Application Note

Microcontroller Abstraction Layer on Jacinto™ and Sitara™ Embedded Processors

Harshit Gupta, Nihar Potturu, Tarun Mukesh Puvvada

EP ASM Automotive

ABSTRACT

AUTOSAR® is a standardized approach to develop automobile software. Microcontroller Abstraction Layer (MCAL) comes under the AUTOSAR Classic Platform specifications. MCALs provide an interface to hardware components by bundling several categories of drivers with them that can manage the respective set of components and offer several supported functionalities. An MCAL package is delivered specific to a target package, that is, MCALs being hardware dependent are offered for each of the supported device families such as Sitara™ AM263x, AM263Px, AM273x MCUs, AM62x, AM62Ax, AM62Px MPUs, Jacinto™ DRA8x, and TDA4x processors.

Additional device peripherals and hardware units which are not standardized by AUTOSAR are delivered as an extended integration using Complex Driver Design (CDD). Though the pack is promoted by the term *MCAL* package, the bundle includes drivers from both MCAL and CDD layers. Any dependency requirement from the upper layers are provided in the package as a BSW Stub. A BSW Stub is an incomplete implementation of an upper layer component that is used to interface the driver layers for a standalone functional package delivery which gets replaced by the complete components in a full AUTOSAR stack offered by leading AUTOSAR vendors.

Table of Contents

List of Figures

List of Tables

Trademarks

Sitara™ and Jacinto™ are trademarks of Texas Instruments.
AUTOSAR® is a registered trademark of AUTOSAR GbR.
Arm® and Cortex® are registered trademarks of Arm Limited.
tresos® is a registered trademark of Elektrobit Automotive GmbH.
Microsoft® and Windows® are registered trademarks of Microsoft Corporation.
Unix® is a registered trademark of The Open Group Limited.
All trademarks are the property of their respective owners.

1 Introduction

Table 1-1 shows acronyms and definitions.

Table 1-1. Full Form Expansions for Abbreviations and Acronyms

Acronym	Definition
API	Application Programming Interface
ARXML	AUTOSAR XML
AUTOSAR	AUTomotive Open System ARchitecture
BSW	Basic Software
CDD	Complex Driver Design
CSP	Compliance Support Package
ECU	Electronic Control Unit
FSQ	Functional Safety Qualification
FuSa	Functional Safety
GUI	Graphical User Interface
ISO	International Organization for Standardization
IRQ	Interrupt ReQuest
ISR	Interrupt Service Routine
MCAL	Microcontroller Abstraction Layer
MCU	Microcontroller Unit
MPU	Microprocessor Unit
Mip	Module Implementation Prefix
RTE	Runtime Environment
SIL	Silicon Integrity Level
SDK	Software Development Kit
SRS	Software Requirement Specification
SWS	Software Specification
TI	Texas Instruments Incorporated
XER5F	eXEcutable for R5F core

MCAL is the same as the Hardware Abstraction Layer that is responsible for providing ways to interact with the underlying hardware system comprising a processor along with memory and the peripherals through the set of drivers based on the documented interface. MCALs are one of the layers under the AUTOSAR stack.

AUTOSAR is an open standardized software framework for *Automotive Development*, established by a group of leading automotive industrialists in 2003. The latest release is R23-11; succeeding releases R22-11, R21-11, R20-11, R19-11, ..., R4.4.0, R4.3.1, R4.3, R4.2, R4.1, R4.0.3, R3.2, R3.1, R3, and R2 that were published in 2006. Versions were numbered as R<*YY*>-<*MM*> where <MM> is month and <YY> is year of release in two digits, starting with R17-03 and prior releases were semantically versioned as v<*Major*>.<*minor*>.<*Revision*> until v4.4.0 and 4.5.0. The standards organization provides specifications under the *Classic Platform* and Adaptive Platform architecture, out of which the Classic Platform architecture standards include MCAL under the Basic Software (BSW) layer.

Figure 1-1 shows the Classic Platform standard layer architecture is comprised of an Application Layer, Runtime Environment, BSW Layer, and a Microcontroller layer just below. The scope of the Classic Platform architecture is from Application to µC.

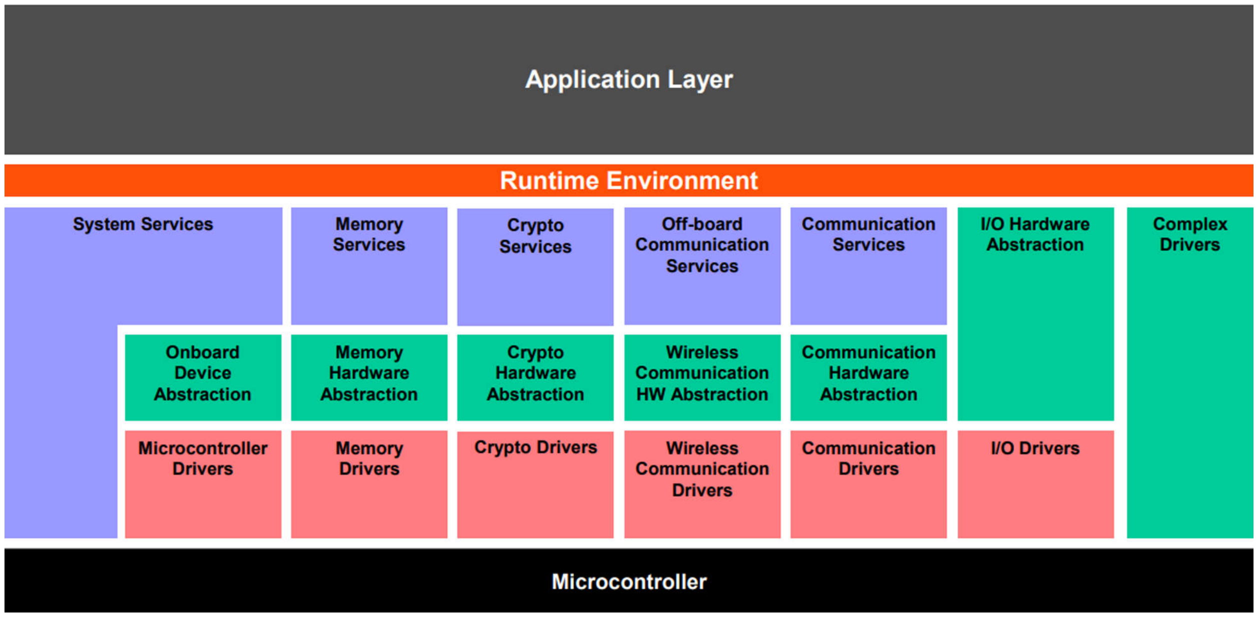


Figure 1-1. AUTOSAR 4.3.1 Architecture

The BSW Layer sandwiched between *Runtime Environment* and *Microcontroller* has the sublayers, specifically, Services Layer, ECU Abstraction Layer, and MCAL Layer colored distinctly in Purple, Green, and Red tint, respectively. Though the *Complex Drivers* block is not *distinctly* colored, the block is another sublayer of BSW that spans entirely within BSW from the microcontroller to the RTE interaction. Figure 1-1 shows the breakdown for MCAL and Complex Drivers.

The MCAL package is meant to be run on Arm® Cortex®-R core fabricated by Texas Instruments on the multiple cores. For TI Jacinto™, Sitara™ MCU, and MPU Arm®-based device families, the Cortex-R5 core is the target core for supporting execution of the MCAL and CDD device drivers. At present, only single-core MCAL support is offered from TI on these device families.

1.1 Configurator

Several settings for any driver can easily be done by using a graphical interface available with any of the AUTOSAR-compliant configuration tools. Several configurator tools are available in market such as Elektrobit (EB) tresos® Studio, Vector DaVinci Configurator, KPIT K-SAR, and so on.

Configuration plug-in in MCALs bundled for TI Sitara MCU, MPU and Jacinto processors are developed and tested on the EB tresos Studio configurator tool offered by Elektrobit. Elektrobit tresos Studio supports some additional features for the configurator plug-ins that can be leveraged on the tool while not providing direct cross-tool usability. This accounts for features such as template-based code generation and XPath (short for Extensible Stylesheet Language Transformation Path) for getting nodes, which are not common among all AUTOSAR-compliant Configurator Tools.

EB tresos Studio is available with a Desktop GUI application on Microsoft® Windows® and can be used on Unix®-based systems as a CLI with `<EB_Installation_Path>/bin/tresos_cmd.sh`.

