논문 2012-07-03

범용 개발 보드를 이용한 차량용 소프트웨어 테스트 시스템 개발

(Testing System for Automotive Software Using a General Purpose Development Board)

금 대 현, 홍 재 승, 진 성 호, 조 정 훈*
(Daehyun Kum, Jaeseung Hong, Sungho Jin, Jeonghun Cho)

Abstract : Recently automotive software has been more complex and needs to be reduced its development time. Software testing of its functionalities and performance should be conducted in an early development phase to reduce time to market and the development cost. Software functional testing can be performed through simulating the hardware, but it is not guaranteed that evaluation of real-time performance using simulation has enough accuracy. Real-time performance can be precisely evaluated with hardware-in-the-loop simulation, but it costs time and effort to set up hardware for testing. In this paper, we suggest a testing system that can evaluate functional requirements and real time properties with a general-purpose development board in the early development phase. In addition, we improve reusability of the testing system through modularized and layered architecture. With the proposed testing system we can contribute to building reliable testing system at low cost without difficulty.

Keywords : Software testing, Automotive software, AUTOSAR

Ⅰ. 서 론

최근 차량의 지능화로 인하여 소프트웨어는 점차 복잡해지고, 개발 주기 또한 단축되고 있다. 이러한 현실을 극복하기 위하여 소프트웨어 플랫폼을 표준화하고 소스 코드의 생성을 자동화하고 있다. 그러나 소프트웨어가 복잡해질수록 테스트에도 많은 시간과 노력이 필요하며, 개발 기간 단축 및 비용 절감을 위하여 개발 초기 단계에 소프트웨어의 기능 및 성능 검증이 요구되고 있다.

개발 초기 단계에서 소프트웨어의 기능 테스트는 하드웨어 및 플랜트 모델의 시뮬레이션을 이용하여 수행 가능하지만, 시스템의 실시간 성능을 테스트하기 위해서는 시뮬레이션만으로 는 한계가 있으며 실제 하드웨어에서의 성능 테스트가 필요하다.

그러나 ECU 하드웨어의 개 발이 완료되기 전에 소프트웨어를 하드웨어에서 테스트하기 위해서는 여러 가지 어려움이 따른다. MCU에서 소프트웨어를 실시간으로 테스트하기 위해서는 네트워크 통신 메시지의 송수신이 가능해야 하고, 실시간으로 테스트 입력과 수행 결과를 주고받을 수 있어야 한다. 센서신호를 MCU에 입력하고 액추에이터 출력 값을 MCU로부터 읽기 위해서는 하드웨어 장치가 필요하며, 시스템의 특성에 따라 테스트 장치도 달라진다. 실제 응용 시스템에 사용될 ECU가 개발되기 전에 이러한 테스트를 하기는 쉽지 않다.

개발 기간 단축 및 신뢰성 향상을 위하여 하드웨어 개발이 완료되기 전에 ECU 하드웨어의 설계에 독립적으로 테스트 할 수 있는 방법이 필요하며, 본 논문에서는 범용 개발용 ECU를 사용하여 개발 초기 단계에 저비용으로 소프트웨어를 실시간으로

* 교신저자(Corresponding Author)
논문접수 : 2011. 07. 12., 수정일 : 2011. 08. 12.,
채택확정 : 2011. 09. 07.
금대현, 홍재승, 진성호 : 대구경북과학기술원
조정훈 : 경북대학교 전자전기컴퓨터학부
※ 본 연구는 교육과학기술부에서 지원하는 대구경북과학기술원 기관고유사업에 의해 수행되었습니다(11-RS-03).

ISSN : 1975-5066

테스트 할 수 있는 시스템을 연구하였다.

응용 소프트웨어를 ECU에 독립적으로 테스트하기 위해서는 ECU 하드웨어 설계와 상관없이 테스트 값을 입력하고 출력 값을 읽을 수 있는 인터페이스가 필요하다. 범용 개발용 ECU는 MCU 레지스터 값을 읽고 쓸 수 있는 인터페이스와 CAN 네트워크 등의 통신 인터페이스가 기본으로 제공된다. 본 연구에서는 레지스터 제어 인터페이스 및 CAN 네트워크 인터페이스를 이용하여 테스트 입력과 수행 결과를 확인할 수 있는 테스트 시스템을 개발하고자 한다. 또한 테스트 시스템을 계층화 및 모듈화하여 재사용성을 향상 시키고자 한다.

