위성 운용을 위한 On-Board Control System 설계 및 구현

신현규*

Design and Implementation of On-Board Control System

Hyun-Kyu Shin*

Abstract

For the accomplishment of satellite's own missions, complicated control procedures and commands are required. Although absolute-Time Command and Command Sequences have been applied for controlling satellite and its operation historically, these command system only has a capability to manage sequential control process and has a limitation that cannot deal with decision and branch for corresponding to the condition of the time. To resolve above limitation, KARI has designed RTCSC which is mainly based on the existing RTCS but adopts a one-byte Op Code for supporting decision making and conditional branch. This paper introduces the design and implementation of RTCSC as On-Board Control System like OBCP, VML and IP.

초 록

위성의 임무를 성공적으로 수행하기 위해서는 복잡한 처리 절차와 이를 위한 명령이 필수적이다. 이를 위해 절대 시간 명령과 순차 명령 집합이 적용되어 위성의 운용에 사용되고 있다. 이러한 명령 체계는 순차적인 제어 흐름을 처리할 수는 있으나 해당 시점에서의 여러 조건에 대한 다양한 의사 결정 및 이에 따른 분기를 처리하기 힘들다는 단점이 있다. 이를 해결하기 위해 한국항공우주연구원 위성비행소프트웨어팀에서는 기존의 순차명령집합인 RTCS의 기본 형태를 유지하고 1 바이트의 연산 코드를 추가함으로써, 조건 판단 및 분기가 가능한 조건형 순차명령 집합인 RTCSC를 개발하였다. 여기서는 OBCP, VML 및 IP와 같은 위성 운용을 위한 On-Board Control System으로서의 RTCSC의 설계 및 그 구현에 대하여 소개한다.

키워드 : 조건형 순차 명령 집합 (RTCSC), 위성 운용 (satellite operation), On-Board Control System (OBCS)

1. 서 론

위성을 운용하기 위해서는 위성의 상태를 확인하고 정해진 절차에 따른 다양한 명령이 전달되어야 한다. 위성에 탑재되는 위성비행소프트웨어는 이러한 명령을 받아 사전에 정의된 동작을 수행함으로써 위성을 제어하고 위성이 본연의 임

접수일(2014년 5월 7일), 수정일(1차 : 6월 16일, 2차 : 6월 24일), 게재 확정일(2014년 7월 1일)
* 위성비행소프트웨어팀 / hkshin@kari.re.kr

무를 완수할 수 있도록 한다. 이와 같은 위성의 운용에는 순차적으로 전달되어야 하는 명령들이 존재한다. 예를 들어 특정 장치를 초기화하기 위해서는 장치가 작동할 수 있는 온도 조건을 허용 범위로 유지하고, 전원을 켜고, 초기값을 설정한 후, 운용 모드로 전환하는 등의 절차가 필요하다. 이러한 명령들은 지상에 의해서 순차적으로 전송될 수도 있지만, 지상에서의 제어는 오퍼레이터의 조작 및 명령 전송 여부 확인 등의 과정을 거치므로 통신 시간의 제약이 있거나, 정해진 시간 요구 조건이 있는 경우에는 적용이 어려울 수 있다. 심우주 탐사 위성의 경우는 통신 제약 사항이 더 크게 작용하게 된다. 이러한 문제점들을 해결하고자 위성비행소프트웨어에는 미리 정의된 순차 명령 집합을 수행할 수 있는 기능이 구현되어 있다. 기존의 저궤도 위성에서는 RTCS (Relative Time Command Sequence)를 이용하여 이러한 순차 명령 집합을 구현해 왔다. RTCS는 하나의 명령과 다음 명령까지 수행 시간 간격(Delta-Time)을 갖는 명령의 집합으로, 위성비행소프트웨어 내에 탑재되거나, 또는 지상 명령에 의해 새롭게 적재될 수 있다. RTCS의 수행 중, 조건에 대한 판단 및 분기가 필요한 경우에는 위성비행소프트웨어 내에 별도의 로직을 구현하고 RTCS가 해당 로직을 활성화 한 후, 관련 로직에서 판단을 수행, 필요시 조건에 맞는 다른 RTCS를 호출하여 실행하도록 하였다. 이러한 방법은 위성비행소프트웨어 내에 연관 코드를 구현함으로써 RTCS 자체의 구조를 단순화하고, 그 수행이 직관적이라는 장점이 있으나, RTCS의 수행과 관련한 로직이 위성비행소프트웨어 내에 고정적으로 구현된다는 점과 조건 및 분기가 여러 부분에서 수행될 경우에는 처리 절차가 다수의 RTCS로 단편화되고 이에 따라 위성비행소프트웨어가 구현해야 하는 로직의 관계가 복잡해지는 단점을 가지고 있다[1]. 위성의 운용에 있어 다양한 명령들이 정해진 시간 순서로 올바르게 동작되어야 하는 여러 절차들이 RTCS로 구성되어 진다는 점을 고려해볼 때, 기존의 RTCS는 위성이 정해진 동작을 수행하도록 할 수는 있으나, 상황 판단 및 그에 따른 서로 다른 처리의 수행이 어렵기 때문에 위성비행소프트웨어 내에 이와 관련한 코드를 추가하게 되고, 이는 위성의 임무 다양성 및 발사 후의 임무 변경 가능성에 대해 상당한 제약이 된다. 이러한 문제점 및 제약 사항을 해결하고자 한국항공우주연구원 위성비행소프트웨어팀에서는 On-Board Control System (OBCS)에 대한 연구를 진행해 오고 있다. OBCS의 기본 개념은 위성비행소프트웨어의 기반 위에서 동작하는 별도의 소프트웨어 처리 환경으로, 절대 시간 명령 및 순차 명령 집합의 기능 뿐 아니라, 위성의 상태를 모니터링하고, 조건 판단 및 그에 따른 분기, 처리를 수행함으로써 지상국과의 상호 작용을 최소화 할 수 있다. 미국과 유럽에서도 위성비행소프트웨어 내에서 실행되는 작고 독립적인 컴퓨터 프로그램인 OBCP (On-Board Control Procedure), VML (Virtual Machine Language) 등을 설계, 개발하여 혜성 탐사선 Rosetta, 우주망원경 Herschel과 Spitzer 에 적용한 바 있으며, 천리안 위성의 경우, 위성의 운영을 자동화하기 위하여 OBCP와 유사한 IP (Interpreted Program)를 실행할 수 있는 환경을 제공하도록 개발되었으며, Astrium에서 개발한 APL (Application Program Language)로 작성되었다[2]. 위성비행소프트웨어팀은 OBCS의 설계 및 구현을 위해, OBCP와 유사한 개념으로 위성비행소프트웨어의 개발 과정에 있어 소프트웨어의 재사용성을 높이고 궤도상에서의 임무 변경에 보다 쉽게 대응할 수 있는 RSA (Reconfigurable Software Architecture)[3], 저궤도 위성에 대한 OBCP의 적용방안[4]에 대한 연구를 수행한 바 있다. 또한, 기존의 RTCS를 확장하여 조건 판단 및 분기가 가능하며, 위성비행소프트웨어와 독립적으로 운용 절차 수립이 용이한 조건형 순차 명령 집합(RTCSC, Relative Time Command Sequence with Condition)[1]에 대한 연구를 통해 그 개념을 실제 환경에 적용 가능하도록 확장시켜 왔다. 여기서는 OBCS를 구현하기 위한 여러 방법 가운데, 기존 연구를 통해 설계한 조건형 순차 명령 집합을 저궤도 및 정지궤도, 나아가 심우주 탐사 위성에 사용될 수 있는 OBCS의 구축 관점에서 설계 및 그 구현에 대하여 소개한다.

