OSEK/VDX-Based Application Software Development Using Generic Embedded System

Y. Kim, T. Lee, S. Lee, S. Cho, and S. Jin

***Abstract*—OSEK/VDX (Open Systems and their Interfaces for the Electronics in Motor Vehicles/Vehicle Distributed eXecutive) is an open platform that has produced specifications for an operating system, a communication and a network management for automotive embedded systems to enhance software portability and reusability. Generic Embedded System developed by DGIST provides an environment to develop an automotive embedded software easily. In this paper, we introduce the Generic Embedded System and OSEK Development Tools and present a detailed case study to describe an example of OSEK-based automotive software application of RCS (Roomlamp Control System). Automotive engineers can learn easily OSEK OS application software and utilize this case study to other application software development through this case study.**

***Index Terms*—OSEK/VDX, automotive application software, generic embedded system, OSEK development tools, room lamp control system.**

I. Introduction

Nowadays, requirements for automotive development is increasing and therefore the number of embedded software in a vehicle is continuously increasing. So, complexity of automotive embedded software is also increasing so that development of an automotive embedded system consume much time and cost [1].

OSEK, which is one of methods to prove this problem, is an open architecture platform that has produced specifications for an operating system, a communication and a network management for automotive embedded systems to enhance software portability and reusability [2].

But actually, automotive engineer consumes much time and effort to understand related specifications and improve software development abilities. For this reason, it is increasing the need for the systematic organization of theoretical knowledge of operating system and development process for application software.

In existing research, engineer has been also developing an operating system, device driver and application software based on an evaluation board at early stages of development. However, this development method has many disadvantages for the operating environment and electronic signal to compare it with embedded system using real automotive environment. Therefore, this development method is difficult to satisfy function and performance of automotive embedded system.

Manuscript received August 5, 2013; revised October 18, 2013.

The authors are with the Robotics Research Division, Daegu Gyeongbuk Institute of Science & Technology (DGIST), Republic of Korea (e-mail: youngjae@dgist.ac.kr, tylee@dgist.ac.kr, shunlee@dgist.ac.kr, sr-cho@dgist.ac.kr, sungho@dgist.ac.kr; corresponding author: S. Lee).

Generic Embedded System and OSEK Development Tools developed by DGIST provide an environment to develop OSEK-OS based automotive embedded software and implement and various functions of real high-end passenger vehicle easily.

In this paper, we present a development method for OSEK OS-based automotive application software via case study using Generic Embedded System. It helps automotive engineers understand the OSEK standardization and develop new application software easily.

The remainder of this paper is organized as follows. In Section II, introduce development environment of OSEK OS-based automotive application software. In Section III, we present a case study to describe an application example of RCS using the Generic Embedded System. Finally, we'll conclude this proposal in Section IV.

II. Elopment Environment

The Generic Embedded System is Hardware Environment to design/develop an automotive body application system. This system has been developed using I/O Signals of electric output unit for real high-end passenger vehicle. It can help engineer to develop a real automotive embedded system. It is composed of ECU Hardware and I/O Simulator.

OSEK Development Tools is a training tool of OSEK/VDX for industrial engineers and university students. It can help them to understand OSEK/VDX concepts. It is composed of OSCAR-OSEK Designer and Training Program & Textbook.

Fig. 1. Composition of OSEK development tools.

A. ECU Hardware

ECU Hardware is one of Generic Embedded System for automotive body domain. It is designed to be suitable for automotive WCS (Window Control System), MCS (outsideMirror Control System) and RCS. This system is used in automotive class 16bit MCU and Driver IC for window motor and outside mirror. It provides various automotive I/O

DOI: 10.7763/IJIMT.2014.V5.476

signals (DIO, ADC, PWM, etc.) and vehicular communication network interfaces (CAN, LIN, FlexRay).

B. I/O Simulator

I/O signal control of automotive embedded software development should use real components to provide the real automotive environment. For this reason, we have developed I/O simulator, which is the other Generic Embedded System, for automotive body system. To develop this simulator, we analyzed electrical signal of automotive components with built-in extra features and designed various layouts for automotive body system.

C. OSCAR-OSEK Designer

OSCAR-OSEK Designer provides an OSEK OS environment based on OSEK/VDX Standardization an automotive embedded software easily [3]. This design tool has developed by OSEK/VDX standardizations [4]-[6].

It is possible for an OSCAR - OSEK Designer to create/modify/delete object information using OIL file and perform verification of OIL file and creation kernel code on GUI window. The application design flow of the OSCAR - OSEK Designer is shown as Fig. 2 [3].

