논문 2010-47SC-6-11

지능형 자동차용 임베디드 플랫폼 소프트웨어 테스팅 연구

(A Survey of Embedded Software Testing for Automotive Standard Platform)

조 현 철*, 박 세 권**, 조 희 섭**

(Hyun Chul Jo, Shiquan Piao, and Hui-Sup Cho)

요 약

지능형 자동차의 개발 및 보급 확대에 따라 차량용 시스템에 전기 전자 소프트웨어 모듈의 비중이 늘어나고 있다. 따라서 이러한 모듈을 포함하는 전자 제어 장치가 증가하게 되고, 소프트웨어와 시스템 장치 간의 상호 연동이 요구된다. 그렇지만 임베디드 플랫폼 간의 인터페이스가 표준화 되어 있지 않아 개발 및 통합이 어려운 실정이다. AUTOSAR 표준화 연구는 유럽의 자동차 제조업체와 부품 제조업체의 공동 협력 결과물로, 차량용 소프트웨어 플랫폼의 산업 표준을 제공한다. 본 연구의 이전 결과물로 AUTOSAR 플랫폼을 지원하는 RTE 코드 생성기 도구를 개발하였다. 본 논문에서는 테스트 과정을 통해 RTE 생성기의 동작 환경 및 표준 플랫폼 지원 결과를 분석하였다. 실험 결과, 코드 생성기가 표준 규격의 요구사항과 방법론에 맞게 파일을 생성함을 확인할 수 있었다.

Abstract

The number of electronic embedded software in vehicle system is ever increasing for years. As a result, the electronic control units have been growing dramatically, and it is required to mutual link between these units. Due to separate API each and every embedded platform, it is difficult to develop and integrate in automotive industry. The AUTOSAR project consists suppliers and manufacturers, and the partnership is a standardized platform to establish and develop an industry standard. On the previous works, we implemented the RTE generated module design based on AUTOSAR architecture. This paper specifically focuses on the testing of the development tool that generates RTE source code. The result satisfied a need for a RTE requirements and AUTOSAR methodology in a vehicle applications.

Keywords : 차량용 표준 플랫폼, 차량용 소프트웨어 AUTOSAR (AUTomotive Open Software ARchitecture), RTE (Run Time Environment), 소프트웨어 테스팅

Ⅰ. 서 론

최근 차량은 지능형 자동차의 도입과 부품 기술 개발에 따라 소프트웨어 모듈과 전자 장치에 의존하고 있다. 차량용 전장 소프트웨어의 시장은 급격히 팽창하고 있으며, 2010년까지 자동차 가격 대비 소프트웨어 모듈 가격의 비중이 약 13%를 차지할 것으로 예측하고 있다[1~2]. 이와 같이 지능형 자동차는 거대하고 복잡한 시스템으로 탈바꿈하고 있지만, 한편으로는 응용 소프트웨어 개발 및 기존 시스템과의 통합의 어려움을 야기한다. 이러한 추세에서 소프트웨어 모듈의 신뢰성이 중요해지고 있다.

AUTOSAR (AUTomotive Open Software ARchitecture) 플랫폼 표준화 연구는 이러한 문제점을 해결하기 위한 통합 소프트웨어 플랫폼을 제공한다. 이는 전자 제어 장치 (ECU; Electronic Control Unit)마다 다른 응용 소프트웨어 모듈의 구성 환경 및 통신 방식

* 정회원-교신저자, ** 정회원, 대구경북과학기술원 미래산업융합기술연구부
(Division of Advanced Industrial Science & Technology, Daegu Gyeongbuk Institute of Science & Technology)
※ 본 연구는 교육과학기술부의 기관고유사업 (2009-IT-2)의 연구 결과로 수행되었음.
접수일자: 2010년7월14일, 수정완료일: 2010년11월3일

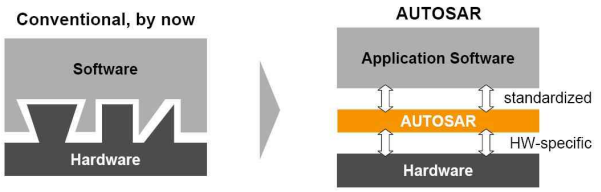

그림 1. 자동차 산업의 새로운 패러다임
Fig. 1. New paradigm in Automotive Technology.

