VECTOR

MICROSAR ICU

Technical Reference

MCAL Emulation in VTT

Version 1.1.1

Authors	Christian Leder
Status	Released

Document Information

History

Author	Date	Version	Remarks
Christian Leder	2014-02-18	1.00.00	Initial creation
Christian Leder	2015-02-16	1.01.00	> Global renaming of Vip to Vtt > Usage of template 5.11.0 for the Technical reference
Christian Leder	2017-07-05	1.01.01	Hint for value type Icu_ValueType added

Reference Documents

No.	Source	Title	Version
[1]	AUTOSAR	AUTOSAR_SWS_ICUDriver.pdf	V4.2.0
[2]	AUTOSAR	AUTOSAR_SWS_DevelopmentErrorTracer.pdf	V3.2.0
[3]	AUTOSAR	AUTOSAR_SWS_DiagnosticEventManager.pdf	V4.2.0
[4]	AUTOSAR	AUTOSAR_TR_BSWModuleList.pdf	V1.6.0

Caution
We have configured the programs in accordance with your specifications in the questionnaire. Whereas the programs do support other configurations than the one specified in your questionnaire, Vector´s release of the programs delivered to your company is expressly restricted to the configuration you have specified in the questionnaire.

Contents

Illustrations

Tables

1 Component History

The component history gives an overview over the important milestones that are supported in the different versions of the component.

Component Version	New Features
1.0.x	Initial version of the Vip ICU driver
2.0.x	Global renaming of Vip to Vtt

Table 1-1 Component history

2 Introduction

This document describes the functionality, API and configuration of the AUTOSAR BSW module ICU as specified in [1].

Supported AUTOSAR Release*:	4	
Supported Configuration Variants:	pre-compile	
Vendor ID:	ICU_VENDOR_ID	30 decimal (= Vector-Informatik, according to HIS)
Module ID:	ICU_MODULE_ID	122 decimal (according to ref. [4])

* For the detailed functional specification please also refer to the corresponding AUTOSAR SWS.

The MICROSAR module ICU implements an interface in C programming language for handling the ICU functionality of the emulated microcontroller. This ICU driver offers services for

- Edge detection
- Edge counting
- Edge timestamping, usable for the acquisition of non-periodic signals
- Periodic signal time measurement
- Controlling emulated wake-up interrupts

2.1 Architecture Overview

The following figure shows where the ICU is located in the AUTOSAR architecture.

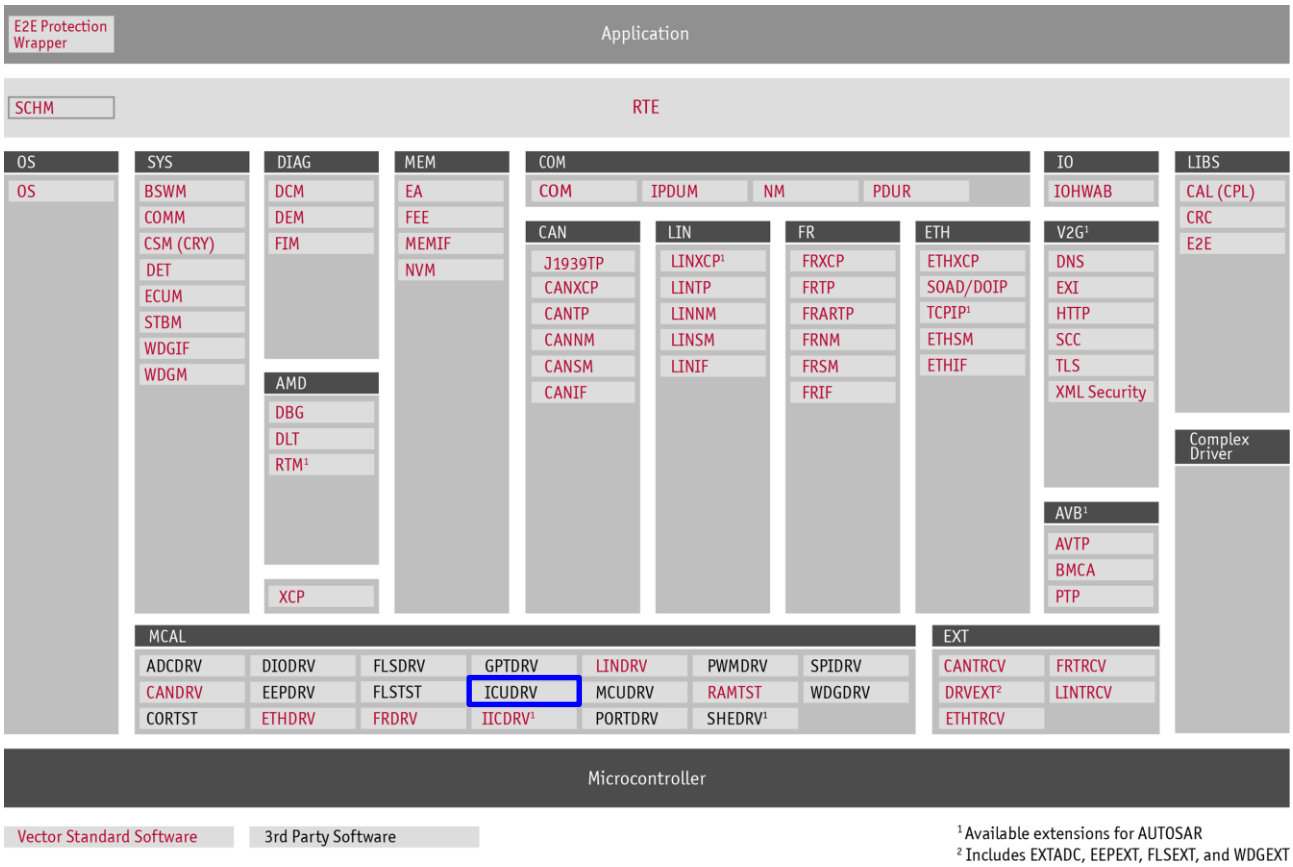

Figure 2-1 AUTOSAR 4.x Architecture Overview

The next figure shows the interfaces to adjacent modules of the ICU. These interfaces are described in chapter 5.

