ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 8, August 2016

Software Design and Implementation for the Electronic Parking Brake System Based on Operating System

Jaeseung Hong[1], Seonghun Lee[2], Donghoon Ban[2], Daehyun Kum[2], Soohyun Kwon[1]

Researcher, Convergence Research Center for Future Automotive Technology, DGIST, Republic of Korea[1]

Senior Researcher, Convergence Research Center for Future Automotive Technology, DGIST, Republic of Korea[2]

ABSTRACT: Electronic Parking Brake(EPB) system is operated by Electronic Control unit(ECU). EPB system has been by replaced a traditional parking brake system, which is a mechanical device to be driven by a parking brake lever or foot pedal. EPB system is more convenient for the women and elderly. Interior design as well as space utilization has also been important recently. By using EPB system, it is possible to ensure more space in a vehicle. Also, EPB system provides better convenience and safety than the traditional system through the cooperative control of Electric Stability Program(ESP) or alone. In this paper, we propose methodology of software architecture design and function development for the EPB system based on software tools.

KEYWORDS: EPB, Automotive, Embedded System and OSEK OS

I. INTRODUCTION

In recent years, the increasing electronic components in the vehicle make unexpected problems. The most typical problem is the increasing software complexity of the electronic system. Embedded systems are a combination system of hardware and software. It requires more complex software in order to perform various electronic components. The increasing software code raises the complexity of the software. It can cause some problems which are not expected during the development phase.

In the early 1990s, European automotive companies together enacted the OSEK standard for reliability and reusable of automotive software. OSEK/VDX is the official name. It was made by merging an independent project of OSEK and VDX.

The OSEK standards consist of 8 parts. In this paper, we use the OSEK OS for automotive software architecture design. The OSEK OS has been developed for Real-Time operating system. It is very suitable for automotive software architecture.

An EPB is controlled by an electronic control actuator. It has replaced the traditional parking brake system, which is a mechanical device driven by a parking brake lever or a foot pedal. When a Vehicle is designed, the utilization of interior space can be increased by using an EPB system. Also, the EPB system provides better convenience and safety than the traditional system through the cooperative control of ESP or through sole control. In this paper, we propose methodology of software architecture design and function development for the EPB system based on software tools.

II. OSEK/VDX

In order to understand software architecture of the EPB system based on the OSEK OS, basic knowledge is required for the OSEK platform. So, we explain simply the OSEK platform by reference [1] in this chapter.

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 8, August 2016

A. OSEK OS

The OSEK OS is a real-time operating system for automotive software. It is a very important part of the OSEK/VDX standard. By introducing the operating system to automotive software, not only portability of applications and interoperability are enhanced but also code reuse and the independence of the hardware are improved.
In the chapter, we explain task scheduling technology in the OSEK/VDX Operating System Specification 2.2.3.

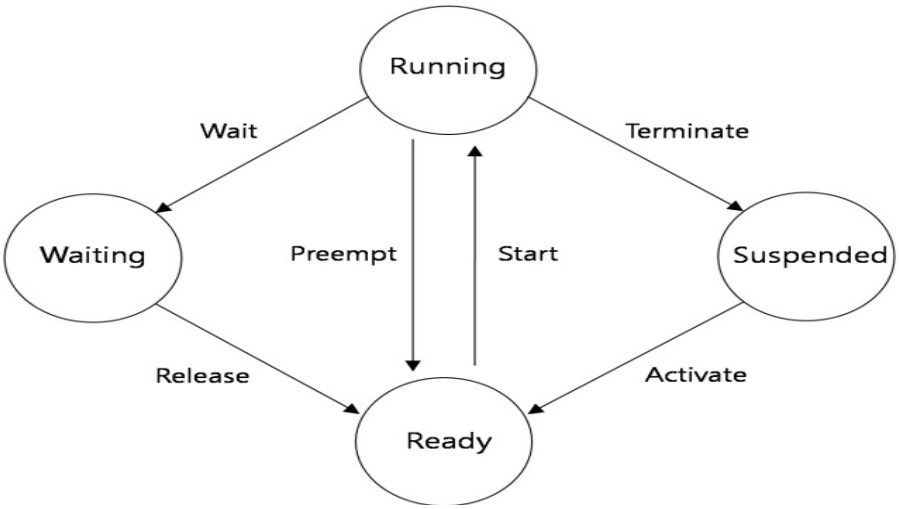

Fig 1OSEK OS Task State Model

Fig 1 shows four types of conditions and changes which OSEK OS has. On single ECU, multiple tasks are concurrently carried out and the status of each task is moved in accordance with a predetermined schedule. The meaning of each state is as follows.

