MCAL User Manual for Sent

32-bit TriCoreTM AURIXTM TC3xx microcontroller

About this document

Scope and purpose

This User Manual is intended to enable users to integrate the Microcontroller Abstraction Layer (MCAL) software for the TriCoreTM AURIXTM family of 32-bit microcontrollers.

This document describes responsibilities of integrator in-charge of integrating MCAL software with the basic software (BSW) stack. This document also provides detailed information on safety, configuration and functions along with examples of usage of significant features.

Note: Detailed information about package installation, safety and other generic information that are common across all modules are provided in MCAL User Manual General.

Intended audience

This document is intended for anyone using the Sent module of the TC3xx MCAL software.

Document conventions

Table 1 Conventions

Convention	Explanation
Bold	Emphasizes heading levels, column headings, table and figure captions, screen names, windows, dialog boxes, menus, sub-menus
Italics	Denotes variable(s) and reference(s)
`Courier` `New`	Denotes APIs, functions, interrupt handlers, events, data types, error handlers, file/folder names, directories, command line inputs, code snippets
>	Indicates that a cascading sub-menu opens when you select a menu item
[cover parentID=<alpha numeric value>]	Used for traceability completeness. Reader should ignore these.

Reference documents

This User Manual should be read in conjunction with the following documents:

- AURIXTM TC3xx MCAL User Manual General

Table of contents

1 SENT driver

1.1 User information

1.1.1 Description

The SENT driver provides the necessary configuration parameters and APIs to communicate with the external sensors over single I/O line for each channel. The SENT driver is implemented as a post-build variant.

The features of the SENT are:

- SENT interface provides a serial communication link typically used to connect sensors or other peripheral devices
- Clock control, address decoding and service request control are managed by the SENT module kernel
- SENT IP-module performs communication according to the SENT specification J2716 JAN2010
- Short PWM Code (SPC) protocol enables the use of enhanced protocol functionality like synchronous, range selection and ID selection protocol mode
- Message storage consists of two 32-bit registers for each channel, representing a flexible double buffer system

1.1.2 Hardware-software mapping

This section describes the system view of the SENT driver and peripherals administered by it.

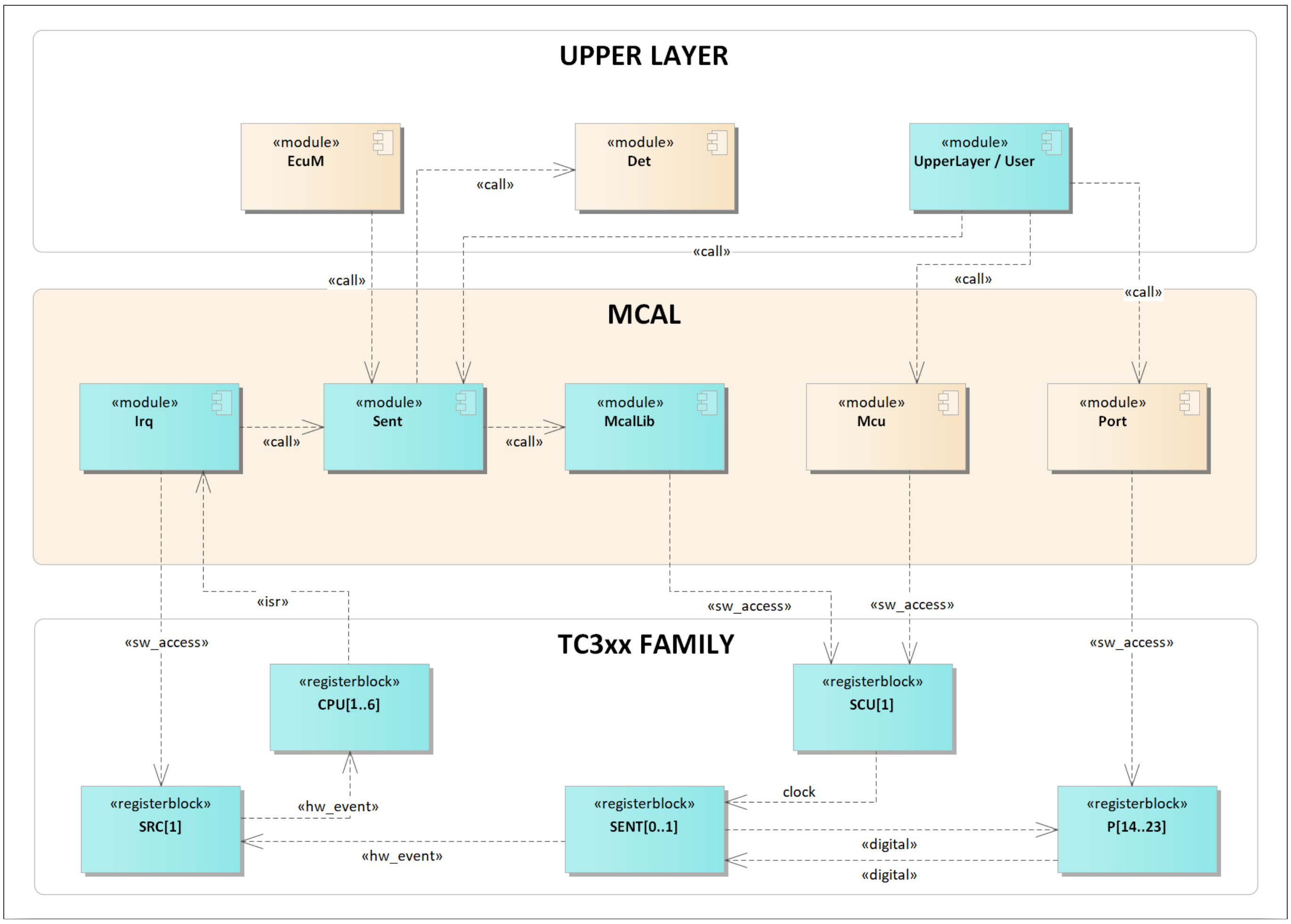


Figure 1 Mapping of hardware-software interfaces

1.1.2.1 SENT: primary hardware peripheral

Hardware functional features

The hardware features of each functional block configured by the driver are listed as follows:

- Reception of data in conformance according to the SENT standard
- Support for standard channel tick times (1 µs – 90 µs)
- Support for the SPC mode
- Digital glitch filter suppressing noise
- Time stamp generation
- Watchdog timer on incoming frames
- Interrupt generation for data reception, protocol error, buffer under-run, buffer over-run, watchdog error interrupts

Users of the hardware

The SENT driver exclusively utilizes the SENT module for its functionality.

Hardware diagnostic features

The SMU alarms configured for the SENT are not monitored by the SENT driver.

Hardware events

The SENT driver uses the following hardware events from the SENT IP:

- Receive success interrupt

- Receive data interrupt
- Receive buffer overflow interrupt
- Transfer data interrupt
- Transmit buffer underflow interrupt
- Frequency range interrupt
- Frequency drift interrupt
- Wrong number of nibble interrupt
- Nibble value out of range interrupt
- CRC error interrupt
- Wrong status and communication nibble interrupt
- Serial data receive interrupt
- Watch dog error interrupt

1.1.2.2 SCU: dependent hardware peripheral

Hardware functional features

The SENT driver depends on the SCU IP for the clock, ENDINIT and reset functionalities. The driver requires the fSPB and fSENT clock signals for functioning.

Users of the hardware

The SCU IP supplies clock for all the peripherals and the MCU driver, and is responsible for configuring the clock tree. To avoid conflicts due to simultaneous writes, update to all the ENDINIT protected registers is performed using the MCALLIB APIs.

Hardware diagnostic features

The SMU alarms configured for the SCU IP are not monitored by the SENT driver.

Hardware events

Hardware events from the SCU are not used by the SENT driver.

1.1.2.3 Port: dependent hardware peripheral

Hardware functional features

The SPC data from SENT and the sensor data to the SENT and signal is routed to the SENT through the port pads. This is configured and enabled through the PORT driver

Users of the hardware

The port pads are configured by the PORT driver.

Hardware diagnostic features

Not applicable.

Hardware events

Hardware events from port pads are not used by the SENT driver.

1.1.2.4 SRC: dependent hardware peripheral

Hardware functional features

The SENT driver depends on the interrupt router for raising an interrupt to the CPU based on transmit, receive and error events, which indicates successful data transmission, reception and failure respectively.

Users of the hardware

The interrupt router is configured either by the IRQ driver or the user software. No functional block of the interrupt router is administrated by the SENT driver

Hardware diagnostic features

The SMU alarms configured for the interrupt router are not monitored by the SENT driver.

Hardware events

The interrupt events raised by the interrupt router are serviced by the CPU. The SENT driver provides interrupt handlers as software interfaces, which must be invoked from the ISR.

1.1.3 File structure

1.1.3.1 C file structure

This section provides details of the C files of the SENT driver.

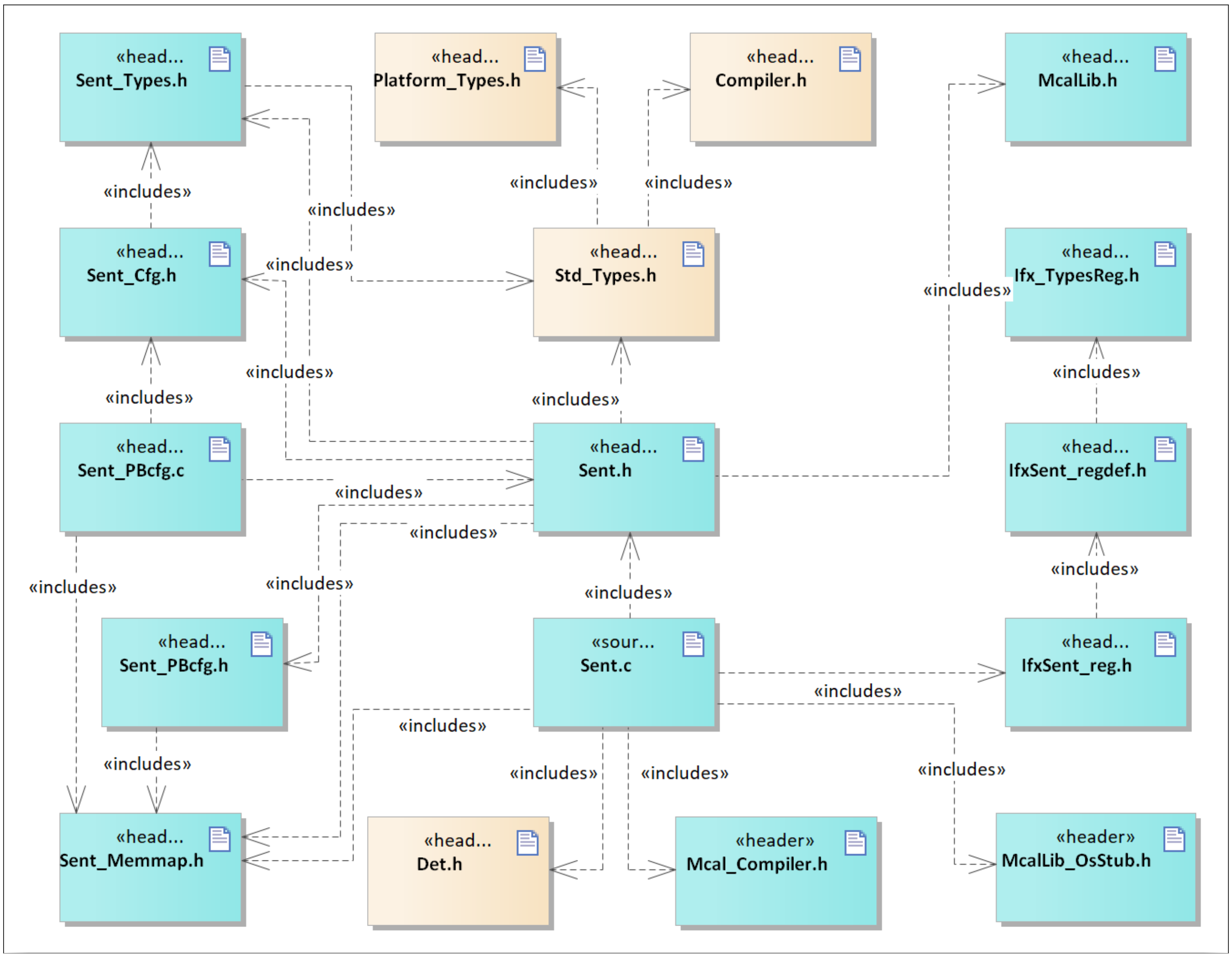


Figure 2 **C file structure**

Table 2 **C file structure**

File name	Description
`Platform_Types.h`	Platform-specific type declaration file as defined by AUTOSAR

(table continues...)