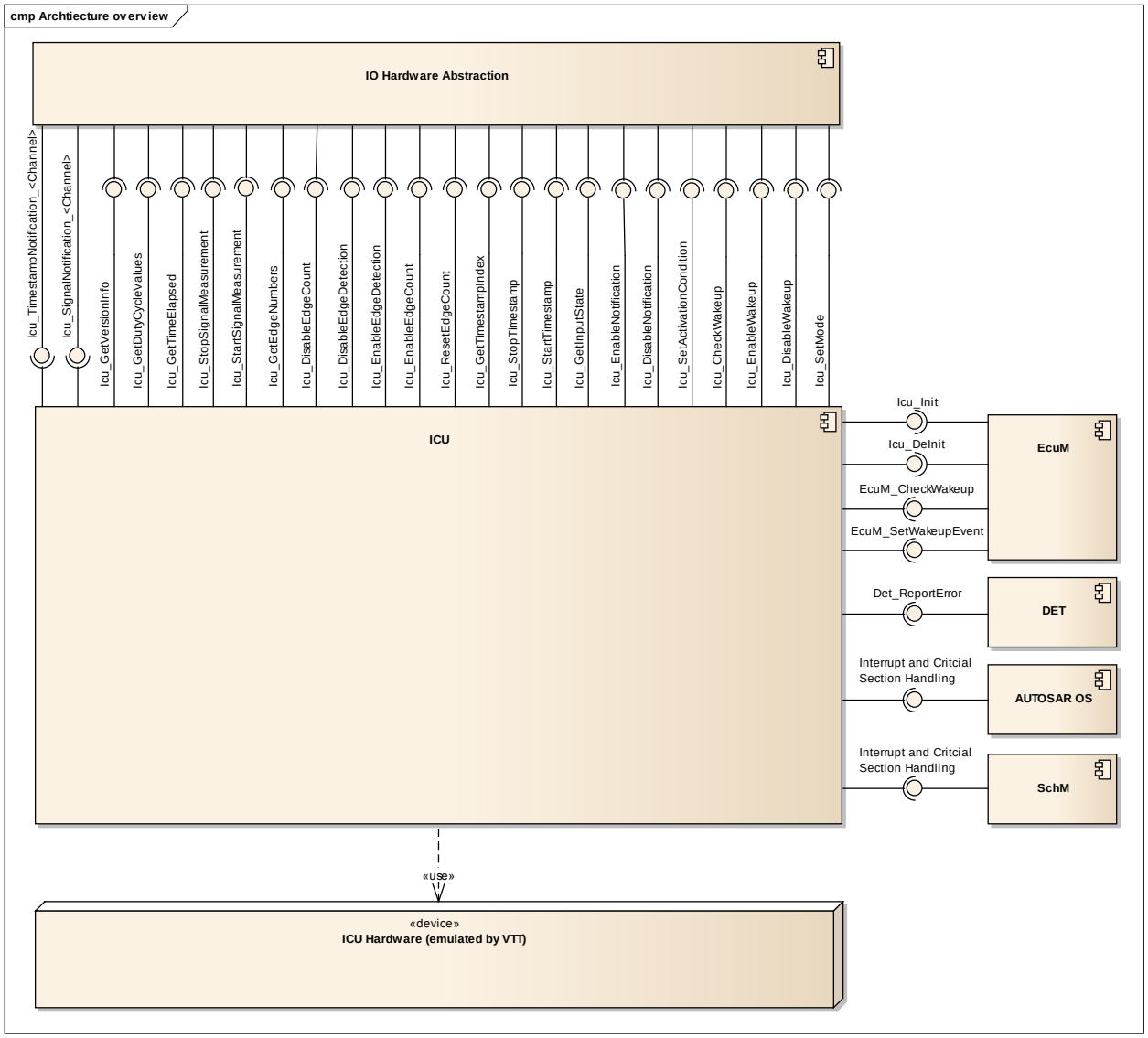

Figure 2-2 Interfaces to adjacent modules of the ICU

3 Functional Description

3.1 Features

The features listed in the following tables cover the complete functionality specified for the ICU.

The AUTOSAR standard functionality is specified in [1], the corresponding features are listed in the tables

> Table 3-1 Supported AUTOSAR standard conform features

> Table 3-2 Not supported AUTOSAR standard conform features

Vector Informatik provides further ICU functionality beyond the AUTOSAR standard. The corresponding features are listed in the table

> Table 3-3 Features provided beyond the AUTOSAR standard

The following features specified in [1] are supported:

Supported AUTOSAR Standard Conform Features
Services to enable and disable edge detection, as well as a service to retrieve the current input state
Services to start and stop edge counting, as well as services for retrieving the number of captured edges and resetting the edge count
Services to start and stop periodic signal time measurement, as well as services for retrieving the captured values and the input state
Services to start and stop edge timestamping, as well as a service to read the current buffer position
Wake-up functionality, as well as services for enabling and disabling wake-up capability of channels
Service for changing the activation edge of configured channels
Services for enabling and disabling notification call-backs

Table 3-1 Supported AUTOSAR standard conform features

3.1.1 Deviations

The following features specified in [1] are not supported:

Not Supported AUTOSAR Standard Conform Features
In contradiction to the AUTOSAR standard, the service `Icu_SetMode` cannot be executed, if there are running operations on channels configured for the modes > Edge counting > Timestamping > Signal Measurement Running channels configured for these modes must explicitly be stopped before the call of `Icu_SetMode`. If development error detection is enabled, the error code `ICU_E_BUSY_OPERATION` will be reported to the DET if there are channels running.

Table 3-2 Not supported AUTOSAR standard conform features

3.1.2 Additions/ Extensions

The following features are provided beyond the AUTOSAR standard:

Features Provided Beyond The AUTOSAR Standard
None

Table 3-3 Features provided beyond the AUTOSAR standard

3.1.3 Limitations

3.1.3.1 Diagnostic Event Manager

Due to the fact that the ICU is emulated, reporting of hardware errors to the DEM is not supported. Because of compatibility reasons, the DEM has to be configured in DaVinci Configurator.

3.2 Emulation

This driver is an emulation of an ICU module.

Caution

Be careful using while loops in order to poll any status.

The user has to ensure, that the application does not block the emulation. So, within every while loop the following function call has to be called:

```
while(ANY_STATUS == temp_status)
{
  Schedule();
}
```

Use the function call Schedule() which is available once the header file of the module ICU is included.

3.3 Initialization

The ICU module is being initialized by calling `Icu_Init(&IcuConfigSet)`. All global variables are initialized by calling `Icu_InitMemory()`. So, `Icu_InitMemory()` has to be called prior to `Icu_Init()`.

