User Manual

for S32K1_S32M24X I2S Driver

Document Number: UM2I2SASRR21-11 Rev0000R2.0.0 Rev. 1.0

Chapter 1

Revision History

Revision	Date	Author	Description
1.0	04.08.2023	NXP RTD Team	S32K1_S32M24X Real-Time Drivers AUTOSAR 4.4 & R21-11 Version 2.0.0

Chapter 2

Introduction

This User Manual describes the NXP Semiconductor I2s driver for S32K1xx_S32M24x. I2s driver configuration parameters and deviations from the specification are described in Driver chapter of this document. I2s driver requirements and APIs are described in the I2s driver software specification document.

Note: The Sai driver has been renamed to I2s starting with release RTD S32K1 Version 1.0.1. The API of the driver maintains the same functionality as previous releases. However, API names have been changed, beginning with the prefix "I2s_" instead of "Sai_". Existing application code should be updated to be in accordance with the new API naming.

2.1 Supported Derivatives

The software described in this document is intended to be used with the following microcontroller devices of NXP Semiconductors:

- s32k116_qfn32
- s32k116_lqfp48
- s32k118_lqfp48
- s32k118_lqfp64
- s32k142_lqfp48
- s32k142_lqfp64
- s32k142_lqfp100

- s32k142w_lqfp48
- s32k142w_lqfp64
- s32k144_lqfp48
- s32k144_lqfp64 / MWCT1014S_lqfp64
- s32k144_lqfp100 / MWCT1014S_lqfp100
- s32k144_mapbga100
- s32k144w_lqfp48
- s32k144w_lqfp64
- s32k146_lqfp64
- s32k146_lqfp100 / MWCT1015S_lqfp100
- s32k146_mapbga100 / MWCT1015S_mapbga100
- s32k146_lqfp144
- s32k148_lqfp100
- s32k148_mapbga100 / MWCT1016S_mapbga100
- s32k148_lqfp144
- s32k148_lqfp176
- s32m241_lqfp64
- s32m242_lqfp64
- s32m243_lqfp64
- s32m244_lqfp64

All of the above microcontroller devices are collectively named as S32K1_S32M24X. Note: MWCT part numbers contain NXP confidential IP for Qi Wireless Power

2.2 Overview

AUTOSAR (AUTomotive Open System ARchitecture) is an industry partnership working to establish standards for software interfaces and software modules for automobile electronic control systems.

AUTOSAR:

- paves the way for innovative electronic systems that further improve performance, safety and environmental friendliness.
- is a strong global partnership that creates one common standard: "Cooperate on standards, compete on implementation".
- is a key enabling technology to manage the growing electrics/electronics complexity. It aims to be prepared for the upcoming technologies and to improve cost-efficiency without making any compromise with respect to quality.
- facilitates the exchange and update of software and hardware over the service life of the vehicle.

2.3 About This Manual

This Technical Reference employs the following typographical conventions:

- **Boldface** style: Used for important terms, notes and warnings.
- *Italic* style: Used for code snippets in the text. Note that C language modifiers such "const" or "volatile" are sometimes omitted to improve readability of the presented code.

Notes and warnings are shown as below:

Note

This is a note.

Warning

This is a warning

2.4 Acronyms and Definitions

Term	Definition
API	Application Programming Interface
ASM	Assembler
BSMI	Basic Software Make file Interface
CAN	Controller Area Network
C/CPP	C and C++ Source Code
CDD	Complex Device Driver
CS	Chip Select
CTU	Cross Trigger Unit
DEM	Diagnostic Event Manager
DET	Development Error Tracer
DMA	Direct Memory Access
ECU	Electronic Control Unit
FIFO	First In First Out
LSB	Least Signifigant Bit
MCU	Micro Controller Unit
MSB	Most Significant Bit
N/A	Not Applicable
RAM	Random Access Memory
I2S	Inter-IC Sound
SIU	Systems Integration Unit
SWS	Software Specification
XML	Extensible Markup Language

2.5 Reference List

#	Title	Version
1	S32K1 Series Reference Manual	Rev. 14, 09/2021
2	S32M24x Reference Manual	Rev. 2 Draft A, 05/2023
3	Errata S32K116_0N96V	Rev. 22/OCT/2021
4	Errata S32K118_0N97V	Rev. 22/OCT/2021
5	Errata S32K142_0N33V	Rev. 22/OCT/2021
6	Errata S32K144_0N57U	Rev. 22/OCT/2021
7	Errata S32K144W_0P64A	Rev. 22/OCT/2021
8	Errata S32K146_0N73V	Rev. 22/OCT/2021
9	Errata S32K148_0N20V	Rev. 22/OCT/2021
10	S32M242_N33V+P73G	Rev. 0, 6/2023
11	S32M244_P64A+P73G	Rev. 0
12	S32K1xx Data Sheet	Rev. 14, 08/2021
13	S32M2xx Data Sheet	Rev. 3 DraftA — 05/2023

Chapter 3

Driver

3.1 Requirements

I2s is a Complex Device Driver (CDD), so there are no AUTOSAR requirements regarding this module.

It has vendor-specific requirements and implementation.

3.2 Driver Design Summary

The Inter-IC Sound (I2S) driver is implemented as an Autosar complex device driver and provides an interface that supports full-duplex and half-duplex serial interfaces with frame synchronization such as I2S, AC97, TDM, and codec/DSP interfaces. The driver uses the SAI and Flexio I2S peripherals in S32K1XX microcontrollers. The driver can be used to configure the SAI and Flexio I2S peripherals and initiate synchronous and asynchronous data transfers. The peripherals can be configured to operate as a receiver or transmitter, operating in slave and master mode. Hardware and software settings can be configured using an Autosar standard configuration tool. The information required for a I2s data transfer will be configured in a data structure that will be sent as parameter to the API of the driver.

3.3 Hardware Resources

The I2s driver uses the SAI and Flexio I2S hardware IPs. For more details and availablitiy please check the device reference manual.

Note

S32M244 device only has support for FlexIO I2S, so any information regarding SAI peripheral in this documentation should be disregarded.

3.4 Deviations from Requirements

The driver deviates from the AUTOSAR CDD_I2S Driver software specification in some places. The table identifies the AUTOSAR requirements that are not fully implemented, implemented differently, not available, not testable or out of scope for the I2S Driver.

Term	Definition
N/A	Not available
N/T	Not testable
N/S	Out of scope
N/I	Not implemented
N/F	Not fully implemented

Below table identifies the AUTOSAR requirements that are not fully implemented, implemented differently, not available, not testable or out of scope for the driver.

Requirement	Status	Description	Notes
Req_Id	N/S	These requirements are not applicable to this specification.	Not a requirement.

3.5 Driver Limitations

I2s driver limitations:

1. SAI FIFO packing feature is not supported for DMA transfer mode.
2. The misra violations are not fixed regarding the rule 8.13.
3. Flexio I2S is not supported DMA mode.
4. Flexio I2S can only initialize one master or one slave.
5. Driver only validated with basic scenario async request and polling using FlexIO master (board 1) send data to FlexIO slave (board 2)
6. Driver example not generated for S32DS. EBT Driver example using sys_init() instead of Mcu_Init() and Port_Init()

7. Only for *k116 and k118*: When reading I2S over Flexio, the buffer provided to I2s_RequestType::pDataBuffer should be aligned using `VAR_ALIGN`. E.g.

```
VAR_ALIGN(uint16 ReadBuffer[BUFF_SIZE], 16)
I2s_RequestType I2s_ReceiveRequest = {
    .pDataBuffer = (I2s_DataType **)ReadBuffer,
    ...
}
```

3.6 Driver usage and configuration tips

3.6.1 SAI and Flexio I2S initializations

In order to use the I2S driver it must be first initialized in transmit/receive for SAI and master/slave for Flexio I2S for each I2S logical channel that mapping with the desired hardware unit using functions I2s_Init(). Once initialized, it cannot be initialized again until it is de-initialized using I2s_DeInit().

3.6.2 SAI and Flexio I2S data transmission

In each mode (transmit/receive for SAI and master/slave for Flexio I2S) are available two types of transfers:

- Synchronous (blocking): the functions which initiate blocking transfers with configured timeout for transmission. If time expires I2s_SyncTransmit() will return error and the transmission will be aborted.
- Asynchronous (non-blocking): I2s_AsyncTransmit() will return status immediately and complete after the last data is transferred.

Uses I2s_AbortTransmit() to abort or stop the ongoing transfer.
Uses I2s_GetStatus() to get the information of the current transfer.
Uses I2s_GetVersionInfo() to get version information of this module.

3.6.3 I2s over SAI DMA transfer

- DMA module has to be initialized prior to usage in DMA mode; also, DMA channels need to be allocated by the application (the driver only takes care of configuring the DMA channels received in the configuration structure)
- There is a difference in ChannelEnable field usage between interrupt and dma mode:
  In interrupt mode, if mux line is enabled then user must turn on only one bit in ChannelEnable, which will be the data line to output data. Number of data buffers to be muxed is specified in ChannelCount field.
  In DMA mode, if mux line is enabled then user must turn on number of bits equal to number of data buffers to be muxed. The data lines corresponding to these bits will output the same as each other.
  Also in DMA mode, if a mux mode is selected, ChannelEnable must be turned on from bit 0, and immediately aboves (for example turning on bit 0 and bit 2 is not a correct configuration).
- When multiple SAI channel is enabled, number of DMA channels must be equal number of SAI channels. And in DMA component:
  Channel arbitration must be set to "Fixed priority".
  The first DMA channel must have the lowest DMA channel number and request source must be set properly (for example SAI0_TX). Other DMA channels must have request field set to "No request".

3.6.4 I2s over SAI full duplex mode

- Configuration:
  - Initialize both Transmitter and Receiver with one is Master-Asynchronous and another is Slave-↩ Synchronous.
  - Setup different data lines between Transmitter and Receiver (BCLK and FS will be shared).
  - The SAI protocol configuration must be the same between Transmitter and Receiver.
  - Transmit data pins are configured for TDM mode (MASK_TRISTATE - transmit data pins are tri-stated when slots are masked or channels are disabled).
- Transmission:
  - Slave request (non-blocking) should be called first (to prepare data transmission) then Master request shall start the transmission.
  - Master request can be used alone if do not need any action from Slave.

3.6.5 Flexio_I2s communicates with Sai

- Flexio_I2s is Master and Sai is Slave:
  - For Sai, A valid frame sync is also ignored (slave mode) or not generated (master mode) for the first 4 bit clocks but Flexio_I2s does not
  - When Sai receives data from Flexio_I2s, it ignores the falling/rising edge on WS line during the 4 first bit clocks until next falling/rising edge, so depending on Frame Sync polarity, the actual received buffer (by Sai) is less than the transferred buffer size (from Flexio_I2s) 1 or 2 words
  - When Sai transmits data, Sai also ignores the falling/rising edge on WS line and does not generate DATA during 4 first bit clocks until next falling/rising edge Flexio_I2s needs to generate 1 more frame clock, the actual received buffer (by Flexio_i2s) is greater than the transferred buffer (from Sai) 2 words
- Sai is Master and Flexio_I2s is Slave:
  - The above issue does not happen when Sai is master and Flexio_I2s is slave, because the first 4 bit clocks without WS edge do not affect Flexio_I2s, transmission start is just delayed with 4 bit clocks.

3.6.6 SAI auto disable clock

If there is no more request (transmit/receive) added in a long time after, "autoDisableClock" can be set in the last request to disable clock generation until the next request.

3.7 Runtime errors

The driver generates the following DET runtime errors at runtime.

Function	Error Code	Condition triggering the error
I2s_SyncTransmit()	I2S_E_TIMEOUT	Timed out. Ongoing transmission timed out.

3.8 Symbolic Names Disclaimer

All containers having symbolicNameValue set to TRUE in the AUTOSAR schema will generate defines like:

#define <Mip>Conf_<Container_ShortName>_<Container_ID>

For this reason it is forbidden to duplicate the names of such containers across the RTD configurations or to use names that may trigger other compile issues (e.g. match existing `#ifdefs` arguments).

