우주비행체의 실시간 시스템 및 소프트웨어 기술 동향

김주년, 천이진
한국항공우주연구소

1. 서론

일반적으로 인공위성과 인공위성 발사체는 많은 양의 데이터를 실시간으로 처리하고 실시간 출력을 보장해야 하는 시스템이다. 우주비행체의 실시간 시스템에서 가장 중요한 것은 실시간 제어가 가능해야 하며 고신뢰성을 보장해야 한다는 것이다. 대부분 우주비행체의 실시간 시스템은 run-time kernel을 사용하거나 실시간 운영체제를 사용한다. 본 논문에서는 실시간 시스템 및 실시간 운영체제의 일반적인 개념, 실시간 운영체제의 시장 현황, 대표적인 실시간 운영체제의 특징 및 사용 예를 기술한다 특히 우리나라가 보유하고 있는 인공위성을 위주로 위성의 실시간 시스템이 어떻게 구성되었는지에 관해 서술한다. 또한 인공위성 발사체의 실시간 시스템에 관하여 일부 기술하였다.

2. 우주비행체의 실시간 시스템 및 운영체제

2.1 실시간 시스템 및 실시간 운영체제

일반적으로 실시간 시스템과 실시간 운영체제는 지정된 시간 제한 내에 확실한 출력을 보장하는 시스템 또는 운영체제라고 할 수 있다. 실시간 시스템과 실시간 운영체제는 보는 각도에 따라서 다양하게 정의되었으며 다음은 이에 대한 몇 가지 경우이다[1].

- 실시간 시스템은 시스템의 정확도가 논리적인 계산 결과의 정확도뿐만 아니라 결과를 전송하는 시간에 대한 정확도를 가져야 한다.
- Koymans, Kuiper, Zijlstra-1988: 실시간 시스템은 시스템과 상관없이 진행하고 있는 비동기적인 환경과 유기적인 관계를 유지시켜 주는 시스템이다.
- IEEE 610.12-1990: 실시간 소프트웨어는 외부 프로세스가 발생하는 실제 시간동안 계산이 수행되도록 운영 시스템이나 운영 모드를 유지시켜줌으로써 외부 프로세스를 제어, 감시, 또는 외부 프로세스에 응답하기 위해 계산 결과를 제한된 시간 내에 사용할 수 있도록 해야 한다.
- Martine Timmerman: 실시간 시스템은 예측 불가능한 외부 자극에 대해 지정된 제한 시간 내에 예측가능한 방법으로 응답하는 시스템을 말한다.

실시간 시스템에 대한 정의가 여러 가지로 표현되지만, 위의 정의에서도 알 수 있듯이 시스템이 시간과 밀접한 관계를 가진다는 것을 알 수 있다. 이 중에서도 주목할 만한 것은 예측 불가능한 외부 자극에 대한 예측 가능한 응답을 하는 시스템이다. 예측 가능한 시스템을 형성하기 위해서는 시스템의 모든 요소 즉, 하드웨어와 소프트웨어는 예측 가능한 시스템을 만족하도록 구성되어야 한다. 실시간 시스템을 지정된 시간 제한(deadline)으로 구분하면 다음과 같다

- Hard real-time: 시스템이 시간제한을 넘기면 시스템에 치명적인 결과가 발생.
- Firm real-time: 시스템이 시간제한을 넘기면 시스템 응답의 질적 저하가 발생.
- Soft real-time: 시간제한을 넘기더라도 시스템을 복구할 수 있으며, 시스템 응답의 질적 저하는 받아들일 수 있음.
- Non real-time: 데드라인을 만족시키지 않더라도 시스템에 영향을 주지 않음.

우주비행체의 내장형컴퓨터(Embedded Computer)는 작은 운용체제라고 할 수 있는 Executive를 포함하는 커널로만 제작이 되는 경우도 있으며 많은 양의 데이터 처리가 필요한 시스템일 경우에는 실시간 운영체제를 탑재하게 된다.

항공우주 분야의 실시간 시스템은 앞서 설명한 실시간 시스템 및 운영체제의 개념과 유사하다. 그러나 응용 소프트웨어 부분에서 우주비행체 탑재컴퓨터의 고 신뢰성을 확보하기 위한 여러 가지 기술들이 추가적으로 포함되어야 한다

실시간 운영체제는 컴퓨터의 자원 혹은 태스크들을 직접 관리하며 임무에 따라 이들 자원을 제어한다. 이것은 태스크 스케쥴링, 시간관리, 인터럽터 관리, 입출력 관리 및 주변장치 관리를 수행할 뿐만 아니라 자가진단(self diagnose), BIT(Built In Test)를 수행하며 메모리 결함 관리등을 수행한다. 항공우주 분야에 사용되는 상용 실시간 운영체제의 경우에는 이와 같은 기본 기능 외에 시스템개발에 편리하고 다양한 기능을 제공하는 것이

일반적이다.

