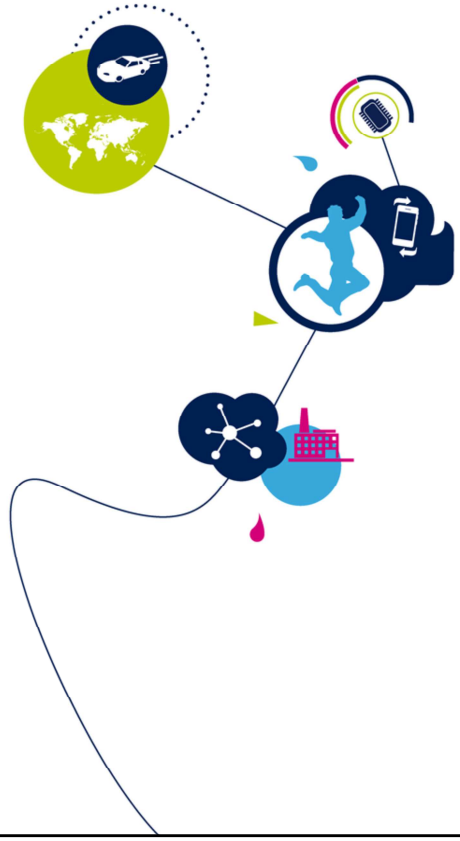
STM32H7 - SAI

Serial Audio Interface

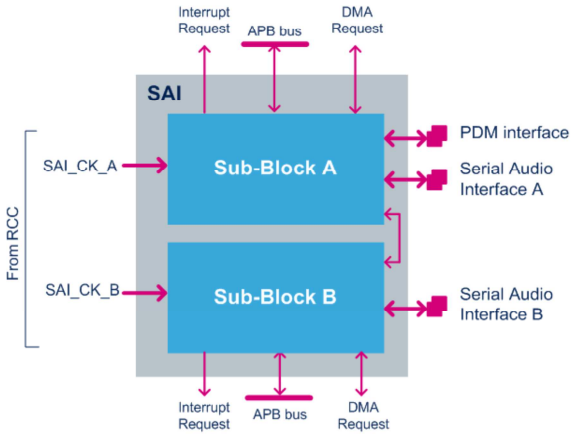
Revision 1.0



life.augmented

Hello, and welcome to this presentation of the STM32 Serial Audio Interface or SAI.
It covers all the features of this interface, which is widely used to connect external audio devices.

Overview



- Provides a communication interface for external audio devices
  - Fully configurable
  - Supports various standards: I2S, TDM, SPDIF…
  - Digital microphone interface
  - Two independent sub-blocks

Application benefits

- Supports a large variety of audio devices
- Simple interface for digital microphones
- Only useful signals are output
- Simple to implement

life.augmented

The SAI integrated inside STM32 products provides an interface allowing the microcontroller to communicate with external audio devices such as amplifiers, ADCs, DACs or audio processors. This interface is fully configurable and supports most audio standards, allowing easy connection to existing audio devices.

Thanks to internal synchronization features, the amount of I/O pins is reduced to its minimum.

Key features (1/2)

3

- Supports several protocols
  - Using Free protocol mode:
    - I2S Philips Standard (Inter-IC Sound)
    - I2S MSB or LSB-justified (Variant of Inter-IC Sound)
    - TDM (Time Division Multiplexing)
    - PCM (Pulse Code Modulation)
    - Others…
  - SPDIF Output (Sony/Philips Digital InterFace)
  - PDM Interface (Pulse Density Modulation Interface)
  - AC'97 (Audio Codec 97 from Intel)

life.augmented

The SAI can be programmed in four different modes:

- Free protocol mode allows the SAI to support standards such as I2S, PCM, TDM, etc. Thanks to its flexibility, it is possible to customize the serial interface if needed.
- SPDIF protocol mode allows the SAI to transmit audio samples using the IEC 60958 standard.
- PDM Interface mode allows the SAI to connect up to 8 digital microphones for beamforming or simple speech capture applications.
- AC'97 protocol.

Key features (2/2)

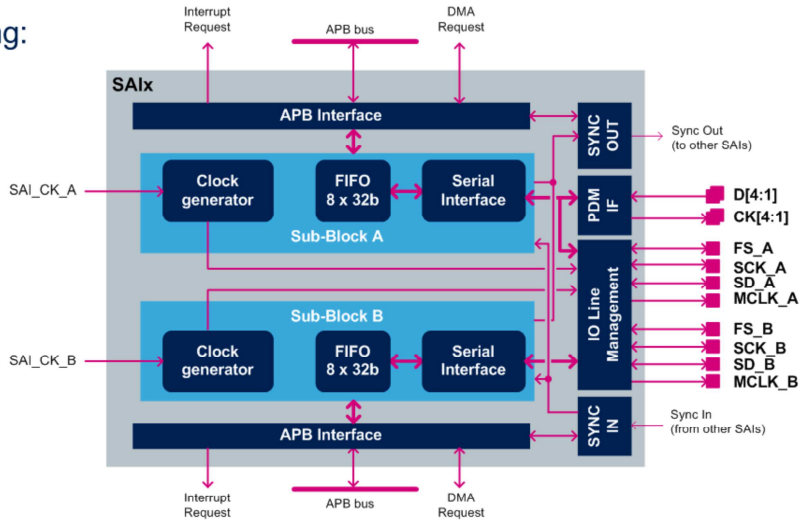
- The SAI supports:
  - All usual audio sampling rates: 44.1,16, 48, 96 and 192 kHz (depending on crystal frequency)
  - Master or Slave mode for each sub-block
  - Data input, output or full-duplex for each sub-block
  - Clock generator for each sub-block
  - Synchronization between sub-blocks or with other SAIs
  - Companding modes (µ-Law, A-Law)
  - 8-word FIFO size
  - 2 DMA interfaces
  - 2 Interrupt lines

life.augmented

The SAI supports all the usual audio sampling rates, according to the crystal frequency used for the application.
In addition, the SAI supports the Master and Slave modes, in half-duplex or full-duplex communication.
It is also possible to synchronize several SAI interfaces together.
The SAI also provides a FIFO buffer of 8 samples, and up to two interrupts and DMA interfaces.

