Proceedings of the 17th World Congress
The International Federation of Automatic Control
Seoul, Korea, July 6-11, 2008

The Software Platform Development of a New Microcontroller for Automotive Body Systems

Jae Ho. Chang*, Chan Woong. Park*, Chan Hong. Park*, Sang il. Lee*, Jin Su. Jang*
Jong Hyun. Baek*, Jeong Ho. Son*, Dae Hwan. Lim*, Kyung Tae. Kim*

**Car Electronic Architecture & Networks System (CARNES) Company Ltd. 1043, Hogye-dong, Dongan-gu, Anyang-city, Kyunggi-do, Korea (Tel: 82-31-389-0333; e-mail: jhjang@carnes.co.kr).*

Abstract: A new software platform (SWP) for automotive body electronic systems is introduced in this paper. This platform consists of three parts which are run time environment (RTE), generic layer and hardware abstraction layer (HAL). The RTE is a kind of dynamic interface layer to connect the application to basic software. The generic layer is independent of hardware and is normally not changed even though a microcontroller is changed. The HAL is a layer which depends on hardware and should be modified if the microcontroller and hardware configurations are changed. Our efforts are mainly focused on the development of RTE, HAL, and a few generic components for our new software platform. Some basic technologies such as configuration concept and code generation are acquired through this project. SWP configuration tool as well as SWP itself was developed for the convenience of application design based and SWP. SWP validator is also implemented for the automatic validation of various SWPs that will be developed in the near future. Finally, our new software platform shows that the reuse of applications can be realized by the new technologies of configuration concept. That was indirectly proven by SWP validator which performs same diagnostic software on two different platforms.

1. INTRODUCTION

Automotive electronics has been growing up for the user's convenience and to increase the reliability of the vehicle with the help of technology improvement of the microcontroller in the semiconductor industry. The more cost-effective high performance microcontrollers enable electronic modules of the vehicle to replace a lot of old mechanical functions. These days there may be up to 70 electronic control units (ECUs) in a high grade automotive which are responsible for controlling major electric functions of the car. These many ECUs cause the cost, reliability, complexity and interaction problems between themselves in the vehicle electrical and electronic systems. To overcome these many problems and to improve the reliability of the product, the reuse of the software modules, which was already proven in the previous mass production and can be applicable to a new ECU again, has been an important issue. Nowadays car manufacturers and Tier 1 suppliers have applied their individual basic software standards to their products to satisfy the requirements of the software reuse. This type of basic software is usually a kind of local standard which has to be maintained and integrated individually. This type of individual approach needs more time for verification and validation of the product. It is not easy to reuse the software modules for specific vehicle applications without any modification since their basic software is not standard and has different architecture. To reduce the efforts for software development and integration time for verification and validation, leading automotive manufactures and suppliers founded the automotive open system architecture (AUTOSAR) initiative in 2003 that aims to develop and standardize basic software architecture for automotive ECUs(AUTOSAR initiative, 2006). They are also trying to launch AUTOSAR standard platform into their products (Kohler, 2006).

The CARNES software platform introduced in this paper is a kind of local standard for automotive body electronic systems. However our platform is much flexible and has the features of configuration concept, code generation and standard API and so on. In other words, our software platform has almost same concept as the standard AUTOSAR platform but only its specification is somehow different from that. This platform consists of three parts which are RTE, generic layer and HAL. The RTE is a dynamic software interface layer generated by SWP configuration tool. This layer plays a role to connect applications to our platform and enables application software modules to be independent of hardware. Therefore, the possibility of the software reuse can be increased by using these configuration concepts. The generic layer includes operating system, flash boot loader (FBL), core, memory (MEM), communication modules (COMM), input and output (I/O) and power management (PM). These parts are independent of hardware and used in common for all software platforms. The HAL is the layer depends on hardware. This part should be modified when the microcontrollers are changed. In this paper, our research is mainly focused on the development of HAL, RTE and the encapsulation of a few generic components for a new microcontroller. Also, related design tools such as SWP configuration tool and SWP validator are designed and implemented to approach the AUTOSAR standard in the near future.

The purpose of this paper is to show that new technologies and design approaches can be successfully applicable to the development of automotive ECUs. In section 2, the integrated environment for SWP development is explained. Section 3 explains RTE, generic layer and HAL for a new microcontroller. The design procedure called V-cycle used for our SWP development is also introduced in brief. The SWP configuration tool and new technologies are disclosed in section 4. The overall explanation about SWP validator and the possibility of application reuse were briefly presented in section 5. Conclusions and further works are summarized in section 6.