Table 2 (continued) C file structure

File name	Description
`Std_Types.h`	Standard type declaration file as defined by AUTOSAR. It is independent of compiler or platform.
`Compiler.h`	Provides macros for the encapsulation of definitions and declarations
`Det.h`	Provides the exported interfaces of DET
`McalLib.h`	Header file (Static) defining prototypes of data structures and APIs of end-init and delay services and included by McalLib.c
`McalLib_OsStub.h`	McalLib_OsStub.h provides macros to support user mode of TriCore™
`Sent_Types.h`	The header file includes general LIN type declarations
`Sent_MemMap.h`	Mapping of code and data (variables, constant variables) to specific memory sections
`Sent.h`	Contains macros, type definitions and function prototypes of the SENT driver
`Sent.c`	Implementation of SENT driver functionality
`Sent_Cfg.h`	The pre-compile configuration macros required for the SENT driver implementation are present in this file
`Sent_PBcfg.h`	Contains SENT driver post build configuration parameter declaration
`Sent_PBcfg.c`	Contains SENT driver post build configuration parameters
`IfxSent_reg.h`	SFR header file for the SENT
`IfxSent_regdef.h`	Includes the register definition file for the SENT
`Ifx_TypesReg.h`	SFR header file

1.1.3.2 Code generator plugin files

This section provides details of the code generator plugin files of the SENT driver.

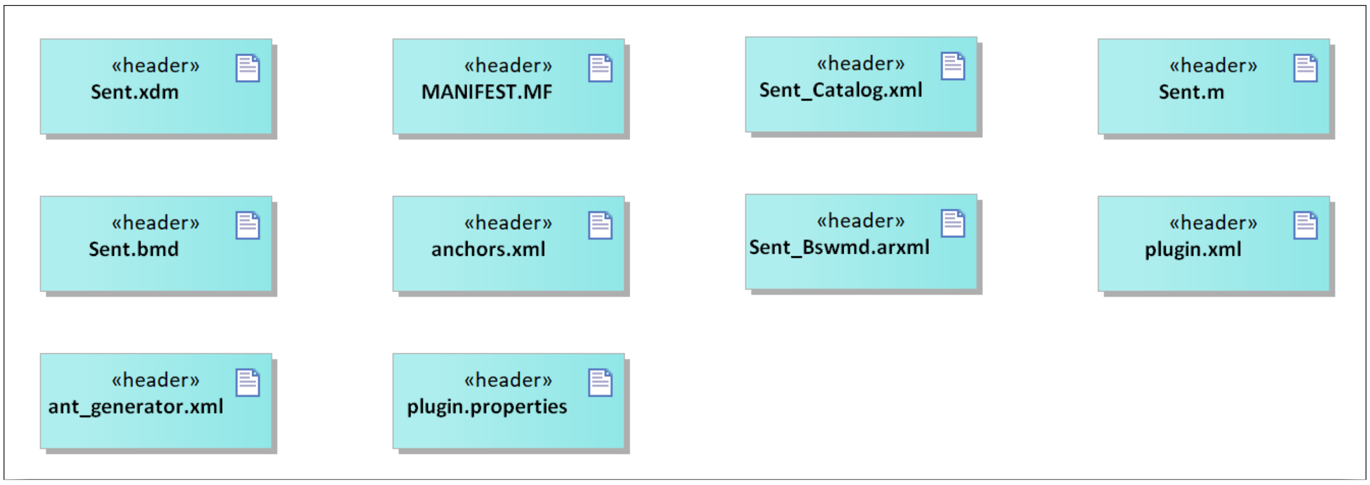


Figure 3 Code generator plugin files

Table 3 Code generator plugin files

File name	Description
`anchors.xml`	Tresos anchors support file for the SENT driver
`plugin.xml`	Tresos plugin support file for the SENT driver
`plugin.properties`	Tresos plugin support file for the SENT driver
`MANIFEST.MF`	Tresos plugin support file containing the metadata for the SENT driver
`ant_generator.xml`	Tresos support file to generate and rename multiple Post-Build configuration when using variation point feature
`Sent_Bswmd.arxml`	AUTOSAR format module description file
`Sent_Catalog.xml`	AUTOSAR format catalog file
`Sent.bmd`	AUTOSAR format XML data model schema file (for each device)
`Sent.m`	Code template macro file for the SENT driver
`Sent.xdm`	Tresos format XML data model schema file

1.1.4 Integration hints

This section lists the key points that an integrator or user of the SENT driver must consider.

1.1.4.1 Integration with AUTOSAR stack

This section lists the modules that are not part of the MCAL, but are required to integrate the SENT driver.

- **EcuM**

  The ECU Manager module is a part of the AUTOSAR stack that manages common aspects of ECU. Specifically, in the context of MCAL, EcuM is used for initialization and de-initialization of the software drivers. The EcuM module provided in the MCAL package is a stub code and needs to be replaced with a complete EcuM module during the integration phase.

- **Memory mapping**

  Memory mapping is a concept from AUTOSAR that allows relocation of text, variables, constants and configuration data to user-specific memory regions. To achieve this, all the relocatable elements of the driver are encapsulated in different memory-section macros. These macros are defined in the `Sent_MemMap.h` file. The `Sent_MemMap.h` file is provided in the MCAL package as a stub code. The integrator must place appropriate compiler pragmas within the memory-section macros. The pragmas ensure that the elements are relocated to the correct memory region. A sample implementation listing the memory-section macros is shown below.

```
/***** GLOBAL RAM DATA -- NON CLEARED LMU *****/
#if defined SENT_START_SEC_VAR_CLEARED_QM_GLOBAL_8
/******User pragmas here *******/
#undef  SENT_START_SEC_VAR_CLEARED_QM_GLOBAL_8
#undef MEMMAP_ERROR
#elif defined SENT_STOP_SEC_VAR_CLEARED_QM_GLOBAL_8
/******User pragmas here *******/
#undef  SENT_STOP_SEC_VAR_CLEARED_QM_GLOBAL_8
#undef MEMMAP_ERROR
#elif defined SENT_START_SEC_VAR_CLEARED_QM_GLOBAL_32
/******User pragmas here *******/

#undef  SENT_START_SEC_VAR_CLEARED_QM_GLOBAL_32
#undef MEMMAP_ERROR
#elif defined SENT_STOP_SEC_VAR_CLEARED_QM_GLOBAL_32
/******User pragmas here *******/
#undef  SENT_STOP_SEC_VAR_CLEARED_QM_GLOBAL_32
#undef MEMMAP_ERROR
/**** CORE[x] CONFIG DATA -- PF[x] ****/ /*[x]=0..5*/
#elif defined SENT_START__SEC__CONFIG_DATA_QM_CORE[x]0_UNSPECIFIED
/******User pragmas here for PF[x] *******/
#undef SENT_START_SEC_CONFIG_DATA_QM_CORE[x]0_UNSPECIFIED
#undef MEMMAP_ERROR

#elif defined SENT_STOP_SEC_CONFIG_DATA_QM_CORE[x]0_UNSPECIFIED
/******User pragmas here for PF[x] *******/
#undef SENT_STOP_SEC_CONFIG_DATA_QM_CORE[x]0_UNSPECIFIED
#undef MEMMAP_ERROR

/***** CODE -- PF[x] *****/
#elif defined SENT_START_SEC_CODE_QM_GLOBAL
/******User pragmas here for PF[x] *******/
#undef SENT_START_SEC_CODE_QM_GLOBAL
#undef MEMMAP_ERROR

#elif defined SENT_STOP_SEC_CODE_QM_GLOBAL
/******User pragmas here for PF[x] *******/
#undef SENT_STOP_SEC_CODE_QM_GLOBAL
#undef MEMMAP_ERROR
#endif

#if defined MEMMAP_ERROR
#error "Sent_MemMap.h, wrong pragma command"
#endif
```

- **DET**

  The DET module is a part of the AUTOSAR stack that handles all the development and runtime errors reported by the BSW modules. The SENT driver reports all the development errors to the DET module through the `Det_ReportError()` API. The user of the SENT driver must process all the errors reported to the DET module through the `Det_ReportError()` API. The `Det.h` and `Det.c` files are provided in the MCAL package as a stub code and needs to be replaced with a complete DET module during the integration phase.

- **DEM**

  The DEM module is not required for the integration of the SENT driver.
- **SchM**

  The SchM is not required for the integration of the SENT driver.
- **Safety error**

  The SENT driver does not report any safety errors.
- **Notifications and callbacks**

  A callout function is linked uniquely with a SENT channel to be notified with the channel's interrupt events or any error/status events. The callout function prototype is defined by Sent_NotifFnPtrType. The callout functions fall under the MCAL layer and are allowed to access SENT registers if required. The application can determine the necessary action based on the event notifications. It is the responsibility of the user to define the SENT callout functions.
- **Operating system**

  OS or application must ensure correct type of service and interrupt priority is configured in the SR register. Enabling and disabling of interrupts must also be managed by the OS or application. The OS files provided by the MCAL package is only an example code and must be updated by the integrator with the actual OS files for the desired function.

1.1.4.2 Multicore and Resource Manager

The SENT driver supports execution of its APIs in parallel from all CPU cores. The user has to allocate resources of SENT to CPU cores at pre-compile time using the Resource Manager module. The following are the key points to be considered with respect to multicore in the driver:

- SENT channel of the SENT driver can be allocated to the CPU cores at pre-compile time.
- SENT channels that are not allocated to a CPU core shall be by default allocated to the master core.
- It must be ensured that the SENT channel ID passed as a parameter while invoking an API belongs to the same core on which the API is invoked.
- Initialization of the SENT channel must start with the master core initialization only after the successful initialization of the master core should there be a trigger for a slave core initialization. The SENT driver of the slave cores can be initialized simultaneously.
- De-initialization of the SENT driver for different slave cores can be initiated simultaneously. The master core de-initialization of the SENT driver should be carried out only after the de-initialization of the SENT driver in all the slave cores.
- DETs will be raised in case APIs are invoked with mismatch of CPU core and controller IDs or hardware object IDs.
- Interrupts raised by a hardware group must be serviced by the CPU core to which the hardware group has been allocated to.
- Locating constants, variables and configuration data to correct memory space should be done by the user. Memory sections are marked GLOBAL (common to all cores) and CORE[x](specific to a CPU core). The following should be considered by the user to ensure better performance of the driver:

  Code section

  The executable code of the SENT driver is placed under single MemMap section. It can be relocated to any PFlash region.

Data section

The RAM variable memory sections marked as specific to a core should be relocated to the DSPR/DLMU of the same core. The sections marked as global should be relocated to the non-cached LMU region.

Configuration data and constants

The configuration data sections marked as specific to a core should be relocated to the PFlash of the same core. The sections marked as global should be relocated to the PFlash of the master core.