2 Software Stack

The MCAL package is comprised of a configurator plug-in, driver source files, example applications for each driver, and documents including user guide, release notes, and Compliance Support Package (CSP) documents for Functional Safety Compliance as per the ISO26262 standard.

The directory structure is as follows:

- `mcal_config/` - Contains the AUTOSAR Configurator Tool compliant Configurator Plug-in that can be tested to load into Elektrobit (EB) tresos Studio configurator tool. Each plug-in corresponds to a particular device variant for each driver and is stored in the respective folder. Plug-ins can either be loaded individually for just the data configuration or be loaded with other dependent plug-ins in a single configurator project to enable referencing of the fields and nodes from other plug-ins.
- `mcal/` - Contains all the source files for MCALs across all driver categories bundled with a package. This directory typically includes around 10 to 25 drivers which can vary based on the supported features of the target device hardware and the device drivers planned for a release.
  - `mcal/examples/` - This path within the `mcal` directory stores the example applications for each of the driver sources that demonstrate the initialization flow and use of various APIs available from a driver module.
- `mcal_docs/` - Contains the documentation including release notes, manifest, and module user guides. The release notes mention the `changelog` with a previous release with new additions and bugs encountered or fixed, and workarounds for known issues and recommendations. User guides contain details on driver features, configurator plug-ins, and APIs.
- `build/` - Includes package-level `makefile` as well as other related files for build tool related settings. Source build is done using the make command in this path.

Specifically, in Sitara MPU MCAL, folders except for the previously-referenced `build/` are found under the `mcal_drv/` folder. For Jacinto MCAL, request and download the previously explained `mcal_config` folder from PSDK-RTOS-AUTO.

2.1 Configurator Plug-in

The *Configurator Plug-in* provides configuration options of module-related parameters which can be edited and generated using a Configurator Tool to get a GUI for the available module fields. Any AUTOSAR-compliant configurator tool can utilize module plug-in files to load the configurable fields from an ARXML file (specifically, the schema ARXML file loads the model, data ARXML can provide pre- and recommended configuration values loadable for each of the fields in schema) and generate the corresponding *include* and *source* files.

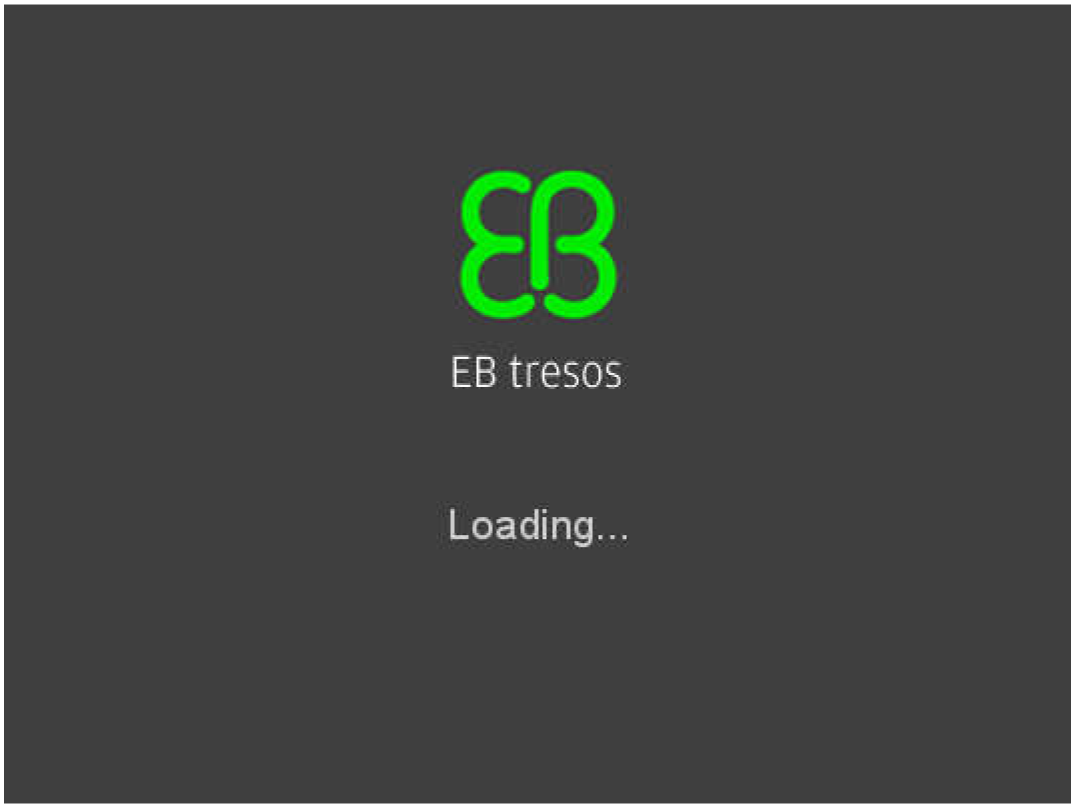
Configurator Plug-in for EB tresos Studio has the following file contents:

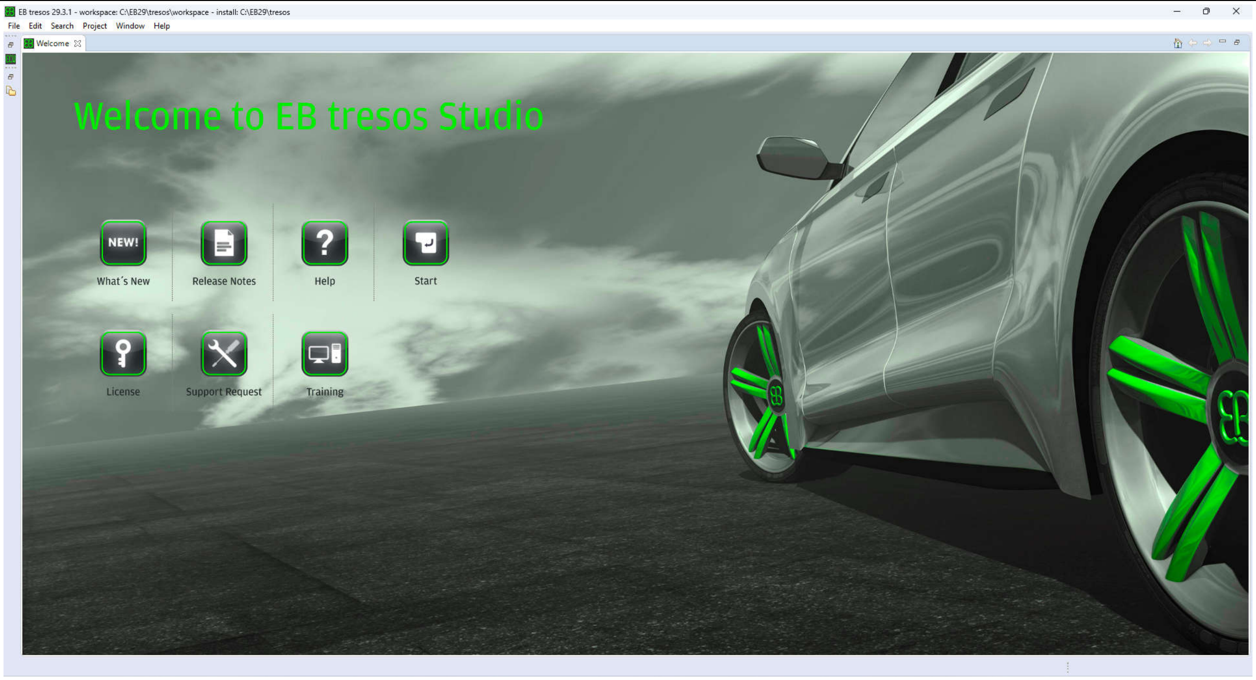
- `config/`
  - `<Module>.xdm` - XML Data Format file which can use XPath and expression type validations supported by EB tresos.
  - `<Module>.arxml` - Standard AUTOSAR XML file which can be used to load the plug-in schema in any AUTOSAR-compliant configurator tool.
- `config_ext/`
  - `<Module>_Rec.xdm` - Recommended XDM Data file that can load the recommended field values for many of the available options in an XDM.
- `generate/`
  - `include/`
    - `<Module>_Cfg.h` - Module Configuration Header file generation template.
  - `src/`
    - `<Module>_Cfg.c` - Module Configuration Source file generation template for Pre-compile configuration variant.
    - `<Module>_Lcfg.c` - Module Configuration Source file generation template for Link-time configuration variant.
    - `<Module>_PBcfg.c` - Module Configuration Source file generation template for Post-Build configuration variant.