Chapter 4

Tresos Configuration Plug-in

This chapter describes the Tresos configuration plug-in for the driver. All the parameters are described below.

- Module I2s
  - Container GeneralConfiguration
    - Parameter I2sDevErrorDetect
    - Parameter I2sMulticoreSupport
    - Parameter I2sEnableUserModeSupport
    - Parameter I2sVersionInfoApi
    - Parameter I2sTimeoutMethod
    - Parameter I2sTimeoutVal
    - Container I2sSaiSupport
      - Parameter I2sSaiEnable
      - Parameter SaiIpDevErrorDetect
      - Parameter SaiDmaFeature
      - Parameter SaiAutoDisableClockFeature
      - Parameter SaiFifoPackFeature
      - Parameter SaiBitClkSwapSupport
      - Parameter SaiEnableDataMasking
    - Container I2sFlexioSupport
      - Parameter I2sFlexioI2sEnable
      - Parameter FlexioI2sIpDevErrorDetect
      - Parameter FlexioI2sDmaFeature
  - Container I2sConfigSet
    - Container I2sConfiguration
      - Parameter I2sIpType
      - Parameter I2sLogicalChannelId
      - Parameter I2sCallback
      - Reference I2sModuleRef
      - Reference I2sChannelEcucPartitionRef
    - Container SaiHwConfiguration
      - Parameter SaiInstanceId
      - Parameter SaiTxRxConfiguration

- Parameter SaiMode
- Parameter SaiMasterClkSrc
- Parameter SaiSyncType
- Parameter SaiBitClkSwap
- Parameter SaiBitClkNegPolar
- Parameter SaiSampleRate
- Parameter SaiDataLine0
- Parameter SaiDataLine1
- Parameter SaiDataLine2
- Parameter SaiDataLine3
- Parameter SaiFrameSize
- Parameter SaiSyncWidth
- Parameter SaiChannelMode
- Parameter SaiMsbFirst
- Parameter SaiEnableDebug
- Parameter SaiSyncEarly
- Parameter SaiSyncNegPolar
- Parameter SaiWord0Width
- Parameter SaiWordNWidth
- Parameter SaiFirstBitIndex
- Parameter SaiRunErrorReport
- Parameter SaiSyncErrorReport
- Parameter SaiFrameStartReport
- Parameter SaiFifoPackEnable
- Parameter SaiMuxMode
- Parameter SaiTransferType
- Parameter SaiElementSize
- Parameter SaiBufferAccessSize
- Parameter SaiDataLineCount
- Parameter SaiWordFlagIndex
- Parameter SaiBitClkInput
- Parameter SaiContOnErr
- Parameter SaiSyncOnDemand
- Reference SaiClkSrcRef
- Reference SaiDmaChannel0
- Reference SaiDmaChannel1
- Reference SaiDmaChannel2
- Reference SaiDmaChannel3

* Container FlexioHwConfiguration
  - Parameter FlexioI2sInstanceId
  - Parameter FlexioI2sBitsWidth
  - Parameter FlexioI2sMode
  - Parameter FlexioI2sBaudRate
  - Parameter FlexioI2sTransferType
  - Parameter FlexioI2sRxLine
  - Parameter FlexioI2sWsLine
  - Parameter FlexioI2sCallbackParam

        - Reference FlexioI2sClkSrcRef
        - Reference FlexioI2sTxLine
        - Reference FlexioI2sSckLine
        - Reference FlexioI2sDmaTxChannel
        - Reference FlexioI2sDmaRxChannel
    - Container CommonPublishedInformation
        - Parameter ArReleaseMajorVersion
        - Parameter ArReleaseMinorVersion
        - Parameter ArReleaseRevisionVersion
        - Parameter ModuleId
        - Parameter SwMajorVersion
        - Parameter SwMinorVersion
        - Parameter SwPatchVersion
        - Parameter VendorApiInfix
        - Parameter VendorId
    - Container I2sClockReferencePoint
        - Reference I2sClockReference

4.1 Module I2s

Configuration of the I2s module.

Included containers:

- GeneralConfiguration
- I2sConfigSet
- CommonPublishedInformation
- I2sClockReferencePoint

Property	Value
type	ECUC-MODULE-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantSupport	true
supportedConfigVariants	VARIANT-POST-BUILD, VARIANT-PRE-COMPILE

4.2 Container GeneralConfiguration

This container contains the global configuration parameters of the Non-Autosar I2s driver.

Included subcontainers:

- I2sSaiSupport
- I2sFlexioSupport

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

4.3 Parameter I2sDevErrorDetect

Switches the Development Error Detection and Notification ON or OFF.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	true

4.4 Parameter I2sMulticoreSupport

This parameter globally enables the possibility to support multicore. If I2sMulticoreSupport is disabled, then for all the variants no partition shall be defined. If I2sMulticoreSupport is enabled, at least one EcucPartition needs to be defined (in all variants).

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false

Property	Value
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
defaultValue	false

4.5 Parameter I2sEnableUserModeSupport

I2s Enable User Mode Support

When this parameter is enabled, the I2s module will adapt to run from User Mode, with

the following measure : configuring REG_PROT for I2s IPs so that the registers under

protection can be accessed from user mode by setting UAA bit in REG_PROT_GCR to 1

for more information and availability on this platform, please see chapter "User Mode Support" in IM.

True: I2s module will adapt to run from User Mode.

False:I2s module will not apdapt to run from User Mode./p>

If this parameter is not ediatable, that means I2s driver can run in User Mode.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.6 Parameter I2sVersionInfoApi

Adds / removes the service I2s_GetVersionInfo() from the code.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	true

4.7 Parameter I2sTimeoutMethod

Configures the timeout method for I2s.

Based on this selection a certain timeout method from OsIf will be used in the driver.

Note: If OSIF_COUNTER_SYSTEM or OSIF_COUNTER_CUSTOM are selected make sure the corresponding timer is enabled in OsIf General configuration.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	OSIF_COUNTER_DUMMY
literals	['OSIF_COUNTER_DUMMY', 'OSIF_COUNTER_SYSTEM', 'OSIF_COU↩ NTER_CUSTOM']

4.8 Parameter I2sTimeoutVal

This is a timeout (microseconds) value which is used to wait for each synchronization transfer

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	1000
max	4294967295
min	1

4.9 Container I2sSaiSupport

Vendor specific: Container for the I2s Sai related configuration parameters.

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

4.10 Parameter I2sSaiEnable

Vendor specific: Enable/Disable Sai support.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF

Property	Value
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
defaultValue	false

4.11 Parameter SaiIpDevErrorDetect

Switches the Development Error Detection and Notification ON or OFF.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	true

4.12 Parameter SaiDmaFeature

Enable DMA feature for transferring and Notification ON or OFF.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A

Property	Value
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.13 Parameter SaiAutoDisableClockFeature

Sai Auto Disable Clock Feature. When enabled, allows the driver to disable clock generation after transmission completed or aborted. The clock generation will be started again at next transmission.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.14 Parameter SaiFifoPackFeature

Enable FIFO packing feature. When this parameter is enabled, the Sai FIFO Packing Enable can be configurated.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.15 Parameter SaiBitClkSwapSupport

Sai Enable Bit Clock Swap Support

When this parameter is enabled, the Sai Bit Clk Swap can be configurated.

If this parameter is not ediatable, that means this feature is not available on this device.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.16 Parameter SaiEnableDataMasking

Enable to allow masking of sent or received data. Data will be masked according to the SaiChannelMode selection.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.17 Container I2sFlexioSupport

Vendor specific: Container for the I2s Flexio related configuration parameters.

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

4.18 Parameter I2sFlexioI2sEnable

Vendor specific: Enable/Disable Flexio support.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
defaultValue	true

4.19 Parameter FlexioI2sIpDevErrorDetect

Switches the Development Error Detection and Notification ON or OFF.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A

Property	Value
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	true

4.20 Parameter FlexioI2sDmaFeature

Enable DMA feature for transferring and Notification ON or OFF.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.21 Container I2sConfigSet

This container contains the configuration parameters and sub containers of the I2s module.

Included subcontainers:

- I2sConfiguration
- SaiHwConfiguration
- FlexioHwConfiguration

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

4.22 Container I2sConfiguration

This container contains the configuration (parameters) of the I2s Controller(s).

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	Infinite
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE

4.23 Parameter I2sIpType

Selects Hardware IP.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	SAI
literals	['SAI', 'FLEXIO_I2S']

4.24 Parameter I2sLogicalChannelId

Identifies the I2s Logical channel ID coresponding with I2s configuration.

Property	Value
type	ECUC-INTEGER-PARAM-DEF

Property	Value
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	1
max	255
min	0

4.25 Parameter I2sCallback

I2s callback. This function will be called for all I2s events.

Property	Value
type	ECUC-FUNCTION-NAME-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
defaultValue	NULL_PTR

4.26 Reference I2sModuleRef

Maps an I2s channel to zero or one ECUC partition to limit the access to this channel group. The ECUC partitions referenced are a subset of the ECUC partitions where the I2s driver is mapped to.

Property	Value
type	ECUC-CHOICE-REFERENCE-DEF
origin	NXP
lowerMultiplicity	1
upperMultiplicity	1

Property	Value
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
requiresSymbolicNameValue	False
destinations	['/TS_T40D2M20I0R0/I2s/I2sConfigSet/SaiHwConfiguration', '/TS_T40D2↩ M20I0R0/I2s/I2sConfigSet/FlexioHwConfiguration']

4.27 Reference I2sChannelEcucPartitionRef

Maps a I2s channel to zero or one ECUC partition to limit the access to this channel group. The ECUC partitions referenced are a subset of the ECUC partitions where the I2s driver is mapped to.

Property	Value
type	ECUC-REFERENCE-DEF
origin	AUTOSAR_ECUC
lowerMultiplicity	0
upperMultiplicity	Infinite
postBuildVariantMultiplicity	true
multiplicityConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/AUTOSAR/EcucDefs/EcuC/EcucPartitionCollection/EcucPartition

4.28 Container SaiHwConfiguration

Configuration of a Sai module available on the platform.

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	0

Property	Value
upperMultiplicity	4
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE

4.29 Parameter SaiInstanceId

Identifies the Sai Instance.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	true
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	1
min	0

4.30 Parameter SaiTxRxConfiguration

Selects Tx or Rx configuration.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	Transmitter
literals	['Transmitter', 'Receiver']

4.31 Parameter SaiMode

Master or Slave.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	Master
literals	['Master', 'Slave']

4.32 Parameter SaiMasterClkSrc

Select master clock source.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	UNSUPPORTED
literals	['UNSUPPORTED']

4.33 Parameter SaiSyncType

SAI run in sync or async mode.

ASYNC: Independent clock.

SYNC_WITH_OTHER: Bit clock and frame sync signal is taken from transmitter/receiver.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	ASYNC
literals	['ASYNC', 'SYNC_WITH_OTHER']

4.34 Parameter SaiBitClkSwap

This field swaps the bit clock used by the transmitter. When the transmitter is configured in asynchronous mode and this bit is set, the transmitter is clocked by the receiver bit clock (RX_BCLK). This allows the transmitter and receiver to share the same bit clock, but the transmitter continues to use the transmit frame sync (TX_SYNC).

When the transmitter is configured in synchronous mode, the transmitter BCS field and receiver BCS field must be set to the same value. When both are set, the transmitter and receiver are both clocked by the transmitter bit clock (TX_BCLK) but use the receiver frame sync (RX_SYNC).

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.35 Parameter SaiBitClkNegPolar

True if bit clock is negative polar (active low with drive outputs on falling edge and sample inputs on rising edge), false otherwise.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.36 Parameter SaiSampleRate

Sample rate for the Sai module.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	16000
max	4294967295
min	0

4.37 Parameter SaiDataLine0

Enable or disable Sai Data Line 0.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	true

4.38 Parameter SaiDataLine1

Enable or disable Sai Data Line 1.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.39 Parameter SaiDataLine2

Enable or disable Sai Data Line 2.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1

Property	Value
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.40 Parameter SaiDataLine3

Enable or disable Sai Data Line 3.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.41 Parameter SaiFrameSize

Frame size in number of words. When in mux line mode, the number of unmasked word in a frame must be divisible by number of channels.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD

Property	Value
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	1
max	16
min	1

4.42 Parameter SaiSyncWidth

Active sync width in number of bit clocks (must not longer than first word width).