1. Running - The CPU is assigned to the task, so that its instructions can be executed.
2. Ready - All functional prerequisites for a transition into the running state exist, and the task only waits for allocation of the processor.
3. Waiting - a task cannot continue execution because it shall wait for a least one event.
4. Suspended - The task is passive and can be activated.

B. OSEK OS SCHEDULING

The scheduler changes the highest priority task to the running state from the ready state after measuring the task priority. The priority number 0 represents the lowest priority and higher number is higher priority. All priorities are statically assigned at the system design stage and does not change except special situations such as using system resources and call interrupt.
OSEK OS uses the pre-emptive scheduling. This strategy can arrange the execution permission of running task. If there is higher priority task than the present running task in the ready queue, the scheduler hands the execution permission over to another higher priority task

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 8, August 2016

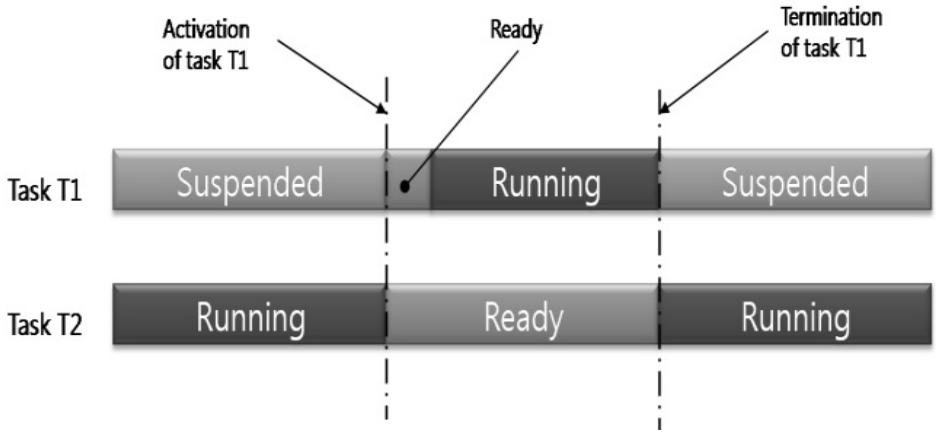

Fig 2Full Pre-emptive Scheduling

Fig 2 and 3 show the scheduling policy of OSEK OS, which are defined in the OSEK/VDX Operating System Specification 2.2.3. The scheduler searches tasks in each priority queue and then transfers the highest priority task to running state. As a result, Fig. 3 shows the example. A task of priority 2 is able to be moved to running state after finishing three tasks of priority 3

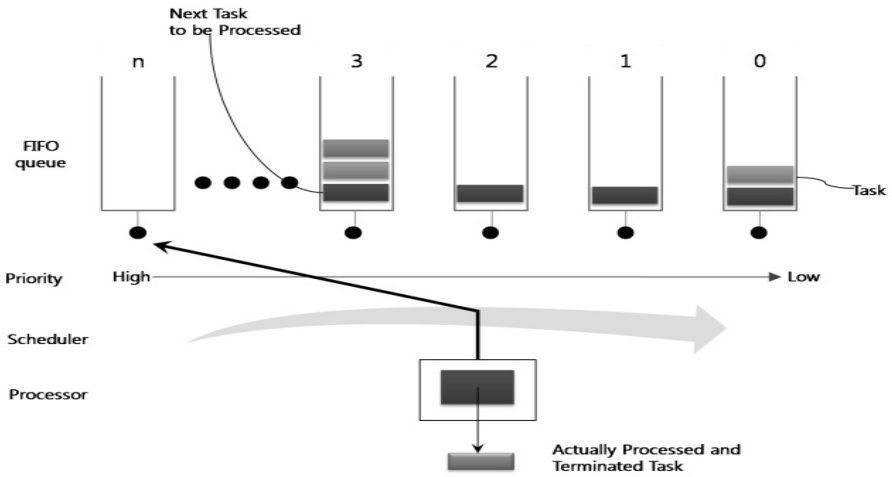

Fig 3The scheduling policy of OSEK OS

C. OSEK DEVELOPMENT PROCESS

Fig 4 explains how to develop automotive software using the OSEK OS. At this state, we are able to use proven auto generated platform modules and systems except for the user of the source code. The OSEK Builder is used in the system design and using the System Generator compile available sources are generated using the System Generator through the system. However the User's source codes as an application for implementing the functions of the system, it have been written by hand coding using traditional program language such as C/C++ so far.

