저궤도 위성용 탑재소프트웨어의 검증시험 환경 구축

이재승° 최종욱 강수연 이종인
한국항공우주연구원 위성전자그룹
{jslee°, jwchoi, sykang, jilee}@kari.re.kr

Introduction to Verification Test Environment of Flight Software for LEO Satellite

Jae-Seung Lee° Jong-Wook Choi Soo-Yeon Kang Jong-In Lee
Satellite Electronics Dept., Korea Aerospace Research Institute

요 약

위성의 개발 및 제작에는 많은 비용과 기간이 소요되며, 일반적으로 사용되는 장비들과는 전혀 다른 우주환경에서 임무를 수행하게 된다. 그리고 위성의 경우에는 발사이후에 발생하는 오류들을 수정하는 것이 거의 불가능하므로 위성의 성공적인 임무완수를 위해서는 철저한 사전검증 작업들이 필요하게 된다. 특히, 위성의 궤도, 자세를 제어하고 실제적인 임무수행을 관할하는 위성탑재소프트웨어에 대한 완벽한 검증이 필요하다. 이러한 소프트웨어의 통합 및 조립시험, 검증시험을 위해 저궤도 위성의 FSW(Flight Software) 개발단계에서 실제 위성시스템과 유사한 인터페이스를 제공하는 개발도구인 STB(Software Test Bed)가 제작되며, 제작된 STB를 통한 FSW의 검증시험 및 분석을 지원하기 위한 구문분석프로그램으로 VTSP(Verification Test Script Parser)를 개발하게 된다. 본 논문에서는 이러한 STB와 VTSP에 대한 전반적인 소개와 함께 개발된 STB와 VTSP를 이용하여 실제 위성탑재소프트웨어를 검증하기 위한 시험환경에 대해 알아보고자 한다.

1. 서 론

위성을 설계, 제작 및 개발하는 데에는 많은 예산과 개발기간이 요구된다. 많은 예산과 시간을 투자하는 만큼 실패에 대한 위험부담도 커지게 된다. 또한 위성은 다른 일반제품과는 달리 우주라는 특수한 공간에서 동작하게 되므로 발사 후에 발생되는 문제점을 해결하는 것은 어렵다. 따라서 개발단계에서 체계적이고 반복적인 검증이 필요하며, 실제 위성개발기간의 상당부분이 이러한 검증시험에 소요되고 있다.

위성의 검증시험은 발사 직전까지 계속되며, 각 서브시스템별로 수행되는 시험과 함께 각 시스템간의 인터페이스 및 조립 후 전체적인 기능시험도 수행하게 된다.

본 논문에서는 이러한 검증시험 중 위성탑재소프트웨어[1]의 검증시험에 대한 전반적인 검증시험 수행환경을 설명하고자 한다. 위성탑재소프트웨어는 위성의 기본적인 기능수행에 핵심적인 역할인 자세제어, 열제어, 전력제어 등을 담당하며, 지상과 위성간의 명령 및 데이터 통신뿐만 아니라, 탑재체 하드웨어를 제어함으로서 실제적인 위성 수행임무를 관장한다. 먼저 이러한 위성탑재소프트웨어의 기능 및 개발환경에 대해 간략히 알아보고, 소프트웨어 검증시험에 사용되는 실제 위성의 시스템과 동일한 기능을 시뮬레이션 해주기 위해 제작된 STB에 대해 알아본다. 또한 STB와 소프트웨어 개발자들간의 인터페이스로 사용되는 VTSP의 기능과 VTSP를 이용한 위성탑재소프트웨어 검증시험에 대해 살펴보도록 한다.

2. 위성탑재소프트웨어의 기능

위성탑재체에 의해 수집된 데이터는 탑재컴퓨터의 메모리에 저장되고 지상국과의 통신을 통하여 지상으로 전송된다. 전송된 텔레메트리 데이터를 분석함으로서 위성의 위치, 궤도, 상태 등에 대한 정보를 점검할 수 있다. 텔레메트리 데이터의 분석결과 문제점이 발견되면 이를 해결하기 위한 명령을 전송하기도 한다.