등을 단일화 및 표준화하여 객체 지향적이고 확장성을 용이하도록 지원한다. 구체적으로는 RTE (Run Time Environment) 및 표준화된 인터페이스를 제공함으로서 전자 제어 장치 구축 및 시스템 통합 환경을 구축한다. 이로 인해, 소프트웨어 모듈의 재사용성이 증대되고 부품의 수정 및 업그레이드를 용이하게 한다.

그림 1은 차량용 표준 소프트웨어 플랫폼의 도입과 관련한 새로운 패러다임을 보여준다. 표준화된 인터페이스의 플랫폼 구조는 특히 하드웨어에 독립적인 응용 소프트웨어 개발을 가능하게 한다[3~4]. 이는 하드웨어의 의존성 탈피로 인해 모듈 개발 기반이 단축되어 차량 소프트웨어를 제품화하기 용이해지고, 종국에 신뢰성 향상이라는 장점을 얻는다. 또한, 하나의 응용 소프트웨어를 여러 차종에 적용이 가능하게 되어, 부품 공급업체의 자율 경쟁 및 개발 집중 체제로 전환될 수 있다.

Ⅱ. 관련 연구; 차량용 표준 플랫폼

1. AUTOSAR Architecture

AUTOSAR (AUTomotive Open Software ARchitecture) 는 2004년에 차량용 표준 플랫폼으로 개념이 정립되었고, 2005년 5월에 R1.0의 규격이 완성되었다. 이 후, 12월에 R2.0으로의 규격 업데이트와 시스템 방법론 (AUTOSAR Methodology) 을 발표하였다. 현재는 R4.0으로 규격의 안정화 작업이 추가되었으며, '플랫폼 표준의 협동과 제품 개발의 경쟁'이란 기본 추구 개념 하에 표준화 작업이 계속 진행 중이다.

그림 2는 AUTSOAR 표준 소프트웨어 플랫폼의 구조를 나타낸다. 크게 응용 소프트웨어 (Application Layer), RTE (Run Time Environment), BSW(Basic Software), 하드웨어 (Micro controller)의 4개 층으로 구성되어 있으며, 이는 하나의 전자 제어 장치를 구성한다.

응용 소프트웨어는 AUTOSAR에서 정의한 표준화된

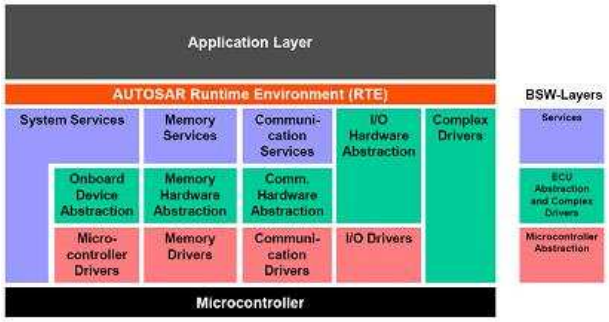

그림 2. AUTOSAR 표준 소프트웨어 플랫폼
Fig. 2. Overview of the AUTOSAR Platform.

인터페이스를 이용하여 RTE를 통해서 아랫단의 BSW와 데이터를 주고받는다[5]. 데이터 통신은 전자 제어 장치 내의 통신과 제어 장치 간 통신이 모두 포함되며, 외부 통신 프로토콜로는 CAN, FlexRay, LIN이 사용된다[6]. 이 때, 응용 소프트웨어 컴포넌트는 통신 네트워크로 통합되기 전에 각각의 전자 제어 장치에 데이터, 설정 상태와 같은 정보를 기술 문서로 제공한다. 이러한 정보들이 전자 제어 장치에 맵핑이 되고 각 설정 정보에 따라 VFB (Virtual Functional Bus)라는 RTE의 실제 구현 경로로 통신이 된다.

