위성 탑재 소프트웨어 기술 동향
A Survey of Satellite Flight Software Technology

이 종 인, 백 홍 열
한국항공우주연구소

1. 서론

근래의 인공위성은 정지궤도, 저궤도, 또는 행성 탐사위성을 막론하고 주어진 임무를 수행하기 위해 컴퓨터를 (on-board computer) 탑재하고 있다. 초기의 인공위성은 비교적 단순한 임무를 수행하였으나 임무가 복잡해 지고 우주궤도에서 지상과의 통신 범위 밖에서도 임무를 수행할 필요성이 대두됨에 따라 탑재컴퓨터는 위성의 필수적인 장치가 되었으며, 이에 따라 위성 탑재컴퓨터에서 수행되는 소프트웨어에 (flight software) 대한 연구도 활발해지고 있다. 본 논문에서는 위성 소프트웨어의 역할, 구성, 개발환경을 살펴보고, 연구 동향에 대하여 살펴본다.

2. 위성 탑재 컴퓨터 시스템

초기의 인공위성은 탑재 컴퓨터 없이 대부분의 기능을 hardwired logic 으로 구현하였다. 이는 요구되는 위성의 임무가 비교적 간단하였으며, 당시에는 우주환경에 견딜 수 있는 고 신뢰도의 탑재 컴퓨터가 없었기 때문이었다. 그러나 위성의 임무가 다양해짐에 따라 임무마다 새로운 hardwired logic 을 구현하는 것은 비용이 많이 들며, 하드웨어가 복잡하게 되었다. 따라서 종래의 복잡한 hardwired logic 을 대체하고 다양한 임무에 대응하기 위하여 컴퓨터가 탑재된 위성의 개발이 시도되었다. 이는 경량의 우주용 고 신뢰도 마이크로프로세서의 개발에 힘입은 바 컸다.

초기에는 1802 와 같은 8-bit 마이크로프로세서가 주로 사용되었으나 근래에는 8086/80186, Mil-Std-1750A 와 같은 16-bit 마이크로프로세서가 널리 사용되고 있으며, 80386/80486, R3000, RAD6000 와 같은 32-bit 마이크로프로세서도 많이 사용되고 있다. 또한 최근에 와서는 64-bit 마이크로프로세서의 사용이 고려되고 있는 추세이다. 그림 1 은 전형적인 위성 탑재컴퓨터 시스템 구조를 나타내며, 표 1 은 주요 위성의 탑재컴퓨터에 사용된 마이크로프로세서의 예를 나타낸다.

위성 탑재컴퓨터 시스템은 크게 분산구조와 중앙집중시스템 구조로 구분할 수 있다. 위성 임무에 따라 다르나 소규모의 위성은 중앙집중시스템의

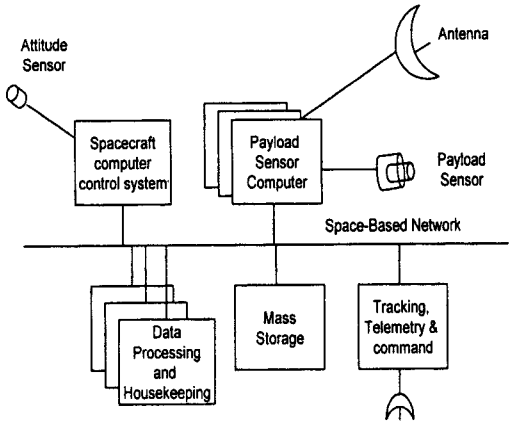

그림 1. 전형적인 위성탑재 컴퓨터 시스템 구조.

표 1. 탑재컴퓨터에 사용된 마이크로프로세서 예.

Spacecraft	OBC Processor	사용언어	임무
우리별 1,2,3 호	80C186, 80960	C	과학 실험, 지구관측
무궁화 3 호	1750A	Ada	통신, 방송
다목적실용위성	80C186	C	과학 실험, 지구 관측
SSTI	R3000	Ada	지구 관측
Landsat-7	1750A	Jovial	지구 관측
Galileo	1802	Assembly, HALS	Interplanetary
Hubble Space Telescope	DF-224, NSSC-1, 80386/387, (80486 으로 upgrade 예정)	Assembly, C	Astronomy
Deep Space-1	Rad6000	C, Lisp	Interplanetary
Cassini	1750A	Ada	Interplanetary
Space Shuttle	IBM AP-101 (GCP)	Ada	Space Transportation

구조를 주로 채택하나 대형위성으로 갈수록 분산시스템 구조를 갖는다. 분산구조 시스템은 주로 버스 구조를 사용하는데 Mil-Std-1553B / Mil-Std-1773 데이터 버스가 그 대표적인 예이다.