이러한 위성의 상태데이터 및 명령과 관련된 모든 작업을 담당하는 것이 바로 탑재소프트웨어이다. 탑재소프트웨어는 위성의 하드웨어 서브시스템 제어와 명령 전송 및 수행의 역할을 담당하며, 지상명령과 임무수행을 자동적으로 수행할 수 있어야 한다.

탑재소프트웨어가 기본적으로 다음과 같은 기능들을 위성에 제공해 주어야한다.

- 업링크된 명령의 수신 및 수행[2]
- 우선순위 및 수행시간에 따른 명령의 스케쥴링
- 하드웨어에 해당 명령의 전달 및 수행
- 위성 상태데이터 수집[3]
- 센서를 이용한 전력계, 자세 제어계, 열 제어계 로직 수행
- 위성 상태데이터를 텔레메트리 프레임으로 포메팅[4] 및 대용량 메모리에 저장, 지상국과의 통신 시 실시간 데이터 저장 및 전송

이러한 탑재소프트웨어에 요구되는 기능들의 정상적인 수행여부를 확인하는 것이 소프트웨어 검증시험의 주요

목적이다.

3. 검증시험 환경

위성의 시스템 기능시험에 사용되는 STB는 위성시스템의 내·외부적인 인터페이스에 대한 에뮬레이션이 가능하도록 하기위한 장치이다. VTSP는 탑재소프트웨어 개발자들이 소프트웨어 검증시험을 수행할 때 제작된 STB로부터 전송되는 위성데이터를 수신·저장 및 분석하여 전체적인 시험과정이 자동적으로 수행되도록 하는 기능을 제공하기위한 프로그램이다.

본 절에서는 이러한 STB와 VTSP 각각의 기능 및 인터페이스에 관해 알아본다.

3.1 STB(Software Test Bed)

위성시스템은 한개 또는 그 이상의 프로세서로 구성되어진다. STB의 구성도 위성시스템의 구성에 따라 달라질 수 있다. 그림 1은 3개의 프로세서를 사용하는 경우에 대한 STB 시스템 개념도의 예를 보여준다.

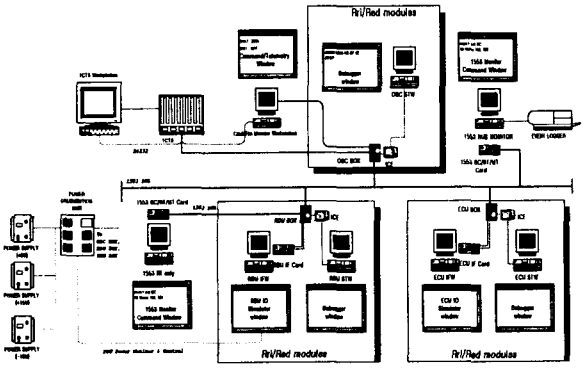

그림 1. STB Diagram

그림 1에 나타난 바와 같이 STB는 실제 위성의 하드웨어들이 직접 연결되는 것과 같은 인터페이스를 가지게 되며, 이러한 환경은 윈도우 운영시스템을 가진 PC에서 STB의 프로세서를 디버깅할 수 있는 기능을 제공해 준다. 각 프로세서와 페이로드는 1553B Bus로 연결되어 데이터 및 명령을 전달하며, 실제 위성에 탑재되는 페이로드들을 대신해서 I/O 시뮬레이터를 이용하여 그 기능을 모사하게 된다. STB에서는 실제 CPU를 사용하지 않고 CPU의 기능을 에뮬레이션 해주는 ICE(In-Circuit Emulator)를 연결하여 사용한다. ICE를 이용함으로서 일반 사용자에게 친숙한 윈도우용 PC에서 탑재소프트웨어의 빌드를 직접 업로드하는 것이 가능할 뿐만 아니라, RAM, ROM, MM(Mass Memory) 등 모든 메모리 영역에의 접근이 용이하다. 제작된 STB를 통해 인터페이스 및 오픈루프 테스트를 직접 수행할 수 있다.

STB의 주요 목적은 각 프로세서에 대한 소프트웨어 개발 및 검증시험을 위한 인터페이스 및 툴을 제공하는 것이며, 실제 위성의 통합 및 조립 전 타겟 하드웨어의 소프트웨어 검증시험을 위해 다음과 같은 기능을 제공하게 된다.