2. AUTOSAR Methodology

그림 3에서 AUTOSAR의 시스템 방법론을 제시한다[15]. 시스템 설정 단계 (System Configuration Description)와 전자 제어 장치 설정 단계 (ECU

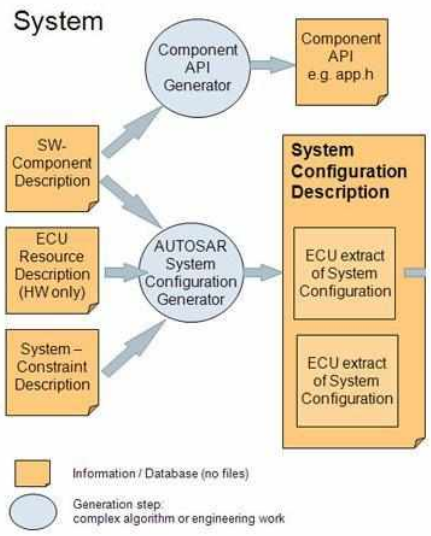

그림 3. AUTOSAR 시스템 방법론: 시스템 설정 단계
Fig. 3. The AUTOSAR System Building Methodology: System Configuration Description.

Configuration Description)의 두 가지 절차를 거치며, 표준 규격에서는 전자를 Contract Phase, 후자를 Generation Phase로 정의한다.

시스템 설정 단계에서는 소프트웨어 컴포넌트의 기술 파일 (SW-Component Description)로부터 컴포넌트 인터페이스를 생성하는데, 설정된 인터페이스는 타 모듈에 독립적이므로 응용 소프트웨어 개발자는 이를 고려하지 않고 모듈을 만들 수 있게 된다. 이는 바로 AUTOSAR 표준 플랫폼의 장점인 재사용성, 이식성의 개념을 내포한다. 이렇게 전자 제어 장치의 설정이 완료되면 다음 단계에서 RTE 생성을 위한 절차를 수행한다. RTE 코드 및 헤더 파일에 대한 생성은 AUTOSAR 전자 제어 장치 설정 파일을 기반으로 한다. 또한, 그림에서 보는 바와 같이 RTE 생성 파일 외에 운영체제, 통신 소프트웨어를 비롯한 BSW와 하드웨어 파일을 생성하는데, 본 연구에서는 표준 플랫폼의 핵심 기능인 RTE 파일을 생성하는 RTE 코드 생성기에 집중한다.

Ⅲ. 기존 연구; 차량용 플랫폼 개발

이전 연구로 차량용 표준 플랫폼을 지원할 RTE 코드 생성기를 개발하였다[7~8]. RTE 코드 생성기란, 응용 소프트웨어의 설계 기술 문서와 전자 제어 장치의 설정 정보를 입력 값으로 받아 소프트웨어 컴포넌트 인터페이스와 관련 RTE 소스 파일을 생성하는 도구이다. 즉, AUTOSAR의 표준 개념을 구현하는 설계 개발 도구를 의미한다. 이러한 코드 생성기는 AUTOSAR 표준 시스템을 기반으로 개발 방법론에 근거하여 구현하였다. 특히, 코드 생성기와 외부 입출력과의 상호 관계, 입출력 변수, 상 하위 모듈과의 상관관계를 다룬 RTE 규격을 고려하여 설계하였다(그림 4)[16]. 위의 그림에서 보다시피, 엔진 초기화 모듈, XML (eXtensible Markup Language) 파서, RTE 템플릿 (RTE Template), 생성기 엔진 모듈로 구성되고, 추가로 내부/외부 응용 프로그램 인터페이스를 정의하였다.

