FIFO is full, and new data coming from the serial interface has to be stored.
  - Underrun occurs when the TX-FIFO is empty, and new data is requested by the serial interface.
  - Example: FIFO Overrun on Slot 1
The SAI supports the audio IEC60958 standard in transmit mode when configured in SPDIF mode. The software has to handle the CS, U and V bits. The SAI generates the Parity bit according to the transmitted data.
In the IEC60958, the block structure is used to decode the Channel Status (CS), and User information (U).
Each block contains 192 frames.
Each frame contains 2 sub-frames.
Each sub-frame contains 32 bits.
A synchronization preamble allows the detection of the block and sub-frame boundaries.
The Fsai_ck_x frequency must be adjusted in order to generate the proper audio sample rate.
The data inside the transmit FIFO must be adjusted as shown in the slide: the MSB must always be at position 23.
The SAI is able to work as an AC’97 link controller. When this protocol is used, the frame length, the slot number, and slot length are fixed by the hardware.
Several events can be enabled in order to generate interrupts.  
- The FIFO request event, the overrun/underrun event,  
- the anticipated or late frame synchronization event,  
- the codec not ready event (only in AC’97),  
- the mute detection event.

The WCKCFG event can be used in order to inform the user that  
the frame length of the SAI has been improperly programmed.  
This feature is only available in MASTER mode.
The SAI needs the bus interface clock (APB clock) and the kernel clock (SAIn_CK_x) to work properly.
For a full-duplex MASTER mode, two data lanes are needed, so two sub-blocks need to be used. The MASTER sub-block A, provides the synchronization to SLAVE sub-block B, using the internal synchronization feature (IO Line Management).

Using internal synchronization, the number of IOs required is reduced to its minimum.
Another example of full-duplex mode use case with the TDM protocol usage.
The slot 1 is inactive (not used) for sub-block A, the slots 2 and 3 are inactive for sub-block B.
For both sub-blocks, the frame structure has 4 slots.
The sub-block A will generate 3 samples per frame.
The sub-block B will receive 2 samples per frame.