  - V<*Number*> - Version-specific folder, where each version relates to a driver hardware variant. These contain any hardware-bound header and source files that totally fit with the hardware variant.
- `generate_swcd/`
  - `swcd/`
    - `<Module>_Bswmd.arxml` - BSW Module Description file for a *plug-in*.
- `META-INF/`
  - `MANIFEST.MF` - Java Manifest file used to describe the plug-in.
  - `CRYPTOMANIFEST.MF` and `CRYPTOMANIFESTSIG.MF` - Signed *plug-in* files to sign each file of a plug-in (with the exception of these files) and contain encrypted plug-in source information for use with a User License.
- `plugin.xml` - Plug-in XML file for several plug-in specific settings.

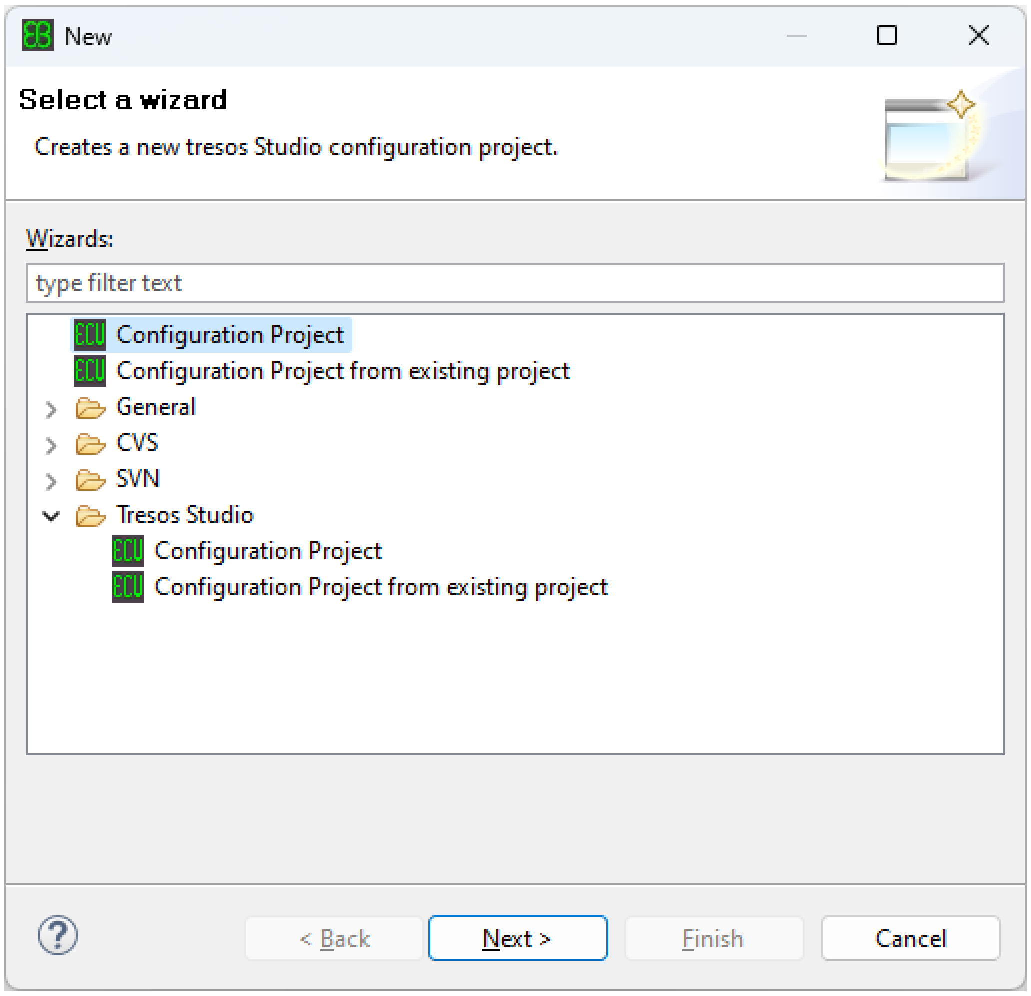
To use a configurator plug-in, follow these steps:

1. Paste the plug-in folders to be loaded under the `plugins/` folder in the EB tresos installation path, generally `C:/EB/tresos/` on a Microsoft® Windows® system. The plug-ins can be acquired for your specific device from TI through the mechanism described in the MCAL software documentation. Do not run EB tresos while the plug-ins are being copied, or else the software can fail to be recognized and EB tresos must be closed and re-launched.
2. Launch the EB tresos Studio application:

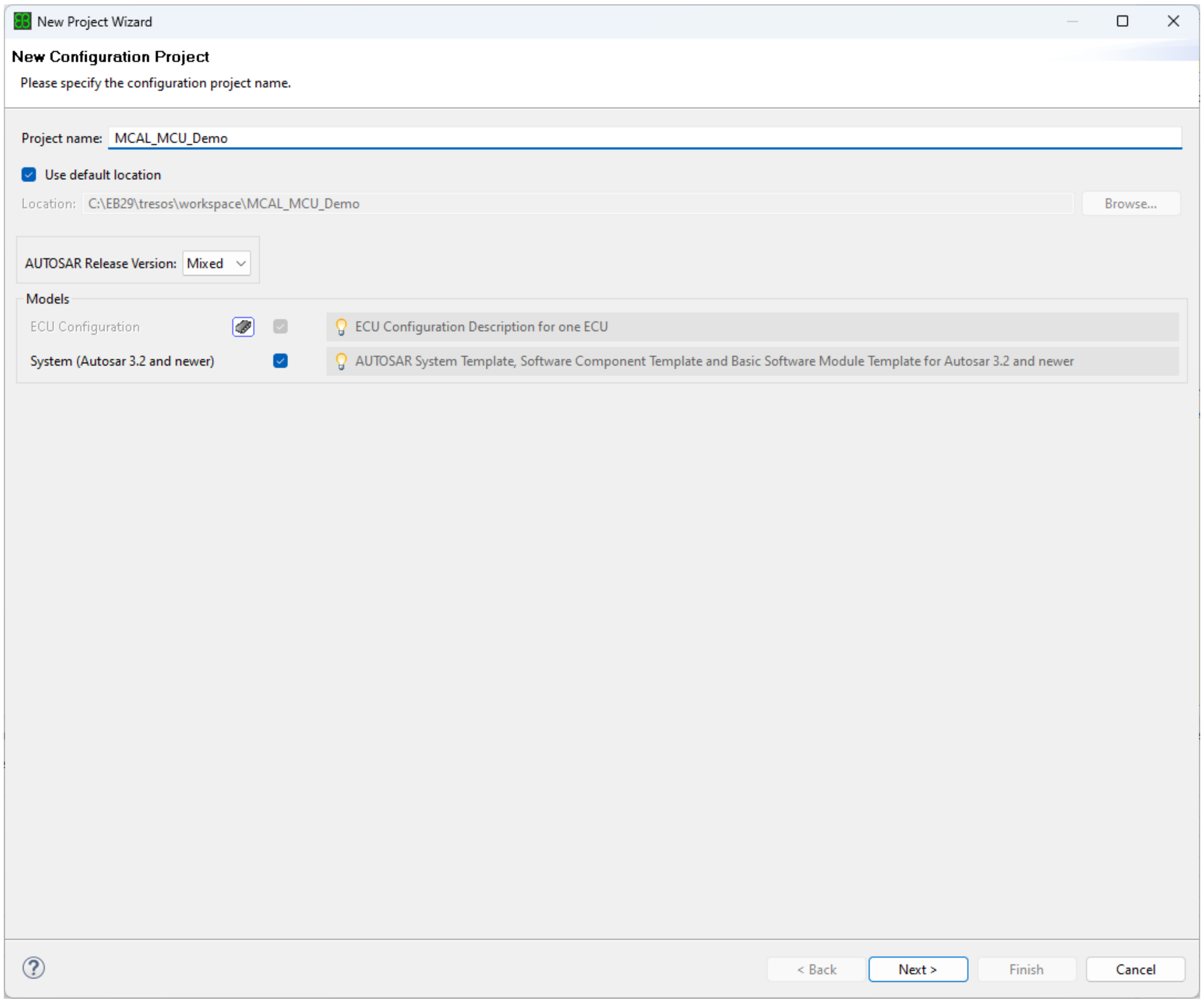



EB tresos 29.3.1 - workspace: C:\EB29\tresos\workspace - install: C:\EB29\tresos
File Edit Search Project Window Help
Welcome
Welcome to EB tresos Studio
What´s New
Release Notes
Help
Start
License
Support Request
Training

3. Create a new *Configuration Project* from *File* → *New* → *Configuration Project*. Set useful options from the *New Project Wizard*.