본 연구에서 제안한 MCU 테스트 시스템을 적용함으로써, 개발 초기 단계에 소프트웨어의 기능뿐 아니라 성능 테스트를 저 비용으로 쉽고 빠르게 수행할 수 있으며, 개발 기간 단축 및 신뢰성 향상에 기여할 수 있을 것이다.

Ⅱ. 관련연구

차량 소프트웨어를 위한 테스트 시스템 개발을 위하여 AUTOSAR와 TTCN-3 표준을 적용하였다. AUTOSAR 표준을 적용하여 차량 소프트웨어의 복잡성을 해결하고 재사용성을 높일 수 있으며, AUTOSAR는 표준화된 XML을 이용하여 소프트웨어 설계 정보를 저장하고 관리한다. 본 연구에서는 AUTOSAR XML을 이용하여 테스트 시스템을 자동으로 생성하고 있다[1-3]. TTCN-3 표준은 통신, 자동차 등 다양한 시스템의 테스트에 적용되고 있으며, AUTOSAR의 적합성 테스트 표준으로 사용되고 있다. TTCN-3을 사용함으로써 테스트 데이터의 재사용성을 향상시킬 수 있다[4-6].

개발 초기 단계에 차량 임베디드 소프트웨어의 테스트를 위하여 다양한 방법을 사용하고 있다. SIL(Software In the Loop) 테스트는 시뮬레이션을 이용하여 하드웨어의 도움 없이 테스트를 할 수 있지만, 실시간 성능 및 네트워크를 고려한 테스트를 하기에는 부족한 측면이 있다[7-8]. HIL(Hardware In the Loop) 테스트는 실제 하드웨어를 사용하여 실시간 테스트를 할 수 있지만, 센서 및 액추에이터 입출력을 위한 하드웨어 장비가 필요하며 테스트가 가능하기 까지 많은 시간과 노력이 필요한 문제가 있다[9]. 본 연구에서는 개발 초기 단계에 센서 및 액추에이터 등 하드웨어의 도움 없이 저비용으로 HIL 테스트를 할 수 있는 테스트 방법을 연구하였다.

Ⅲ. MCU 테스트 시스템 설계

1. 테스트 시스템 소개

테스트 시스템은 그림 1에서와 같이 테스터와 타겟 시스템으로 구성되며, 통신 인터페이스를 통해서 테스트를 수행한다. 테스터는 호스트 컴퓨터에서 실행되는 테스트용 소프트웨어 프로그램이며, 타겟 시스템은 개발된 소프트웨어가 실행되는 범용 ECU이다.

테스터는 정의된 테스트 케이스에 따라 타겟 시스템을 테스트하는 역할을 수행하며, 타겟 시스템으로 테스트 입력을 보내고, 수행 결과를 받아서 합격을 판단한다. 타겟 시스템은 개발된 소프트웨어가 실행되고 있는 환경이며 MCU 레지스터 및 CAN 네트워크 통신을 위한 인터페이스가 제공된다. 통신 인터페이스는 MCU 레지스터 제어 장치와 CAN 메시지 송수신 장치로 구성된다. MCU 레지스터 제어 장치는 레지스터에 직접 접근 하는 방법으로 테스트 입력값을 쓰고 출력 값을 읽고, CAN 메시지 송수신 장치를 통하여 실제 네트워크에 메시지를 전송하고 수신 메시지를 읽는다.

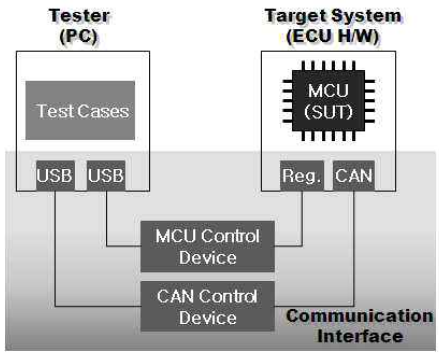

그림 1. 테스트 시스템 개념
Fig. 1. Concept of the Testing System

테스터는 재사용성을 향상시키기 위하여 계층화 및 모듈화 하였으며, 그림 1과 같이 크게 두 개의 계층으로 이루어진다. 테스트 케이스 계층은 타겟 시스템의 테스트를 위한 테스트 케이스 컴포넌트, 테스트 평가 컴포넌트로 구성되고, 테스트 어댑터 계층은 타겟 시스템과의 통신을 위하여 테스트 어댑터, 통신 인터페이스로 구성된다.