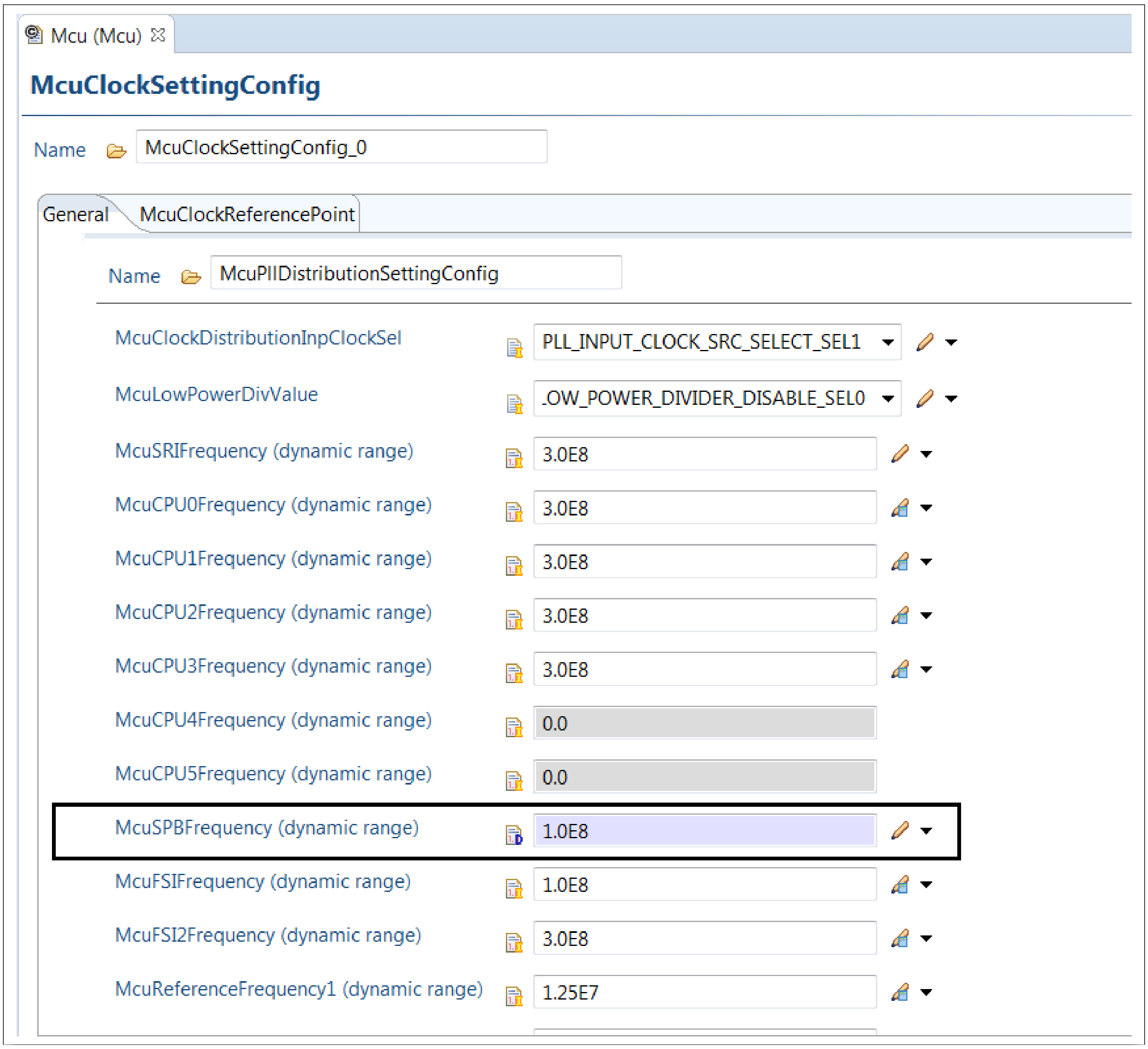
Note: Relocating code, data or constants to a distant memory region would impact execution timings.

Note: If the driver operates from a single (master) core, all the sections may be relocated to the PFlash/ DSPR/DLMU of the same CPU core.

1.1.4.3 MCU support

The SENT driver is dependent on the MCU driver for the clock configuration. The initialization of SENT driver must be started only after completion of the MCU initialization. The following must be considered while configuring the MCU driver in the EB tresos:

The fSENT defines the application clock frequency for the SENT Kernel. The fSENT which is derived from SPB (100 MHz) allows the SENT to operate at a constant baud rate (frequency). The required fSENT is 100 MHz.


Mcu (Mcu)

McuClockSettingConfig

Name: McuClockSettingConfig_0

General | McuClockReferencePoint

Name: McuPllDistributionSettingConfig

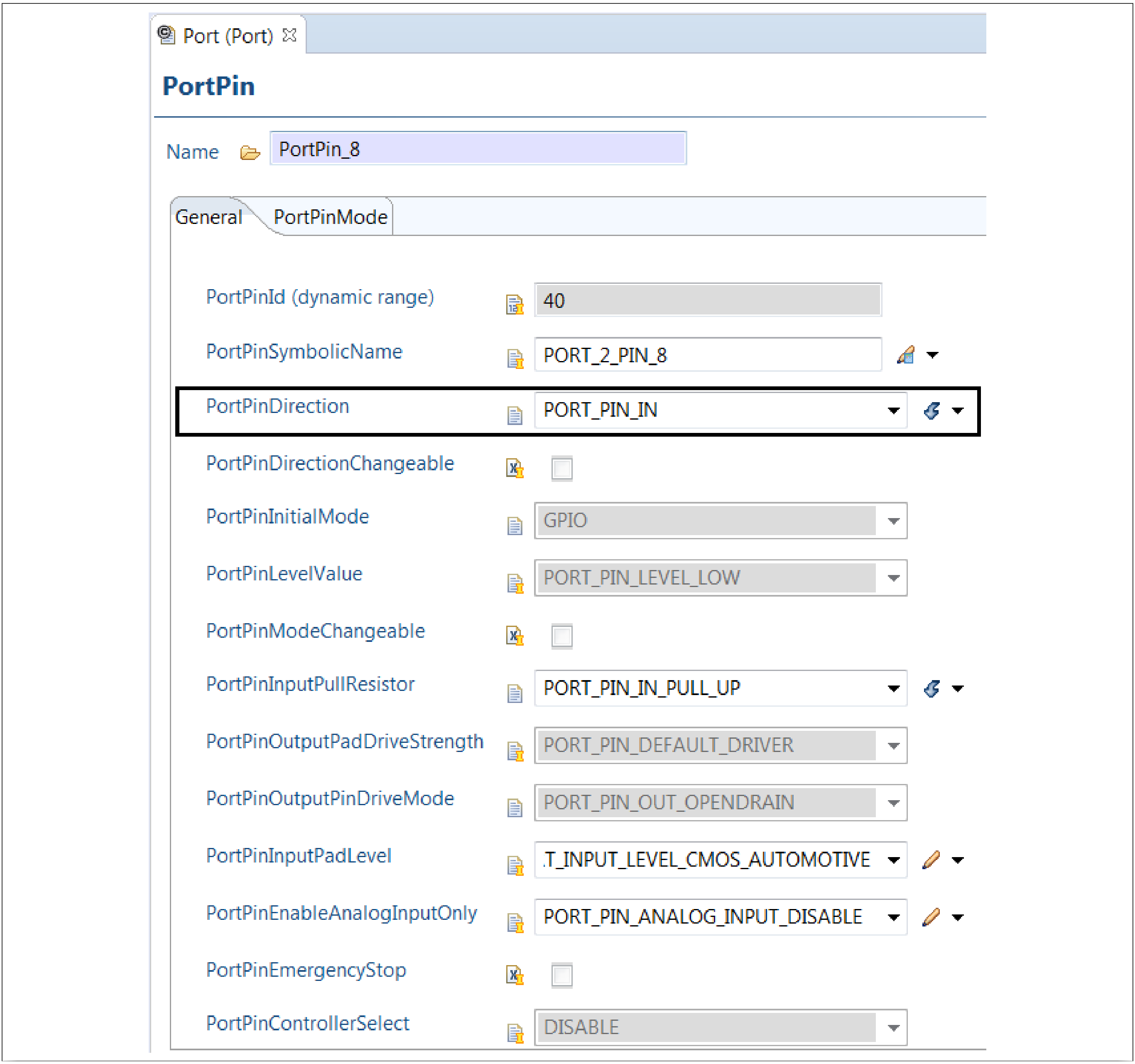
McuClockDistributionInpClockSel	PLL_INPUT_CLOCK_SRC_SELECT_SEL1
McuLowPowerDivValue	.OW_POWER_DIVIDER_DISABLE_SEL0
McuSRIFrequency (dynamic range)	3.0E8
McuCPU0Frequency (dynamic range)	3.0E8
McuCPU1Frequency (dynamic range)	3.0E8
McuCPU2Frequency (dynamic range)	3.0E8
McuCPU3Frequency (dynamic range)	3.0E8
McuCPU4Frequency (dynamic range)	0.0
McuCPU5Frequency (dynamic range)	0.0
McuSPBFrequency (dynamic range)	1.0E8
McuFSIFrequency (dynamic range)	1.0E8
McuFSI2Frequency (dynamic range)	3.0E8
McuReferenceFrequency1 (dynamic range)	1.25E7

Figure 4 MCU Configuration

1.1.4.4 Port support

The PORT driver configures the port pins of the entire microcontroller. The user must configure the port pins used by the SENT driver through the port configuration and initialize the port pins prior to invoking the SENT initialization.

- **Port configuration for the Standard Sent operation**


Port (Port)

