Document Title	Requirements on Free Running Timer
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	211
Document Status	obsolete
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	R22-11

Document Change History

Date	Release	Changed by	Change Description
2022-11-24	R22-11	AUTOSAR Release Management	• Document set to obsolete
2021-11-25	R21-11	AUTOSAR Release Management	• No content changes
2020-11-30	R20-11	AUTOSAR Release Management	• No content changes
2019-11-28	R19-11	AUTOSAR Release Management	• No content changes • Changed Document Status from Final to published
2018-10-31	4.4.0	AUTOSAR Release Management	• Editorial changes
2016-12-08	4.3.1	AUTOSAR Release Management	• Editorial changes
2016-11-30	4.3.0	AUTOSAR Release Management	• Editorial changes
2014-10-31	4.2.1	AUTOSAR Release Management	• Editorial changes
2013-10-31	4.1.2	AUTOSAR Release Management	• Editorial changes

Document Change History			
Date	**Release**	**Changed by**	**Change Description**
2013-03-15	4.1.1	AUTOSAR Administration	• Link Requirement with BSW Feature Document • Updating format of requirements according to TPS_StandardizationTemplate
2010-09-30	3.1.5	AUTOSAR Administration	• Legal disclaimer revised
2008-08-13	3.1.1	AUTOSAR Administration	• Legal disclaimer revised
2007-12-21	3.0.1	AUTOSAR Administration	• Document meta information extended • Small layout adaptations made
2007-01-24	2.1.15	AUTOSAR Administration	• “Advice for users” revised • “Revision Information” added
2006-05-16	2.0	AUTOSAR Administration	• Initial release

Disclaimer

Table of Contents

1 Scope of Document

This document defines requirements on the Software Free Running Timer (SWFRT) functionality. The OS SWS Specification shall satisfy these requirements.

Constraints
The hardware of a particular microcontroller might not be able to support free-running timer features – then this functionality SHOULD be LEFT OUT.
This is especially true if

- Hardware timer is not available (or used for different feature, which has incompatible requirements)
- Hardware timer is available, but is not independent. Dependency does not fit.
- Hardware timer does not meet the range/resolution/interval requirements
- Pre Scaler not available or not sufficient
- Hardware timer is available, but use would cause too high interrupt load.
  I.e. it is not much use to emulate a free-running timer by software causing the CPU to have enormous calculation load.

The configurability and its dependencies to other modules is the most crucial part in this module since many times the timer, which is used for the free running timer, shall be shared between modules. The module realizing the SW-FRT shall rather import the settings of any other tools concerning the timer/clock than to define the settings.

Note: This document is obsolete and will be removed in an upcoming release.

2 Conventions to be Used

- The representation of requirements in AUTOSAR documents follows the table specified in [5].

- In requirements, the following specific semantics shall be used (based on the Internet Engineering Task Force IETF).

  The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as:

  - SHALL: This word means that the definition is an absolute requirement of the specification.
  - SHALL NOT: This phrase means that the definition is an absolute prohibition of the specification.
  - MUST: This word means that the definition is an absolute requirement of the specification due to legal issues.
  - MUST NOT: This phrase means that the definition is an absolute prohibition of the specification due to legal constraints.
  - SHOULD: This word, or the adjective "RECOMMENDED", mean that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.
  - SHOULD NOT: This phrase, or the phrase "NOT RECOMMENDED" mean that there may exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.
  - MAY: This word, or the adjective “OPTIONAL“, means that an item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because the vendor feels that it enhances the product while another vendor may omit the same item. An implementation, which does not include a particular option, MUST be prepared to interoperate with another implementation, which does include the option, though perhaps with reduced functionality. In the same vein an implementation, which does include a particular option, MUST be prepared to interoperate with another implementation, which does not include the option (except, of course, for the feature the option provides.)