Block diagram

- The SAI embeds:
  - Two independent Sub-Blocks offering:
    - A clock generator,
    - A flexible serial interface,
    - FIFO buffers,
    - APB interface,
    - DMA and interrupt services,
  - Synchronization mechanism
  - IO line management
  - A PDM interface



life.augmented

The SAI is composed of two independent sub-blocks (sub-block A and B).
Each sub-block has its own APB interface, clock generator, FIFO buffer, DMA interface, and Interrupt interface.

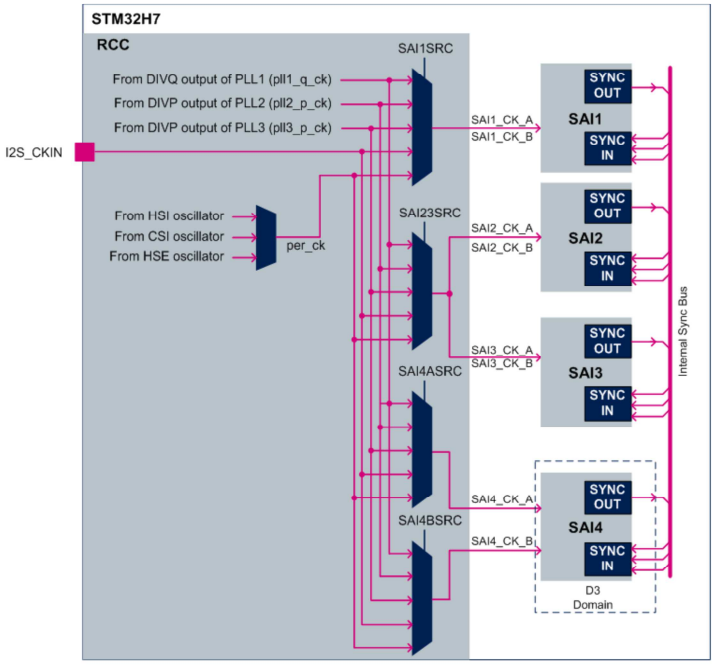
Each sub-block can be configured in Receiver or Transmitter mode and in Master or Slave mode with its own protocol. Internal and external synchronization allows two sub-blocks to be synchronized, or several SAI interfaces to be synchronized.

Each sub-block can handle up to four IOs. For each sub-block, FS is the frame synchronization, SCK is the bit clock, SD is the serial data, and MCLK is the Master clock.

In addition, a PDM interface allows the connection of up to 8 digital microphones.

SAI in the circuit

- Rich kernel clock selection:
  - One of the 3 PLL outputs, or
  - One of the oscillator outputs, or
  - Clock from pad
- Flexible internal synchronization:
  - Each SAI can provide a synchronization to the 3 other SAIs,
  - Each SAI can be synchronized with one of the 3 other SAIs.
- Possibility to support both 48 kHz and 44.1 kHz streams simultaneously



The STM32H7 embeds 4 SAIs.
Each SAI can receive a kernel clock (SAI_CK_x):

- From DIVQ output of PLL1,
- From DIVP output of PLL2 or PLL3,
- From HSI, CSI or HSE oscillators, or
- From an input PAD: I2S_CKIN.

The kernel clock is used by the SAI in order to generate the timing of the serial audio interface when configured in Master mode.
Note that SAI4 has two independent multiplexers for the kernel clock selection because it is the only one located in the D3 domain.
It is possible to support 48 kHz and 44.1 kHz audio streams in parallel.
The PLLs embedded in the STM32H7 can work in Fractional mode, making the generation of audio frequencies very flexible.

The internal synchronization bus allows the synchronization of several SAI together if needed, in order to support multi-lane devices.

Free protocol modes (1/13)

- The Free protocol mode must be selected to configure the SAI in:
  - I2S Philips Standard
  - I2S MSB or LSB-justified
  - TDM or PCM
- The Free protocol mode is used to adjust the following parameters:
  - Data justification (LSB/MSB first)
  - Data size, Slot (or channel) size
  - Number of slots per frame
  - Data position into a slot
  - Sampling edge of the serial clock
  - Frame size, Frame polarity, Frame period
  - Frame active level size
  - Frame synchronization mode
  - Master/Slave mode
  - Single or multiple or full-duplex data lanes

