



MCAL CAN Module Software Design Document

Document Version : 97
Document Owner : Texas Instruments
Document Status : Published
Last Approval Date : Mar 11, 2022

TI Confidential - NDA Restrictions
Copyright ©2022 Texas Instruments Incorporated

-
- [Revision History](#)
 - [Terms and Abbreviations](#)
 - [Introduction](#)
 - [Overview](#)
 - [AUTOSAR Architecture](#)
 - [CAN MCAL](#)
 - [Purpose and Scope](#)
 - [Module Overview](#)
 - [Requirements](#)
 - [Features Supported](#)
 - [Features Not Supported / NON Compliance](#)
 - [Assumptions](#)
 - [Constraints](#)
 - [Hardware and SW platforms](#)
 - [Dependencies](#)
 - [Stakeholders](#)
 - [References](#)
 - [Design Description.](#)
 - [Fundamental Operation.](#)
 - [Classic Can \(Normal Operation\)](#)
 - [CAN FD](#)
 - [Interrupt Service Routine](#)
 - [Directory Structure](#)
 - [Configurator](#)
 - [NON Standard configurable parameters](#)







- Implementation specific parameters (computed)
 - Variant Support
- Error Classification
 - Development Errors
 - Error Detection
 - Error notification (DET)
 - Runtime Errors
 - Error notification (DEM)
- Implementation Details
 - Data structures and resources
 - Maximum number of controller and mailboxes.
 - Can_ControllerType
 - Can Controller Pre-Compile Configuration(Can_ControllerType_PC)
 - Mailbox Configuration(Can_MailboxType)
 - Can Mailbox Pre-Compile Configuration(Can_MailboxType_PC)
 - Can Configuration(Can_ConfigType)
 - Dynamic Behavior - Control Flow Diagram
 - States
 - Dynamic Behavior - Data Flow Diagram
 - Application Parameters
 - Can_Write
 - Can_DisableControllerInterrupts
 - Can_EnableControllerInterrupts
 - Can_MainFunction_Write
 - Can_MainFunction_BusOff
 - Can_MainFunction_Read

- Can_MainFunction_Wakeup
- Can_GetVersionInfo
- Can_MainFunction_Mode
- Can_TestLoopBackModeEnable
- Can_TestLoopBackModeDisable
- Can_RegisterReadbackType.
- Can_SetBaudrate
- Can_GetControllerMode
- Can_GetControllerErrorState
- Can_DelInit
- Safety Diagnostic Features
- Low Level Definitions
 - Driver API's
 - Can_Init
 - Can_SetControllerMode
 - Can_Write
 - Can_DisableControllerInterrupts
 - Can_EnableControllerInterrupts
 - Can_MainFunction_Write
 - Can_MainFunction_BusOff
 - Can_MainFunction_Read
 - Can_MainFunction_Wakeup
 - Can_GetVersionInfo
 - Can_MainFunction_Mode
 - Can_SetBaudrate
 - Can_GetControllerMode





- Can_SetControllerMode
 - Can_GetControllerErrorState
 - Can_Delnit
 - Can_RegisterReadback
 - Can_EnableIntr
 - Can_DisableIntr
 - Can_GetIntrStatus
 - Can_ClearIntrStatus
- Performance Objectives
 - Resource Consumption Objectives
 - Critical timing and Performance
 - CAN Dma mode
 - MCAN Tx Buffer Mode
- Test Criteria
- Template Revision History

1 Revision History

Version	Date	Author	Document Status	Comments
0.1	 25 Jul 2018	Sujith S	DONE	First Version
0.2	 25 Jul 2018	Sujith S	DONE	Format conversion and review completed
0.3	 25 Jul 2018	Sujith S	DONE	Updated section "Development Errors" to include requirement mapping
0.4	 19 Jan 2020	Sunil M S	DONE	Updates w.r.o porting AUTOSAR 4.3.1 Version
0.5	 07 Oct 2021	Harish S	IN PROGRESS	Added Safety Diagnostic Features and changed design document format as per ASPICE
0.6	 24 Jan 2022	Nikki S	IN PROGRESS	JACINTOREQ-1870



Version	Date	Author	Document Status	Comments
0.7	 28 Feb 2022	Jayapriya J	IN PROGRESS	Added Safety Diagnostic APIs and removed obsolete design Ids
v97	 04 Mar 2022	Nikki S	DONE	Review Comments Addressed

2 Terms and Abbreviations

Abbreviation /Term	Meaning / Explanation
CAN controller	A CAN controller serves exactly one physical channel
L-PDU	Data Link Layer Protocol Data Unit. Consists of Identifier, Data Length and Data (SDU).
L-SDU	Data Link Layer Service Data Unit. Data that is transported inside the L-PDU
DLC	Data Length Code
ISR	Interrupt Service Routine
MCAL	Microcontroller Abstraction Layer
SFR	Special Function Register. Hardware register that controls the controller behavior
SPAL	Standard Peripheral Abstraction Layer

Abbreviation /Term	Meaning / Explanation
Physical Channel	A physical channel represents an interface to the CAN Network. Different physical channels of the CAN Hardware Unit may access different networks
N-PDU	Network Protocol Data Unit of the CAN Transport Layer
N-SDU	Service Data Unit of the CAN Transport Layer. Data that is transported inside the N-PDU.
HTH	CAN hardware transmit handle
Hardware Object	A Hardware Object is defined as message buffer inside the CAN RAM of the CAN Hardware Unit. Also often called Message Object

3 Introduction

3.1 Overview

The figure below depicts the AUTOSAR layered architecture as 3 distinct layers, Application, Runtime Environment (RTE) and Basic Software (BSW). The BSW is further divided into 4 layers, Services, Electronic Control Unit Abstraction, MicroController Abstraction (MCAL) and Complex Drivers.



3.2 AUTOSAR Architecture

MCAL is the lowest abstraction layer of the Basic Software. It contains software modules that interact with the Microcontroller and its internal peripherals directly. Can driver is part of the Microcontroller Drivers (block, show above). Below shows the position of the Can driver in the AUTOSAR Architecture.



3.3 CAN MCAL

Can module offers following services.

- On L-PDU transmission, the Can module writes the L-PDU in an appropriate buffer inside the CAN controller hardware.
- On L-PDU reception, the Can module calls the RX indication callback function with ID, DLC and pointer to L-SDU as parameter.
- The Can module provides an interface that serves as periodical processing function, and which must be called by the Basic Software Scheduler module periodically.
- The Can module provides services to control the state of the CAN controllers. Bus-off and Wake-up events are notified by means of callback functions

The figure below shows Can driver interaction with other modules of AUTOSAR stack.



3.4 Purpose and Scope

The Detailed Design document provides the design details of CAN driver and aims to provide a guide to a design that could be implemented by a software developer.

The scope of this document is to describe the software design procedure of CAN module.

3.4.1 Module Overview

The figure below shows the block diagram of CAN-FD module. MCAN Hardware IP (mcanss_10_rel.1.1.x) is being used.



The Controller Area Network (CAN) is a serial communications protocol which efficiently supports distributed real-time control. CAN has high immunity to electrical interference. In a CAN network, many short messages are broadcast to the entire network, which provides for data consistency in every node of the system.

The CAN module (also known as MCAN) supports both classic CAN and CAN FD (CAN with Flexible Data-Rate) specifications. CAN FD feature allows high throughput and increased payload per data frame. The classic CAN and CAN FD devices can coexist on the same network without any conflict.



SoC can support multiple instances of the CAN module, please refer the device TRM (can be found in SoC User Manual) for accurate count. Each MCAN module supports flexible bit rates greater than 1 Mbps and is compliant to ISO 11898-1:2015.

The main features of the CAN are

- Conforms with CAN Protocol 2.0 A, B and ISO 11898-1
- Full CAN FD support (up to 64 data bytes)
- AUTOSAR and SAE J1939 support
- Up to 32 dedicated Transmit Buffers
- Configurable Transmit FIFO, up to 32 elements
- Configurable Transmit Queue, up to 32 elements
- Configurable Transmit Event FIFO, up to 32 elements
- Up to 64 dedicated Receive Buffers
- Two configurable Receive FIFOs, up to 64 elements each
- Up to 128 standard filter elements supported
- Up to 64 extended filter elements supported
- Internal Loopback mode for self-test
- Timestamp Counter

3.5 Requirements

The CAN driver shall implement as per requirements detailed in [Reference 1 - AUTOSAR 4.3.1](#).

3.6 Features Supported

Below listed are some of the key features that are expected to be supported

- Initialization and de-initialization of all CAN/MCAN controllers on SoC
- Transmission of CAN frames and confirmation
- Reception of CAN frames
- Polling mode for Read and Write confirmation
- Support for CAN error detection and reporting
- Mailbox objects – Full CAN for both Tx and Rx (32 Tx and 64 Rx)
- The requirement id's listed below shall be supported

3.7 Features Not Supported / NON Compliance

- [NON Compliance] Hardware wakeup is not supported
- [NON Compliance] No support for pretended networking
- [NON Compliance] Support for TTCAN
- [AUTOSAR Optional] Support for trigger transmit API
- [AUTOSAR Optional] L-PDU callouts is not supported
- Supports additional/device specific configuration parameters, refer section [Implementation specific parameters in configurator](#).

3.8 Assumptions

Below listed are assumed to valid for this design/implementation, exceptions and other deviations are listed for each explicitly. Care should be taken to ensure these assumptions are addressed.



1. The functional clock to the CAN module is expected to be on before calling any CAN module API. The CAN driver as such doesn't perform any PRCM programming to get the functional clock.
2. The clock-source selection for CAN is not performed by the CAN driver, other entities such as SBL or MCAL module MCU shall perform the same.
3. The pin configuration used for CAN is not performed by the CAN driver, other entities such as SBL or MCAL module PORT shall perform the same.

Note that assumption 1, 2 & 3 are specified by AUTOSAR CAN specification.

3.9 Constraints

Some of the critical constraints of this design are listed below

- HTH's and HRH's shall be grouped into 2 groups, i.e. a group where a HTH/HRH is mapped to only one hardware mailbox (1:1 mapping) and another group where a HTH/HRH is mapped to a group of hardware mailboxes (1:n mapping). Number of hardware mailboxes assigned to a HTH/HRH 'n', can be configured through 'CanHwObjectCount' variable.
Please note that if we are using FIFO for receive, we cannot guarantee exact number of messages received for particular message id since it is going to common FIFO pool. For Eg: If you configure CanID1 as 0xC1, FIFO Size as 3 and CanID2 as 0xD1, FIFO Size as 2, Total Fifo Size allocated will be of 5. If you transmit 0xC1 4 time's and transmit 0xD1 2 time's, All 4 0xC1 will be stored in FIFO and only single 0xD1 will be stored in FIFO. Other 0xD1 message will be lost as there is no place to store it.
- In case RxProcessing Type is selected as MIXED mode, interrupts will be enabled for all the Rx buffers. If any mail box(Rx Buffer) is configured for MIXED and HwObjUsesPolling set to TRUE, then dummy interrupt will be occurred which cannot be avoided due to hardware limitation.
- In cases where MCU module is not employed (supported) to configure the clock source for CAN module ([Assumptions#2](#))

3.10 Hardware and SW platforms

Hardware Platforms




- Refer to specified SoC User Manual to check if ADC module is supported.









Software Platforms








- Bare-Metal

3.11 Dependencies

In addition to dependencies listed in section 5 of [Reference 1 - AUTOSAR 4.3.1](#), CAN driver shall depend on these modules to realize the required functionality, two clocks are provided to the MCAN module: the peripheral synchronous clock (interface clock - MCANx_ICLK) and the peripheral asynchronous clock (functional clock - MCANx_FCLK).