3 Acronyms and Abbreviations

Abbreviation	*Description*
API	Application Programming Interface
BSW	Basic Software
COM	Communications
ECU	Electronic Control Unit
GPT	General Purpose Timer (SWS Module)
HW	Hardware
Tick	One increment of the HW timer =HW Timer Tick; If not explicitly noted Hardware timer is meant. TickType consists of many HW-Timer Ticks; If this is meant it will be pointed out explicitly.
Interval of Timer	Distance in time between two measure points
OS	AUTOSAR Operating System
Range of Timer	Maximum interval the timer may cover
Reset Timer	Timers which start on exceeding of a predefined margin with an also predefined value.
Resolution of Timer	Minimal time interval which may be measured
SI	International System of Units (abbreviated SI from the French language name Système International d'Unités)
SLA	Software Layered Architecture
SWC	Software Component
SWFRT	Software extending features of HW Free running timers
Test Value	Value against which the present read out is tested (e.g. compared).
Wrap Around	The action taken when a timer reaches the defined maximum value.

Each requirement has its unique identifier starting with a prefix SWFRT.

4 Functional Overview

This chapter describes the requirements on functionality of the module Free Running Timer. Chapter 4.1 introduces SWFRT by an overview, 4.2 and 4.3 contain the requirements. The functionality will be accessed by Low Level SW as well as by application. Therefore the location in the SLA needs to be in the services area (SLA ***ID: 02-06***).

Functionality in scope:

A) The Software Free Running Timer (SWFRT) module provides a piece of code accessing one or more hardware timers. This hardware timer must not be modified by any other SW module during runtime (free running hardware timer or reset timer, SRS_Gpt_12404: configure as continuous mode). The timer may perform functionality with different purposes as well. SWFRT code maps the possibly varying hardware functionality always to the same SW functionality: i.e.

- SWFRT starts with zero as long as no time has passed yet.
- SWFRT increments up to maximum.
  The maximum may differ from the byte/word/… maximum
- The increment exceeding the maximum re-starts the SWFRT with zero (which might be wrap around as one special case)

Functionality A) abstracts the GPT read-out function (SRS_Gpt_12117) or direct hardware access (timer units may be managed directly by OS, see chapter 5 SWS OS).
B) The SWFRT further on should extend a possible restricted range of the HW. Especially when the amount of bits of the HW-timer is restricted the range needs to be extended. For this extension the SWFRT increases a cycle counter. The interval this counter counts is the maximum range of the HW timer.
The HW-timer being used for functionality A) and functionality B) is not necessarily the identical timer; two different HW-timers which are started at different times and range could fulfil this functionality as well. Therefore an offset between the HW-timer of functionality A) and the timer of functionality B) may occur.

Use Cases in Scope:

UC A: SWFRT (Functionality B) **should** enable the implementation of software timers with different resolutions, different ranges and intervals to be measured. Application may use SWFRT to measure times (few ms up to days range)
UC B: -removed; (number B: left intentionally for references)
UC C: SWFRT (Functionality A) **shall** enable “small” defined time delays in the normal program flow. A loop may use the SWFRT to supervise the time interval of a (faulty) hardware when fast reaction time is asked for. “Small” should be understood as a delay which can not be met by OSEK functionality (i.e. a few hundred nano-seconds).
UC D: When the above delays exceed tolerable times (e.g. very long response time of extern HW), an OS reschedule while waiting a bit longer than “small” time-interval might be applied. The timeout will be registered by checking whether the expected event had happened within a defined time.

Restriction on overhead:

Which of the two possible functionalities is applied is up to the imported configuration requirements of SWFRT(minimum and maximum interval to be measured, range and resolution of timer) as well as reasonable resource consumption.
High frequent notification functions shall be avoided. Which is: Do NOT use SRS_Gpt_12120: GPT Notifications to provide long ranges. Instead build long ranges based on OS Tasks calling SWFRT main functions.

A typically used scenario will be following sequence from a user perspective:

1. Read the HW-FRT or counter.
2. Perform some action.
3. Test cyclically the success of this action
4. Read out above FRT again AND
5. If the difference to a subsequent read out of this FRT does not exceed a predefined timeout mark the action as success.

SWFRT SW functionality uses sometimes more than one incremental counter. An increment of the HW-counter by one shall be called “tick”. Further on the microcontroller hardware (HW) could provide incrementing and/or decrementing timers (only). The ticks will represent significantly different values (ns, ms, s). An overflow or an exceeding of the set maximum (/minimum) value re-starts automatically the timer with a zero (/maximum). This action is called wrap around. Within the defined range of the SWFRT timer any calculations of times need to adjust calculations to the wrap around value.
Hardware features shall to be abstracted. Following features shall be considered:

- Microcontroller’s external clock (quartz)
- Microcontroller’s PLL
- Microcontroller’s (fractional) pre-scaler(s) for the used clock
- Microcontrollers register width of the used timer (-combination)
- Reset value after wrap around/wrap around margin
- Microcontrollers access to these registers (!)
- Operation Modes of Microcontroller (Sleep/Stop/Freeze etc.)
- Clock hardware dependencies in between the microcontroller’s timer channels (“Hardware Clock Tree”)
- Absence of
  frequency modulation of system clock (!),
  external not time based clock supply e.g. angle driven clock (!)