New
Select a wizard
Creates a new tresos Studio configuration project.
Wizards:
type filter text
Configuration Project
Configuration Project from existing project
General
CVS
SVN
Tresos Studio
Configuration Project
Configuration Project from existing project
< Back
Next >
Finish
Cancel

a. Use any appropriate name for the project.



New Configuration Project

Please specify the configuration project name.

Project name: MCAL_MCU_Demo

[✓] Use default location

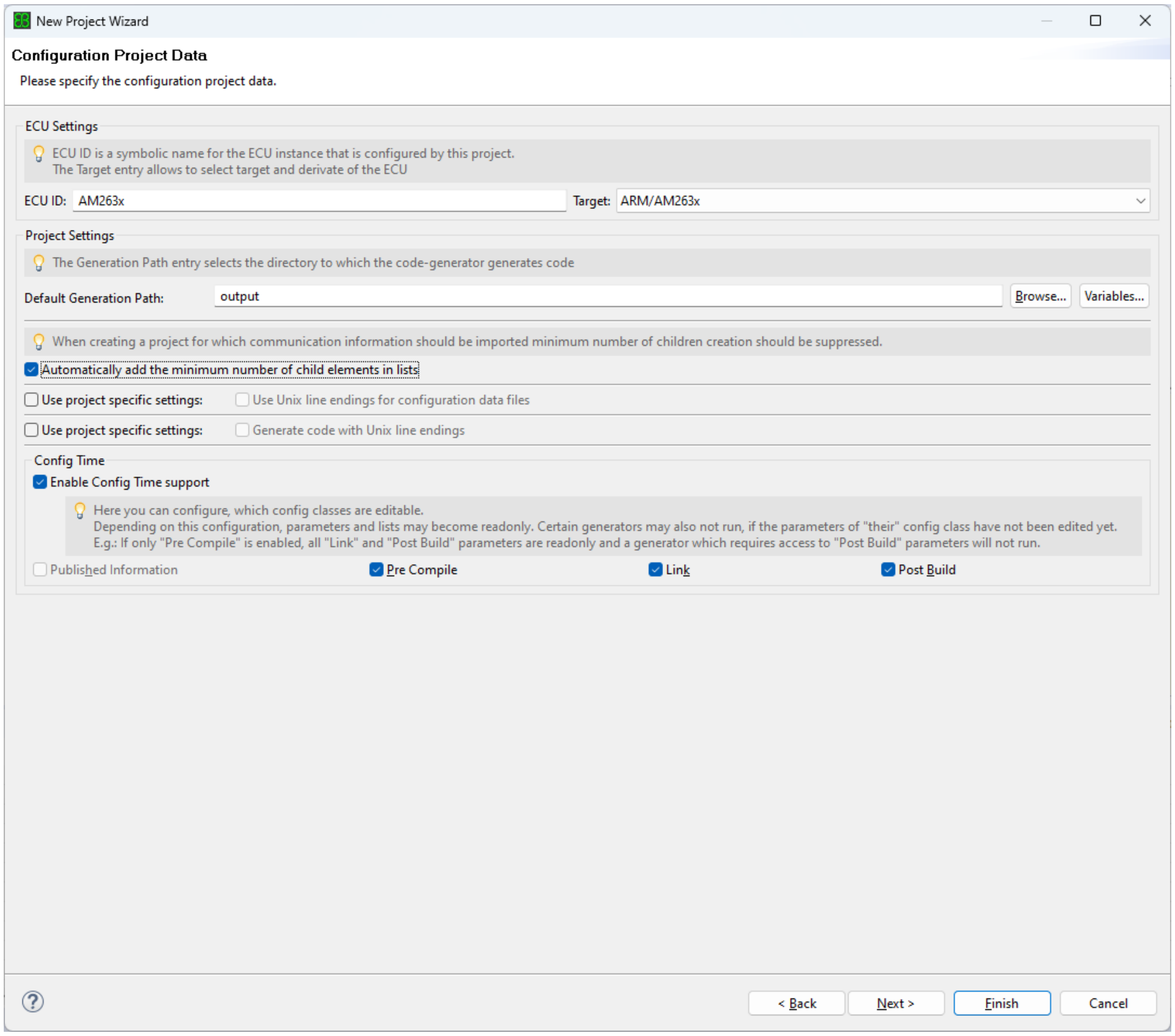
Location: C:\EB29\tresos\workspace\MCAL_MCU_Demo

AUTOSAR Release Version: Mixed

Models

ECU Configuration	✓	ECU Configuration Description for one ECU
System (Autosar 3.2 and newer)	✓	AUTOSAR System Template, Software Component Template and Basic Software Module Template for Autosar 3.2 and newer

b. Provide relevant ECU ID (users choice) and choose the right plug-in *Target* as ARM/AM2*xx* as matching with the AM2x MCU series. Seeing the *Target* drop-down populated with your device plug-in means the plug-in was recognized correctly by EB tresos.



Configuration Project Data

Please specify the configuration project data.

ECU Settings

ECU ID is a symbolic name for the ECU instance that is configured by this project.
The Target entry allows to select target and derivate of the ECU

ECU ID: AM263x Target: ARM/AM263x

Project Settings

The Generation Path entry selects the directory to which the code-generator generates code

Default Generation Path: output

When creating a project for which communication information should be imported minimum number of children creation should be suppressed.

[✓] Automatically add the minimum number of child elements in lists

[ ] Use project specific settings: [ ] Use Unix line endings for configuration data files

[ ] Use project specific settings: [ ] Generate code with Unix line endings

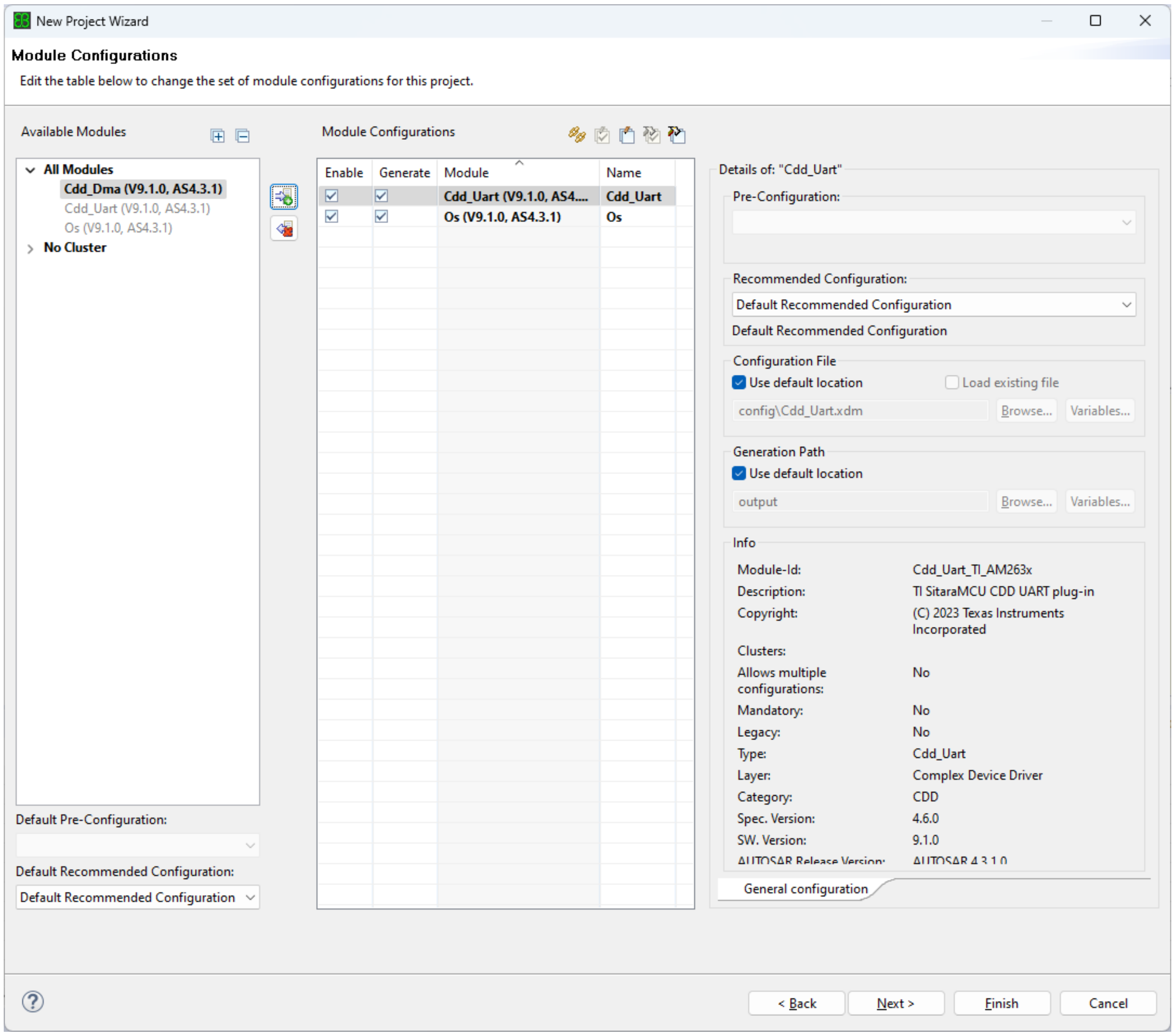
Config Time

[✓] Enable Config Time support

Here you can configure, which config classes are editable.
Depending on this configuration, parameters and lists may become readonly. Certain generators may also not run, if the parameters of "their" config class have not been edited yet.
E.g.: If only "Pre Compile" is enabled, all "Link" and "Post Build" parameters are readonly and a generator which requires access to "Post Build" parameters will not run.

