Development of Automotive Embedded System based on AUTOSAR Methodology

Kisoon Sung, Taeman Han
Electronics and Telecommunications Research Institute
{kssung, tmhan}@etri.re.kr

Abstract. Recently it has been increased embedded system using a variety of electronic parts in car domain. AUTOSAR (AUTomotive Open System Architecture) is the most notable industrial standard for developing automotive embedded software. AUTOSAR is a partnership of automotive manufacturers and suppliers working together to develop and establish an open industry standard for automotive E/E architectures. In this paper, we will introduce AUTOSAR briefly and demonstrate the result of automotive software - LDWS (Lane Detection & Warning System) - development.

Keywords: AUTOSAR, ECU, Simulink, LDWS

1 Introduction

Recently it has been increased embedded system using a variety of electronic parts in car domain. AUTOSAR is the most notable industrial standard for developing automotive embedded software. AUTOSAR is a partnership of automotive manufacturers and suppliers working together to develop and establish an open industry standard for automotive E/E architectures. AUTOSAR aims to improve complexity management of integrated E/E architectures through increased reuse and exchangeability of SW modules between OEMs and suppliers. Moreover AUTOSAR intends to improve flexibility for product modification, upgrade and update, reliability of E/E system, and to enable detection of errors in early design phase. To achieve that, AUTOSAR standardizes open software architecture, methodology and application interface to reflect these solutions. AUTOSAR architecture part suggests the open software architecture that includes a complete basic software stack for ECUs as an integration platform for hardware independent software applications. AUTOSAR methodology part defines exchange formats and description templates to enable a seamless configuration process of the basic software stack and the integration of application software in ECUs. And it defines the methodology how to use this framework as well. [1]

In this paper, we will introduce AUTOSAR briefly and demonstrate the result of automotive software, LDWS development. This paper is organized as follows. In next section, we will introduce AUTOSAR architecture and methodology. We will describe the AUTOSAR software code composition as well. In section three, we will
describe the development procedure LDWS software. Finally, in section four, we draw some conclusion.

2 AUTOSAR Overview

2.1 AUTOSAR software architecture

Fig 1 shows the AUTOSAR software architecture that distinguishes four basic layer, application, RTE (RunTime Environment), BSW (Basic SoftWare) and microcontroller. The RTE provides the infrastructure services that enable communication to occur between AUTOSAR software-components as well as acting as the means by which AUTOSAR software-components access basic software modules including the OS and communication service. [2] The BSWs are divided into functional groups that are System, Memory, Diagnostic and Communication Services. These services are used in automotive system commonly.

![AUTOSAR Software Architecture](image)

Fig. 1. AUTOSAR Software Architecture

2.2 AUTOSAR Methodology

The AUTOSAR methodology is foreseen to support activities descriptions and use of tools in AUTOSAR. The AUTOSAR methodology is not a complete process description but rather a common technical approach for some steps of system development. It defines activities and work-product except role and responsibility. AUTOSAR meta-model defines the contents of the work product and is a formal description of all the information that is produced or consumed in the AUTOSAR methodology.
Fig. 2. AUTOSAR Methodology [3]

Fig. 2 shows the outline of AUTOSAR methodology. In each step, AUTOSAR-based software is developed as follow procedure.

**Step 1**
1) Define the item to development, requirement, and constraints
2) Describe SW-Component independently of hardware
3) Describe HW independently of Application Software
4) Describe System – network topology, communication

**Step 2**
1) Distribute SW-Component description to ECU

**Step 3**
1) Generate required configuration for AUTOSAR infrastructure per ECU

**Step 4**
1) Generate software executable based on configuration information for each ECU

In General, it is supported by tool though all over the development step, for example Tresos by EB, SystemDesk by dSpace, and Davinci by Vector.

### 3 Develop AUTOSAR based system; LDWS

In this section, we demonstrate the development of a LDWS controller according to AUTOSAR methodology. First, we set up the development environment. Table 1 shows the tool chain to develop LDWS controller. And our target hardware is Infenion XC-2300 chip.

**Table 1.** Tool chain to develop LDWS controller

<table>
<thead>
<tr>
<th>Tool</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>AutoWorks</td>
<td>ETRI</td>
</tr>
<tr>
<td>Simulink</td>
<td>Mathworks</td>
</tr>
<tr>
<td>CANoe</td>
<td>Vector</td>
</tr>
<tr>
<td>Trace 32</td>
<td>Lauterbach</td>
</tr>
</tbody>
</table>
3.1 System Configuration step

In this step, we describe the procedure and result of the LDWS system configuration according to Step1 in chapter2.2. In addition, we show the model simulation and application logic generation activities using Simulink as well. First, we design software architecture complying with the SW component and the AUTOSAR standardized interface meta-model. And then we configure internal behavior of SW component. Internal behavior is designed by creating runnable entity, triggering event, point of data in/out, etc. Runnable entity is the smallest code-fragment and a subject for scheduling by OS. Fig. 3 is the LDWS software architecture in Autoworks. It describes SW components and their port-interfaces.