Design Identifier	Description
 MCAL-6079 - SWS_Can_00024 : Configuration Dependencies PUBLISHED	SWS_Can_00024 : Configuration Dependencies
 MCAL-6173 - SWS_Can_00244 : Other Driver Services PUBLISHED	SWS_Can_00244 : Other Driver Services
 MCAL-6108 - SWS_Can_00238 : Other Driver Services PUBLISHED	SWS_Can_00238 : Other Driver Services

Design Identifier	Description
 MCAL-6106 - SWS_Can_00281 : System Services PUBLISHED	SWS_Can_00281 : System Services
 MCAL-6048 - ECUC_Can_00313 : CanCpuClockRef PUBLISHED	ECUC_Can_00313 : CanCpuClockRef
 MCAL-5944 - CANFDCLK PUBLISHED	CANFDCLK
 MCAL-5954 - SWS_Can_00239 : Other Driver Services PUBLISHED	SWS_Can_00239 : Other Driver Services
 MCAL-6042 - CanDefaultOSCounterId PUBLISHED	CanDefaultOSCounterId
 MCAL-5987 - SWS_Can_00110 : Scheduled function PUBLISHED	SWS_Can_00110 : Scheduled function
 MCAL-5981 - ECUC_Can_00383 : CanControllerSyncJumpWidth PUBLISHED	ECUC_Can_00383 : CanControllerSyncJumpWidth
 MCAL-6148 - ECUC_Can_00476 : CanControllerPropSeg PUBLISHED	ECUC_Can_00476 : CanControllerPropSeg

Design Identifier	Description
 MCAL-6177 - ECUC_Can_00073 : CanControllerPropSeg PUBLISHED	ECUC_Can_00073 : CanControllerPropSeg
 MCAL-6067 - ECUC_Can_00075 : CanControllerSeg2 PUBLISHED	ECUC_Can_00075 : CanControllerSeg2
 MCAL-5997 - ECUC_Can_00477 : CanControllerSeg1 PUBLISHED	ECUC_Can_00477 : CanControllerSeg1
 MCAL-5966 - ECUC_Can_00478 : CanControllerSeg2 PUBLISHED	ECUC_Can_00478 : CanControllerSeg2
 MCAL-5930 - ECUC_Can_00074 : CanControllerSeg1 PUBLISHED	ECUC_Can_00074 : CanControllerSeg1
 MCAL-6127 - CanEcc_Enable PUBLISHED	CanEcc_Enable
 MCAL-6120 - ECUC_Can_00479 : CanControllerSyncJumpWidth PUBLISHED	ECUC_Can_00479 : CanControllerSyncJumpWidth

3.12 Stakeholders

- Developers
- Test Engineers
- Customer Integrator

3.13 References

	Specification	Comment / Link
1	AUTOSAR 4.3.1	AUTOSAR Specification for CAN Driver.
2	BSW General Requirements / Coding guidelines	Autosar and Coding guidelines for the Mcal drivers.
3	Software Product Specification (SPS)	Product Functional requirements.
4	Software Architecture	Mcal Software Architecture.

4 Design Description.

4.1 Fundamental Operation.

The MCAN module performs CAN protocol communication as per ISO 11898-1:2015. The bit rate can be programmed to values greater than 1 Mbps. Additional transceiver hardware is required for the connection to the physical layer (CAN bus).

For communication on a CAN network, individual message frames can be configured. The message frames and identifier masks are stored in the Message RAM. All functions concerning the handling of messages are implemented in the Message Handler. The register set of the MCAN module can be accessed directly via the module interface. These registers are used to control and configure the CAN core and the Message Handler, and to access the Message RAM.



Can dependent modules

- **CAN Core:** The CAN core consists of the CAN protocol controller and the Rx/Tx shift register. It handles all ISO 11898-1:2015 protocol functions and supports 11-bit and 29-bit identifiers.
- **Message Handler:** The Message Handler (Rx Handler and Tx Handler) is a state machine that controls the data transfer between the single-ported Message RAM and the CAN core's Rx/Tx shift register. It also handles the acceptance filtering.
- **Message RAM:** The main purpose of the Message RAM is to store Rx/Tx messages, Tx Event elements, and Message ID Filter elements.
- **Message RAM Interface:** Enables connection between the Message RAM and the other blocks in the MCAN module.
- **Registers and Message Object Access:** Data consistency is ensured by indirect accesses to the message objects. During normal operation, all software and DMA accesses to the Message RAM are done through interface registers. The interface registers have the same word-length as the Message RAM.
- **Module Interface:** The MCAN module registers are accessed by the user software through a 32-bit peripheral bus interface.
- **Clocking:** Two clocks are provided to the MCAN module, the peripheral synchronous clock (interface clock - MCANx_ICLK) and the peripheral asynchronous clock (functional clock - MCANx_FCLK). An entity outside CAN driver shall provide/configure such as SBL, MCAL module MCU shall perform the same.

4.2 Classic Can (Normal Operation)

Once the MCAN module is initialized and the INIT bit is reset to zero, the MCAN module synchronizes itself to the CAN bus and is ready for communication. After passing the acceptance filtering, received messages including Message Identifier (ID) and Data Length Code (DLC) are stored into a dedicated Rx Buffer or into Rx FIFO 0/Rx FIFO 1. For messages to be transmitted dedicated Tx Buffers and/or a Tx FIFO or a Tx Queue can be initialized or updated.

4.3 CAN FD

The CAN FD standard allows extended frames to be transmitted, up to 64 data bytes in a single frame and at a higher bit rate for the data phase of a frame, up to 8 Mbps. The CAN FD standard introduces the ability to switch from one bit rate to another. Extended Data Length (EDL), as shown in Figure, sets a data length of up to 8 or up to 64 data bytes. Bit Rate Switching (BRS) indicates whether two bit rates (the data phase is transmitted at a different bit rate to the arbitration phase) are enabled.



Can FD Frame

There are two variants of CAN FD frame transmission

- CAN FD frame transmission without bit rate switching
- CAN FD frame transmission where control field, data field, and CRC field are transmitted with a higher bit rate than the beginning and the end of the frame

Some of the important fields of this packet are detailed below.

In the CAN frames FDF = recessive (logical 1) signifies a CAN FD frame, FDF = dominant (logical 0) signifies a Classic CAN frame. In a CAN FD frame, the two bits following FDF - res and BRS, decide whether the bit rate inside of this CAN FD frame is switched. A CAN FD bit rate switch is signified by res = dominant and BRS = recessive. Note that the coding of res = recessive is reserved for future expansion of the protocol.

In case the MCAN module receives a frame with FDF = recessive and res = recessive, it will signal a Protocol Exception Event. When Protocol Exception Handling is enabled, this causes the operation state to change from Receiver to Integrating at the next sample point. In case Protocol Exception Handling is disabled the MCAN will treat a recessive res bit as an form error and will respond with an error frame.

CAN FD operation is enabled by programming the FDOE bit. In case FDOE = 1, transmission and reception of CAN FD frames is enabled. Transmission and reception of Classic CAN frames is always possible. Whether a CAN FD frame or a Classic CAN frame is transmitted can be configured via the FDF bit in the respective Tx Buffer element.

A mode change during CAN operation is only recommended under the following conditions

- The failure rate in the CAN FD data phase is significant higher than in the CAN FD arbitration phase. In this case disable the CAN FD bit rate switching option for transmissions.
- During system startup all nodes are transmitting Classic CAN messages until it is verified that they are able to communicate in CAN FD format. If this is true, all nodes switch to CAN FD operation.
- Wakeup messages in CAN Partial Networking have to be transmitted in Classic CAN format.
- End-of-line programming in case not all nodes are CAN FD capable. Non CAN FD nodes are held in silent mode until programming has completed. Then all nodes switch back to Classic CAN communication.




 MCAL-6062 - ECUC_Can_00475 : CanControllerTxBitRateSwitch PUBLISHED	ECUC_Can_00475 : CanControllerTxBitRateSwitch
Design ID	Description








4.4 Interrupt Service Routine






Section 7 of [Reference 1 - AUTOSAR 4.3.1](#), details the expected behavior and control flow for ISR implementation, please refer the same.

An separate interrupt handler shall be provided for each instance of the Can controller. e.g. If the SoC has 2 instances of Can controller, two ISR

- Can_0_Int0ISR : ISR for Can controller 1
- Can_1_Int0ISR : ISR for Can controller 2

Design ID	Description
 MCAL-6085 - SWS_Can_00016 : L-PDU transmission PUBLISHED	SWS_Can_00016 : L-PDU transmission
 MCAL-6145 - SWS_Can_00059 : L-PDU transmission PUBLISHED	SWS_Can_00059 : L-PDU transmission
 MCAL-5915 - SWS_Can_00100 : L-PDU transmission PUBLISHED	SWS_Can_00100 : L-PDU transmission

Design ID	Description
 MCAL-6013 - SWS_Can_00396 : L-PDU reception PUBLISHED	SWS_Can_00396 : L-PDU reception
 MCAL-5943 - SWS_Can_00279 : L-PDU reception PUBLISHED	SWS_Can_00279 : L-PDU reception
 MCAL-5921 - SWS_Can_00060 : L-PDU reception PUBLISHED	SWS_Can_00060 : L-PDU reception
 MCAL-5962 - SWS_Can_00033 : Implement interrupt service routines PUBLISHED	SWS_Can_00033 : Implement interrupt service routines
 MCAL-6170 - SWS_Can_00415 : Can_PduType PUBLISHED	SWS_Can_00415 : Can_PduType
 MCAL-6138 - SWS_Can_00237 : Remote transmission requests PUBLISHED	SWS_Can_00237 : Remote transmission requests
 MCAL-6129 - SWS_Can_00276 : L-PDU transmission PUBLISHED	SWS_Can_00276 : L-PDU transmission

Design ID	Description
 MCAL-6081 - SWS_Can_00427 : L-PDU transmission PUBLISHED	SWS_Can_00427 : L-PDU transmission
 MCAL-6066 - ECUC_Can_00480 : CanControllerTrcvDelayCompensationOffset PUBLISHED	ECUC_Can_00480 : CanControllerTrcvDelayCompensationOffset
 MCAL-6044 - ECUC_Can_00318 : CanTxProcessing PUBLISHED	ECUC_Can_00318 : CanTxProcessing
 MCAL-6041 - SWS_CAN_00501 : L-PDU reception PUBLISHED	SWS_CAN_00501 : L-PDU reception
 MCAL-5928 - SWS_Can_00423 : L-PDU reception PUBLISHED	SWS_Can_00423 : L-PDU reception

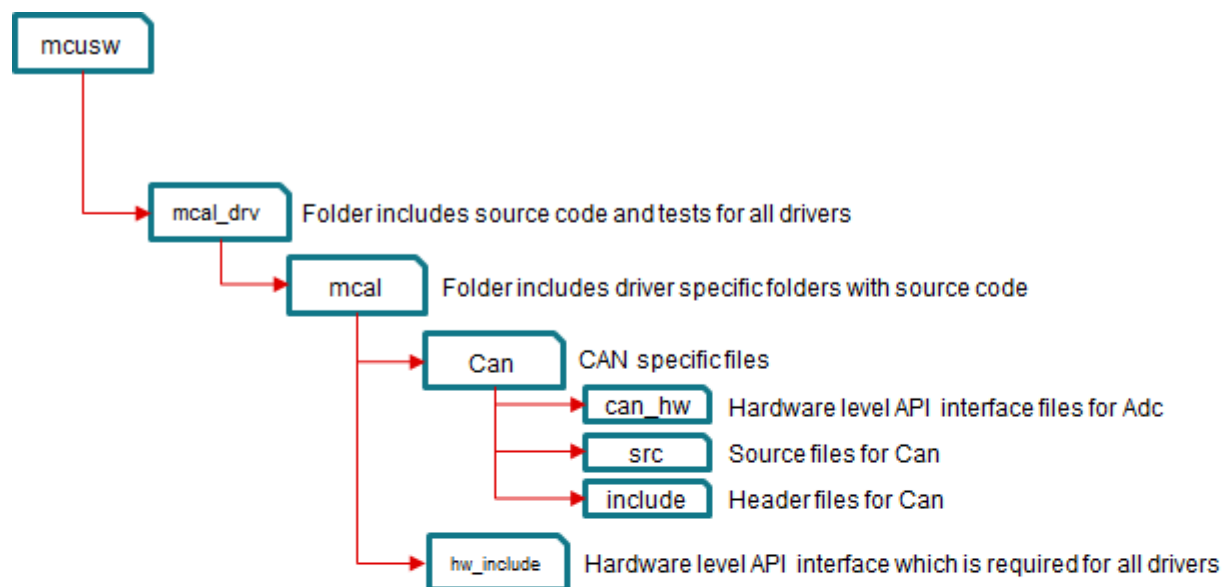
4.5 Directory Structure

The directory structure is as depicted in figures below, the source files can be categorized under “Driver Implementation” and “Example Application”