2. INTEGRATED DEVELOPMENT ENVIRONMENT (IDE) FOR SWP DEVELOPMENT

The integrated development environment (IDE) for SWP development is explained in this section. Originally this was introduced into CARNES from Siemens. However this IDE has been updated and modified by ourselves for the development of CARNES software platform.

2.1 The IDE for SWP development

The purpose of IDE for SWP development is to manage version, change and quality of platform software from the initial stage of the development for mass production. The IDE comprises the tool to manage version and change of software, text editor for programming, quality tool for checking the reliability of the code, compiler and linker that depends on microcontrollers and code generation tool to create the interface layer between application and SWP etc. The interaction among all these tool components is administrated by script language. This IDE sometimes should be modified for a new project because some parts like compiler and linker of the IDE depend on microcontrollers. The IDE used in this project was also modified to support the compiler corresponding to a new microcontroller. The configuration of our IDE is shown in Fig. 1.

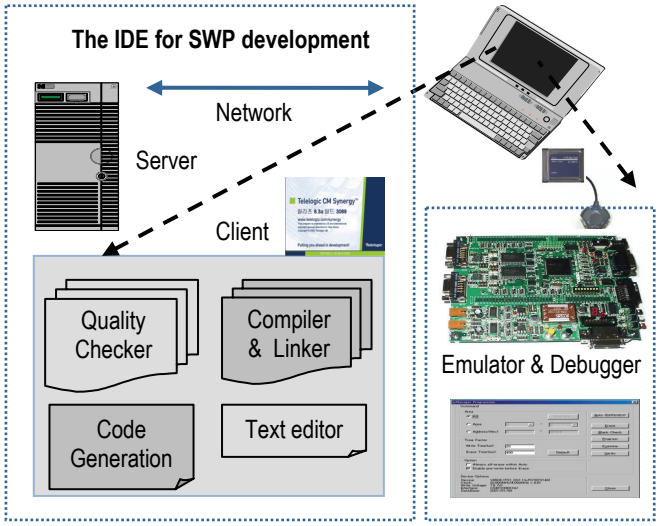

Fig. 1. The integrated development environment for SWP development

2.2 Emulator and debugger for real debugging

For real debugging of platform as shown in Fig. 1, the extra equipments including emulator and debugger are needed for each microcontroller because our IDE was focused on only managing software and creating executable file before final execution on the target is performed. Our IDE can be used for all projects related to software development if some parts depending on hardware are supplemented on necessity.

3. SOFTWARE PLATFORM DEVELOPMENT

In this section CARNES software platform is briefly outlined . At first original software platform was provided to CARNES from Siemens. It was modified and upgraded for the use of automotive body electronics. The mass production of original software platform with upgraded features such as Vector CAN encapsulation, KWP2000 protocol and software update module via K-line and CAN etc. is now in progress. This section just describes the software platform we have newly developed for covering from low-end to high-end ECUs. It is divided into three layers which are RTE, generic layer and HAL. Fig. 2 is the overall block diagram of our platform.

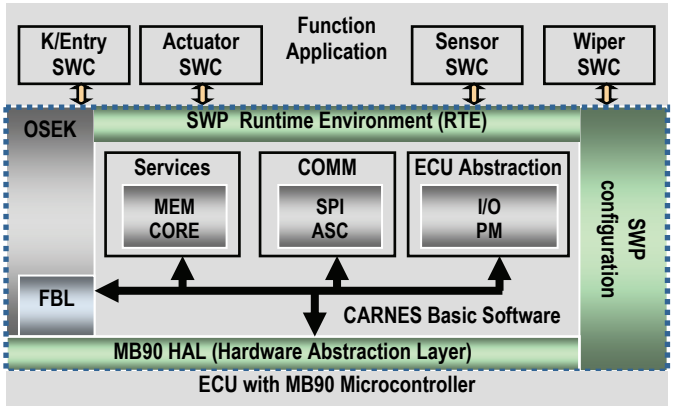

Fig. 2. The overall block diagram of CARNES software platform

3.1 Run Time Environment (RTE)

The RTE is created by SWP configuration tool. This layer is a kind of dynamic layer to interface applications with software platform. It is the first local standard to introduce a type of middleware to automotive ECUs. The RTE makes the applications designed by different suppliers reusable by introducing the concept of virtual function bus decoupling the applications from the basic software. The roles of RTE include hardware configuration, the configuration of generic layer and the configuration of applications. This type of approach is one of the core technologies that have been used for AUTOSAR standard platform.