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	1
max	32
min	1

4.43 Parameter SaiChannelMode

Configures if transmit data pins are configured for TDM mode or Output mode.

MASK_TRISTATE - TDM mode, transmit data pins are tri-stated when slots are masked or channels are disabled.

MASK_ZERO - Output mode, transmit data pins are never tri-stated and will output zero when slots are masked or channels are disabled.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A

Property	Value
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	MASK_TRISTATE
literals	['MASK_TRISTATE', 'MASK_ZERO']

4.44 Parameter SaiMsbFirst

True if data is MSB first, false if LSB first.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	MSB
literals	['MSB', 'LSB']

4.45 Parameter SaiEnableDebug

Enables/disables transmitter or receiver operation in Debug mode.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD

Property	Value
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.46 Parameter SaiSyncEarly

True if frame sync is one bit clock early.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.47 Parameter SaiSyncNegPolar

True if frame sync is negative polarity (active low), false otherwise.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.48 Parameter SaiWord0Width

Configures the number of bits in the first word in each frame.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	16
max	32
min	1

4.49 Parameter SaiWordNWidth

Configures the number of bits in each word except the first word in the frame.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	16
max	32
min	8

4.50 Parameter SaiFirstBitIndex

Index from LSB of first bit to be transmitted/received, valid range from 0 to 31.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	1
max	31
min	0

4.51 Parameter SaiRunErrorReport

Execute the callback function on underrun/overrun event.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.52 Parameter SaiSyncErrorReport

Execute the callback function when an error on external frame sync has occured.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false

Property	Value
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.53 Parameter SaiFrameStartReport

Execute the callback function on frame start event event.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.54 Parameter SaiFifoPackEnable

Sai FIFO Packing Enable

Enable FIFO packing base on element size:

1-byte: 8-bit FIFO packing is enabled.

2-byte: 16-bit FIFO packing is enabled.

4-byte: FIFO packing is disabled.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP

Property	Value
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	false

4.55 Parameter SaiMuxMode

Mux type.

MUX_DISABLED: Each data line is a channel, uses a seperate memory block.

MUX_LINE: Only one data line (data line 0 for DMA mode) is used. Words on data line is alternated between channels, each channel data is a seperate memory block.

MUX_MEM: Words in memory block is alternated between channels, each channel data is on a seperate data line.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	MUX_DISABLED
literals	['MUX_DISABLED', 'MUX_LINE', 'MUX_MEM']

4.56 Parameter SaiTransferType

Select the Sai transfer type, using Interrupt or DMA.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	INTERRUPT
literals	['INTERRUPT', 'DMA']

4.57 Parameter SaiElementSize

Size in bytes of each read/write from/to buffer and should not be less than Sai Word Width.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	2
max	4
min	1

4.58 Parameter SaiBufferAccessSize

Number of read/write operations done on FIFO during one interrupt/dma operation. Maximum value is given by FIFO depth.

When FIFO packing is disabled, this value is equal with number of words each time read from/write to buffer.

If FIFO packing is enabled, the driver will actually read from/write to buffer the number multiplied by number of words packed into 1 FIFO read/write operation(example: N, with 1byte words and packing => N*(32/8) elements read from buffer per operation).

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	4
max	8
min	1

4.59 Parameter SaiDataLineCount

Number of Sai Data Lines to enable, only used when both line mux mode and interrupt mode is selected.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	2
max	4
min	2

4.60 Parameter SaiWordFlagIndex

Configure which word sets the start of word flag, should less than frame size.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP

Property	Value
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	15
min	0

4.61 Parameter SaiBitClkInput

Use slave mode timing in datasheet for master mode.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.62 Parameter SaiContOnErr

True continue transfering the same work that caused FIFO eror, False start the next frame.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1

Property	Value
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	true

4.63 Parameter SaiSyncOnDemand

True frame sync is generated only when fifo is not empty (transmit) or not full (receive), False otherwise.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	true

4.64 Reference SaiClkSrcRef

Reference to the I2sClockReferencePoint from which the clock is derived.

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
requiresSymbolicNameValue	False
destination	/TS_T40D2M20I0R0/I2s/I2sClockReferencePoint

4.65 Reference SaiDmaChannel0

Specifies channel 0 DMA channel used.

Property	Value
type	ECUC-CHOICE-REFERENCE-DEF
origin	NXP
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
requiresSymbolicNameValue	False
destinations	['/AUTOSAR/EcucDefs/Mcl/MclConfig/dmaLogicChannel_Type']

4.66 Reference SaiDmaChannel1

Specifies channel 1 DMA channel used.

Property	Value
type	ECUC-CHOICE-REFERENCE-DEF
origin	NXP
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
requiresSymbolicNameValue	False
destinations	['/AUTOSAR/EcucDefs/Mcl/MclConfig/dmaLogicChannel_Type']

4.67 Reference SaiDmaChannel2

Specifies channel 2 DMA channel used.

Property	Value
type	ECUC-CHOICE-REFERENCE-DEF
origin	NXP
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
requiresSymbolicNameValue	False
destinations	['/AUTOSAR/EcucDefs/Mcl/MclConfig/dmaLogicChannel_Type']

4.68 Reference SaiDmaChannel3

Specifies channel 3 DMA channel used.

Property	Value
type	ECUC-CHOICE-REFERENCE-DEF
origin	NXP
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
requiresSymbolicNameValue	False
destinations	['/AUTOSAR/EcucDefs/Mcl/MclConfig/dmaLogicChannel_Type']

4.69 Container FlexioHwConfiguration

Configuration of a Flexio module available on the platform.

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	0
upperMultiplicity	Infinite
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE

4.70 Parameter FlexioI2sInstanceId

Identifies the Flexio Channel.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	true
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	1
max	3
min	0

4.71 Parameter FlexioI2sBitsWidth

Number of bytes in a word - Support only 1 byte,2 byte,4 byte.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true

Property	Value
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	1
max	4
min	1

4.72 Parameter FlexioI2sMode

Master or Slave.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	Master
literals	['Master', 'Slave']

4.73 Parameter FlexioI2sBaudRate

The actual baud rate in Hz that can be set by the driver. May be substantially different than the requested baud rate if the frequency of the Flexio input clock is too high or too low.

For best results it is recommended to use an input clock with a frequency 200-500 times greater than the desired baud rate

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

Property	Value
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	20000
max	4294967295
min	1

4.74 Parameter FlexioI2sTransferType

Select the Flexio I2s transfer type, using Interrupt or Polling or DMA.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	INTERRUPT
literals	['INTERRUPT', 'POLLING', 'DMA']

4.75 Parameter FlexioI2sRxLine

FlexioI2sRxLine is enabled indirectly via the label node 'Enable Flexio Add pin' of Mcl component referenced from FlexioI2sTxLine.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD

Property	Value
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	true

4.76 Parameter FlexioI2sWsLine

FlexioI2sWsLine is enabled indirectly via the label node 'Enable Flexio Add pin' of Flexio_Mcl_Ip component referenced from FlexioI2sSckLine.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	true

4.77 Parameter FlexioI2sCallbackParam

FlexioI2s callback Param. This param will be called for all FlexioI2s events.

Property	Value
type	ECUC-FUNCTION-NAME-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
defaultValue	NULL_PTR

4.78 Reference FlexioI2sClkSrcRef

Reference to the I2sClockReferencePoint from which the clock is derived.

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
requiresSymbolicNameValue	False
destination	/TS_T40D2M20I0R0/I2s/I2sClockReferencePoint

4.79 Reference FlexioI2sTxLine

Reference to the FLEXIO logic channel, which is set in the Mcl driver configuration.

In Master mode, Channel N is refered in TX line and Channel N+1 is refered in SCK line.

In Slave mode, Channel N is refered in TX line and Channel N+1/N+2 are refered in SCK line.

Note: Implementation Specific Parameter.

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/TS_T40D2M20I0R0/Mcl/MclConfig/FlexioCommon/FlexioMclLogicChannels

4.80 Reference FlexioI2sSckLine

Reference to the FLEXIO logic channel, which is set in the Mcl driver configuration.

In Master mode, Channel N is refered in TX line and Channel N+1 is refered in SCK line.

In Slave mode, Channel N is refered in TX line and Channel N+1/N+2 are refered in SCK line.

Note: Implementation Specific Parameter.

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/TS_T40D2M20I0R0/Mcl/MclConfig/FlexioCommon/FlexioMclLogicChannels

4.81 Reference FlexioI2sDmaTxChannel

Specifies Flexio I2s Tx Dma Channel used.

Property	Value
type	ECUC-CHOICE-REFERENCE-DEF
origin	NXP
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
requiresSymbolicNameValue	False
destinations	['/AUTOSAR/EcucDefs/Mcl/MclConfig/dmaLogicChannel_Type']

4.82 Reference FlexioI2sDmaRxChannel

Specifies Flexio I2s Rx Dma Channel used.

Property	Value
type	ECUC-CHOICE-REFERENCE-DEF
origin	NXP
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
requiresSymbolicNameValue	False
destinations	['/AUTOSAR/EcucDefs/Mcl/MclConfig/dmaLogicChannel_Type']

4.83 Container CommonPublishedInformation

Common container, aggregated by all modules. It contains published information about vendor and versions.

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

4.84 Parameter ArReleaseMajorVersion

Major version number of AUTOSAR specification on which the appropriate implementation is based on.

Property	Value
type	ECUC-INTEGER-PARAM-DEF

Property	Value
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	4
max	4
min	4

4.85 Parameter ArReleaseMinorVersion

Minor version number of AUTOSAR specification on which the appropriate implementation is based on.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	7
max	7
min	7

4.86 Parameter ArReleaseRevisionVersion

Revision version number of AUTOSAR specification on which the appropriate implementation is based on.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false

Property	Value
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	0
max	0
min	0

4.87 Parameter ModuleId

Module ID of this module from Module List.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	255
max	255
min	255

4.88 Parameter SwMajorVersion

Major version number of the vendor specific implementation of the module. The numbering is vendor specific.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1

Property	Value
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	2
max	2
min	2

4.89 Parameter SwMinorVersion

Minor version number of the vendor specific implementation of the module. The numbering is vendor specific.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	0
max	0
min	0

4.90 Parameter SwPatchVersion

Patch level version number of the vendor specific implementation of the module. The numbering is vendor specific.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A

Property	Value
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	0
max	0
min	0

4.91 Parameter VendorApiInfix

In driver modules which can be instantiated several times on a single ECU, BSW00347 requires that the name of APIs is extended by the VendorId and a vendor specific name.

This parameter is used to specify the vendor specific name. In total, the implementation specific name is generated as follows:

<ModuleName>_>VendorId>_<VendorApiInfix>.

E.g. assuming that the VendorId of the implementor is 123 and the implementer chose a VendorApiInfix of "v11r456" a api name Can_Write defined in the SWS will translate to Can_123_v11r456Write.

This parameter is mandatory for all modules with upper multiplicity > 1. It shall not be used for modules with upper multiplicity =1.

Property	Value
type	ECUC-STRING-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	

4.92 Parameter VendorId

Vendor ID of the dedicated implementation of this module according to the AUTOSAR vendor list.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	43
max	43
min	43

4.93 Container I2sClockReferencePoint

This container contains a parameter, which represents a reference to a container of the type McuClockReferencePoint (defined in module MCU).

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	Infinite
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE

4.94 Reference I2sClockReference

Reference to a container of the type McuClockReferencePoint, to select an input clock.

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
lowerMultiplicity	1

Property	Value
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/AUTOSAR/EcucDefs/Mcu/McuModuleConfiguration/McuClockSetting↩ Config/McuClockReferencePoint

This chapter describes the Tresos configuration plug-in for the I2S Driver. The most of the parameters are described below.