3.4 States

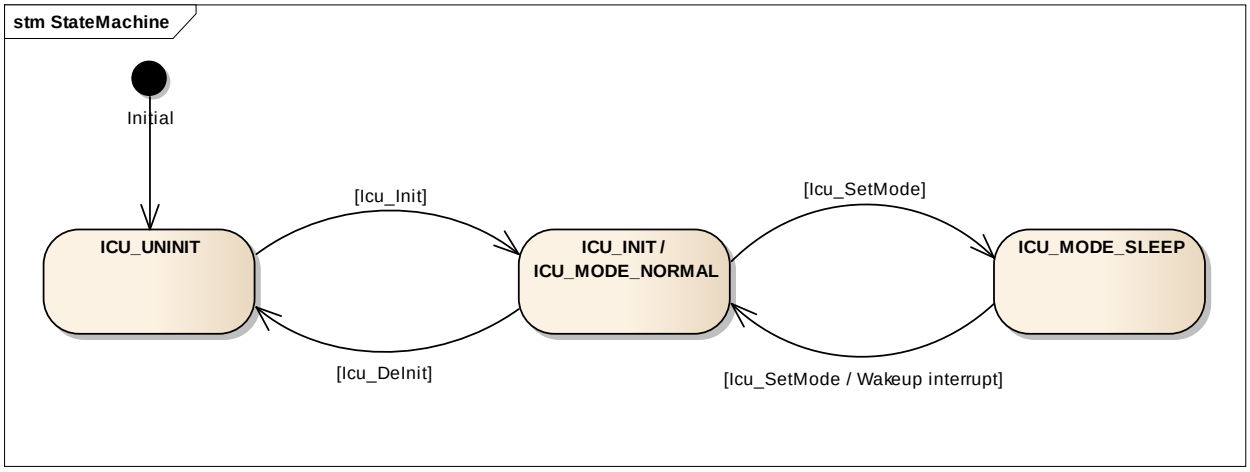

Figure 3-1 Module States

3.5 Main Functions

The ICU module does not provide any cyclic main functions.

3.6 Error Handling

3.6.1 Development Error Reporting

By default, development errors are reported to the DET using the service `Det_ReportError()` as specified in [2], if development error reporting is enabled (i.e. pre-compile parameter `ICU_DEV_ERROR_DETECT==STD_ON`).

If another module is used for development error reporting, the function prototype for reporting the error can be configured by the integrator, but must have the same signature as the service `Det_ReportError()`.

The reported ICU ID is 122.

The reported service IDs identify the services which are described in 5.3. The following table presents the service IDs and the related services:

Service ID	Service
0x00	Icu_Init
0x01	Icu_DeInit
0x02	Icu_SetMode
0x03	Icu_DisableWakeup
0x04	Icu_EnableWakeup
0x05	Icu_SetActivationCondition
0x06	Icu_DisableNotification
0x07	Icu_EnableNotification
0x08	Icu_GetInputState
0x09	Icu_StartTimestamp
0x0A	Icu_StopTimestamp

Service ID	Service
0x0B	Icu_GetTimestampIndex
0x0C	Icu_ResetEdgeCount
0x0D	Icu_EnableEdgeCount
0x0E	Icu_DisableEdgeCount
0x0F	Icu_GetEdgeNumbers
0x10	Icu_GetTimeElapsed
0x11	Icu_GetDutyCycleValues
0x12	Icu_GetVersionInfo
0x13	Icu_StartSignalMeasurement
0x14	Icu_StopSignalMeasurement
0x15	Icu_CheckWakeup
0x16	Icu_EnableEdgeDetection
0x17	Icu_DisableEdgeDetection

Table 3-4 Service IDs

The errors reported to DET are described in the following table:

Error Code		Description
0x0A	ICU_E_PARAM_CONFIG	Service Icu_Init called with wrong parameter
0x0B	ICU_E_PARAM_CHANNEL	API service called with invalid channel identifier
0x0C	ICU_E_PARAM_ACTIVATION	API service called with invalid activation
0x0D	ICU_E_PARAM_BUFFER_PTR	API service called with NULL_PTR as buffer pointer
0x0E	ICU_E_PARAM_BUFFER_SIZE	API service called with invalid size as parameter
0x0F	ICU_E_PARAM_MODE	API service called with invalid parameter ‘mode’
0x14	ICU_E_UNINIT	API service called while the driver is not initialized
0x15	ICU_E_NOT_STARTED	Service Icu_StopTimestamp called for a channel that is not running
0x16	ICU_E_BUSY_OPERATION	Service Icu_SetMode called during a running operation
0x17	ICU_ALREADY_INITIALIZED	Driver already initialized
0x18	ICU_PARAM_NOTIFY_INTERVAL	Notification interval is not within the expected range
0x19	ICU_E_PARAM_VINFO	Service Icu_GetVersionInfo is called with NULL_PTR as parameter

Table 3-5 Errors reported to DET

3.6.1.1 Parameter Checking

AUTOSAR requires that API functions check the validity of their parameters. The checks in Table 3-6 are internal parameter checks of the API functions. These checks are for development error reporting and can be en-/disabled.

The following table shows which parameter checks are performed on which services:

Service \ Check	ICU_E_PARAM_CONFIG	ICU_E_PARAM_CHANNEL	ICU_E_PARAM_ACTIVATION	ICU_E_PARAM_BUFFER_PTR	ICU_E_PARAM_BUFFER_SIZE	ICU_E_PARAM_MODE	ICU_E_UNINIT	ICU_E_NOT_STARTED	ICU_E_BUSY_OPERATION	ICU_E_ALREADY_INITIALIZED	ICU_E_PARAM_NOTIFY_INTERVAL	ICU_E_PARAM_VINFO
Icu_Init	■									■		
Icu_DeInit							■					
Icu_SetMode						■	■		■			
Icu_DisableWakeup		■					■					
Icu_EnableWakeup		■					■					
Icu_SetActivationCondition		■	■				■					
Icu_DisableNotification		■					■					
Icu_EnableNotification		■					■					
Icu_GetInputState		■					■					
Icu_StartTimestamp		■		■	■		■				■	
Icu_StopTimestamp		■					■	■				
Icu_GetTimestampIndex		■					■					
Icu_ResetEdgeCount		■					■					
Icu_EnableEdgeCount		■					■					
Icu_DisableEdgeCount		■					■					
Icu_GetEdgeNumbers		■					■					
Icu_StartSignalMeasurement		■					■					
Icu_StopSignalMeasurement		■					■					
Icu_EnableEdgeDetection		■					■					
Icu_DisableEdgeDetection		■					■					
Icu_GetTimeElapsed		■					■					
Icu_GetDutyCycleValues		■		■			■					
Icu_GetVersionInfo												■
Icu_CheckWakeup							■					

Table 3-6 Development Error Reporting: Assignment of checks to services

3.6.2 Production Code Error Reporting

Info
Production errors are not supported in this emulation.