3.2 Generic layer

The generic layer is independent of hardware. This layer is always same even if a microcontroller is changed. Therefore,

most of the components contained in the generic layer were reused. However, some functions were redesigned and modified. First, the operating system was changed and encapsulated into our SWP because the operating system used in the original platform does not support a new microcontroller selected in this project any more. Second, 3rd party software stack like Vector CAN was successfully encapsulated into our SWP. Related CAN station manager already developed in the previous project was slightly modified and perfectly transferred into our SWP. Third, the key word protocol 2000 (KWP2000) which is one of the ISO standards also became a part of the generic layer for diagnosis. This software stack was already developed and proved as an application in the previous project. Of course, this component was redesigned and thoroughly verified as a part of our SWP. As a result KWP2000 has become an indispensable element of our SWP for the diagnosis of automotive ECUs. Additionally, the software update modules were newly developed including PC download utility for reprogramming application software via CAN or K-line in the field.

3.3 Hardware Abstraction Layer (HAL)

The hardware abstraction layer (HAL) relies on hardware. In the selection of microcontroller, all information from car manufacture and Tier1 suppliers was gathered up to decide which microcontroller is optimum for automotive body electronics. In the end, Fujitsu MB90 series was chosen to cover from very low cost ECU such as assistant door module (ADM) to high-end ECU like body control module (BCM). The performance of this controller is not enough to implement high integrated BCM which has multi-functions of smart key, tire pressure management system as well as BCM itself in the near future. However, the development of HAL for MB90 was started because the cost is very sensitive factor for mass production. All components of the HAL for OS, FBL, MEM, IO, CORE, COMM, CAN and PM were developed. From the initial stage all development procedure follows the well-known V-cycle approach [2]. Therefore all documents related to SWP development consist of software requirement specification (SRS), software design document (SDD), coding, software verification and validation plan (SVVP) and software verification and validation report (SVVR) per each function module. About 200 documents were created throughout the SWP development.

4. SWP CONFUGURATION TOOL

The configuration concept is one of the most important technologies. This technology can give the SWP the feature of the reuse of application software based on SWP by introducing configuration concept. The development of this tool is the first step to reach the AUTOSAR standard platform.

3.1 The overview of configuration tool

CARNES developed this tool by himself using JAVA and XML technologies used extensively in computer areas. This type of tool is already widely used for integrating lots of applications with 3rd party software like operating system. The role of this tool is to generate C sources and header files which combine applications with software platform as shown in Fig. 3 for easy understanding. In other words, the RTE explained in subsection 3.1 is the combination of generated C sources and header files. The characteristics of this layer are dynamic and changed every time per each application.

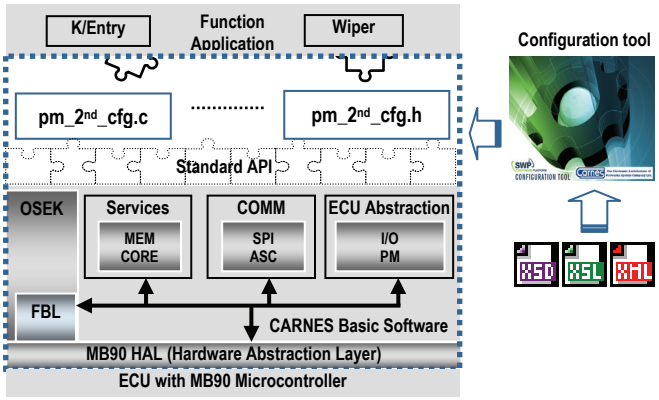

Fig. 3. The creation of RTE by SWP configuration tool

The inside of SWP configuration tool is disclosed in Fig. 4. This tool has the various functions of XML project view, open project, consistency check and code generation etc. This tool separates the application software modules from basic software by introducing the concept of virtual function bus on any ECU.