PortPin

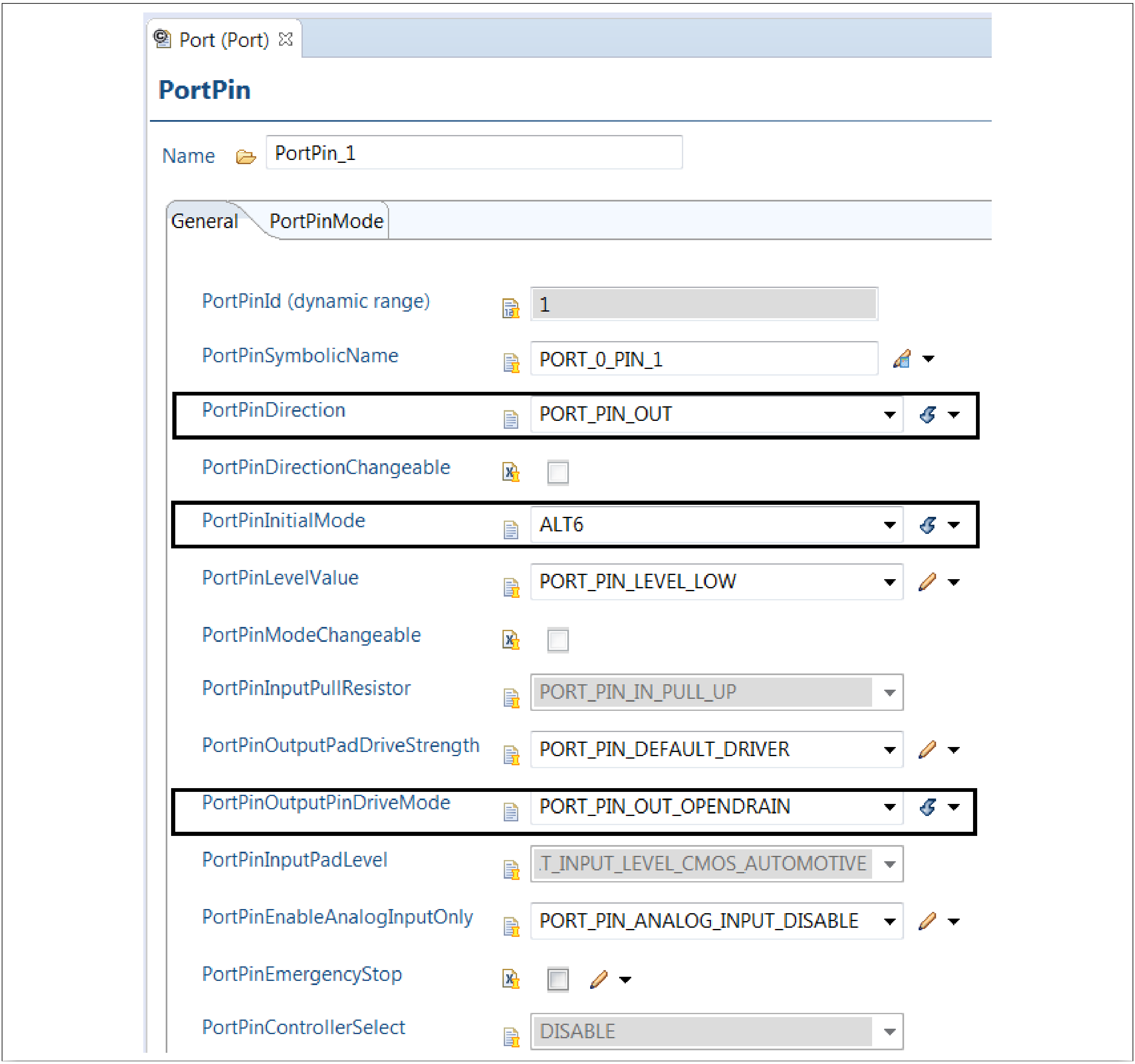
Name	PortPin_8

General | PortPinMode

PortPinId (dynamic range)	40
PortPinSymbolicName	PORT_2_PIN_8
PortPinDirection	PORT_PIN_IN
PortPinDirectionChangeable	☐
PortPinInitialMode	GPIO
PortPinLevelValue	PORT_PIN_LEVEL_LOW
PortPinModeChangeable	☐
PortPinInputPullResistor	PORT_PIN_IN_PULL_UP
PortPinOutputPadDriveStrength	PORT_PIN_DEFAULT_DRIVER
PortPinOutputPinDriveMode	PORT_PIN_OUT_OPENDRAIN
PortPinInputPadLevel	.T_INPUT_LEVEL_CMOS_AUTOMOTIVE
PortPinEnableAnalogInputOnly	PORT_PIN_ANALOG_INPUT_DISABLE
PortPinEmergencyStop	☐
PortPinControllerSelect	DISABLE

Figure 5 **Port Configuration for standard mode**

- **Port configuration for the SPC mode**


Port (Port)

PortPin

Name	PortPin_1

General | PortPinMode

PortPinId (dynamic range)	1
PortPinSymbolicName	PORT_0_PIN_1
PortPinDirection	PORT_PIN_OUT
PortPinDirectionChangeable	☐
PortPinInitialMode	ALT6
PortPinLevelValue	PORT_PIN_LEVEL_LOW
PortPinModeChangeable	☐
PortPinInputPullResistor	PORT_PIN_IN_PULL_UP
PortPinOutputPadDriveStrength	PORT_PIN_DEFAULT_DRIVER
PortPinOutputPinDriveMode	PORT_PIN_OUT_OPENDRAIN
PortPinInputPadLevel	T_INPUT_LEVEL_CMOS_AUTOMOTIVE
PortPinEnableAnalogInputOnly	PORT_PIN_ANALOG_INPUT_DISABLE
PortPinEmergencyStop	☐
PortPinControllerSelect	DISABLE

Figure 6 **Port configuration for SPC mode**

1.1.4.5 DMA support

The SENT driver does not use any services provided by the DMA driver.

1.1.4.6 Interrupt connections

The interrupt connections of the SENT driver are described in this section.

The SENT driver is responsible for handling the SENT channel-specific interrupt requests and call the channel-specific registered callout function. Also, the callback functions/notifications configured should be unique for different channels. The SENT SRN interrupt handler shall invoke ISR Sent_Isr with the relevant interrupt node number. Also, each channel’s interrupts are limited to a single interrupt node only. There are only 10 interrupt node available for SENT. User can configure one interrupt node to more than one SENT physical channel (for example, SENT Physical Channel 0 linked to SRN2, SENT Physical Channel 1 linked to SRN0 and SENT Physical Channel 2 linked to SRN2 and so on).

RDI indicates a receive data interrupt. It is activated when a received frame is moved to a Receive Data Register (RDR). RSI indicates a receive frame success interrupt, that is, the CRC was successful. Both RDI and RSI will be issued together in normal use cases where the frame size is not bigger than 8 nibbles and the CRC is correct. RBI indicates a receive buffer overrun interrupt. It is activated when a new frame is transferred to a Receive Data Register RDR while the old value was still not read by the host (overwrite), that is, the kernel wants to set any of the two interrupts RSI and RDI and finds any of these two interrupts already set. TDI indicates a transmit interrupt. It is activated when data is moved from a SCR to a transmit shift register. TBI indicates a transfer buffer under run interrupt. It is set after data has been completely transferred (PLEN exceeded) and no new data was written to SCRx. In addition, the protocol error interrupts are available: FRI, FDI, NNI, NVI and CRCI. If one of the protocol interrupts is activated, data is to be treated as invalid according to SENT specification J2716 JAN2010. WSI, SDI SCRI treats the interrupts referring to the Status and Communication nibble. WDI is the Watch Dog Error Interrupt. It is issued if the time between two frames is too long.

```
#include "Sent.h"’
ISR(SENTSR0_ISR)
{
  /* Enable Global Interrupts */
  ENABLE();
Sent_Isr(0);
}
```

1.1.4.7 Example usage

Examples of SENT driver API usage are as follows:

1.1.4.7.1 Configuration of the driver

The SENT driver must be configured before usage and configuration files are generated and made available during the software build process.

To configure the SENT driver, the following guidelines shall be followed properly.

- Configuration of system clock: Before using the SENT driver, the MCU driver needs to be configured and initialized for the system clock and the system peripheral bus (SPB) clock. The SENT driver clock is derived from the SPB clock. This configuration is done using the MCU driver.
- Configuration of the port pins: For all the port pins that would be used by the SENT driver as input/output pins, configure the same in the PORT driver.
- Configuration of SENT interrupts: Configure the interrupt priority, type of service and interrupt type in the IRQ driver.
- Configuration of SENT driver: Select the required API configuration and choose channel dependent parameters like baud-rate, data length of the frame, CRC mode and so on.

Initialization of SENT driver

Refer to the *Integration hints* section and add all dependent modules. Follow the sequence in the application code:

1. Initialize the MCU and the clock `Mcu_Init` API.
2. Initialize the PORT driver using the `Port_Init` API.
3. Initialize the IRQ to enable the interrupt generation.
4. Initialize the SENT driver using the `Sent_Init` API.

Sample code for SENT driver initialization is as follows:

```
/* Mcu Initialization */
 Mcu_Init(&Mcu_Config);
 Mcu_InitClock(0U);
 while(Mcu_GetPllStatus() != MCU_PLL_LOCKED);
 Mcu_DistributePllClock ();
/* Port Initialization */
 Port_Init(&Port_Config);
/* SENT Initialization */
 Sent_Init(&Sent_Config);
/* Further APIs of SENT driver can be called now */
```

Enabling and disabling the channel

After SENT initialization the following sequence can be followed.

```
/* Enable Channel */
 Sent_SetChannel(ChannelId_0, SENT_ENABLE);
/* Disable Channel */
 Sent_SetChannel(ChannelId_0, SENT_DISABLE);
```

Reading data from standard SENT mode

```
/* Mcu Initialization */
  Mcu_Init(&Mcu_Config);
  Mcu_InitClock(0U);
  while(Mcu_GetPllStatus() != MCU_PLL_LOCKED);
  Mcu_DistributePllClock ();
 /* Port Initialization */
  Port_Init(&Port_Config);
 /* SENT Initialization */
  Sent_Init(&Sent_Config);
 /* Enable Channel */
  Sent_SetChannel(ChannelId_0, SENT_ENABLE);
  Delay(3);
  Sent_ReadData0 = Sent_ReadData(ChannelId_0);
```

Reading the data from SPC mode

```
/* Mcu Initialization */
  Mcu_Init(&Mcu_Config);
  Mcu_InitClock(0U);
  while(Mcu_GetPllStatus() != MCU_PLL_LOCKED);
  Mcu_DistributePllClock ();
 /* Port Initialization */
  Port_Init(&Port_Config);
 /* SENT Initialization */
  Sent_Init(&Sent_Config);
 /* Enable Channel */
  Sent_SetChannel(ChannelId_0, SENT_ENABLE);
  Delay(3);
  #if (SENT_SPC_USED == STD_ON)
  Sent_Spc.Mode = SYNC_MODE;
  Sent_Spc.Delay = 0;
  Sent_Spc.PulseLength = 3; /* 3 ticks */
  Sent_Spc.TimeBase = PULSE_LAST_SYNC_FREQ;
  Sent_Spc.TriggerSource = PULSE_START_IMMED;
  Sent_SpcGenPulse(ChannelId_0, &Sent_Spc);
  #endif
  Sent_ReadChannelStatus(ChannelId_0, &Sent_Stat);
  Sent_ReadData0 = Sent_ReadData(ChannelId_0);
```

1.1.5 Key architectural considerations

There are no key architectural considerations for the driver.

1.2 Assumptions of Use (AoU)

There are no AoU for the SENT driver.

1.3 Reference information

1.3.1 Configuration interfaces

The following diagram depicts the hierarchy along with the extensions provided for SENT module.

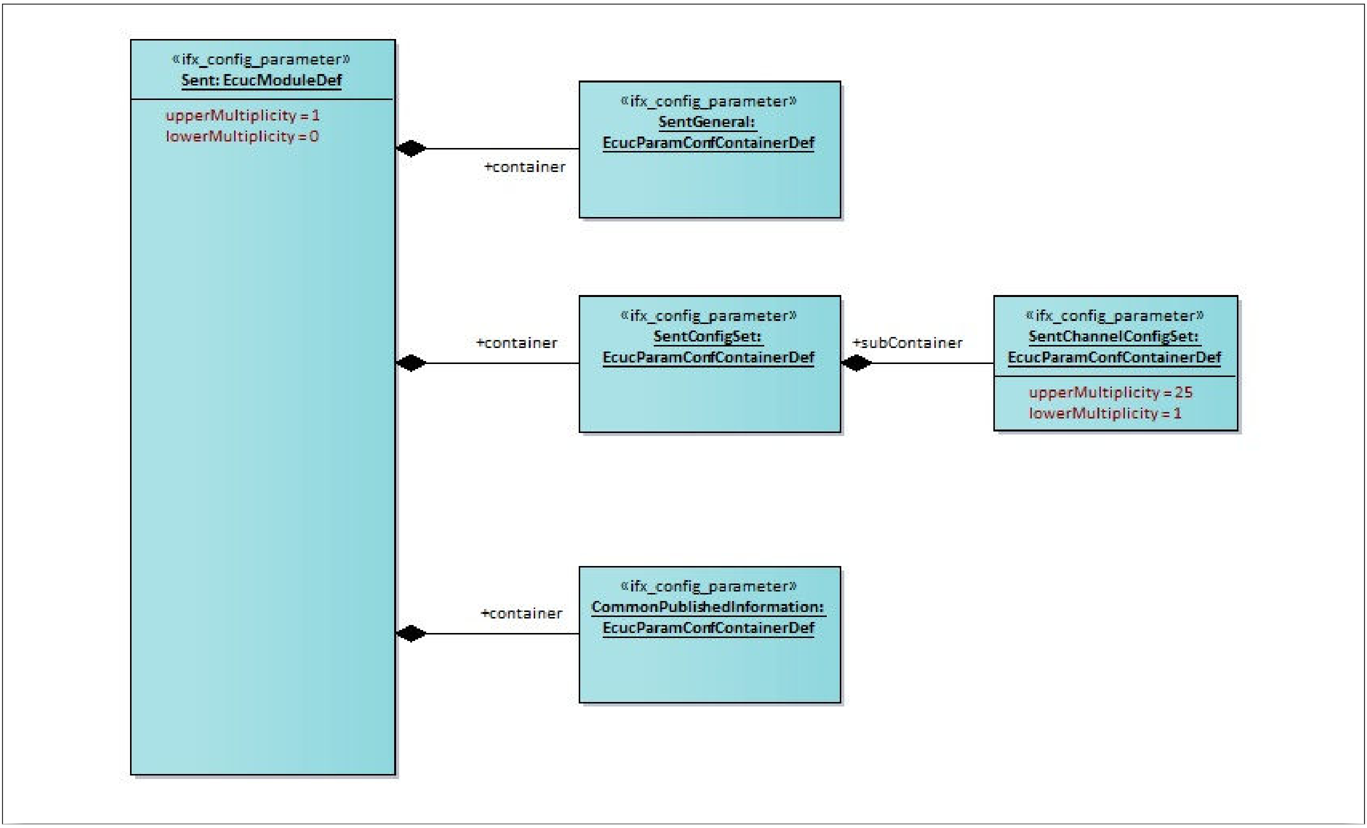


Figure 7 **Container hierarchy along with their configuration parameters**

1.3.1.1 Container: CommonPublishedInformation

This container contains published information about vendor and versions.