Chapter 5

Module Index

5.1 Software Specification

Here is a list of all modules:

Chapter 6

Module Documentation

6.1 I2s Driver

6.1.1 Detailed Description

Data Structures

- struct I2s_StatusStructType

  Definition of the type of transfer status. *More...*
- struct I2s_ConfigType

  This type contains initialization data. *More...*
- struct I2s_RequestType

  Definition for Request Buffer. This is the structure which is passed to I2s_SyncTransmit or I2s_AsyncTransmit function. This holds the necessary information required for the communication of I2S Hw with the Slave device. *More...*
- struct I2s_Ipw_ConfigType

  This structure contains config structures for all IPs available. *More...*
- struct I2s_HwConfigType

  This structure holds the HWUnit configuration parameters. *More...*
- struct I2s_LogicalConfigState

  This structure holds the Logical configuration parameters. *More...*

Macros

- #define I2S_E_UNINIT

  API service used without I2s module initialization.
- #define I2S_E_BUSY

  I2s module is busy with a running operation.
- #define I2S_E_TIMEOUT

  I2s module is timeout with a running operation.
- #define I2S_E_INIT_FAILED

  I2s module is not properly initialized.

- #define I2S_E_PARAM_POINTER

  API service is called using an invalid pointer (e.g. the pointer should not be NULL).
- #define I2S_E_PARAM_CHANNEL

  API service is called using an invalid logical channel.
- #define I2S_E_PARAM_CONFIG

  API service is called using an invalid core.
- #define I2S_E_INCORRECT_DIRECTION

  I2s module is not properly direction initialized.
- #define I2S_INIT_ID

  API service ID for I2s_Init function.
- #define I2S_DEINIT_ID

  API service ID for I2s_DeInit function.
- #define I2S_SYNCTRANSMIT_ID

  API service ID for I2s_SyncTransmit function.
- #define I2S_ASYNCTRANSMIT_ID

  API service ID for I2s_AsyncTransmit function.
- #define I2S_GETSTATUS_ID

  API service ID for I2s_GetStatus function.
- #define I2S_ABORTTRANSMIT_ID

  API service ID for I2s_AbortTransmit function.
- #define I2S_GETVERSIONINFO_ID

  API service ID for I2s_GetVersionInfo function.
- #define SAI_IP_FPACK_8_BIT
- #define SAI_IP_FPACK_16_BIT
- #define FEATURE_SAI_IP_HAS_CHMOD

  Channel mode tristate or output zero.
- #define FEATURE_SAI_IP_MSEL_BUS_CLK

  Master clock from Bus clock.
- #define FEATURE_SAI_IP_MSEL_SOSC_DIV1_CLK

  Master clock from SOSC_DIV1 clock.
- #define FEATURE_SAI_IP_MSEL_MCLK_PIN

  Master clock from MCLK pin.
- #define FEATURE_SAI_IP_MSEL_OTHER_MCLK_PIN

  Master clock from MCLK pin from other SAI instance.
- #define FEATURE_SAI_IP_HAS_PARAM

  SAI has PARAM register.
- #define SAI0_IP_CHANNEL_COUNT

  SAI channel count.
- #define SAI_IP_NULL_PTRS

  SAI NULL PTRs.

Types Reference

- typedef uint8 I2s_HwUnitCount

  This gives the number of I2S configuration.
- typedef uint8 I2s_DataType

  This gives the transfer data type.
- typedef uint16 I2s_WordMaskType

  I2s Word Mask for transmitted data.
- typedef uint8 I2s_LogicalChannel

  This gives the logical channel.

Enum Reference

- enum I2s_DataDirectionType

  Definition of the type of transfer direction.
- enum I2s_StatusType

  Definition for different state and errors of Operation Status.
- enum I2s_TransmitReceiveType

  Definition of the transmitter/recevier of a SAI hw unit.
- enum I2s_ModeType

  Definition of the master/slave mode.
- enum I2s_SupportedIPsType

  This enum contains all IPs which can integrate SAI functionalities.

Function Reference

- void I2s_Init (const I2s_ConfigType *const pConfigPtr)

  Initializes the I2S module.
- void I2s_DeInit (void)

  DeInitializes the I2S module.
- Std_ReturnType I2s_SyncTransmit (I2s_LogicalChannel LogicChn, const I2s_RequestType *const pRequest)

  Sends or receives an I2S message blocking.
- Std_ReturnType I2s_AsyncTransmit (I2s_LogicalChannel LogicChn, const I2s_RequestType *const p↩ Request)

  Starts an asynchronous transmission on the I2S bus.
- Std_ReturnType I2s_GetStatus (I2s_LogicalChannel LogicChn, I2s_StatusStructType *const pStatus)

  Gets the status of an I2S instance.
- void I2s_AbortTransmit (I2s_LogicalChannel LogicChn)

  Aborts the I2s ongoing transmission.

6.1.2 Data Structure Documentation

6.1.2.1 struct I2s_StatusStructType

Definition of the type of transfer status.

Definition at line 146 of file CDD_I2s_Types.h.

6.1.2.2 struct I2s_ConfigType

This type contains initialization data.

This contains initialization data for the I2S driver. It shall contain:

- The number of I2S modules to be configured
- I2S dependent properties for used HW units

Definition at line 167 of file CDD_I2s_Types.h.

Data Fields

Type	Name	Description
const I2s_HwUnitCount	cfgCount	Number of I2S configurations. < Pointer to I2s hardware unit configuration array
const I2s_HwConfigType *const *	pHwConfig	

6.1.2.3 struct I2s_RequestType

Definition for Request Buffer. This is the structure which is passed to I2s_SyncTransmit or I2s_AsyncTransmit function. This holds the necessary information required for the communication of I2S Hw with the Slave device.

Definition at line 181 of file CDD_I2s_Types.h.

Data Fields

Type	Name	Description
uint32	bufferSize	Buffer Size : The number of words for reading or writing of each channel. The size of each buffer word element is "I2s Buffer Element Size" selected in configurator. < Direction of the data. Can be either Send or Receive.
I2s_DataDirectionType	dataDirection	
I2s_DataType **	pDataBuffer	Buffer to Store or to transmit Serial data. <

6.1.2.4 struct I2s_Ipw_ConfigType

This structure contains config structures for all IPs available.

Definition at line 193 of file I2s_Ipw_Types.h.

6.1.2.5 struct I2s_HwConfigType

This structure holds the HWUnit configuration parameters.

Definition at line 208 of file I2s_Ipw_Types.h.

6.1.2.6 struct I2s_LogicalConfigState

This structure holds the Logical configuration parameters.

Definition at line 223 of file I2s_Ipw_Types.h.

6.1.3 Macro Definition Documentation

6.1.3.1 I2S_E_UNINIT

```
#define I2S_E_UNINIT
```

API service used without I2s module initialization.

Development errors. The following errors shall be detectable by the I2S module depending on its configuration (development / production mode).

All error codes

Definition at line 135 of file CDD_I2s.h.

6.1.3.2 I2S_E_BUSY

```
#define I2S_E_BUSY
```

I2s module is busy with a running operation.

Definition at line 140 of file CDD_I2s.h.

6.1.3.3 I2S_E_TIMEOUT

```
#define I2S_E_TIMEOUT
```

I2s module is timeout with a running operation.

Definition at line 145 of file CDD_I2s.h.

6.1.3.4 I2S_E_INIT_FAILED

```
#define I2S_E_INIT_FAILED
```

I2s module is not properly initialized.

Definition at line 150 of file CDD_I2s.h.

6.1.3.5 I2S_E_PARAM_POINTER

```
#define I2S_E_PARAM_POINTER
```

API service is called using an invalid pointer (e.g. the pointer should not be NULL).

Definition at line 155 of file CDD_I2s.h.

6.1.3.6 I2S_E_PARAM_CHANNEL

```
#define I2S_E_PARAM_CHANNEL
```

API service is called using an invalid logical channel.

Definition at line 160 of file CDD_I2s.h.

6.1.3.7 I2S_E_PARAM_CONFIG

```
#define I2S_E_PARAM_CONFIG
```

API service is called using an invalid core.

Definition at line 165 of file CDD_I2s.h.

6.1.3.8 I2S_E_INCORRECT_DIRECTION

```
#define I2S_E_INCORRECT_DIRECTION
```

I2s module is not properly direction initialized.

Definition at line 170 of file CDD_I2s.h.

6.1.3.9 I2S_INIT_ID

```
#define I2S_INIT_ID
```

API service ID for I2s_Init function.

All AUTOSAR API's service IDs

Definition at line 179 of file CDD_I2s.h.

6.1.3.10 I2S_DEINIT_ID

```
#define I2S_DEINIT_ID
```

API service ID for I2s_DeInit function.

Definition at line 184 of file CDD_I2s.h.

6.1.3.11 I2S_SYNCTRANSMIT_ID

```
#define I2S_SYNCTRANSMIT_ID
```

API service ID for I2s_SyncTransmit function.

Definition at line 189 of file CDD_I2s.h.

6.1.3.12 I2S_ASYNCTRANSMIT_ID

```
#define I2S_ASYNCTRANSMIT_ID
```

API service ID for I2s_AsyncTransmit function.

Definition at line 194 of file CDD_I2s.h.

6.1.3.13 I2S_GETSTATUS_ID

```
#define I2S_GETSTATUS_ID
```

API service ID for I2s_GetStatus function.

Definition at line 199 of file CDD_I2s.h.

6.1.3.14 I2S_ABORTTRANSMIT_ID

```
#define I2S_ABORTTRANSMIT_ID
```

API service ID for I2s_AbortTransmit function.

Definition at line 204 of file CDD_I2s.h.

6.1.3.15 I2S_GETVERSIONINFO_ID

```
#define I2S_GETVERSIONINFO_ID
```

API service ID for I2s_GetVersionInfo function.

Definition at line 209 of file CDD_I2s.h.

6.1.3.16 SAI_IP_FPACK_8_BIT

```
#define SAI_IP_FPACK_8_BIT
```

8-bit FIFO packing is enabled

Definition at line 80 of file Sai_Ip_FeatureDefines.h.

6.1.3.17 SAI_IP_FPACK_16_BIT

```
#define SAI_IP_FPACK_16_BIT
```

16-bit FIFO packing is enabled

Definition at line 81 of file Sai_Ip_FeatureDefines.h.

6.1.3.18 FEATURE_SAI_IP_HAS_CHMOD

```
#define FEATURE_SAI_IP_HAS_CHMOD
```

Channel mode tristate or output zero.

Definition at line 83 of file Sai_Ip_FeatureDefines.h.

6.1.3.19 FEATURE_SAI_IP_MSEL_BUS_CLK

```
#define FEATURE_SAI_IP_MSEL_BUS_CLK
```

Master clock from Bus clock.

Definition at line 85 of file Sai_Ip_FeatureDefines.h.

6.1.3.20 FEATURE_SAI_IP_MSEL_SOSC_DIV1_CLK

```
#define FEATURE_SAI_IP_MSEL_SOSC_DIV1_CLK
```

Master clock from SOSC_DIV1 clock.

Definition at line 87 of file Sai_Ip_FeatureDefines.h.

6.1.3.21 FEATURE_SAI_IP_MSEL_MCLK_PIN

```
#define FEATURE_SAI_IP_MSEL_MCLK_PIN
```

Master clock from MCLK pin.

Definition at line 89 of file Sai_Ip_FeatureDefines.h.

6.1.3.22 FEATURE_SAI_IP_MSEL_OTHER_MCLK_PIN

```
#define FEATURE_SAI_IP_MSEL_OTHER_MCLK_PIN
```

Master clock from MCLK pin from other SAI instance.

Definition at line 91 of file Sai_Ip_FeatureDefines.h.

6.1.3.23 FEATURE_SAI_IP_HAS_PARAM

```
#define FEATURE_SAI_IP_HAS_PARAM
```

SAI has PARAM register.

Definition at line 93 of file Sai_Ip_FeatureDefines.h.

6.1.3.24 SAI0_IP_CHANNEL_COUNT

```
#define SAI0_IP_CHANNEL_COUNT
```

SAI channel count.

Definition at line 95 of file Sai_Ip_FeatureDefines.h.