테스터는 그림 2와 같이 구성되며, 각 모듈별 특

성은 다음과 같다. 테스트 케이스 컴포넌트는 타겟 시스템의 테스트를 위한 테스트 케이스이다. 응용 시스템에 따라 변경된다. 테스트 평가 컴포넌트는 테스트 입력을 타겟 시스템으로 보내고, 타겟 시스템으로부터의 결과 값을 수신하여 합격 유무를 판단하는 컴포넌트이다. 모든 테스트 단계 및 테스트 환경에서 재사용 될 수 있으며, 응용 시스템이 변하더라도 재사용 가능하다.

테스트 어댑터는 타겟 시스템의 실행 환경 또는 통신 방법에 따라 설계하는 컴포넌트이다. 테스트 컴포넌트와 타겟 시스템과의 테스트 데이터 송수신을 위하여 데이터 변환을 위한 모듈이다. 통신 인터페이스는 타겟 시스템과의 물리적인 통신을 위한 하드웨어 드라이버에 해당하는 컴포넌트이며, MCU 레지스터를 제어를 위한 하드웨어 드라이버와 네트워크 메시지 송수신을 위한 CAN 드라이버가 포함된다.

본 테스트 시스템은 레지스터의 제어를 통해서 센서 및 액추에이터의 역할을 대체함으로써, 테스트 시스템을 쉽고 빠르게 만들 수 있는 장점이 있다. PC 프로그램인 테스터에서 발생하는 지연시간은 차량 바디 시스템에서는 무시할 수 있는 작은 시간이며, 사용자 편의성을 위한 다양한 차량 바디 전장 시스템의 테스트에 적합하다.

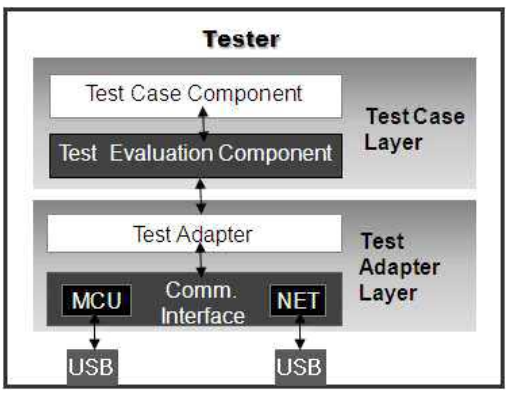

그림 2. 테스터 구조
Fig. 2. Architecture of the Tester

2. 테스트 케이스 계층 설계

테스트 케이스 컴포넌트는 그림 3과 같이 테스트 ID, 테스트 데이터 템플릿, 테스트 케이스, 테스트 컨트롤 모듈로 구성된다. 테스트 ID 모듈은 응용 시스템의 테스트 입력과 출력 ID를 정의한 모듈이며, 테스트 데이터 템플릿은 테스트 입출력 메시지를 정의할 수 있는 구조체이다. 테스트 케이스 모듈은 테스트 ID 및 테스트 데이터 템플릿을 이용하여 테스트 입력 및 예상 결과 값을 정의하고, 응답 시간 등의 제약 조건을 설정할 수 있다. 테스트 컨트롤 모듈은 테스트 케이스의 실행 순서 등을 제어한다.

테스트 평가 컴포넌트는 테스트 입력 송신 모듈, 테스트 결과 수신 및 평가 모듈로 구성되며, 테스트 수행 및 평가 과정을 동적으로 모델링한 컴포넌트이다. 테스트 컨트롤 모듈로부터 테스트 데이터를 수신하여 테스트 입력을 타겟 시스템으로 송신한다. 타겟 시스템에서 수행된 결과 값을 수신하여 예상 결과 값과 제약 조건들을 비교하여 테스트의 합격 유무를 판단한다. 테스트 결과를 다시 테스트 컨트롤 모듈에 보내어 결과가 합격일 경우 다음 테스트를 진행하고, 불합격일 경우 테스트를 중단하거나 다른 액션을 취하게 된다.

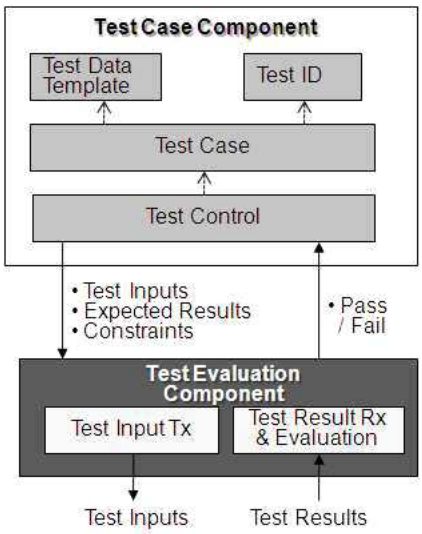

그림 3. 테스트 케이스 계층
Fig. 3. Test Case Layer

3. 어댑터 계층 설계

테스트 어댑터는 그림 4와 같이 테스트 케이스 인터페이스, MCU 어댑터, CAN 어댑터, 테스트 신호 변환기로 구성된다. 테스트 케이스 인터페이스는 타겟 시스템과의 통신 수단에 관계없이 테스트 평가 컴포넌트와의 테스트 데이터 교환을 가능하게 하는 모듈이다. 테스트 입력 메시지를 받으면, 입력 신호의 타입에 따라서 MCU 레지스터로 보내거나,