These hardware features are to be defined locally in conjunction with the MCU, GPT and OS module as configuration parameters (Their set of parameters may be non portable to a different microcontroller). The set of them leads to the conversion rules of one timer with defined resolution and range (may be not portable to different configuration); the resulting code needs to be generated from scratch for each new configuration. Applying these conversion rules will lead then to functions (macros) reading the free running timer(s) with a defined resolution as well as maximum/minimum interval which could be measured. The “user” is interested in one set which consists of timer, rules, resolution and range.
All above will map into the configuration chapter of the involved modules.

5 Requirements Tracing

Requirement	Description	Satisfied by
RS_BRF_01048	AUTOSAR module design shall support modules to cooperate in a multitasking environment	SRS_Frt_00044
RS_BRF_01056	AUTOSAR BSW modules shall provide standardized interfaces	SRS_Frt_00033, SRS_Frt_00034, SRS_Frt_00047
RS_BRF_01096	AUTOSAR shall support start-up and shutdown of ECUs	SRS_Frt_00020, SRS_Frt_00029, SRS_Frt_00041, SRS_Frt_00048
RS_BRF_01104	AUTOSAR shall support sleep and wake-up of ECUs and buses	SRS_Frt_00048
RS_BRF_01472	AUTOSAR shall support modes	SRS_Frt_00022
RS_BRF_01856	AUTOSAR microcontroller abstraction shall provide access to internal MCU configuration	SRS_Frt_00023, SRS_Frt_00024, SRS_Frt_00025, SRS_Frt_00026
RS_BRF_01904	AUTOSAR microcontroller abstraction shall provide access to hardware timers	SRS_Frt_00019, SRS_Frt_00020, SRS_Frt_00021, SRS_Frt_00022, SRS_Frt_00023, SRS_Frt_00024, SRS_Frt_00025, SRS_Frt_00026, SRS_Frt_00028, SRS_Frt_00029, SRS_Frt_00030, SRS_Frt_00031, SRS_Frt_00032, SRS_Frt_00033, SRS_Frt_00034, SRS_Frt_00041, SRS_Frt_00044, SRS_Frt_00047, SRS_Frt_00048

6 Requirements Specification

Requirements of the same kind within each chapter are grouped under the following headlines:

Functional Requirements:
- Configuration (which elements of the module need to be configurable)
- Initialization
- Normal Operation
- Shutdown Operation
- Fault Operation
- …

Non-Functional Requirements:
- Timing Requirements
- Resource Usage
- Usability
- Output for other WPs (e.g. Description Templates, Tooling,...)
- ...

6.1 Functional Requirements

6.1.1 Configuration

This chapter states the requirements on configurability of the module.

6.1.1.1 [SRS_Frt_00019] HW Timer Type shall be configured

[

Type:	New
Description:	This defines depending on range, resolution and max/min interval to be measured the hw-timer(s) which shall be used for which functionality of the SWFRT module.Pick one type of timer that fulfils the resolution range etc. requirements. This could be either a counter of OS TickTypeor a HW timer of the microcontroller
Rationale:	Restrict the possibilities and the resulting variants / overhead of which timer type may be used for implementation
Use Case:	Define - allowed ranges for Quartz, PLL and if resulting timer provides a constant frequency, - whether timer shall count up or down (no hindering reason for use, but the necessary program will differ), - preferred register width, - if this timer requires a wrap around margin different to register width. - wrap around value, - whether pre-scalers may be used, - which values (range) for which pre-scalers may be set, - if / which timers could be cascaded, - time between wrap around, e.g.Pick the System Timer of Tricore
Dependencies:	--
Supporting Material:	--

](RS_BRF_01904)

6.1.1.2 [SRS_Frt_00020] The configuration and initialization shall be performed by the module providing the SWFRT functionality (OS) if the GPT Timer is not used .