3. 위성 탑재소프트웨어의 역할 및 구성

위성탑재 소프트웨어는 종래의 hardwired logic 에서 수행되던 위성의 기능을 탑재컴퓨터에서 수행하는 것으로 일반적으로 다음의 역할을 수행한다.

- 위성의 상태 정보의 수집, 저장 및 전송 (state-of-health telemetry gathering, formatting, storing, and down-link)
- 지상 명령의 수신 및 처리 (command reception, verification, and processing)
- 위성의 궤도 및 자세제어 (orbit and attitude control)
- 위성의 전력시스템 제어 (electrical power control)
- 위성의 열 제어시스템 제어 (thermal control)
- 위성 탑재체의 관리 (instrument payloads management)
- 위성 시스템의 장애 관리 (fault management)

탑재소프트웨어는 크게 시스템 소프트웨어와 (system software) 응용 소프트웨어로 (application software) 구분할 수 있다. 시스템 소프트웨어는 탑재컴퓨터에서 위성운영 소프트웨어가 실행될 수 있는 기본 환경을 제공한다. Real-time multi-tasking operating system, executive, I/O driver, communication handler 나 각종 utility 등이 이에 속한다.

응용 소프트웨어는 위성 고유의 임무를 수행하기 위한 소프트웨어로 위성 운영에 필요한 house-keeping function (명령 수신 및 처리, 궤도 및 자세제어, 전력계 제어, 열 제어)이나 mission operation (탑재체 운영)을 수행하는 소프트웨어를 말한다. 그림 2 는 전형적인 위성탑재소프트웨어 구성을 나타낸다.

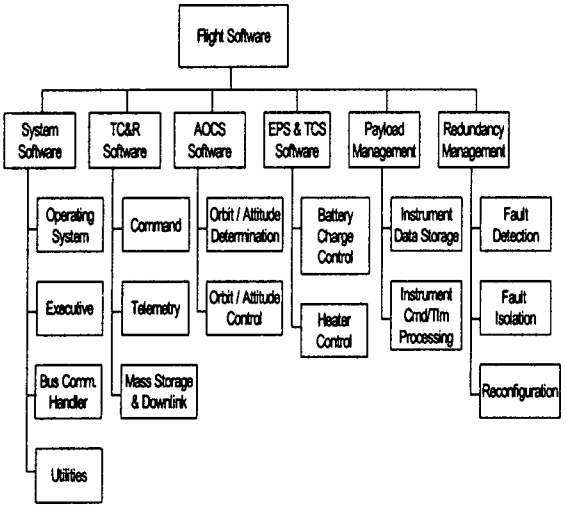

그림 2. 전형적인 위성탑재 소프트웨어 구성.

3.1 시스템 소프트웨어(System software)

시스템 소프트웨어는 kernel 또는 real-time multi-tasking operating system (OS)을 토대로 실행관리자 (executive), 위성체 내부의 데이터버스를 (예: 1553B data bus) 통한 통신 처리부, 각종 유틸리티 소프트웨어로 구성된다. 실시간 운영체제는 task scheduling, interrupt processing, inter-task communication, timer handling, 등의 기능을 지원하며 간단한 cyclic scheduler 를 자체 개발하여 사용하는 경우도 있으나 VRTX, VxWorks, 등의 Commercial-Off-The-Shelf (COTS) 제품을 사용하기도 한다. Executive 는 system startup / boot-loader 와 내부 버스를 통한 모듈간의 통신을 포함하며 응용 소프트웨어의 실행을 관리한다. 각 응용 소프트웨어의 실행은 실행 주기 (execution cycle)가 미리 정해진 static/deterministic scheduling 방식이 주로 이용되며 대부분의 task 는 periodic 하다. 또 궤도상의 위성 탑재 소프트웨어를 수정, 재구성하기 위한 코드/데이터 로드 처리 기능도 갖는다. 그림 3 는 시스템 소프트웨어와 응용 소프트웨어의 계층적 구조를 나타낸다.