2.2 실시간 운영체제의 시장 현황

현재 사용되고 있는 실시간 운영체제는 상용으로 판매되는 제품과 무료로 배포되는 제품을 합치면 그 수는 엄청나게 많다. 실제 운영체제를 선택할 때 가장 궁금한 것은 어떤 제품이 가장 많이 사용되는 가 하는 것이다. Real-Time Magazine[2] 조사에 의하면 RTOS(Real-Time *Operating System* 실시간 운영체제)를 사용하고 있는 회사는 응용 시스템의 크기와 형태에 따라서 하나 이상의 RTOS를 이용하는 것으로 나타났다. 그림 1은 각 RTOS 사용에 대한 퍼센트율과 각 회사가 사용하고 있는 RTOS를 1, 2, 3 순위로 구분하여 나타낸 것이다. 그림에서 알 수 있듯이 1순위로 가장 많이 사용되고 있는 RTOS는 OSE이며 시장 점유율은 16%정도이다. 하지만 2순위와 3순위로는 OSE의 점유율이 낮다는 것을 알 수 있다. 반면 VxWorks의 경우 각 순위에 대해 고르게 사용되고 있는 것을 알 수 있다. 전체적으로 볼 때 VxWorks가 가장 많이 사용되고 있는 것을 알 수 있다 (그림 2). 그림 2에서 알 수 있듯이 전체적인 시장 점유율로 볼 때 VxWorks가 25%정도 차지한다.

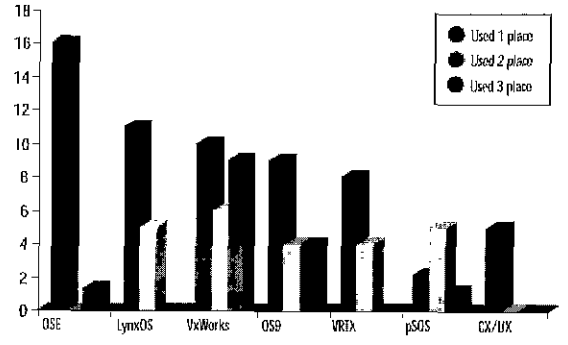

그림 1. RTOS 시장 점유율(1, 2, 3 순위).

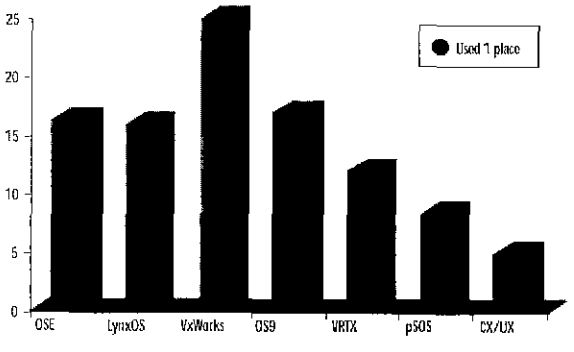

그림 2. RTOS 시장 점유율(전체).

또한 조사 결과에 의하면 50%정도가 현재 사용하고 있는 RTOS를 바꾸고 싶어하며, 39%는 사용하고 있는 RTOS를 그대로 사용하기를 원하고 11%는 미결정 상태인 것으로 나타났다. 이것은 아직까지 RTOS 시장이 아직 안정화되지 않았음을 보여준다. 그리고 RTOS를 변경하고자 하는 가장 큰 이유는 사용 중인 RTOS가 사용하고자 하는 프로세서를 지원하지 않기 때문이라고 조사되었다. 이것은 응용 시스템을 개발할 경우 사용할 프로세서를 먼저 결정한 후 다음 RTOS를 결정하기 때문이다 따라서 어떤 RTOS를 사용할 것인가 보다 어떤 프로세서를 사용할 것인가에 대해 더욱 신중해야 한다. 일단 사용할 프로세서가 결정되고 나면 RTOS 선택에 있어서 RTOS가 지원하는 언어, 디버거, 지원되는 컴파일러가 중요한 변수가 된다. 즉, RTOS가 사용할 프로세서를 지원하더라도 개발에 사용할 언어 또는 컴파일러를 지원하지 않거나 디버거 환경이 좋지 않을 경우에는 전체 시스템 개발이 어렵게 된다. 그림 1과 2에서 나타난 RTOS는 VRTX를 제외하고 다음과 같은 일반적인 사용자 불만사항을 가지고 있다[1].

- 구입 비용이 너무 비쌈.
- 가격에 비해 성능이 낮고 디버거 환경이 열악함.
- 응용 프로그램을 변경하기에는 시간이 오래 걸리기 때문에 사용 중인 RTOS를 변경하지 못함.

표 1. RTOS 제품별 특성.