[ ] Published Information [✓] Pre Compile [✓] Link [✓] Post Build

c. Add the right set of modules from the available set. Click the *Next* button if any *Importers* or *Exporters* must be used; otherwise, directly choose the *Finish* button.



New Project Wizard

Module Configurations

Edit the table below to change the set of module configurations for this project.

Available Modules

- All Modules
  - Cdd_Dma (V9.1.0, AS4.3.1)
  - Cdd_Uart (V9.1.0, AS4.3.1)
  - Os (V9.1.0, AS4.3.1)
- No Cluster

Default Pre-Configuration:

Default Recommended Configuration: Default Recommended Configuration

Module Configurations

Enable	Generate	Module	Name
✓	✓	Cdd_Uart (V9.1.0, AS4....	Cdd_Uart
✓	✓	Os (V9.1.0, AS4.3.1)	Os

Details of: "Cdd_Uart"

Pre-Configuration:

Recommended Configuration: Default Recommended Configuration

Default Recommended Configuration

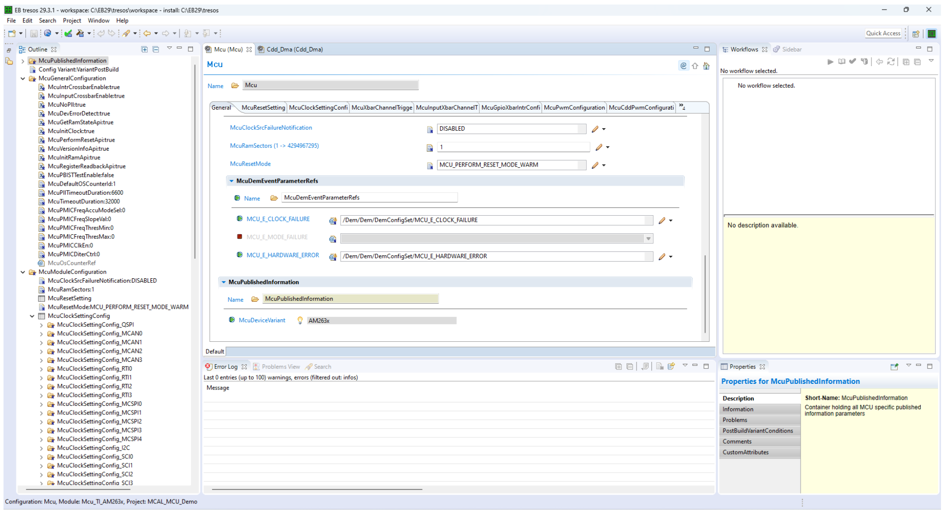
Configuration File
- ✓ Use default location
- Load existing file
- config\Cdd_Uart.xdm

Generation Path
- ✓ Use default location
- output

Info

Module-Id:	Cdd_Uart_TI_AM263x
Description:	TI SitaraMCU CDD UART plug-in
Copyright:	(C) 2023 Texas Instruments Incorporated
Clusters:	
Allows multiple configurations:	No
Mandatory:	No
Legacy:	No
Type:	Cdd_Uart
Layer:	Complex Device Driver
Category:	CDD
Spec. Version:	4.6.0
SW. Version:	9.1.0
AUTOSAR Release Version:	AUTOSAR 4.3.1.0

4. Open the generic editor for each module and find the available option fields to set and update the configuration.



EB tresos 29.3.1 - workspace: C:\EB29\tresos\workspace - install: C:\EB29\tresos

Outline

- McuPublishedInformation
- Config Variant:VariantPostBuild
- McuGeneralConfiguration
  - McuIntrCrossbarEnable:true
  - McuInputCrossbarEnable:true
  - McuNoPll:true
  - McuDevErrorDetect:true
  - McuGetRamStateApi:true
  - McuInitClock:true
  - McuPerformResetApi:true
  - McuVersionInfoApi:true
  - McuInitRamApi:true
  - McuRegisterReadbackApi:true
  - McuPBISTTestEnable:false
  - McuDefaultOSCounterId:1
  - McuPllTimeoutDuration:6600
  - McuTimeoutDuration:32000
  - McuPMICFreqAccuModeSel:0
  - McuPMICFreqSlopeVal:0
  - McuPMICFreqThresMin:0
  - McuPMICFreqThresMax:0
  - McuPMICClkEn:0
  - McuPMICDiterCtrl:0
  - McuOsCounterRef
- McuModuleConfiguration
  - McuClockSrcFailureNotification:DISABLED
  - McuRamSectors:1
  - McuResetSetting
  - McuResetMode:MCU_PERFORM_RESET_MODE_WARM
  - McuClockSettingConfig
    - McuClockSettingConfig_QSPI
    - McuClockSettingConfig_MCAN0
    - McuClockSettingConfig_MCAN1
    - McuClockSettingConfig_MCAN2
    - McuClockSettingConfig_MCAN3
    - McuClockSettingConfig_RTI0
    - McuClockSettingConfig_RTI1
    - McuClockSettingConfig_RTI2
    - McuClockSettingConfig_RTI3
    - McuClockSettingConfig_MCSPI0
    - McuClockSettingConfig_MCSPI1
    - McuClockSettingConfig_MCSPI2
    - McuClockSettingConfig_MCSPI3
    - McuClockSettingConfig_MCSPI4
    - McuClockSettingConfig_I2C
    - McuClockSettingConfig_SCI0
    - McuClockSettingConfig_SCI1
    - McuClockSettingConfig_SCI2
    - McuClockSettingConfig_SCI3

Mcu (Mcu) | Cdd_Dma (Cdd_Dma)

Mcu

Name: Mcu

Tabs: General | McuResetSetting | McuClockSettingConfi | McuXbarChannelTrigge | McuInputXbarChannelT | McuGpioXbarIntrConfi | McuPwmConfiguration | McuCddPwmConfigurati

McuClockSrcFailureNotification	DISABLED
McuRamSectors (1 -> 4294967295)	1
McuResetMode	MCU_PERFORM_RESET_MODE_WARM

McuDemEventParameterRefs

Name	McuDemEventParameterRefs
MCU_E_CLOCK_FAILURE	/Dem/Dem/DemConfigSet/MCU_E_CLOCK_FAILURE
MCU_E_MODE_FAILURE	
MCU_E_HARDWARE_ERROR	/Dem/Dem/DemConfigSet/MCU_E_HARDWARE_ERROR

McuPublishedInformation

Name	McuPublishedInformation
McuDeviceVariant	AM263x

Default

Error Log | Problems View | Search

Last 0 entries (up to 100) warnings, errors (filtered out: infos)

Message

Workflows | Sidebar

No workflow selected.

No workflow selected.

No description available.