4 Integration

This chapter gives necessary information for the integration of the MICROSAR ICU into an application environment of an ECU.

4.1 Scope of Delivery

The delivery of the ICU contains the files which are described in the chapters 4.1.1 and 4.1.2:

4.1.1 Static Files

File Name	Description
Icu.h	The module header defines the interface of the ICU. This file must be included by upper layer software components
Icu.c	This C-source contains the implementation of the module's functionalities
Icu_Irq.h	The Icu_Irq header defines the interfaces for the emulated hardware
Icu_Irq.c	This C-Source contains the implementation of the interfaces for the emulated hardware
DrvIcu_VttCanoe01Asr.jar	This jar-file contains the generator and the validator for the DaVinci Configurator
VTTIcu_bswmd.arxml	Basic Software Module Description according to AUTOSAR for VTT Emulation
Icu_bswmd.arxml	Optional Basic Software Module Description. Placeholder for real target (semiconductor manufacturer) in VTT only use case

Table 4-1 Static files

4.1.2 Dynamic Files

The dynamic files are generated by the configuration tool DaVinci Configurator.

File Name	Description
Icu_Cfg.h	The configuration-header contains the static configuration part of this module
Icu_PBcfg.c	The configuration-source contains the object independent part of the runtime configuration

Table 4-2 Generated files

4.2 Include Structure

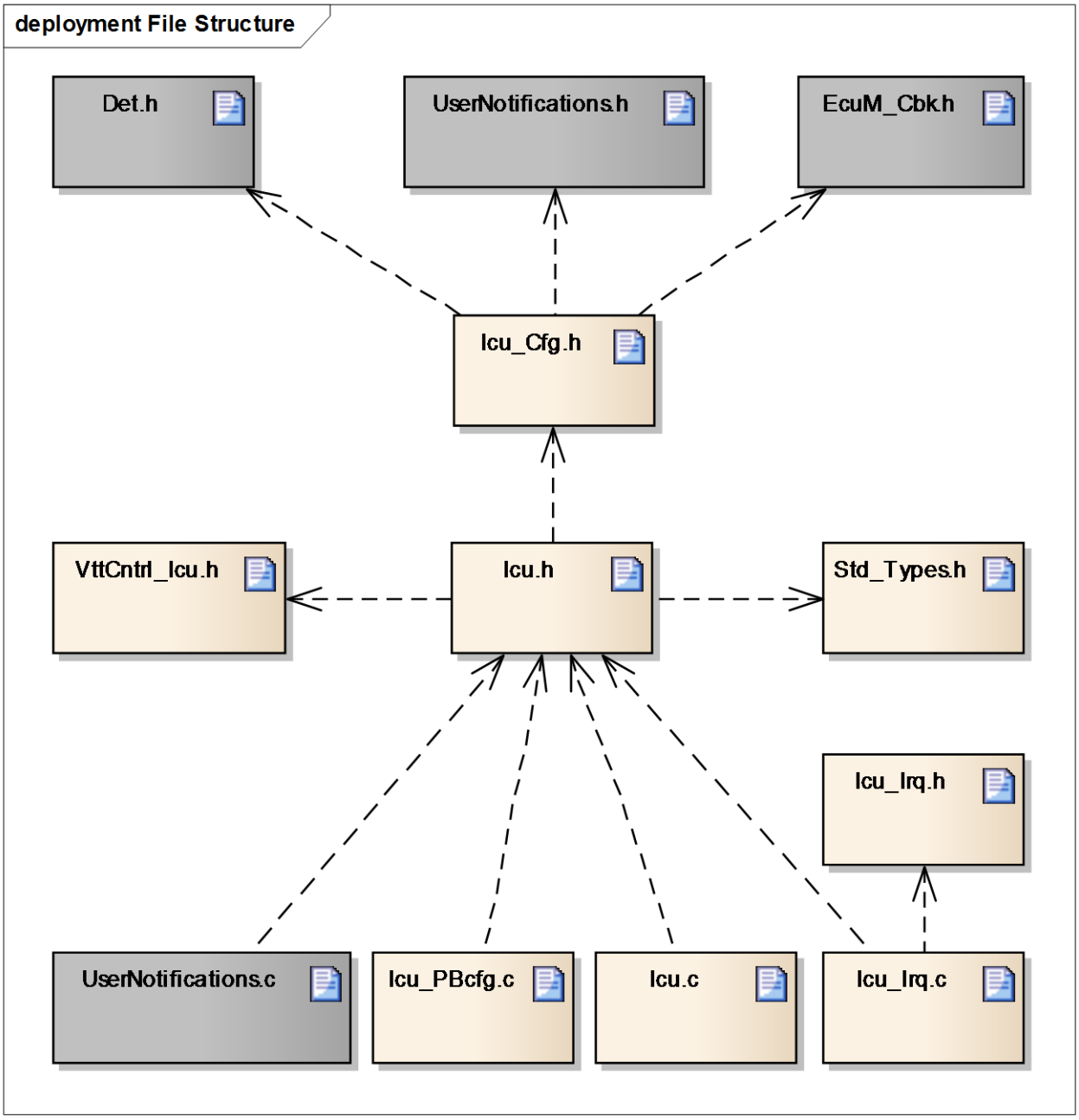

Figure 4-1 Include Structure

4.3 Dependencies on SW Modules

4.3.1 AUTOSAR OS (Optional)

An operating system can be used for task scheduling, interrupt handling, global suspend and restore of interrupts and creating of the Interrupt Vector Table.

4.3.2 DET (Optional)

The ICU module depends on the DET (by default) in order to report development errors. Detection and reporting of development errors can be enabled or disabled by the switch "Enable Development Error Detection".

4.3.3 SchM (Optional)

Beside the AUTOSAR OS the Schedule Manager provides functions that module ICU calls at begin and end of critical sections.

4.3.4 EcuM (Optional)

The module EcuM delivers functionalities to use low-power modes offered by the hardware (e.g. wakeup functionalities). Also the initialization is done by the EcuM module.