2. 조건형 순차 명령 집합의 설계

2.1 조건형 순차 명령 집합의 요구 사항

위성의 운용을 위한 OBCS로써 새로운 조건형 순차 명령 집합에 대한 주요 요구 사항은 다음과 같다.

. Delta-time으로 연결된 순차 명령의 실행을 보장하여야 한다. (기존 RTCS의 속성)
. 특정 조건이 만족함을 기다릴 수 있어야 한다.
. 특정 위치로의 분기가 가능해야 한다.
. 특정 저장 공간(변수)를 이용하여 간단한 로직의 구현이 가능해야 한다.
. 실행 상태를 변경할 수 있어야 한다.
. 순차 명령 집합에서 시작 위치를 변경할 수 있어야 한다.
. 다른 순차 명령 집합을 수행하도록 할 수 있어야 한다.
. On-Orbit에서의 수정이 용이해야 한다.

또한, 새롭게 설계 구현되는 조건형 순차명령 집합은 기존 위성비행소프트웨어에 대한 의존성 및 종속성을 최소화하도록 하여, 다양한 플랫폼에서 동작이 가능해야 한다.

2.2 접근 방안

OBCS의 구축 관점에서 RTCSC가 가져야 하는 가장 큰 요구 조건은 RTCSC가 동작하는 해당 시점에서 위성비행소프트웨어의 메모리를 참조하여 이에 따른 조건 판단 및 적절한 분기가 가능해야 한다는 것이다. 이를 위하여 조건 판단 및 분기, 간단한 연산을 포함하는 연산코드(Op Code)를 추가하였다. 연산코드는 RTCSC의 Execution Environment가 각 연산코드를 해석하고 정해진 동작을 수행하도록 하는 내용을 표시하는 것으로, 컴퓨터 아키텍처에서의 인스트럭션과 유사하다. 연산코드는 지원해야 하는 기능에 따라 다양한 형태의 구성이 가능한데, 본 연구에서는 기존의 RTCS, 즉 Delta-time을 포함하는 명령은 “CMD" 연산코드를 할당하고, 추가적으로 조건 비교 및 간단한 산술 연산, 대입문을 위한 연산코드를 구성하였다. 각 연산코드는 처리를 위한 데이터를 갖게 되는데 이는 인스트럭션 필드로 정의하였다.