1.3.1.1.1 ArMajorVersion

Table 4 **Specification for ArMajorVersion**

Name	`ArMajorVersion`		
Description	Parameter provides the major version of the AUTOSAR specification.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 255		
Default value	4		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		

1.3.1.1.2 ArMinorVersion

Table 5 Specification for ArMinorVersion

Name	`ArMinorVersion`		
Description	Parameter provides the minor version of the AUTOSAR specification.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 255		
Default value	As per AUTOSAR minor version.		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		

1.3.1.1.3 ArPatchVersion

Table 6 Specification for ArPatchVersion

Name	`ArPatchVersion`		
Description	Parameter provides the patch version of the AUTOSAR specification.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 255		
Default value	As per AUTOSAR patch version.		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		

1.3.1.1.4 ModuleId

Table 7 Specification for ModuleId

Name	`ModuleId`		
Description	This parameter provides the module Id. The default value is set to 255 as this is the module ID of the SENT driver.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	0 - 255		
Default value	255		

(table continues...)

Table 7 (continued) Specification for ModuleId

Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		

1.3.1.1.5 SwMajorVersion

Table 8 Specification for SwMajorVersion

Name	`SwMajorVersion`		
Description	Specifies the major version of the driver software.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 255		
Default value	As per Driver		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	Scope	LOCAL
Dependency	-		

1.3.1.1.6 SwMinorVersion

Table 9 Specification for SwMinorVersion

Name	`SwMinorVersion`		
Description	Specifies the minor version of the driver software.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 255		
Default value	As per Driver		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		

1.3.1.1.7 SwPatchVersion

Table 10 Specification for SwPatchVersion

Name	`SwPatchVersion`		
Description	Specifies the patch version of the driver software.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 255		
Default value	As per Driver		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		

1.3.1.1.8 VendorId

Table 11 Specification for VendorId

Name	`VendorId`		
Description	Specifies the vendor ID for Infineon.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 65535		
Default value	17		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		

1.3.1.2 Container: Sent

This container contains the general configuration parameters of the SENT driver

1.3.1.2.1 Config Variant

Table 12 Specification for Config Variant

Name	`Config Variant`		
Description	Selects the config-variant for the SENT module.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef

(table continues...)

Table 12 (continued) Specification for Config Variant

Range	VariantPostBuild: Post Build Support.		
Default value	VariantPostBuild		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		

1.3.1.2.2 SentDeInitApi

Table 13 Specification for SentDeInitApi

Name	`SentDeInitApi`		
Description	Switches the DeInit Api ON or OFF. TRUE: enabled (ON). FALSE: disabled (OFF).		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		

1.3.1.2.3 SentDevErrorDetect

Table 14 Specification for SentDevErrorDetect

Name	`SentDevErrorDetect`		
Description	Switches the Default Error Tracer (Det) detection and notification ON or OFF. TRUE: enabled (ON) FALSE: disabled (OFF)		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		

(table continues...)

Table 14 (continued) Specification for SentDevErrorDetect

Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		

1.3.1.2.4 SentSpcFeatureSupport

Table 15 Specification for SentSpcFeatureSupport

Name	`SentSpcFeatureSupport`		
Description	Switches the SPC feature support ON or OFF. TRUE: enabled (ON) FALSE: disabled (OFF)		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		

1.3.1.2.5 SentVersionInfoApi

Table 16 Specification for SentVersionInfoApi

Name	`SentVersionInfoApi`		
Description	Switches the Sent_GetVersionInfo function ON or OFF.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-

(table continues...)

Table 16 (continued) Specification for SentVersionInfoApi

Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		

1.3.1.2.6 SentIndex

Table 17 Specification for SentIndex

Name	`SentIndex`		
Description	Specifies the Instance Id of this module instance. If only one instance is present it shall have the Id 0.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 255		
Default value	0		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		

1.3.1.2.7 SentResetSfrAtInit

Table 18 Specification for SentResetSfrAtInit

Name	`SentResetSfrAtInit`		
Description	Switches the SFR reset at initialization ON or OFF. TRUE: enabled (ON) FALSE: disabled (OFF)		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		

1.3.1.2.8 SentInitDeInitApiMode

Table 19 Specification for SentInitDeInitApiMode

Name	`SentInitDeInitApiMode`		
Description	Defines the mode in which the Init and DeInit APIs will be used. The default value of this parameter is set to Supervisor to enable maximum access rights to the registers used by the SENT driver.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	SENT_MCAL_SUPERVISOR: operating mode used is Supervisory SENT_MCAL_USER1: operating mode used is USER-1		
Default value	SENT_MCAL_SUPERVISOR		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		

1.3.1.2.9 SentMultiCoreErrorDetect

Table 20 Specification for SentMultiCoreErrorDetect

Name	`SentMultiCoreErrorDetect`		
Description	Switches the multi-core error detection and notification to ON or OFF. - TRUE: enabled (ON) - FALSE: disabled (OFF)		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		

1.3.1.3 Container: SentConfigSet

This container contains the module kernel specific configuration parameters.

1.3.1.3.1 SentSystemClock

Table 21 Specification for SentSystemClock

Name	`SentSystemClock`		
Description	This parameter refers to the system clock configured by MCU driver. This reference is used for BaudRate computation.		
Multiplicity	1..1	**Type**	EcucReferenceDef
Range	Reference to Node: McuClockReferencePointConfig		
Default value	NULL		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		

1.3.1.3.2 SentSleepModeEnable

Table 22 Specification for SentSleepModeEnable

Name	`SentSleepModeEnable`		
Description	Switches the SentSleepModeEnable ON or OFF. TRUE: enabled (ON). FALSE: disabled (OFF).		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		

1.3.1.3.3 SentModuleClkDiv

Table 23 Specification for SentModuleClkDiv

Name	`SentModuleClkDiv`

(table continues...)

Table 23 (continued) Specification for SentModuleClkDiv

Description	This parameter refers to the 8-bit divider used to generate the SENT module clock. This value will be used to divide the MCU SPB clock f_{Spb} and derive the f_{Sent} SENT module clock.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	1 – 255		
Default value	1		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		

1.3.1.3.4 SentBaudFracStep

Table 24 Specification for SentBaudFracStep

Name	`SentBaudFracStep`		
Description	This parameter value will generate the SENT fractional divider clock $f_{fracdiv}$ which is an input clock for all SENT channels. This parameter derives the clock as follows: $f_{fracdiv} = f_{SENT} / (1024 – SentBaudFracStep)$ where SentBaudFracStep = 0 - 1023.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	0 – 1023		
Default value	1023		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Post Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	SentModuleClkDiv		

1.3.1.4 Container: SentChannelConfigSet

This container contains the channel specific configuration parameters.

1.3.1.4.1 SentChLogiIndex

Table 25 Specification for SentChLogiIndex

Name	`SentChLogiIndex`
Description	This parameter refers to SENT logical channel number.

(table continues...)

Table 25 (continued) Specification for SentChLogiIndex

Multiplicity	1	**Type**	EcucBooleanParamDef
Range	0 – (x-1) where n is the Maximum No. of SENT channels available in a particular device.		
Default value	x where x is the index of the Sent Channel in the Config set.		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Post Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	SentModuleClkDiv		

1.3.1.4.2 SentChanPreDiv

Table 26 Specification for SentChanPreDiv

Name	`SentChanPreDiv`		
Description	This parameter refers to the setting of SENT channel pre-divider clock f_{pdiv_x} where x depends on the device variant. This parameter derives the clock as follows: $f_{pdiv\_x} = f_{fracdiv}$ / (SentChanPreDiv + 1)		
Multiplicity	1	**Type**	EcucIntegerParamDef
Range	0 – 2047		
Default value	7		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Post Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	SentBaudFracStep		

1.3.1.4.3 SentChanBaudDiv

Table 27 Specification for SentChanBaudDiv

Name	`SentChanBaudDiv`		
Description	This parameter value is used to derive the baud rate frequency for channel x (f_{tick_x}) where x depends on the device variant. This parameter derives the baud rate as follows: $f_{tick\_x} = f_{pdiv\_x}$ * 56 / SentChanBaudDiv		
Multiplicity	1	**Type**	EcucIntegerParamDef
Range	2200 - 52428		

(table continues...)

Table 27 **(continued) Specification for SentChanBaudDiv**

Default value	2200		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Post Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	SentBaudFracStep		

1.3.1.4.4 SentChanCRCMode

Table 28 **Specification for SentChanCRCMode**

Name	`SentChanCRCMode`		
Description	This parameter decides the CRC mode to be used for fast channel/slow channel data communication.		
Multiplicity	1	**Type**	EcucEnumerationParamDef
Range	SENT_STANDARD: Standard CRC Calculation as per standard SENT_IFX_ALTERNATE: Alternative CRC Calculation as used in IFX Hall Sensors		
Default value	SENT_STANDARD		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Post Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	SentBaudFracStep		

1.3.1.4.5 SentChPhyIndex

Table 29 **Specification for SentChPhyIndex**

Name	`SentChPhyIndex`		
Description	This parameter refers to SENT physical channel number.		
Multiplicity	1	**Type**	EcucEnumerationParamDef
Range	SENT0: Signifies the physical channel 0. SENTx: This parameter signifies physical channel number, where x is varies from 0 to maximum number of units as per device variant. For example SENT0, SENT1,... SENTx, where x depends on device variant.		
Default value	SENT0		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-

(table continues...)

Table 29 (continued) Specification for SentChPhyIndex

Value configuration class	Post Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		

1.3.1.4.6 SentRxInput

Table 30 Specification for SentRxInput

Name	`SentRxInput`		
Description	This parameter selects the alternate input for the RX signal for the given Sent channel.		
Multiplicity	1	**Type**	EcucEnumerationParamDef
Range	SENT_0_A: Signifies the receive input channel 0. SENT_0_x: This parameter signifies the receive input channel, where x is varies from 0 to maximum number of units as per device variant. For example SENT0, SENT1, ..., SENTx, where x depends on device variant.		
Default value	SENT_0_C		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Post Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		

1.3.1.4.7 SentChanStatusNibbleCRCInc

Table 31 Specification for SentChanStatusNibbleCRCInc

Name	`SentChanStatusNibbleCRCInc`		
Description	This parameter defines whether status nibble should be used for CRC calculation.		
Multiplicity	1	**Type**	EcucBooleanParamDef
Range	FALSE: Status nibble not included for CRC calculation as per standard TRUE: Status nibble included for CRC calculation as used in IFX Hall Sensors		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Post Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		

1.3.1.4.8 SentChanEnESF

Table 32 Specification for SentChanEnESF

Name	`SentChanEnESF`		
Description	This parameter decides whether standard serial mode or extended serial encoding mode should be used. If standard serial mode is used, processing will be done after 16 SENT frames (4-bit ID, 8-bit data, 4-bit CRC). If extended serial mode is used, processing will be done after 18 SENT frames (4 or 8-bit ID, 12 or 16-bit data, 6-bit CRC).		
Multiplicity	1	**Type**	EcucBooleanParamDef
Range	FALSE: Standard serial data encoding used. TRUE: Extended serial data encoding used.		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Post Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		

1.3.1.4.9 SentChanSerialProcEn

Table 33 Specification for SentChanSerialProcEn

Name	`SentChanSerialProcEn`		
Description	This parameter decides whether automatic processing of serial data should be enabled or not. If enabled, serial data can be read through Sent_ReadSerialData once SDI interrupt has been activated. If not enabled, status nibble can be read manually through Sent_ReadChannelStatus for each SENT frame once RDI/RSI interrupt has been activated. The user should collate the serial data accordingly from the status nibbles of respective SENT frames as per standard		
Multiplicity	1	**Type**	EcucBooleanParamDef
Range	FALSE: Automatic serial data processing is disabled. TRUE: Automatic serial data processing is enabled.		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Post Build	**Multiplicity configuration class**	-

(table continues...)