3.2 원격측정명령 소프트웨어(TC & software)

위성과 지상간의 통신과 위성내부의 데이터를 처리한다. 수신된 명령을 검증한 후 처리하고, telemetry 를 수집, protocol 에 따라 포맷하여 mass storage 에 저장하고, 지상으로 전송한다. 저장명령은 탑재 컴퓨터내의 command storage 영역에서 관리되고 scheduling 된다. 원격검침 데이터를 (telemetry data) mass storage 에 저장하고, 지상명령에 의해 retrieve 하며 선택된 전송 모드에 따라 지상으로 전송한다. 통신 및 명령 처리 (communication and command interface), 명령 실행 (command dispatch library), 데이터 수집(data acquisition), 저장명령 처리 (stored command processing), 대용량 저장장치 및 전송 (mass storage management & downlink) 등으로

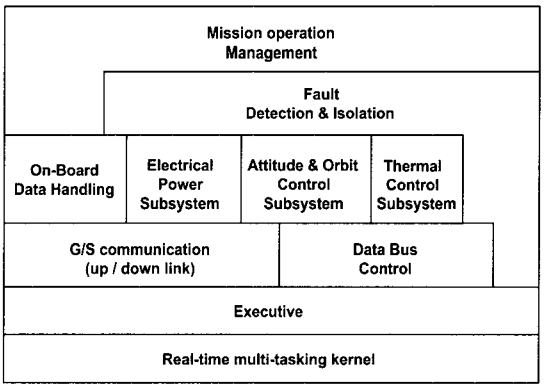

그림 3. 위성 탑재소프트웨어의 계층적 구조.

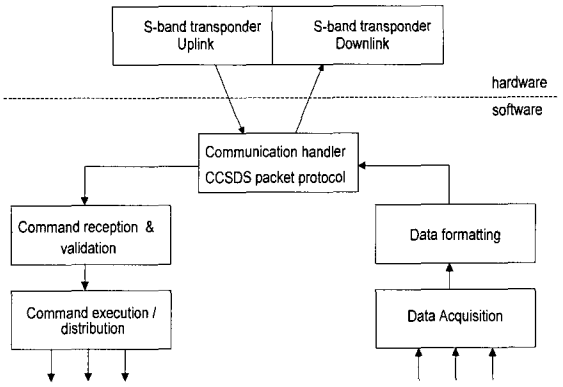

그림 4. 원격측정명령 소프트웨어 구조.

구성된다. 그림 4는 일반적인 원격측정명령 소프트웨어의 구조를 보여 준다.

위성 상태정보의 수집, 저장 및 전송

위성의 각 서브시스템 및 탑재체로부터의 State-Of-Health를 (SOH) 수집하여 포맷한 후 mass storage에 저장하여 명령에 따라 지상으로 전송한다. 지상으로의 데이터 전송방식으로 Time Division Multiplexing (TDM) 방식과 Consultative Committee for Space Data Systems (CCSDS)의 packet telemetry standard가 많이 사용되고 있다.

지상명령의 수신 및 처리

지상에서 수신된 실시간 또는 저장명령 (real-time/stored command)을 수신하여 검증하고 이를 수행하거나 하드웨어로 전달하는 기능이다. 저장명령은 보통 절대시간명령과 상대시간 명령군으로 (absolute time command / relative-timed command sequence) 구분되며 탑재컴퓨터의 메모리에 저장된 후 지정된 시간에 수행되거나 (절대시간 명령), 지상명령 또는 탑재소프트웨어에 의해 기동 되어 실행된다 (상대시간 명령군). PROM에 저장된 저장명령군은 위성 발사 후 태양전지판의 자동 전개나 장애 감지 후 안전모드로의 자동 전환에 이용되는 경우도 있다. 지상의 명령은 보통 암호화 (encryption) 되어 제3자에 의한 침해를 방지하게 되며 전송방식으로 일부 위성에 CCSDS packet standard가 사용되고 있다.

3.3 위성의 궤도 및 자세제어(AOCS software)

자세제어 센서로부터의 (gyro, earth sensor, sun sensor, star tracker, GPS receiver, 등) 데이터를 처리하여 위성의 궤도나 자세를 결정하고 (orbit determination navigation /attitude determination) 구동 장치를 (thruster, reaction wheel, magnetic torquer, 등) 사용하여 궤도나 자세를 제어한다 (orbit/attitude control). Kalman Filter logic과 PID controller가 주로 사용된다.