6.1.3.25 SAI_IP_NULL_PTRS

```
#define SAI_IP_NULL_PTRS
```

SAI NULL PTRs.

Definition at line 101 of file Sai_Ip_FeatureDefines.h.

6.1.4 Types Reference

6.1.4.1 I2s_HwUnitCount

```
typedef uint8 I2s_HwUnitCount
```

This gives the number of I2S configuration.

Definition at line 126 of file CDD_I2s_Types.h.

6.1.4.2 I2s_DataType

```
typedef uint8 I2s_DataType
```

This gives the transfer data type.

Definition at line 131 of file CDD_I2s_Types.h.

6.1.4.3 I2s_WordMaskType

```
typedef uint16 I2s_WordMaskType
```

I2s Word Mask for transmitted data.

Definition at line 136 of file CDD_I2s_Types.h.

6.1.4.4 I2s_LogicalChannel

```
typedef uint8 I2s_LogicalChannel
```

This gives the logical channel.

Definition at line 141 of file CDD_I2s_Types.h.

6.1.5 Enum Reference

6.1.5.1 I2s_DataDirectionType

```
enum I2s_DataDirectionType
```

Definition of the type of transfer direction.

Enumerator

I2S_SEND_DATA	Used to send data.
I2S_RECEIVE_DATA	Used to receive data.

Definition at line 155 of file CDD_I2s_Types.h.

6.1.5.2 I2s_StatusType

```
enum I2s_StatusType
```

Definition for different state and errors of Operation Status.

Enumerator

I2S_STATUS_UNINITIALIZED	Function unsupported
I2S_STATUS_COMPLETED	Function completed successfully

Enumerator

I2S_STATUS_ERROR	Function didn't complete successfully
I2S_STATUS_TIMEOUT	Function timed out
I2S_STATUS_BUSY	Function busy
I2S_STATUS_ABORTED	Function aborted

Definition at line 149 of file I2s_Ipw_Types.h.

6.1.5.3 I2s_TransmitReceiveType

`enum I2s_TransmitReceiveType`

Definition of the transmitter/recevier of a SAI hw unit.

Enumerator

I2S_TRANSMITTER	Transmitter.
I2S_RECEIVER	Receiver.

Definition at line 162 of file I2s_Ipw_Types.h.

6.1.5.4 I2s_ModeType

`enum I2s_ModeType`

Definition of the master/slave mode.

Enumerator

I2S_MASTER	Master.
I2S_SLAVE	Slave.

Definition at line 171 of file I2s_Ipw_Types.h.

6.1.5.5 I2s_SupportedIPsType

`enum I2s_SupportedIPsType`

This enum contains all IPs which can integrate SAI functionalities.

Definition at line 180 of file I2s_Ipw_Types.h.

6.1.6 Function Reference

6.1.6.1 I2s_Init()

```
void I2s_Init (
            const I2s_ConfigType *const pConfigPtr )
```

Initializes the I2S module.

This function performs software initialization of I2S driver.

Parameters

in	*pConfig*	Pointer to I2S driver configuration set.

Returns

void

Note

Service ID: 0x00.

Synchronous, non re-entrant function.

6.1.6.2 I2s_DeInit()

```
void I2s_DeInit (
            void  )
```

DeInitializes the I2S module.

This function performs software de initialization of I2S modules to reset values. The service influences only the peripherals, which are allocated by static configuration and the runtime configuration set passed by the previous call of I2s_Init() The driver needs to be initialized before calling I2s_DeInit(). Otherwise, the function I2s_DeInit shall raise the development error I2S_E_UNINIT and leave the desired de initialization functionality without any action.

Parameters

in	*void*	

Returns

void

Note

Service ID: 0x01.

Synchronous, non re-entrant function.

6.1.6.3 I2s_SyncTransmit()

```
Std_ReturnType I2s_SyncTransmit (
            I2s_LogicalChannel LogicChn,
            const I2s_RequestType *const pRequest )
```

Sends or receives an I2S message blocking.

Sends the slave address and based on the direction of the message it sends or receives data by using a blocking mechanism.

Parameters

in	*LogicChn*	I2S logical channel to be addressed.
in	*pRequest*	Pointer to data information to be used

Returns

Std_ReturnType.

Return values

E_NOT_OK	If the I2S Instance is not valid or I2S driver is not initialized or pRequest is NULL or I2S Instance is in busy state.
E_OK	Otherwise.

Note

Service ID: 0x02.

Synchronous, non reentrant function.

6.1.6.4 I2s_AsyncTransmit()

```
Std_ReturnType I2s_AsyncTransmit (
            I2s_LogicalChannel LogicChn,
            const I2s_RequestType *const pRequest )
```

Starts an asynchronous transmission on the I2S bus.

Sends the slave address and enables the interrupts that will send or receive data depending on the direction of the message.

Parameters

in	*LogicChn*	I2S logical channel to be addressed.
in	*pRequest*	Pointer to data information to be used

Returns

Std_ReturnType.

Return values

E_NOT_OK	If the I2S Instance is not valid or I2S driver is not initialized or pRequest is NULL or I2S Instance is in busy state.
E_OK	Otherwise.

Note

Service ID: 0x03.

Synchronous, non reentrant function.

6.1.6.5 I2s_GetStatus()

```
Std_ReturnType I2s_GetStatus (
            I2s_LogicalChannel LogicChn,
            I2s_StatusStructType *const pStatus )
```

Gets the status of an I2S instance.

Gets the status of an I2S instance and checks for errors.

Parameters

in	*LogicChn*	I2S logical channel to be addressed.
out	*pStatus*	Pointer for storing the current transfer status

Returns

Std_ReturnType.

Return values

E_NOT_OK	If the I2S Instance is not valid or I2S driver is not initialized or pStatus is NULL.
E_OK	Otherwise.

Note

Service ID: 0x04.

Synchronous, non re-entrant function.

6.1.6.6 I2s_AbortTransmit()

```
void I2s_AbortTransmit (
            I2s_LogicalChannel LogicChn )
```

Aborts the I2s ongoing transmission.

Aborts the I2s ongoing transmission.

Parameters

in	*LogicChn*	I2S logical channel to be addressed.

Returns

void.

Note

Service ID: 0x05.

Synchronous, non reentrant function.

6.2 FLEXIO_I2S IPL

6.2.1 Detailed Description

Data Structures

- struct Flexio_I2s_Ip_CommonStateType
- struct Flexio_I2s_Ip_StateType

  Master internal context structure. More...
- struct Flexio_I2s_Ip_MasterConfigType

  Master configuration structure. More...
- struct Flexio_I2s_Ip_SlaveConfigType

  Slave configuration structure. More...

Enum Reference

- enum Flexio_I2s_Ip_StatusType

  Status type Implements : Flexio_I2s_Ip_StatusType_Class.
- enum Flexio_I2s_Ip_DriverType

  Driver type: INTERRUPT/POLLING/DMA Implements : Flexio_I2s_Ip_DriverType_Class.
- enum Flexio_I2s_Ip_EventType

Function Reference

- void Flexio_I2s_Ip_MasterInit (uint8 Instance, uint8 Channel, const Flexio_I2s_Ip_MasterConfigType *ConfigPtr)

  Initialize the FLEXIO_I2S master mode driver.
- void Flexio_I2s_Ip_MasterDeinit (uint8 Instance, uint8 Channel)

  De-initialize the FLEXIO_I2S master mode driver.
- Flexio_I2s_Ip_StatusType Flexio_I2s_Ip_MasterSendData (uint8 Instance, uint8 Channel, const uint8 *TxBuff, uint32 TxSize)

  Perform a non-blocking send transaction on the I2S bus.
- Flexio_I2s_Ip_StatusType Flexio_I2s_Ip_MasterSendDataBlocking (uint8 Instance, uint8 Channel, const uint8 *TxBuff, uint32 TxSize, uint32 Timeout)

  Perform a blocking send transaction on the I2S bus.
- Flexio_I2s_Ip_StatusType Flexio_I2s_Ip_MasterReceiveData (uint8 Instance, uint8 Channel, uint8 *Rx↩ Buff, uint32 RxSize)

  Perform a non-blocking receive transaction on the I2S bus.
- Flexio_I2s_Ip_StatusType Flexio_I2s_Ip_MasterReceiveDataBlocking (uint8 Instance, uint8 Channel, uint8 *RxBuff, uint32 RxSize, uint32 Timeout)

  Perform a blocking receive transaction on the I2S bus.
- Flexio_I2s_Ip_StatusType Flexio_I2s_Ip_MasterTransferAbort (uint8 Instance, uint8 Channel)

  Aborts a non-blocking I2S master transaction.

- Flexio_I2s_Ip_StatusType Flexio_I2s_Ip_MasterGetStatus (uint8 Instance, uint8 Channel, uint32 *Bytes↩ Remaining)

  Get the status of the current non-blocking I2S master transaction.
- void Flexio_I2s_Ip_SlaveInit (uint8 Instance, uint8 Channel, const Flexio_I2s_Ip_SlaveConfigType *ConfigPtr)

  Initialize the FLEXIO_I2S slave mode driver.
- void Flexio_I2s_Ip_SlaveDeinit (uint8 Instance, uint8 Channel)

  De-initialize the FLEXIO_I2S slave mode driver.
- Flexio_I2s_Ip_StatusType Flexio_I2s_Ip_SlaveSendData (uint8 Instance, uint8 Channel, const uint8 *Tx↩ Buff, uint32 TxSize)

  Perform a non-blocking send transaction on the I2S bus.
- Flexio_I2s_Ip_StatusType Flexio_I2s_Ip_SlaveSendDataBlocking (uint8 Instance, uint8 Channel, const uint8 *TxBuff, uint32 TxSize, uint32 Timeout)

  Perform a blocking send transaction on the I2S bus.
- Flexio_I2s_Ip_StatusType Flexio_I2s_Ip_SlaveReceiveData (uint8 Instance, uint8 Channel, uint8 *RxBuff, uint32 RxSize)

  Perform a non-blocking receive transaction on the I2S bus.
- Flexio_I2s_Ip_StatusType Flexio_I2s_Ip_SlaveReceiveDataBlocking (uint8 Instance, uint8 Channel, uint8 *RxBuff, uint32 RxSize, uint32 Timeout)

  Perform a blocking receive transaction on the I2S bus.
- Flexio_I2s_Ip_StatusType Flexio_I2s_Ip_SlaveTransferAbort (uint8 Instance, uint8 Channel)

  Aborts a non-blocking I2S slave transaction.
- Flexio_I2s_Ip_StatusType Flexio_I2s_Ip_SlaveGetStatus (uint8 Instance, uint8 Channel, uint32 *Bytes↩ Remaining)

  Get the status of the current non-blocking I2S slave transaction.
- void Flexio_I2s_Ip_IrqHandler (uint8 FlexIOChannel, uint8 ShifterMaskFlag, uint8 ShifterErrMaskFlag, uint8 TimerMaskFlag)

  Interrupt handler for FlexIO.

6.2.2 Data Structure Documentation

6.2.2.1 struct Flexio_I2s_Ip_CommonStateType

FlexIO common context structure This is a common structure used by all FlexIO drivers as a part of their context structure. It is needed for common operations such as interrupt handling.

Definition at line 136 of file Flexio_I2s_Ip_Types.h.

Data Fields

Type	Name	Description
uint8	Instance	FlexIO device instance number
uint8	ResourceCount	Count of internal resources used (shifters and timers)
uint8	ResourceIndex	Index of first used internal resource instance (shifter and timer)

6.2.2.2 struct Flexio_I2s_Ip_StateType

Master internal context structure.

This structure is used by the driver for its internal logic. It must be provided by the application through the FLEXIO_I2S_DRV_MasterInit() function, then it cannot be freed until the driver is de-initialized using FLEXIO_I2S_DRV_MasterDeinit(). The application should make no assumptions about the content of this structure.

Definition at line 151 of file Flexio_I2s_Ip_Types.h.

6.2.2.3 struct Flexio_I2s_Ip_MasterConfigType

Master configuration structure.