Table 33 (continued) Specification for SentChanSerialProcEn

Origin	IFX	**Scope**	LOCAL
Dependency	-		

1.3.1.4.10 SentChanSerialCrcDisable

Table 34 Specification for SentChanSerialCrcDisable

Name	`SentChanSerialProcEn`		
Description	This parameter decides whether the serial data's CRC should be verified internally by SENT hardware. If TRUE, then it is responsibility of the application to verify the CRC of the received serial data.		
Multiplicity	1	**Type**	EcucBooleanParamDef
Range	TRUE: Serial data CRC not verified by SENT hardware FALSE: Serial data CRC verified by SENT hardware		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Post Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		

1.3.1.4.11 SentChanFrameCrcDisable

Table 35 Specification for SentChanFrameCrcDisable

Name	`SentChanFrameCrcDisable`		
Description	This parameter decides whether the serial data's CRC should be verified internally by SENT hardware. If TRUE, then it is responsibility of the application to verify the CRC of the received serial data.		
Multiplicity	1	**Type**	EcucBooleanParamDef
Range	TRUE: Serial data CRC not verified by SENT hardware FALSE: Serial data CRC verified by SENT hardware		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Post Build	**Multiplicity configuration class**	-

(table continues...)

Table 35 **(continued) Specification for SentChanFrameCrcDisable**

Origin	IFX	**Scope**	LOCAL
Dependency	-		

1.3.1.4.12 SentChanFrameChk

Table 36 **Specification for SentChanFrameChk**

Name	`SentChanFrameChk`		
Description	This parameter decides whether the current SENT frame should be verified against preceding SENT frame/ last valid preceding SENT frame for successive sync pulse difference (> 1.5625 %). If SENT_PAST_SYNC_PULSE is selected, the sync pulse of the current frame is compared to sync pulse of the immediate preceded frame. This is the preferred option as per standard. If SENT_PAST_VALID_SYNC_PULSE is selected, the sync pulse of the current frame is compared to sync pulse of the last valid preceded frame.		
Multiplicity	1	**Type**	EcucEnumerationParamDef
Range	SENT_PAST_SYNC_PULSE: Check current SENT frame against past sync pulse SENT_PAST_VALID_SYNC_PULSE: Check current SENT frame against last valid sync pulse		
Default value	SENT_PAST_SYNC_PULSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Post Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		

1.3.1.4.13 SentChanFrameDataLen

Table 37 **Specification for SentChanFrameDataLen**

Name	`SentChanFrameDataLen`		
Description	This parameter determines the number of data nibbles per SENT frame. It does not include sync pulse, status nibble, CRC nibble, or the additional zero length nibble. If more than 8 nibbles are configured, RDI interrupt is issued each time 8 nibbles are written into RDR register of that channel. At the end of the last data frame also, RDI interrupt is issued. If no RDI interrupt occurs at the last data frame, an error has occurred. RSI interrupt shall be issued at every successful receive of a single SENT frame.		
Multiplicity	1	**Type**	EcucEnumerationParamDef
Range	0 – 255		

(table continues...)

Table 37 (continued) Specification for SentChanFrameDataLen

Default value	6		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Post Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		

1.3.1.4.14 SentChanDriftErrEn

Table 38 Specification for SentChanDriftErrEn

Name	`SentChanDriftErrEn`		
Description	This parameter determines whether drift errors should be enabled or not. Certain sensors triggered by SPC tend to have a long pause period and the accumulated drift could be more than 1.5625%, then it useful to disable this feature.		
Multiplicity	1	**Type**	EcucBooleanParamDef
Range	FALSE: Ignore drift errors TRUE: Drift errors enabled		
Default value	TRUE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Post Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		

1.3.1.4.15 SentChanCRZEn

Table 39 Specification for SentChanCRZEn

Name	`SentChanCRZEn`		
Description	If TRUE, augmentation is selected, (i.e. a ZERO NIBBLE is added at the end of CRC calculation (only in calculation)). E.g. as 7th nibble (in case of 6 data nibbles).		
Multiplicity	1	**Type**	EcucBooleanParamDef
Range	FALSE: Zero nibble is not augmented for CRC calculation TRUE: Zero nibble is augmented for CRC calculation		
Default value	TRUE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-

(table continues...)

Table 39 (continued) Specification for SentChanCRZEn

Value configuration class	Post Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		

1.3.1.4.16 SentChanIgnoreEndPulse

Table 40 Specification for SentChanIgnoreEndPulse

Name	`SentChanIgnoreEndPulse`		
Description	This parameter determines whether end pulse should be ignored or not. For some systems with an end pulse, during synchronize or re-synchronize of reception, if calibration pulses are detected one immediately following the other, the first calibration pulse shall be ignored as it may be a pause pulse with duration matching the calibration pulse range.		
Multiplicity	1	**Type**	EcucBooleanParamDef
Range	FALSE: End pulse not ignored TRUE: End pulse ignored		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Post Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		

1.3.1.4.17 SentChanInPulse

Table 41 Specification for SentChanInPulse

Name	`SentChanInPulse`		
Description	This parameter determines the pulse polarity of the respective input channel.		
Multiplicity	1	**Type**	EcucEnumerationParamDef
Range	SENT_ACTIVE_LOW: Pulse polarity is active low SENT_ACTIVE_HIGH: Pulse polarity is active high		
Default value	SENT_ACTIVE_LOW		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Post Build	**Multiplicity configuration class**	-

(table continues...)

Table 41 (continued) Specification for SentChanInPulse

Origin	IFX	**Scope**	LOCAL
Dependency	-		

1.3.1.4.18 SentChanOutPulse

Table 42 Specification for SentChanOutPulse

Name	`SentChanOutPulse`		
Description	This parameter determines the pulse polarity of the respective input channel.		
Multiplicity	1	**Type**	EcucEnumerationParamDef
Range	SENT_ACTIVE_LOW: Pulse polarity is active low SENT_ACTIVE_HIGH: Pulse polarity is active high		
Default value	SENT_ACTIVE_LOW		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Post Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		

1.3.1.4.19 SentChanGlitchFilterDepth

Table 43 Specification for SentChanGlitchFilterDepth

Name	`SentChanGlitchFilterDepth`		
Description	This parameter determines the number of input samples that should be taken into account for the digital glitch filter.		
Multiplicity	1	**Type**	EcucEnumerationParamDef
Range	0 – 15		
Default value	0		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Post Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		

1.3.1.4.20 SentChanDataView

Table 44 Specification for SentChanDataView

Name	`SentChanDataView`		
Description	This parameter determines the sequence in which the received data nibble shall be presented to the user for the respective channel. For example 0x76540123 means Received nibble 0 goes to bits 12-15 of RDR Received nibble 1 goes to bits 8-11 of RDR Received nibble 2 goes to bits 4-7 of RDR Received nibble 3 goes to bits 0-3 of RDR Received nibble 4 goes to bits 16-19 of RDR Received nibble 5 goes to bits 20-23 of RDR Received nibble 6 goes to bits 24-27 of RDR Received nibble 7 goes to bits 28-31 of RDR		
Multiplicity	1	**Type**	EcucEnumerationParamDef
Range	0x01234567 – 0x76543210		
Default value	0x76543210		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Post Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		

1.3.1.4.21 SentChanFreqDriftCheckLen

Table 45 Specification for SentChanFreqDriftCheckLen

Name	`SentChanFreqDriftCheckLen`		
Description	This parameter provides Frequency Drift Check based on Frame Length (FDFL). This is used for frames with pause pulse only. If set the drift error is not checked by HW. Pause Pulse expected and no HW check of drift error must always be set (=1) together with FDFL. *Note: If FDFL is set, RCR.CFC is ignored and the checks described there are not executed.*		
Multiplicity	1	**Type**	EcucEnumerationParamDef
Range	True: Enable Frequency Drift Check based on Frame Length False: Disable Frequency Drift Check based on Frame Length		
Default value	FALSE		

(table continues...)

Table 45 (continued) Specification for SentChanFreqDriftCheckLen

Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Post Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		

1.3.1.4.22 SentChanCalloutFn

Table 46 Specification for SentChanCalloutFn

Name	`SentChanCalloutFn`		
Description	This parameter provides callout function to be invoked for events notification or error handling of the respective channel.		
Multiplicity	1	**Type**	EcucEnumerationParamDef
Range	Values: SENT Channel Callout notification pointer of type FUNCTION_NAME(Selectable)/ Address (Loadable) If IFX SENT is used, it should be configured as SentChanCalloutFn		
Default value	NULL_PTR		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Post Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		

1.3.2 Functions - Type definitions

This section describes all the type definitions used by APIs.

1.3.2.1 Sent_ChannelIdxType

Table 47 Specification for Sent_ChannelIdxType

Syntax	`Sent_ChannelIdxType`
Type	uint8
File	Sent_Types.h
Range	0 – (n-1) where n is the maximum number of SENT channels available in a particular device.
Description	Type definition to indicate the SENT channel number.
Source	IFX

1.3.2.2 Sent_NotifType

Table 48 Specification for Sent_NotifType

Syntax	`Sent_NotifType`
Type	uint32
File	Sent_Types.h
Range	Refer SENT Event Classification for more details.
Description	Type definition to indicate the interrupt events for a SENT channel.
Source	IFX

1.3.2.3 Sent_NotifFnPtrType

Table 49 Specification for Sent_NotifFnPtrType

Syntax	`Sent_NotifFnPtrType`
Type	typedef void (*Sent_NotifFnPtrType) (Sent_ChannelIdxType ChannelId, Sent_NotifType Stat)
File	Sent_Types.h
Range	User configurable function name
Description	Type definition for a callout function pointer; for a SENT channel. It provides two parameters of type Sent_ChannelIdxType – Logical channel number Sent_NotifType – interrupt status/error notification events
Source	IFX

1.3.2.4 Sent_ChannelCfgType

Table 50 Specification for Sent_ChannelCfgType

Syntax	`Sent_ChannelCfgType`	
Type	Structure	
File	Sent_Types.h	
Range	uint32 ChanRxCtrl	RCR value for the respective channel
	uint32 ChanIOCtrl	IOCR value for the respective channel
	uint32 ChanDataView	Receive Data View register (VIEWx) value for the respective channel
	Sent_NotifFnPtrType CallbackFn	Function pointer for user callback notification.
	uint16 ChanPreDiv	CPDR value for the respective channel
	uint16 ChanFracDiv	CFDR value for the respective channel
	uint8 ChanId	SENT physical channel identifier
	uint8 ChanFrameLen	Number of data nibbles per frame

(table continues...)