조건형 순차 명령 집합의 주요 요구 사항 중의 하나인 순차 실행을 보장하기 위하여 기존 연구[1]에서는 순차 명령이 아닌 연산코드의 위치까지는 기존 RTCS가 ATC (Absolute-Time Command)로 변환되는 방식을 그대로 이용하여 각 영역별 순차 실행을 보장하는 형태로 설계되었으나, ATC로의 변환 과정이 기존 위성 비행소프트웨어의 ATC 처리 방식에 의존적이고, ATC로 변환된 명령들이 처리될 것이라고 예상되는 시점까지 조건형 순차명령 집합의 수행이 Pending되는 점, 또한 이미 ATC로 변환된 명령에 대한 중지, 취소 시 발생하는 여러 문제의 복잡성을 감안해 볼 때, ATC로의 변환 대신 조건형 순차 명령 집합의 처리 엔진이 해당 시점에 명령을 관련 로직으로 전달(Dispatch)함으로써, 외부 컴포넌트와의 의존성을 줄이고 처리 로직을 단순화하도록 변경하였다.

특정 위치로 분기하기 위해서는 순차 명령 집합을 서술하는 단계에서 라벨(Label)을 정의하고, 분기하는 경우 라벨을 기술하도록 하였다. 이 정보는 기술된 순차 명령 집합의 컴파일 과정에서 실제 연산코드의 위치로 재해석된다.

간단한 로직의 구현을 위한 특정 저장 공간은 헤더 부분에 4개의 저장소를 배치하여, 조건형 순차 명령 집합에서 내부 카운터, 판단 조건, 지상 명령 저장 등으로 사용될 수 있도록 하였다.

조건 판단의 경우, 해당 조건은 대부분 위성비행소프트웨어 내의 변수 또는 특정 텔레메트리의 값을 이용하므로, 이를 참조하는 형식으로 하고, 순차 명령 집합을 작성하는 단계에서는 변수 이름 또는 니모닉(Mnemonic)을 기술하고, 컴파일 단계에서 실제 주소로 변환한다.

다른 순차 명령 집합에 대한 수행명령(Activation)은 기존의 RTCS가 다른 RTCS를 활성화하여 수행 할 수 있었던 것처럼 위성비행소프트웨어에 명령 체계를 동일하게 사용한다.

기존의 RTCS는 Excel을 이용하여 처리 순서를 정의, 위성비행소프트웨어 내에 탑재되기 위한 헤더 파일로 변환되는 과정을 이용하였으나, 조건형 순차 명령 집합에서는 텍스트 편집기 등으로 기술된 순차 명령 집합을 컴파일러를 통해 바이트 코드(Byte Code)를 생성하도록 하며, 이 바이트 코드를 운용 중인 위성에 전송하거나 기존 방식처럼 헤더 파일을 이용하여 위성비행소프트웨어 내에 탑재하는 것도 가능하도록 하였다.

2.3 설계

조건형 순차 명령 집합의 전체적인 운용 흐름은 다음과 같다.

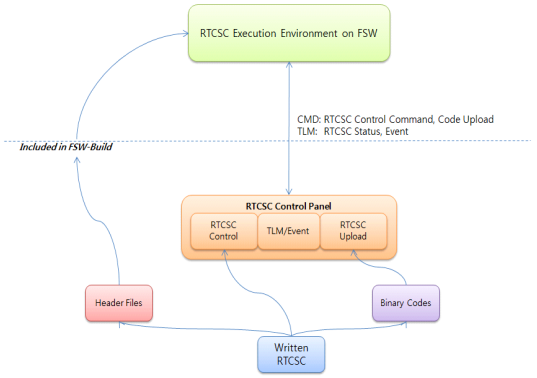

그림 1. Overall Operation

작성된 조건형 순차 명령 집합은 컴파일러를 통해 바이트 코드 형태의 바이너리 코드 (Binary Code)로 변환되며, 지상 검증 및 발사 전 환경에서는 헤더 파일의 형태로 위성비행소프트웨어 내에 탑재될 수 있다. 바이너리 코드로 변환된 조건형 순차 명령 집합은 Control Panel을 통해 위성비행소프트웨어 내에서 동작하는 순차명령집합의 처리 엔진으로 전송될 수 있다. Control Panel은 위성에 탑재된 조건형 순차 명령 집합의 동작을 제어할 수 있는 지상 환경으로 기존 위성의 명령 및 텔레메트리 환경을 그대로 이용한다. Control Panel은 위성에서 동작하는 조건형 순차 명령 집합의 상태를 소스코드 (텍스트로 기술된 조건형 순차 명령 집합)와 함께 표시함으로써 동작 상태를 보다 알기 쉽도록 하였다. 위성비행소프트웨어에는 조건형 순차 명령 집합이 동작하기 위한 환경을 구성하고, 지상으로부터 명령을 수신하고, 동작 상태를 기존 텔레메트리를 통해 지상으로 전송하도록 하였다.