![Fig. 3. The LDWS software architecture in Autoworks](image)

<table>
<thead>
<tr>
<th>SW component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LaneDeSrc</td>
<td>Detect lane departure and notice to AlarmManager</td>
</tr>
<tr>
<td>InterDomain</td>
<td>Get turn signal from other ECU, Turn signal is transferred via CAN</td>
</tr>
<tr>
<td>AlarmManager</td>
<td>Decide to send warning signal to the driver by comparing departure direction and Turn signal</td>
</tr>
<tr>
<td>AlarmAct</td>
<td>Start to send warning signal to the driver, for example shaking the wheel or sound</td>
</tr>
</tbody>
</table>

After complete software architecture, we design hardware topology and communication matrix. LDWS is composed of three ECUs, lane detector, turn signal detector and warning controller. And then, we configure AUTOSAR CAN communication model, Signal-Pdu-Frame.

![Fig. 4. Hardware Topology in Autoworks](image)
After all configurations, we map the SWC to ECU, DataElement of AUTOSAR interface to real signal. The result of the system configuration step is SW component description, hardware resource description and system description.

To simulate on MIL level, we use the Simulink that support AUTOSAR software development. This process is out of scope of AUTOSAR. We can import SW component description (.arxml) in matlab command window and Simulink translates AUTOSAR meta-model to Simulink library. We use state-flow to simulate the LDWS function.

3.2 ECU Configuration step

In this step, we describe ECU configuration activities using Autoworks according to Step3, 4 in chapter2.2. LDWS include OS, SchM, EcuM, Com, PduR, CanIf, CanTrcv, CanDriver, Gpt, Mcu, Dio, Port and RTE. We edit relative parameters using Autoworks and the result is the ECU configuration description.

OS and RTE configuration is one of the most important part in ECU configuration phase because it is related with all over the scheduling attributes. In OS configuration time, we create tasks and configure priority, system stack size and others for each created task. And then, in RTE configuration time, we map runnable entity to task and configure the runnable entity’s position in task and OS event related with runnable entity. In mapping time, we consider the kind of event for runnable entity because that will affect the performance. For example, runnable entity with waitpoint has to be mapped to its own task. Because mapping multiple runnable entities with waitpoint to the same task can lead to big delay times if e.g. a waitpoint is resolved by the incoming event, but the task is still waiting at a different waitpoint.

Lastly BSW configuration code and RTE code is generated based on the ECU configuration description. RTE code includes application skeleton and OS taskbody, all API for interfaces and data access. BSW configuration code include variable code that dependent on configuration information.

3.3 HIL test

Test is the out of scope of AUTOSAR. But is is the normal process of an automotive application system development.

Our HIL test environment consists of CANoe, Trace32 debugger and the ECU containing AUTOSAR-based LDWS application. CANoe simulate the environment of the ECU by generating turn signal. Fig. 6 shows the HiL test environment of the LDWS system, developed following the AUTOSAR methodology.
4 Conclusion

In this study, we demonstrated all over the procedure of AUTOSAR application development. We showed the overview of AUTOSAR architecture and methodology, and described each step of development AUTOSAR application. AUTOSAR prepares standardized workflow, format and template to raise the reusability and reliability of the automotive software. So, the benefit of adopting AUTOSAR is the reduction of the period for developing new application by reusing application interfaces and BSW core functions. In our study, we reused some application interface provided by AUTOSAR and 50% of BSW parameter configuration that derived from prior project. Consequently we can develop the application in half the time of a prior application.

However AUTOSAR has to compensate its standard in the view of functional safety. Lately automotive functional safety is a hot issue in vehicle industry. According to ISO26262-automotive functional safety standard, AUTOSAR methodology is the part of “Development at the software level”. ISO26262 suggests various feature of software, for example, freedom of interference among software component or time critical operation and function. AUTOSAR must add new features to methodology and template to qualify ISO26262 for AUTOSAR based-application.

Finally, LDWS application in this study is the project of research purpose, so it’s difference from real production more or less.

Acknowledgments. This work was supported by the Industrial Strategic Technology Development Program funded by the Ministry of Knowledge Economy (MKE, Korea) [10041648, Robust Fault-Resilient SW for Vehicle Processors].

References