5 API Description

For an interfaces overview please see Figure 2-2.

5.1 Type Definitions

The types defined by the ICU are described in this chapter.

Type Name	C-Type	Description	Value Range
Icu_ModeType	enum	Allow enabling/disabling of all interrupts, which are not required for the ECU wakeup	`ICU_MODE_NORMAL` Normal operation: All used interrupts are enabled according to the notification requests
			`ICU_MODE_SLEEP` Reduced power operation: In sleep mode, only those notifications are available, which are configured as wake-up capable.
Icu_ChannelType	uint8	Identifier for an ICU channel	The identifier is used to track the channel. The measurement mode is coded into the highest two bits. The lower bits contain the channel id, which represents the emulated Hardware Unit. The id has a valid range of 0 to 31. Several Channels can use the same id. But only one channel of those can be active at the same time
Icu_InterruptSourceType	uint8	Identifier for an ICU interrupt	0 … 31 Represents the interrupt source (similar to `Icu_ChannelType`)
Icu_InputStateType	enum	Input state of an ICU channel	`ICU_ACTIVE` An activation edge has been detected
			`ICU_IDLE` No activation edge has been detected since the last call of `Icu_GetInputState()` or `Icu_Init()`.
Icu_ActivationType	enum	Type for the activation event of an ICU channel	`ICU_RISING EDGE` An appropriate action shall be executed when a rising edge occurs on an ICU input channel

Type Name	C-Type	Description	Value Range
			`ICU_FALLING_EDGE` An appropriate action shall be executed when a falling edge occurs on an ICU input channel
			`ICU_BOTH_EDGES` An appropriate action shall be executed when either a rising or a falling edge occurs on an ICU input channel
Icu_ValueType	uint32	Width of the buffer for timestamp ticks and measured elapsed time ticks **Note** Unit of timestamp ticks is the CANoe tick time which is defined in Nanoseconds (ns).	0x00 … 0xFFFFFFFF
Icu_DutyCycleType	c-struct	Type contains the values for calculating the duty cycle	`PeriodTime` This shall be the coherent period-time measured on a channel
			`ActiveTime` This shall be the coherent active-time measured on a channel
Icu_IndexType	uint8	Type to abstract the return value of the service `Icu_GetTimestampIndex()`	0x00 … 0xFF
Icu_EdgeNumberType	uint32	Type to abstract the return value of the service `Icu_GetEdgeNumbers()`	0x00 … 0xFFFFFFFF
Icu_MeasurementModeType	enum	Type for the measurement mode of an ICU channel	`ICU_MODE_SIGNAL_EDGE_DETECT` Mode for detecting edges
			`ICU_MODE_SIGNAL_MEASUREMENT` Mode for measuring different times between various configurable edges
			`ICU_MODE_TIMESTAMP` Mode for capturing timer values on configurable edges
			`ICU_MODE_EDGE_COUNTER` Mode for counting edges on configurable edges

Type Name	C-Type	Description	Value Range
Icu_SignalMeasurmentPropertyType	enum	Type for the measurement property of a signal measurement channel	ICU_LOW_TIME The channel is configured for reading the elapsed signal low time
			ICU_HIGH_TIME The channel is configured for reading the elapsed signal high time
			ICU_PERIOD_TIME The channel is configured for reading the elapsed signal period time
			ICU_DUTY_CYCLE The channel is configured to read values, which are needed for calculating the duty cycle(coherent active and period time)
			ICU_ACTIVE_TIME The channel is configured for reading the elapsed Signal Active Time
Icu_TimestampBufferType	enum	Type for the format of the timestamping buffer	ICU_LINEAR_BUFFER The buffer will just be filled once
			ICU_CIRCULAR_BUFFER After reaching the end of the buffer, the buffer pointer starts at the beginning of the buffer

Table 5-1 Type definitions

5.2 Interrupt Service Routines provided by ICU

5.2.1 IcuIsr_<0…31>

Prototype	
`void` **`IcuIsr_<0…31>`** `(void)`	
Parameter	
-	-
Return code	
-	-
Functional Description	
Interrupt Service Routine for each Hardware Unit. The ISRs are called from the VTT Kernel if an ongoing hardware unit has detected a configured edge. Within the function, a handler is called which do the state transitions, calculates the signal times and calls the notifications.	

Particularities and Limitations
> This function is synchronous.
> This function is non-reentrant.

Table 5-2 IcuIsr<0…31>

5.3 Services provided by ICU

5.3.1 Icu_InitMemory

Prototype	
`void` **`Icu_InitMemory`** `(void)`	
Parameter	
-	-
Return code	
-	-
Functional Description	
This service initializes the global variables in case the startup code does not work	
Particularities and Limitations	
> This function is synchronous.	
> This function is non reentrant.	
> Module must not be initialized.	
Expected Caller Context	
> Called during startup	

Table 5-3 Icu_InitMemory

5.3.2 Icu_Init

Prototype	
`void` **`Icu_Init`** `(P2CONST(Icu_ConfigType, AUTOMATIC, ICU_APPL_CONST) ConfigPtr)`	
Parameter	
`ConfigPtr`	Pointer to the configuration struct of the ICU
Return code	
-	-
Functional Description	
The service initializes the module with the values of the structure referenced by the parameter ‘ConfigPtr’. Furthermore, notification and wakeup capability for all channels are disabled and the state of all configured channels is reset to `ICU_IDLE`.	
Particularities and Limitations	
> This service is synchronous	
> This service is non reentrant	
> Module must not be initialized.	

Expected Caller Context
> ECU State Manager or comparable software module, responsible for driver initialization after startup.

