



Title of Paper: Missing Resolutions in AUTOSAR 2.1

Author's Name: Tahsina Jamal S

Mail Address: Embedded Systems & Software (EmSyS)
Larsen & Toubro Limited
Plot No. 25 – 30
EPIP 1st Phase
Industrial Area, Whitefield
Bangalore – 560 066
INDIA

Email ID: tahsina.jamal@Intemsys.com

Abstract:

With the increasing electronics in automotives, managing software in the automotive is becoming more and more difficult. Any small change in the requirements may end up in a completely new development cycle. However as ECUs become more complex and feature rich, it is the design and function of software driven interfaces that increasingly differentiate the players and determine the winners. Development cost is also a major factor in profitability, especially where margins are slim. Though software modularity is not new, with the advent of standards as proposed by AUTOSAR, now the developers have the advantages of distributed development, simpler integration & easier maintenance.

However in the Release 2.1 that happened in August 2007 lot of inconsistencies are found in the areas of BSW Configurations, File Names, Typedefs of the variables used across the BSW modules.

Discrepancy is noticed in the naming of the configuration parameters between the specification and the module definition file.

Among the different variants of configuration, the Post-Build variant provides flexibility to the OEMs in modifying configuration data at a later stage of ECU manufacture. This is achieved by flashing the code in to the ECU by Tier 1 supplier leaving some locations blank for the configuration data. The vehicle dependent data is then flashed in to the ECU at the OEM's place. To achieve this, the post-build configuration data should be independent of the code. As per 2.1 specifications, the post-build configuration data has reference to code in the form of notification functions which will result in unresolved symbols if compiled separately.

The file names mentioned in the specifications of the dependent modules of the BSW also are not consistent, giving rise to integration problems.

This paper will deal in detail the discrepancies observed in 2.1, evaluate the resolutions proposed in 3.0 & will propose the solutions to achieve resolutions.

INTRODUCTION

The challenges faced by the automotive industry in terms of environmental compatibility and safety are very high. Because of this the number of functions controlled by the electronics in an automobile has increased dramatically. These functions are embedded in various electronic control units distributed all over the automobile, making the ECUs software rich. Hence standardization is a must to maintain control over such a complex network. As standardization & reusability of software components from different suppliers between various vehicle platforms and OEMs increases, managing highly integrated & complex software becomes difficult.

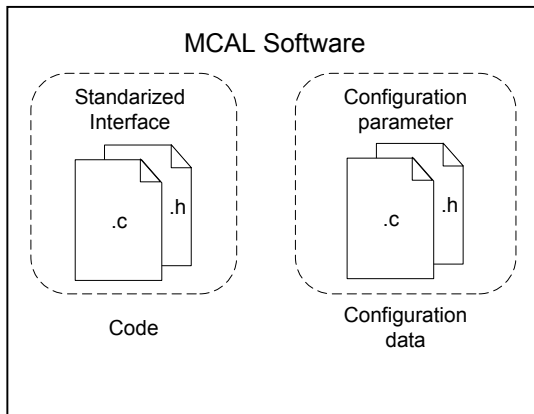
Under such circumstances AUTOSAR consortium has evolved with the following major goals

- Modularity
- Scalability
- Exchangeability
- Reusability and
- Standardized Interfaces

In order to meet its objectives, AUTOSAR has released specifications standardizing the central architectural elements across functional domains, allowing industry competition to focus on implementation. Though AUTOSAR stands for standardization of the software architecture of automotive systems and promote the development of interchangeable software components, the AUTOSAR specifications are not consistent to address the same. AUTOSAR consortium has been continuously working towards eliminating inconsistencies in the specifications. AUTOSAR specifications version 3.0 has already resolved substantial anomalies. However there are a few issues pending for resolution in Release 3.0. This paper is an attempt towards identifying those issues and suggests a suitable resolution for MCAL layer.