Properties

Properties for McuPublishedInformation

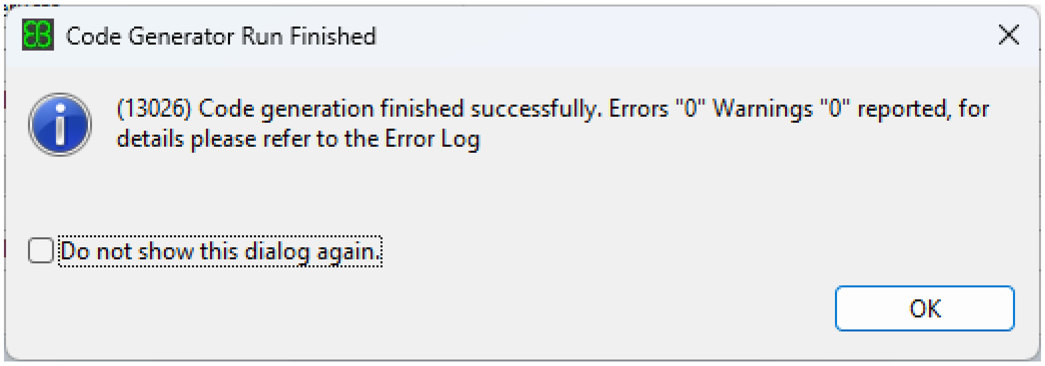
Description | Information | Problems | PostBuildVariantConditions | Comments | CustomAttributes

Short-Name: McuPublishedInformation

Container holding all MCU specific published information parameters

Configuration: Mcu, Module: Mcu_TI_AM263x, Project: MCAL_MCU_Demo

5. Once ready with the values, under the *Context* menu, application *Toolbar* or from the *Application Menu Bar*, click on *Verify* project and then proceed with *Build* project if there are 0 errors.



(13026) Code generation finished successfully. Errors "0" Warnings "0" reported, for details please refer to the Error Log

6. Expand the output folder generated under the project and copy the generated `include/<Mip>_Cfg.h` and `src/<Mip>_Cfg.c` configuration-variant-based configuration source files to use with the end application of any driver.

Documentation on installing and using EB tresos Studio is available under the `<EB_tresos_Installation_Path>/doc/` folder. See also the Users Guide available at `EB/tresos/doc/2.0_EB_tresos_Studio/2.1_Studio_documentation_users_guide.pdf` for more details on using this tool.

2.2 Source Files

Source files include the driver implementation and related header files. These include multiple files separated into the interface exposed for the user of the driver and private source that is closer to a particular target hardware MCU as well as interrupt sources.

Header (.h) and Source (.c) files for a particular driver were named starting with <Mip> for Module Implementation Prefix in Pascal_Snake_Case (that is, a combination of `PascalCase` and `snake_case`) to comply with the AUTOSAR specification for naming and directory structure. <Mip> named files are those having Public Interface contents including Macros, Data Types, and APIs that are exposed to use from a device driver. This source is kept as generic and compliant with the module SRS and SWS specifications. APIs have a fixed, well-defined interface for any given functionality expected by this layer, making the APIs reliable for use with the source library. APIs remain backward compatible; therefore, no cleanup or version migration changes are ever necessary.

Use of highly hardware-specific source named <Mip>_Priv (containing the Private implementation) in the driver application is not recommended. The <Mip>_Priv identifiers and symbols can be updated anytime a hardware design is migrated and need not provide a backward compatibility upon the existing source implementation easily on account of any feature support and bug fixes.

Files named <Mip>_Irq contain the IRQ and Interrupt handling source that is used to service interrupt routine (ISR) which is set using the ARM Nested Vectored Interrupt Controller implementation with TI Vectored Interrupt Module (VIM). Any files included other than just the <Mip> named files are considered as a private driver source whose content is not recommended for direct consideration as an interface usable from the built driver library, even if the files or symbols remain exposed to any part of source or the bundled library.

Example application source can be taken as a go-to reference when setting up the dependencies by initialization for any module in the package and can even work as a starter application for your own custom driver-based program. The example source is available as an exemplary demonstration and holds as no application dependency for any driver library, except for the configuration file header generated for examples that is also used for building driver source library when no different generated configuration header file is used.

2.3 User Guide

MCAL user documentation is available as *Release Notes* present in the root directory, that is, immediately under the MCAL source path. For Sitara™ MCU, additional user documentation be present as either *PDF* documents (for older MCAL versions up to v08.06.02) or *HTML* format web pages (for v09 and later versions, such as the *AM263 User Guide* or AM273 User Guide). User guides contain details on the source features that cover configurable parameters and APIs provided per driver.

See also the *MCAL User Guide* for the latest Sitara™ MPU and Jacinto™ devices.

2.4 Compliance Support Package (CSP)

A Compliance Support Package (CSP) is a collection of document reports provided by Texas Instruments towards aiding the qualification of TI's software components for compliance with safety standards. The reports include different set of documents as available for Baseline Quality (BQ) < Automotive Quality (AQ) < Functional Safety Quality (FSQ). As intended for use in Automobiles, MCALs are aimed towards FSQ. At present, Compliance is only accounted for the driver source, including the configurator plug-ins, whereas other files such as example application and BSW stubs are neither covered nor claimed on the FuSa usage.

CSP documents even go through assessment from an external organization with Functional Safety expertise for the needed SIL and Automotive SIL (ASIL) level certification of the compliance as per the relevant safety standards, such as IEC61508, ISO26262:2018. Following a CSP Audit, the organization can then certify the TI device on the basis of the quality threshold clearance of the company. CSP is not a generic term but is specific to TI's set of qualification documents for Safety compliance.

Until the date of publishing this application note, Automotive Functional Safety certification granted by TÜV SÜD for MCAL v09.01.00 for AM263x Sitara MCU and MCUSW v03.00.00 (based on Processor SDK RTOS v08.06.00) for J7 Jacinto family of devices have ASIL-D/SIL-3 certification and MCAL v09.00.00.02 for AM62x and AM62Ax Sitara MPU and v09.01.00 for the AM273x Sitara MCU family of devices. MCAL certifications are available from TI.com.

3 Utilizing the Source - Building Drivers and Examples

Driver source can be built as a library, and related example application binaries can be built for the executable `.xer5f` format and flashable `.appimage` format. To build either of these, makefiles are included with the bundle for an effortless build. Install the Make client (GMake on Microsoft® Windows®) as a prerequisite to be able to build the source library and examples. Without make, all the steps given under the make command must be used in the same sequence in a command prompt to build the executable output file.

3.1 One-Time Setup

The Code Composer Studio™ integrated development environment (IDE) and TI Arm® code generation tools (CGT) - compiler *Build Dependency* tools must be installed on the system used to build the MCAL source. Make sure the paths to these tools are correct under the `build/Rules.make` file under the TI MCAL package. Edit the relevant paths for these tool directories and other derived applications. Use the exact dependency version in use under the original `Rules.make` contents. The recommended versions can also be cross-checked from the user guide and release notes to make sure that the versions are compatible with the bundle version that was received.

3.2 Build Instructions

Follow these steps to build the drivers:

1. Open a terminal and navigate to the MCAL source directory and `build/` folder.
2. Use the appropriate build target to build the required library or executable binary (`.xer5f`)/ user application (`.appimage`).
   a. The build command syntax for Sitara MCU is:

      ```
      make [-s] <build_target> [PLATFORM=<soc_name>] [PROFILE=<release/debug>] ...
      ```

      In this command, options shown inside square brackets **[ ]** are optional and not required to be provided for a build unless intended to be changed from the set default.

      i. The **-s** flag is used to suppress additional make command logs to simplify the make output.
      ii. *PLATFORM* is the target device series to build for. As an example, for the AM273x Sitara™ MCU family, the *PLATFORM* option can take values from am273x and am2732s
      iii. *PROFILE* sets the build optimization level. Debug build is an unoptimized build that can easily be stepped through and followed along during execution, release build is built for efficient optimizations around the timing and memory span of the compiled source.
   b. The build command syntax for Sitara MPU, Jacinto devices is:

      ```
      make [-s] <build_target> [BOARD=<device_evm_name>] [SOC=<soc_name>]
      [SDK_INSTALL_PATH=<sdk_install_path>] [PROFILE=<release/debug>] ...
      ```

      In this command, options shown inside square brackets **[ ]** are optional and not required to be provided for a build unless intended to be changed from the set default under the `build/Rules.make` file.

      i. The **-s** flag is used to suppress additional make command logs to simplify the make output.
      ii. *BOARD* and *SOC* are the target device series. As an example, for AM6x Sitara™ MPU family, the `<soc_name>` option can take values from am62x, am62ax, or am62px. For the Jacinto family the `<soc_name>` option can take values from a valid j7xxx.
      iii. *SDK_INSTALL_PATH* is the installation path of the corresponding MCU+ SDK RTOS for the SOC family that is present at the installation location.
      iv. *PROFILE* gives the build optimization level. `debug` build is an unoptimized build that can easily be stepped through and followed along during execution. `release` build is built for efficient optimizations around the timing and memory span of the compiled source.
3. Once the build finishes, find the generated files under the `binary/` directory within the package, right above the *build* folder. Driver libraries are in *.aer5f* format and an example application is the `.xer5f` executable and flashable `.appimage` file.