Product Name	Target CPU Supported	Language	Multitasking Strategy	Debugger Supported
OSE	68K, ARM, MPC, PPC	C&C++	Round robin, time slice, fixed priority, rate monotonic scheduling, cooperative multitasking, proprietary	SingleStep Debugging Solution
Lynx OS	MIPS MPC, PPC x86, μSPARC	Ada Assem, C&C++ Fortran Java	Round robin, time slice, fixed priority, tasks can dynamically alter priorities rate monotonic scheduling	TotalView TimeScan Insure++ Totsl/db,gdb
Vx-Works	68K, PPC x86 i960 ARM, MIPS SPARC RAD6000	C&C++ Assem Java	multitasking, unlimited number of tasks, preemptive and round-robin scheduling, fast and deterministic context switching, 256 priority levels	Symbolic debugging and disassembly
VRTX	68K, PPC ARM x86	Assem, C&C++ Java	Round robin, time slice, fixed priority, tasks can dynamically alter priorities	XRAY
pSOS	68K, PowerPC MPC ARM, MIPS x86	Assem, C&C++ Java	Round robin, time slice, fixed priority, tasks can dynamically alter priorities, rate monotonic scheduling, cooperative multitasking proprietary	CAD-UL, ARM Consortium third party integration
INTEGRITY	PowerPC	Ada Assem, C&C++	Round robin, time slice, tasks can dynamically alter priorities, proprietary	MULTI Source-Level Debugger
ThreadX	68HC1X, 68K MPC, PPC ARM, MIPS x86, SPARC TI DSPs	Assem, C&C++	Round robin time slice, fixed priority, tasks can dynamically alter priorities, cooperative multi-tasking, proprietary	MULTI SecCode SingleStep CodeWarrior XRAY

표 1은 시장 점유율이 높은 RTOS 및 항공우주 산업에 사용되고 있는 RTOS 제품 중 일부의 특성을 나타낸 것이다. 표에서는 RTOS가 지원하는 프로세서, 언어, 디

버거를 나타내었고, 또한 각 RTOS의 멀티태스킹 특징을 요약하였다. 표에서도 알 수 있듯이 하나의 RTOS가 모든 프로세서를 지원하지 않기 때문에 시스템 개발 시 사용되는 프로세서가 RTOS 선택에 매우 큰 영향을 줌을 알 수 있다. 이외에도 RTOS 선택 시 고려되어여 할 사항은 요구되는 최대 ROM 및 RAM 바이트 수, 최대 지원되는 태스크 숫자, 태스크당 요구되는 RAM 바이트 수, 컨텍스트(Context) 스위칭 시간, Resource Locking 기능, 메시지 전송 방법, 소스코드 제공 여부 및 비용 등을 들 수 있다.

실제 RTOS를 선택함에 있어서 어떤 RTOS가 가장 많이 사용되는 여부가 중요하다기보다는 RTOS가 지원하는 프로세서, 컴파일러, 언어, 디버거를 포함하는 개발 환경, 그리고 시스템에 적합한 실시간 설계 개념을 포함하는 여부가 더 중요하다.

3. 실시간 운영체제의 항공우주분야 응용

항공우주분야의 각 회사는 대부분 독자적인 실시간 운영체제를 개발 사용하고 있고, 상용 프로그램을 이용할 경우에도 개발 환경이 잘 공개되지 않기 때문에 사용하는 RTOS를 파악하기란 매우 힘들다. RTOS를 개발하고 있는 업체가 제공하는 성공 사례나 응용 예를 통해 항공우주분야 회사가 사용하고 있는 RTOS를 간접적으로 알 수 있다.

VxWorks의 경우는 최근 NASA JPL의 Pathfinder 미션에서 사용되었고, LynxOS의 주 소비자가 Aerospatiale, DASA, Lockheed Martin, NASA, TRW 등임을 고려할 때 이들 회사가 개발한 시스템의 일부로서 사용되었음을 짐작할 수 있다. VRTX의 경우는 NASA의 TOMS-EP(Total Ozone Mapping Spectrometer Earth Probe)와 다목적 실용위성 (KOMPSAT: Korea Multi-purpose Satellite) 및 ROCSAT의 실시간 운영체제로서 사용되었다. OS9은 군용 VME와 Compact PCI 시스템 및 Patriot 미사일 유도 시스템과 space shuttle의 launch pad에도 사용되었다. 또한 OSE의 주 소비자가 Air Force Research Lab, Boeing North America, FMT Aircraft Gate support Systems, Lockheed Martin임을 고려할 때 이들 회사의 개발 프로그램에 사용되었음을 알 수 있다. INTEGRITY는 Boeing-Sikorsky 팀이 RAH-66 Comanche 헬기 개발에 사용하였다. 이와 같은 경향은 각 회사가 과거 독자적으로 실시간 운영체제를 개발 사용하던 것으로부터 상용 실시간 운영체제를 이용하고 있다는 것을 보여준다.

상용 실시간 운영체제는 일반적으로 디버거를 포함한 개발 환경과 이식성에 있어서 우수하다. JPL이 사용한 VxWoks의 경우, Pathfinder 개발에 사용된 AIX, SPARC, 68K-based VME board, RAD6000, PowerPC에 대해 이식성이 우수했다고 알려졌다. 그러나 기존에 사용하던 실시간 운영체제를 상용 실시간 운영체제로 변경할 때, 프로그래머가 상용 실시간 운영체제에 익숙하지 않으면 디자인, 사양 등이 필요 이상으로 설정될 수 있고 나아가 시스템 설계에 영향을 주게 된다.