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 8, August 2016

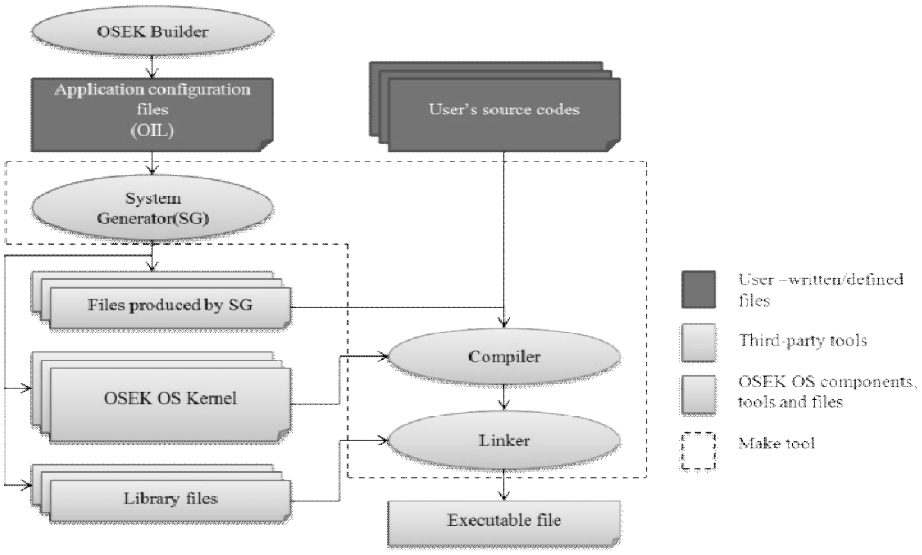

Fig 4OSEK OS Implementations

III. EPB SYSTEM

Recently, the automotive industry has shown increased interest in intelligent vehicles for better user convenience. Therefore the X-by-wire system has been developed, using electronic components such as electronic sensors, actuators or an ECU. The X-by-wire system has replaced many mechanical systems. With the X-by-wire system, we are able to predict a developmental cost decrease and an increase in user convenience and flexibility by extension. TheEPB system, a type of X-by-wire system, has steadily been replacing the existing mechanical parking brake system.
The EPB system is dramatically improved in terms of user convenience and reliability compared to the existing system. A parking lever or foot pedal is not necessary when using the EPB System. Fig 5 illustrates the basic structure of the EPB System.

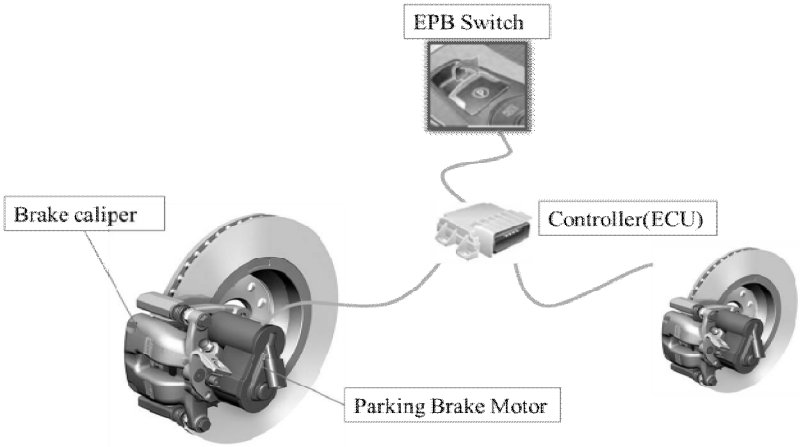

Fig 5EPB Systems

This creates more interior space in a motor vehicle. If the EPB system exists in the vehicle, this level of power can be obtained easily by pressing a button. This is a great advantage for the elderly and infirm. In addition to these

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 8, August 2016

advantages, through cooperative control of the ESP system or via the single operation of the EPB, enhanced additional functions for convenience and safety can be realized.