네트워크 신호로 보내는 역할을 한다. 그리고 MCU 레지스터로 부터 읽은 값이나, CAN 수신 메지시를 테스트 평가 컴포넌트로 보낸다.

MCU 어댑터는 테스트 입력이 발생할 경우, 테스트 ID와 입력 값으로부터 레지스터 주소를 찾고 레지스터 값으로 변환하는 역할을 한다. 반대로 MCU 레지스터에서 값을 읽을 경우, 레지스터 주소와 값으로부터 테스트 ID와 결과 값으로 변환한다. CAN 어댑터도 같은 방법으로 테스트 ID 및 입력 값으로부터 CAN 메시지를 만들고, 수신된 CAN 메시지로부터 테스트ID 및 결과 값으로 변환한다.

테스트 신호 변환기는 레지스터 주소 및 CAN 메시지를 테스트 ID와 값으로 변환에 필요한 기능을 제공한다. 통신 인터페이스는 실제 MCU 및 CAN 네트워크와 통신을 하기위한 하드웨어 드라이버를 제공한다.

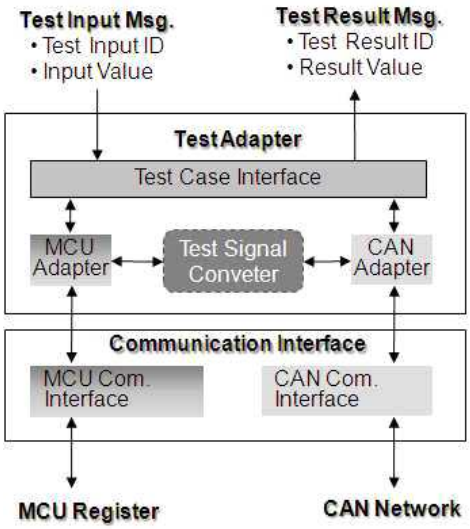

그림 4. 테스트 어댑터 계층
Fig. 4. Test Adpater Layer

IV. MCU 테스트 시스템 개발

1. 테스트 시스템 구성

그림 5는 테스트 시스템의 실제 구성도를 보여준다. MCU는 Freescale사의 S12XDP512를 사용하였으며, 타겟 시스템은 MCU 레지스터 인터페이스 및 CAN 통신 인터페이스가 내장된 범용 개발 보드를 활용하였다. 테스터는 일반 PC 환경에서 실행 가능하며, 테스트 컴포넌트 및 테스트 평가 컴포넌트는 테스트 케이스 설계 언어인 TTCN-3으로 개발하였다.

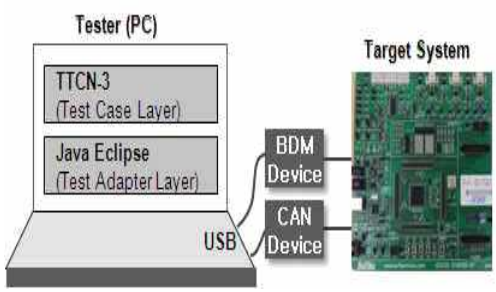

그림 5. 테스트 시스템
Fig. 5 Testing System

테스트어댑터 및 통신 인터페이스는 이클립스 기반의 자바로 개발하였다. 그리고 타겟 시스템과 테스터 사이의 테스트 데이터 송수신을 위하여 BDM(Background Debugger Module)과 CAN송수신 장치를 이용하였다. BDM은 MCU 레지스터를 읽고 쓸 수 있는 장치로써, DIO, PWM, ADC와 같은 하드웨어 신호를 레지스터를 이용하여 제어하는 역할을 한다.

```
/* Test ID */
const integer TI_VisualAlarm := 1;
const integer TI_Door := 2;
...
const integer ER_MapLampLeftPWM := 65;
const integer ER_MapLampRightPWM := 66;
...
/* Test Data Template */
template SIGNAL setTI_VisualAlarm (integer Value) :=
{
  signalID := TI_VisualAlarm,
  signalLength := 1,
  signalValue := Value,
  type := CAN,
}
...
/* Test Case */
Group TestCase_1 {
  const TI TI_TestCase1 := {setTI_VisualAlarm (1) };
  const ER ER_TestCase1 := {getER_RoomLamp (100) };
}
...
/* Test Control */
execute( TC (TI_TestCase_1, ER_TestCase_1 ) );
execute( TC (TI_TestCase_2, ER_TestCase_2 ) );
...
```