위에서 언급한 모든 시스템에 대해서 각각의 실시간 운영체제가 어떻게 사용되었는지를 살펴보는 것도 중요한 의미를 가지지만, 어떤 개발 환경이나 운영체제가 사용되었는지 실제 경우에 대해서 자료를 구하기가 매우 어렵다. 그리므로 본 논문에서는 국내에서 개발한 경험이 있는 국내 우주비행체 위주로 기술한다.

3.1 실시간 운영체제의 예 : 다목적실용위성[5,6]

본 절에서는 다목적실용위성의 실시간 운영체제인 VRTX가 다목적실용위성의 Heritage 위성인 TOMS-EP에 어떻게 사용되었는 지에 관해 기술한다. 여기에 제시된 정보는 TOMS-EP의 자료를 그대로 인용하지 않고 VRTX 운영체제와 각 태스크간의 관계를 재구성한 것이다. TOMS-EP에서 사용된 VRTX를 이용한 설계 개념은 다음과 같다.

- TOMS-EP의 탑재 소프트웨어는 실시간, 멀티태스킹 환경에서 수행한다.
- VRTX를 이용하여 태스크 스케쥴링과 태스크 스위칭을 수행한다.
- 높은 우선 순위의 태스크에 의해 인터럽트 되지 않으면 해당 사이클에서 모든 태스크를 수행하고 다음 이벤트를 기다린다.
- VRTX의 메시지 큐(Message Queue), VRTX 이벤트 플랙, 글로벌 플랙을 이용하여 태스크 간의 메시지를 전송한다.
- 태스크가 처리할 내용이 있음을 알리기 위해 이벤트 플랙을 사용한다.
- 현재 수행 중인 태스크가 VRTX 이벤트에 pending되어 있으면 다음 우선 순위를 가지는 태스크로 태스크 스위칭을 한다.
- 인터럽트가 발생하면 인터럽트 서비스 루틴에서 태스크나 함수를 수행하지 않고, 인터럽트 서비스 루틴은 이벤트 플랙만 발생시킨다. VRTX는 발생한 이벤트 플랙에 대해 해당 태스크를 수행한다.

이렇게 함으로써 모든 태스크가 VRTX에 의해 제어를 받게 된다.

그림 3은 TOMS-EP의 상위 레벨 Context Diagram을 나타낸 것이다. 그림에서 각 태스크는 VRTX에 의해 초기에 생성되고 이벤트 플랙에 따라서 VRTX는 각 태스크를 수행한다. 동일한 VRTX를 사용하고 있는 다목적실용위성의 경우 기본적으로 TOMS-EP와 크게 다르지는 않지만, 태스크간의 복잡성을 줄이고, deterministic한 특성을 보장하기 위해 TOMS-EP에 비해서 태스크 수가 적고, 태스크간의 이벤트 플랙에 의한 메시지 전달이 적은 편이다. 실제 태스크를 생성시키고 운영하는 개념은 실시간 운영체제에 영향을 받는 것이 아니므로 각

RTOS에 대해 비슷하게 적용할 수 있다. 중요한 것은 실시간 운영체제의 특성을 잘 파악하고 설계 개념을 적용하는 것이다.

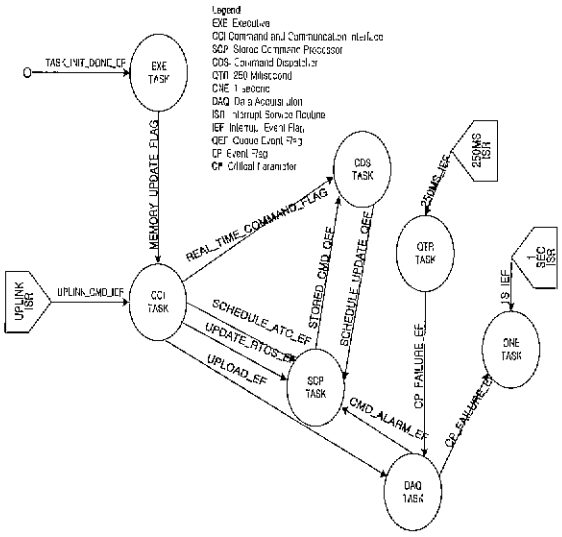

그림 3. TOMS-EP Context Diagram 예.

3.2 실시간 운영체제의 예 : 무궁화위성 3호[7]

무궁화위성 3호는 Lockheed martin사의 A2100 bus의 모델을 근간으로 A2100의 버스를 일부 변경하므로써 사용자의 요구사항에 맞추도록 설계 제작된 인공위성이다. A2100 버스 탑재컴퓨터의 특징은 하드웨어구조는 분산형구조를 취하면서 데이터 처리는 중앙형 구조를 가진다는 것이다. 하나의 탑재컴퓨터가 분산된 각 서브시스템으로부터 필요한 데이터 송수신을 요구하게 되고, 4개의 원격장치(RIU: Remote Interface Unit)는 OBC(On Board Computer)의 요구에 따라 해당 텔레메트리 데이터를 송신하거나 원격명령을 수신하여 명령을 수행한다. OBC와 4개의 RIU간 통신은 MIL-STD-1553B 데이터 버스를 통하여 이루어진다. 탑재컴퓨터를 제외한 RIU의 로직들은 대부분 FPGA에 의해 구동된다. 무궁화 3호위성 데이터 처리부는 MIL-STD-1553B 데이터 버스를 이용한 분산형 구조라는 점에서는 다목적실용위성 1호의 데이터 처리부와 유사하나, 데이터 처리 및 운용면에서는 탑재컴퓨터만 위성 버스를 관리한다는 점에서 전력계, 원격측정명령계, 자세제어계의 탑재컴퓨터를 운용하는 다목적실용위성과는 다르다고 할 수 있다.