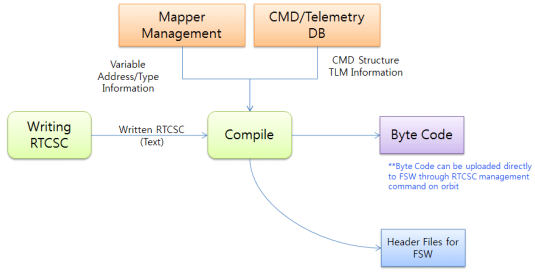

그림 2. Compilation

작성된 조건형 순차 명령 집합은 컴파일러를 통해 바이트 코드로 변환되는 과정에서 위성비행소프트웨어 내의 변수 또는 텔레메트리 니모닉에 대한 실제 주소로의 변환이 이루어지며, 이 정보는 Mapper Management에 의해 관리된다. 특정 위치로의 분기를 지원하기 위한 라벨에 대한 내용도 실제 연산코드로의 위치로 변경된다.

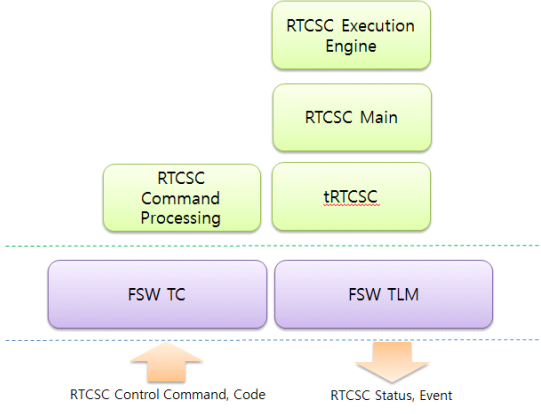

그림 3. Execution Environment on FSW

그림 3은 위성비행소프트웨어 내에 위치하는 조건형 순차 명령 집합의 실행 환경에 대한 것으로 지상으로부터 전송된 명령은 기존의 명령 처

리 로직을 거쳐 조건형 순차 명령 집합에 대한 명령 처리 로직에서 관련된 동작을 수행한다. 별도의 tRTCSC라는 태스크(Task)를 사용하여 기존의 위성비행소프트웨어의 동작과 분리하였으며, 이 태스크 상에서 실질적인 동작이 이루어지게 된다. 조건형 순차 명령 집합의 동작 상황 및 이때 발생하는 이벤트는 기존의 텔레메트리를 통해 지상으로 전송되며, Control Panel을 통해 실시간으로 시현될 수 있다.

그림 4는 조건형 순차 명령 집합의 상태도(State Diagram)을 나타낸 것으로, Empty, Ready, Activated 및 Terminated의 상태를 갖는다.

. Empty: 아직 조건형 순차 명령 집합이 로드되지 않은 상태
. Ready: 지상 명령 또는 소프트웨어 빌드를 통해 조건형 순차 명령 집합이 적재된 상태
. Activated: Activation 명령에 의해 수행 중인 상태
. Terminated: 조건형 순차 명령 집합의 실행이 종료된 상태

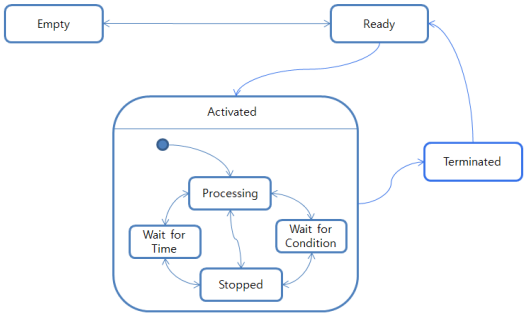

그림 4. RTCSC의 상태도

기존 연구[1]에서는 Terminated 상태가 존재하지 않고, Activated 상태에서 실행이 종료되면, Ready 상태로 자동 변경되는 형태로 설계하였으나, 종료된 상태를 명시적으로 나타내기 위하여 TERM 연산코드에 의해 Terminated로 변경되며, 이후 명령에 의해 Ready 상태로의 변경이 가능하도록 하였다. Activated 상태의 세부 상태는 다음과 같다.

. Processing: 조건형 순차 명령 집합이 처리 엔진에 의해 해석되어 처리되고 있는 상태
. Wait for Time: 조건형 순차 명령 집합에 기술된 시간 정보에 의해 일시 정지(Pending) 되어 있는 상태
. Wait for Condition: 조건형 순차 명령 집합에 기술되어 있는 조건문이 만족하기를 기다리며 일시 정지 되어 있는 상태
. Stopped: 지상 명령 등에 의해 조건형 순차 명령 집합의 수행이 일시 중지되어 있는 상태

기존 연구[1]에서는 조건형 순차 명령 집합의 동작을 위한 헤더를 그림 5와 같이 4개의 변수 영역, 상태, 현재 실행 위치 및 시간 정보를 포함하는 32 바이트 크기로 설계하였다. 이후 설계 변경 및 구현을 통해, 표 1과 같은 디스크립터(Descriptor)를 이용하여 헤더를 대체하도록 하였다.