Table 50 **(continued) Specification for Sent_ChannelCfgType**

	uint8 Sent_InterruptNode	Interrupt node to channel ID
Description	Data structure containing the pointer to SENT channels configuration parameters required for initialization.	
Source	IFX	

1.3.2.5 Sent_CoreConfigType

Table 51 **Specification for Sent_CoreConfigType**

Syntax	Sent_CoreConfigType	
Type	Structure	
File	Sent_Types.h	
Range	const Sent_ChannelCfgType *ChannelConfigPtr	Pointer to the base array of SENT channel configuration
	Sent_ChannelIdxType MaxChannels	Max number of channels
Description	Data structure containing the pointer to SENT module configuration parameters required for initialization. Pointer to object of this type is passed to API Sent_Init to initialize the SENT driver.	
Source	IFX	

1.3.2.6 Sent_RxSerialDataType

Table 52 **Specification for Sent_RxSerialDataType**

Syntax	`Sent_RxSerialDataType`	
Type	Structure	
File	Sent_Types.h	
Range	uint16 Data	12/16 bit serial data.
	uint8 MsgId	4/8 bit message Id.
	uint8 CRC	6-bit CRC
	uint8 Configuration	0 – 12 bit data and 8 bit MsgId 1 – 16 bit data and 4 bit MsgId
Description	Data structure containing information of serial data (slow channel).	
Source	IFX	

1.3.2.7 Sent_ChanOpType

Table 53 **Specification for Sent_ChanOpType**

Syntax	`Sent_ChanOpType`
Type	Enumeration

(table continues...)

Table 53 (continued) Specification for Sent_ChanOpType

File	Sent_Types.h
Range	SENT_ENABLE – Channel shall be enabled SENT_DISABLE – Channel shall be disabled
Description	Enumeration data structure to enable/disable a channel.
Source	IFX

1.3.2.8 Sent_ChanStateType

Table 54 Specification for Sent_ChanStateType

Syntax	`Sent_ChanStateType`
Type	Enumeration
File	Sent_Types.h
Range	SENT_STOP – Channel is disabled SENT_INITIALIZED – Channel enabled, waiting for sync/calibration pulse SENT_RUNNING – one or more sync pulse received, but frequency/Drift not in range SENT_SYNCHRONIZED – Frequency/Drift in range
Description	Enumeration data structure indicating the channel's state.
Source	IFX

1.3.2.9 Sent_ChanStatusType

Table 55 Specification for Sent_ChanStatusType

Syntax	`Sent_ChanStatusType`	
Type	Structure	
File	Sent_Types.h	
Range	uint32 RxTimeStamp	Time the last frame for the respective channel was received. The time is captured during the falling edge of status/communication pulse.
	Sent_ChanStateType ChanStat	Status of the SENT channel
	uint32 IntStat	Snapshot of the INTSTAT register for that channel
	uint8 RxCrc	Last received frame's CRC
	uint8 StatCommNibble	Status and communication nibble value
Description	Data structure containing status information for a channel.	
Source	IFX	

1.3.2.10 Sent_SpcTrigSrcType

Table 56 Specification for Sent_SpcTrigSrcType

Syntax	Sent_SpcTrigSrcType	
Type	Enumeration	
File	Sent.h	
Range	PULSE_STOP	No pulse is generated
	PULSE_START_IMMED	Pulse generated immediately
	PULSE_START_SYNC	Pulse starts each time on the next falling edge after the sync/calibration pulse is received (Used for SPC Bi-Directional mode)
	PULSE_START_EXT_TRIGGER	Pulse starts after each external trigger event
Description	Enumeration data structure which pertains to the trigger type used for SPC transmission.	
Source	IFX	

1.3.2.11 Sent_SpcMode

Table 57 Specification for Sent_SpcMode

Syntax	Sent_SpcMode	
Type	Enumeration	
File	Sent.h	
Range	SYNC_MODE	This indicates SPC synchronous or range selection or id selection mode
	BIDIRECTIONAL_MODE	This indicates SPC Bi-directional mode
	RANGE_SELECTION	This indicates SPC range selection mode
	ID_SELECTION	This indicates SPC ID selection mode
Description	Enumeration data structure which pertains to the SPC transmission mode.	
Source	IFX	

1.3.2.12 Sent_SpcType

Table 58 Specification for Sent_SpcType

Syntax	Sent_SpcType	
Type	Structure	
File	Sent.h	
Range	Sent_SpcTimeBaseType TimeBase	Time base used for SPC transmission.
	Sent_SpcTrigSrcType TriggerSource	Trigger type used for SPC transmission.
	Sent_SpcMode Mode	SPC mode of operation

(table continues...)

Table 58 (continued) Specification for Sent_SpcType

	uint8 Delay	This describes the delay time in ticks after which the SPC pulse will be sent.
	uint8 PulseLength	Length of the pulse in tick times.
Description	Data structure containing information required for a specific SPC transmission. It supports all the SPC modes.	
Source	IFX	

1.3.2.13 Sent_ChannelMapType

Table 59 Specification for Sent_ChannelMapType

Syntax	Sent_ChannelMapType	
Type	Structure	
File	Sent_Ttyps.h	
Range	uint8 Sent_ChannelCore	ID of core to which channel is mapped
	Sent_ChannelIdxType Sent_ChannelIndex	Channel index in core channel configuration
Description	Data structure containing core number and channel index of channel mapping.	
Source	IFX	

1.3.2.14 Sent_ConfigType

Table 60 Specification for Sent_ConfigType

Syntax	`Sent_ConfigType`	
Type	Structure	
File	Sent.ht	
Range	const Sent_CoreConfigType *SentCorePtr [MCAL_NO_OF_CORES]	Pointer to the base array of SENT channel core configuration
	const Sent_ChannelMapType* Sent_LogicalChanId	Pointer to Logical ID mapping
	const Sent_ChannelIdxType* Sent_IntrMapping	Pointer to Interrupt node to Channel Id
	const Sent_ChannelIdxType* Sent_PhysicalChanId	Pointer to Physical Id mapping
	uint32 ModuleClkDiv	SENT module clock divider
	uint16 ModuleFracDivStep	SENT module fractional divider clock
	uint8 NumChannelsConfigured	Number of SENT channels configured

(table continues...)

Table 60 (continued) Specification for Sent_ConfigType

Description	Data structure containing the pointer to SENT module configuration parameters required for initialization. Pointer to object of this type is passed to API Sent_Init to initialize the SENT driver.
Source	IFX

1.3.2.15 Sent_SpcTimeBaseType

Table 61 Specification for Sent_SpcTimeBaseType

Syntax	`Sent_SpcTimeBaseType`	
Type	Enumeration	
File	Sent.h	
Range	PULSE_NOMINAL_FREQ	Pulse based on the configured nominal frequency
	PULSE_LAST_SYNC_FREQ	Pulse based on last measured sync/calibration pulse frequency
Description	Enumeration data structure which pertains to the time base used for SPC transmission.	
Source	IFX	

1.3.2.16 Sent_GlitchStatusType

Table 62 Specification for Sent_GlitchStatusType

Syntax	`Sent_GlitchStatusType`	
Type	Structure	
File	Sent_Types.h	
Range	uint8 RisingEdge	Snapshot of the Rising Edge Glitch Flag Status
	uint8 FallingEdge	Snapshot of the Falling Edge Glitch Flag Status
Description	Data structure containing glitch filter status information for a channel.	
Source	IFX	

1.3.3 Functions - APIs

This section lists all the APIs of the SENT driver.

1.3.3.1 Sent_Init

Table 63 Specification for Sent_Init API

Syntax	`void Sent_Init (const Sent_CfgType *ConfigPtr)`
Service ID	0x00
Sync/Async	Synchronous

(table continues...)

Table 63 (continued) Specification for Sent_Init API

Safety Level	Refer to the release notes for the safety related info	
Reentrancy	Non Reentrant	
Parameters (in)	ConfigPtr	Pointer to SENT configuration structure
Parameters (out)	None	
Parameters (in-out)	None	
Return	void	
Description	Initializes the SENT module and the respective channels based on the configuration values passed in the pointer ConfigPtr.	
Source	IFX	
Error handling	SENT_E_INIT_FAILED, SENT_MASTER_CORE_UNINIT, SENT_E_ALREADY_INITIALIZED, SENT_E_CORE_NOT_CONFIGURED	
Configuration dependencies	-	

1.3.3.2 Sent_SetChannel

Table 64 Specification for Sent_SetChannel API

Syntax	`void Sent_SetChannel (const Sent_ChannelIdxType ChanId, Sent_ChanOpType Operation)`	
Service ID	0x01	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Reentrancy	Non Reentrant	
Parameters (in)	ChanId	SENT logical channel number
	Operation	Operation type SENT_ENABLE – Enable channel SENT_DISABLE – Disable channel
Parameters (out)	None	
Parameters (in-out)	None	
Return	void	
Description	Enable/Disable the SENT channel.	
Source	IFX	
Error handling	SENT_E_UNINIT, SENT_E_INVALID_CHANNEL, SENT_E_CORE_CHANNEL_MISMATCH, SENT_E_CHANNEL_NOT_CONFIGURED	
Configuration dependencies	-	

1.3.3.3 Sent_ReadData

Table 65 Specification for Sent_ReadData API

Syntax	`uint32 Sent_ReadData (const Sent_ChannelIdxType ChannelId)`
Service ID	0x02
Sync/Async	Synchronous
Safety Level	Refer to the release notes for the safety related info
Reentrancy	Non Reentrant
Parameters (in)	ChannelId
Parameters (out)	None
Parameters (in-out)	None
Return	uint32
Description	Reads the current SENT frame received.
Source	IFX
Error handling	SENT_E_UNINIT, SENT_E_INVALID_CHANNEL, SENT_E_CHANNEL_NOT_CONFIGURED, SENT_E_CHANNEL_NOT_ENABLED, SENT_E_CORE_CHANNEL_MISMATCH
Configuration dependencies	-

1.3.3.4 Sent_ReadSerialData

Table 66 Specification for Sent_ReadSerialData API

Syntax	`void Sent_ReadSerialData (const Sent_ChannelIdxType ChannelId, Sent_RxSerialDataType * DataPtr)`	
Service ID	0x03	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Reentrancy	Non Reentrant	
Parameters (in)	ChannelId	SENT channel number
Parameters (out)	DataPtr	Data pointer pointing to the serial data read from the SENT Channel
Parameters (in-out)	None	
Return	void	
Description	Reads the SENT slow channel frame received.	
Source	IFX	

(table continues...)

Table 66 (continued) Specification for Sent_ReadSerialData API

Error handling	SENT_E_PARAM_POINTER, SENT_E_UNINIT, SENT_E_INVALID_CHANNEL, SENT_E_CHANNEL_NOT_CONFIGURED, SENT_E_CHANNEL_NOT_ENABLED, SENT_E_CORE_CHANNEL_MISMATCH
Configuration dependencies	-

1.3.3.5 Sent_ReadChannelStatus

Table 67 Specification for Sent_ReadChannelStatus API

Syntax	`void Sent_ReadChannelStatus (const Sent_ChannelIdxType ChannelId, Sent_ChanStatusType * StatPtr)`	
Service ID	0x04	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Reentrancy	Not Reentrant	
Parameters (in)	ChannelId	SENT logical channel number
Parameters (out)	StatPtr	Pointer pointing to the status of the SENT Channel
Parameters (in-out)	None	
Return	void	
Description	Reads the SENT channel’s current status.	
Source	IFX	
Error handling	SENT_E_PARAM_POINTER, SENT_E_UNINIT, SENT_E_INVALID_CHANNEL, SENT_E_CHANNEL_NOT_CONFIGURED, SENT_E_CORE_CHANNEL_MISMATCH	
Configuration dependencies	-	

1.3.3.6 Sent_SpcGenPulse

Table 68 Specification for Sent_SpcGenPulse API

Syntax	`void Sent_SpcGenPulse (const Sent_ChannelIdxType ChanId, const Sent_SpcType *SpcCfgPtr)`	
Service ID	0x05	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Reentrancy	Non Reentrant	
Parameters (in)	ChanId	SENT Channel's status has to be read
	SpcCfgPtr	Pointer to SPC configuration structure

(table continues...)