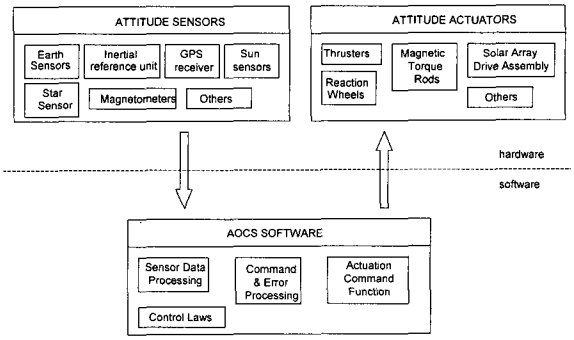

그림 5. 궤도 및 자세제어 소프트웨어 구조.

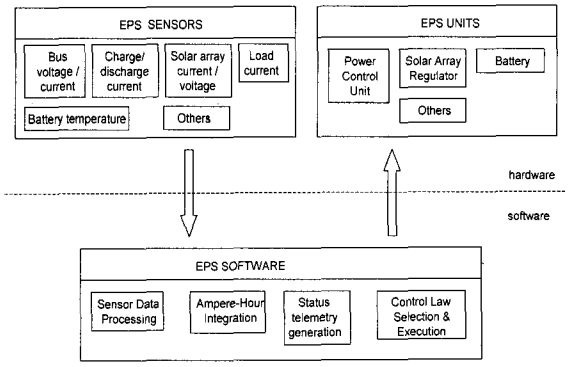

그림 6. 전력계 소프트웨어 구조.

지구저궤도위성의 conste-llation maintenance와 지구정지궤도위성의 stationkeeping도 수행한다. 센서 데이터 처리부, command & error processing부 control law 처리부, actuator command generation부 mode transition logic 및 기타 solar array deployment test logic 등으로 구성된다.

3.4 위성의 전력시스템 제어(EPS software)

축전지의 상태를 모니터하여 충.방전을 제어하며 전력계 센서로부터의 데이터를 (전류, 전압, 온도, 압력, 등) 처리하여 축전지의 충전/방전을 제어한다. Ampere-hour integration에 의한 충전/방전 감시, Peak Power Tracking (PPT), Direct Energy Transfer (DET), 또는 Temperature Compensated Voltage Limit (TCVL), 등에 의한 충전 제어를 수행하여 축전지를 과 방전 (over discharge), 과열로부터 (over-temperature) 보호한다. 비상시에는 load shedding을 수행한다.

3.5 위성의 열 제어(TCS software)

위성 주요부분의 온도를 온도센서로부터 측정하여 히터를 제어함으로써 위성운영에 필요한 적정한 온도를 유지한다.

3.6 탑재체의 관리(Instruments Management)

위성임무를 위한 탑재체 (instruments)의 운영을 관리한다. 탑재체의 SOH telemetry 및 관측 데이터를

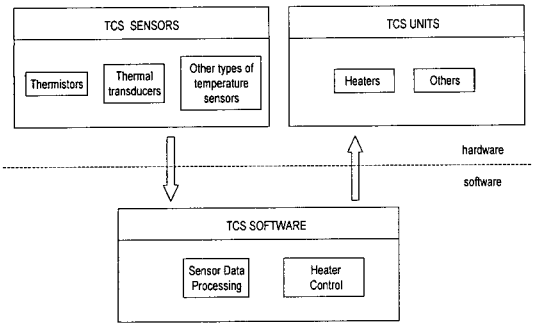

그림 7. 열 제어 소프트웨어 구조.

수집하여 mass storage 에 압축, 포맷, 저장하고 지상 명령에 따라 전송한다. 지상명령 (실시간 또는 저장 명령) 이나 보조 정보를 (ancillary data) 탑재체로 전달한다. 탑재체의 관측데이터는 TT&C 와 독립적인 통신 경로를 통하여 지상으로 전송하는 경우가 많다. 또 비상시에는 탑재체를 안전 모드로 전환한다.