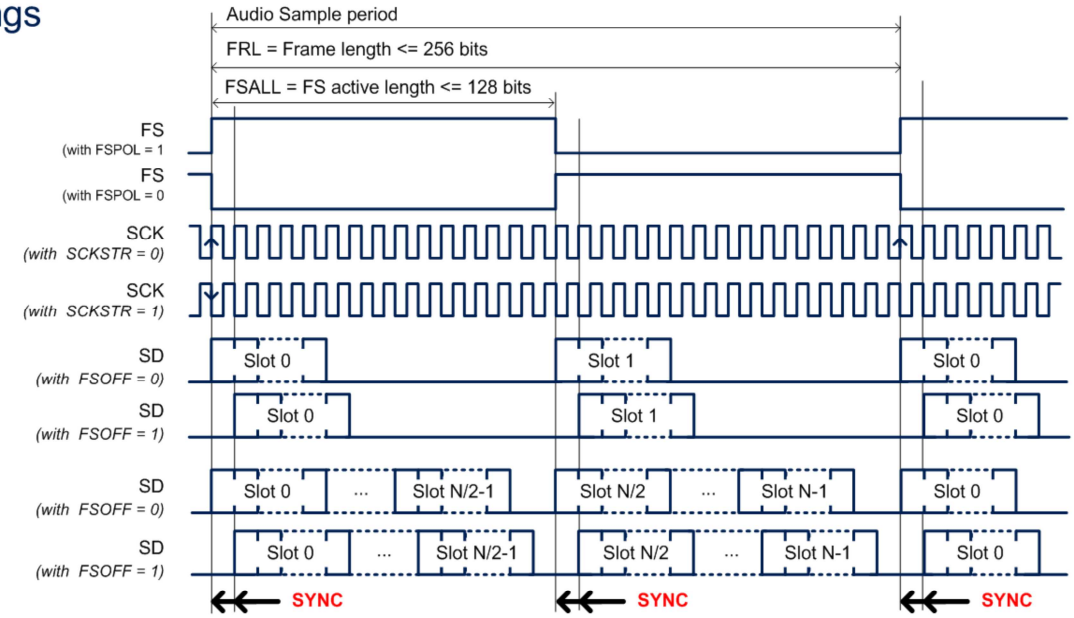
life.augmented

The Free protocol mode makes it possible to emulate most of the common audio standard interfaces thanks to the flexibility of changing the behavior of several parameters such as:

- Data justification,
- Data size and position,
- Frame size,
- Frame period,
- Frame polarity,
- Sampling edge for the clock,
- Number of slots…

Free protocol modes (2/13)

- I2S-like timings

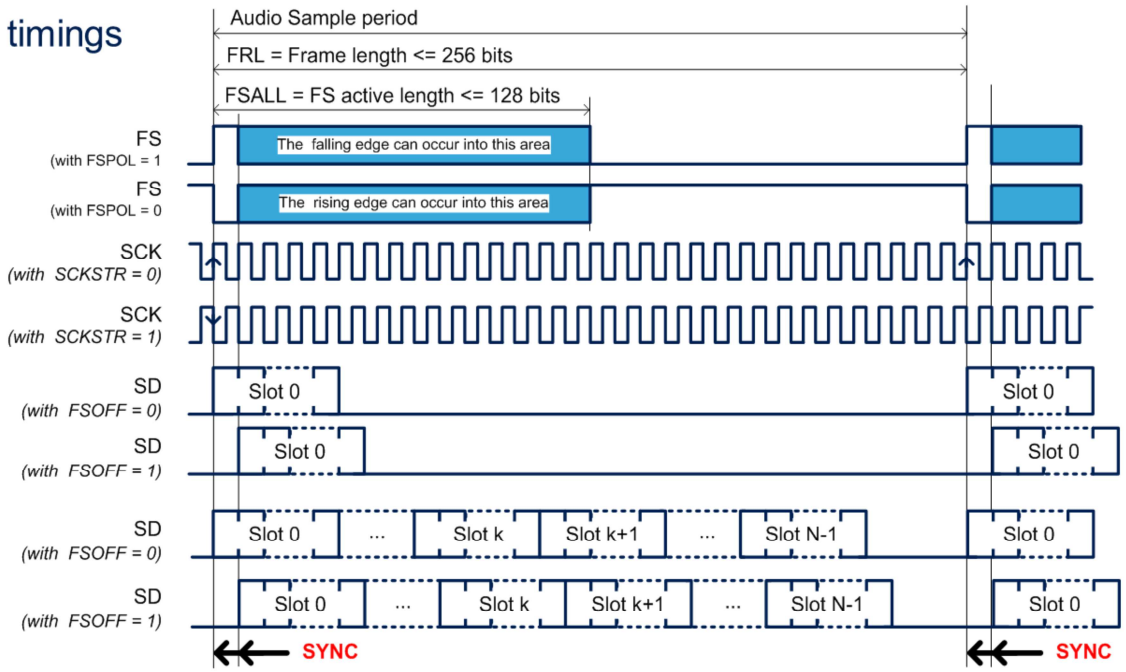


The following example shows some of the possibilities of the interface, for the I2S-like protocols.
In an I2S-like protocol, each edge of the frame synchronization (FS) is used to align the slot positions.

- The frame length, the duty cycle, and polarity can be adjusted.
- The clock data strobe edge can be selected as well.
- The position of the slots with respect to the frame edges can be selected.
- The size of the slots can be also adjusted.
- There must be an even number of slots per frame in I2S-like protocols.

Free protocol modes (3/13)

- TDM-like timings

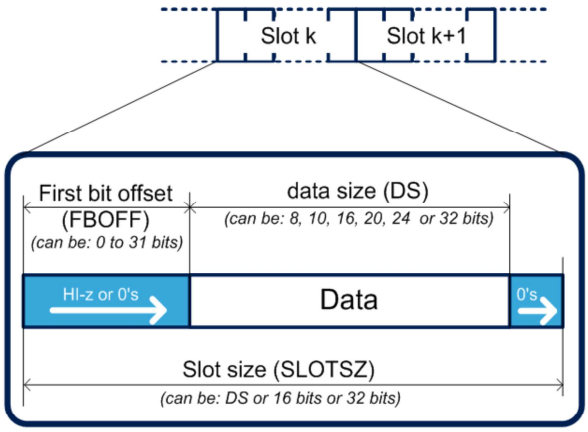


The following example shows some of the possibilities of the interface for the TDM-like protocols. In a TDM-like protocol:

- 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 data strobe edge can be selected,
- The position of the slots with respect to the frame active edge can be selected,
- The size of the slots can be also adjusted,
- The amount of slots per frame (up to 16).