무궁화 3호 탑재컴퓨터의 기본모델인 Lockheed Martin사의 A2100 탑재컴퓨터는 Honeywell 사가 개발한 탑재컴퓨터를 사용한다. 이 탑재 컴퓨터는 Honeywell 사의 CPU 설계 기술로 제작되었으며 MIL-STD-1750A의 명령 셋(Instruction Set)을 만족한다. Honewell 사의 탑재컴퓨터에 주로 사용되는 언어는 Ada이다. 무궁화 3호의 탑재컴퓨터도 Ada 언어와 MIL-STD-1750A 탑재컴퓨터가 사용되었다. Ada 경우에는 실시간 운영체제를 사용하지 않고, 반면 고유의 실시간 처리 커널(Run Time Kernel)을 이용한 비행소프트웨어(FSW: Flight Software)를 제작할 수 있다.

Ada 언어는 1980년대부터 미국방성에서 mission-critical 분야에 주로 사용하기 위해 개발한 언어로써 특히 항공우주분야에 많이 사용되고 있다. 무궁화위성 3호 탑재소프트웨어는 Ada run-time kernel을 사용하여 설계되었고, 별도의 실시간 운영체제는 사용되지 않았던 알려져 있다. 본 논문에서는 Ada run-time kernel에 관한 설명보다는 무궁화위성 3호 소프트웨어의 계층적 구조에 관하여 언급하고자 한다. Ada run-time kernel에 관한 내용은 관련 서적을 참조하길 바란다.

무궁화 3호 비행소프트웨어의 CSCI(Computer Software Configuration Items)는 Honeywell의 시작 롬코드(Start-Up ROM code), 부트스트랩 로더(Bootstrap Loader) 그리고 응용비행소프트웨어(Application Flight Software)로 나누어진다. 시작 롬코드는 전원인가 리셋시(power on reset) 프로세서와 메모리를 테스트하며 부트스트랩 로더는 시작 롬코드가 종료되면 PROM 데이터들을 RAM 영역에 복사한 후 RAM 로딩이 종료되면 FSW(Flight SoftWare)를 수행하도록 한다.

참고로 무궁화 3호 탑재소프트웨어를 개발에 사용된 Ada 개발 소프트웨어로는 TARTAN사의 1750-A Ada X-complier(SPARC host)와 VERDIX사의 Sun Ada Compiler가 사용되었다.

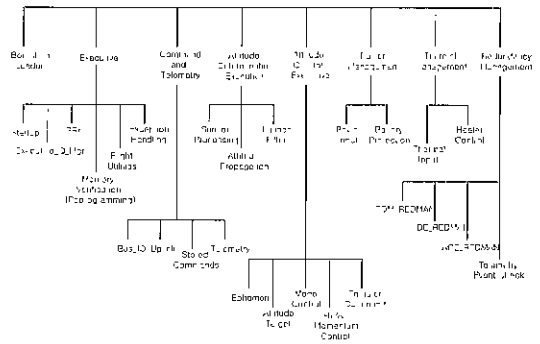

그림 4 무궁화 3호 탑재소프트웨어의 계층적 구조.

무궁화 3호 탑재소프트웨어는 그림에서와 같이 크게 7개의 태스크와 1개의 부트스트랩 로더로 구성되어있다. 7개의 태스크는 응용 탑재소프트웨어를 구성하는 요소이다. 원격명령 및 텔레메트리 태스크는 원격명령 수신, 저장된 명령을 수행하고 텔레메트리를 전송하는 기능을 한다. 센서처리, 칼만필터, 자세 전파등의 기능을 수행하는 자세결정 executive와 자세제어를 수행하는 자세 제어 executive가 있다. 전력과 열을 제어하는 전력관리와 열관리 프로그램이 있다. 그리고 결함 발생시 결

함을 극복하기 위한 잉여관리 프로그램(RedMan Redundancy Management)이 탑재되어있다.

특히 Executive 는 항상 16Hz로 수행되는 모든 동기테스크 또는 비동기 태스크를 활성화시킨다. 무궁화위성 3호 경우 Executive테스크는 텔레메트리는 2Hz, 상향(비동기) 2Hz, 자세결정 executive 8Hz, 자세제어 executive 2Hz, 저장된 명령 1Hz, 전력 및 열 제어 0.125Hz로 활성화시킨다.