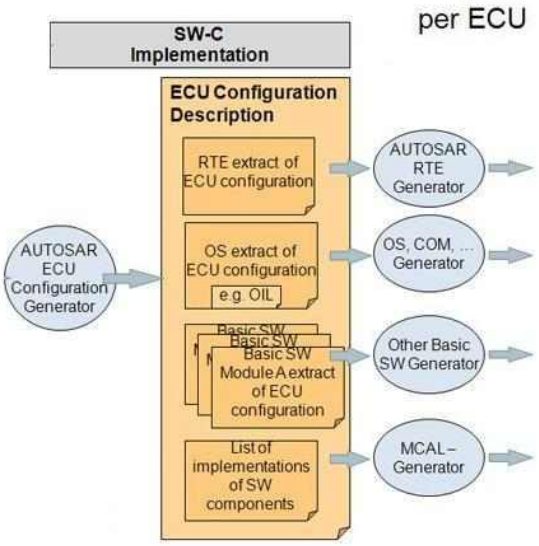

그림 4. AUTOSAR 시스템 방법론: 전자 제어 장치 설정 단계
Fig. 4. The AUTOSAR System Building Methodology: ECU Configuration Description.

엔진 초기화 모듈은 데이터 값과 모듈 환경의 초기화를 담당하고, XML 파서는 입력된 XML 형식의 기술 문서를 처리한다. 이후, 파서를 거쳐 생성된 데이터 값으로부터 RTE 코드 생성에 필요한 아이템을 추출 및 지정된 데이터 구조에 저장을 하고 RTE 템플릿을 이용하여 생성기 엔진 모듈에서 최종 파일을 생성한다[11].

RTE 템플릿 구조는 RTE의 함수 기능적인 요구사항과 비 기능적인 요구사항 및 일반적인 요구사항을 모두

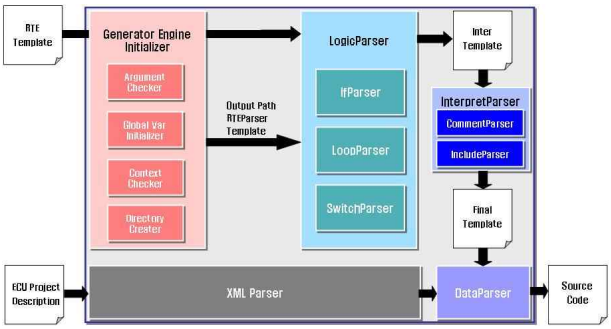

그림 5. RTE 코드 생성기 클러스터 환경
Fig. 5. RTE Code Generator Cluster Structure.

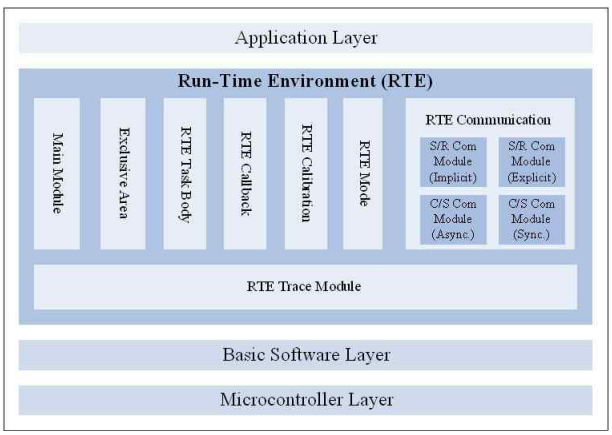

그림 6. RTE 템플릿 구조
Fig. 6. RTE Template Structure.

참조 하였고, 추가적으로 AUTOSAR 규격의 헤더 파일과 코드 파일의 관계를 참고하여 8개의 모듈로 구성하였다(그림 5)[7~8].