This structure is used to provide configuration parameters for the flexio_i2s master at initialization time. Implements : Flexio_I2s_Ip_MasterConfigType_Class

Definition at line 181 of file Flexio_I2s_Ip_Types.h.

Data Fields

Type	Name	Description
Flexio_I2s_Ip_DriverType	driverType	Driver type: interrupts/polling/DMA
uint8	ResourceIndex	Index of first used internal resource instance (shifter and timer)
uint16	DividerValue	Calculated divider value
uint8	byteWidth	
uint8	txPin	Flexio pin to use for transmit
uint8	rxPin	Flexio pin to use for receive
uint8	sckPin	Flexio pin to use for serial clock
uint8	wsPin	Flexio pin to use for word select
Flexio_I2s_CallbackType	callback	User callback function. Note that this function will be called from the interrupt service routine, so its execution time should be as small as possible. It can be NULL if it is not needed
void *	callbackParam	Parameter for the callback function
Flexio_I2s_Ip_StateType *	MasterState	Pointer to master state

6.2.2.4 struct Flexio_I2s_Ip_SlaveConfigType

Slave configuration structure.

This structure is used to provide configuration parameters for the flexio_i2s slave at initialization time. Implements : Flexio_I2s_Ip_SlaveConfigType_Class

Definition at line 210 of file Flexio_I2s_Ip_Types.h.

Data Fields

Type	Name	Description
Flexio_I2s_Ip_DriverType	driverType	Driver type: interrupts/polling/DMA
uint8	ResourceIndex	Index of first used internal resource instance (shifter and timer)
uint8	byteWidth	
uint8	txPin	Flexio pin to use for transmit
uint8	rxPin	Flexio pin to use for receive
uint8	sckPin	Flexio pin to use for serial clock
uint8	wsPin	Flexio pin to use for word select
Flexio_I2s_CallbackType	callback	User callback function. Note that this function will be called from the interrupt service routine, so its execution time should be as small as possible. It can be NULL if it is not needed
void *	callbackParam	Parameter for the callback function
Flexio_I2s_Ip_StateType *	SlaveState	Pointer to master state

6.2.3 Enum Reference

6.2.3.1 Flexio_I2s_Ip_StatusType

enum Flexio_I2s_Ip_StatusType

Status type Implements : Flexio_I2s_Ip_StatusType_Class.

Enumerator

FLEXIO_I2S_IP_STATUS_TX_UNDERRUN	TX underrun error
FLEXIO_I2S_IP_STATUS_RX_OVERRUN	RX overrun error
FLEXIO_I2S_IP_STATUS_ABORTED	A transfer was aborted

Definition at line 81 of file Flexio_I2s_Ip_Types.h.

6.2.3.2 Flexio_I2s_Ip_DriverType

enum Flexio_I2s_Ip_DriverType

Driver type: INTERRUPT/POLLING/DMA Implements : Flexio_I2s_Ip_DriverType_Class.

Enumerator

FLEXIO_I2S_IP_DRIVER_TYPE_INTERRUPT	Driver uses INTERRUPT for data transfers
FLEXIO_I2S_IP_DRIVER_TYPE_POLLING	Driver is based on POLLING

Definition at line 97 of file Flexio_I2s_Ip_Types.h.

6.2.3.3 Flexio_I2s_Ip_EventType

```
enum Flexio_I2s_Ip_EventType
```

Enumerator

FLEXIO_I2S_IP_EVENT_RX_FULL	Rx buffer is full
FLEXIO_I2S_IP_EVENT_TX_EMPTY	Tx buffer is empty
FLEXIO_I2S_IP_EVENT_END_TRANSFER	The current transfer is ending. Only FLEXIO instance uses this event. The difference between this and event TX_EMPTY is: TX_EMPTY is generated when all data has been pushed to hardware fifo, users should not call DeInit here or some last data will be lost; END_TRANSFER is generated when all data has been pushed to line, the transmission will be stopped before users can start transmit again, user can call DeInit here. For receiving case, this event is the same as RX_FULL.
FLEXIO_I2S_IP_EVENT_ERROR	An error occurred during transfer

Definition at line 110 of file Flexio_I2s_Ip_Types.h.

6.2.4 Function Reference

6.2.4.1 Flexio_I2s_Ip_MasterInit()

```
void Flexio_I2s_Ip_MasterInit (
            uint8 Instance,
            uint8 Channel,
            const Flexio_I2s_Ip_MasterConfigType * ConfigPtr )
```

Initialize the FLEXIO_I2S master mode driver.

This function initializes the FLEXIO_I2S driver in master mode.

Parameters

in	*Instance*	FLEXIO peripheral instance number
in	*Channel*	FLEXIO_I2S logical channel number
in	*ConfigPtr*	Pointer to the FLEXIO_I2S master user configuration structure. The function reads configuration data from this structure and initializes the driver accordingly. The application may free this structure after the function returns.

Returns

void.

6.2.4.2 Flexio_I2s_Ip_MasterDeinit()

```
void Flexio_I2s_Ip_MasterDeinit (
            uint8 Instance,
            uint8 Channel )
```

De-initialize the FLEXIO_I2S master mode driver.

This function de-initializes the FLEXIO_I2S driver in master mode. The driver can't be used again until reinitialized.

Parameters

in	*Instance*	FLEXIO peripheral instance number
in	*Channel*	FLEXIO_I2S logical channel number

Returns

void.

6.2.4.3 Flexio_I2s_Ip_MasterSendData()

```
Flexio_I2s_Ip_StatusType Flexio_I2s_Ip_MasterSendData (
            uint8 Instance,
            uint8 Channel,
            const uint8 * TxBuff,
            uint32 TxSize )
```

Perform a non-blocking send transaction on the I2S bus.

This function starts the transmission of a block of data and returns immediately. The rest of the transmission is handled by the interrupt service routine (if the driver is initialized in interrupt mode) or by the Flexio_I2s_Ip↩ _MasterGetStatus function (if the driver is initialized in polling mode). Use Flexio_I2s_Ip_MasterGetStatus() to check the progress of the transmission.

Parameters

in	*Instance*	FLEXIO peripheral instance number
in	*Channel*	FLEXIO_I2S logical channel number
in	*TxBuff*	Pointer to the data to be transferred
in	*TxSize*	Length in words of the data to be transferred. Word length is configured via byteWidth parameter in the init config structure.

Returns

Error or success status returned by API

6.2.4.4 Flexio_I2s_Ip_MasterSendDataBlocking()

```
Flexio_I2s_Ip_StatusType Flexio_I2s_Ip_MasterSendDataBlocking (
            uint8 Instance,
            uint8 Channel,
            const uint8 * TxBuff,
            uint32 TxSize,
            uint32 Timeout )
```

Perform a blocking send transaction on the I2S bus.

This function sends a block of data, and only returns when the transmission is complete.

Parameters

in	*Instance*	FLEXIO peripheral instance number
in	*Channel*	FLEXIO_I2S logical channel number
in	*TxBuff*	Pointer to the data to be transferred
in	*TxSize*	Length in words of the data to be transferred. Word length is configured via byteWidth parameter in the init config structure.
in	*Timeout*	Timeout for the transfer in milliseconds

Returns

Error or success status returned by API

6.2.4.5 Flexio_I2s_Ip_MasterReceiveData()

```
Flexio_I2s_Ip_StatusType Flexio_I2s_Ip_MasterReceiveData (
            uint8 Instance,
            uint8 Channel,
            uint8 * RxBuff,
            uint32 RxSize )
```

Perform a non-blocking receive transaction on the I2S bus.

This function starts the reception of a block of data and returns immediately. The rest of the reception is handled by the interrupt service routine (if the driver is initialized in interrupt mode) or by the Flexio_I2s_Ip_MasterGetStatus function (if the driver is initialized in polling mode). Use Flexio_I2s_Ip_MasterGetStatus() to check the progress of the reception.

Parameters

in	*Instance*	FLEXIO peripheral instance number
in	*Channel*	FLEXIO_I2S logical channel number
out	*RxBuff*	Pointer to the buffer where to store received data
in	*RxSize*	Length in words of the data to be transferred. Word length is configured via byteWidth parameter in the init config structure.

Returns

Error or success status returned by API

6.2.4.6 Flexio_I2s_Ip_MasterReceiveDataBlocking()

```
Flexio_I2s_Ip_StatusType Flexio_I2s_Ip_MasterReceiveDataBlocking (
            uint8 Instance,
            uint8 Channel,
            uint8 * RxBuff,
            uint32 RxSize,
            uint32 Timeout )
```

Perform a blocking receive transaction on the I2S bus.

This function receives a block of data and only returns when the reception is complete.

Parameters

in	*Instance*	FLEXIO peripheral instance number
in	*Channel*	FLEXIO_I2S logical channel number
out	*RxBuff*	Pointer to the buffer where to store received data
in	*RxSize*	Length in words of the data to be transferred. Word length is configured via byteWidth parameter in the init config structure.
in	*Timeout*	Timeout for the transfer in milliseconds

Returns

Error or success status returned by API

6.2.4.7 Flexio_I2s_Ip_MasterTransferAbort()

```
Flexio_I2s_Ip_StatusType Flexio_I2s_Ip_MasterTransferAbort (
            uint8 Instance,
            uint8 Channel )
```

Aborts a non-blocking I2S master transaction.

This function aborts a non-blocking I2S transfer.

Parameters

in	*Instance*	FLEXIO peripheral instance number
in	*Channel*	FLEXIO_I2S logical channel number

Returns

Error or success status returned by API

6.2.4.8 Flexio_I2s_Ip_MasterGetStatus()

```
Flexio_I2s_Ip_StatusType Flexio_I2s_Ip_MasterGetStatus (
            uint8 Instance,
            uint8 Channel,
            uint32 * BytesRemaining )
```

Get the status of the current non-blocking I2S master transaction.

This function returns the current status of a non-blocking I2S master transaction. A return code of FLEXIO_I2S_IP_STATUS_BUSY means the transfer is still in progress. Otherwise the function returns a status reflecting the outcome of the last transfer. When the driver is initialized in polling mode this function also advances the transfer by checking and handling the transmit and receive events, so it must be called frequently to avoid overflows or underflows.

Parameters

in	*Instance*	FLEXIO peripheral instance number
in	*Channel*	FLEXIO_I2S logical channel number
out	*BytesRemaining*	Remaining number of bytes to be transferred

Returns

Error or success status returned by API

6.2.4.9 Flexio_I2s_Ip_SlaveInit()

```
void Flexio_I2s_Ip_SlaveInit (
            uint8 Instance,
```

```
            uint8 Channel,
            const Flexio_I2s_Ip_SlaveConfigType * ConfigPtr )
```

Initialize the FLEXIO_I2S slave mode driver.

This function initializes the FLEXIO_I2S driver in slave mode.

Parameters

in	*Instance*	FLEXIO peripheral instance number
in	*Channel*	FLEXIO_I2S logical channel number
in	*ConfigPtr*	Pointer to the FLEXIO_I2S slave user configuration structure. The function reads configuration data from this structure and initializes the driver accordingly. The application may free this structure after the function returns.

Returns

void.

6.2.4.10 Flexio_I2s_Ip_SlaveDeinit()

```
void Flexio_I2s_Ip_SlaveDeinit (
            uint8 Instance,
            uint8 Channel )
```

De-initialize the FLEXIO_I2S slave mode driver.

This function de-initializes the FLEXIO_I2S driver in slave mode. The driver can't be used again until reinitialized.

Parameters

in	*Instance*	FLEXIO peripheral instance number
in	*Channel*	FLEXIO_I2S logical channel number

Returns

void.

6.2.4.11 Flexio_I2s_Ip_SlaveSendData()

```
Flexio_I2s_Ip_StatusType Flexio_I2s_Ip_SlaveSendData (
            uint8 Instance,
```

```
            uint8 Channel,
            const uint8 * TxBuff,
            uint32 TxSize )
```

Perform a non-blocking send transaction on the I2S bus.

This function starts the transmission of a block of data and returns immediately. The rest of the transmission is handled by the interrupt service routine (if the driver is initialized in interrupt mode) or by the Flexio_I2s_Ip_↩ SlaveGetStatus function (if the driver is initialized in polling mode). Use Flexio_I2s_Ip_SlaveGetStatus() to check the progress of the transmission.