INCONSISTENCIES BETWEEN RELEASE 2.1 AND RELEASE 3.0

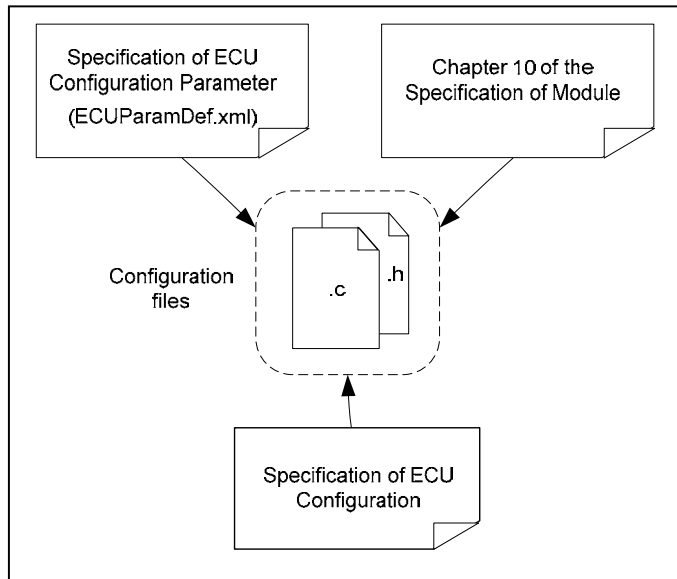
The implementation of MCAL software primarily involves the development of standardized interfaces & configuration parameters. The standardized interfaces comprise the definition



of APIs, Data types and File names. The inputs for implementing the standardized interfaces are from the individual module specification (SWS).

AUTOSAR proposes to store all the configuration parameters in separate c—files for the ease of modifying them at different stages. The inputs for implementing the

configuration files are from the Specification of ECU Configuration Parameters (XML), Specification of ECU Configuration (AUTOSAR_ECU_Configuration.pdf) and Chapter 10 of the AUTOSAR SWS of individual module driver.



As for the discrepancies in the Specification 2.1, the major differences exist between the specification of individual modules and Specification of ECU Configuration Parameters (Available as an XML file named

AUTOSAR EcucParamDef.arxml). The inconsistencies present in the configuration parameters of release 2.1 and how they are addressed in Release 3.0 is described in the following sections.

Configuration - Optional functionality

This allows the flexibility of excluding parts of the source code that are not needed. The MACROs for enabling / disabling APIs are not consistent between the Software Specification (SWS) and the Specification of ECU Configuration Parameters (XML). The MACROs are mentioned in one document and missed in the other.

API Enabling/Disabling	Release 2.1	
	XML	SWS
GptStartStopTimerApi	✓	X
IcuSignalMeasurementApi	X	✓
IcuEdgeDetectionApi	X	✓
PortSetPinModeApil	X	✓

API Enabling/Disabling	Release 3.0	
	XML	SWS
GptStartStopTimerApi	X	X
IcuSignalMeasurementApi	✓	✓
IcuEdgeDetectionApi	X	X
PortSetPinModeApil	✓	✓

Configuration Parameters - MCAL

Configuration Parameter	Release 2.1	
	XML	SWS
CanWakeupSupport	X	✓
CanControllerId	✓	X
CanControllerTimeQuanta	✓	X
CanHandleType	X	✓
PortPinInitialMode	X	✓
SpiMaxChannel	X	✓
SpiMaxJob	X	✓
SpiMaxSequence	X	✓
WdgIndex	✓	X
FrNumCtrl	X	✓
FrModulePrefix	X	✓
GAssumedPrecision	X	✓
GdBitMax	X	✓
GdBitMin	X	✓
GdMaxMicrotick	X	✓
GdMaxPropagationDelay	X	✓
GdMinPropagationDelay	X	✓
McuClockReferencePoint	✓	X
McuClockReferencePointFrequency	✓	X
McuRamSectors	✓	X

Configuration Parameter	Release 3.0	
	XML	SWS
CanWakeupSupport	✓	✓
CanControllerId	✓	✓
CanControllerTimeQuanta	✓	✓
CanHandleType	✓	✓
PortPinInitialMode	✓	✓
SpiMaxChannel	✓	✓
SpiMaxJob	✓	✓
SpiMaxSequence	✓	✓
WdgIndex	✓	✓
FrNumCtrl	X	X
FrModulePrefix	X	X
GAssumedPrecision	✓	✓
GdBitMax	✓	✓
GdBitMin	✓	✓
GdMaxMicrotick	✓	✓
GdMaxPropagationDelay	✓	✓
GdMinPropagationDelay	✓	✓
McuClockReferencePoint	✓	✓
McuClockReferencePointFrequency	✓	✓
McuRamSectors	✓	✓

The inconsistencies that were present in the configuration of optional functions as well as the other configuration parameters of Release 2.1 have been addressed in Release 3.0. Before the release of 3.0, BSW developers resolved themselves. One of such solutions is, if the configuration option was found in the XML and dropped in SWS, it was dropped in the `***_Cfg.c / ***_Cfg.h` but still implemented in the ECU parameter definition (epd) by virtue of the requirement `ecuc_sws_1003` that states elements with a lower multiplicity greater than zero cannot be deleted. If the configuration option was present in the SWS and dropped in the XML, the configuration option was implemented as a vendor specific configuration parameter. [`ecuc_sws_1014`: Additional vendor specific parameters can be added anywhere within the module definition].