Table 5-4 Icu_Init

5.3.3 Icu_DeInit

Prototype	
`void Icu_DeInit ( void )`	
Parameter	
-	-
Return code	
-	-
Functional Description	
This service de-initializes the ICU driver.	
Particularities and Limitations	
> This service is synchronous > This service is non reentrant > This service is always available > This service shall not be called during a running operation	
Expected Caller Context	
> Task context	

Table 5-5 Icu_DeInit

5.3.4 Icu_SetMode

Prototype	
`void Icu_SetMode ( Icu_ModeType Mode )`	
Parameter	
`Mode`	`ICU_MODE_NORMAL`, normal operation, `ICU_MODE_SLEEP`, reduced power mode - in sleep mode only those notifications are available which are configured as wakeup capable.
Return code	
-	-

Functional Description
Function for ICU mode selection. This function sets the operation mode to the given mode parameter. In `ICU_MODE_NORMAL` mode all notifications are available as > configured by function Icu_SetActivationCondition() or Icu_DefaultStartEdge. > selected by the Icu_DisableNotification() and Icu_EnableNotification() functions before or after the call of Icu_SetMode(). In `ICU_MODE_SLEEP` mode > only those wakeup events are available which are configured as wakeup capable, enabled via Icu_EnableWakeup() after Icu_Init() and which are not disabled via function Icu_DisableWakeup(). All channels are stopped except those channels > which have been configured as wakeup capable and > which were explicitly enabled by the call of Icu_EnableWakeup()
Particularities and Limitations
> This service is synchronous > This service is non-reentrant > This service is configurable > This service shall not be called during a running operation
Expected Caller Context
> Task context

Table 5-6 Icu_SetMode

5.3.5 Icu_DisableWakeup

Prototype	
`void` **`Icu_DisableWakeup`** `( Icu_ChannelType Channel )`	
Parameter	
`Channel`	Numeric identifier or symbolic name of an ICU channel
Return code	
-	-
Functional Description	
This function disables the wakeup capability of a single ICU channel and is only feasible for ICU channels configured statically as wakeup capable.	
Particularities and Limitations	
> This service is synchronous > This service is reentrant for different ICU channels > This service is configurable	
Expected Caller Context	
> Task context	

Table 5-7 Icu_DisableWakeup

5.3.6 Icu_EnableWakeup

Prototype	
`void` **`Icu_EnableWakeup`** `( Icu_ChannelType Channel )`	
Parameter	
`Channel`	Numeric identifier or symbolic name of an ICU channel
Return code	
-	-
Functional Description	
This function enables the wakeup capability of a single ICU channel and is only feasible for ICU channels configured statically as wakeup capable.	
Particularities and Limitations	
> This service is synchronous > This service is reentrant for different ICU channels > This service is configurable	
Expected Caller Context	
> Task context	

Table 5-8 Icu_EnableWakeup

5.3.7 Icu_CheckWakeup

Prototype	
`void` **`Icu_CheckWakeup`** `( EcuM_wakeupSourceType WakeupSource )`	
Parameter	
`WakeupSource`	Reported wakeup source ID
Return code	
-	-
Functional Description	
This function enables the wakeup capability of a single ICU channel and is only feasible for ICU channels configured statically as wakeup capable.	
Particularities and Limitations	
> This service is synchronous > This service is reentrant for different ICU channels > This service is configurable	
Expected Caller Context	
> Task context	

Table 5-9 Icu_CheckWakeup

5.3.8 Icu_SetActivationCondition

Prototype	
`void` **`Icu_SetActivationCondition`** `( Icu_ChannelType Channel, Icu_ActivationType Activation )`	
Parameter	
`Channel`	Numeric identifier or symbolic name of an ICU channel
`Activation`	Type of Activation: > ICU_RISING_EDGE > ICU_FALLING_EDGE > ICU_BOTH_EDGES
Return code	
-	-
Functional Description	
This function sets the activation-edge for the given channel according to the parameter `Activation`. This function supports channels which are configured for one of the following measurement modes: > `ICU_MODE_SIGNAL_EDGE_DETECT` > `ICU_MODE_TIMESTAMP` > `ICU_MODE_EDGE_COUNTER`	
Particularities and Limitations	
> This service is synchronous > This service is reentrant for different ICU channels	
Expected Caller Context	
> Task context	

Table 5-10 Icu_SetActivationCondition

5.3.9 Icu_DisableNotification

Prototype	
`void` **`Icu_DisableNotification`** `( Icu_ChannelType Channel )`	
Parameter	
`Channel`	Numeric identifier or symbolic name of an ICU channel
Return code	
-	-
Functional Description	
This function disables the ICU signal notification of the given channel. This function supports channels, which are configured for one of the following measurement modes: > `ICU_MODE_SIGNAL_EDGE_DETECT` > `ICU_MODE_TIMESTAMP`	

Particularities and Limitations
> This service is synchronous
> This service is reentrant for different ICU channels
Expected Caller Context
> Task context

Table 5-11 Icu_DisableNotification

5.3.10 Icu_EnableNotification

Prototype	
`void` **`Icu_EnableNotification`** `( Icu_ChannelType Channel )`	
Parameter	
`Channel`	Numeric identifier or symbolic name of an ICU channel
Return code	
-	-
Functional Description	
This function enables the ICU signal notification of the given channel. This function supports channels, which are configured for one of the following measurement modes: > `ICU_MODE_SIGNAL_EDGE_DETECT` > `ICU_MODE_TIMESTAMP`	
Particularities and Limitations	
> This service is synchronous > This service is reentrant for different ICU channels	
Expected Caller Context	
> Task context	

Table 5-12 Icu_EnableNotification

5.3.11 Icu_GetInputState

Prototype	
`Icu_InputStateType` **`Icu_GetInputState`** `( Icu_ChannelType Channel )`	
Parameter	
`Channel`	Numeric identifier or symbolic name of an ICU channel
Return code	
`Icu_InputStateType`	`ICU_ACTIVE`: An activation edge has been detected `ICU_IDLE`: No activation edge has been detected since the last call of `Icu_GetInputState()` or `Icu_Init()`

Functional Description
This function returns the status of the ICU input. Only channels which are configured for the following measurement modes are supported: > ICU_MODE_SIGNAL_MEASUREMENT > ICU_MODE_SIGNAL_EDGE_DETECT If an activation edge has been detected this function returns ICU_ACTIVE. For Signal Measurement a channel should be set to ICU_ACTIVE not until this measurement has completed and the driver is able to provide useful information on the input signal. Once the function has returned the status ICU_ACTIVE, the stored status will be set to ICU_IDLE until the next edge is detected. If no activation edge has been detected this function returns ICU_IDLE.
Particularities and Limitations
> This service is synchronous > This service is reentrant for different ICU channels > This service is configurable
Expected Caller Context
> Task context

Table 5-13 Icu_GetInputState

5.3.12 Icu_StartTimestamp

Prototype	
`void Icu_StartTimestamp ( Icu_ChannelType Channel, Icu_ValueType *BufferPtr, Icu_IndexType BufferSize, Icu_IndexType NotifyInterval )`	
Parameter	
Channel	Numeric identifier or symbolic name of an ICU channel
BufferPtr	Pointer to the buffer-array in which the timestamp values shall be stored **Note** Unit of timestamp ticks is the CANoe tick time which is defined in Nanoseconds (ns).
BufferSize	Size of the external buffer (number of fields of the array)
NotifyInterval	Notification interval (number of events)
Return code	
-	-