[

Type:	New
Description:	If the GPT Timer is not used the configuration and initialization shall be performed by the module providing the SWFRT functionality (OS).
Rationale:	Use HW most efficiently
Use Case:	There are usually timers such as “System Timer”, “Periodic Interrupt Timer”, “GPT Timer”, etc. which might be used for SWFRT and other modules. Which type is to be used is selected by Requirement 6.1.1.1. They have still features which need to be elaborated and selected per microcontroller – but not per implementation. The setting should not be overridden by each other nor be forgotten
Dependencies:	SRS_Frt_00021
Supporting Material:	--

](RS_BRF_01904, RS_BRF_01096)

6.1.1.3 [SRS_Frt_00021] The elements necessary to calculate the duration of ticks shall be the imported configuration items

[

Type:	New
Description:	The configuration of a new hw-timer is set up if appropriate hw-timer configuration is not available. This is a requirement on the dependencies in Ch 10 of the SWS. This shall ensure whether the set up is done by OS or whether OS will reuse a timer from a different module (e.g. GPT)
Rationale:	Use HW most efficiently
Use Case:	HW Timer is able to provide big range as well as resolution. It may be used for OS TickTypeas well as for timing functions of SW FRT. Just different mask operations need to be applied.
Dependencies:	SRS_Frt_00020
Supporting Material:	--

](RS_BRF_01904)

6.1.1.4 [SRS_Frt_00022] It shall be possible to state which HW Timer is used

[

Type:	Valid
Description:	--
Rationale:	The code will vary significant depending on the used timer
Use Case:,	Define which timers will be supported.
Dependencies:	--
Supporting Material:	--

](RS_BRF_01472,RS_BRF_01904)

6.1.1.5 [SRS_Frt_00023] The Duration of one Tick shall be set up

[

Type:	Valid
Description:	Depending on the access to the timer register this results in different resolutions – this resolution must be known.
Rationale:	The combination of SRS_Frt_00021 and SRS_Frt_00020 define the settings

	for which timer to be used and its rules. These rules are to be defined per microcontroller and HW timer respectively OS GlobalTimeTickType.
Use Case:	The register TIM0 will provide one tick as 12,5 ns for a TC1766 running at a speed of 80MHz. In use case -C- (from Introduction chapter) a loop shall read cyclically the timer value and test a possibly faulty hardware. The maximum test interval is 500ns so the difference in between first and its consecutive readings is predefined (Pre-compile/Link/Post-build) as 40 Either Basic SW module as well as Application is provided in this way with an abstracted time.
Dependencies:	[SRS_Frt_00021], [SRS_Frt_00020]
Supporting Material:	--

](RS_BRF_01904, RS_BRF_01856)

6.1.1.6 [SRS_Frt_00024] The SWFRT shall support different resolutions and ranges

[

Type:	Valid
Description:	The SWFRT shall support different resolutions and ranges. I.e. set up a set of different tick lengths in a way that ranges and resolutions are covered. These are supported with ticks representing different time quantaSee range definition in Table in chapter 2.1. The range shall be assumed to start with 0 up to a maximum.
Rationale:	--
Use Case:	A PIT Register-set will provide one tick as 6.4 µs for a Star12 running at a speed of 40MHz and using a pre-scaler of 256. An access to the 16 bits of the register set register will provide ticks in the range 0 ... 420ms. Since intervals bigger than 420 ms cannot be covered an additional main-function counter shall be implemented for ranges from 0 … 2.6E3 s (1.8 days)
Dependencies:	--
Supporting Material:	--

](RS_BRF_01904, RS_BRF_01856)

6.1.1.7 [SRS_Frt_00025] Access methods to time information shall be provided for different users.

[

Type:	Valid
Description:	Different timers, masks to timers might be needed. If so each access method must be defined.
Rationale:	Avoid multiple conversions between tick –counting and SI unit based comparison; use instead unique approach with predefined test values
Use Case:	There are accesses possible to a basic tick as well as an access to every n^{th} tick. Whereas n is dependent on the microcontroller (e.g. reading bits 8 … 24 of the respective counter only). If the access crosses the bit boundary of 16/32 or exceeds one clock cycle special care has to be put into consistency
Dependencies:	SRS_Frt_00019, SRS_Frt_00020, SRS_Frt_00021, SRS_Frt_00022, SRS_Frt_00023; SRS_Frt_00034
Supporting Material:	--

](RS_BRF_01904, RS_BRF_01856)

6.1.1.8 [SRS_Frt_00026] Set up Target Count Values: Time differences in SI units shall be calculated offline at configuration time.

