

Use of AUTOSAR Framework to Assist Software Field Programming

2010-01-0671

Published
04/12/2010

Sandeep U. Menon
Electrical & Controls Integration Lab, General Motors R&D

Copyright © 2010 SAE International

ABSTRACT

Field reprogramming of electronic control units (ECU) via the serial communication bus for feature upgrades, software fixes is an area which has potential cost impacts to the OEMs. Due to increasing software complexity, feature content, up-integration considerations for ECUs in next generation vehicle platforms the expectation is for flash memory requirements to increase significantly in the future. A reduced reprogramming cycle that builds on top of the existing system framework would be of interest from a cost and timing aspects. Additionally deployment of the next generation of telematics based remote programming techniques would also benefit from shorter reprogramming time in ECUs. An approach to address field reprogramming time would be to migrate to high baud rate communication protocols like FlexRay or Ethernet from the traditional CAN based systems currently in use.

In this paper an alternative or complementary approach is proposed that would consider targeted programming on software modules instead of the complete application software when multiple ECUs in a vehicle platform share a common software partition/ framework. This approach could be explored in AUTOSAR (AUTomotive Open System ARchitecture) based environment. AUTOSAR which has been jointly developed by OEMs, suppliers and tool vendors, is becoming a defacto standard for open automotive software architecture. The AUTOSAR software partition and layered architecture provide a capability to assist field reprogramming. By applying some enhancement to the memory partitioning/linking in an AUTOSAR-based ECU this option could potentially assist software component based reprogramming independent of the complete application software

INTRODUCTION

Memory requirements in ECUs for the automotive domain are projected to increase significantly due to increased software content and deployment of complex features. Embedded software in ECUs is not only driving most of the innovations and vehicle value in the automotive domain, but is also becoming a bigger source of costly-to-repair defects for vehicle systems [1]. As part of the ECU development process ECU suppliers need to provide capabilities to (re)program ECUs at EOL and for software service upgrades via the network communication bus. Current implementations consider the application software as single block of flash memory which has to be updated atomically. One of the key limiting factors for in-vehicle programming is the bus bandwidth. Due to this limitation network buses like Ethernet and FlexRay are being investigated as an option to provide increased throughput and reduced programming time. However even with increased network bandwidth, reprogramming application software due to defects still remain inefficient due to the fact that complete application software has to be reprogrammed even for minor software defects which can be isolated to few application software modules.

Most automotive software architectures currently in production are customized and designed in house by ECU vendors. The differences in software partitioning and hardware abstraction specific associated with software architectures designed by individual development house make them inflexible for generic component based programming strategies at the OEM level. Most in house software architectures and interfaces are subject to significant revisions to accommodate implementation needs. Compliance to the multitude of partition / definitions for individual software architectures is not measurable or verifiable at the OEM level. This limits the capability to design a common reprogramming strategy, via the communication bus for

**Documents**

Document Sharing Hub

individual software modules at application software layer and the expertise to isolate and replace these modules in flash memory sections of the ECU.

AUTOSAR provides an industry standard software partition and layered structure which could be useful to localize the AUTOSAR Software Component's (SWC) programming using the communication network to individual partitions. One of the objectives of AUTOSAR is to create and establish open standards for automotive E/E (Electrics/Electronics) architectures which will facilitate the basic infrastructure to support reusable vehicle software, user interfaces and management for most vehicle application domains. The s/w architecture framework and partition defined for software could potentially be expended into the flash memory domain with minimal revisions. This paper explores the possibility of limiting software reprogramming to individual SWCs in an AUTOSAR framework and to identify possible hurdles to meeting this capability.

BOOTLOADER PRINCIPLES

A bootloader in most automotive ECUs is a flash based application with an integrated network communication protocol stack with capabilities listed in [5]. The bootloader supports download/update of application and calibration software via the communication bus through a generic hardware (ALDL) interface in the vehicle. To protect against inappropriate access to the flash memory, the flash routines for the memory access in most cases are downloaded at programming time and executed from micro-controller's RAM space as shown in Figure 1. The flash bootloader (FBL) will invoke the micro-controller flash routines like read, write, and erase operations during programming session. The FBL software will receive segmented data via the communication bus which are reassembled and written into the appropriate flash memory blocks.

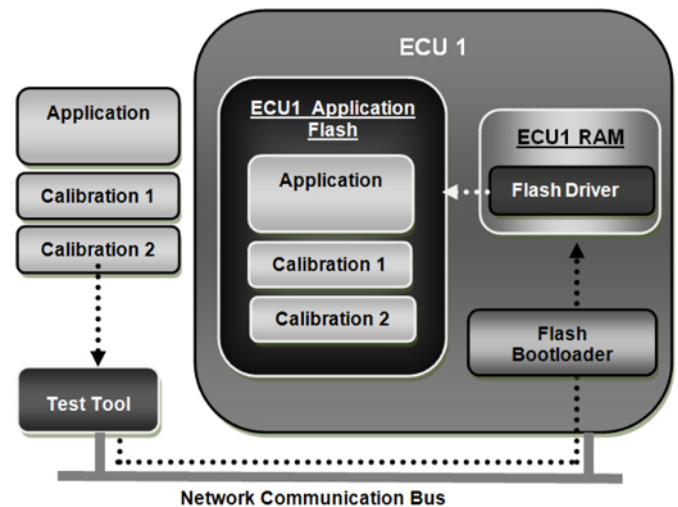


Figure 1. Bootloader communication path

The micro-controller flash memory space is usually divided into memory blocks/sectors of fixed or variable sizes per the semiconductor vendor's specifications. Most micro-controllers provide options to erase the memory blocks instantaneously or by specifying individual block units. The micro-controller may also have external GPIO input pins which need to be controlled to allow write access to the flash memory. The bootloader software defines the application memory blocks (AMB) which are multiples of the smallest flash memory blocks in memory. The boundaries of the AMB need to have start and end address which overlap with the flash memory block addresses. The bootloader receives information at the beginning of programming session which defines the memory start address and size information of the data to follow in the communication bus. This information will be validated before any programming activities can be initiated. The component file for flash download will have a checksum value for the complete application software which is provided at the beginning of the download to the bootloader. After the completion of download, the FBL uses the flash driver to read back the memory area written to compute the checksum value using the same algorithm and compare it to the test tool provided value.