Driver Implemented by



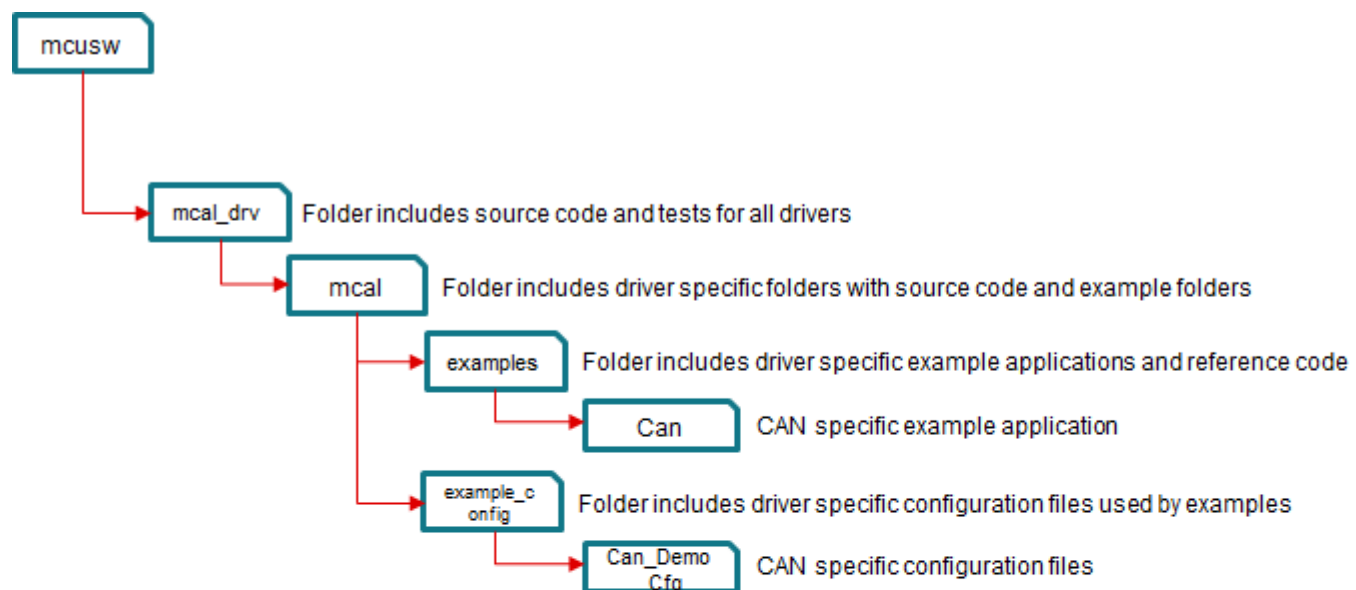
- Can.h and Can_Irq.h: Shall implement the interface provided by the driver
- Can.c, Can_Mcan.c, Can_Irq.c and Can_Priv.c: Shall implement the driver functionality











Example Application








- Can_Cfg.h and Can_Cfg.c: Shall implement the generated configuration for pre-compile variant
- Can_PBcfg.c: Shall implement the generated configuration for post-build variant
- CanApp.c: Shall implement the example application that demonstrates the use of the driver



Design ID	Description
 MCAL-6144 - SWS_Can_00397 : File Structure PUBLISHED	SWS_Can_00397 : File Structure
 MCAL-6075 - SWS_Can_00035 : File Structure PUBLISHED	SWS_Can_00035 : File Structure
 MCAL-6014 - SWS_Can_00436 : File Structure PUBLISHED	SWS_Can_00436 : File Structure
 MCAL-6004 - SWS_Can_00388 : File Structure PUBLISHED	SWS_Can_00388 : File Structure
 MCAL-5993 - SWS_Can_00435 : File Structure PUBLISHED	SWS_Can_00435 : File Structure
 MCAL-5965 - SWS_Can_00390 : File Structure PUBLISHED	SWS_Can_00390 : File Structure

4.6 Configurator





The AUTOSAR CAN Driver Specification details mandatory parameters that shall be configurable via the configurator. Please refer section 10 of [Reference 1 - AUTOSAR 4.3.1](#).

Design ID	Description
 MCAL-6140 - ECUC_Can_00489 : Can Configuration Container PUBLISHED	ECUC_Can_00489 : Can Configuration Container
 MCAL-5935 - ECUC_Can_00343 : CanConfigSet PUBLISHED	ECUC_Can_00343 : CanConfigSet
 MCAL-6079 - SWS_Can_00024 : Configuration Dependencies PUBLISHED	SWS_Can_00024 : Configuration Dependencies
 MCAL-6077 - SWS_Can_00022 : Code configuration PUBLISHED	SWS_Can_00022 : Code configuration
 MCAL-5949 - SWS_Can_00221 : VARIANT-POST-BUILD PUBLISHED	SWS_Can_00221 : VARIANT-POST-BUILD
 MCAL-5907 - SWS_Can_00220 : VARIANT-PRE-COMPILE PUBLISHED	SWS_Can_00220 : VARIANT-PRE-COMPILE
 MCAL-6061 - CanDeviceVariant PUBLISHED	CanDeviceVariant

4.6.1 NON Standard configurable parameters

Following lists this design's specific configurable parameters

Parameter	Usage comment
CanControllerInstance	Selects Can Controller Instance configured i.e MCAN0 or MCAN1
CanDefaultOSCounterId	Default Os Counter Id if node reference to OsCounter ref CanOsCounterRef is not set. The driver shall implement timed-wait for all waits (e.g. waiting for reset to complete). This timed wait shall use OS API GetCounterValue ()
CanTypeofInterruptFunction	Type of ISR function CAT1 : interrupt void func(void) CAT2 : ISR(func)
CanRegisterReadbackAPI	Enable/Disable configuration readback safe TI API.If this parameter is set to true the readback API shall be supported. Otherwise the API is not supported.
CanDemEventParameterRefs	Reference to the DemEventParameter which shall be issued when the error timeout on blocking API call occurs.
CanDeviceVariant	Select SOC being used. This parameter shall be used by driver to impose device specific constraints. The user guide shall detail the device specific constraints (if any)
CANFDCLK	Clock frequency that is provided to the Can FD module in Mhz. This clock is required to calculate BRP(Baud Rate Prescaler) value.


Parameter	Usage comment
CanLoopBackTest_Enable	Enable/Disable LoopBack test API.If this parameter is set to true the LoopBack mode shall be supported which is used for internal testing. Otherwise the API is not supported.
 MCAL-5995 - CanDemEventParameterRefs PUBLISHED	CanDemEventParameterRefs
 MCAL-5994 - ECUC_Can_00431 : CanOsCounterRef PUBLISHED	ECUC_Can_00431 : CanOsCounterRef
 MCAL-5903 - CanControllerInstance PUBLISHED	CanControllerInstance
 MCAL-6147 - CanLoopBackTest_Enable PUBLISHED	CanLoopBackTest_Enable
Design ID	Description

4.6.2 Implementation specific parameters (computed)

The configurator shall determine the maximum number of mailboxes and controllers that are configured and generate macros to define the same. This shall be used to perform range checks on controller and mailbox configurations provided at driver initialization time. Refer section [Refer section \(API's\)](#)

4.6.3 Variant Support

The driver shall support both VARIANT-POST-BUILD & VARIANT-PRE-COMPILE

Design ID	Description
 MCAL-6140 - ECUC_Can_00489 : Can Configuration Container PUBLISHED	ECUC_Can_00489 : Can Configuration Container

4.7 Error Classification

Errors are classified in two categories, development error and runtime / production error.

4.7.1 Development Errors




Type or error	Relevance	Related error code	Value [hex]
---------------	-----------	--------------------	----------------



API service called with wrong parameter	Development	CAN_E_PARAM_POINTER CAN_E_PARAM_HANDLE CAN_E_PARAM_DLC CAN_E_PARAM_CONTROLLER	0x01 0x02 0x03 0x04
API service used without initialization	Development	CAN_E_UNINIT	0x05
Invalid transition for the current mode.	Development	CAN_E_TRANSITION	0x06
Received can message is lost.	Development	CAN_E_DATALOST	0x07
Parameter baud rate has an invalid value.	Development	CAN_E_PARAM_BAUDRATE	0x08
invalid ICOM configuration Id	Development	CAN_E_ICOM_CONFIG_INVALID	0x09
Invalid Configuration set selection.	Development	CAN_E_INIT_FAILED	0x0A






Error Detection

The detection of development errors is configurable (ON / OFF) at pre-compile time. The switch CanDevErrorDetection will activate or deactivate the detection of all development errors.

Design ID	Description
 MCAL-6018 - ECUC_Can_00064 : CanDevErrorDetection PUBLISHED	ECUC_Can_00064 : CanDevErrorDetection
 MCAL-5977 - SWS_Can_00089 : Development Errors PUBLISHED	SWS_Can_00089 : Development Errors
 MCAL-5955 - SWS_Can_00026 : Development Errors PUBLISHED	SWS_Can_00026 : Development Errors

Error notification (DET)

All detected development errors are reported to Det_ReportError service of the Development Error Tracer (DET).

Design ID	Description
 MCAL-5911 - SWS_Can_00058 : Can Module User Services PUBLISHED	SWS_Can_00058 : Can Module User Services
 MCAL-6116 - SWS_Can_00007 : Event Triggered Notification PUBLISHED	SWS_Can_00007 : Event Triggered Notification
 MCAL-5929 - SWS_Can_00099 : Event Triggered Notification PUBLISHED	SWS_Can_00099 : Event Triggered Notification
 MCAL-6114 - ECUC_Can_00113 : CanTimeoutDuration PUBLISHED	ECUC_Can_00113 : CanTimeoutDuration
 MCAL-6022 - SWS_Can_00091 : Development Errors PUBLISHED	SWS_Can_00091 : Development Errors

4.7.2 Runtime Errors

The following runtime/production errors are detectable by Can driver.

Type of Error	Related Error code	Value (Hex)
This error is reported if any hardware read/write event failure occurs	CAN_E_HARDWARE_ERROR	Defined By Integrator



Error notification (DEM)

All detected run time errors shall be reported to Dem_ReportErrorStatus () service of the Diagnostic Event Manager (DEM).

5 Implementation Details

5.1 Data structures and resources






The sections below lists some of key data structures that shall be implemented and used in driver implementation.


5.1.1 Maximum number of controller and mailboxes.

These values depends on the SoC and integrators shall not modify the same.

Type	Identifier	Comments
uint32	CAN_MAX_CONTROLLER	Defines the maximum number of controllers that are configured. It's used internally by the driver object as size parameter to allocate can controller object type structures.
uint32	CAN_MAX_MAILBOXES	Defines the maximum number of mailboxes that are configured. It's used internally by the driver object as a size parameter to allocate can controller mailbox arrays.