그림 6. 테스트 케이스 컴포넌트
Fig 6. Test Case Component

2. 테스트 케이스 계층 개발

테스트 케이스 및 테스트 컴포넌트는 TTCN-3 테스트 표준 언어를 사용하여 개발하였다.

TTCN-3를 사용함으로써 테스트 케이스의 향후 재사용성이 증대되고, 테스트 평가 컴포넌트의 개발이 쉽고, 신뢰성 또한 높일 수 있다.

그림 6은 실내등 시스템의 테스트 케이스 컴포넌트 개발 예를 간략히 보여준다. 테스트 ID와 테스트 데이터 템플릿을 이용하여 테스트 케이스를 작성하고, 테스트 컨트롤 모듈에서 테스트 케이스를 실행시킨다. 테스트 평가 컴포넌트는 테스트 입력 값을 타겟 시스템으로 송신하고, 수신된 결과 값에 대해서 합격 여부를 판단한다. TTCN-3언어를 활용하여 테스트 입력부터 결과 판단까지 테스트 과정을 개발할 수 있다.

3. 테스트 어댑터 계층 개발

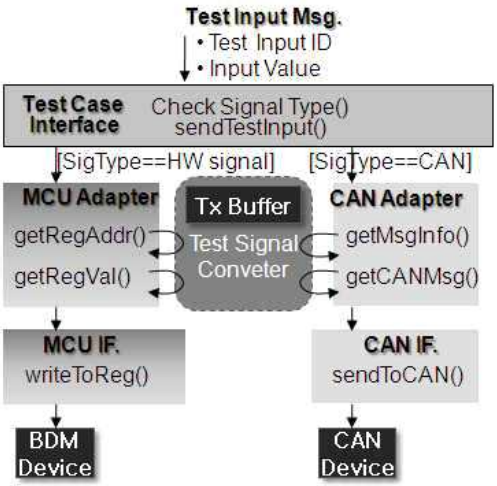

그림 7. 테스트 입력 송신 과정
Fig 7. Transmission Process of Test Input

테스트 어댑터 및 통신 인터페이스는 자바 이클립스 기반에서 개발하였다. 그림 7은 테스트 입력을 타겟 시스템으로 송신하는 프로세스를 간략히 보여준다. 테스트 컴포넌트로부터 테스트 메시지를 받으면 테스트 ID의 타입을 검색한다. 만약 테스트 ID가 하드웨어 신호이면 MCU 어댑터를 호출하고, CAN 네트워크 신호이면 CAN 어댑터를 호출한다.

MCU 신호의 테스트 입력 송신 과정은 다음의 절차를 따른다. 입력 값을 쓰기 위해서 getRegAddr() 함수를 이용하여 레지스터 주소를 확인한다. 그리고 getRegVal() 함수를 이용하여 레지스터 값을 계산하고, MCU 인터페이스를 호출하여 BDM을 통하여 MCU 레지스터에 값을 쓰게 된다. 이때, MCU의 하드웨어 신호 종류에 따라 레지스터 주소와 값으로 변환하는 방법은 다르며, DIO 신호를 예로 살펴보면, 테스트 ID에 해당하는 포트와 핀, 해당 핀의 특성을 찾는다. 그리고 먼저 포트의 값을 읽은 후 해당 핀의 값만 변경 시킬 수 있도록 비트 연산을 실행하여 레지스터 값을 계산한다.

다음으로 CAN 신호의 입력 값 송신 과정을 살펴보면, 보통 CAN 메시지는 다수의 신호로 구성되므로, getMsgInfo() 함수를 이용하여 해당 테스트 ID가 속한 메시지의 정보를 얻는다. CAN 메시지 ID, 메시지 길이, 메시지에 속한 다른 신호의 ID 등을 얻게 된다. 다음으로는 getCANMsg() 함수를 이용하여 송신할 메시지를 만드는 단계로써, 메시지에 속한 입력신호 이외의 신호들의 값은 송신 버퍼에 저장된 값으로 채우고, 입력 신호의 값은 테스트 입력 신호의 값으로 채운다. 그리고 각 신호의 시작 비트와 비트 길이를 참조하여 CAN 메시지를 만든 다음 sendToCAN() 함수를 호출하여 CAN 송수신 장치를 통하여 타겟 시스템으로 메시지를 전송한다.