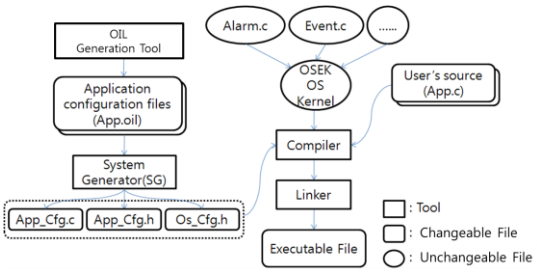

Fig. 2. Application design flow of OSCAR-OSEK designer.

D. Training Program and Textbook

The Training Program provides automotive engineers with a case study to develop OSEK OS-based automotive application software easily and systematically. It consists of an automotive body control system which includes MCS, WCS and RCS. This program is composed of requirement analysis, OS configuration, driver setting and software design to study step by step. Training Textbook provides curriculum for the OSEK development environment, instruction of OSEK designer and implementation/ verification methods for automotive body control system (outside mirror, window, roomlamp).

III. Case Study: RCS

In this section, we'll present a case study of implementation and application methods for RCS using OSEK Development Tools.

A. Signal Analysis of RCS

We use only three inputs of ECU hardware. DoorOpen switch is used to change open/close state of the door. IGN switch is used to input current ignition key state. The analog sensor switch is used to monitor outdoor brightness. Signal of switch inputs is transmitted from MCU circuit to TLE8201 chip. This chip controls roomlamp by SPI communication.

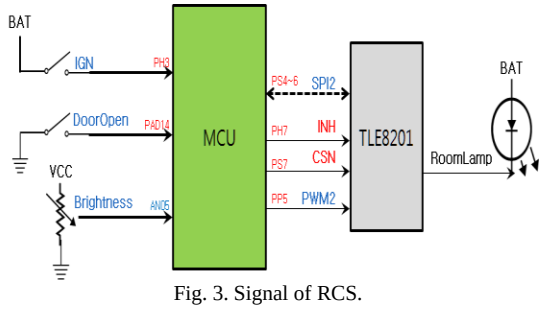

Fig. 3. Signal of RCS.

B. Setup for Development Tools

We consider that input of the passenger ECU module as RCS is developed by network-based ECU System via CAN communication. Main features of this RCS system are not only ON/OFF control but also Dimming and Auto control. To develop this intelligent RCS, we used Generic Embedded System with two ECU Hardware and one I/O simulator.

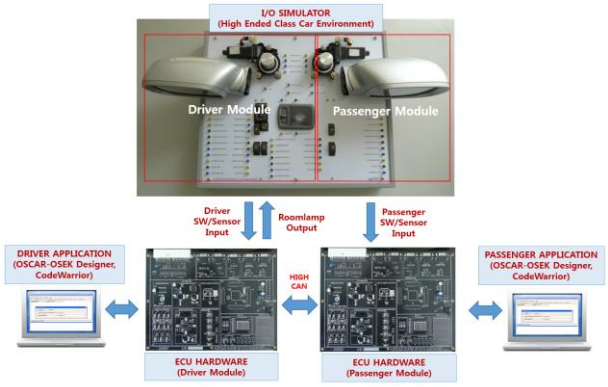

Fig. 4. KIT configurations for RCS.

C. Development Process

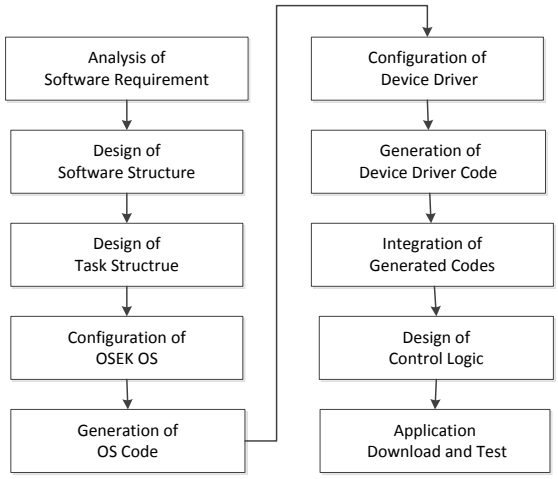

Fig. 5. Development Process of RCS.