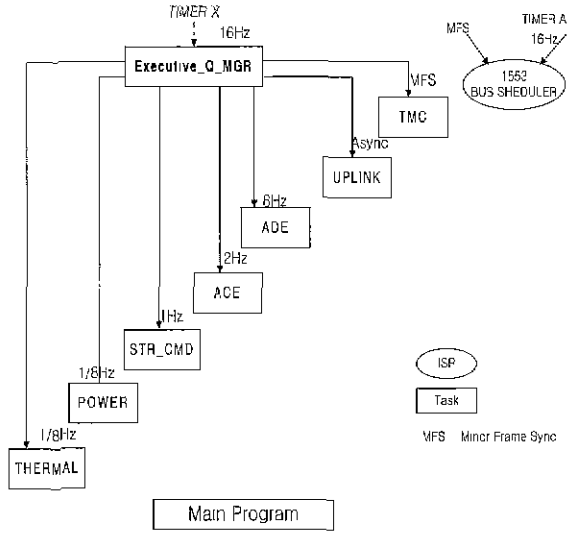

그림 5. 무궁화위성 3호 Executive 동작.

그림 6은 무궁화위성 3호의 데이터 흐름을 보여준다. 그림에서 탑재컴퓨터 1750A가 탑재소프트웨어의 기능을 수행하고 있는 과정에서 BCRTM(Bus Control Remote Terminal Monitor)로부터 1553B 데이터 수신에 인터럽터를 받고 CTU(Command Telemetry Unit)로부터 텔레메트리 프레임 전송 인터럽터도 받으며 WDT(Watch Dog Timer)에게 MeOK 신호를 계속적으로 출력하도록 구성되어있다. 그리고 BRIU (Bus RIU)를 비롯한 다른 서브시스템들은 각 서브시스템마다의 데이터를 1553B 데이터 버스를 통해 송수신 기능을 수행하도록 구성되었다.

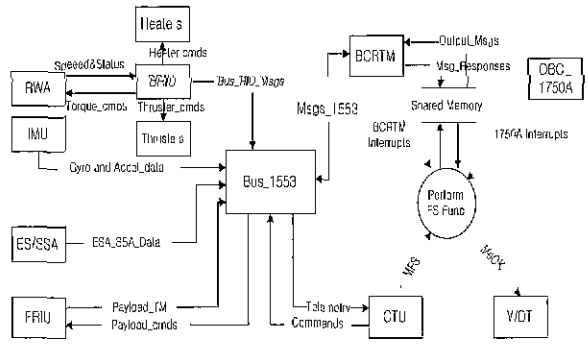

그림 6. 무궁화위성 3호의 데이터 흐름도.

3.3 실시간 운영체제 예: 소형위성[8-10]

소형위성도 중,대형 위성과 마찬가지로 거의 모든 위성이 탑재컴퓨터를 이용한 자세제어 및 서브시스템 관리를 수행한다. 소형위성 중 영국 써리 대학의 UoSat에 기초한 대부분 시스템들이 BekTek사의 SCOS(Spacecraft Operating System)를 이용하여 설계되었다. 이 SCOS는 실시간 다중태스킹 커널, 메시지 전달 기능, AX.25 프로토콜 드라이버, DMA/인터럽터에 기초한 I/O 드라이버 셋등을 제공한다. BekTek SCOS는 강력한 API(Application Program Interface)를 제공하고 다중 프로세서(Multi-Processor) 응용이 용이한 특징을 가지고 있다. 또한 모든 응용 프로그램은 I/O 스트림에 통하여 상호 연결되어 있다.

그림7은 미국 아리조나 주립대의 소형위성인 ASUSAT(Arizona State University Satellite) 탑재소프트웨어의 구성을 보여준다. ASUSAT의 탑재소프트웨어는 SCOS 운영체제를 이용하여 설계되었는데 Executive 프로그램이 위성운영 및 관리에 필요한 6개의 태스크를 관리하면서 위성 소프트웨어를 운영하고 나타내주고 있다.

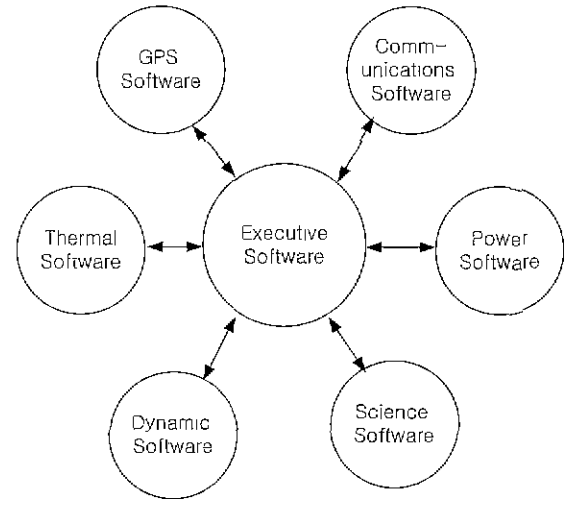

그림 7. ASUSAT의 서브시스템 태스크.