5.1.2 Can_ControllerType






Type	Identifier	Comments
Can_BaudConfigType	*DefaultBaud	Pointer to the default Baud structure
Can_BaudConfigType	**BaudRateConfigList	List of available Baud rate structures
Design ID		Description
 MCAL-5910 - SWS_Can_91013 : Can_ControllerStateType PUBLISHED		SWS_Can_91013 : Can_ControllerStateType
 MCAL-6146 - ECUC_Can_00354 : CanController PUBLISHED		ECUC_Can_00354 : CanController
 MCAL-6098 - SWS_Can_00222 : Imported Type PUBLISHED		SWS_Can_00222 : Imported Type
 MCAL-6094 - SWS_Can_00039 : Can_ReturnType PUBLISHED		SWS_Can_00039 : Can_ReturnType
 MCAL-6036 - ECUC_Can_00065 : CanIdType PUBLISHED		ECUC_Can_00065 : CanIdType


Design ID	Description
 MCAL-6027 - SWS_Can_00416 : Can_IdType PUBLISHED	SWS_Can_00416 : Can_IdType

5.1.3 Can Controller Pre-Compile Configuration(Can_ControllerType_PC)

Type	Identifier	Comments
uint8	ControllerId	This parameter provides the controller ID which is unique in a given CAN Driver.
boolean	CntrActive	Defines if a CAN controller is used in the configuration
uint32	CntrAddr	Specifies the CAN controller base address.(Need to provide Message RAM Base Address, please refer MCAN Message RAM Configuration For details)
Can_TxRxProcessingType	RxInterrupt	CAN_TX_RX_PROCESSING_INTERRUPT/ CAN_TX_RX_PROCESSING_MIXED/CAN_TX_RX_PROCESSING_POLLING
Can_TxRxProcessingType	TxInterrupt	CAN_TX_RX_PROCESSING_INTERRUPT/ CAN_TX_RX_PROCESSING_MIXED/CAN_TX_RX_PROCESSING_POLLING












Type	Identifier	Comments
boolean	BusOffProcessingInterrupt	TRUE = Interrupt mode enabled FALSE = Polling mode
Can_ControllerInstance	CanControllerInst	Specifies theCan controller Instance selected.
boolean	CanFDModeEnabled	Controller is in CAN FD Mode or not.
Design ID		Description
 MCAL-5907 - SWS_Can_00220 : VARIANT-PRE-COMPILE PUBLISHED		SWS_Can_00220 : VARIANT-PRE-COMPILE
 MCAL-6035 - ECUC_Can_00317 : CanRxProcessing PUBLISHED		ECUC_Can_00317 : CanRxProcessing
 MCAL-6025 - ECUC_Can_00382 : CanControllerBaseAddress PUBLISHED		ECUC_Can_00382 : CanControllerBaseAddress
 MCAL-5986 - ECUC_Can_00316 : CanControllerId PUBLISHED		ECUC_Can_00316 : CanControllerId
 MCAL-5934 - ECUC_Can_00315 : CanControllerActivation PUBLISHED		ECUC_Can_00315 : CanControllerActivation

Design ID	Description
 MCAL-5951 - ECUC_Can_00490 : CanHardwareObjectUsesPolling PUBLISHED	ECUC_Can_00490 : CanHardwareObjectUsesPolling

5.1.4 Mailbox Configuration(Can_MailboxType)




Type	Identifier	Comments
uint8	CanHandleType	Specifies the type (Full-CAN or Basic-CAN) of a hardware object. .
uint32	MBIdType	CanIdType 0=standard 1=Extended 2= Mixed
Can_HwHandleType	HwHandle	Actual HW Mailbox object in the controller.
uint16	CanHwObjectCount	Number of hardware objects used to implement one HOH
Can_MailBoxDirectionType	MBDir	Direction of Mailbox whether TRANSMIT or RECEIVE
const Can_ControllerType_PC	*Controller	Reference to CAN Controller to which the HOH is associated to.

Type	Identifier	Comments
Can_HwFilterType	**HwFilterList	List of HW Filter structure of this mailbox.
uint32	HwFilterCnt	HW filter count
uint8	CanFdPaddingValue	If PduInfo->SduLength does not match possible DLC values CanDrv will use the next higher valid DLC for transmission with initialization of unused bytes to the value of the corresponding CanFdPaddingValue.
Design ID		Description
 MCAL-6142 - SWS_Can_00429 : Can_HwHandleType PUBLISHED		SWS_Can_00429 : Can_HwHandleType
 MCAL-5906 - ECUC_Can_00323 : CanHandleType PUBLISHED		ECUC_Can_00323 : CanHandleType
 MCAL-6083 - SWS_CAN_00496 : Can_HwType PUBLISHED		SWS_CAN_00496 : Can_HwType
 MCAL-6168 - ECUC_Can_00467 : CanHwObjectCount PUBLISHED		ECUC_Can_00467 : CanHwObjectCount

Design ID	Description
 MCAL-6046 - SWS_Can_00386 : Driver Scope CAN Hardware Units of same type PUBLISHED	SWS_Can_00386 : Driver Scope CAN Hardware Units of same type
 MCAL-6023 - ECUC_Can_00468 : CanHwFilter PUBLISHED	ECUC_Can_00468 : CanHwFilter
 MCAL-6020 - ECUC_Can_00322 : CanControllerRef PUBLISHED	ECUC_Can_00322 : CanControllerRef
 MCAL-6000 - ECUC_Can_00470 : CanHwFilterMask PUBLISHED	ECUC_Can_00470 : CanHwFilterMask
 MCAL-5918 - ECUC_Can_00469 : CanHwFilterCode PUBLISHED	ECUC_Can_00469 : CanHwFilterCode




5.1.5 Can Mailbox Pre-Compile Configuration(Can_MailboxType_PC)

Type	Identifier	Comments
uint16	CanObjectId	Holds the handle ID of HRH or HTH.

Design ID	Description
 MCAL-6163 - ECUC_Can_00326 : CanObjectId PUBLISHED	ECUC_Can_00326 : CanObjectId
 MCAL-6110 - ECUC_Can_00324 : CanHardwareObject PUBLISHED	ECUC_Can_00324 : CanHardwareObject
 MCAL-6102 - ECUC_Can_00327 : CanObjectType PUBLISHED	ECUC_Can_00327 : CanObjectType

5.1.6 Can Configuration(Can_ConfigType)

Type	Identifier	Comments
Can_ControllerType	**CanControllerList	List of enabled Controllers.
uint8	CanMaxControllerCount	MaxCount of Controller in Controller List
Can_MailboxType	**MailBoxList	MailBox array for all controllers.
boolean	MaxMbCnt	Max MailBox Count in MB list in all controller

Type	Identifier	Comments
uint32	MaxBaudConfigID	Max Baud Config Index in BaudRateConfigList in all controller
 MCAL-5983 - SWS_CAN_00487 : CanGeneral Types PUBLISHED	SWS_CAN_00487 : CanGeneral Types	
 MCAL-5904 - SWS_Can_00440 : CanGeneral Types PUBLISHED	SWS_Can_00440 : CanGeneral Types	
Design ID	Description	
 MCAL-5980 - SWS_Can_00413 : Can_ConfigType PUBLISHED	SWS_Can_00413 : Can_ConfigType	
 MCAL-6158 - SWS_Can_00439 : CanGeneral Types PUBLISHED	SWS_Can_00439 : CanGeneral Types	
 MCAL-6111 - ECUC_Can_00320 : CanIndex PUBLISHED	ECUC_Can_00320 : CanIndex	
 MCAL-5979 - ECUC_Can_00328 : CanGeneral PUBLISHED	ECUC_Can_00328 : CanGeneral	

5.2 Dynamic Behavior - Control Flow Diagram

5.2.1 States

As detailed in section 7.3 of [Reference 1 - AUTOSAR 4.3.1](#), Can would be in one of the following states. Un Initialized, Initialized, started, stopped, sleep and wakeup. Please note that "wakeup" is not supported, refer ([Features Not Supported / NON Compliance](#))

A variable shall be maintained on per channel basis to track and maintain the state. The diagram below shows transitions of states and it's associated service API's.



CAN Controller State : Sourced from AUTOSAR Spec

States are

- **STATE_UNINT:** This is the state when the Hardware is just started.
- **STATE_STOPPED:** This is the state in which Hardware is in when after the initialization routine is called. Can controller is fully initialized but does not participate in the bus transactions.
- **STATE_START:** This is the state in which hardware is in when it is fully operational ie., it is sending and receive messages from the bus on CAN network
- **STATE_SLEEP:** This is the state in which the hardware is in when the controller is sleeping.

Service API [Can_SetControllerMode\(\)](#) can be used to transition into following states CAN_T_START, CAN_T_STOP, CAN_T_SLEEP.

5.3 Dynamic Behavior - Data Flow Diagram

Not Applicable

5.4 Application Parameters

Can_Init

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
CfgPtr	Pointer to post-build configuration data.	0-0xFFFFFFFF	-	-	N.A

Can_SetControllerMode

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
Controller	CAN controller for which the status shall be changed	0-0xFF	-	-	N.A
Transition	Transition value to request new CAN controller state	0-3	-	-	N.A

5.4.1 Can_Write

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
Hth	Information which HW-transmit handle shall be used for transmit. Implicitly this is also the information about the controller to use because the Hth numbers are unique inside one hardware unit	0-0xFF	-	-	N.A
*PduInfo	Pointer to SDU user memory, Data Length and Identifier.	0-0xFFFFFFFF	-	-	N.A

5.4.2 Can_DisableControllerInterrupts

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
Controller	CAN controller for which interrupts shall be disabled.	0-0xFF	-	-	N.A



5.4.3 Can_EnableControllerInterrupts

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
Controller	CAN controller for which interrupts shall be re-enabled	0-0xFF	-	-	N.A

5.4.4 Can_MainFunction_Write

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
void	-	-	-	-	N.A



5.4.5 Can_MainFunction_BusOff

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
void	-	-	-	-	N.A

5.4.6 Can_MainFunction_Read

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
void	-	-	-	-	N.A

5.4.7 Can_MainFunction_Wakeup

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
-----------	-------------	-----------------------	---------------	---------------	---------



void	-	-	-	-	N.A
------	---	---	---	---	-----

5.4.8 Can_GetVersionInfo

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
versioninfo	Pointer to where to store the version information of this module.	0-0xFFFFFFFF	-	-	N.A

5.4.9

Can_MainFunction_Mode

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
void	-	-	-	-	N.A

5.4.10 Can_TestLoopBackModeEnable

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
Controller	CAN controller for which interrupts shall be Re-enabled.	0-0xFF	-	-	N.A
Mode	0 - Disable Digital mode. 1 - Disable Analog mode.	0-0xFF	-	-	N.A

5.4.11 Can_TestLoopBackModeDisable

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
Controller	CAN controller for which interrupts shall be disabled.	0-0xFF	-	-	N.A
Mode	0 - Disable Digital mode. 1 - Disable Analog mode.	0-0xFF	-	-	N.A

5.4.12 Can_RegisterReadbackType.

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
Controller	CAN controller for which Register readback has to be performed	0-0xFF	-	-	N.A
RegRbPtr	Inout parameters: Pointer to where to store the readback values.	0-0xFFFFFFFF	-	-	N.A

5.4.13 Can_SetBaudrate

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
Controller	CAN controller, whose baud rate shall be set	0-0xFF	-	-	N.A
BaudRateConfigID	references a baud rate configuration by ID	0-0xFFFF	-	-	N.A

5.4.14 Can_GetControllerMode

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
Controller	CAN controller for which the status shall be requested.	0-0xFF	-	-	N.A
ControllerModePtr	Pointer to a memory location, where the current mode of the CAN controller will be stored.	0-0xFF	-	-	N.A

5.4.15 Can_GetControllerErrorState

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
-----------	-------------	-----------------------	---------------	---------------	---------



ControllerId	Abstracted CanIf ControllerId which is assigned to a CAN controller, which is requested for ErrorState	0-0xFF	-	-	N.A
ErrorStatePtr	Pointer to a memory location, where the error state of the CAN controller will be stored.	0-0xFFFFFFFF	-	-	N.A

5.4.16 Can_DelInit

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
void	-	-	-	-	N.A

5.5 Safety Diagnostic Features

MCAN2 - CRC in Message

MCAN hardware logic includes embedded diagnostic that shall detect CRC error. If the CRC check sum of a received message is incorrect, then CRC error shall be generated. Protocol error in Arbitration phase is generated when CRC error is detected. MCAN driver has API to read the status of CRC error. Error response/handling is the responsibility of the System Integrator.

MCAN4 - Timeout on FIFO activity

MCAN hardware logic includes embedded diagnostic that shall detect timeout on FIFO Activity. MCAN driver has API to detect timeout on FIFO activity. Error response/handling is the responsibility of the System Integrator.