[

Type:	Valid
Description:	Target Count Values are those against which the read timer value is compared. The Target Count Values shall be configured in SI Units. The equivalent in ticks is stored in the ECU's memory.
Rationale:	Runtime shall be kept low: the margins against which timer differences are tested shall be calculated at configuration time (instead of multiplying at runtime).
Use Case:	The offline calculated target count values may be of the any configuration class. The Target Count Values are those constants which will be compared at runtime against the present value of the timer. This implies that range, resolution and valid timer interval must be respected for the compare instruction. Doing so the code reduces to compare instructions. Values required by user modules are expressed in their XML. The automatic configuration editor for the SWFRT checks other modules for times and, when it finds then, uses knowledge of the timer's range and resolution to calculate the times in counter ticks. These values are then placed back in the user's XML so that the user's code generation has access to those values.
Dependencies:	SRS_Frt_00025
Supporting Material:	--

](RS_BRF_01904, RS_BRF_01856)

6.1.1.9 [SRS_Frt_00028] Continuous Running Mode shall be ensured

[

Type:	Valid
Description:	The used HW timer may perform functionality with different purposes as well. This hardware shall be a free running hardware timer or reset timer, SRS_Gpt_12404: configure as continuous mode.
Rationale:	--
Use Case:	--
Dependencies:	--
Supporting Material:	--

](RS_BRF_01904)

6.1.2 Initialisation

6.1.2.1 [SRS_Frt_00029] An init function independent of whether any registers need to be set or modified shall be available

[

Type:	Valid
Description:	If MCU driver performs the initialization, SWFRT init function must be called after MCU driver init had been called. If GPT driver performs the initialization SWFRT init function must be called after GPT driver had been called.
Rationale:	Ensure timer and PLL is initialized.
Use Case:	--
Dependencies:	--
Supporting Material:	--

](RS_BRF_01904, RS_BRF_01096)

6.1.3 Normal Operation

6.1.3.1 [SRS_Frt_00030] The read - out value shall start with Zero

[

Type:	Valid
Description:	The read - out value starts with Zero; even if HW counts down from maximum to zero
Rationale:	Enable to define a standard interface
Use Case:	e.g. hardware starts with 0xE000 and runs down to 0x100, due to some scaling factors needed, all adaptations to the read out value shall be done within SWFRT
Dependencies:	--
Supporting Material:	--

](RS_BRF_01904)

6.1.3.2 [SRS_Frt_00031] The SWFRT shall increment i.e. Consecutive read out values will increase – unless the defined range of the SWFRT was exceeded

[

Type:	Valid
Description:	This means: invert the counter when the HW timer counts down; this means further on: adjust any offsets which may be present when HW timer counts from an margin down to zero or from an margin up to overflow
Rationale:	Enable to define a standard interface
Use Case:	--
Dependencies:	--
Supporting Material:	--

](RS_BRF_01904)

6.1.3.3 [SRS_Frt_00032] Wrap around shall work without software interaction.

[

Type:	Valid
Description:	--
Rationale:	Save runtime. Don't make time 'walk' i.e. Interrupt consumes time and thus adds time which is not tracked by the timer.
Use Case:	Hardware timer shall be configured to run continuously. There shall be no action necessary to restart the timer. Wrap around shall load the restart value with support of HW:e.g. No additional free running timer is available. A CapCom Timer shall be shared. Its configuration is as follows: CapCom Timer starts at 0xFFFF, reload margin value is 0x3ff, reload value is 0xFFFF, counter is configured as down counter. After counting down to 0x3FF reload 0xFFFF without software interaction.
Dependencies:	--
Supporting Material:	--

](RS_BRF_01904)

6.1.3.4 [SRS_Frt_00033] There shall be a function to achieve an atomic read the of the timer's value.