우리별 1,2호의 운영체제는 실시간, 선점형, 다중 운영체제 Nucleus에 바탕을 두고 개발되었으며, 실시간 다중태스크 관리 운영체제를 사용하여 태스크 스케줄링, 태스크간 통신, 시스템 자원할당을 지원한다. 우리별 1,2호의 실시간 운영체제도 BekTek사의 SCOS가 사용되었다. 한국과학기술원 인공위성센터의 발표 논문에 따르면 소형 인공위성 주 제어 컴퓨터의 운영체제는 기본적으로 다음과 같은 기능이 요구된다고 명시하고 있다.

- 실시간 선점형 다중 태스크 스케줄링 기능
- 각 테스크의 실행 주기가 프로그램 가능하고 개별적 on-off가능
- 컴퓨터 상태 메시지 제공

- 암호화된 명령을 통한 태스크 로딩 기능
- 동적인 태스크의 생성 및 관리
- 동적인 커널의 확장

우리별 3호의 탑재소프트웨어는 KASCOM 혹은 OBC186에 동작하도록 구성되었다. 우리별 3호의 소프트웨어는 우리별 1,2호에서 사용된 탑재 데이터 처리 시스템(OBDH: On Board Data Handling System)의 CPU가 80186에서 80960으로 전환됨으로써 운영체계의 변화가 있었다. 우리별 3호의 실시간 운영체계인 Star-keeper는 실시간 다중태스킹의 능력을 가지고 있는데 이 운영체계의 특징은 다음과 같이 요약된다.

- Priority based preemptive micro kernel
- Based on Nucleus RTX(a subset of VRTX)
- inter-task communication using stream
- window based development environment(XRAY, 960 simulator)
- full system support including fixed and variable memory management, clock management, queue event management
- Async- and synchronous I/O handler

다음 그림은 KITSAT-3의 탑재소프트웨어 구조를 보여준다.

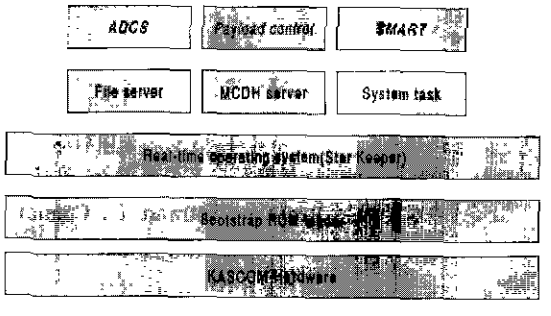

그림 8. KITSAT-3의 탑재소프트웨어 구조.

3.4 인공위성 발사체 분야 실시간 처리 시스템[11,12]

인공위성 분야의 기술에 관한 폐쇄성은 인공위성 발사체 분야에서도 적용되는데 인공위성 발사체의 소프트웨어 구성에 관한 자료 확보 또한 거의 불가능하다. 그러나 발사체의 기능, 동작, 데이터 처리량을 유추하여 탑재컴퓨터의 처리능력 및 소프트웨어의 구조를 개략적으로 해석할 수 있을 뿐이다. 본 절에서는 인공위성 발사체 실시간운영체제 및 탑재소프트웨어 구성의 예를 보였으며 한국형 인공위성 발사체의 탑재시스템에 관하여 언급한다.

호주의 ASRI(Australian Space Research Institute Ltd.)에서 인공위성 발사체를 개발하기 위해 제작중인 AUSROC은 다중태스킹 실시간 운영체계인 JAMOSX을 사용한다. JAMOSX 실시간 운영체제는 레이어 시스템(layered system)으로 사용자 레이어, 커널 레이어 그리고 하드웨어 레이어의 3개 레이어로 구성되어 있다. 운영체제의 각 레이어 관계는 다음 그림 9와 같다. 사용자 레이어는 여러 태스크를 수행하기 위한 사용자의 프로그램이 내장된다. 커널 레이어는 스케쥴링, 태스크스위칭등 기본기능을 수행하는 부분과 태스크를 생성, 삭제 및 지연시키는 확장된 커널기능으로 구성된다. 하드웨어 레이어는 CPU를 비롯한 디바이스 구동 및 인터럽터 처리 루틴등으로 구성되었다.

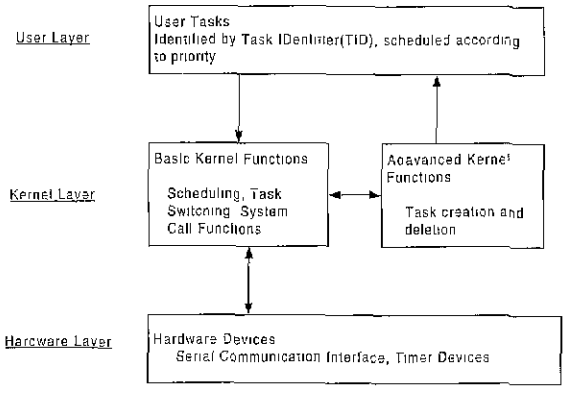

그림 9. AUSROC 실시간 운영체제(JAMOSX) 레이어.