Functional Description
This service starts the capturing of timer values on the edges > activated by the service Icu_SetActivationCondition() (rising / falling / both edges) > to an external buffer > at the beginning of the buffer If a notification function is configured and `NotifyInterval` is greater than 0, the notification function is being called when the number of events specified by `NotifyInterval` has been captured. If circular buffer handling is configured (for the given channel), the buffer pointer will be reset to the beginning of the buffer after the buffer end has been reached. If linear buffer handling is configured and the capture functionality reaches the end of the buffer, the driver stops capturing timer values.
Particularities and Limitations
> This service is asynchronous > This service is reentrant for different ICU channels > This service is configurable
Expected Caller Context
> Task context

Table 5-14 Icu_StartTimestamp

5.3.13 Icu_StopTimestamp

Prototype	
`void` **`Icu_StopTimestamp`** `( Icu_ChannelType Channel )`	
Parameter	
`Channel`	Numeric identifier or symbolic name of an ICU channel
Return code	
-	-
Functional Description	
This service stops the capturing of timer values on the given channel.	
Particularities and Limitations	
> This service is synchronous > This service is reentrant for different ICU channels > This service is configurable	
Expected Caller Context	
> Task context	

Table 5-15 Icu_StopTimestamp

5.3.14 Icu_EnableEdgeDetection

Prototype
`void` **`Icu_EnableEdgeDetection`** `( Icu_ChannelType Channel)`

Parameter	
`Channel`	Numeric identifier or symbolic name of an ICU channel
Return code	
-	-
Functional Description	
This service starts the detection of configured signal edges.	
Particularities and Limitations	
> This service is synchronous > This service is reentrant for different ICU channels > This service is configurable	
Expected Caller Context	
> Task context	

Table 5-16 Icu_StartTimestamp

5.3.15 Icu_DisableEdgeDetection

Prototype	
`void` **`Icu_DisableEdgeDetection`** `( Icu_ChannelType Channel )`	
Parameter	
`Channel`	Numeric identifier or symbolic name of an ICU channel
Return code	
-	-
Functional Description	
This service stops the detection of edges on the given channel.	
Particularities and Limitations	
> This service is synchronous > This service is reentrant for different ICU channels > This service is configurable	
Expected Caller Context	
> Task context	

Table 5-17 Icu_StopTimestamp

5.3.16 Icu_GetTimestampIndex

Prototype	
`IcuIndexType` **`Icu_GetTimestampIndex`** `( Icu_ChannelType Channel )`	
Parameter	
`Channel`	Numeric identifier or symbolic name of an ICU channel
Return code	
`Icu_IndexType`	Current timestamp index
Functional Description	
This service reads the timestamp index of the given channel, which is the next to be written. The service returns 0 in case the service is called before `Icu_StartTimestamp()` (no buffer is defined in this case).	
Particularities and Limitations	
> This service is synchronous > This service is reentrant for different ICU channels > This service is configurable	
Expected Caller Context	
> Task context	

Table 5-18 Icu_GetTimestampIndex

5.3.17 Icu_ResetEdgeCount

Prototype	
`void` **`Icu_RestEdgeCount`** `( Icu_ChannelType Channel )`	
Parameter	
`Channel`	Numeric identifier or symbolic name of an ICU channel
Return code	
-	-
Functional Description	
The value of the counted edges will be reset to zero with the call of this function.	
Particularities and Limitations	
> This service is synchronous and reentrant for different ICU channels > This service is configurable	
Expected Caller Context	
> Task context	

Table 5-19 Icu_ResetEdgeCount

5.3.18 Icu_EnableEdgeCount

Prototype
`void` **`Icu_EnableEdgeCount`** `( Icu_ChannelType Channel )`

Parameter	
Channel	Numeric identifier or symbolic name of an ICU channel
Return code	
-	-
Functional Description	
This function enables the counting of edges on the given channel. Only the configured edges are counted (rising edge/falling edge/both edges).	
Particularities and Limitations	
> This service is synchronous > This service is reentrant for different ICU channels > This service is configurable	
Expected Caller Context	
> Task context	

Table 5-20 Icu_EnableEdgeCount

5.3.19 Icu_DisableEdgeCount

Prototype	
`void Icu_DisableEdgeCount ( Icu_ChannelType Channel )`	
Parameter	
Channel	Numeric identifier or symbolic name of an ICU channel
Return code	
-	-
Functional Description	
This function disables the counting of edges on the given channel.	
Particularities and Limitations	
> This service is synchronous > This service is reentrant for different ICU channels > This service is configurable	
Expected Caller Context	
> Task context	

Table 5-21 Icu_DisableEdgeCount

5.3.20 Icu_GetEdgeNumbers

Prototype	
`Icu_EdgeNumberType` **`Icu_GetEdgeNumbers`** `( Icu_ChannelType Channel )`	
Parameter	
`Channel`	Numeric identifier or symbolic name of an ICU channel
Return code	
`Icu_EdgeNumberType`	Number of counted edges
Functional Description	
This function reads the number of counted edges after the last call of `Icu_ResetEdgeCount()`.	
Particularities and Limitations	
> This service is synchronous > This service is reentrant for different ICU channels > This service is configurable	
Expected Caller Context	
> Task context	

Table 5-22 Icu_GetEdgeNumbers

5.3.21 Icu_GetTimeElapsed

Prototype	
`Icu_ValueType` **`Icu_GetTimeElapsed`** `( Icu_ChannelType Channel )`	
Parameter	
`Channel`	Numeric identifier or symbolic name of an ICU channel
Return code	
`Icu_ValueType`	Captured time in ticks **Note** Unit of timestamp ticks is the CANoe tick time which is defined in Nanoseconds (ns).
Functional Description	
> This function reads the elapsed Signal Low Time for the given channel that is configured in Measurement Mode “Signal Measurement, Signal Low Time”. The elapsed time is measured between a falling edge and the consecutive rising edge of the channel. > This function reads the elapsed Signal High Time for the given channel that is configured in Measurement Mode “Signal Measurement, Signal High Time”. The elapsed time is measured between a rising edge and the consecutive falling edge of the channel. > This function reads the elapsed Signal Period Time for the given channel that is configured in Measurement Mode “Signal Measurement, Signal Period Time”. The elapsed time is measured between consecutive rising (or falling) edges of the channel. The period start edge is configurable. This function returns “0” in case > no requested time has been captured > the capturing of a requested time is in progress and not finished > a captured time was already returned once by this function and this function is called again	