For a build to be performed, the build target name is the only required item in the `make build` command. Other available values can be updated by passing the property `NAME = <value>`; otherwise, the default value for a respective package gets used in build. The build commands only work with the standalone MCAL package. The commands are not the same if the MCAL is a part of an entire AUTOSAR stack delivered by any one vendor as each vendor uses unique compilation settings.

Build-related details and additional information is found under the user guide content.

3.3 Build Command Syntax

Examples of make syntax are provided in the following:

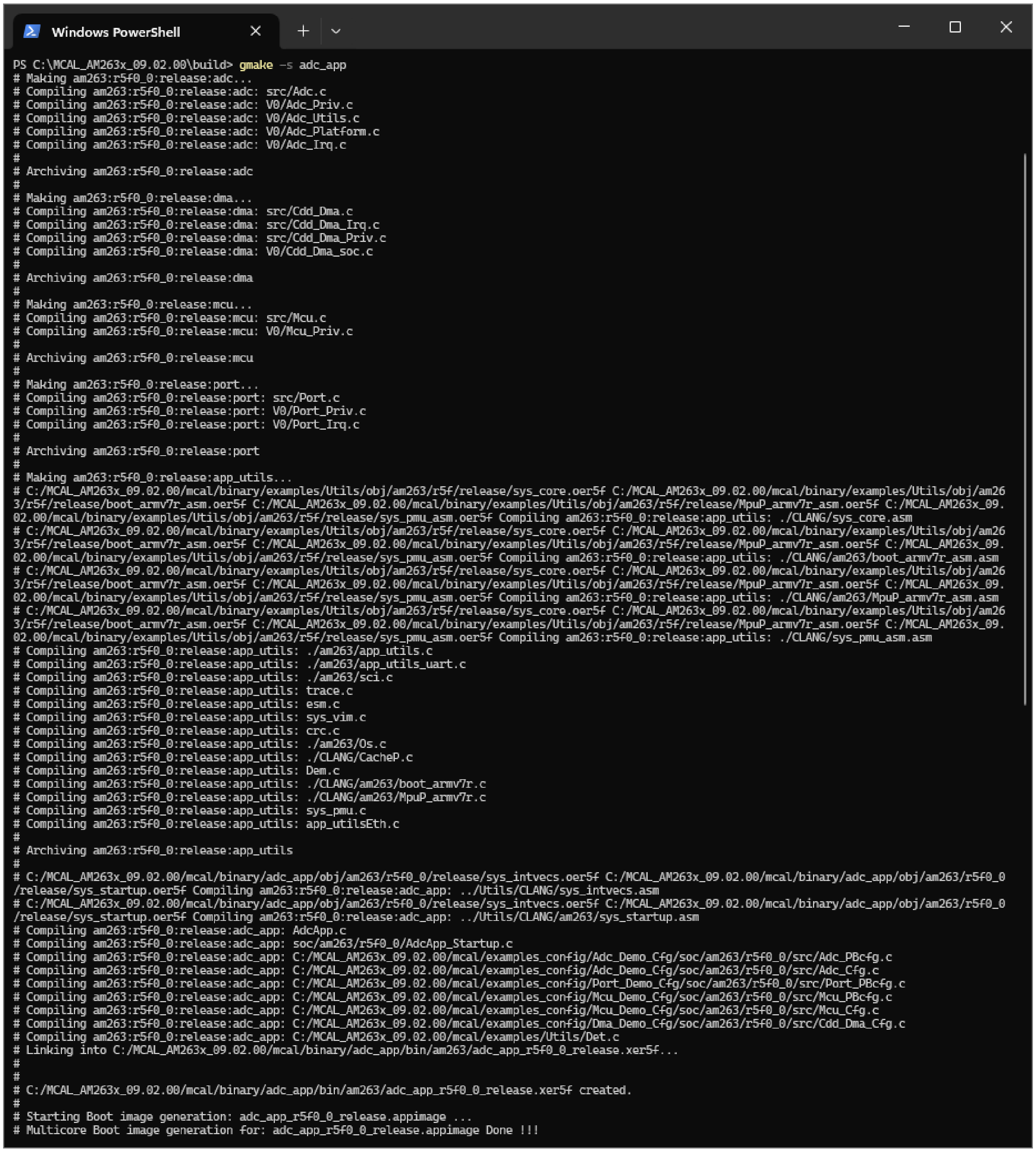
- J7, Sitara MPU:

```
make [-s] <build_target> [BOARD=<device_evm_name>] [SOC=<soc_name>]
[SDK_INSTALL_PATH=<sdk_install_path>] [PROFILE=<release/debug>] ...
```

- Sitara MCU:

```
make [-s] <build_target> [PLATFORM=<soc_name>] [PROFILE=<release/debug>] ...
```