- 1553B Bus에서의 프로세서 내부 통신 디버깅
- 원격측정명령계 소프트웨어의 디버깅
- 각 서브시스템의 독립된 시험 및 통합시험
- 1553B Bus를 통한 탑재체 인터페이스 시험
- 위성탑재소프트웨어의 오류 디버깅 및 수정
- 위성시스템의 문제점 분석 및 진단, 분류

3.2 VTSP(Verification Test Script Parser)

3.1절에서 설명한 STB를 이용하여 검증시험을 수행하기 위해서는 STB로부터 전송되는 위성데이터를 전달받아 저장 및 분석하거나 지상명령을 STB로 보내기 위한 인터페이스 도구가 필요하다. STB 자체로도 탑재소프트웨어와 메모리 영역 및 1553 통신에 대한 디버깅을 수행할 수 있다. 실제로 STB에서 ICE를 통한 디버깅 툴을 사용함으로서 사용자에게 친숙한 윈도우환경에서 실시간 디버깅 및 브레이크 포인트를 이용한 단계적인 디버깅이 가능하지만, 그 절차가 복잡하고 준비과정이 오래 걸린다. 또한 반복적인 시험이 이루어져야 하는 경우, 같은 작업을 계속적으로 반복해야 한다. 이러한 반복적인 검증시험이 효율적으로 이루어지기 위해서는 명령 및 위성데이터 와 관련된 검증시험을 빠르고 자동적으로 처리되도록 할 필요가 있다. 즉, 모든 검증시험은 오랜 기간동안 지속적인 반복 및 수정을 통하여 검증하여야 함으로 테스트 수행자가 작성한 스크립트를 기반으로 검증시험이 이루어질 수 있어야 한다. 이를 위해 VTSP는 일반 사용자에게 친숙한 C-언어와 유사한 스크립트 언어를 제공한다. VTSP의 목적은 TCTS(Telemetry/Command Test Set)에 프로그래머블한 시험 인터페이스를 제공하는 것이다. 그림 2는 이러한 VTSP와 TCTS간의 연결환경을 보여준다.

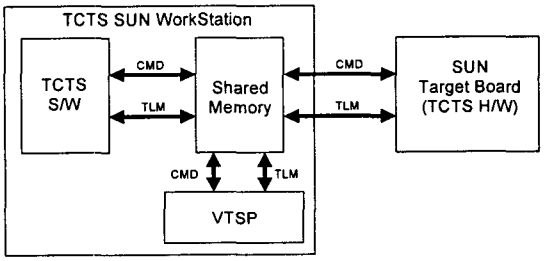

그림 2. How VTSP works with TCTS H/W

VTSP는 시험 수행자가 작성한 스크립트를 컴파일하여 실행한 후 그 결과를 자동적으로 출력하여 주는 일종의 구문분석기이다. VTSP를 개발하기 위하여 구문분석 프로그램 제작 툴인 FLEX[5]와 BISON[6]이 사용되어 진다. 검증시험을 위해 작성된 스크립트가 입력되면 FLEX

가 먼저 정해진 정규 표현식들을 인식하고, BISON은 인식되어진 정규표현의 조각들을 분석하여 주어진 문법에 맞게 분석결과를 생성한다. 그림 3은 컴파일러의 관점에서 구문분석을 수행하기 위해 FLEX와 BISON이 어떻게 서로 연결되는지를 보여준다.

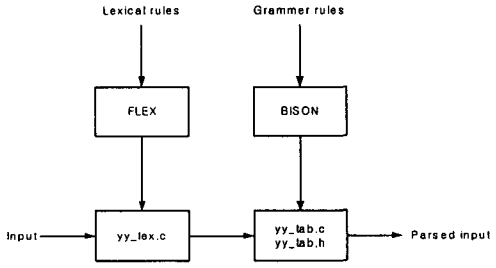

그림 3. Relation between FLEX and BISON

4. 탑재소프트웨어 검증시험

실제 위성을 운영한다고 생각하면 STB는 위성 본체에 해당한다고 볼 수 있다. ICE에서 위성탑재소프트웨어가 부팅되어 수행되며, 페이로드가 직접 장착되지는 않았지만 I/O 시뮬레이터를 이용하여 하드웨어 데이터 획득 및 명령 전송등의 인터페이스가 가능하다.