IV. THE EPB SYSTEM SOFTWARE DESIGN

A. DESIGN OF THE SOFTWARE ARCHITECTURE FOR EPB SYSTEM BASED ON OSEK OS

Based on the OSEK OS, the software structure consists of various modules. Figure 6 shows the software architecture of the EPB system designed by us. These include application tasks (Init, Input, Run, and Warning), the in-vehicle network driver (CAN), the I/O device driver (ADC, PWM, and DIO) and the interrupt service routine. Each module is connected to each other module, and the arrow in the figure denotes the control flow for other modules using events, messages or function calls. The task modules are granted priority by the OSEK OS to operate in a multi-tasking manner. Each task transmits/receives events and messages to synchronize. The CAN ISR (interrupt service routine) serves to receive CAN signals.

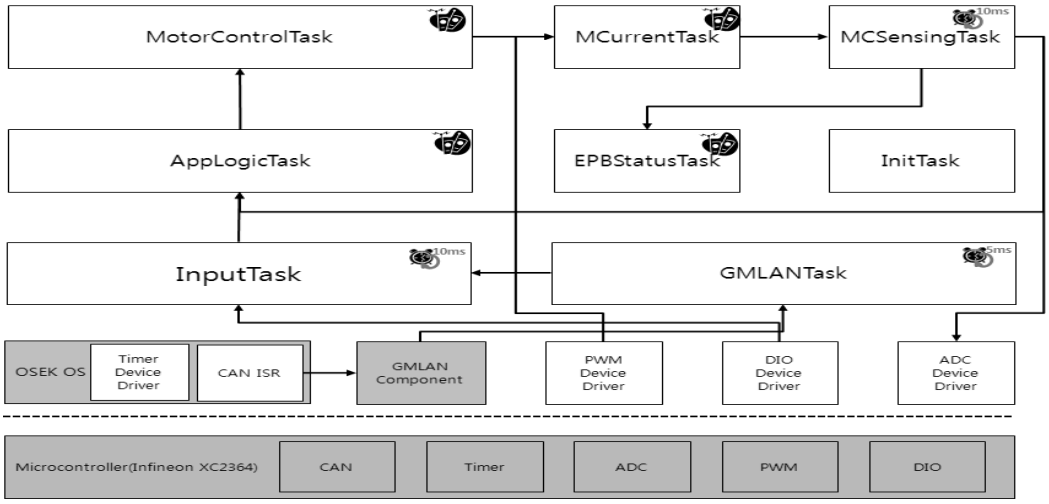

Fig 6Design of Software Architecture for EPB System Based on OSEK OS

1. InputTask: receiving input signals from various I/O drivers to pass to the AppLogicTask
2. AppLogicTask: activating the MotorControlTask after determining the behaviour of the EPB system by input signals
3. MotorControlTask: turning the motor to the forward direction or the reverse direction using the PWM device driver by receiving events
4. MCurrentTask: deciding whether to stop the motor while the motor is turning and sending the status of the EPB system using an EVENT to the EPBStatusTask
5. MCSensingTask: monitoring the current every 10ms using the ACD device driver
6. EPBStatusTask: informing other tasks of the status of the EPB system
7. ADC Device Driver: sensing the battery level and EPB system temperature using the ADC device
8. DIO Device Driver: controlling the digital input/output
9. PWM Device Driver: operating a calliper to apply or release using the PWM device
10. CAN ISR (Interrupt Service Routine): managing the received CAN signals; it is configured by the OSEK OS configuration tool.

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 8, August 2016

B. Functional Verification and Automatic Code Generation

Recently, development methodology has attracted attention with Matlab/Stateflow as shown in Fig 7. It is very useful to verify the software algorithm in advance. We are able to remove the logical errors beforehand through a simulation.