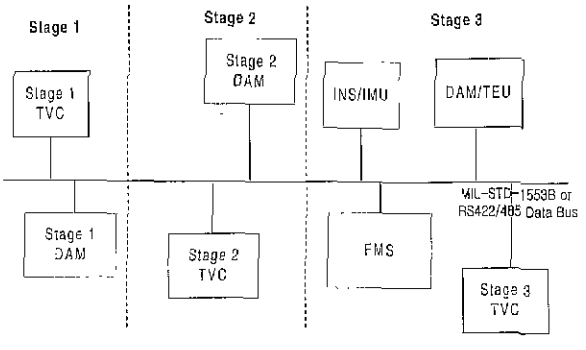

그림 10. 한국형 인공위성 발사체 탑재부 구성.

한국항공우주연구소에서 현재 개발 계획중인 인공위성발사체에도 고성능 상용 실시간 운영체제 탑재의 필요성이 대두되고 있다. 인공위성 발사체의 비행제어 비스템의 구성요소로는 일반적으로 비행관리시스템(FMS: Flight Management System), 관성항법장치(INS: Inertial Navigation System), 추력벡터제어기(TVC: Thruster Vector Controller), 비행종단시스템(FTS: Flight Termination System), 데이터획득모듈(DAM: Data Acquisition Module) 그리고 텔레메트리모듈(TEU: Telemetry Encoder Module)등이 사용된다. 또한 인공위성 발사체의 데이터 버스로는 MIL-STD-1553B뿐만 아니라 RS-422/485 데이터 버스등이 널리 사용된다.

다음 그림10은 향후 국내 인공위성 발사체에 사용 가능한 실시간 시스템의 구성 기본 요소들을 보여준다. 국내 발사체 개발은 아직 기초 단계이지만 비행관리 시스

템의 설계와 발사체내의 고속 텔레메트리 시스템에 관한 기본 설계는 과학로켓-3(KSR-3)의 과제 수행을 통해 일부 수행되었다. 특히 FMS는 발사체의 전반적인 관리를 위해 고성능 및 고신뢰도를 갖는 탑재컴퓨터을 이용하며 상용 운영체제를 이용하여 제작될 계획이다.

4. 결론

인공위성 및 발사체등의 우주비행체가 임무를 수행하기 위해서는 상당히 높은 신뢰도를 가지도록 구성되어야 한다. 특히 탑재컴퓨터와 탑재소프트웨어는 인공위성 또는 발사체를 운용하는데 핵심적인 기능을 수행하므로 실시간 데이터 처리가 가능해야한다. 실시간 데이터 처리를 수행하기 위해 소형 커널을 이용한 탑재소프트웨어를 제작 할 수 있지만, 그러나 대부분 실시간 시스템에서는 실시간 운용체제를 사용한다. 본 논문에서는 국내 위성 개발 경험을 바탕으로 다목적실용위성, 무궁화위성 3호 그리고 우리별 3호의 실시간 운영체제에 관하여 설명하였다. 또한 국내외 자료확보가 용이한 소형위성 및 발사체의 실시간 데이터 처리용 탑재소프트웨어 구조를 살펴봄으로써 우주 산업에 사용되는 실시간 시스템 및 운영체제의 동향을 살펴보았다.

참고 문헌

[1] "What makes a good RTOS", *Real Time Magazine*, Feb. 28 ,2000

[2] "Real-Time Operating Systems Market Review", *Real Time Magazine*, Fourth Quater Edition, 1995

[3] "Buyer's Guide 2000", *Embedded Systems*, 2000

[4] "Survey of Real-Time Operating Systems", Georgia Institute of Technology, Feb. 15, 1994

[5] TOMS-EP Flight Software Critical Design Audit Package, July, 1992.

[6] KOMPSAT Flight Software Critical Design Audit Package, March, 1997.

[7] 최성봉 외, *정지궤도 통신위성 핵심 서브시스템 및 운용시스템 개발*, 한국항공우주연구소, 1998

[8] http://www.eas.asu.edu/~nasasg/asusat/SUBSYSTEMS/softmain.html

[9] 김형신 외, "우리별 1,2호 주 컴퓨터부", *한국우주과학회지*, 제13권 제2호 특집호, pp. s41-s51, 1996.

[10] 김형신, "Introduction to KITSAT Flight Software", 한국항공우주연구소 세미나자료, 인공위성센터, 2000.

[11] Luke Francis Kearney, *A Real-Time Multi-Tasking Operating System For A Remotely Programmable Micro-Controller(JAMOSX)*, Queensland University of Technology, 1988

[12] 채연석 외, *최종연구보고서 3단형 과학로켓 개발사업(II)*, 한국항공우주연구소, 1999

김주년

1992년 경북대학교 전자공학과 졸업. 1992년 Applied Materials Korea 근무. 1995년 경북대학교 전자공학과 석사. 1995년~현재 한국항공우주연구소 로켓탑재기기연구그룹. 관심분야는 Digital Avionic System on Spaceborne Vehicles

천이진

1993년 경북대학교 전자공학과 졸업. 동대학원 석사(1995년). 1995년~현재 한국항공우주연구소 위성전자연구그룹. 관심분야는 Real-time Software for Control 및 Variable Structure Control of Spacecraft Attitude Tracking.