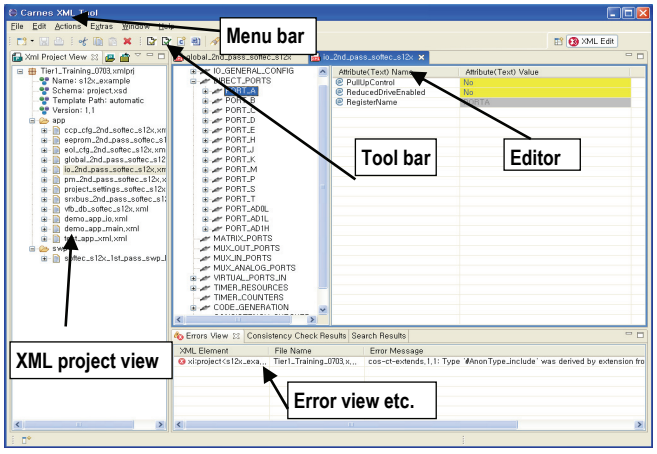

Fig. 4. The inside of SWP configuration tool

The concept of our configuration tool is the same as that of configuration tool for AUTOSAR standard platform. But the input interface file format is different from that of AUTOSAR platform configuration tool. Therefore the 1st version of our tool could not support AUTOSAR platform. To overcome this problem, we have updated our tool to give it the feature of interfacing input file format which AUTOSAR already defined. The development of our new tool was successfully completed and it has been being used for our next project of developing the 1st CARNES AUTOSAR platform.

3.2 Project support through configuration tool

It was decided for SWP to be applied to 12 ECUs of 5 vehicles for mass production. Separate projects are now in progress. The platform used for mass production is the original one with the updated features that Vector CAN, KWP2000 and some external device drivers like EEPROM are successfully integrated. After customized SWP by the requirement of Tier1 was created and converted into a library and then it is delivered to them with our configuration tool. The role of our configuration tool is to provide their applications with high transferability reducing mistakes of the developer. Tier1 suppliers try to reuse their application by means of our configuration tool. Once their applications are successfully launched into our SWP, it will be easily transferred again by configuration tool under SWP environment even though the SWP itself is changed as a new one.

5. SWP VALIDATOR

The topic about how to test and evaluate the SWP is discussed in this section. Many verification and validation tests should be performed before the delivery of SWP library. This job is very important for the reliability of ECUs. However this type of job requires much time, efforts and patience. Nevertheless, it is tedious and often causes many mistakes decreasing the reliability. In general, the development of SWP validator which is one of the automatic testing tools is the mandatory to reach the reliable SWP without faults.

5.1 The overview of SWP validator

The automotive ECUs usually require high reliability. To meet this requirement many Tier 1 suppliers have their own testing methods for their product. Our updated original platform and the newly developed platform should be tested as well before they are delivered to Tier 1 suppliers. A lot of time and effort is put into testing SWP library according to the increase of projects and vehicle events. To avoid a waste of time and effort, CARNES has developed an SWP validator which is one of the automatic testing tools for SWP validation. This tool is composed of test code configurator, the control part for automatic testing, the part creating test report and diagnostic software executed on various microcontrollers. Fig. 5 shows the overall configuration of our SWP validator.

5.2 Test cases for SWP validator

All test cases executed on our SWP validator are created based on the experiences of SWP development. After the analysis of all functions belonging to SWP such as OS, FBL, MEM, IO, COMM, and PM, it is decided that which tests should be performed before the delivery of SWP library. All test cases selected were implemented as diagnostic software modules and all of those were managed in the test code configurator. Test code configurator makes a decision which functions should be tested considering the specification of individual ECU before the delivery of our SWP library. Table 1. only shows a small part of all test items. All the test items fit for both S12x and MB90 platform are created and executed on the target board. All of the same functions were simultaneously executed on two different platforms to measure test results and compare the performance. This is one of the obvious evidences proving that the reuse of applications can be successfully realized under SWP environment.