Multiple Configuration Sets

This allows the description of several configuration sets for a module. Though the Specification of ECU Configuration, (AUTOSAR_ECU_Configuration.pdf) describes the multiple configuration sets definition, this attribute (MULTIPLE-CONFIGURATION-CONTAINER) was not present in the XML of Release 2.1. Before the release of 3.0, the multiple configuration sets were handled through the multiplicity attribute. Wherever multiple configuration sets were required, the upper multiplicity was kept as *. In Release 3.0 the attribute MULTIPLE-CONFIGURATION-CONTAINER is included in the XML. The Release 3.0 apart from including the attribute also explains the symbolic names that can be given to these multiple configuration containers through 2 new requirements. These requirements describe that the name given to the container shall be a valid C- literal and shall be unique for the whole ECU. The benefit of these requirements is that it will reduce the burden on the tool to check for the uniqueness of the name given to the container.

Implementation Configuration Classes

The AUTOSAR Basic Software supports three different configuration classes namely, Pre-compile time, Link time and Post-build time. In the Release 2.1, Specification of ECU Configuration described an attribute called Implementation Config Class for all the configuration parameters and an attribute called Implementation Config Variant for the module. The same were missing in the XML version of Specification of the ECU Configuration Parameters. In Release 3.0 these attributes are added in XML and

explanation about the mapping of the ConfigurationVariant to the ConfigurationClass is added in the Specification of ECU Configuration. The implementation of different configuration variants are detailed in Release 3.0 and no significant discrepancy is noted.

The Post-build Configuration variant

There are two types of Post-build configurations one is Post-build time loadable and the other is Post-build time selectable. The difference between these two configuration classes is only relevant for the memory mapping of the configuration data during linking stage and has no significance with respect to ECU configuration definition. However it is required to know which type of Post-build is adopted for a project to decide the steps involved in the development. In order to make the post-build time loadable re-configuration possible, it should be possible to compile and link the configuration data that is generated as normal C independent of the source code. Hence the configuration data should be strictly separate from the source code. However in the configuration specifications of the various BSW modules, the function pointers for notification functions are a part of the configuration data. Since the notification functions form a part of the source code, the configuration data when compiled and linked independent of the source code will result in linker errors as the function pointers are unresolved. There should be a method defined to overcome this linker error. One of the suggestions is to adopt the method involved in changing the calibration data in an ECU for reloading configuration data. The activity can be made tool based wherein the tool has to read the addresses of the symbols to be resolved from the Map of the Executable and create a hex file which can be flashed to some known memory location in the ECU.

Typedefs

AUTOSAR specifies various values like status, job result, mode, etc. for obtaining the information of a module. The status value denotes the current status of the underlying abstraction layer or driver as the case may be. The job result denotes the result of the last job. For uniformity across layers and to avoid name clashes with other modules, AUTOSAR specifications describe the naming of the status values and the job result values for each module. This is applicable especially to the values that are visible outside the modules. Hence all dependent modules of different layers have to use the same

names as described in the specification. In Release 2.1 inconsistencies were noticed with respect to these names in Memory modules and Watchdog modules. The name of the job result value that denotes that the job has been cancelled was mentioned as MEMIF_JOB_CANCELLED in 2.1 Specification of Memory Abstraction Interface whereas it is mentioned as MEMIF_JOB_CANCELED in the 2.1 Specification of Module Flash Driver and Specification of EEPROM Driver. This will lead to integration problems during distributed development. In the Release 3.0, this is corrected in the Specification of EEPROM Driver and is still not corrected in the Specification of Module Flash Driver.

Module Name	Job Result Name	
	Release 2.1	Release 3.0
Memory Abstraction Interface	MEMIF_JOB_CANCELLED	MEMIF_JOB_CANCELLED
Flash Driver	MEMIF_JOB_CANCELED	MEMIF_JOB_CANCELLED ^{Note:1}
EEPROM Driver	MEMIF_JOB_CANCELED	MEMIF_JOB_CANCELLED

Note1: It is suggested to implement the job result name as MEMIF_JOB_CANCELLED.