[

Type:	Valid

Description:	This function reads timer ticks. The conversion of timer ticks to time in SI units (seconds, milliseconds, microseconds, nanoseconds) is not included.
Rationale:	Avoid inconsistent access.
Use Case:	The Timer value must be read consistent (even across byte boundaries or more than one clock cycle). This may involve protected access to 8bit-/16bit-/32bit-/64bit-registers: For example Tricore TC1766 offers a timer width of 56 bit. These 56 bits may be accessed by TIM0 ... TIM6 Registers. Whereas TIM0 reads ticks. TIM1 reads each 16^{th} tick TIM2: 265^{th} , TIM3: 4096^{th} , TIM4: 65536^{th} TIM5: 2^{20th} TIM6: 2^{32th} The registers will provide consistency even over more than 32 bits if registers are read in the HW-defined order
Dependencies:	--
Supporting Material:	--

](RS_BRF_01904,RS_BRF_01056)

6.1.3.5 [SRS_Frt_00047] The SWFRT shall provide a “user” dependent API (function / macro) to convert ticks to time.

[

Type:	Valid
Description:	This function has a number of ticks as a parameter and converts its parameter to time in SI units(seconds, milliseconds, microseconds, nanoseconds).
Rationale:	Allow conversion to SI based time units.
Use Case:	A) Peripheral devices need a start-up time before they may be accessed. This start-up time is specified in the HW description. A timeout [in SI Units] needs to be implemented to avoid reading to non valid data. This timeout needs to be mapped a) to a hw timer which could cope with the interval b) to a value which gives the ticks of this timer B) Diagnostics communication requires variable inter-frame times (STMIN). They need to be set as a measure interval which may be 100µs up to 900µ (9 values) and a second measure interval of 1 ... 127 ms (126 values). These 135 values are to be calculated offline based on the available timers and cyclic main functions.
Dependencies:	--
Supporting Material:	--

](RS_BRF_01904, RS_BRF_01056)

6.1.3.6 [SRS_Frt_00034] The module shall provide functionality to calculate the ticks elapsed between a previously stored value (passed as a parameter) and the current timer value.

[

Type:	Valid
Description:	The caller needs to provide the last read out value.
Rationale:	Support different levels of functionality respectively code size and execution time.
Use Case:	Read the present timer value and use time from function in parameter to calculate the difference
Dependencies:	--
Supporting Material:	--

](RS_BRF_01904, RS_BRF_01056)

6.1.4 Shutdown Operation

6.1.4.1 [SRS_Frt_00041] There shall be no shutdown of SWFRT.

[

Type:	Valid
Description:	--
Rationale:	There is nothing to shut down; not all timers can be stopped.
Use Case:	--
Dependencies:	--
Supporting Material:	--

](RS_BRF_01904, RS_BRF_01096)

6.1.4.2 [SRS_Frt_00048] SW FRT functionality shall be guaranteed after its Init function and is not available in 'SLEEP', 'Wakeup I', 'StartUP I', 'Go OFF II' and 'Power Off' of the ECU.

[

Type:	Valid
Description:	The functionality will return undefined results in the above states of ECU, therefore it shall not be used in these states.
Rationale:	PLL might be not available/ reduced etc
Use Case:	Do NOT use this functionality when there is the risk of unknown timer settings.
Dependencies:	--
Supporting Material:	--

](RS_BRF_01904, RS_BRF_01104,RS_BRF_01096)

6.1.5 Fault Operation

There are no specific requirements.

6.2 Non-Functional Requirements

6.2.1 Timing Requirements

There are no specific timing requirements

6.2.2 Resource Usage

6.2.2.1 [SRS_Frt_00044] The SWFRT shall not block timers for usage.

[

Type:	valid
Description:	Allow more than one module to use the same timer. If the other modules requirements are in similar range the reuse of their configuration shall be enabled.
Rationale:	Enable the sharing of timers
Use Case:	If a PWM works with a frequency which is in the range of the SWFRT requirements this timer shall be offered for use.
Dependencies:	--

Supporting Material:	--

](RS_BRF_01904, RS_BRF_01048)

7 Referenced AUTOSAR documents

[1] Layered Software Architecture
AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf

[2] Glossary
AUTOSAR_TR_Glossary.pdf

[3] Specification of GPT Driver
AUTOSAR_SWS_GPTDriver.pdf

[4] Specification of Operating System
AUTOSAR_SWS_OS.pdf

[5] Software Standardization Template
AUTOSAR_TPS_StandardizationTemplate.pdf