MEMORY PARTITIONING FOR AUTOSAR

The AUTOSAR framework supports the implementation of application level functions independent of their distribution on different ECUs; to improve design flexibility and to simplify integration activities. The major partitions in software in the AUTOSAR framework as shown in Figure 2 are listed below:

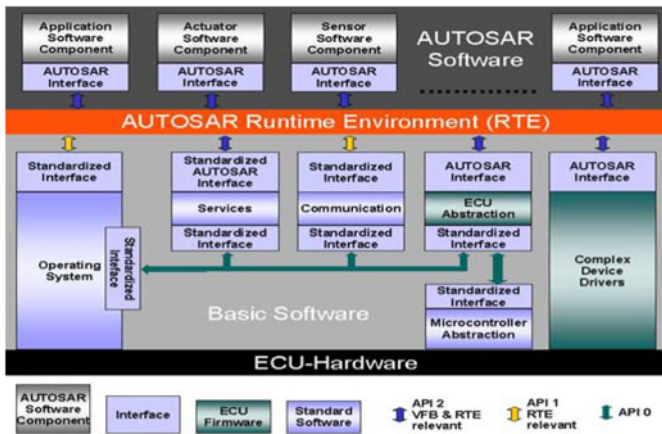


Figure 2. AUTOSAR Partition [2]

BSW (Base Software): Basic Software layer is the standardized software that does not support any application level features. BSW offers both hardware-dependent and hardware-independent services and necessary abstraction to the Run Time Environment (RTE) layer.

RTE (Runtime Environment): This layer handles the information exchange between the application software components and BSW. RTE allows software components to the use of ECU's resources without direct interaction between the software component and the ECU's OS, Basic Software Modules, etc.

SWCs (Application S/W Components): This is the layer where the actual application level functionality is situated. The layer is composed of application software components that interact with the run time environment.

To translate the AUTOSAR software-level partition into an ECU flash memory partition (as shown in Figure 3) with the same set of abstraction components appropriate linker level definitions need to be formulated. Linking is one of the last steps in the compilation process of the software build, which is used to convert the instruction list/object files created from the assembler to a memory image of software. Most embedded software implementations are built from multiple files. The linker performs the necessary steps to identify the address location in memory for the program instructions and data to produce the executable hex file. Capability to control the memory location assignment of software modules is important to facilitate allocation of data structures and instructions needed for OS, interrupts, etc. to operate correctly. A developer created linker map file is used in many cases to specify the order/address at which the object files are allocated in the microcontroller's memory space.

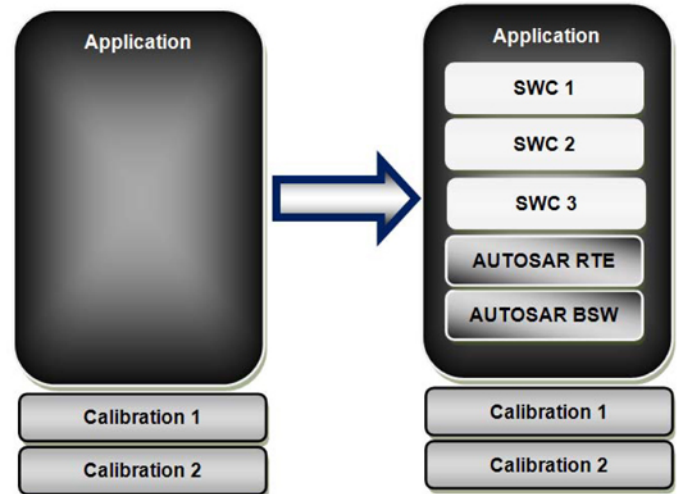


Figure 3. AUTOSAR Application Memory Partition

The fundamental problem with extraction and replacement of multiple revisions of SWC memory sections between of software builds would be the concern with broken function links accessed externally. With revisions to the software, there is a potential for function start addresses to be displaced and internal to be revised. Use of non-AUTOSAR architectures may include infrastructure (ECU hardware specific) functionality/access in application software modules. AUTOSAR allows the application software to be abstracted from the infrastructure software. With the use of SWC, interfaces functions are limited to the RTE layer only. Additionally no functionality is implemented in SWCs to directly invoke OS function calls, H/W ports definitions/access from the application functions. The SWC memory allocation for linker mapping as it exists currently may need to be enhanced to meet the requirements for reprogrammable software components. Application reprogramming is considered only for SWCs and not for other AUTOSAR layers like RTE or BSW. This is due to the complex linkage between different infrastructure software components, which make it difficult for individual BSWs and the RTE layer to be reprogrammed independently. Potential options to isolate and program BSW and RTE are not considered a cost effective solution and it may be necessary to have them reprogrammed together as part of change management impacting these layers.

AUTOSAR already provides provisions to support memory mapping (MemMap.h) via keywords to control the assignment of variables and functions to specific memory sections [4]. The assignment of sections to dedicated memory areas/addresses for a given microcontroller is outside the scope of AUTOSAR and is expected to be performed using linker control files. The memory allocation identifiers stay undefined if compiler/linker does not support the memory mapping functionality. AUTOSAR also mandates the need to