3.7 시스템 장애 관리(Fault management)

Redundancy management 라고도 한다. 위성의 탑재 컴퓨터를 비롯하여 주요 장치는 single-point-failure 를 방지하기위해 redundancy 를 지니도록 설계된다. 위성의 상태를 감시하고 장애의 발견 시에(fault detection) 정해진 논리에 따라 장애발생 장치의 전원을 차단하고 redundant side 로 교체한다(fault isolation, system reconfiguration). 일반적으로 장애 탐지 후 위성을 안전한 상태로 (safe-hold mode) 두고 지상의 명령을 기다리게 되나 (fail-safe), 행성 탐사 위성과 같이 위성의 autonomy 가 중요한 경우는 자동 복구 후 중단된 수행을 계속하는 경우도 있다 (fail-operational).

4. 탑재소프트웨어의 개발

4.1 탑재 소프트웨어 개발 환경

위성탑재 소프트웨어는 real-time embedded system 소프트웨어로 고 신뢰성이 요구되므로 이의 개발 환경은 매우 중요하다. 여기서는 위성 탑재 소프트웨어의 개발에 필요한 환경을 살펴본다.

개발 언어

Ada, C, C++ 언어가 주로 사용되며 time-critical application 에서는 assembly language 도 사용된다. 인증된 컴파일러와 build, test tools 의 선정도 중요하다.

개발 방법론 및 지원 도구

소프트웨어 개발 프로세스로는 Dod-Std-2167A 와 같은 waterfall development model 과 evolutionary development model, spiral development model, 등이 있으며 프로젝트 특성에 따라 적합한 모델을 선정하게 되나, 주로 병행하여 사용된다. 근래에는 객체지

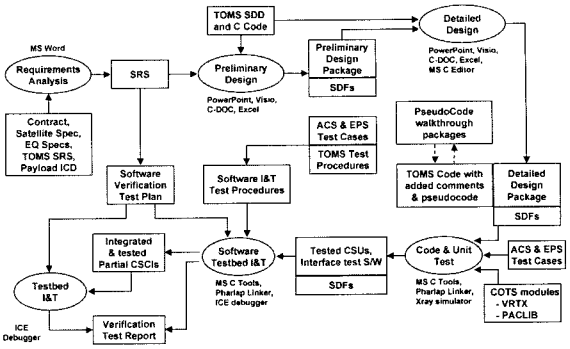

그림 8. 탑재소프트웨어 개발과정 예. (다목적 실용위성)

향 소프트웨어 개발 방식이 (Object-Oriented software Development) 사용되고 있는데, HOOD (Hierarchical Object-Oriented Development) 방식은 유럽에서 (ESA) 많이 사용되고 있다. 소프트웨어 개발을 지원하는 CASE (Computer-Aided Software Engineering) tools 을 사용하는 경우도 많으나 개발자들이 그 도구에 익숙해 지는데 시간이 필요하다.

시험 도구

위성소프트웨어는 고 신뢰도를 요구하므로 철저한 시험과 검증 (verification and validation)이 요구된다. 소프트웨어의 검증을 위해 software testbed 나 simulator, test set 를 제작하여 탑재소프트웨어를 시험하며, 시스템 통합 시험을 위해서는 시험 지원 소프트웨어 (ground support software)가 필요하다.

4.2 탑재 소프트웨어 개발 과정

보통 위성 소프트웨어 개발에 적용되는 표준으로 Dod-Std-2167A 와 같은 개발 표준에 따른다. 일반적으로 위성 탑재소프트웨어는 요구사항 분석 및 사양서 작성 (software requirements analysis and specification), 예비설계 (preliminary design), 상세 설계 (detailed design), 구현 및 유닛 테스트 (implementation and unit test), 통합 시험 (integration test), 검증 시험 (verification test & validation), 운영 및 유지 (operation and maintenance)의 과정을 거쳐 개발된다. 그러나 개발 과정은 프로그램 고유의 특성에 맞도록 적절히 수정되어 적용된다. 그림 8 은 위성탑재 소프트웨어 개발의 한 예를 보여준다.

5. 탑재소프트웨어 연구 동향

Radiation-hardened 탑재 컴퓨터의 성능 향상은 종전의 극히 제한된 메모리 및 CPU throughput 의 제약으로부터 벗어나 탑재소프트웨어 기술이 향상될 수 있는 계기가 되었다. 위성 탑재소프트웨어의 최근 연구동향은 탑재 소프트웨어 신뢰도 향상, 개발 비

용 절감 및 생산성 향상, autonomy 의 증가 등에 초점이 맞추어진다.