MCAN6 - Stuff Error Detection

MCAN hardware logic includes embedded diagnostic that shall detect Stuff error. Whenever a transmitter detects five consecutive bits of identical value in the bitstream to be transmitted, it automatically inserts a complementary bit into the actual transmitted bit stream. If a 6th consecutive equal bit is detected in a received segment that should have been coded by bit stuffing, the CAN module will flag a Stuff Error. Protocol error in Arbitration phase is generated when stuff error is detected. MCAN driver has API to read the status of stuff error. Error response/handling is the responsibility of the System Integrator

MCAN7 - Form Error Detection



MCAN hardware logic includes embedded diagnostic that shall detect form error. A fixed format part of a received frame has the wrong format then it will be form error. Protocol error in Arbitration phase is generated when form error is detected. MCAN driver has API to read the status of form error. Error response/handling is the responsibility of the System Integrator

MCAN8 - Acknowledge Error Detection

MCAN hardware logic includes embedded diagnostic that shall detect ACK error. The message transmitted by the MCAN module was not acknowledged by another node then ACK error occurred. Protocol error in Arbitration phase is generated when ACK error is detected. MCAN driver has API to read the status of ACK error. Error response/handling is the responsibility of the System Integrator

MCAN9 - Bit Error Detection

MCAN hardware logic includes embedded diagnostic that shall detect bit error. Protocol error in Arbitration phase is generated when bit error is detected. MCAN driver has API to read the status of bit error. Error response/handling is the responsibility of the System Integrator

MCAN12 - Software readback of written configuration / MCAN14 - Periodic software readback of static configuration registers

This shall be handled by immediately reading back the written config to the registers. This shall be handled in MCAN API implementation in MCAL. MCAL SW shall support this diagnostic inside the module and raise an error from the API in case of any issue. Error response/handling is the responsibility of the System Integrator.



MCAN-Driver should provide the API to readback the static config registers for the MCAN module.






- MCANSS_CTRL
- MCAN_DBTP
- MCAN_RWD





- MCAN_CCCR
- MCAN_NBTP
- MCAN_TDCR
- MCAN_GFC
- MCAN_SIDFC
- MCAN_XIDFC
- MCAN_RXF0C
- MCAN_RXBC
- MCAN_RXF1C
- MCAN_TXBC
- MCAN_TXESC
- MCAN_TXEFC

System Integrator shall use this API for comparison.

Design ID	Description
 MCAL-6126 - CAN: Safety Diagnostics: MCAN9: Bit Error Detection PUBLISHED	CAN: Safety Diagnostics: MCAN9: Bit Error Detection
 MCAL-5909 - CAN: Safety Diagnostics: MCAN7: Form Error Detection PUBLISHED	CAN: Safety Diagnostics: MCAN7: Form Error Detection

Design ID	Description
 MCAL-5917 - CAN: Safety Diagnostics: MCAN8: Acknowledge Error Detection PUBLISHED	CAN: Safety Diagnostics: MCAN8: Acknowledge Error Detection
 MCAL-5922 - CAN: Safety Diagnostics: MCAN4: Timeout on FIFO activity PUBLISHED	CAN: Safety Diagnostics: MCAN4: Timeout on FIFO activity
 MCAL-6028 - CAN: Safety Diagnostics: MCAN12: Software Readback of written configuration PUBLISHED	CAN: Safety Diagnostics: MCAN12: Software Readback of written configuration.
 MCAL-6088 - CAN: Safety Diagnostics: MCAN14: Periodic software readback of static configuration registers. PUBLISHED	CAN: Safety Diagnostics: MCAN14: Periodic software readback of static configuration registers.
 MCAL-6149 - CanRegisterReadbackAPI PUBLISHED	CanRegisterReadbackAPI

Design ID	Description
 MCAL-6052 - CAN: Safety Diagnostics: MCAN6: Stuff Error Detection PUBLISHED	CAN: Safety Diagnostics: MCAN6: Stuff Error Detection
 MCAL-6091 - CAN: Safety Diagnostics: MCAN2: CRC in Message PUBLISHED	CAN: Safety Diagnostics: MCAN2: CRC in Message




6 Low Level Definitions







6.1 Driver API's

For the standard API's please refer 8.3 of the *CAN Driver* AutoSar Specification as listed in [Reference 1 - AUTOSAR 4.3.1](#). Sections below highlight other design considerations for the implementation.

6.1.1 Can_Init




Refer section 8.3.1 [Reference 1 - AUTOSAR 4.3.1](#).






Design Identifier	Description
 MCAL-6090 - SWS_Can_00255 : Can_Init Requirements PUBLISHED	SWS_Can_00255 : Can_Init Requirements
 MCAL-6053 - SWS_Can_00174 : Can_Init DET Driver State CAN_UNINIT PUBLISHED	SWS_Can_00174 : Can_Init DET Driver State CAN_UNINIT
 MCAL-6017 - SWS_Can_00250 : Can_Init Requirements PUBLISHED	SWS_Can_00250 : Can_Init Requirements






Design Identifier	Description
 MCAL-6015 - SWS_Can_00236 : Can_Init ignore remote transmission requests PUBLISHED	SWS_Can_00236 : Can_Init ignore remote transmission requests
 MCAL-6006 - SWS_Can_00407 : Can_Init Requirements PUBLISHED	SWS_Can_00407 : Can_Init Requirements
 MCAL-6003 - SWS_Can_00053 : Can_Init Requirements PUBLISHED	SWS_Can_00053 : Can_Init Requirements
 MCAL-5988 - SWS_Can_00291 : Can_Init Requirements PUBLISHED	SWS_Can_00291 : Can_Init Requirements
 MCAL-5963 - SWS_Can_00021 : Can_Init Requirements PUBLISHED	SWS_Can_00021 : Can_Init Requirements
 MCAL-5908 - SWS_Can_00408 : Can_Init DET Controller State UNINIT PUBLISHED	SWS_Can_00408 : Can_Init DET Controller State UNINIT






6.1.2 Can_SetControllerMode







Refer section 8.3.1 [Reference 1 - AUTOSAR 4.3.1](#).







Design Identifier	Description
 MCAL-6172 - SWS_Can_00412 : Can_SetControllerMode state transitions(SLEEP -> STOPPED) PUBLISHED	SWS_Can_00412 : Can_SetControllerMode state transitions(SLEEP -> STOPPED)
 MCAL-6169 - SWS_Can_00261 : Can_SetControllerMode state transitions(STOPPED -> STARTED) PUBLISHED	SWS_Can_00261 : Can_SetControllerMode state transitions(STOPPED -> STARTED)
 MCAL-6152 - SWS_Can_00409 : Can_SetControllerMode state transitions(STOPPED -> STARTED) PUBLISHED	SWS_Can_00409 : Can_SetControllerMode state transitions(STOPPED -> STARTED)
 MCAL-6133 - SWS_Can_00267 : Can_SetControllerMode state transitions(SLEEP -> STOPPED) PUBLISHED	SWS_Can_00267 : Can_SetControllerMode state transitions(SLEEP -> STOPPED)




Design Identifier	Description
 MCAL-6118 - SWS_Can_00405 : Can_SetControllerMode state transitions(STOPPED -> SLEEP) PUBLISHED	SWS_Can_00405 : Can_SetControllerMode state transitions(STOPPED -> SLEEP)
 MCAL-6073 - SWS_Can_00263 : Can_SetControllerMode state transitions(STARTED -> STOPPED) PUBLISHED	SWS_Can_00263 : Can_SetControllerMode state transitions(STARTED -> STOPPED)
 MCAL-6050 - SWS_Can_00264 : Can_SetControllerMode state transitions(STARTED -> STOPPED) PUBLISHED	SWS_Can_00264 : Can_SetControllerMode state transitions(STARTED -> STOPPED)
 MCAL-6011 - SWS_Can_00290 : Can_SetControllerMode state transitions(STOPPED -> SLEEP) PUBLISHED	SWS_Can_00290 : Can_SetControllerMode state transitions(STOPPED -> SLEEP)
 MCAL-6010 - SWS_Can_00282 : Can_SetControllerMode state transitions(STARTED -> STOPPED) PUBLISHED	SWS_Can_00282 : Can_SetControllerMode state transitions(STARTED -> STOPPED)

Design Identifier	Description
 MCAL-6001 - SWS_Can_00410 : Can_SetControllerMode state transitions(STARTED -> STOPPED) PUBLISHED	SWS_Can_00410 : Can_SetControllerMode state transitions(STARTED -> STOPPED)
 MCAL-5978 - SWS_Can_00268 : Can_SetControllerMode state transitions(SLEEP -> STOPPED) PUBLISHED	SWS_Can_00268 : Can_SetControllerMode state transitions(SLEEP -> STOPPED)
 MCAL-5975 - SWS_Can_00262 : Can_SetControllerMode state transitions(STOPPED -> STARTED) PUBLISHED	SWS_Can_00262 : Can_SetControllerMode state transitions(STOPPED -> STARTED)
 MCAL-5952 - SWS_Can_00265 : Can_SetControllerMode state transitions(STOPPED -> SLEEP) PUBLISHED	SWS_Can_00265 : Can_SetControllerMode state transitions(STOPPED -> SLEEP)
 MCAL-5940 - SWS_Can_00411 : Can_SetControllerMode state transitions(STOPPED -> SLEEP) PUBLISHED	SWS_Can_00411 : Can_SetControllerMode state transitions(STOPPED -> SLEEP)

Design Identifier	Description
 MCAL-5992 - SWS_Can_00372 : Can_SetControllerMode state transitions PUBLISHED	SWS_Can_00372 : Can_SetControllerMode state transitions
 MCAL-6122 - SWS_Can_00373 : Can_SetControllerMode state transitions PUBLISHED	SWS_Can_00373 : Can_SetControllerMode state transitions
 MCAL-6125 - SWS_Can_00370 : Can_SetControllerMode state transitions PUBLISHED	SWS_Can_00370 : Can_SetControllerMode state transitions
 MCAL-6128 - SWS_Can_00017 : Can_SetControllerMode Software triggered state transitions PUBLISHED	SWS_Can_00017 : Can_SetControllerMode Software triggered state transitions
 MCAL-6161 - SWS_Can_00274 : Bus Off state transitions(STARTED -> STOPPED) PUBLISHED	SWS_Can_00274 : Bus Off state transitions(STARTED -> STOPPED)


Design Identifier	Description
 MCAL-6154 - SWS_Can_00384 : Can_SetControllerMode State transition value CAN_T_START PUBLISHED	SWS_Can_00384 : Can_SetControllerMode State transition value CAN_T_START
 MCAL-6151 - SWS_Can_91010 : Controller State UNINIT PUBLISHED	SWS_Can_91010 : Controller State UNINIT
 MCAL-6115 - SWS_Can_00246 : Driver State Machine CAN_READY PUBLISHED	SWS_Can_00246 : Driver State Machine CAN_READY
 MCAL-6096 - SWS_Can_00259 : Driver State Machine CAN_STOPPED PUBLISHED	SWS_Can_00259 : Driver State Machine CAN_STOPPED
 MCAL-6087 - SWS_Can_00258 : Driver State Machine CAN_SLEEP PUBLISHED	SWS_Can_00258 : Driver State Machine CAN_SLEEP
 MCAL-6043 - SWS_Can_00272 : Bus Off state transitions(STARTED -> STOPPED) PUBLISHED	SWS_Can_00272 : Bus Off state transitions(STARTED -> STOPPED)







Design Identifier	Description
 MCAL-6026 - SWS_Can_00199 : Can_SetControllerMode DET CAN_E_PARAM_CONTROLLER PUBLISHED	SWS_Can_00199 : Can_SetControllerMode DET CAN_E_PARAM_CONTROLLER
 MCAL-5991 - SWS_Can_00417 : Can_StateTransitionType PUBLISHED	SWS_Can_00417 : Can_StateTransitionType
 MCAL-5982 - SWS_Can_00404 : Driver State Machine CAN_SLEEP PUBLISHED	SWS_Can_00404 : Driver State Machine CAN_SLEEP
 MCAL-5958 - SWS_Can_91009 : Driver State Machine CAN_UNINIT PUBLISHED	SWS_Can_91009 : Driver State Machine CAN_UNINIT
 MCAL-5945 - SWS_Can_00020 : Bus Off state transitions(STARTED -> STOPPED) PUBLISHED	SWS_Can_00020 : Bus Off state transitions(STARTED -> STOPPED)
 MCAL-5936 - SWS_Can_00103 : Driver State Machine CAN_UNINIT PUBLISHED	SWS_Can_00103 : Driver State Machine CAN_UNINIT







Design Identifier	Description
 MCAL-5933 - SWS_Can_00398 : Can_SetControllerMode state transitions PUBLISHED	SWS_Can_00398 : Can_SetControllerMode state transitions
 MCAL-5912 - SWS_Can_00200 : Can_SetControllerMode DET CAN_E_TRANSITION PUBLISHED	SWS_Can_00200 : Can_SetControllerMode DET CAN_E_TRANSITION
 MCAL-6181 - SWS_Can_00245 : Driver State Machine CAN_START PUBLISHED	SWS_Can_00245 : Driver State Machine CAN_START


6.1.3 Can_Write

Refer section 8.3.1 [Reference 1 - AUTOSAR 4.3.1](#).