RTE의 통신 방식으로는 서버/클라이언트 방식 (S/C; Server/Client)과 센더/리시버 방식 (S/R; Sender/Receiver)을 모두 지원한다. 메인 모듈로 RTE를 초기화 및 종결화 할 수 있고, Task Body 모듈에서는 실제 소프트웨어 컴포넌트의 기능을 담당하는 함수를 처리한다[9~10].

또한 하위 모듈로의 함수 제공은 Callback 모듈에서 담당한다. 또한 Exclusive Area 모듈로 우선순위가 높은 데이터를 보호하는 기능을 하고, RTE Trace 모듈에서는 VFB 레벨에서 상 하위 단의 소프트웨어를 추적한다.

Ⅳ. 플랫폼 소프트웨어 테스팅 연구

1. 테스트 환경

본 장에서는 RTE 생성기 클러스터에 대한 테스트 환경 및 결과에 대해서 기술한다[12]. 테스트 케이스는 RTE 클러스터의 요구사항 및 설계서를 바탕으로 정의하였고 구체적인 테스트 절차를 거쳤다[13~14].

테스트 프로그램으로 두 개의 전자 제어 장치가 CAN 네트워크로 연결된 간단한 예제 환경을 만들었고, RTE를 통해 하나의 장치에서의 통신 여부는 물론, 다른 제어 장치로의 데이터 흐름 여부를 체크하였다. 이후, 이전 연구에서 개발된 RTE 코드 생성기를 이용하여 최종 소스 코드를 생성하였고 이 때, RTE 설정 내용과 소프트웨어 컴포넌트의 기술 문서 파일을 포함한 XML 파일을 입력 값으로 사용하였다.

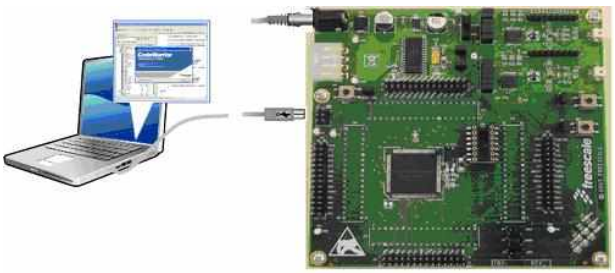

그림 7. 테스트 환경
Fig. 7. Test Environment.

표 1. 테스트 환경 분류표
Table 1. Table Listings for Testing.

시험환경		
소프트웨어	개발 환경	CodeWarrior
	펄	Active Perl 5.8.9
하드웨어	개발 PC	인텔 펜티엄4 XP
	개발 보드	프리스케일 S12X
	디버거	USB 디버거

테스트 환경은 소프트웨어 및 하드웨어 시스템 환경 구축으로 이루어진다. (표 1) 이는 개발 PC에서 펄 실행 환경이 설치되고 펄을 실행할 수 있는 AUTOSAR 개발 환경으로, XML 입력 파일 생성을 지원한다.

2. 테스트 결과

테스트는 AUTOSAR 표준 문서에서 제공하는 RTE 함수의 기능 요구사항을 바탕으로 케이스를 항목 별로 나누어 진행하였다. 구체적으로는 시스템 지원, Task Body 생성, 소스코드 생성의 3개 항목으로 통합하여 정리하였고, 생성된 코드를 위의 테스트 케이스를 바탕으로 분석하였다.