TCTS는 탑재소프트웨어에 의해 포메팅된 텔레메트리를 STB로부터 지속적으로 수신하며 사용자의 지상명령을 STB로 전송하는 기능, 즉 위성 및 지상의 송수신장치의 역할을 한다. VTSP는 TCTS와의 공유 메모리를 이용하여 다운링크된 위성데이터를 분석하고 작성된 스크립트에 따라 필요한 명령을 TCTS에 전달하며 검증시험을 위해 입력된 스크립트를 분석하고 다운링크받은 위성데이터를 이용하여 검증시험을 수행한 후 해당 결과를 자동적으로 생성하여 출력해 주는 기능을 수행하므로 지상국의 수신 데이터처리 장비에 비유될 수 있다.

위성 탑재소프트웨어의 검증시험은 오랜 준비기간과 시험기간 동안 반복적으로 수행된다. 검증시험이 완료된 후에도 위성발사 전까지 계속적인 시험이 이루어지게 되며, 검증시험의 항목에 영향을 주는 수정사항이 발생하면 관련된 모든 검증시험 항목들은 다시 시험을 수행해야 한다. 그림 4는 이러한 검증시험이 수행되는 방법을 나타낸다.

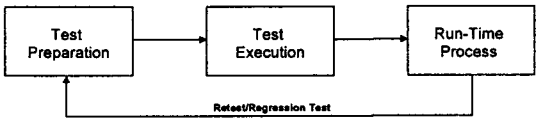

그림 4. How test goes

그림 4에서 'Retest'는 검증시험에서 'Fail'이 발생했을 경우 다시 수행하는 시험을 의미하며, 'Regression Test'는 요구조건의 변경, 시험항목 자체의 수정 또는 다른 변경 및 수정사항으로 인하여 다시 수행하는 시험을 의미한다.

5. 결 론

본 논문에서는 저궤도 위성의 탑재소프트웨어 검증시험과 관련하여 STB, VTSP 등을 이용한 검증시험 환경에 대하여 설명하였다. 많은 비용과 인원, 시간이 투자되는 인공위성 개발을 위해서는 소프트웨어뿐만 아니라 다른 모든 서브시스템에서의 검증시험은 필수적이다. 또한 발사되어 우주공간에서 임무를 수행하고 있는 위성에서 문제가 발생할 경우 이를 해결하기는 매우 어렵다. 이러한 이유로 위성개발기간의 상당부분을 검증시험에 할당하고 있다. 그러므로 검증시험 환경의 구축은 위성의 임무완수에 매우 중요한 요건이라 할 수 있다. 그러나 위성에 사용되는 프로세서 종류의 변경이나 몇 개의 프로세서를 사용할 것이냐에 따라 검증시험 환경을 새롭게 구축해야 하므로 개발기간이나 개발비용 측면에서 비효율적이다. 이러한 문제점을 해결하기 위하여 다양한 위성시스템에 적용 가능한 탑재소프트웨어 검증시험 환경에 대한 연구가 진행되고 있다.

참고문헌

[1] 이종인, "아리랑 위성 탑재 소프트웨어 소개", 한국정보과학회 가을학술발표 논문집(III), 제25권 제2호, 1998, pp.662~664.

[2] 강수연, "아리랑 위성의 Command/Telemetry 시스템", 한국정보과학회 가을학술발표 논문집(III), 제25권 제2호, pp.662~664, 1998.

[3] 이재승, "다목적실용위성 2호의 데이터 획득을 위한 자동적인 포맷테이블 생성", 한국정보과학회 가을학술발표 논문집(III), 제28권 제2호, pp.508~510, 2001.

[4] 이재승, "다목적실용위성 2호에서의 Telemetry 데이터 흐름", 한국정보과학회 봄학술발표 논문집(A), 제29권 제1호, 2002, pp.337~339.

[5] Vern Paxson, "Flex version 2.5", The University of California, 1995.

[6] Charles Donnelly, Richard Stallman, "Bison(The YACC-compatible Parser Generator", Free Software Foundation, 1995.