테스트 결과의 수신은 그림 8과 같다. 테스트 케이스 인터페이스의 rxThread로 부터 시작하며, rxThread 내의 readTestResult() 함수에서 MCU 어댑터와 CAN 어댑터를 주기적으로 호출하여 결과

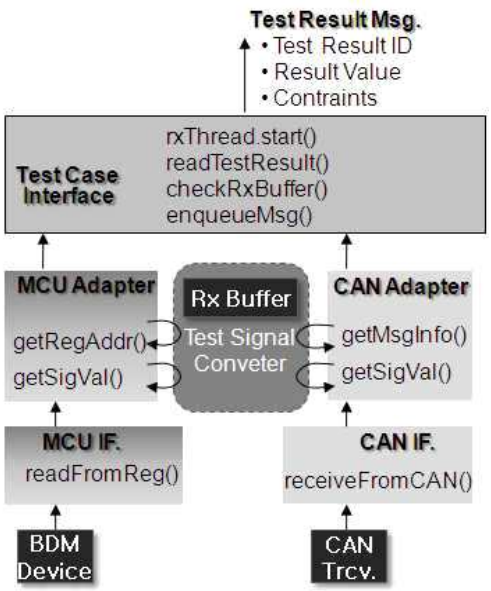

그림 8. 테스트 결과 수신 과정
Fig 8. Receiving Process of Test Result

```
/* MCU Register info from Signal ID */
public int[] getRegAddrPwm(int signalId)
{
   ...
   switch (signalId)
   {
   case TestSignalID.ER_RoomLampPWM:
      mcuReg[McuGP.CHANNEL] = 0;
      mcuReg[McuGP.POLARITY] = PWMHIGH;
      mcuReg[McuGP.ALIGN]   = PWMLEFT;
      mcuReg[McuGP.PERIOD] = 0x0314;
      mcuReg[McuGP.DUTY]    = 0x031C;
      break;
   ...
}

/* MCU Register Value to Signal Value */
public int getSigValPwm(byte period, byte duty, int polarity)
{
   ...
   if(polarity == PWMHIGH){
      signalValue = (duty)*100;
      signalValue = signalValue/period;
   }else if(polarity == PWMLOW){
    ...
}
```

그림 9. 테스트 신호 변환 모듈
Fig. 9. Test Signal Convertor Module

를 확인한다.

먼저 MCU 어댑터를 통한 테스트 결과 수신 과정을 살펴보면, 테스트 출력 ID에 해당하는 MCU 레지스터를 읽는다. 이 과정에서 getRegAddr() 함수로부터 각 테스트 ID에 해당하는 레지스터 주소를 확인하고, 주소에 해당하는 레지스터 값을 읽는다. getSigVal() 함수를 이용 하여 읽은 레지스터 값으로부터 테스트 ID의 출력 값으로 변환한다. PWM의 예를 들어 설명하면, PWM 파형의 기간과 듀티 사이클로 부터 출력 신호 값으로 변환한다. 우선 테스트 ID에 해당하는 PWM 파형 및 듀티 사이클 레지스터 주소를 찾는다. 다음으로 레지스터 값을 읽고 출력 신호 값으로 변환한다.

CAN 메시지의 수신 과정은 먼저 메시지가 수신되면, getMsgInfo() 함수를 통해서 수신된 CAN 메시지의 ID, 데이터 등의 정보를 확인한다. 다음으로 getSigVal() 함수에서 CAN 메시지에 포함된 테스트 ID와 결과 값을 계산한다. 수신된 결과 값들은 모두 수신 버퍼에 저장되며, 테스트 케이스 어댑터의 checkRxBuffer() 함수에서 수신된 값이 이전 값과 변경된 경우 enqueueMsg() 함수를 이용하여 테스트 결과 메시지를 테스트 평가 컴포넌트로 보낸다.

테스트 ID와 MCU 레지스터 및 CAN 메시지 사이의 변환은 테스트 신호 변환기에서 이루어 진다. 테스트 결과 수신시 PWM 신호의 변환 과정을 간략히 표현하면 그림 9와 같다. 먼저 getRegAddrPwm() 함수로부터 레지스터 주소 및 PWM 채널의 필요한 정보들을 얻는다. 다음은 readFromReg() 함수로부터 필요한 레지스터 값을 읽은 후, getSigValPwm() 함수에서 레지스터 값과 PWM 채널의 설계 정보를 기반으로 테스트 결과 값으로 변환한다.