Development process of OSEK OS based embedded software is shown as Fig. 5. Analysis of Software requirement specifies functional requirements of software systems and defines the goals of software development. Step of design of software structure composes details of software structure which is possible to convert real embedded program based on analysis of requirements. Step of task structure design means that complex software is divided into tasks which are programmed segment to perform based on

real-time requirements. Step of OSEK OS design configures OS and task information, task resources and event/alarm information. After configuration of OSEK OS, we generate OS code through the code generation function of the OSCAR - OSEK Designer. Generated codes integrate with OSEK kernel and library files using integration tools when configuration and implementation of device driver based on ECU specification is completed. After code integration, we design control logics to implement the actual functions of task. Finally, we download completed integration program to target ECU and test whether the functions of the system is correctly implemented.

D. Software Requirement

RCS is automotive equipment for convenience that can obtain safety and vision when vehicle is parked and stopped. Software requirement of proposed RCS is shown as Table I.

TABLE I: SOFTWARE REQUIREMENT OF RCS

Door State		IGN State		Roomlamp
Previous	Current	Previous	Current	Output
Close	Close	-	-	Off
-	-	Off	Off	On
Open	Close	Off	Off	Dimming
Close	Close	On	On	Auto

E. Software Structure

The software structure of RCS consists of application program, operating system and device driver like Fig 6. ECU hardware is located directly below this software. The application program is implemented by tasks which are performed in accordance with the real-time requirements. Proposed RCS is implemented by five tasks. OSCAR-OSEK using this system is a real-time operating system which manages tasks with priority. The device driver is an interface between operating system, application program and ECU hardware. Purposed system consists of seven device drivers.

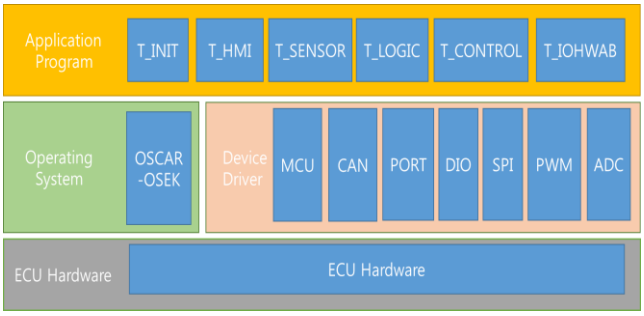

Fig. 6. Software structure of RCS.

F. Task Structure

Task structure of RCS is shown as Fig. 7. This system uses internal message to transmit information to other tasks. T_INIT task is the first task to be executed only once when OS starts up. T_INIT task calls Initialization Function, activate Task and alarm. T_IOHWAB task is an I/O hardware abstraction task. The aim of this task is to provide ECU hardware independent data transition from driver modules up to other tasks. T_IOHWAB task supports a complete abstraction of the DIO, ADC, PWM, SPI driver in this system. T_HMI task receives periodically input status messages of automotive doors and ignition switch to T_IOHWAB task. T_HMI task analyzes received messages to determine automotive door and ignition state. T_SENSOR task checks ADC of outside brightness from T_IOHWAB and transmit outside brightness state to T_LOGIC task. T_LOGIC task is activated with messages to send T_HMI and T_SENSOR task. This task has a logic of roomlamp control mode to perform control logic algorithm. T_CONTROL task analyzes messages of T_LOGIC task so that it determines roomlamp brightness. T_CONTROL task requests change of roomlamp brightness to T_IOHWAB task. Property of tasks and managed massages is shown as Table II, Table III.

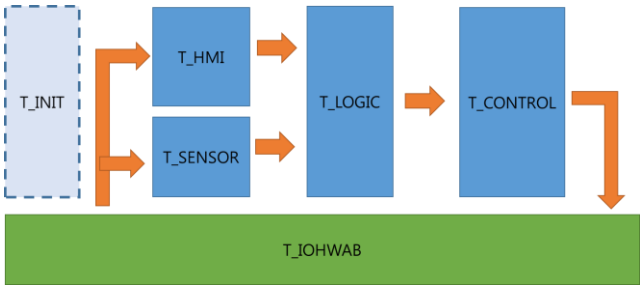

Fig. 7. Task structure of RCS.