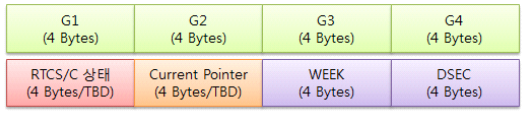

그림 5. 이전 구현에서의 헤더 구성

디스크립터에는 처리 엔진이 개별적인 조건형 순차 명령 집합을 관리하기 위한 RTCSC 번호를 포함하여 동작 상태, 타이머의 설정 여부, 관련한 시간 정보 및 로직 구현을 위한 내부 저장 영역이 위치한다. 내부 저장 영역은 기존과 동일하게 4바이트 크기의 영역 4개를 사용하였고, 각 영역별로 타입을 다르게 설정할 수 있도록 하기 위하여 각 저장 영역의 타입을 디스크립터 내에 기록하도록 하였다.

기존 연구[1]와의 큰 차이는 디스크립터의 후반부에 스케줄링과 관련되어 태스크 정보가 위치한다는 점이다. 기존 연구[1]에서는 다수의 조건형 순차 명령 집합이 Round-Robin 방식으로 스케줄링 되도록 설계하였으나, 다양한 임무 특성

을 지원하고, 보다 효율적으로 자원을 사용하기 위하여 자체적인 스케줄링 기법을 고안, 이를 위한 여러 정보를 디스크립터에 기술하여, 처리 엔진이 해당 정보를 활용하도록 하였다. 디스크립터의 정보는 지상 명령을 통해 변경이 가능하며, 이는 조건형 순차 명령 집합의 동작에 직접적인 영향을 미치게 된다.

표 1. 개선된 RTCSC 디스크립터

RTCSC_NO	RTCSC Status	Timer Status
Timer OBT		
Timer Branch Target		
PC	Break Point	
Reference OBT		
Local Variables		
LV Type		
Code Size		
Pending CMD	Task Level	Task Duration
Task Period	Task Offset	Task Alloc. / Spare

조건형 순차 명령 집합의 바이트 코드는 그림 6과 같은 형태로 이루어진다. 기존의 RTCS의 형태를 그대로 유지하면서 그 앞에 연산코드 1 byte가 추가되었다. 기존의 RTCS에서 RTC를 이루는 명령의 경우 "01"의 연산코드를 가지며, 조건이나 분기, 기타 연산을 위한 연산코드는 비명령의 형태로 처리된다.

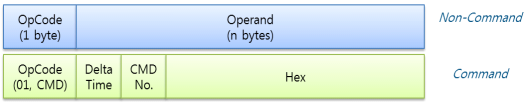

그림 6. 바이트 코드 형식

- **SET** : 저장 공간 **(LV1~LV4)**에 특정 값을 설정
- **INC** : 저장 공간 **(LV1~LV4)**의 값을 정해진 만큼 증가
- **Condition**
  - **CG: X**가 **Y**보다 클때까지 **Pending**
  - **CGE: X**가 **Y**보다 크거나 같을때까지 **Pending**
  - **CE: X**가 **Y**와 같을 때가지 **Pending**
- **Branch**
  - **B:** 무조건 분기 **(Goto)**
  - **BG : X**가 **Y**보다 큰 경우 분기
  - **BGE : X**가 **Y**보다 같거나 큰 경우 분기
  - **BE: X**가 **Y**와 같은 경우 분기
- **Wait :** 주어진 시간 만큼 **Pending.**

그림 7. 연산코드의 예

그림 7에서와 같이 연산코드는 매우 간단한 연산 및 조건 판단이 가능한 형태로 설계되었으며, 특정 플랫폼에서의 확장을 위해 필요한 경우, 내장 함수를 사용할 수 있도록 하였다.

표 2. 연산코드 목록

CMD
SETTYPE
SET
INC
WAIT
CE, CNE, CG, CGE, CL, CLE
BE, BNE, BG, BGE, BL, BLE
GOTO
SETTIME, RESETTIME
YIELD
TERM

표 2는 이번 연구를 통해 제안, 확정된 연산코드의 목록으로 기존 RTCS에서의 명령을 나타내는 CMD를 비롯하여, 내부 저장소의 타입을 결정하고, 해당 값을 설정하는 SETTYPE 과 SET, 조건 판단 및 분기에 관련한 Cxx, Bxx 를 포함한다. SETTIME 및 RESETTIME은 타이머의 동작을 제어하는 연산코드로 타임아웃 시간을 정의할 수 있으며, 이때 주어진 연산코드로의 Jump를 통하여, 타임아웃이 발생한 경우의 처리 로직을 기술할 수 있다. YIELD는 스케줄링과 관련한 유일한 연산코드로 자신에게 부여된 실행 우선권을

스케줄러에게 양도한다. TERM은 조건형 순차 명령 집합의 상태를 Terminated로 변경하여, 수행이 종료됨을 알리게 된다.

3. 조건형 순차 명령 집합의 구현

3.1 구현 환경

조건형 순차 명령 집합을 실제 위성에 적용하기 위해서는 그림 1과 같이, 조건형 순차 명령 집합의 작성 단계에서 컴파일, 바이너리 코드 생성, 위성비행소프트웨어에의 적재 및 On-Orbit에서의 코드 전송, 위성비행소프트웨어 상에서의 동작 환경, 지상에서의 동작 제어 환경이 모두 구현되어야 한다. 본 연구에서는 컴파일러, Execution Environment, Control Panel 등 RTCSC가 작성되고 이를 On-Board에서 동작하도록 하기 위한 모든 구성 요소를 구현함으로써 조건형 순차 명령 집합이 OBCS로써 실제 위성의 운영에 활용될 수 있는 지 확인하였다. 이를 위한 각 구성 요소의 개발, 구현 환경은 표 3과 같다.