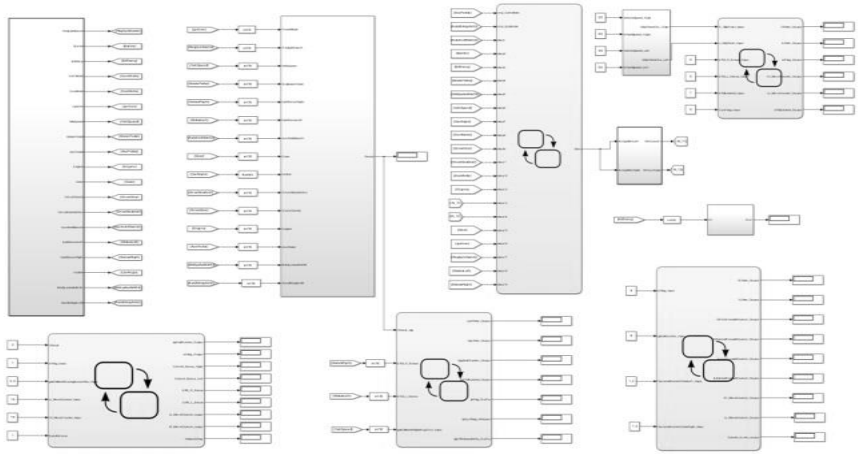

Fig 7Application Development using the Matlab/Stateflow

In addition, it is possible to generate a compliable source code automatically using the code generator after the simulation. Fig 8 shows an automatic code generation in C code level. We are able to compile this C code immediately after integrating into the project file.

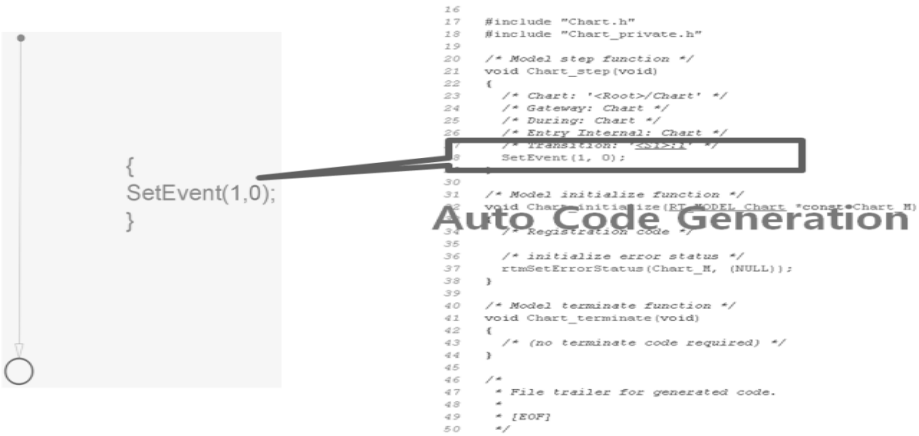

Fig 8Automatic Code Generation

C. Functional Operation Flowchart and Placed on the Operating System

We need to assigned functions to each task after the architecture design. Fig 9 shows that how to arrange functions in designed architecture. Tasks need some method to communicate with each other. OSEK OS provides the EVENT and MESSAGE mechanism for communication with each task.