Table 68 (continued) Specification for Sent_SpcGenPulse API

Parameters (out)	None
Return value	void
Description	This function generates a Master pulse for SPC Sync transmission and it is also used for the bi-directional mode.
Source	IFX
Error handling	SENT_E_PARAM_POINTER, SENT_E_UNINIT, SENT_E_INVALID_CHANNEL, SENT_E_CHANNEL_NOT_CONFIGURED SENT_E_CHANNEL_NOT_ENABLED, SENT_E_CORE_CHANNEL_MISMATCH
Configuration dependencies	-

1.3.3.7 Sent_SetWdgTimer

Table 69 Specification for Sent_SetWdgTimer API

Syntax	`void Sent_SetWdgTimer (const Sent_ChannelIdxType ChanId, const uint16 WdgTimerReloadVal)`	
Service ID	0x06	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Reentrancy	Non Reentrant	
Parameters (in)	ChanId	SENT Channel's status has to be read
	WdgTimerReloadVal	Timer reload value
Parameters (out)	None	
Parameters (in-out)	None	
Return	void	
Description	This API allows enabling of internal watchdog timer for SENT channel ChanId with timer value WdgTimerReloadVal. To stop the watchdog timer, WdgTimerReloadVal should be set to 0.	
Source	IFX	
Error handling	SENT_E_UNINIT, SENT_E_INVALID_CHANNEL, SENT_E_CHANNEL_NOT_CONFIGURED, SENT_E_CHANNEL_NOT_ENABLED, SENT_E_CORE_CHANNEL_MISMATCH	
Configuration dependencies	-	

1.3.3.8 Sent_GetVersionInfo

Table 70 Specification for Sent_GetVersionInfo API

Syntax	`void Sent_GetVersionInfo (Std_VersionInfoType * VersionInfoPtr)`	
Service ID	0x07	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Reentrancy	Reentrant	
Parameters (in)	Noe	-
Parameters (out)	versioninfo	Pointer to store the version information of this module.
Parameters (in-out)	None	
Return	void	
Description	This API retrieves the vendor-id, module-id along with major and minor version of the SENT driver.	
Source	IFX	
Error handling	SENT_E_PARAM_POINTER	
Configuration dependencies	-	

1.3.3.9 Sent_DeInit

Table 71 Specification for Sent_DeInit API

Syntax	`void Sent_DeInit (void)`
Service ID	0x0A
Sync/Async	Synchronous
Safety Level	Refer to the release notes for the safety related info
Reentrancy	Non Reentrant
Parameters (in)	None
Parameters (out)	None
Parameters (in-out)	None
Return	void
Description	This API provides service to de-initialize the SENT hardware and its channel's registers.
Source	IFX
Error handling	SENT_E_UNINIT, SENT_E_SLAVE_CORE_INIT
Configuration dependencies	-

1.3.3.10 Sent_ReadGlitchFilterStatus

Table 72 Specification for Sent_ReadGlitchFilterStatus API

Syntax	`Std_ReturnType Sent_ReadGlitchFilterStatus (const Sent_ChannelIdxType ChannelId)`	
Service ID	-	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Reentrancy	Non- Reentrant	
Parameters (in)	ChannelId	SENT logical channel number
Parameters (out)	None	
Parameters (in-out)	None	
Return	Std_ReturnType	E_OK: Power Mode changed E_NOT_OK: Service is rejected
Description	This function reads the status of the glitch filter	
Source	IFX	
Error handling	SENT_E_UNINIT, SENT_E_INVALID_CHANNEL, SENT_E_CHANNEL_NOT_CONFIGURED, SENT_E_CHANNEL_NOT_ENABLED, SENT_E_CORE_CHANNEL_MISMATCH	
Configuration dependencies	-	

1.3.3.11 Sent_ResetGlitchFilterStatus

Table 73 Specification for Sent_ ResetGlitchFilterStatus API

Syntax	`Std_ReturnType Sent_ ResetGlitchFilterStatus (const Sent_ChannelIdxType ChannelId)`	
Service ID	-	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Reentrancy	Non- Reentrant	
Parameters (in)	ChannelId	SENT logical channel number
Parameters (out)	None	
Parameters (in-out)	None	
Return	Std_ReturnType	E_OK: Power Mode changed E_NOT_OK: Service is rejected
Description	This function reads the status of the glitch filter.	
Source	IFX	

(table continues...)

Table 73 (continued) Specification for Sent_ ResetGlitchFilterStatus API

Error handling	SENT_E_UNINIT, SENT_E_INVALID_CHANNEL, SENT_E_CHANNEL_NOT_CONFIGURED, SENT_E_CHANNEL_NOT_ENABLED, SENT_E_CORE_CHANNEL_MISMATCH
Configuration dependencies	-

1.3.3.12 Sent_FDFLParameters

Table 74 Specification for Sent_FDFLParameters API

Syntax	`Std_ReturnType Sent_FDFLParameters (const Sent_ChannelIdxType ChannelId)`	
Service ID	-	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Reentrancy	Non- Reentrant	
Parameters (in)	ChannelId	SENT logical channel number
Parameters (out)	FDFLParam	
Parameters (in-out)	None	
Return	Std_ReturnType	E_OK: Power Mode changed E_NOT_OK: Service is rejected
Description	This function checks for the frequency drift in the received channel.	
Source	IFX	
Error handling	SENT_E_UNINIT, SENT_E_INVALID_CHANNEL, SENT_E_CHANNEL_NOT_CONFIGURED, SENT_E_CHANNEL_NOT_ENABLED, SENT_E_CORE_CHANNEL_MISMATCH	
Configuration dependencies	-	

1.3.4 Notifications and callbacks

This section lists all the notifications and callbacks of the SENT driver.

A callout function is linked uniquely with a SENT channel to be notified with the channel's interrupt events or any error/status events. The callout function prototype is defined by Sent_NotifFnPtrType. The callout functions fall under the MCAL layer and are allowed to access the SENT registers if required. The application can determine the necessary action based on the event notifications. It is responsibility of the user to define the SENT callout functions.

1.3.4.1 SENT event classification

The following table provides the events that will be raised by the SENT driver through the callout function per channel.

Event	Event description	Value (Hex)
SENT_INT_RSI_EVENT	Successful reception of SENT frame after verification of CRC	0x1
SENT_INT_RDI_EVENT	Successful reception of SENT frame and data has been moved to the RDR register but CRC may not been verified by HW (depends on SentChanFrameCrcDisable configuration parameter)	0x2
SENT_INT_RBI_EVENT	Receive buffer overflow occurred	0x4
SENT_INT_TDI_EVENT	Successful transmission of SPC master pulse	0x8
SENT_INT_TBI_EVENT	Transmit buffer underflow occurred	0x10
SENT_INT_FRI_EVENT	Synchronization/Calibration pulse deviation occurred from nominal value (more than +/-25%)	0x20
SENT_INT_FDI_EVENT	Subsequent Synchronization/Calibration pulse deviation from its predecessor (more than 1.5625%)	0x40
SENT_INT_NNI_EVENT	More nibbles received than expected or next Synchronization/Calibration pulse indicating less nibbles received	0x80
SENT_INT_NVI_EVENT	Too long or too short nibble pulse received	0x100
SENT_INT_CRCI_EVENT	CRC verification failed for the last received SENT frame	0x200
SENT_INT_WSI_EVENT	This occurs only in standard serial mode; where Status/Communication nibble shows a start bit in a frame other than first SENT frame	0x400
SENT_INT_SDI_EVENT	Successful reception of all serial data bits	0x800
SENT_INT_SCRI_EVENT	CRC verification failed for the serial data received	0x1000
SENT_INT_WDI_EVENT	Watchdog timer for the channel expired; since it didn’t receive the SENT frame within the desired time	0x2000
SENT_TRANS_INPROGRESS_EVENT	Timeout error indicating a transfer is still ongoing	0x4000

1.3.5 Scheduled functions

The SENT driver does not support any scheduled functions.

1.3.6 Interrupt service routines

This section lists all the interrupt handlers of the SENT driver.

1.3.6.1 Sent_Isr

Table 75 Specification for Sent_Isr API

Syntax	void Sent_Isr (uint8 IntrNode)	
Service ID	-	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Reentrancy	Non-Reentrant	
Parameters (in)	IntrNode	Interrupt node for the channel
Parameters (out)	-	
Parameters (in-out)	-	
Return	void	
Description	This function is the interrupt handler and collects the status of the relevant channels and inform the user.	
Source	IFX	
Error handling	None	
Configuration dependencies	-	

1.3.7 Callout

The SENT driver does not provide any callout.

1.3.8 Error Handling

This section describes the various errors reported by the SENT driver.

Error: Description	Source	Error ID (AS422)	Type (AS422)	Error ID (AS440)	Type (AS440)
SENT_E_INVALID_CHANNEL: Synchronous transmission service called at invalid channel.	IFX	0x02	DET	0x02	DET
SENT_E_PARAM_POINTER :API service is called with a NULL pointer as its parameter.	IFX	0x03	DET	0x03	DET
SENT_E_UNINIT: Service is called before Init.	IFX	0x05	DET	0x05	DET
SENT_E_INIT_FAILED : Service is called when initialization is failed.	IFX	0x10	DET	0x10	DET
SENT_E_ALREADY_INITIALIZED: Service is called when Sent driver is already initialized.	IFX	0x14	DET	0x14	DET
SENT_E_CORE_NOT_CONFIGURED: SENT channel is not configured for this Core.	IFX	0x64	DET	0x64	DET

Error: Description	Source	Error ID (AS422)	Type (AS422)	Error ID (AS440)	Type (AS440)
SENT_E_CORE_CHANNEL_MISMATCH:SENT channel is not allocated to this core.	IFX	0x65	DET	0x65	DET
SENT_MASTER_CORE_UNINIT: Core Initialization called when master initialization is not done.	IFX	0x66	DET	0x66	DET
SENT_E_SLAVE_CORE_INIT: Master core de-initialization called before de-initialization of slave cores.	IFX	0x67	DET	0x67	DET
SENT_E_CHANNEL_NOT_CONFIGURED: Sent channel is not configured.	IFX	0x68	DET	0x68	DET
SENT_E_CHANNEL_NOT_ENABLED: Sent channel is not enabled.	IFX	0x69	DET	0x69	DET

1.3.9 Deviations and limitations

This section describes the deviations and limitations of the SENT driver.

1.3.9.1 Deviations

This section describes the deviations of the SENT driver.

1.3.9.1.1 Software specification deviations

The SENT driver does not have any deviations.

1.3.9.1.2 AMDC violations

The SENT driver does not have any AMDC violations.

1.3.9.1.3 VSMD violations

The SENT driver does not have any VSMD violations.

1.3.9.2 Limitations

This section describes the limitations of the SENT driver.

Table 76 Known limitations

Reference	Limitation
Interrupt handling	The SENT channels report 14 interrupt per channel. But number of interrupt nodes available are 10. Hence interrupts of each channel are limited to a single interrupt node only.

Revision history

Date	Version	Description
2023-06-20	4.0	Document is released
2023-05-25	3.1	• Safety Level Tagged value added for all API's as captured in release notes
2021-11-11	3.0	Document is released
2021-11-08	2.1	• Updated the range for SentChanBaudDiv
2020-11-27	2.0	Document is released
2020-11-26	1.1	• Error handling format of all the APIs updated in Functions - APIs section • Error handling section format updated • Updated default value of SentRxInput
2020-08-13	1.0	Document is released.
2020-08-10	0.1	• Initial version • SENT driver chapter moved from TC3xx_SW_MCAL_UM_DEMO to this document • Updated default values of SentDevErrorDetect and SentMultiCoreErrorDetect

Edition 2023-06-20

Published by

Infineon Technologies AG

81726 Munich, Germany

Do you have a question about any aspect of this document?

Email: erratum@infineon.com

Document reference
IFX-eoi1596794428321