표 3. 구현 환경

구성 요소	구현 환경
Compiler, Binary Code Generator	C#, Java Visual Studio 2008, Eclipse COCO/R
Execution Environment	C Tornado, Eclipse GK2-FSS
Control Panel	Java Eclipse
Execution Time Measurement	C, Java Eclipse GK2-FSS

표 3 은 각 구성 요소 별 구현 환경에 대한 내용으로, 각 구성 요소에 알맞은 개발 환경에서 연구를 진행하였다. 그림 8은 조건형 순차 명령 집합을 구현하기 위한 여러 개발 환경에 대한 모습이다.

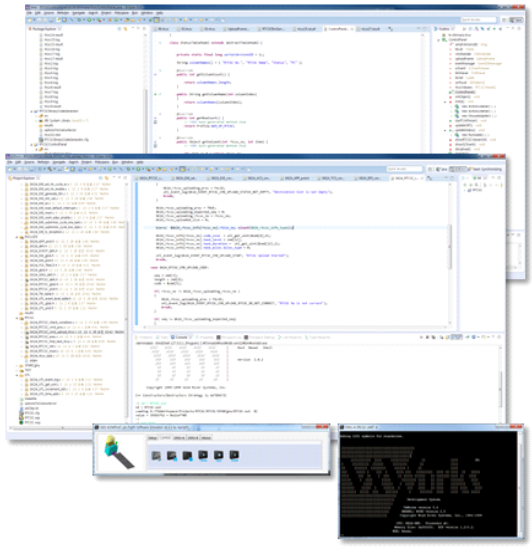

그림 8. 개발 환경

3.2 컴파일러 및 바이너리 코드 생성기

텍스트 형태로 작성된 조건형 순차 명령 집합을 위성비행소프트웨어에 적재하거나 바이너리 코드로 만들기 위해서는 이에 대한 구문을 분석하고, 위성비행소프트웨어에 탑재 가능한 형태로 변환하는 과정이 필요하다. 컴파일러 및 바이너리 코드 생성기 (Binary Code Generator)는 이 과정을 처리하며, 컴파일러는 C# 환경에서 COCO/R을 이용하여 구문 분석 및 파서를 생성하였다. 컴파일러를 통해 생성된 1차 해석 결과는 바이너리 코드 생성기를 통해 위성으로 전송될 수 있는 형태로 변환되며, Java로 구현되었다. 이 과정에서는 Control Panel에서 현재 진행 상태를 텍스트 형태의 원본 코드와 연동하여 시현하기 위한 소스 코드 정보를 생성, 바이너리 코드에 부가 정보로 포함하도록 하였다.

```
_INIT:
   SET %G1, 100;
   SET %G2, 1;
   ALIAS %G2, DeviceNumber;

_CHANGE_TO_NORM:
   CMD 20, CECMD01;
   CMD 20, CECMD02;
   CMD 25, CECMD03;
   CMD 20, CESETNORM;
   CE TEAMODE, "NORM";

_MONITORING_START:
   BE DeviceNumber, 2, _DEVICE_2;
   BE DeviceNumber, 3, _DEVICE_3;

_DEVICE_1:
   CMD 10, CEACQDATA1;
   GOTO _DATA_CHECK;

_DEVICE_2:
   CMD 10, CEACQDATA2;
   GOTO _DATA_CHECK;

_DEVICE_3:
   CMD 10, CEACQDATA3;

_DATA_CHECK:
   WAIT 100;
   BNE TEAACQSTS, "CMPL", _TRIGGER_C;
   BG TEACCQMAX, %G1, _TRIGGER_C;
   INC DeviceNumber, 1;
   BNE DeviceNumber, 4, _MONITORING_START;
   SET DeviceNumber, 1;
   GOTO _MONITORING_START;

_TRIGGER_C:
   CMD 10, CEISOLATE;

_END:
   TERM;
```

그림 9. 텍스트로 작성된 RTCSC의 예

그림 9는 텍스트로 작성된 조건형 순차 명령 집합의 예시로써, 각 부분이 라벨을 이용하여 구별되고 있으며, 앞에서 언급한 여러 연산코드를 이용하여 작성되었다. 텍스트 형태의 조건형 순차 명령 집합은 컴파일러 및 바이너리 코드 생성기를 통해 그림 10과 같은 형태의 바이너리 코드로 변환된다.

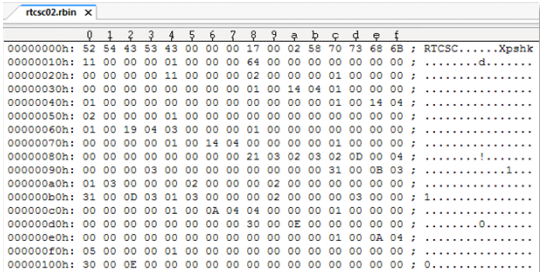

그림 10. 바이너리 코드

바이너리 코드에는 조건형 순차 명령 집합의 운영과 관련된 Descriptor 정보가 함께 기록된다.