● 시스템 지원

시스템 지원 항목에서는 프로그래밍 언어와 각종 초기화 여부를 확인하였다. 우선, RTE 생성기를 이용하여 생성된 소스코드가 C/C++를 지원하는지 확인하였다. 다음으로, XML 파일의 기술 내용과 RTE 생성기의 생성 결과를 분석하여 초기화 값이 설정대로 출력되었는지 점검하였다. 또한, 생성된 RTE 소스코드에 RTE 초기화 및 종결화 기능의 제공하는지 여부를 확

```
/*************************************************
* RTE life cycle API
*************************************************

Std_ReturnType Rte_Start(void)
{
/* set default values for internal data */
Rte_SenderSWcPport2_displayData1 = (char)0;

  /* activate the Alarms used for TimingEvents */
(void)SetRelAlarm(Alarm0, 1, 1);
(void)SetRelAlarm(Alarm1, 1000, 1000);

 return RTE_E_OK;
}

Std_ReturnType Rte_Stop(void)
{
(void)CancelAlarm(Alarm0);
(void)CancelAlarm(Alarm1);

 return RTE_E_OK;
}
```

그림 8. 테스트 결과 코드 :RTE 초기화 및 종결화
Fig. 8 The test result: RTE Start/Stop.

인하였다.

코드 생성 결과, 타입 헤더파일에서 각각의 타입이 기 설정된 대로 정의되었고, 생성된 RTE 소스코드가 C 언어 형식으로 작성되었다. 또한 설정 파일의 포트와 데이터 값은 초기 값으로 제대로 설정됨을 확인하였다. RTE 생성기의 경우, 초기화 및 종결 화 함수를 생성하였고 이는 생성된 코드 내용에 포함되었다. (그림 8)

● Task Body 생성

Task Body가 설정대로 생성되었는지를 확인함과 동시에, Task Body 내의 Runnable Entity 생성 순서 및 활성화 정보와 할당 여부를 점검하였다. 여기에서 Runnable Entity란 소프트웨어 컴포넌트의 일부로 다른 Entity에 독립적으로 실행되고 스케쥴링 될 수 있는 특징이 있고. Task Body란 Runnable Entity의 호출을 포함하는 코드의 집합체이다.

Task Body는 RTE 생성기가 전자 제어 장치 기술 파일의 운영 체제 항목에서 설정한 내용대로 구성되는지 여부를 확인하였다. 또한, Runnable Entity는 기술 문서의 순서대로 활성 하는지 확인하였고, Runnable Entity가 Task에 할당되지 않은 경우, RTE 생성기가 이 설정을 보여주는지 확인하였다.

```
/*****************************************************
* Task bodies for all model tasks
*****************************************************

/*****************************************************
* Task:     DispTask
* Priority:
* Schedule:
*****************************************************
TASK(DispTask) /* MISRA rule 27:  */
{
EventMaskType ev;

Rte_Task_Dispatch(DispTask);

 for(;;)
 {
  Rte_Task_WaitEvent(DispTask, TE2);
  (void)WaitEvent(TE2);
  Rte_Task_WaitEventRet(DispTask, TE2);
  (void)GetEvent(DispTask, &ev);
  (void)ClearEvent(ev & (TE2));

  if ((ev & TE2) != (EventMaskType)0)
 {
   {
        /* call runnable */
 Rte_Runnable_DisplaySWc1_displayRunnable1_Start();
        displayRunnable1();
 Rte_Runnable_DisplaySWc1_displayRunnable1_Return();
 }
 }
        }
}
```

그림 9. 테스트 결과 코드 :Task Body
Fig. 9. The test result: Task Body.

테스트 결과, 입력 XML 파일과 RTE 소스 파일의 경우 동일한 Task가 생성되었고 (그림 9), Body 내에 Runnable entity가 순서대로 호출되었다. 입력 파일에서 Task와 Runnable entity의 할당 여부를 확인한 결과, 할당되지 않은 경우, RTE 생성기의 실행 화면에서 에러 메시지가 출력됨을 확인하였다.

● 소스코드 생성

소스코드 생성 항목에서는 XML 파일을 입력 받아 요구하는 RTE 소스 파일을 에러 없이 생성하는지 확인한다. 또한 하나의 전자 제어 장치의 기술 파일 당 하나의 소스 코드가 생성되는지 점검하였다. 세부적으로 RTE 생성기가 한 개 이상의 기술 문서 파일들을 입력받아, 템플릿에 적용 후 코드 파일 및 헤더 파일을 생성하는지를 실험하였다.