Free protocol modes (4/13)

- Slot configuration:
- Up to 16 slots per audio frame
- Each slot can be defined as active or not
- Possibility to adjust the position of the data within a slot by defining the first bit offset FBOFF.
- Possibility to set the data line in HiZ
  - For inactive slots
  - During FBOFF (First Bit OFFset) area which is used to control the position of the data inside each slot.



life.augmented

The SAI is able to handle up to 16 slots, and each slot can be individually activated or not. The inactive slots can be set in HiZ.
The slot size is always bigger than or equal to the data size.
The SAI allows to control the position of the data inside each slot, and to set the un-used parts of the slots to HiZ if needed.
This function can be helpful when the data line is shared between several devices.

Free protocol modes (5/13)

- Master and Slave modes:
- In Master mode:
  - The SAI provides the timing signals:
    - The bit clock (SCK), the frame synchronization (FS), and the master clock if needed (MCLK)
  - The serial data line (SD) can be in input or output
- In Slave mode:
  - The SAI receives the timing signals from an external device:
    - The bit clock (SCK) and the frame synchronization (FS)
    - The serial data line (SD) can be in input or output

life.augmented

In Master mode, the SAI can generate the master clock (MCLK) depending on the audio system configuration.
This master clock provides a reference clock to the external audio codecs.
In Master mode, the SAI generates the frame synchronization signal (FS) and the bit clock (SCK). The data line SD can be either input or output.

In Slave mode, the MCLK signal is not used.
In Slave mode, the SAI receives the frame synchronization signal (FS) and the bit clock (SCK) from another device (external or internal). The data line SD can be either input or output.

Free protocol modes (6/13)

- Sampling Rate Adjustment
  - The sampling rate must be adjusted in Master mode.
  - The sampling rate adjustment depends on the generation of the master clock (MCLK).
- The master clock (MCLK) is often requested by external audio codec as reference clock.
  - Most of the external audio codecs are sensitive to jitter:
    ➔ The MCLK must be as clean as possible in order to avoid the degradation of audio performance.
  - The MCLK generated by the SAI guarantees a good clock quality.

life.augmented

In Master mode it is up to the SAI to generate the appropriate timings to provide the correct sampling rate.
In Slave mode, the sampling rate is provided by the external audio device.

Free protocol modes (7/13)

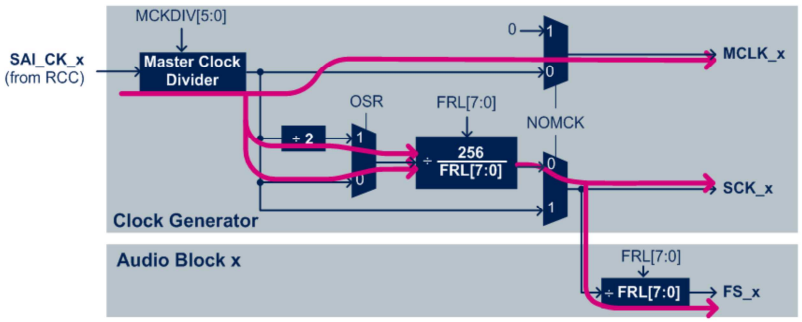
- Sampling Rate Adjustment, when MCLK is generated (NOMCK = 0):

$$f_{MCLK} = \frac{f_{SAI\_CK}}{MCKDIV} \quad (1)$$

$$f_{FS} = \frac{f_{MCLK}}{256 \times (OSR + 1)}$$

$$f_{SCK} = f_{FS} \times (FRL + 1)$$

FRL+1 = 8, 16, 32, 64 128 or 256



f_{MCLK} is the master clock frequency
f_{FS} is the sampling rate frequency (~ frame period)
f_{SCK} is the bit clock frequency

(1) When MCKDIV = 0
$f_{MCLK} = f_{SAI\_CK}$

life.augmented

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, SCK and the FS.

When the master clock (MCLK) is generated, the frame length must be a power of two.
The ratio between the FS frequency and the MCLK frequency is set to 256 or 512, according to the OSR bit.

The clock SAI_CK is provided by the STM32H7's RCC block.

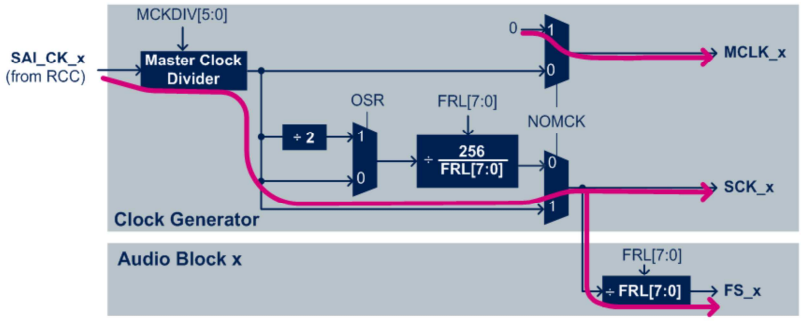
Free protocol modes (8/13)

- Sampling Rate Adjustment, when MCLK is not generated:

$$f_{FS} = \frac{f_{SCK}}{(FRL + 1)}$$

$$f_{SCK} = \frac{f_{SAI\_CK}}{MCKDIV}$$

FRL+1 = any values between 8 and 256



f_{FS} is the sampling rate frequency (~ frame period)
f_{SCK} is the bit clock frequency

life.augmented

When the MCLK is not generated, the frame length can take any value from 8 to 256.
In this case, the frequency of the SCK bit clock is directly given by the clock received on SAI_CK input, divided by the MCKDIV value.