4. 테스트 자동화

테스트 시스템의 구조를 계층화하고 모듈화 함으로써 자동 생성이 가능하다. 최근 차량 소프트웨어의 개발 기간 및 신뢰성 향상을 위하여 AUTOSAR 표준 플랫폼을 적용하고 있으며, 표준 플랫폼을 활용함으로써 테스트 시스템의 생성을 자동화할 수 있다. AUTOSAR는 모든 소프트웨어 설계 정보를 XML 형태로 저장하고 관리한다.

AUTOSAR XML을 이용하면 테스트 시스템의 생성을 자동화 가능하다. 테스터의 테스트 ID, 테스트 데이터 템플릿, 테스트 신호 변환 모듈은 응용 시스템의 특성에 따라 변경되는 모듈이며, AUTOSAR XML로부터 자동 생성된다. 테스트 시스템의 나머지 모듈은 응용 시스템과 상관없이 재사용 가능한 모듈이다.

우선 응용 소프트웨어 컴포넌트 XML 파일로부터 테스트 입력 및 출력 정보를 추출한다. 다음으로 DIO, ADC, PWM, CAN 네트워크 등 베이직 소프트웨어 설계 정보가 저장된 XML 파일로부터 테스트 신호 변환 모듈을 생성한다.

5. 차량 실내등 시스템 적용 사례

그리고 본 논문에서 제안한 테스트 시스템을 차량 실내등 시스템에 적용하였다. 실내등 시스템은 12개의 DIO 입력과 9개의 CAN 입력이 있으며, 8개의 PWM 출력과 1개의 CAN 출력메시지로 구성된다.

AUTOSAR 응용 소프트웨어 컴포넌트 XML로부터 테스트 ID 및 테스트 데이터 템플릿 모듈을 생성하고, DIO, PWM, CAN 등 AUTOSAR 베이직 소프트웨어 모듈의 XML 파일로부터 테스트 신호 변환 모듈을 생성하였다.

차량 실내등 소프트웨어를 범용 개발 ECU에 다운로드하여 실행하고, AUTOSAR XML로 부터 생성된 테스터를 활용하여 자동으로 테스트하였다. 그림 10은 테스트 케이스의 예를 보여주고 있다. 테

스트 케이스는 실내등 스위치를 켠 후, 10ms 이내에 실내등이 켜져야 한다. 테스트 어댑터는 실내등 스위치의 신호 타입을 구별하고 해당 MCU의 레지스터 주소에 입력값을 쓴다. MCU에서 실행된 결과값은 예상 결과 값과 일치하며 10ms 이내에 응답이 완료되었으므로 해당 테스트 케이스는 합격이다. 이러한 방법으로 모든 테스트 케이스에 대해서 기능 및 성능 테스트를 하였다.

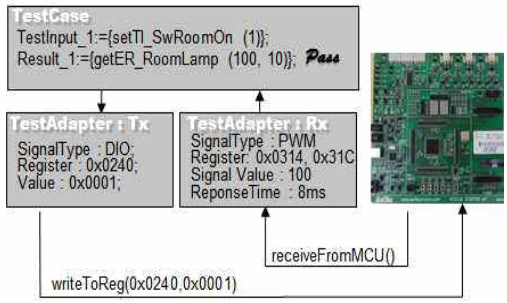

그림 10. 실내등 시스템의 테스트 케이스
Fig. 10. Test Case of the Interior Lighting System

제안한 방법을 차량 실내등 시스템에 적용한 결과 소프트웨어의 설계 정보만 이용하여 테스트 실행 환경을 자동으로 생성함으로써 테스트 시간 및 노력을 최소화할 수 있었다. 특히 개발 초기 단계에 차량 실내등을 위한 ECU 하드웨어의 개발이 완료되기 전에 MCU 하드웨어 환경에서 소프트웨어의 기능 및 성능을 검증할 수 있었다.

V. 결 론

최근 차량 소프트웨어의 복잡성이 증가하고, 개발 기간 단축의 요구가 증대하고 있다. 테스팅에 소요되는 시간과 노력을 절감하여 차량 소프트웨어 경쟁력 향상의 필요성이 있다.