Figure 3-1 shows a Microsoft Windows 11 example where the `gmake -s adc_app` command builds the ADC application along with the driver sources from the `gmake` dependencies.



```
PS C:\MCAL_AM263x_09.02.00\build> gmake -s adc_app
# Making am263:r5f0_0:release:adc...
# Compiling am263:r5f0_0:release:adc: src/Adc.c
# Compiling am263:r5f0_0:release:adc: V0/Adc_Priv.c
# Compiling am263:r5f0_0:release:adc: V0/Adc_Utils.c
# Compiling am263:r5f0_0:release:adc: V0/Adc_Platform.c
# Compiling am263:r5f0_0:release:adc: V0/Adc_Irq.c
#
# Archiving am263:r5f0_0:release:adc
#
# Making am263:r5f0_0:release:dma...
# Compiling am263:r5f0_0:release:dma: src/Cdd_Dma.c
# Compiling am263:r5f0_0:release:dma: src/Cdd_Dma_Irq.c
# Compiling am263:r5f0_0:release:dma: src/Cdd_Dma_Priv.c
# Compiling am263:r5f0_0:release:dma: V0/Cdd_Dma_soc.c
#
# Archiving am263:r5f0_0:release:dma
#
# Making am263:r5f0_0:release:mcu...
# Compiling am263:r5f0_0:release:mcu: src/Mcu.c
# Compiling am263:r5f0_0:release:mcu: V0/Mcu_Priv.c
#
# Archiving am263:r5f0_0:release:mcu
#
# Making am263:r5f0_0:release:port...
# Compiling am263:r5f0_0:release:port: src/Port.c
# Compiling am263:r5f0_0:release:port: V0/Port_Priv.c
# Compiling am263:r5f0_0:release:port: V0/Port_Irq.c
#
# Archiving am263:r5f0_0:release:port
#
# Making am263:r5f0_0:release:app_utils...
# C:/MCAL_AM263x_09.02.00/mcal/binary/examples/Utils/obj/am263/r5f/release/sys_core.oer5f C:/MCAL_AM263x_09.02.00/mcal/binary/examples/Utils/obj/am26
3/r5f/release/boot_armv7r_asm.oer5f C:/MCAL_AM263x_09.02.00/mcal/binary/examples/Utils/obj/am263/r5f/release/MpuP_armv7r_asm.oer5f C:/MCAL_AM263x_09.
02.00/mcal/binary/examples/Utils/obj/am263/r5f/release/sys_pmu_asm.oer5f Compiling am263:r5f0_0:release:app_utils: ./CLANG/sys_core.asm
# C:/MCAL_AM263x_09.02.00/mcal/binary/examples/Utils/obj/am263/r5f/release/sys_core.oer5f C:/MCAL_AM263x_09.02.00/mcal/binary/examples/Utils/obj/am26
3/r5f/release/boot_armv7r_asm.oer5f C:/MCAL_AM263x_09.02.00/mcal/binary/examples/Utils/obj/am263/r5f/release/MpuP_armv7r_asm.oer5f C:/MCAL_AM263x_09.
02.00/mcal/binary/examples/Utils/obj/am263/r5f/release/sys_pmu_asm.oer5f Compiling am263:r5f0_0:release:app_utils: ./CLANG/am263/boot_armv7r_asm.asm
# C:/MCAL_AM263x_09.02.00/mcal/binary/examples/Utils/obj/am263/r5f/release/sys_core.oer5f C:/MCAL_AM263x_09.02.00/mcal/binary/examples/Utils/obj/am26
3/r5f/release/boot_armv7r_asm.oer5f C:/MCAL_AM263x_09.02.00/mcal/binary/examples/Utils/obj/am263/r5f/release/MpuP_armv7r_asm.oer5f C:/MCAL_AM263x_09.
02.00/mcal/binary/examples/Utils/obj/am263/r5f/release/sys_pmu_asm.oer5f Compiling am263:r5f0_0:release:app_utils: ./CLANG/am263/MpuP_armv7r_asm.asm
# C:/MCAL_AM263x_09.02.00/mcal/binary/examples/Utils/obj/am263/r5f/release/sys_core.oer5f C:/MCAL_AM263x_09.02.00/mcal/binary/examples/Utils/obj/am26
3/r5f/release/boot_armv7r_asm.oer5f C:/MCAL_AM263x_09.02.00/mcal/binary/examples/Utils/obj/am263/r5f/release/MpuP_armv7r_asm.oer5f C:/MCAL_AM263x_09.
02.00/mcal/binary/examples/Utils/obj/am263/r5f/release/sys_pmu_asm.oer5f Compiling am263:r5f0_0:release:app_utils: ./CLANG/sys_pmu_asm.asm
# Compiling am263:r5f0_0:release:app_utils: ./am263/app_utils.c
# Compiling am263:r5f0_0:release:app_utils: ./am263/app_utils_uart.c
# Compiling am263:r5f0_0:release:app_utils: ./am263/sci.c
# Compiling am263:r5f0_0:release:app_utils: trace.c
# Compiling am263:r5f0_0:release:app_utils: esm.c
# Compiling am263:r5f0_0:release:app_utils: sys_vim.c
# Compiling am263:r5f0_0:release:app_utils: crc.c
# Compiling am263:r5f0_0:release:app_utils: ./am263/Os.c
# Compiling am263:r5f0_0:release:app_utils: ./CLANG/CacheP.c
# Compiling am263:r5f0_0:release:app_utils: Dem.c
# Compiling am263:r5f0_0:release:app_utils: ./CLANG/am263/boot_armv7r.c
# Compiling am263:r5f0_0:release:app_utils: ./CLANG/am263/MpuP_armv7r.c
# Compiling am263:r5f0_0:release:app_utils: sys_pmu.c
# Compiling am263:r5f0_0:release:app_utils: app_utilsEth.c
#
# Archiving am263:r5f0_0:release:app_utils
#
# C:/MCAL_AM263x_09.02.00/mcal/binary/adc_app/obj/am263/r5f0_0/release/sys_intvecs.oer5f C:/MCAL_AM263x_09.02.00/mcal/binary/adc_app/obj/am263/r5f0_0
/release/sys_startup.oer5f Compiling am263:r5f0_0:release:adc_app: ../Utils/CLANG/sys_intvecs.asm
# C:/MCAL_AM263x_09.02.00/mcal/binary/adc_app/obj/am263/r5f0_0/release/sys_intvecs.oer5f C:/MCAL_AM263x_09.02.00/mcal/binary/adc_app/obj/am263/r5f0_0
/release/sys_startup.oer5f Compiling am263:r5f0_0:release:adc_app: ../Utils/CLANG/am263/sys_startup.asm
# Compiling am263:r5f0_0:release:adc_app: AdcApp.c
# Compiling am263:r5f0_0:release:adc_app: soc/am263/r5f0_0/AdcApp_Startup.c
# Compiling am263:r5f0_0:release:adc_app: C:/MCAL_AM263x_09.02.00/mcal/examples_config/Adc_Demo_Cfg/soc/am263/r5f0_0/src/Adc_PBcfg.c
# Compiling am263:r5f0_0:release:adc_app: C:/MCAL_AM263x_09.02.00/mcal/examples_config/Adc_Demo_Cfg/soc/am263/r5f0_0/src/Adc_Cfg.c
# Compiling am263:r5f0_0:release:adc_app: C:/MCAL_AM263x_09.02.00/mcal/examples_config/Port_Demo_Cfg/soc/am263/r5f0_0/src/Port_PBcfg.c
# Compiling am263:r5f0_0:release:adc_app: C:/MCAL_AM263x_09.02.00/mcal/examples_config/Mcu_Demo_Cfg/soc/am263/r5f0_0/src/Mcu_PBcfg.c
# Compiling am263:r5f0_0:release:adc_app: C:/MCAL_AM263x_09.02.00/mcal/examples_config/Mcu_Demo_Cfg/soc/am263/r5f0_0/src/Mcu_Cfg.c
# Compiling am263:r5f0_0:release:adc_app: C:/MCAL_AM263x_09.02.00/mcal/examples_config/Dma_Demo_Cfg/soc/am263/r5f0_0/src/Cdd_Dma_Cfg.c
# Compiling am263:r5f0_0:release:adc_app: C:/MCAL_AM263x_09.02.00/mcal/examples/Utils/Det.c
# Linking into C:/MCAL_AM263x_09.02.00/mcal/binary/adc_app/bin/am263/adc_app_r5f0_0_release.xer5f...
#
#
# C:/MCAL_AM263x_09.02.00/mcal/binary/adc_app/bin/am263/adc_app_r5f0_0_release.xer5f created.
#
# Starting Boot image generation: adc_app_r5f0_0_release.appimage ...
# Multicore Boot image generation for: adc_app_r5f0_0_release.appimage Done !!!
```

Figure 3-1. Make Output Example

4 Obtaining an MCAL package

Texas Instruments delivers MCAL packages for utilization on devices as both: AUTOSAR stack bundle from several of the top entire stack vendors such as Vector, ETAS, and as a standalone MCAL+CDD bundle which can be requested from your nearest customer support and field engineer who can share TI's license export form on the latest MCAL version in TI's official website. Examples include: MCAL AM263x v10.00.00, MCAL AM263Px v10.00.00, and MCAL AM273x v09.01.00. Similarly for Sitara MPU devices and the Jacinto 7 entry device, the MCAL+CDD standalone bundle can be requested from the nearest customer support or field engineers or through the *Software Export* request form. Examples include: MCAL AM62x v10.00.00, MCU+SDK AM62x v10.00.00, and Processor SDK RTOS (MCU+ SDK) J722S v10.00.00. The MCAL versions are matched with device MCU+SDK release versions. MCAL for the remaining Jacinto devices is publicly available on TI's website ti.com under the MCUSW folder as part of RTOS SDK (example: J721E RTOS SDK), where the configurator plug-ins package for the drivers must be requested separately.

While requesting an MCAL package, the EB tresos Studio User License can also be requested to make use of the configurator plug-ins in TI's partner Configurator tool.

MCAL started on AM263x and AM273x Sitara MCUs with the v08.06 release at the beginning of 2023, followed by several releases. The latest release of v09.02.00 was in May 2024. MCALs have previously been available for Sitara MPU and Jacinto family of devices, the latest version is v10.00. These release versions also follow semantic versioning as explained under the introduction section for earlier AUTOSAR standard versions and according to the device SDK (MCU-PLUS-SDK) release version of the timeline available for a device from TI's official website.

Many other releases and long-term support are planned for MCALs available for various products, including the MCAL packages for Jacinto, Sitara MPU, Sitara MCU, C2000, Hercules MCU, Radar, MSP, and any other such TI devices.

5 Summary

Texas Instruments Incorporated delivers the MCAL package to enable an AUTOSAR stack on the Functional Safety compliant, automotive-grade Embedded Processors ranging from Jacinto Processors to Sitara Microprocessor and Microcontroller units. The MCAL package provides a regular SDK that includes simple device drivers to enable hardware interface by means of software APIs. The MCAL package is also useful for getting the driver interface maintained as standard component API as mandated by the AUTOSAR standard for MCAL and CDD drivers while easing for a risk-tolerant system access in compliance with the ISO 26262:2018 Automotive Functional Safety standard. The package can be requested for a standalone use on TI's chips and can also be obtained from any of TI's leading AUTOSAR stack vendor partners within the complete stack.

6 References

1. Texas Instruments, *Platform MCAL Development - AM263 User Guide*
2. Texas Instruments, *Platform MCAL Development - AM273 User Guide*
3. Texas Instruments, *Sitara MPU and Jacinto device user guide*
4. Texas Instruments, *Jacinto AUTOSAR MCAL Drivers Certificate*
5. Texas Instruments, *[FAQ] Sitara AM263x/AM273x MCAL - Getting Started Guide | TI E2E™ forums*
6. Texas Instruments, *[FAQ] How to get EB Tresos studio for MCAL development and customer license? | TI E2E™ forums*
7. Texas Instruments, *Functional safety | TI.com*
8. Texas Instruments, *Functional Safety for AM2x and Hercules™ Microcontrollers Product Overview*
9. Texas Instruments, *AutoSAR MCAL — Hercules Safety MCU Resource Guide*