In the Release 2.1 of Specification of Watchdog Driver it is described that the types shall be used from WdgIf and also in the Specification of Watchdog Interface it is described that the underlying Wdg Driver shall use the WdgIf types. But in the other sections of the Specification of Watchdog Driver (WDG018 & WDG019), the status types were not consistent with WdgIf types. This is applicable to Release 3.0 also. Though this will not cause integration issues as the status type is internal to the module, the inconsistency will be confusing for the software developer. It is suggested to use the WdgIf status types as this ensures uniformity.

API Definitions

According to *PORT086* requirement (Release 2.1), the Port_SetPinDirection API is available based on the configuration of the runtime parameter PORT_PIN_DIRECTION_CHANGEABLE. The availability of a function cannot be turned on/off at run time. This has to be done at pre-compile time only. Moreover this parameter (PORT_PIN_DIRECTION_CHANGEABLE) is with respect to individual port pins and hence cannot be used for configuring the availability of the function at the module level. In Release 2.1 the availability of the function can be configured through the parameter PORT_PIN_DIRECTION_CHANGES_ALLOWED. This parameter is replaced by

PortSetPinDirectionApi in Release 3.0. However the requirement *PORT086* has not changed even in Release 3.0. The resolution is to drop the *PORT086* and use the Port_SetPinDirectionApi parameter for configuring the availability of Port_SetPinDirection API.

In the CAN module, in both Release 2.1 & 3.0, the parameter CanControllerBaudRate is mentioned as configurable. The Controller baud rate is dependent on CanControllerSeg1, CanControllerSeg2, CanControllerPropSeg, CanControllerTimeQuanta, etc. and can be derived from them. It is suggested to drop the CanControllerBaudRate parameter and derive the values from the other parameters.

In the Release 2.1 of the FEE module, Requirement *FEE075* described that Fee_MainFunction shall return E_NOT_OK, if the block requested for reading has been invalidated by the upper layer module. Since the Fee_MainFunction returns void, this cannot be implemented. This has been addressed in Release 3.0.

The API Parameter checking is not available for Fee_Read & Fee_Write APIs in both Release 2.1 and Release 3.0. It is recommended to employ error checking for the passed parameters especially for size and the pointer validity.

In the Release 2.1, the Service Id of two APIs Fr_GetAbsoluteTimerIRQStatus and Fr_GetRelativeTimerIRQStatus in the FlexRay driver specification is same namely, 0X20. This has been addressed in release 3.0.

Header File Structure

Inconsistencies were noticed in the Header file structure section (Section 5.1) of the Release 2.1 specifications. These are present in Release 3.0 also. The figure given for the header file structure and the text do not match (FEE & EA modules). In the LIN Driver module, a header file is shown to include a source file in the figure (Lin.h includes Lin_Irq.c). In the CAN Driver module, the file names in the text and figure do not match case (Figure shows Canif_Cbk.h). The software developer may consider this as an optional requirement and if does not pay attention to the case in the file names, will lead to

integration issues. It is suggested to follow the 'General Requirements on Basic Software modules' for naming conventions and include structures.

REFERENCES

Release 3.0 Specifications: https://svn2.autosar.org/repos2/22_Releases/30_Release3.0

Release 2.1 Specifications: https://svn2.autosar.org/repos2/22_Releases/release2.1

ACKNOWLEDGEMENT

I take this opportunity to thank everyone in Automotive Group of EmSyS for their contribution towards writing this paper.

CONCLUSION

In addition to the Specification of ECU Configuration Parameters (XML), the release 2.1 also contained another document of the same name 'Specification of ECU Configuration Parameters' in pdf format (AUTOSAR_ECU_ConfigurationParameters.pdf). Lot of discrepancies was noted between the XML and the pdf version of Specification of ECU Configuration Parameters in terms of the organization of the containers, naming of the Configuration Parameters, etc. Since the Specification of ECU Configuration Parameters was removed from the set of deliverables for Release 3.0 these discrepancies are not discussed in this paper.

Also the known inconsistencies that were documented in the specifications in the Release 2.1 are not addressed in this paper.

This paper brings out the discrepancies found in the release 2.1 specifications of AUTOSAR and outlines how these inconsistencies have been addressed in release 3.0 specification of AUTOSAR. We have mainly concentrated on the Microcontroller Abstraction Layer (MCAL) and this paper can be a ready reference for those who are in the process of migration of the MCAL to Release 3.0.