Free protocol modes (9/13)

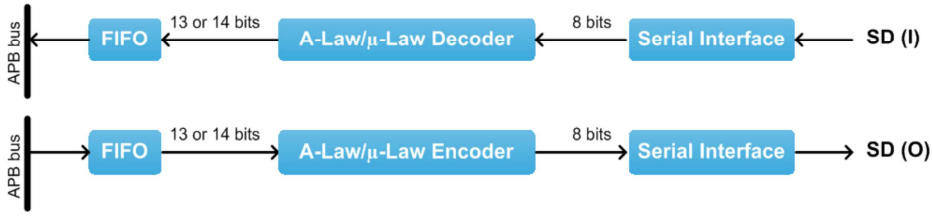
- SAI synchronization
  - The SAI can synchronize its two sub-blocks together (internal synchronization).
  - The SAI is also able to synchronize sub-blocks of different SAIs together (external synchronization).
  - If the synchronization is not used, each sub-block is independent.
    Some examples:
    - SAI_A in I2S Philips Master, SAI_B in SPDIF
    - SAI_A in TDM SLAVE, SAI_B in AC'97
  - If the 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 characteristics.
    - It is not possible to synchronize 2 SAIs using different protocols characteristics.

life.augmented

The internal synchronization can be used for communications needing two data lanes, such as full-duplex I2S.
The external synchronization can be used for communications needing 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.

Free protocol modes (10/13)

- Companding:
  - Companding can be used to reduce the data size on the serial interface to 8 bits.
  - The µ-Law and A-Law formats encode data into 8-bit code elements with MSB alignment.
  - Two companding modes are supported: µ-Law and A-Law which are a part of the CCITT G.711 recommendation
  - The companding standard employed in the United States and Japan is the µ-Law and allows 14 bits of dynamic range.
  - The European companding standard is A-Law and allows 13 bits of dynamic range.



life.augmented

In order to reduce the data size, it is possible to insert in the data path, an A-law or micro-law compander.
Note that A-law and micro-law are not lossless compressors.

Companding modes are generally used in telephony:

- The small values are amplified and the big values are attenuated.
- The SNR tends to be identical for strong and for weak signals.

Free protocol modes (11/13)

- Mute mode
  - In Transmit mode:
    - Can be used to force the transmitted samples to zero, or to repeat the previous transmitted sample
    - Mute mode can be selected at anytime during an on-going frame, and takes effect at the start of the next frame.
    - During Mute mode, the TX-FIFO pointers are still incremented.
  - In Receive mode:
    - Can be used to detect if an amount of consecutive frames have been received with the data in the active slots set to 0.
    - The amount of consecutive frames can be programmed.
    - An interrupt can be generated (if enabled).

life.augmented

The SAI also provides a Mute function.
In Transmit mode, the user can choose to send zeroes on muted slots or the previous transmitted value. The previous transmitted value is limited to configurations having one or two slots per frame.
Note that in Transmit mode, the TxFIFO pointer is still incremented, meaning that data which was present in the FIFO and for which the Mute mode is requested is discarded.

The Receive Mute mode can be helpful to detect an amount of consecutive slots having all data reset to zero.

Free protocol modes (12/13)

- 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 moment: not too late, not too early.
  - Status flag is available and an interrupt can be generated as well.
  - After an anticipated or late frame detection error, the application software has to re-start the SAI.

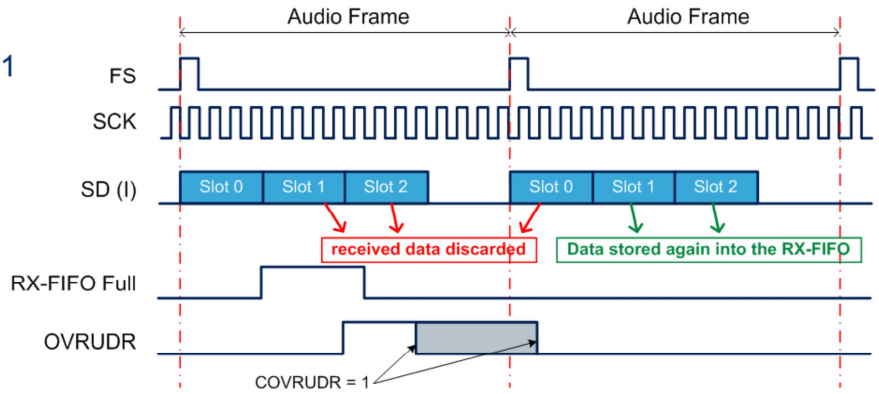
life.augmented

The Anticipated or Late frame error detection function increases the interface's reliability by detecting unexpected frame synchronization misalignment. A status flag is set and an interrupt can be generated as well. The application software will have to then re-start the SAI interface.

Free protocol modes (13/13)

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

- Overrun/Underrun handling
  - Overrun occurs when the RX-FIFO is full, and that a new data coming from the serial interface has to be stored.
  - Underrun occurs when the TX-FIFO is empty, and that a new data is requested by the serial interface.
  - Example: FIFO overrun on Slot 1



life.augmented

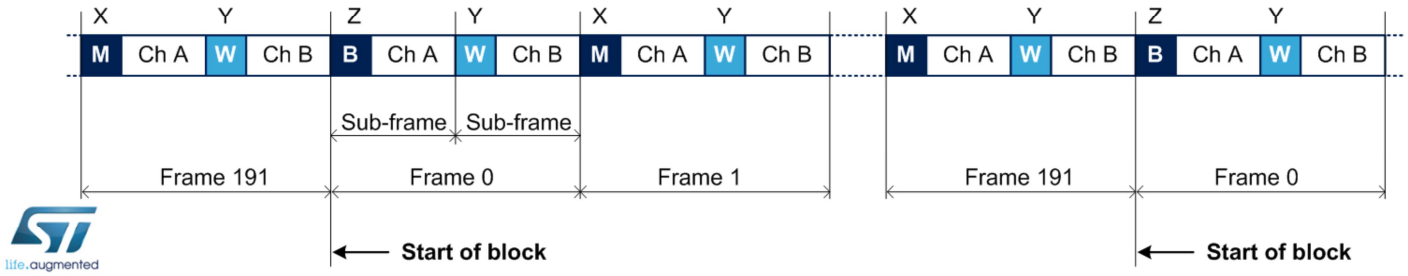
SPDIF protocol (1/4)