Design Identifier	Description
 MCAL-5962 - SWS_Can_00033 : Implement interrupt service routines PUBLISHED	SWS_Can_00275 : Can_Write Non-blocking




Design Identifier	Description
 MCAL-6132 - SWS_Can_00213 : Can_Write CAN_BUSY PUBLISHED	SWS_Can_00213 : Can_Write CAN_BUSY
 MCAL-6119 - SWS_Can_00214 : Can_Write Preemptive call sequence PUBLISHED	SWS_Can_00214 : Can_Write Preemptive call sequence
 MCAL-6112 - SWS_CAN_00505 : Can_Write DET CAN_E_PARAM_POINTER PUBLISHED	SWS_CAN_00505 : Can_Write DET CAN_E_PARAM_POINTER
 MCAL-6105 - SWS_CAN_00506 : Can_Write DET CAN_E_PARAM_POINTER PUBLISHED	SWS_CAN_00506 : Can_Write DET CAN_E_PARAM_POINTER
 MCAL-6095 - SWS_Can_00216 : Can_Write DET CAN_E_UNINIT PUBLISHED	SWS_Can_00216 : Can_Write DET CAN_E_UNINIT
 MCAL-6072 - SWS_Can_00212 : Can_Write Frame CAN message PUBLISHED	SWS_Can_00212 : Can_Write Frame CAN message







Design Identifier	Description
 MCAL-6029 - SWS_CAN_00502 : Can_Write Frame CanFdPaddingValue PUBLISHED	SWS_CAN_00502 : Can_Write Frame CanFdPaddingValue
 MCAL-6016 - SWS_Can_00217 : Can_Write DET CAN_E_PARAM_HANDLE PUBLISHED	SWS_Can_00217 : Can_Write DET CAN_E_PARAM_HANDLE
 MCAL-5999 - SWS_CAN_00486 : Can_Write Frame type detection PUBLISHED	SWS_CAN_00486 : Can_Write Frame type detection
 MCAL-5916 - SWS_CAN_00219 : Can_Write DET CAN_E_PARAM_POINTER PUBLISHED	SWS_CAN_00219 : Can_Write DET CAN_E_PARAM_POINTER
 MCAL-5913 - SWS_Can_00218 : Can_Write DET CAN_E_PARAM_DLC PUBLISHED	SWS_Can_00218 : Can_Write DET CAN_E_PARAM_DLC
 MCAL-6164 - SWS_Can_00275 : Can_Write Non-blocking. PUBLISHED	SWS_Can_00275 : Can_Write Non-blocking.

Design Identifier	Description
 MCAL-6076 - ECUC_Can_00485 : CanFdPaddingValue PUBLISHED	ECUC_Can_00485 : CanFdPaddingValue

6.1.4 Can_DisableControllerInterrupts





Refer section 8.3.1 [Reference 1 - AUTOSAR 4.3.1](#).





Design Identifier	Description
 MCAL-6153 - SWS_Can_00049 : Can_DisableControllerInterrupts Disable interrupts PUBLISHED	SWS_Can_00049 : Can_DisableControllerInterrupts Disable interrupts
 MCAL-6150 - SWS_Can_00205 : Can_DisableControllerInterrupts DET CAN_E_UNINIT PUBLISHED	SWS_Can_00205 : Can_DisableControllerInterrupts DET CAN_E_UNINIT
 MCAL-6002 - SWS_Can_00204 : Can_DisableControllerInterrupts Track individual enabling and disabling of interrupts PUBLISHED	SWS_Can_00204 : Can_DisableControllerInterrupts Track individual enabling and disabling of interrupts

Design Identifier	Description
 MCAL-5989 - SWS_Can_00206 : Can_DisableControllerInterrupts DET CAN_E_PARAM_CONTROLLER PUBLISHED	SWS_Can_00206 : Can_DisableControllerInterrupts DET CAN_E_PARAM_CONTROLLER
 MCAL-5938 - SWS_Can_00202 : Can_DisableControllerInterrupts Disable interrupts PUBLISHED	SWS_Can_00202 : Can_DisableControllerInterrupts Disable interrupts
 MCAL-6130 - SWS_Can_00197 : Can_SetControllerMode Disable interrupts PUBLISHED	SWS_Can_00197 : Can_SetControllerMode Disable interrupts
 MCAL-6047 - SWS_Can_00420 : Reset interrupt flag PUBLISHED	SWS_Can_00420 : Reset interrupt flag
 MCAL-5985 - SWS_Can_00419 : Disable unused interrupts PUBLISHED	SWS_Can_00419 : Disable unused interrupts
 MCAL-5914 - SWS_Can_00426 : Can_SetControllerMode Disable interrupts PUBLISHED	SWS_Can_00426 : Can_SetControllerMode Disable interrupts

6.1.5 Can_EnableControllerInterrupts





Refer section 8.3.1 [Reference 1 - AUTOSAR 4.3.1](#).

Design Identifier	Description
 MCAL-6068 - SWS_Can_00050 : Can_EnableControllerInterrupts Enable interrupts PUBLISHED	SWS_Can_00050 : Can_EnableControllerInterrupts Enable interrupts
 MCAL-6030 - SWS_Can_00210 : Can_EnableControllerInterrupts DET CAN_E_PARAM_CONTROLLER PUBLISHED	SWS_Can_00210 : Can_EnableControllerInterrupts DET CAN_E_PARAM_CONTROLLER
 MCAL-5948 - SWS_Can_00208 : Can_EnableControllerInterrupts Enable interrupts PUBLISHED	SWS_Can_00208 : Can_EnableControllerInterrupts Enable interrupts
 MCAL-5919 - SWS_Can_00209 : Can_EnableControllerInterrupts DET CAN_E_UNINIT PUBLISHED	SWS_Can_00209 : Can_EnableControllerInterrupts DET CAN_E_UNINIT

Design Identifier	Description
 MCAL-6156 - SWS_Can_00196 : Can_SetControllerMode Enable interrupts PUBLISHED	SWS_Can_00196 : Can_SetControllerMode Enable interrupts
 MCAL-6156 - SWS_Can_00196 : Can_SetControllerMode Enable interrupts PUBLISHED	SWS_Can_00425 : Can_SetControllerMode Enable interrupts
 MCAL-5950 - CanTypeofInterruptFunction PUBLISHED	CanTypeofInterruptFunction
 MCAL-6097 - SWS_Can_00425 : Can_SetControllerMode Enable interrupts PUBLISHED	SWS_Can_00425 : Can_SetControllerMode Enable interrupts







6.1.6 Can_MainFunction_Write

Refer section 8.3.1 [Reference 1 - AUTOSAR 4.3.1](#).

Design Identifier	Description
 MCAL-6107 - SWS_Can_00031 : Can_MainFunction_Write CanTxProcessing POLLING PUBLISHED	SWS_Can_00031 : Can_MainFunction_Write CanTxProcessing POLLING
 MCAL-6021 - SWS_Can_00179 : Can_MainFunction_Write DET CAN_E_UNINIT PUBLISHED	SWS_Can_00179 : Can_MainFunction_Write DET CAN_E_UNINIT
 MCAL-5942 - SWS_Can_00178 : Can_MainFunction_Write CanTxProcessing No POLLING PUBLISHED	SWS_Can_00178 : Can_MainFunction_Write CanTxProcessing No POLLING
 MCAL-5968 - SWS_Can_00011 : Transmit Data Consistency PUBLISHED	SWS_Can_00011 : Transmit Data Consistency






6.1.7 Can_MainFunction_BusOff


Refer section 8.3.1 [Reference 1 - AUTOSAR 4.3.1](#).

Design Identifier	Description
 MCAL-6054 - SWS_Can_00109 : Can_MainFunction_BusOff POLLING PUBLISHED	SWS_Can_00109 : Can_MainFunction_BusOff POLLING
 MCAL-6049 - SWS_Can_00184 : Can_MainFunction_BusOff DET CAN_E_UNINIT PUBLISHED	SWS_Can_00184 : Can_MainFunction_BusOff DET CAN_E_UNINIT
 MCAL-6039 - SWS_Can_00183 : Can_MainFunction_BusOff No POLLING PUBLISHED	SWS_Can_00183 : Can_MainFunction_BusOff No POLLING
 MCAL-6176 - ECUC_Can_00355 : CanMainFunctionBusoffPeriod PUBLISHED	ECUC_Can_00355 : CanMainFunctionBusoffPeriod
 MCAL-6134 - SWS_Can_00273 : Bus Off state transitions(STARTED -> STOPPED) PUBLISHED	SWS_Can_00273 : Bus Off state transitions(STARTED -> STOPPED)
 MCAL-6131 - ECUC_Can_00314 : CanBusoffProcessing PUBLISHED	ECUC_Can_00314 : CanBusoffProcessing

6.1.8 Can_MainFunction_Read




Refer section 8.3.1 [Reference 1 - AUTOSAR 4.3.1](#).

Design Identifier	Description
 MCAL-5964 - SWS_Can_00180 : Can_MainFunction_Read CanRxProcessing No POLLING PUBLISHED	SWS_Can_00180 : Can_MainFunction_Read CanRxProcessing No POLLING
 MCAL-5961 - SWS_Can_00108 : Can_MainFunction_Read CanRxProcessing POLLING PUBLISHED	SWS_Can_00108 : Can_MainFunction_Read CanRxProcessing POLLING
 MCAL-5941 - SWS_Can_00181 : Can_MainFunction_Read DET CAN_E_UNINIT PUBLISHED	SWS_Can_00181 : Can_MainFunction_Read DET CAN_E_UNINIT
 MCAL-6057 - SWS_Can_00395 : Receive Data Consistency PUBLISHED	SWS_Can_00395 : Receive Data Consistency
 MCAL-6012 - SWS_Can_00012 : Receive Data Consistency PUBLISHED	SWS_Can_00012 : Receive Data Consistency

Design Identifier	Description
 MCAL-5947 - SWS_CAN_00489 : Receive Data Consistency PUBLISHED	SWS_CAN_00489 : Receive Data Consistency

6.1.9 Can_MainFunction_Wakeup

Refer section 8.3.1 [Reference 1 - AUTOSAR 4.3.1](#).

Design Identifier	Description
 MCAL-6101 - SWS_Can_00185 : Can_MainFunction_Wakeup No POLLING PUBLISHED	SWS_Can_00185 : Can_MainFunction_Wakeup No POLLING
 MCAL-6135 - ECUC_Can_00319 : CanWakeupProcessing PUBLISHED	ECUC_Can_00319 : CanWakeupProcessing
 MCAL-6060 - ECUC_Can_00357 : CanMainFunctionWakeupPeriod PUBLISHED	ECUC_Can_00357 : CanMainFunctionWakeupPeriod

6.1.10 Can_GetVersionInfo







Refer section 8.3.1 [Reference 1 - AUTOSAR 4.3.1](#).

Design Identifier	Description
 MCAL-6084 - SWS_Can_00177 : Can_GetVersionInfo DET CAN_E_PARAM_POINTER PUBLISHED	SWS_Can_00177 : Can_GetVersionInfo DET CAN_E_PARAM_POINTER
 MCAL-6063 - ECUC_Can_00106 : CanVersionInfoApi PUBLISHED	ECUC_Can_00106 : CanVersionInfoApi

6.1.11 Can_MainFunction_Mode








Refer section 8.3.1 [Reference 1 - AUTOSAR 4.3.1](#).