코드 생성 결과, 전자 제어 장치 설정에 따라 제어 장치 별로 RTE 소스코드와 타입 및 설정 정보를 포함하는 헤더파일, Callback 함수의 헤더파일이 생성되었다(그림 10). 이 때, Callback 함수는 AUTOSAR 플랫폼에서 한 모듈이 하위 층에 제공하는 함수를 일컫는

```
------------------------------------------
File Name    : Rte_Main.h
Description  : Rte LifeCycle File
Target system: Freescale S12XF512

------------------------------------------
       A U T H O R   I D E N T I T Y
------------------------------------------
Initials     :
Name         :
Department   : Automotive Embedded System
------------------------------------------
       R E V I S I O N   H I S T O R Y
------------------------------------------
Date         : Mon Apr 19 14:31:41 2010
Version      : 0.80
Description : Creation
******************************************


/* RTE Life-Cycle API */

/* double include preventer */
#ifndef _RTE_MAIN_H
# define _RTE_MAIN_H

# include "Rte.h"

Std_ReturnType Rte_Start(void);
Std_ReturnType Rte_Stop(void);

#endif /* _RTE_MAIN_H */
```

그림 10. 테스트 결과 코드 :소스코드 생성
Fig. 10. The test result: Code generation.

표 2. 테스트 케이스에 따른 결과 표
Table 2. Result Table for Test Case.

대분류	중분류	지원여부
시스템 지원	ANSI C/C++	Pass
	초기화 값 지원	Pass
	RTE 초기화/종결화	Pass
Task Body 생성	Task Body 구성	Pass
	Runnable Entity 활성화	Pass
	Runnable Entity 할당	Pass
소스코드 생성	입력 파일 지원	Pass
	코드 파일 생성	Pass
	전자 제어 장치별 코드 생성	Pass

다. 기술 문서의 설정 내용대로 변수가 설정되었고, 저장된 변수와 템플릿에서 사용된 변수가 동일함을 확인하였다. 이 때, 동일한 XML 파일 입력으로 RTE 생성기를 여러 번 실행하여 생성 코드의 동일성 및 시스템의 안정성을 점검하였다.

표 2에 기 분류한 통합 테스트 케이스에 따른 소스코드의 기능 지원 여부를 나타내는 결과를 정리하였다. 표에서 보는 바와 같이 시스템 지원, Task Body 생성, 소스 코드 생성의 대분류와 그에 속하는 중분류의 요구사항 기능들을 모두 만족시킴을 확인할 수 있었다.

V. 결 론

본 논문에서는 지능형 자동차용 임베디드 표준 플랫폼을 위한 코드 생성기 도구를 테스트하였다. 이전 연구에서 표준 규격에 기반 한 새로운 도구를 구현하였으나, 간단한 예제 프로그램을 돌려보는 수준의 검증에 그쳤다. 이에 대한 확장 연구로 표준 문서의 요구사항을 참고하여 정리한 케이스를 토대로 테스트 과정을 거쳤다. 실험으로는 펄 실행 환경이 설치된 개발 PC에서 기 구현한 RTE 생성기 프로그램을 실행하였고, 테스트에 사용할 프로젝트 파일로는 XML 입력 파일을 이용하였다. 또한, 개발 보드에 이식하여 하드웨어 상에서 실험을 함으로서, 소프트웨어 상의 단순한 프로그램 생성을 통한 테스트와의 차별성을 가졌다. 실험 결과 AUTOSAR의 개발 방법론을 그대로 유지함을 보였고, 생성된 소스코드와 헤더 파일의 경우 규격을 제대로 내포하고 있음을 확인할 수 있었다.

추가 연구로 기 구현된 코드 생성기의 최적화 작업이 요구되고, 이에 따른 새로운 테스트 케이스로의 검증 작업이 필요하다.