- The SAI can generate audio samples using SPDIF protocol:
  - On SPDIF mode, only the SD_x IO is used, other IOs are free
  - The data size is forced to 24 bits.
  - The data are Manchester encoded (or biphase-mark)
  - The SAI generates automatically the preambles
  - The SAI generates automatically the parity
  - The application has to handle the CS, U and V bits

The SAI supports the audio IEC 60958 standard, in Transmit mode when configured for the SPDIF protocol.
The SAI generates the preambles and the parity bit (P) according to the transmitted data.
The software has to handle the CS, U and V bits.

SPDIF protocol (2/4)

- The block structure is used to organize the Channel Status and User information.
  - Each block contains 192 frames
  - Each frame contains 2 sub-frames
  - A preamble allows the detection of the block and sub-frame boundaries
    - Preamble **B** detects the start of new block and the start of a Channel A
    - Preamble **M** detects the start of a Channel A (when it is not a block boundary)
    - Preamble **W** detects the start of a Channel B



In IEC60958 specifications, the block structure is used to decode the Channel Status (CS), and User information (U).

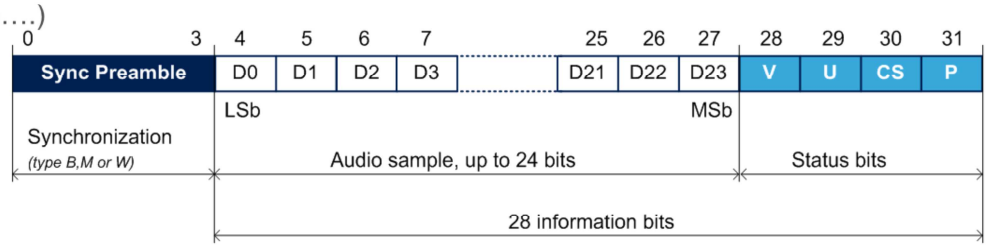
- Each block contains 192 frames
- Each frame contains 2 sub-frames

The SAI automatically generates the B, M and W preambles.

- Preamble B detects the start of new block, and the start of a Channel A
- Preamble M detects the start of a Channel A (when it is not a block boundary)
- Preamble W detects the start of a Channel B

SPDIF protocol (3/4)

- The sub-frame format contains 32 bits divided into 3 fields:
  - The preamble
  - Up to 24-bit data
  - 4 Status bits
    - V is the validity bit, it means that the current sample can be directly converted into an analog signal.
    - P is the parity bit of the received sub-frame, it is used to check the received sub-frame
    - U is the User data channel, each message is composed of 192 bits
    - CS is the Channel Status, each message is composed of 192 bits (i.e. sampling rate, sample length….)



life.augmented

Each sub-frame contains 32 bits divided into 3 fields:

- A synchronization preamble allowing the detection of the block and sub-frame boundaries
- A payload of 24 bits
- Status bits: V, U, CS and P

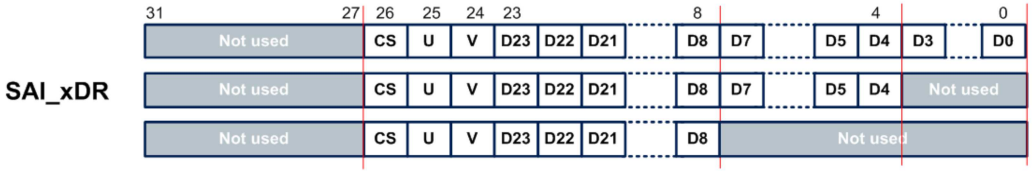
SPDIF protocol (4/4)

- Symbol rate:
  - The audio sample rate (f_S) can be adjusted using the following formula:

$$f_S = \frac{f_{SAI\_CK}}{64}$$

f_{SAI_CK}	Audio sample rate
2.8224 MHz	44.1 kHz
3.072 MHz	48 kHz
6.144 MHz	96 kHz

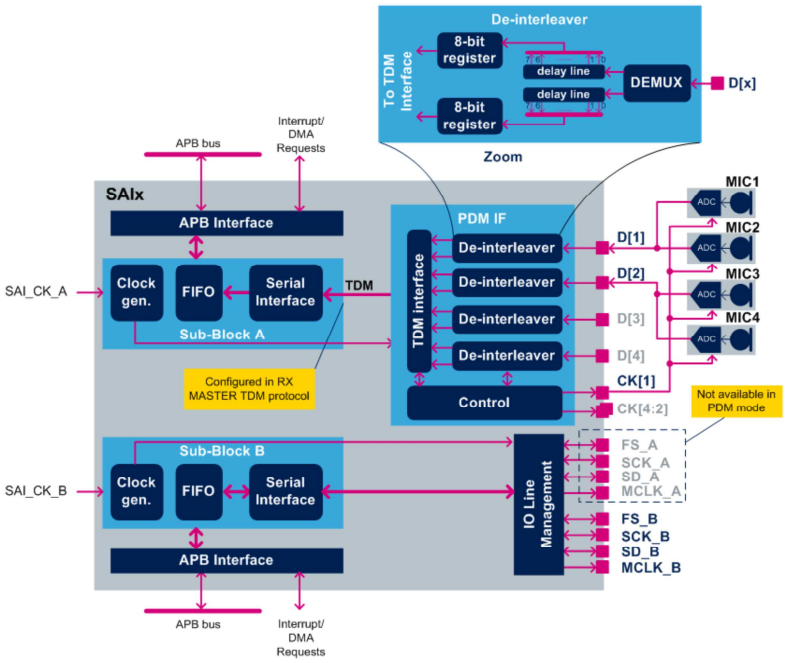
- Data format:
  - The data register must contain CS,U and V bits, plus the data



The Fsai_ck frequency must be adjusted in order to generate the proper audio sample rate (FS).
The data inside the transmit FIFO must be aligned as shown in this slide: the MSB of the data must always be at position 23.