3.3 Execution Environment

조건형 순차 명령 집합은 서론에서 언급한 OBCP, VML, IP 등처럼 위성비행소프트웨어 위에서 실행되는 독립적인 소프트웨어 동작 환경이다. 따라서 조건형 순차 명령 집합이 정상적으로 동작하기 위해서는 위성비행소프트웨어 내에 조건형 순차 명령 집합을 해석하고, 정해진 동작을 처리하기 위한 환경이 필요하다. Execution Environment는 위성비행소프트웨어에서 조건형 순차 명령 집합을 운용하기 위한 환경으로, C로 작성되었으며, VxWorks에서 하나의 태스크 형태로 작동하도록 설계, 구현하였다. 본 연구에서는 tRTCSC라는 이름으로 태스크를 구성하였으며, 기존 위성비행소프트웨어에서 주 작업을 처리하는 상위 태스크보다 한 단계 낮은 우선순위를 갖도록 설계하였다. 이 태스크를 통해 기존 명령 처리 절차에 의해 Execution Environment로 전달된 RTCSC 관련 명령을 처리하고, 스케줄링을 통하여 정해진 시간과 순서에 맞게 RTCSC를 연산코드별로 실행한다. RTCSC와 관련된 여러 정보를 담은 텔레메트리는 기존의 텔레메트리 처리 절차를 그대로 활용하도록 설계, 구현하였다. 실제 위성비행소프트웨어를 모사하기 위하여 위성비행소프트웨어 팀에서 자체 개발한 GK2-FSS 기반에서 연구를 수행하였다.

3.4 Control Panel

위성의 상태 및 임무 수행과 관련된 다양한 정보는 텔레메트리를 통하여 지상으로 전송되며, 지상에서는 이를 분석하여 위성의 상태를 파악, 이에 따른 적절한 조치를 취하게 된다. 조건형 순차 명령 집합의 Control Panel는 기본적인 명령/텔레메트리의 기능과 더불어 텔레메트리를 재가공하여 현재의 동작 상태를 소스 코드 상에 함께 표시함으로써 운영자가 보다 편리하게 진행 상황을 확인할 수 있다. 이러한 Control Panel은 다음과 같은 네 가지로 그 기능을 구분할 수 있다.

. 텔레메트리 분석을 통한 조건형 순차 명령 집합의 작동 상태 모니터링
. 조건형 순차 명령 집합의 상태를 변경하는 명령 전송
. 텔레메트리를 통해 전달된 위성체 이벤트 표시
. 바이너리 코드 생성기를 통해 생성된 바이너리 코드를 위성비행소프트웨어에 전달하는 코드 업로드

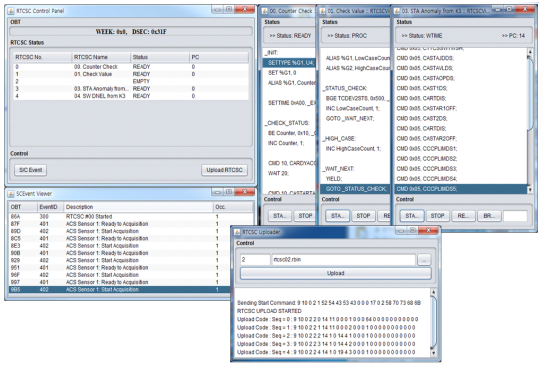

그림 11. Control Panel

그림 11은 동작중인 Control Panel의 예를 보여주는 것으로, 작동상태를 소스 코드와 연관하여 표시하며, 바이너리 코드에 대한 자동 업로드도 확인할 수 있다.

3.5 Execution Time Measurement

각 조건형 순차 명령 집합은 디스크립터에 매 동작 주기마다 수행 가능한 시간이 기록되어 있다. Hard Real-time[5]인 위성비행소프트웨어 환경에서 정해진 시간을 넘어서는 것은 절대적으로 피해야 한다. 조건형 순차 명령 집합의 스케줄러는 지정된 시간을 넘지 않도록 보수적으로 스케줄링을 수행한다. 이를 위해 각 연산코드의 평균 수행 시간 및 운용 환경의 제어 로직 -스케줄러 자체의 수행시간, 타이머 확인, 연산코드 Fetch 등-에 대한 평균값도 산출하여 이용하였다. 이때 시간 측정을 위해 GK2-FSS에 구현된 TRACE 기능을 활용하였으며, 측정 단위는 50 nsec이다.

```
YIELD      AVG: 66   (  3.3  usec )
CE         AVG: 1294 ( 64.7  usec )
WAIT       AVG: 402  ( 20.1  usec )
RESETTIME  AVG: 121  (  6.05 usec )
INC        AVG: 556  ( 27.8  usec )
SETTIME    AVG: 559  ( 27.95 usec )
SET        AVG: 533  ( 26.65 usec )
SETTYPE    AVG: 321  ( 16.05 usec )
TERM       AVG: 30   (  1.5  usec )
BE         AVG: 1245 ( 62.25 usec )
CMD        AVG: 897  ( 44.85 usec )
GOTO       AVG: 171  (  8.55 usec )
```