참 고 문 헌

[1] G. Leen and D. Heffernan, “Expanding Automotive Electronic Systems,” IEEE Computer, pp. 88-93, Jan. 2002.
[2] 조상복, “미래형 자동차 전장부품/시스템 기술,” 전자공학회, 제 34권, 제 5호, 18-19쪽, 2007년.
[3] A.L. Sangiovanni-Vincentelli et al, “Benefits and Challenges for Platform-Based Design,” in Proc. Design Automation Conf., ACM Press, pp. 409-414, 2004.
[4] A.L. Sangiovanni-Vincentelli, Di Natale. M, “Embedded System Design for Automotive Applications,” Computer, IEEE, Vol. 40, Issue 10, pp. 42-51, Oct, 2007.
[5] Schreiner, D., Schordan, M., Goschka, K. M., “Component Based Middleware-Synthesis for AUTOSAR Basic Software,” in Proc. of IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing, pp. 236-243, Tokyo, Japan, March, 2009.
[6] W. Lawrenz, CAN System Engineering: From Theory to Practical Applications. New York: Springer-Verlag, 1997.
[7] H. -C. Jo, S. -Q. Piao et al, “RTE Template Structure for AUTOSAR based Embedded Software Platform,” in Proc. of the IEEE/ASME International conference on Mechatronic and Embedded Systems and Applications, Beijing, China, pp. 233-237 Oct. 2008.
[8] H. -C. Jo, S. -Q. Piao et al, “Automatic Source code generator based on AUTOSAR RTE Template for Vehicular Applications,” in Proc. of the ASME/IEEE International conference on Mechatronic and Embedded Systems and Applications, San Diego, CA, Sep. 2009.
[9] Krithi Ramamritham, John A. Stankovic and Wei Zhao, “Distributed Scheduling of Tasks with Deadlines and Resource Requirements,” IEEE Transactions on Computer, Vol. 38, No. 8, pp. 1110-1123, Aug. 1989.
[10] J. R. Pimentel, “An Incremental Approach to Task and Message Scheduling for AUTOSAR Based Distributed Automotive Applications,” in Proc. of the 4th International Workshops on Software Engineering for Automotive Systems, .IEEE, May, 2007.
[11] Honekamp, U., “The Autosar XML Schema and Its Relevance for Autosar Tools,” Software, IEEE, Vol. 26, Issue 4, pp. 73-76, 2009.

[12] 김인철, “소프트웨어 로봇 연구용 테스트베드,” 전자공학회논문지, 제33권, 제3호, 28-35쪽, 2006년 3월.
[13] AUTOSAR, AUTOSAR BSW & RTE Conformance Test Specification, 2009.
[14] AUTOSAR, Conformance Test Agency Accreditation, 2009.
[15] AUTOSAR, Methodology, 2009.
[16] AUTOSAR, Requirements on RTE, 2009.

저 자 소 개

조 현 철(정회원)
2005년 한양대학교 전자공학과 학사.
2007년 서울대학교 전기공학과 석사.
2007년～대구경북과학기술원 (DGIST) 연구원.
<주관심분야 : 임베디드 소프트웨어, 음성 인식, 통신 신호처리>

조 희 섭(정회원)
1999년 경북대학교 전자공학과 학사.
2001년 경북대학교 전자공학과 석사.
2001년～2005년 LG전자 선임연구원.
2005년～대구경북과학기술원 (DGIST) 선임연구원.
<주관심분야 : 임베디드 하드웨어, Real-time OS>

박 세 권(정회원)
1999년 북경대학교 전자학과 학사.
2002년 영남대학교 정보통신 공학과 석사.
2004년 영남대학교 정보통신 공학과 박사 수료.
2005년～대구경북과학기술원 (DGIST) 연구원.
<주관심분야 : 유/무선 통신시스템, 무선자원관리>