PDM Interface (1/2)

- The PDM interface remaps the bitstream received from the microphones into TDM frames.
- 8-bit delay lines allow beam-forming applications
- Up to 8 microphones can be connected (example given for 4 microphones)
- Sub-Block A must be configured in TDM mode.
- Sub-Block B is free for other applications



The PDM interface remaps the bitstream received from the digital microphones into TDM frames.
The PDM interface waits for the reception of 8 bits from each microphone, before sending a new TDM frame.
In addition, the PDM interface offers an 8-bit delay line for each microphone stream.
These delay lines are working with the resolution of the bitstream clock provided to the microphones.
It enables beamforming applications, and removes constraints on microphone placements.

When the PDM interface is enabled, the serial interface of the sub-block A cannot be used to connect an external device.
This serial interface is connected internally to the PDM interface, and the sub-block A must be configured in TDM mode as an RX MASTER.
The figure shows an example of connection of 4 digital

microphones. Note that each data line D[1], D[2], D[3] or D[4] can be connected to one or two digital microphones.

The sub-block B is still available for other applications, and can be used to connect an external device using TDM, PCM, I2S, or any other supported protocol.

PDM Interface (2/2)

- The frequency of the bit clock (f_{SCK_A}) must be adjusted in order to get the proper sampling frequency of the microphones ($f_{CK[x]}$), according to the formula:

$$f_{SCK\_A} = 2 \times f_{CK[x]} \times (MICNBR + 1)$$

- The frame length must be adjusted according to the number of microphones:

$$FRL = [16 \times (MICNBR + 1)] - 1$$

MICNBR = 0, if 1 or 2 microphones are connected to D[1]
MICNBR = 1, if 3 or 4 microphones are connected to D[1] and D[2]
MICNBR = 2, if 5 or 6 microphones are connected to D[1], D[2] and D[3]
MICNBR = 3, if 7 or 8 microphones are connected to D[1], D[2], D[3] and D[4]

life.augmented

With this PDM interface, the bit clock frequency has to be adjusted according to the sampling frequency and the number of microphones. The frame length is also adjusted according to the number of connected microphones.

AC’97 protocol

- The SAI is able to work as an AC’97 link controller.
  - The number of slots is set to 13:
    - Tag slot: Slot 0 (16-bit),
    - Data slots: Slots 1 to 12 (20-bit)
  - The frame length is fixed at 256 bits

life.augmented

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 set by the hardware.

Interrupts and DMA

- Interrupts:

Interrupt event	Description	How to clear interrupt
FREQ	FIFO request (FIFO threshold reached)	SAI_xDR read or write (2)
OVRRUDR	Overrun/Underrun error	COVRUDR = 1
AFSDET	Anticipated frame sync. detected	CAFSDET = 1
LFSDET	Late frame sync. detected	CLFSDET = 1
CNRDY	Codec Not Ready (only in AC'97 mode)	CCNRDY = 1
WCKCFG	Incorrect frame length configuration (1)	CWCKCFG = 1
MUTEDET	Mute detection	CMUTEDET = 1

(1) When WCKCFG is set to 1, the SAI is automatically disabled (SAIxEN=0)
(2) More precisely, when the FIFO level is below the threshold.

- DMA:
  - DMA requests can be generated when the FIFO threshold is reached

life.augmented

Several events can be enabled in order to generate interrupts.
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 only makes sense in Master mode.

Low-power modes

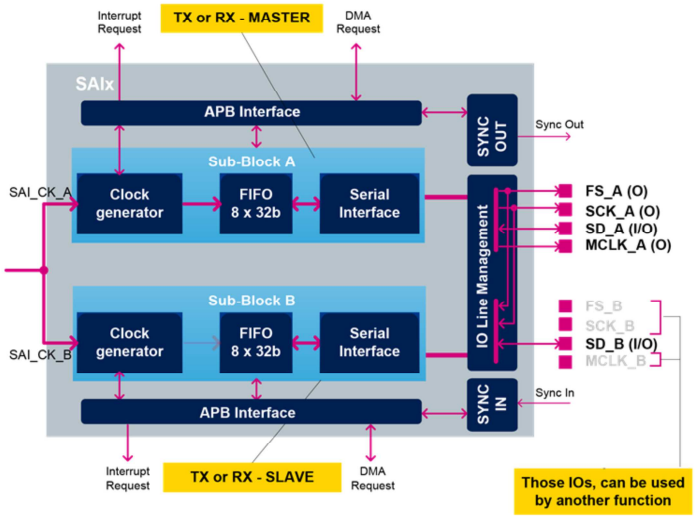
Mode	Description
Run	Active.
Sleep	Active. Peripheral interrupts cause the device to exit Sleep mode.
Stop	Frozen. Peripheral registers content is kept.
Standby	Powered-down. The peripheral must be reinitialized after exiting Standby mode.

life.augmented

The following table shows an overview of the SAI activity for the various possible power modes.
The SAI is active in Run and Sleep modes, frozen in Stop mode or powered-down in Standby mode.
The SAI needs the bus interface clock (APB clock) and the kernel clock (SAI_CK_x) to work properly.