Design Identifier	Description
 MCAL-6113 - SWS_Can_00379 : Can_MainFunction_Mode DET CAN_E_UNINIT PUBLISHED	SWS_Can_00379 : Can_MainFunction_Mode DET CAN_E_UNINIT



Design Identifier	Description
 MCAL-6070 - SWS_Can_00369 : Can_MainFunction_Mode POLLING PUBLISHED	SWS_Can_00369 : Can_MainFunction_Mode POLLING
 MCAL-6078 - ECUC_Can_00376 : CanMainFunctionModePeriod PUBLISHED	ECUC_Can_00376 : CanMainFunctionModePeriod
 MCAL-6070 - SWS_Can_00369 : Can_MainFunction_Mode POLLING PUBLISHED	SWS_Can_00369 : Can_MainFunction_Mode POLLING
 MCAL-6034 - ECUC_Can_00438 : CanMainFunctionRWPeriodRef PUBLISHED	ECUC_Can_00438 : CanMainFunctionRWPeriodRef
 MCAL-6032 - ECUC_Can_00437 : CanMainFunctionRWPeriods PUBLISHED	ECUC_Can_00437 : CanMainFunctionRWPeriods
 MCAL-6009 - ECUC_Can_00484 : CanMainFunctionPeriod PUBLISHED	ECUC_Can_00484 : CanMainFunctionPeriod

6.1.12 Can_SetBaudrate

Refer section 8.3.1 [Reference 1 - AUTOSAR 4.3.1](#).

Design Identifier	Description
 MCAL-6174 - SWS_CAN_00500 : Can_SetBaudrate Re-initialization PUBLISHED	SWS_CAN_00500 : Can_SetBaudrate Re-initialization
 MCAL-6162 - SWS_Can_00260 : Can_SetBaudrate state transitions PUBLISHED	SWS_Can_00260 : Can_SetBaudrate state transitions
 MCAL-6086 - SWS_Can_00056 : Can_SetBaudrate Requirements PUBLISHED	SWS_Can_00056 : Can_SetBaudrate Requirements
 MCAL-6008 - SWS_CAN_00492 : Can_SetBaudrate DET CAN_E_UNINIT PUBLISHED	SWS_CAN_00492 : Can_SetBaudrate DET CAN_E_UNINIT
 MCAL-5984 - SWS_Can_00256 : Can_SetBaudrate state transitions PUBLISHED	SWS_Can_00256 : Can_SetBaudrate state transitions
 MCAL-5960 - SWS_Can_00422 : Can_SetBaudrate state transitions PUBLISHED	SWS_Can_00422 : Can_SetBaudrate state transitions
 MCAL-5956 - SWS_Can_00062 : Can_SetBaudrate Requirements PUBLISHED	SWS_Can_00062 : Can_SetBaudrate Requirements



Design Identifier	Description
 MCAL-5946 - SWS_CAN_00494 : Can_SetBaudrate DET CAN_E_PARAM_CONTROLLER PUBLISHED	SWS_CAN_00494 : Can_SetBaudrate DET CAN_E_PARAM_CONTROLLER
 MCAL-5925 - SWS_CAN_00493 : Can_SetBaudrate DET CAN_E_PARAM_BAUDRATE PUBLISHED	SWS_CAN_00493 : Can_SetBaudrate DET CAN_E_PARAM_BAUDRATE
 MCAL-6175 - ECUC_Can_00481 : CanControllerFdBaudRate PUBLISHED	ECUC_Can_00481 : CanControllerFdBaudRate
 MCAL-6160 - ECUC_Can_00482 : CanSetBaudrateApi PUBLISHED	ECUC_Can_00482 : CanSetBaudrateApi
 MCAL-6051 - ECUC_Can_00471 : CanControllerBaudRateConfigID PUBLISHED	ECUC_Can_00471 : CanControllerBaudRateConfigID
 MCAL-6037 - ECUC_Can_00005 : CanControllerBaudRate PUBLISHED	ECUC_Can_00005 : CanControllerBaudRate
 MCAL-5957 - ECUC_Can_00473 : CanControllerFdBaudrateConfig PUBLISHED	ECUC_Can_00473 : CanControllerFdBaudrateConfig

Design Identifier	Description
 MCAL-5931 - ECUC_Can_00435 : CanControllerDefaultBaudrate PUBLISHED	ECUC_Can_00435 : CanControllerDefaultBaudrate
 MCAL-5905 - ECUC_Can_00387 : CanControllerBaudrateConfig PUBLISHED	ECUC_Can_00387 : CanControllerBaudrateConfig

6.1.13 Can_GetControllerMode


Refer section 8.3.1 [Reference 1 - AUTOSAR 4.3.1](#).

Design Identifier	Description
 MCAL-6180 - SWS_Can_91017 : Can_GetControllerMode DET : CAN_E_PARAM_CONTROLLER PUBLISHED	SWS_Can_91017 : Can_GetControllerMode DET : CAN_E_PARAM_CONTROLLER
 MCAL-6139 - SWS_Can_91015 : Can_GetControllerMode Return Controller Mode PUBLISHED	SWS_Can_91015 : Can_GetControllerMode Return Controller Mode

Design Identifier	Description
 MCAL-6024 - SWS_Can_91018 : Can_GetControllerMode DET : CAN_E_PARAM_POINTER PUBLISHED	SWS_Can_91018 : Can_GetControllerMode DET : CAN_E_PARAM_POINTER
 MCAL-5973 - SWS_Can_91016 : Can_GetControllerMode DET : CAN_E_UNINIT PUBLISHED	SWS_Can_91016 : Can_GetControllerMode DET : CAN_E_UNINIT






6.1.14 Can_SetControllerMode

Refer section 8.3.1 [Reference 1 - AUTOSAR 4.3.1](#).

 MCAL-6137 - SWS_Can_00198 : Can_SetControllerMode DET CAN_E_UNINIT PUBLISHED	SWS_Can_00198 : Can_SetControllerMode DET CAN_E_UNINIT
Design Identifier	Description

6.1.15 Can_GetControllerErrorState

Refer section 8.3.1 [Reference 1 - AUTOSAR 4.3.1](#).

Design Identifier	Description
 MCAL-6141 - SWS_Can_91007 : Can_GetControllerErrorState DET : CAN_E_PARAM_POINTER PUBLISHED	SWS_Can_91007 : Can_GetControllerErrorState DET : CAN_E_PARAM_POINTER
 MCAL-6121 - SWS_Can_91008 : Can_GetControllerErrorState Read Error State PUBLISHED	SWS_Can_91008 : Can_GetControllerErrorState Read Error State
 MCAL-6103 - SWS_Can_91006 : Can_GetControllerErrorState DET : CAN_E_PARAM_CONTROLLER PUBLISHED	SWS_Can_91006 : Can_GetControllerErrorState DET : CAN_E_PARAM_CONTROLLER
 MCAL-6058 - SWS_Can_91005 : Can_GetControllerErrorState DET : CAN_E_UNINIT PUBLISHED	SWS_Can_91005 : Can_GetControllerErrorState DET : CAN_E_UNINIT
 MCAL-6099 - SWS_Can_91003 : Can_ErrorStateType PUBLISHED	SWS_Can_91003 : Can_ErrorStateType

6.1.16 Can_DeInit

Refer section 8.3.1 [Reference 1 - AUTOSAR 4.3.1](#).




Design Identifier	Description
 MCAL-6124 - SWS_Can_91012 : Can_DeInit DET : Can Controller Started PUBLISHED	SWS_Can_91012 : Can_DeInit DET : Can Controller Started
 MCAL-6055 - SWS_Can_91011 : Can_DeInit DET : Can Driver not ready PUBLISHED	SWS_Can_91011 : Can_DeInit DET : Can Driver not ready

6.1.17 Can_RegisterReadback

As noted from previous implementation, the can configuration registers could potentially be corrupted by other entities (s/w or h/w). One of the recommended detection methods would be to periodically read-back the configuration and confirm configuration is consistent. The service API defined below shall be implemented to enable this detection.

	<i>Description</i>	<i>Comments</i>
--	--------------------	-----------------


Service Name	Can_RegisterReadback	Can potentially be turned OFF, refer (NON Standard configurable parameters)
Syntax	Std_ReturnType Can_RegisterReadback(uint8 Controller, Can_RegisterReadbackType *RegRbPtr)	Can_RegisterReadbackType defines the type, that holds critical values
Service ID	NA	
Sync / Async	Sync	
Reentrancy	Non Reentrant	
Parameter in	Controller	Identifies a unique valid controller
Parameters out	RegRbPtr	Pointer of type Can_RegisterReadbackType , which holds read values
Return Value	Std_ReturnType	E_OK or E_NOT_OK in case of invalid channel id






Design Identifier	Description
 MCAL-6028 - CAN: Safety Diagnostics: MCAN12: Software Readback of written configuration PUBLISHED	CAN: Safety Diagnostics: MCAN12: Software Readback of written configuration.
 MCAL-6088 - CAN: Safety Diagnostics: MCAN14: Periodic software readback of static configuration registers. PUBLISHED	CAN: Safety Diagnostics: MCAN14: Periodic software readback of static configuration registers.
 MCAL-6149 - CanRegisterReadbackAPI PUBLISHED	CanRegisterReadbackAPI

6.1.18 Can_EnableIntr

This API Enables CRC,BIT-0,BIT-1,ACK,FORM,STUFF,TOO Errors.


	Description	Comments
Service Name	Can_EnableIntr	Can_EnableIntr Enables CRC,BIT-0,BIT-1,ACK,FORM,STUFF,TOO Errors.






	Description	Comments
Syntax	Std_ReturnType Can_EnableIntr(uint8 Controller,uint8 CanErrVar)	
Service ID	NA	
Sync / Async	Sync	
Reentrancy	Reentrant	
Parameter in	Controller	Controller for which errors has to be enabled
Parameters out	NONE	
Return Value	Std_ReturnType	E_OK or E_NOT_OK in case of Enable Interrupt Status
Design Identifier		Description
 MCAL-6126 - CAN: Safety Diagnostics: MCAN9: Bit Error Detection PUBLISHED		CAN: Safety Diagnostics: MCAN9: Bit Error Detection

Design Identifier	Description
 MCAL-5909 - CAN: Safety Diagnostics: MCAN7: Form Error Detection PUBLISHED	CAN: Safety Diagnostics: MCAN7: Form Error Detection
 MCAL-5917 - CAN: Safety Diagnostics: MCAN8: Acknowledge Error Detection PUBLISHED	CAN: Safety Diagnostics: MCAN8: Acknowledge Error Detection
 MCAL-5922 - CAN: Safety Diagnostics: MCAN4: Timeout on FIFO activity PUBLISHED	CAN: Safety Diagnostics: MCAN4: Timeout on FIFO activity
 MCAL-6052 - CAN: Safety Diagnostics: MCAN6: Stuff Error Detection PUBLISHED	CAN: Safety Diagnostics: MCAN6: Stuff Error Detection
 MCAL-6091 - CAN: Safety Diagnostics: MCAN2: CRC in Message PUBLISHED	CAN: Safety Diagnostics: MCAN2: CRC in Message

6.1.19 Can_DisableIntr


This API Disables Interrupt for CRC,BIT-0,BIT-1,ACK,FORM,STUFF,TOO Errors.






	Description	Comments
Service Name	Can_DisableIntr	Can_DisableIntr Disables Interrupt for CRC,BIT-0,BIT-1,ACK,FORM,STUFF,TOO Errors.
Syntax	Std_ReturnType Can_DisableIntr(uint8 Controller,uint8 CanErrVar)	
Service ID	NA	
Sync / Async	Sync	
Reentrancy	Reentrant	
Parameter in	Controller	Controller for which interrupt errors has to be Disabled.
Parameters out	NONE	
Return Value	Std_ReturnType	E_OK or E_NOT_OK in case of Disable Interrupt Status
Design Identifier		Description
 MCAL-6126 - CAN: Safety Diagnostics: MCAN9: Bit Error Detection PUBLISHED		CAN: Safety Diagnostics: MCAN9: Bit Error Detection

Design Identifier	Description
 MCAL-5909 - CAN: Safety Diagnostics: MCAN7: Form Error Detection PUBLISHED	CAN: Safety Diagnostics: MCAN7: Form Error Detection
 MCAL-5917 - CAN: Safety Diagnostics: MCAN8: Acknowledge Error Detection PUBLISHED	CAN: Safety Diagnostics: MCAN8: Acknowledge Error Detection
 MCAL-5922 - CAN: Safety Diagnostics: MCAN4: Timeout on FIFO activity PUBLISHED	CAN: Safety Diagnostics: MCAN4: Timeout on FIFO activity
 MCAL-6052 - CAN: Safety Diagnostics: MCAN6: Stuff Error Detection PUBLISHED	CAN: Safety Diagnostics: MCAN6: Stuff Error Detection
 MCAL-6091 - CAN: Safety Diagnostics: MCAN2: CRC in Message PUBLISHED	CAN: Safety Diagnostics: MCAN2: CRC in Message

6.1.20 Can_GetIntrStatus


This API gets the interrupt status for CRC,BIT-0,BIT-1,ACK,FORM,STUFF,TOO Errors.