TABLEI I: PROPERTY OF TASKS

Name	Type	Priority	Autostart	Activation
T_INIT	Basic	2	Yes	AutoStart
T_HMI	Basic	4	No	Message
T_SENSOR	Basic	6	No	Message
T_LOGIC	Extended	8	No	Event
T_CONTROL	Extended	10	No	Alarm&Event
T_IOHWAB	Extended	12	No	Alarm

TABLE III: LIST OF INTERNAL MESSAGES

Message	SENDOR	RECEIVER	NOTIFICATION
DOOR_MSG	T_IOHWAB	T_HMI	ACT_TASK
IGN_MSG	T_IOHWAB	T_HMI	ACT_TASK
ADC_MSG	T_IOHWAB	T_SENSOR	ACT_TASK
HMI_MSG	T_HMI	T_LOGIC	SET_EVENT
SENSOR_MSG	T_SENSOR	T_LOGIC	SET_EVENT
LOGIC_MSG	T_LOGIC	T_CONTROL	NONE
CONTROL_MSG	T_CONTROL	T_IOHWAB	NONE

G. Development Result

We integrated generated code with OSEK OS kernel and device driver using CodeWarrior tools which was an integration development environment of Freescale and implemented task control logic codes. After downloading roomlamp application software on ECU hardware, we performed logic level tests to check LED lighting. Finally, we connected the ECU Hardware to the I/O simulator with high-end class passenger vehicle environment and tested verification and validation of I/O signal control on the actual load signal level. We could see that proposed system satisfied software requirements via roomlamp of I/O simulator through this test. As a result, we could develop easily and quickly OSEK based application software using OSEK Development Tools through case study.

IV. Conclusion

In this paper, we introduced the Generic Embedded System and presented a detailed case study for OSEK OS-based automotive application software using this development tool. Automotive engineers which have less experienced automotive embedded system can learn easily OSEK OS application software and utilize this case study to other application software development. In the future, we'll develop OSEK based training program which departmentalize development level to integrate automotive engineers with different experiences in university, institute and enterprise.

Acknowledgment

This work was supported by the DGIST R&D Program of the Ministry of Science, ICT and Future Planning of Korea (13-RS-03).

References

[1] S. Patil and L. Kapaleshwari, "Embedded Software – Issues and Challenges," SAE Technical Paper 2009-01-1617, 2009.
[2] OSEK Group. [Online]. Available: http://portal.osek-vdx.org.
[3] OSCAR-OSEK user manual, DGIST, Republic of Korea, 2012.
[4] OSEK Group. OSEK/VDX Operating System Specification 2.2.3. [Online]. Available: http://portal.osek-vdx.org.
[5] OSEK Group. OSEK/VDX Communication Version 3.0.3. [Online]. Available: http://portal.osek-vdx.org.
[6] OSEK Group. OSEK/VDX System Generation OIL: OSEK Implementation Language Version 2. [Online]. Available: http://portal.osek-vdx.org.
[7] W.-K. Chen, *Linear Networks and Systems*, Belmont, CA: Wadsworth, 1993, pp. 123-135.

Youngjae Kim received the B.S. and M.S. degrees in electronics engineering from Kyungpook National University, Daegu, Korea, in 2008 and 2010 respectively. Since 2010, he has been a Researcher with Daegu Gyeongbuk Institute of Science and Technology, Daegu, Korea. His research interests include automotive embedded systems, modeling and simulation for electric vehicles and motor application system.

Taeyoung Lee received the B.S. degrees in electronics engineering from Keimyung University, Daegu, Korea, in 2008 respectively.

He received the M.S. degrees in electronics engineering from Kyungpook National University, Daegu, Korea, in 2010 respectively.

Since 2010, he has been a researcher with Daegu Gyeongbuk Institute of Science and Technology, Daegu, Korea. His research interests include automotive embedded systems, modeling and simulation for electric vehicles and motor application system.

Seonghun Lee received the B.S., M.S. and Ph.D degrees in electronics engineering from Kyungpook National University, Daegu, Korea, in 1996, 1998 and 2007 respectively.

From 1999 to 2002, he was a Junior Research Engineer with Daewoo Precision, Busan, Korea.

From 2002 to 2005, he was a senior researcher with Agency for Defense Development, Daejeon, Korea.

Since 2005, he has been a senior researcher with Daegu Gyeongbuk Institute of Science and Technology, Daegu, Korea. His research interests include automotive embedded systems, modeling and simulation for electric vehicles and motor application system.

Sungrae Cho received the B.S. degree in computer engineering from Kyungil University, Gyeongsan, Korea, in 1998. And he received the M.S. degree in computer engineering from Yeungnam University, Gyeongsan, Korea, in 2000. Since 2005, he has been a Senior Researcher with Daegu Gyeongbuk Institute of Science and Technology, Daegu, Korea. His research interests include real-time operating system, automotive embedded software and software testing.

Sungho Jin received the B.S. and M.S. degrees in electronics engineering from Kyungpook National University, Daegu, Korea, in 1989 and 1991 respectively.

He is a principal research engineer with Daegu Gyeongbuk Institute of Science and Technology, Daegu, Korea. His research interests include automotive embedded systems, modeling and simulation for electric vehicles and motor application system.

.