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 8, August 2016

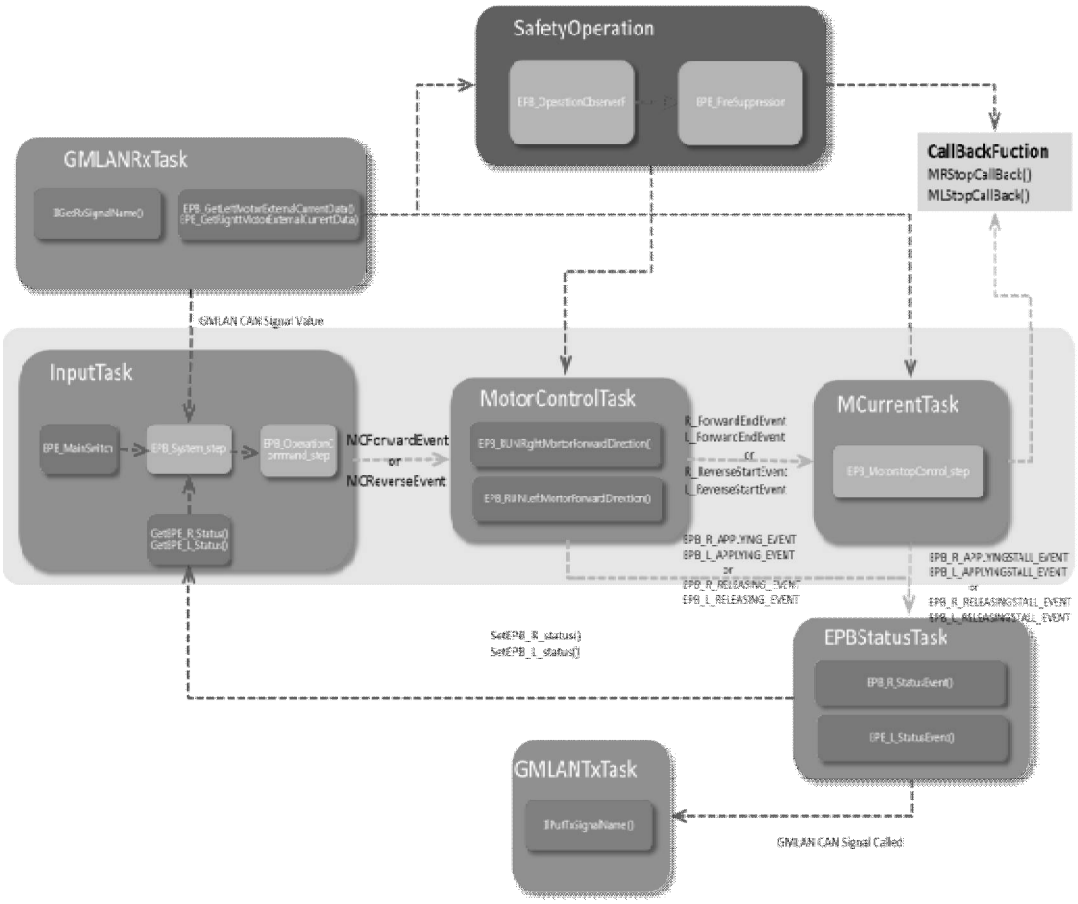

Fig 9Flowchart and Placed on the Operating System

V. CONCLUSIONS

The EPB System operates with several of module together such as I/O device, CAN Network, Actuator Control Logic and Tasks. If the software of this system is developed as firmware, we have to consider the synchronization between the behaviours of each module carefully in the design phase. However as we develops the software of EPB system based on OSEK OS in this research, each module is defined as a task and the synchronization between modules is controlled easily using the EVENT, ALARM and MESSAGE which are provided by OS API. Also, we are able to improve the safety and reliability of the system by using the validated and standardized Real-Time OS.

ACKNOWLEDGEMENTS

This work was supported by the DGIST R&D Program of the Ministry of Science, ICT and Future Planning(16-RS-03).

REFERENCES

[1]
[2] Jaeseung Hong, Daehyun Kum, SunghoJin, Jeonghun Cho, A Method of Optimizing the Scheduler for Improving the Performance of the OSEK OS, Applied Mechanics and Materials,Vol. 110 – 116, 2011, P 5141-5145
[3] OSEK, 02.17.2005, "OSEK/VDX Operating System Specification 2.2.3"
[4] OSEK, 2004, "OSEK/VDX Implementation Language (OIL) 2.5"
[5] J.-L. B´echennec, M. Briday, S. Faucou, and Y. 2006, Trinquet. Trampoline: An OpenSource implementation of the OSEK/VDX RTOS specification. In IEEE Conference on Emerging Technologies and Factory Automation, 2006. ETFA '06., pages 62.69

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 8, August 2016

[6] Sungrae Cho. Hyunki Ryu1. SunghoJin. Junho Lee., An Implementation of Automotive Operation System based on OSEK/VDX and Methods for Ensuring the Reliability of that Implementation, KSAE 2009 Annual Conferencem, 2009
[7] Yi-qiang PENG, Wei LI, Research on Fuzzy Control Strategies for Automotive EPB System withAMESim/Simulink Co-simulation, Control and Decision Conference, p1707-1712,2009
[8] Liao, Y., Huang, C., Chen, C., Cheng, S. et al., Novel Design of the Integrated Electronic Parking Brake System, SAE Technical Paper 2010-01-1707, 2010, doi:10.4271/2010-01-1707.
[9] Jaeseung Hong, Daehyun Kum, SunghoJin, 2013. Software Architecture Design for the Electronic Parking Brake Based on OSEK OS, 2013 International Symposium on Embedded Technology