	Description	Comments
Service Name	Can_GetIntrStatus	Can_GetIntrStatus gets the interrupt status for CRC,BIT-0,BIT-1,ACK,FORM,STUFF,TOO Errors.
Syntax	Can_IrqStatusType Can_GetIntrStatus(uint8 Controller)	
Service ID	NA	
Sync / Async	Sync	
Reentrancy	Reentrant	
Parameter in	Controller	Controller for which it reads the interrupt error status
Parameters out	NONE	
Return Value	Std_ReturnType	E_OK or E_NOT_OK in case of Get Interrupt Status
Design Identifier		Description
 MCAL-6126 - CAN: Safety Diagnostics: MCAN9: Bit Error Detection PUBLISHED		CAN: Safety Diagnostics: MCAN9: Bit Error Detection

Design Identifier	Description
 MCAL-5909 - CAN: Safety Diagnostics: MCAN7: Form Error Detection PUBLISHED	CAN: Safety Diagnostics: MCAN7: Form Error Detection
 MCAL-5917 - CAN: Safety Diagnostics: MCAN8: Acknowledge Error Detection PUBLISHED	CAN: Safety Diagnostics: MCAN8: Acknowledge Error Detection
 MCAL-5922 - CAN: Safety Diagnostics: MCAN4: Timeout on FIFO activity PUBLISHED	CAN: Safety Diagnostics: MCAN4: Timeout on FIFO activity
 MCAL-6052 - CAN: Safety Diagnostics: MCAN6: Stuff Error Detection PUBLISHED	CAN: Safety Diagnostics: MCAN6: Stuff Error Detection
 MCAL-6091 - CAN: Safety Diagnostics: MCAN2: CRC in Message PUBLISHED	CAN: Safety Diagnostics: MCAN2: CRC in Message

6.1.21 Can_ClearIntrStatus

This API clears the interrupt status for CRC,BIT-0,BIT-1,ACK,FORM,STUFF,TOO Errors.

	<i>Description</i>	<i>Comments</i>
Service Name	Can_ClearIntrStatus	Can_ClearIntrStatus clears the interrupt status for CRC,BIT-0,BIT-1,ACK,FORM,STUFF,TOO Errors.
Syntax	Std_ReturnType Can_ClearIntrStatus(uint8 Controller)	
Service ID	NA	
Sync / Async	Sync	
Reentrancy	Reentrant	
Parameter in	Controller	Controller for which it clears the interrupt error status.
Parameters out	NONE	
Return Value	Std_ReturnType	E_OK or E_NOT_OK in case of Clear Interrupt Status
Design Identifier		Description
 MCAL-6126 - CAN: Safety Diagnostics: MCAN9: Bit Error Detection PUBLISHED		CAN: Safety Diagnostics: MCAN9: Bit Error Detection

Design Identifier	Description
 MCAL-5909 - CAN: Safety Diagnostics: MCAN7: Form Error Detection PUBLISHED	CAN: Safety Diagnostics: MCAN7: Form Error Detection
 MCAL-5917 - CAN: Safety Diagnostics: MCAN8: Acknowledge Error Detection PUBLISHED	CAN: Safety Diagnostics: MCAN8: Acknowledge Error Detection
 MCAL-5922 - CAN: Safety Diagnostics: MCAN4: Timeout on FIFO activity PUBLISHED	CAN: Safety Diagnostics: MCAN4: Timeout on FIFO activity
 MCAL-6052 - CAN: Safety Diagnostics: MCAN6: Stuff Error Detection PUBLISHED	CAN: Safety Diagnostics: MCAN6: Stuff Error Detection
 MCAL-6091 - CAN: Safety Diagnostics: MCAN2: CRC in Message PUBLISHED	CAN: Safety Diagnostics: MCAN2: CRC in Message



7 Performance Objectives

7.1 Resource Consumption Objectives

ROM - Program(KB)	ROM - Data(KB)	RAM - Program(KB)	RAM - Data(KB)	EEPROM (KB)	% CPU Utilization
30	NA	4	NA	NA	NA

7.2 Critical timing and Performance

Not Applicable



8 Decision Analysis & Resolution (DAR)

Sections below list some of the important design decisions and rational behind those decision.

8.1 CAN Dma mode

The Can Receive message can be processed either through CPU or via DMA. The method chosen will impact receive throughput.

No.	Decision Criteria	Alternatives	Selected alternative	Rationale	Trade-offs
1	Minimal restrictions on the system and guaranteed “no receive message drop” in the system.	<ul style="list-style-type: none"> • DMA Mode DMA Mode – The can controller will generate a DMA events to the system EDMA.CAN message will be copied from CAN mailbox to destination address by DMA • Advantages: <ul style="list-style-type: none"> • CPU loading is low and constant irrespective of the number of CAN messages received. • Less probability of mailbox overflow as the DMA copy happens without CPU intervention • Disadvantages: 	<ul style="list-style-type: none"> • DMA Mode DMA Mode – The can controller will generate a DMA events to the system EDMA.CAN message will be copied from CAN mailbox to destination address by DMA 	In case of ADAS use case, the CPU loading is low and there is no chance of CAN mailbox overflow in case Thus in all respect (complexity, efficiency), CPU mode is sufficient for the ADAS use case. So it is recommended to employ cpu mode.	<ul style="list-style-type: none"> • Complexity involved in designing the EDMA parameters. • Cache coherency needs to be taken care. This will result in Cache module dependency in driver or in the AUTOSAR stack • Need of a common DMA complex driver with resource management as the EDMA is at system level and is common across SoC

No.	Decision Criteria	Alternatives	Selected alternative	Rationale	Trade-offs
		<ul style="list-style-type: none"> Complexity involved in designing the EDMA parameters. Cache coherency needs to be taken care. This will result in Cache module dependency in driver or in the AUTOSAR stack Need of a common DMA complex driver with resource management as the EDMA is at system level and is common across SoC CPU Mode The Can controller will raise an interrupt. The CPU needs to copy the message and invoke the CanIfRxIndication callback. Advantages: 			

No.	Decision Criteria	Alternatives	Selected alternative	Rationale	Trade-offs
		<ul style="list-style-type: none"> • Simple implementation • No cache coherency is needed and no dependency on cache APIs • Disadvantages: • CPU load is function of rate of CAN messages. • High probability of mailbox overflow during high receive message rate as the CPU is involved in reading the FIFO 			



8.2 MCAN Tx Buffer Mode





Along with dedicated Tx Buffers, MCAN also supports Tx FIFO/Queue. This buffer mode is configurable and can be used in one those configurations. The mode selected will affect the priority in which will messages will go out on bus. Support of any of these modes is necessary in order to support multiplexed transmission.

No.	Decision Criteria	Alternatives	Selected alternative	Rationale	Trade-offs
1	Minimal restrictions on the system and support multiplexed transmission in order to avoid priority inversion.	<ul style="list-style-type: none"> • FIFO Mode In this mode, messages are stored into memory in First In First Out(FIFO) manner <ul style="list-style-type: none"> • Advantages: <ul style="list-style-type: none"> • Less software overhead as FIFO management is done by MCAN controller • Messages are sent in the order in which they are being stored into FIFO • Disadvantages: 	<ul style="list-style-type: none"> • FIFO Mode In this mode, messages are stored into memory in First In First Out(FIFO) manner 	In case of ADAS use case, the CPU loading is low and priority inversion cannot occur. Thus in all respect (complexity, efficiency), Queue mode is recommended.	<ul style="list-style-type: none"> • Since messages are being sent in the orders which are stored into FIFO, priority inversion can happen if message with higher priority are stored at later location in the FIFO. • Messages should be carefully written into FIFO in order to avoid priority inversion.

No.	Decision Criteria	Alternatives	Selected alternative	Rationale	Trade-offs
		<ul style="list-style-type: none"> Since messages are being sent in the orders which are stored into FIFO, priority inversion can happen if message with higher priority are stored at later location in the FIFO. 			

No.	Decision Criteria	Alternatives	Selected alternative	Rationale	Trade-offs
		<ul style="list-style-type: none"> Messages should be carefully written into FIFO in order to avoid priority inversion. QUEUE Mode In this mode, messages will be stored into first free location in the memory allocated for Queue <ul style="list-style-type: none"> Advantages: <ul style="list-style-type: none"> Messages are sent in the order of their priority hence priority inversion will not happen. 			

No.	Decision Criteria	Alternatives	Selected alternative	Rationale	Trade-offs
		<ul style="list-style-type: none"> • Can be treated as Tx Buffer. • Disadvantages: <ul style="list-style-type: none"> • Messages are written into first free location in Tx Queue, hence leads more software overhead. 			
Design ID			Description		
 MCAL-6074 - SWS_Can_00277 : Priority Inversion PUBLISHED			SWS_Can_00277 : Priority Inversion		
 MCAL-6033 - SWS_Can_00403 : Priority Inversion PUBLISHED			SWS_Can_00403 : Priority Inversion		






Design ID	Description
 MCAL-5959 - SWS_Can_00401 : Priority Inversion PUBLISHED	SWS_Can_00401 : Priority Inversion
 MCAL-5923 - SWS_Can_00402 : Priority Inversion PUBLISHED	SWS_Can_00402 : Priority Inversion
 MCAL-6005 - SWS_Can_00234 : Mandatory Callback Interface PUBLISHED	SWS_Can_00234 : Mandatory Callback Interface
 MCAL-5976 - ECUC_Can_00095 : CanMultiplexedTransmission PUBLISHED	ECUC_Can_00095 : CanMultiplexedTransmission

9 Test Criteria





The sections below identify some of the aspects of design that would require emphasis during testing of this design implementation

- **Internal Loopback** : Internal loopback could be used for enhancements (or development)
 - Configure CAN controller to operate in internal loopback mode and check for transmission and reception of the message.
- **Board to Board Loopback**
 - Connect two CAN controllers through CAN bus. Send message from first controller and check for reception of the same message on the other controller.
- **Multiplexed Transmission**
 - Configure message RAM in mixed configuration i.e. to have both buffers as well as FIFO in case of DCAN or Queue in case MCAN. Check if message is getting properly configured as well as check for Multiplexed transmission (priority inversion).
- **Different Baud-Rates**
 - Check for transmission and reception at different baud rates.
- **Inter-packet delay**
 - Include tests cases where inter-packet delay is 0 (i.e. ST_MIN = 0)

10 Template Revision History

Author Name	Description	Version	Date
Yaniv Machani	Initial version	0.1	 03 Oct 2018
Yaniv Machani	Updated to include EP views	0.4	 02 Nov 2018
Yaniv Weizman	Restructuring and editing to further meet the A-SPICE and EP requirements	0.5	 27 Dec 2018
Yaniv Weizman	Adding link to Architecture review template	0.6	 22 Oct 2019
Yaniv Weizman	Adding requirement type column for requirements table (Functional/Non-Functional). Adding DAR table	0.65	 13 Nov 2019



Author Name	Description	Version	Date
Yaniv Weizman	Adding tables for Testing guidelines	0.7	 18 Nov 2019
Krishna	Updated based on ASPICE requirements	0.8	 20 Aug 2020
Krishna	Updated based on the feedback from Jon N	0.9	 09 Oct 2020
Krishna	Updated the traceability scheme	1.0	 17 Dec 2020