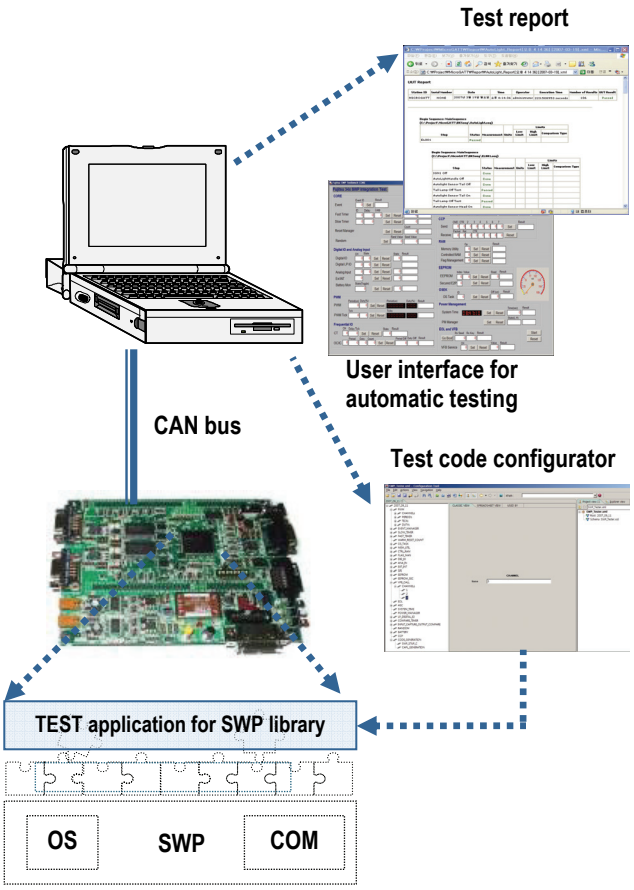

Fig. 5. The overall configuration of the SWP validator

Table 1. Test items

Function	Test items
OS	• Task scheduling
CORE	• Fast timer callback test
	• system time test
FBL	• EOL integration test
MEM	• EEPROM read/write test
	• Controlled RAM test
I/O	• Read/write functions for digital output
	• Controlled RAM test
COM	• SPI transfer test
	• CAN communication test
PM	• Transition to low power
	• Measure the period of low power task

5.3 SWP validation report

The SWP library and applications are separately developed in the projects. When the problems occur between SWP library and applications, it is not easy to inspect thoroughly which part caused them. In many cases car manufacture usually request a report how to verify SWP and applications. There is a function to create validation report in our SWP validator. After the test cases selected is performed, the test report is created from the SWP validator. This is a brief report that includes which item is tested and its test results are summarized.

6. CONCLUSIONS

In this paper, the software platform HAL and some generic component were developed for a new microcontroller. The RTE was also created by the SWP configuration tool which was developed based on JAVA & XML technologies. In order to reach AUTOSAR standards, current configuration tool was successfully upgraded to AUTOSAR platform configuration tool. It has been being used for our next AUTOSAR project. Additionally, SWP validator was implemented to verify SWP library efficiently before the delivery to Tier 1. The core technologies such as JAVA, XML and configuration concept were acquired throughout the SWP development project. This kind of core technologies, which are needed for upgrading our platform to AUTOSAR standards, will be spread widely in the near future. Finally the possibility of application reuse was revealed by executing same diagnostic module without any modification on two different platforms. However, upgrading our platform to AUTOSAR standards and implementing integrated design tool step by step with the idea of AUTOSAR methodology are still remained in no distant future.

REFERENCES

AUTOSAR initiative (2006). AUTOSAR – Enabling technology for Advanced Automotive Electronics, *Media release*, AUTOSAR.

Boddeker (2006). Experiences with RTE, *AUTOSAR 5th Premium Member Conference*, AUTOSAR.

Thompson (2006). A Control Centric Perspective of AUTOSAR, *AUTOSAR 5th Premium Member Conference*, AUTOSAR.

Colombero, Zoeller (2006). Model Based Application Development and AUTOSAR RTE interface, *AUTOSAR 5th Premium Member Conference*, AUTOSAR.

Freund, Moestel (2006). Model-Based OEM-Software Branding of AUTOSAR ECUs, *AUTOSAR 5th Premium Member Conference*, AUTOSAR.

Kohler, Kampfer (2006). VW-Hella internal AUTOSAR Demonstrator, *AUTOSAR 5th Premium Member Conference*, AUTOSAR.

Scott Loveland.,Geoffrey Miller.,Richard Prewitt Jr., Michael Shannon.,(2004), *Software Testing Techniques: Finding the defects that Matter,* Charles River Media.

Claudia Dencker.,(2003), *Common Mistakes in test cases in* pacific northwest Software Quality Conference, Software SETT corporation.