Application examples (1/3)

- Master full-duplex or dual lane:
  - Sub-block A is Master
  - Sub-block B is Slave
  - Sub-block B is synchronized to sub-block A

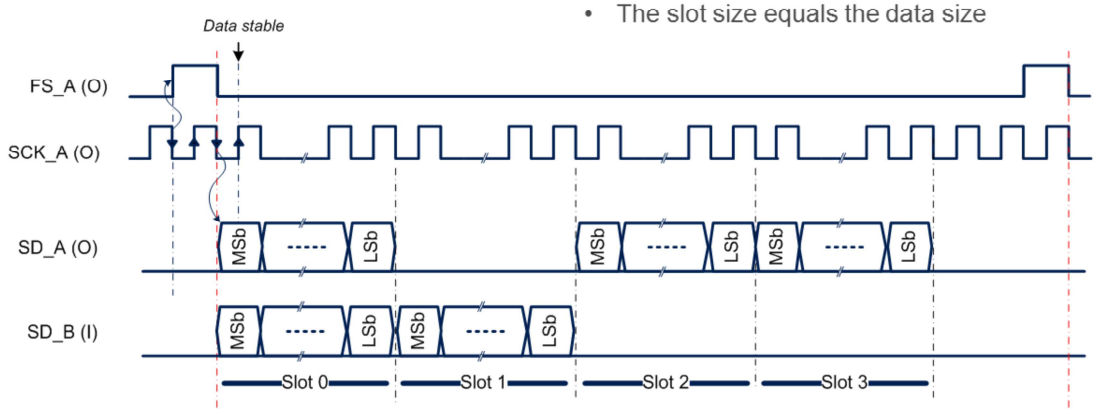


For a full-duplex Master mode, two data lanes are needed, so two sub-blocks need to be used.
The master sub-block A provides the synchronization to the slave sub-block B, using the internal synchronization feature (IO Line Management).

Note that in this example, the sub-block B only uses the SD_B.
The amount of IOs is reduced to its minimum thanks to the internal synchronization.

Application examples (2/3)

- ***TDM Master, 4 slots:***
  - SAI_A programming overview:
    - Master TX mode,
    - 4 slots (NBSLOT = 3), with Slots 0, 2 and 3 activated (SLOTEN = 0x0D)
    - The slot size equals the data size
  - SAI_B programming overview:
    - Slave RX mode,
    - 4 slots (NBSLOT = 3), with Slots 0 and 1 activated (SLOTEN = 0x03)
    - Internal synchronization enabled (SYNCEN = 1)
    - The slot size equals the data size



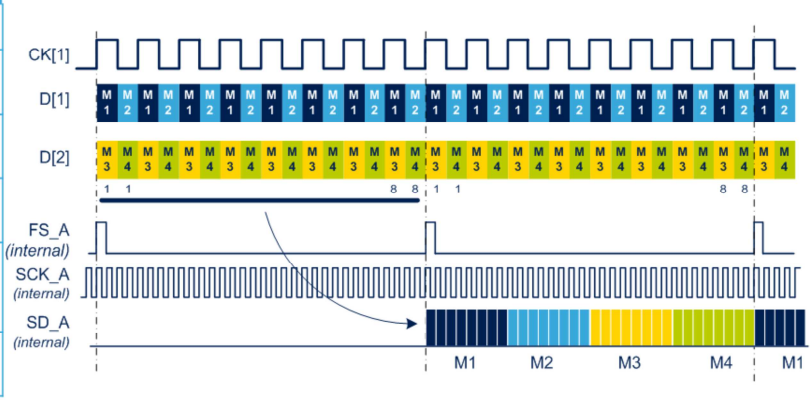
life.augmented

This is another kind of Full-duplex mode, using the TDM protocol.
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.
Sub-block A will generate 3 samples per frame.
Sub-block B will receive 2 samples per frame.

Application examples (3/3)

- **Connecting 4 microphones working at 16 kHz (CK[1] = 1.024 MHz):**
  - SAI_A programming overview:

Description (1)	Fields values
Master RX mode	MODE = 1
TDM protocol, 4 slots of 8 bits	PRTCFG = 0, NBSLOT = 3, SLOTEN = 0xF, SLOTSZ = 0
Slot size of 8 bits, frame length of 32 bits	FRL = 31, FSALL = 0, DS = 2
FS active high, no frame offset, no slot offset	FSPOL = 1, FSOFF = 0, FBOFF = 0
Set the SCK_A clock to 4.096 MHz. No master clock generated.	MCKDIV = 14, NOMCK = 1
PDM configuration: up to 4 microphones, with a single clock	MICNBR = 1, CKEN1 = 1, PDMEN = 1



(1) The kernel clock SAI_CK_A is supposed to be 61.44 MHz

life.augmented

This example shows the most important SAI settings in order to capture the samples provided by 4 digital microphones.
In typical applications, the microphones receive a bitstream clock frequency 64 times higher than the wanted audio rate.
If the application needs to handle a 16 kHz audio stream, then the bitstream clock provided to the digital microphones must be 16 kHz multiplied by 64, which corresponds to a clock frequency of 1.024 MHz.
As there are 4 data streams, the bitclock SCK_A must be 4 times higher than the bitstream clock provided to the microphones, which results in a bitclock frequency of 4.096 MHz.
Using this configuration, the SAI_A writes into its RX FIFO an 8-bit data every time a slot is received.
In order to reconstruct the 16 kHz audio signal, the software has to perform a low-pass filtering of each microphone stream, followed by a decimation by 64.