본 연구에서는 개발초기 단계에 ECU 하드웨어 개발이 완료되기 전에 차량용 소프트웨어를 테스트할 수 있는 테스팅 시스템을 제안하였다. MCU 레지스터 제어 장치와 CAN 네트워크 장치를 활용하여 범용 개발 ECU에서 쉽고 빠르게 테스트 할 수 테스트 시스템을 개발하였다. 또한 테스트 시스템을 모듈화함으로써 재사용성을 향상시킬 수 있었으며, AUTOSAR 소프트웨어 표준 플랫폼을 사용하여 테스트 시스템의 생성을 자동화할 수 있었다.

본 연구에서 제안한 방법을 사용함으로써 테스팅 비용 및 시간을 절감하고 소프트웨어 신뢰성 향상에 기여할 수 있을 것이다.

참고문헌

[1]. D. Kum, J. Son , J. Son, M. Kim, "Automotive Embedded System Software Development and Validation with AUTOSAR and Model-based Approach," Journal of Control, Automation, and System Engineering, Vol. 13, No. 12, pp.1179-1185, 2007.

[2]. D. Kum, S. Lee, G. Park, J. Cho, "Automated Testing Techniques for Automotive Software Components with TTCN-3," Journal of KISS(C): Computing Practices and Letters, Vol 16, No. 5, pp.541-545, 2010.

[3]. 권규호, 이정욱, 김기석, 김재영, 김주만, "자동차 전장용 실시간 태스크 스케줄링 알고리즘," 대한임베디드공학회 논문지, Vol. 5, No. 2, pp.103-110, 2010.

[4]. J. Grabowski, D. Hogrefe, G. Rethy, I. Schieferdecker, A. Wiles, C. Willcock, "An Introduction to the testing and test control notation (TTCN-3)," Computer Networks Vol. 42, No. 3, pp.375-403, 2003.

[5]. D. Wang, J. Kuang, W. Tan, "Conformance Testing for the Car Lights System Based on AUTOSAR Standard," Proceedings on the 2011 IEEE International Conference on Communication Software and Networks, pp.345-348, 2011.

[6]. J. Grossmann, D. Serbanescu, I. Schieferdecker, "Testing Embedded Real Time Systems with TTCN-3," Proceedings on 2009 International Conference on Software Testing Verification and Validation, pp. 81-90, 2009.

[7]. W. Kwon, S. Choi, "Real-Time Distributed Software-In-the-Loop Simulation for Distributed Control Systems," Proceedings on the 1999 IEEE International Symposium of Computer-Aided Control System Design, pp.115-119, 1996.

[8]. A Mjeda, P. McElligott, K. Ryan, S. Thiel,

"Model-Based Testing Design for Embedded Automotive Software," SAE World Congress, 2009.

[9]. D. Ramaswamy, R. McGee, S. Sivashankar, A. Deshpande, J. Allen, K. Rzemien, W. Stuart, "A Case Study in Hardware-In-the-Loop Testing: Development of an ECU for a Hybrid Electric Vehicle," SAE World Congress, 2004.

저 자 소 개

금 대 현 (Daehyun Kum)

2001년 계명대학교 자동차공학과 학사. 2003년 계명대학교 자동차공학과 석사. 현재, 경북대학교 전자전기컴퓨터학과 박사과정. 2003~2005 LG전자 연구원. 2005~현재 대구경북과학기술원 선임연구원.
관심분야: 임베디드소프트웨어, 테스팅자동화.
Email: kumdh@dgist.ac.kr

홍 재 승 (Jaeseung Hong)

2008년 동양대학교 컴퓨터공학부 학사. 2010년 경북대학교 전자전기컴퓨터학과 석사. 현재, 대구경북과학기술원 연구원.
관심분야:Embedded Linux Fast Boot, OSEK OS, 임베디드 시스템 소프트웨어 테스팅.
Email: psman2@dgist.ac.kr

진 성 호(Sungho Jin)

1989년 경북대학교 전자공학과 학사. 1991년 경북대학교 전자공학과 석사. 현재, 대구경북과학기술원선임연구원.
관심분야 : 임베디드 소프트웨어, 스마트 센서 / 액추에이터.
Email : sungho@dgist.ac.kr

조 정 훈(Jeonghun Cho)

1996년 KAIST 전기및전자공학과 학사. 1998년 KAIST 전기및전자공학과 석사. 2003년 KAIST 전자전산학과 박사. 2005년 하이닉스반도체 선임연구원. 현재, 경북대학교 전자공학부 부교수.
관심분야: 임베디드 시스템 최적화, 컴파일러, 소프트웨어 최적화, HW/SW Codesign
Email: jcho@ee.knu.ac.kr