Parameters

in	*Instance*	FLEXIO peripheral instance number
in	*Channel*	FLEXIO_I2S logical channel number
in	*TxBuff*	Pointer to the data to be transferred
in	*TxSize*	Length in words of the data to be transferred. Word length is configured via byteWidth parameter in the init config structure.

Returns

Error or success status returned by API

6.2.4.12 Flexio_I2s_Ip_SlaveSendDataBlocking()

```
Flexio_I2s_Ip_StatusType Flexio_I2s_Ip_SlaveSendDataBlocking (
            uint8 Instance,
            uint8 Channel,
            const uint8 * TxBuff,
            uint32 TxSize,
            uint32 Timeout )
```

Perform a blocking send transaction on the I2S bus.

This function sends a block of data, and only returns when the transmission is complete.

Parameters

in	*Instance*	FLEXIO peripheral instance number
in	*Channel*	FLEXIO_I2S logical channel number
in	*TxBuff*	Pointer to the data to be transferred
in	*TxSize*	Length in words of the data to be transferred. Word length is configured via byteWidth parameter in the init config structure.
in	*Timeout*	Timeout for the transfer in milliseconds

Returns

Error or success status returned by API

6.2.4.13 Flexio_I2s_Ip_SlaveReceiveData()

```
Flexio_I2s_Ip_StatusType Flexio_I2s_Ip_SlaveReceiveData (
            uint8 Instance,
            uint8 Channel,
            uint8 * RxBuff,
            uint32 RxSize )
```

Perform a non-blocking receive transaction on the I2S bus.

This function starts the reception of a block of data and returns immediately. The rest of the reception is handled by the interrupt service routine (if the driver is initialized in interrupt mode) or by the Flexio_I2s_Ip_SlaveGetStatus function (if the driver is initialized in polling mode). Use Flexio_I2s_Ip_SlaveGetStatus() to check the progress of the reception.

Parameters

in	*Instance*	FLEXIO peripheral instance number
in	*Channel*	FLEXIO_I2S logical channel number
out	*RxBuff*	Pointer to the buffer where to store received data
in	*RxSize*	Length in words of the data to be transferred. Word length is configured via byteWidth parameter in the init config structure.

Returns

Error or success status returned by API

6.2.4.14 Flexio_I2s_Ip_SlaveReceiveDataBlocking()

```
Flexio_I2s_Ip_StatusType Flexio_I2s_Ip_SlaveReceiveDataBlocking (
            uint8 Instance,
            uint8 Channel,
            uint8 * RxBuff,
            uint32 RxSize,
            uint32 Timeout )
```

Perform a blocking receive transaction on the I2S bus.

This function receives a block of data and only returns when the reception is complete.

Parameters

in	*Instance*	FLEXIO peripheral instance number
in	*Channel*	FLEXIO_I2S logical channel number
out	*RxBuff*	Pointer to the buffer where to store received data
in	*RxSize*	Length in words of the data to be transferred. Word length is configured via byteWidth parameter in the init config structure.
in	*Timeout*	Timeout for the transfer in milliseconds

Returns

Error or success status returned by API

6.2.4.15 Flexio_I2s_Ip_SlaveTransferAbort()

```
Flexio_I2s_Ip_StatusType Flexio_I2s_Ip_SlaveTransferAbort (
            uint8 Instance,
            uint8 Channel )
```

Aborts a non-blocking I2S slave transaction.

This function aborts a non-blocking I2S transfer.

Parameters

in	*Instance*	FLEXIO peripheral instance number
in	*Channel*	FLEXIO_I2S logical channel number

Returns

Error or success status returned by API

6.2.4.16 Flexio_I2s_Ip_SlaveGetStatus()

```
Flexio_I2s_Ip_StatusType Flexio_I2s_Ip_SlaveGetStatus (
            uint8 Instance,
            uint8 Channel,
            uint32 * BytesRemaining )
```

Get the status of the current non-blocking I2S slave transaction.

This function returns the current status of a non-blocking I2S slave transaction. A return code of FLEXIO_I2↩ S_IP_STATUS_BUSY means the transfer is still in progress. Otherwise the function returns a status reflecting the outcome of the last transfer. When the driver is initialized in polling mode this function also advances the transfer by checking and handling the transmit and receive events, so it must be called frequently to avoid overflows or underflows.

Parameters

in	*Instance*	FLEXIO peripheral instance number
in	*Channel*	FLEXIO_I2S logical channel number
out	*BytesRemaining*	Remaining number of bytes to be transferred

Returns

Error or success status returned by API

6.2.4.17 Flexio_I2s_Ip_IrqHandler()

```
void Flexio_I2s_Ip_IrqHandler (
             uint8 FlexIOChannel,
             uint8 ShifterMaskFlag,
             uint8 ShifterErrMaskFlag,
             uint8 TimerMaskFlag )
```

Interrupt handler for FlexIO.

This function shall manage all the interrupts of a FlexIO module

Parameters

in	*FlexIOChannel*	FlexIO channel to be addressed.
in	*ShifterMaskFlag*	shifters status
in	*ShifterErrMaskFlag*	shifters error status
in	*TimerMaskFlag*	FlexIO timers status

Returns

void.

Note

Internal driver function.

6.3 SAI IPL

6.3.1 Detailed Description

Data Structures

- struct Sai_Ip_XferState

  Transmit or receive state. More...
- struct Sai_Ip_StateStructType

  Structure for internal use. This structure is used by the driver for its internal logic. It must be provided by the application through the initialize functions, then it cannot be freed until the driver is de-initialized using Deinit functions. The application should make no assumptions about the content of this structure. More...
- struct Sai_Ip_ConfigType

  User config structure.
  ***Note**: entries in this structure are affected by FEATURE_SAI_IP_HAS_CHMOD, which is device dependent and controlled from feature header file of the used device. More...*

Types Reference

- typedef void(* Sai_Ip_WriteHandlerType) (uint32 *const u32Data, const uint8 u8Instance, const uint8 u8↩ Pack, const uint8 u8ChnIdx)

  Sai write data handler.
- typedef void(* Sai_Ip_ReadHandlerType) (uint32 *const u32Data, const uint8 u8Instance, const uint8 u8↩ ChnIdx)

  Sai read data handler.
- typedef void(* Sai_Ip_TransferCallbackType) (Sai_Ip_ReportType event)

  Sai callback function type for nonblock transfer, also called to report events (Sai_Ip_ReportType).

Enum Reference

- enum Sai_Ip_ReportType

  Define the enum of the events which can trigger SAI callback.
- enum Sai_Ip_StatusType

  SAI status return type.
- enum Sai_Ip_TransferType

  Transfer type.
- enum Sai_Ip_MuxType

  Data mux line or mux memory.
- enum Sai_Ip_SyncType

  SAI run in sync or async mode.
  ***Note**: entries in this enum are affected by ::FEATURE_SAI_IP_SYNC_WITH_OTHER_INST, which is device dependent and controlled from feature header file of the used device.*
- enum Sai_Ip_MasterClkSelType

  Select master clock.
- enum Sai_Ip_MaskType

  Data line state for masked word, or if data line is disabled.

Function Reference

- void Sai_Ip_SetMaster (const uint8 u8Instance, const boolean bTx)

  Set active for either transmitter or receiver.
- void Sai_Ip_TxInit (const uint8 u8Instance, const Sai_Ip_ConfigType *pConfig)

  Initialize the transmitter of driver.
- void Sai_Ip_RxInit (const uint8 u8Instance, const Sai_Ip_ConfigType *pConfig)

  Initialize the receiver of driver.
- void Sai_Ip_TxDeInit (const uint8 u8Instance)

  De-initialize transmitter.
- void Sai_Ip_RxDeInit (const uint8 u8Instance)

  De-initialize receiver.
- Sai_Ip_StatusType Sai_Ip_SendBlocking (const uint8 u8Instance, const uint8 *const aData[], const uint32 u32Count, const uint32 u32Timeout)

  Send a block of data, return when transfer complete.
- void Sai_Ip_Send (const uint8 u8Instance, const uint8 *const aData[], const uint32 u32Count)

  Send a block of data, return immidiately.
- Sai_Ip_StatusType Sai_Ip_GetSendingStatus (const uint8 u8Instance, uint32 *pCountRemain)

  Get status of a non-blocking transfer.
- Sai_Ip_StatusType Sai_Ip_ReceiveBlocking (const uint8 u8Instance, uint8 *const aData[], const uint32 u32Count, const uint32 u32Timeout)

  Receive a block of data, return when transfer complete.
- void Sai_Ip_Receive (const uint8 u8Instance, uint8 *const aData[], const uint32 u32Count)

  Receive a block of data, return immidiately.
- Sai_Ip_StatusType Sai_Ip_GetReceivingStatus (const uint8 u8Instance, uint32 *pCountRemain)

  Get status of a non-blocking transfer.
- void Sai_Ip_AbortTransfer (const uint8 u8Instance)

  Abort all ongoing transferring data (both sending and receiving)

6.3.2 Data Structure Documentation

6.3.2.1 struct Sai_Ip_XferState

Transmit or receive state.

Definition at line 227 of file Sai_Ip_Types.h.

6.3.2.2 struct Sai_Ip_StateStructType

Structure for internal use. This structure is used by the driver for its internal logic. It must be provided by the application through the initialize functions, then it cannot be freed until the driver is de-initialized using Deinit functions. The application should make no assumptions about the content of this structure.

Definition at line 242 of file Sai_Ip_Types.h.

6.3.2.3 struct Sai_Ip_ConfigType

User config structure.
Note: entries in this structure are affected by FEATURE_SAI_IP_HAS_CHMOD, which is device dependent and controlled from feature header file of the used device.

Implements Sai_Ip_ConfigType_struct

Definition at line 282 of file Sai_Ip_Types.h.

Data Fields

Type	Name	Description
Sai_Ip_SyncType	eSyncMode	Sync mode.
Sai_Ip_MasterClkSelType	eMasterClkSrc	Select master clock source.
boolean	bBitClkNegPolar	True if bit clock is negative polar (active low with drive outputs on falling edge and sample inputs on rising edge), false otherwise.
boolean	bBitClkInternal	True if bit clock is generated internally.
uint32	u32MasterClkFreq	Clock frequency from selected master clock source, only for internally generated master clock
uint32	u32BitClkFreq	Desired bit clock frequency in hertz, only for internally generated master clock and bit clock. Real divisor is derived by: floor(floor(MasterClockFreq / BitClkFreq) / 2)) * 2, value range is from 2 to 512. If out of range value is assigned, maximum or minimum value is used. Real closest BitClockFreq is derived by: MasterClockFreq / real_divisor
uint8	u8ChannelEnable	Turn on each bit to enable each channel. 4 bit for 4 channels.
uint8	u8FrameSize	Frame size in number of words. When in mux line mode, the number of unmasked word in a frame must be divisible by number of channels
uint8	u8SyncWidth	Active sync width in number of bit clocks (must not longer than first word width).
Sai_Ip_MaskType	eMaskMode	Data line state for mask word or when data line is disabled (apply only for transmitter).
boolean	bDebugEnable	True if transfer is enabled in debug mode after completing the current frame, false if transfer is disabled in debug mode.
boolean	bMsbFirst	True if data is MSB first, false if LSB first.
boolean	bSyncEarly	True if frame sync is one bit clock early.
boolean	bSyncNegPolar	True if frame sync is negative polar (active low), false otherwise.
boolean	bSyncInternal	True if frame sync is generated internally
uint8	u8Word0Width	First word width in number of bit clocks.
uint8	u8WordNWidth	Other words width in number of bit clocks.
uint8	u8FirstBitIndex	Index from LSB of first bit to be transmitted/received, valid range from 0-31.