Particularities and Limitations
> This service is synchronous
> This service is reentrant for different ICU channels
> This service is configurable
Expected Caller Context
> Task context

Table 5-23 Icu_GetTimeElapsed

5.3.22 Icu_StartSignalMeasurement

Prototype	
`void` **`Icu_StartSignalMeasurement`** `( Icu_ChannelType Channel )`	
Parameter	
`Channel`	Numeric identifier or symbolic name of an ICU channel
Return code	
-	-
Functional Description	
This function starts the measurement of signals on the given channel, beginning with the configured default start edge, which occurs first after the call of this function.	
Particularities and Limitations	
> This service is asynchronous	
> This service is reentrant for different ICU channels	
> This service is configurable	
Expected Caller Context	
> Task context	

Table 5-24 Icu_StartSignalMeasurement

5.3.23 Icu_StopSignalMeasurement

Prototype	
`void` **`Icu_StopSignalMeasurement`** `( Icu_ChannelType Channel )`	
Parameter	
`Channel`	Numeric identifier or symbolic name of an ICU channel
Return code	
-	-
Functional Description	
This function stops the measurement of signals on the given channel.	

Particularities and Limitations
> This service is synchronous
> This service is reentrant for different ICU channels
> This service is configurable
Expected Caller Context
> Task context

Table 5-25 Icu_StopSignalMeasurement

5.3.24 Icu_GetDutyCycleValues

Prototype	
`void Icu_GetDutyCycleValues ( Icu_ChannelType Channel, Icu_DutyCycleType *DutyCycleValues )`	
Parameter	
`Channel`	Numeric identifier or symbolic name of an ICU channel
`DutyCycleValues`	Captured duty cycle values in ticks
Return code	
-	-
Functional Description	
This function reads the coherent active time and period time for the given ICU Channel, if it is configured in Measurement Mode “Signal Measurement, Duty Cycle Values”. The definition on which edge the period starts is configurable per channel. This function returns “0” in case > no coherent active- and period time has been captured > the capturing of a requested high- and period time is in progress and not finished > captured duty cycle values were already returned once by this function and this function is called again	
Particularities and Limitations	
> This service is synchronous > This service is reentrant for different ICU channels > This service is configurable	
Expected Caller Context	
> Task context	

Table 5-26 Icu_GetDutyCycleValues

5.3.25 Icu_GetVersionInfo

Prototype	
`void Icu_GetVersionInfo ( P2VAR(Std_VersionInfoType, AUTOMATIC, ICU_APPL_DATA) versioninfo )`	
Parameter	
`versioninfo`	Pointer to where to store the version information
Return code	
-	-
Functional Description	
This function returns the version information of the module. The version information includes: > Module Id > Vendor Id > Software version numbers	
Particularities and Limitations	
> This service is synchronous. > This service is reentrant. > This service is configurable	
Expected Caller Context	
> Task context	

Table 5-27 Icu_GetVersionInfo

5.4 Services used by ICU

In the following table services provided by other components, which are used by the ICU are listed. For details about prototype and functionality refer to the documentation of the providing component.

Component	API
DET	Det_ReportError

Table 5-28 Services used by the ICU

5.5 Configurable Interfaces

5.5.1 Notifications

At its configurable interfaces the ICU defines notifications that can be mapped to callback functions provided by other modules. The mapping is not statically defined by the ICU but can be performed at configuration time. The function prototypes that can be used for the configuration have to match the appropriate function prototype signatures, which are described in the following sub-chapters.

5.5.1.1 Signal Edge Detection Notification

Prototype	
`void` **`<IcuSignalNotification>`** `(void)`	
Parameter	
-	-
Return code	
-	-
Functional Description	
This edge notification function is called upon the occurrence of a configured event (rising edge, falling edge, or both).	
Particularities and Limitations	
> The service `Icu_EnableNotification()` has to be called previously	
Call context	
> This notification function is called in interrupt context and shall therefore be kept as simple as possible > This function is the notification for channels configured for the mode `ICU_MODE_SIGNAL_EDGE_DETECT`	

Table 5-29 IcuSignalNotification

5.5.1.2 Timestamp Notification

Prototype	
`void` **`<IcuTimestampNotification>`** `(void)`	
Parameter	
-	-
Return code	
-	-
Functional Description	
This notification function is called after the occurrence of n configured events (rising edge, falling edge, or both), whereas n is the number of events configured by the parameter ‘`NotifyInterval`’ in the service `Icu_StartTimestamp()`.	
Particularities and Limitations	
> The service `Icu_EnableNotification()` has to be called previously	
Call context	
> This notification function is called in interrupt context and shall therefore be kept as simple as possible > This function is the notification for channels configured for the mode `ICU_MODE_TIMESTAMP`	

Table 5-30 IcuTimestampNotification

6 Configuration

The ICU supports the configuration variants

> VARIANT-PRE-COMPILE

The configuration classes of the ICU parameters depend on the supported configuration variants. For their definitions please see the VTTIcu_bswmd.arxml file.

6.1 Configuration with DaVinci Configurator 5

The ICU module is configured with the help of the configuration tool DaVinci Configurator 5 (CFG5). The definition of each parameter is given in the corresponding BSWMD file.

7 Glossary and Abbreviations

7.1 Glossary

Term	Description
CANoe	Tool for simulation and testing of networks and electronic control units.
DaVinci Configurator	Configuration and generation tool for MICROSAR components

Table 7-1 Glossary

7.2 Abbreviations

Abbreviation	Description
API	Application Programming Interface
AUTOSAR	Automotive Open System Architecture
BSW	Basis Software
DEM	Diagnostic Event Manager
DET	Development Error Tracer
ECU	Electronic Control Unit
EcuM	ECU State Manager
ICU	Input Capture Unit
IoHwAb	BSW Module I/O Hardware Abstraction (Connection to RTE)
ISR	Interrupt Service Routine
MICROSAR	Microcontroller Open System Architecture (the Vector AUTOSAR solution)
OS	Operating System
RTE	Runtime Environment
SchM	BSW Module Scheduler
VTT	vVIRTUALtarget

Table 7-2 Abbreviations

8 Contact

Visit our website for more information on

- News
- Products
- Demo software
- Support
- Training data
- Addresses

www.vector.com