그림 12. 연산코드 별 수행시간

그림 12 는 측정된 연산코드 별 수행시간이며, 그림 13 은 조건형 순차 명령 집합이 동작하는 tRTCSC 태스크의 수행 시간 제한을 2000 usec으로 설정했을 때의 각 태스크 별 수행 시간 결과를 나타낸 것이다.

```
OBT: f5e   tONE    38.25 usec  tRTCSC  1793.15 usec    tTWO    19.3 usec
OBT: f5f   tONE    38.25 usec  tRTCSC  1895.85 usec    tTWO    19.3 usec
OBT: f60   tONE    38.25 usec  tRTCSC  1793.15 usec    tTWO    19.3 usec
OBT: f61   tONE    38.25 usec  tRTCSC  1723.6 usec tTWO    19.3 usec
OBT: f62   tONE    38.25 usec  tRTCSC  1793.15 usec    tTWO    1329.3 usec
OBT: f63   tONE    38.25 usec  tRTCSC  1895.85 usec    tTWO    19.3 usec
OBT: f64   tONE    38.25 usec  tRTCSC  1793.15 usec    tTWO    19.3 usec
OBT: f65   tONE    38.25 usec  tRTCSC  1895.8 usec tTWO    19.3 usec
OBT: f66   tONE    38.25 usec  tRTCSC  1808.8 usec tTWO    19.3 usec
OBT: f67   tONE    38.25 usec  tRTCSC  1895.85 usec    tTWO    19.3 usec
OBT: f68   tONE    38.25 usec  tRTCSC  1793.15 usec    tTWO    19.3 usec
OBT: f69   tONE    38.25 usec  tRTCSC  1889.9 usec tTWO    19.3 usec
OBT: f6a   tONE    38.25 usec  tRTCSC  1793.15 usec    tTWO    19.3 usec
```

그림 13. 태스크 별 수행시간

분석 결과, 실제 수행 시간은 평균적으로 약 1880 usec이었으며, 수행 시간 제한인 2000 usec을 초과하는 경우가 없는 것으로 나타나, Hard Real-time 환경에서의 스케줄링 요구 사항을 만족함을 확인하였다.

4. 결 론

위성의 임무가 다양해지고, 복잡해지면서 위성비행소프트웨어 환경에서 독립적으로 운영될 수 있는 작은 프로그램인 OBCS가 대두되었고, 미국과 유럽 등은 VML 및 OBCP 등을 개발 적용하여왔고, 천리안 위성에도 IP 가 구현되어 운용되고 있다. 위성비행소프트웨어팀에서는 OBCS에

대한 연구를 수행하였고, ATC와 RTCS를 통해 이루어지는 위성 운영 방식을 개선, 대체할 수 있는 새로운 조건형 순차 명령 집합을 설계하였다. 기존의 순차 명령 집합인 RTCS의 단순한 구조는 그대로 유지하고, 연산코드를 추가함으로써 조건 판단 및 분기 등의 처리 로직이 가능한 조건형 순차 명령 집합의 설계 및 구현을 통해 위성비행소프트웨어로의 직접적인 구현이 없이도 제어 로직 및 분기 처리를 수행할 수 있음을 확인하였다. 조건형 순차 명령 집합의 작성에서 위성비행소프트웨어에의 통합, 업로드, 실행 제어, 모니터링에 이르는 전 과정을 설계, 구현하였다. 또한 현재의 위성비행소프트웨어 환경 및 조건형 순차 명령 집합에 특화되어 설계된 고유의 스케줄러를 통해 응용프로그램 수준에서의 태스킹을 설계, 적용하였고 이는 위성비행소프트웨어와의 의존성을 최소함으로써 다른 플랫폼으로의 이식성을 높였다. 본 연구를 통해 조건형 순차 명령 집합이 위성비행소프트웨어 위에서 독립적으로 운영 가능한 OBCS로써 활용될 수 있음을 확인하였다. 나아가 본 연구 결과를 바탕으로 보다 다양한 기능을 제공하고 범용적인 프로그래밍이 가능한 시스템의 설계 및 구현과 관련한 연구를 수행할 계획이다.

참 고 문 헌

1. 신현규, “위성운용을 위한 조건형 순차 명령 집합 설계”, 한국정보처리학회 학술발표대회, 제 19권 제 2호, 2012, pp. 48-50
2. 박수현, “위성의 운영 프로그래밍을 위한 탑재소프트웨어 기술 동향”, 항공우주산업기술동향 제 11권 제 1호, 2013, pp. 64-74
3. 신현규, 천이진, “위성 탑재 소프트웨어를 위한 Reconfigurable Software Architecture", 한국정보처리학회 학술발표대회, 2010, 제 17권 제 2호, pp. 1555~1557
4. 신현규, 이재승, 최종욱, 천이진, “저궤도 위성에서 OBCP의 구현 방안”, 한국우주과학회보, 2011, 제 20권 제 2호, pp.65
5. Burns, A, “Scheduling hard real-time systems: a review“, Software Engineering Journal, 1991, Vol. 6, Issue 3, pp. 116-128