Data Fields

Type	Name	Description
boolean	bRunErrorReport	Underrun/overrun error report.
boolean	bSyncErrorReport	Enable sync error report.
boolean	bFrameStartReport	Enable frame start report.
Sai_Ip_MuxType	eMuxMode	Enable line mux, memory mux or mux is disabled.
Sai_Ip_TransferType	eTransferType	Transfer using dma or interrupt.
uint8	u8ElementSize	Size in bytes of each read/write from/to buffer.
uint8	u8ChannelCount	Number of channels to enable, only used when both line mux mode and interrupt mode is selected.
uint8	u8BufferAccessSize	Number of words each time read from/write to buffer
uint8	u8WordFlagIndex	Configure which word sets the start of word flag, should less than frame size
boolean	bBitClkInput	Use slave mode timing in datasheet for master mode
boolean	bContOnErr	True continue transfering the same work that caused FIFO eror, False start the next frame
boolean	bSyncOnDemand	True frame sync is generated only when fifo is not empty (transmit) or not full (receive), False otherwise
Sai_Ip_TransferCallbackType	pCallback	User callback function, called when transfer complete or selected events occurred.
Sai_Ip_StateStructType *	pState	Sai IP internal state.

6.3.3 Types Reference

6.3.3.1 Sai_Ip_WriteHandlerType

```
typedef void(* Sai_Ip_WriteHandlerType) (uint32 *const u32Data, const uint8 u8Instance, const uint8 u8↩
Pack, const uint8 u8ChnIdx)
```

Sai write data handler.

Definition at line 211 of file Sai_Ip_Types.h.

6.3.3.2 Sai_Ip_ReadHandlerType

```
typedef void(* Sai_Ip_ReadHandlerType) (uint32 *const u32Data, const uint8 u8Instance, const uint8 u8↩
ChnIdx)
```

Sai read data handler.

Definition at line 215 of file Sai_Ip_Types.h.

6.3.3.3 Sai_Ip_TransferCallbackType

```
typedef void(* Sai_Ip_TransferCallbackType) (Sai_Ip_ReportType event)
```

Sai callback function type for nonblock transfer, also called to report events (Sai_Ip_ReportType).

Definition at line 219 of file Sai_Ip_Types.h.

6.3.4 Enum Reference

6.3.4.1 Sai_Ip_ReportType

```
enum Sai_Ip_ReportType
```

Define the enum of the events which can trigger SAI callback.

This enum should include the events for all platforms

Enumerator

SAI_IP_RX_COMPLETE	Rx transfer complete, user can start another transfer
SAI_IP_TX_COMPLETE	Tx transfer complete, user can start another transfer
SAI_IP_ERROR	DMA error while transfer
SAI_IP_FRAME_START	Indicate a frame start
SAI_IP_RUN_ERROR	Overrun/underrun error. This event is useful for tx when user want to know whether all data in tx fifo has been pushed out to line and tx deinit can be called, since tx complete event only indicates all data from buffer has been pushed to hardware fifo
SAI_IP_SYNC_ERROR	Frame sync error

Definition at line 111 of file Sai_Ip_Types.h.

6.3.4.2 Sai_Ip_StatusType

```
enum Sai_Ip_StatusType
```

SAI status return type.

This structure is used as return type

Enumerator

SAI_IP_STATUS_UNINITIALIZED	Function unsupported
SAI_IP_STATUS_COMPLETED	Function completed successfully

Enumerator

SAI_IP_STATUS_ERROR	Function didn't complete successfully
SAI_IP_STATUS_TIMEOUT	Function timed out
SAI_IP_STATUS_BUSY	Function busy
SAI_IP_STATUS_ABORTED	Function aborted

Definition at line 128 of file Sai_Ip_Types.h.

6.3.4.3 Sai_Ip_TransferType

enum Sai_Ip_TransferType

Transfer type.

Enumerator

SAI_IP_INTERRUPT	Transfer type is interrupt

Definition at line 140 of file Sai_Ip_Types.h.

6.3.4.4 Sai_Ip_MuxType

enum Sai_Ip_MuxType

Data mux line or mux memory.

Enumerator

SAI_IP_MUX_DISABLED	Each data line is a channel, uses a seperate memory block
SAI_IP_MUX_LINE	Only one data line (data line 0 for DMA mode) is used. Words on data line is alternated between channels, each channel data is a seperate memory block
SAI_IP_MUX_MEM	Words in memory block is alternated between channels, each channel data is on a seperate data line.

Definition at line 150 of file Sai_Ip_Types.h.

6.3.4.5 Sai_Ip_SyncType

enum Sai_Ip_SyncType

SAI run in sync or async mode.
Note: entries in this enum are affected by ::FEATURE_SAI_IP_SYNC_WITH_OTHER_INST, which is device dependent and controlled from feature header file of the used device.

Enumerator

SAI_IP_ASYNC	Independent clock
SAI_IP_SYNC_WITH_OTHER	Bit clock and frame sync signal is taken from transmitter/receiver

Definition at line 161 of file Sai_Ip_Types.h.

6.3.4.6 Sai_Ip_MasterClkSelType

enum Sai_Ip_MasterClkSelType

Select master clock.

Enumerator

SAI_IP_BUS_CLK	Master clock is module Bus clock
SAI_IP_EXTERNAL_CLK	Master clock is from external
SAI_IP_SOSC_DIV1_CLK	Master clock is from external oscillator/crystal
SAI_IP_EXTERNAL_OTHER_CLK	Master clock is from external other sai instance

Definition at line 177 of file Sai_Ip_Types.h.

6.3.4.7 Sai_Ip_MaskType

enum Sai_Ip_MaskType

Data line state for masked word, or if data line is disabled.

Enumerator

SAI_IP_MASK_TRISTATE	Line is in high z state
SAI_IP_MASK_ZERO	Line is output zero

Definition at line 202 of file Sai_Ip_Types.h.

6.3.5 Function Reference

6.3.5.1 Sai_Ip_SetMaster()

```
void Sai_Ip_SetMaster (
            const uint8 u8Instance,
            const boolean bTx )
```

Set active for either transmitter or receiver.

When both of transmitter and receiver are configured as master and async mode, at the specific time only one master can be active. This function must be called to alternate between sending and receiving operation:

- Before calling the first tx/rx transfer
- After a tx/rx transfer end and before start another rx/tx transfer.

Parameters

in	*u8Instance*	Peripheral instance number
in	*bTx*	True if next operation is tx, false otherwise

6.3.5.2 Sai_Ip_TxInit()

```
void Sai_Ip_TxInit (
            const uint8 u8Instance,
            const Sai_Ip_ConfigType * pConfig )
```

Initialize the transmitter of driver.

Parameters

in	*u8Instance*	Peripheral instance number
in	*pConfig*	Pointer to the user configuration structure. The function reads configuration data from this structure and initializes the driver accordingly. The application may free this structure after the function returns.

6.3.5.3 Sai_Ip_RxInit()

```
void Sai_Ip_RxInit (
            const uint8 u8Instance,
            const Sai_Ip_ConfigType * pConfig )
```

Initialize the receiver of driver.

Parameters

in	*u8Instance*	Peripheral instance number
in	*pConfig*	Pointer to the user configuration structure. The function reads configuration data from this structure and initializes the driver accordingly. The application may free this structure after the function returns.

6.3.5.4 Sai_Ip_TxDeInit()

```
void Sai_Ip_TxDeInit (
            const uint8 u8Instance )
```

De-initialize transmitter.

This function de-initializes driver. The driver can't be used again until reinitialized. The context structure is no longer needed by the driver and can be freed after calling this function.

Parameters

in	*u8Instance*	Peripheral instance number

6.3.5.5 Sai_Ip_RxDeInit()

```
void Sai_Ip_RxDeInit (
            const uint8 u8Instance )
```

De-initialize receiver.

This function de-initializes driver. The driver can't be used again until reinitialized. The context structure is no longer needed by the driver and can be freed after calling this function.

Parameters

in	*u8Instance*	Peripheral instance number

6.3.5.6 Sai_Ip_SendBlocking()

```
Sai_Ip_StatusType Sai_Ip_SendBlocking (
            const uint8 u8Instance,
            const uint8 *const aData[],
            const uint32 u32Count,
            const uint32 u32Timeout )
```

Send a block of data, return when transfer complete.

Should be called immediately after a transfer complete to avoid data underrun error.

Parameters

in	*u8Instance*	Peripheral instance number
in	*aData*	Array of pointer to each data block to transfer, each data block corresponds to an enabled channels If mux memory is selected, only first data block is used
in	*u32Count*	Number of words to transfer for each channel. The size of each buffer word element is "Sai Buffer Element Size" selected in configurator. In DMA mode, count number upper limit is limited by DMA BITER/CITER register length.
in	*u32Timeout*	Timeout to return when transfer take too long.

Returns

Success, error or timeout status.

6.3.5.7 Sai_Ip_Send()

```
void Sai_Ip_Send (
            const uint8 u8Instance,
            const uint8 *const aData[],
            const uint32 u32Count )
```

Send a block of data, return immidiately.

When transfer completed, the callback function will be executed. User should use this callback function to immidiately start an other transfer to avoid data underrun error.

Parameters

in	*u8Instance*	Peripheral instance number
in	*aData*	Array of pointer to each data block to transfer, each data block corresponds to an enabled channels If mux memory is selected, only first data block is used
in	*u32Count*	Number of words to transfer for each channel. The size of each buffer word element is "Sai Buffer Element Size" selected in configurator. In DMA mode, count number upper limit is limited by DMA BITER/CITER register length.

6.3.5.8 Sai_Ip_GetSendingStatus()

```
Sai_Ip_StatusType Sai_Ip_GetSendingStatus (
            const uint8 u8Instance,
            uint32 * pCountRemain )
```

Get status of a non-blocking transfer.

Parameters

in	*u8Instance*	Peripheral instance number
out	*pCountRemain*	Number of words remain for each channel. This parameter can be NULL

Returns

Status of the transfer, can be success, aborted, timeout or busy. Note that for tx, success status imply that all data has been pushed to hardware fifo and another transfer can be started.

6.3.5.9 Sai_Ip_ReceiveBlocking()

```
Sai_Ip_StatusType Sai_Ip_ReceiveBlocking (
            const uint8 u8Instance,
            uint8 *const aData[],
            const uint32 u32Count,
            const uint32 u32Timeout )
```

Receive a block of data, return when transfer complete.

Should be called immidiately after a transfer complete to avoid data overrun error.

Parameters

in	*u8Instance*	Peripheral instance number
out	*aData*	Array of pointer to each data block to transfer, each data block corresponds to an enabled channels If mux memory is selected, only first data block is used
in	*u32Count*	Number of words to transfer for each channel. The size of each buffer word element is "Sai Buffer Element Size" selected in configurator. In DMA mode, count number upper limit is limited by DMA BITER/CITER register length.
in	*u32Timeout*	Timeout to return when transfer take too long.

Returns

Success, error or timeout status.

6.3.5.10 Sai_Ip_Receive()

```
void Sai_Ip_Receive (
            const uint8 u8Instance,
            uint8 *const aData[],
            const uint32 u32Count )
```

Receive a block of data, return immidiately.

When transfer completed, the callback function will be executed. User should use this callback function to immidiately start another transfer to avoid data overrun error.

Parameters

in	*u8Instance*	Peripheral instance number
out	*aData*	Array of pointer to each data block to transfer, each data block corresponds to an enabled channels If mux memory is selected, only first data block is used
in	*u32Count*	Number of words to transfer for each channel. The size of each buffer word element is "Sai Buffer Element Size" selected in configurator. In DMA mode, count number upper limit is limited by DMA BITER/CITER register length.

6.3.5.11 Sai_Ip_GetReceivingStatus()

```
Sai_Ip_StatusType Sai_Ip_GetReceivingStatus (
            const uint8 u8Instance,
            uint32 * pCountRemain )
```

Get status of a non-blocking transfer.

Parameters

in	*u8Instance*	Peripheral instance number
out	*pCountRemain*	Number of words remain for each channel. This parameter can be NULL

Returns

Status of the transfer, can be success, aborted, timeout or busy.

6.3.5.12 Sai_Ip_AbortTransfer()

```
void Sai_Ip_AbortTransfer (
            const uint8 u8Instance )
```

Abort all ongoing transferring data (both sending and receiving)

Parameters

in	*u8Instance*	Peripheral instance number

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.