신뢰도 향상

탑재소프트웨어 신뢰도 향상을 위하여 NASA 에서는 Independent Verification and Validation (IV&V) 조직을 두고 연구를 진행하고 있다. Software Quality Assurance 분야의 연구도 진행되고 있다. 특히 Fault Detection and Isolation, Recovery (FDIR) 분야는 formal method 를 사용하여 시스템 요구사항의 specification, completeness, correctness 를 분석하고 검증하는 추세이다.

개발 비용의 절감 및 생산성 향상

Multi-mission satellite bus 개념에 따라 공통의 탑재소프트웨어를 개발하여 여러 가지 임무에 사용하려는 연구가 진행 중이다. 하드웨어 인터페이스 및 소프트웨어의 표준화를 달성하여 다양한 임무에 적용하고 소프트웨어의 재사용을 추구한다 (예: 1553/1773 data bus, CCSDS standard). CASE tools 을 사용하여 위성 탑재소프트웨어 시험 및 평가 과정을 자동화하는 연구나, 새로운 소프트웨어 개발 방법론의 적용도 시도되고 있다 (예: object-oriented/component-based software development).

Autonomy 의 증대

위성과 지구와의 통신 제약의 극복과 위성 운영 비용의 절감을 위한 위성의 autonomy 는 중요한 연구 분야가 되고 있다. 정해진 일련의 명령을 운영자의 개입 없이 자동으로 수행하는 단계를 넘어, 대응 규칙 (event-driven rules)에 따라 미리 프로그램된 명령군을 수행하여 예상치 못한 상황에 대처하거나, 인공 지능을 이용하여 위성 스스로 명령을 생성하여 상황에 대처하도록 하는 연구가 행성 탐사위성 개발에 시도되고 있다 (예: NASA 의 Deep Space-1 의 Remote Agent Experiment). 또 위성 operation 비용 절감을 위하여 지상에서 여러 가지의 명령들 대신에 필요한 임무 (goal)를 high-level scripts 로 지정하면 이를 위성내부에서 해석하여 여러 명령으로 변환하여 수행하는 spacecraft control language 에 대한 연구나, 탑재체 데이터의 on-board processing 에 관한 연구도 진행 되고있다.

기타

위성탑재 컴퓨터의 고성능화와 데이터 저장장치 (solid state recorder)의 용량 증가에 따라 on-board file management system 의 구현에 대한 연구와, 위성 발사 후 하드웨어 장애를 우회하거나 임무의 변경에 따라 탑재 소프트웨어의 코드 및 데이터 수정해야 할 경우에 대비하여 탑재 소프트웨어의 on-orbit reconfiguration 을 용이하게 하기 위한 연구도 진행되고 있다.

6. 결론

위성 탑재소프트웨어는 위성의 탑재컴퓨터에서 수행되어 위성의 운영을 관리하는 real-time embedded 소프트웨어이다. 탑재 소프트웨어는 위성의 종류나 임무, on-board processing 하드웨어 구조에 따라 다르나 본 논문에서는 위성 탑재소프트웨어의 일반적 기능과 구조, 개발 환경에 대하여 살펴보고, 앞으로의 연구 방향에 대하여 전망해 보았다.

국내에서는 우리별-1,2,3 호 위성 및 다목적 실용위성의 탑재소프트웨어 개발 경험이 있으나, 탑재소프트웨어 분야는 선진국에서 기술 이전을 제한하는 분야로 고 신뢰도의 위성 탑재소프트웨어 개발에 대한 체계적인 연구가 요구된다.

참고문헌

[1] Wiley J. Larson, James R. Wertz, *Space Mission Analysis and Design*, Microcosm, Inc., California, 1995.

[2] Charles E. Trevathan, Thomas D. Taylor, et al., "Development and Application of NASA's First Standard Spacecraft Computer," *Communications of the ACM*, vol. 27, no. 9, September, 1984.

[3] Edward B. Gamble, Jr., Reid Simmons, "The Impact of Autonomy Technology on Spacecraft Software Architecture: A Case Study," *IEEE Intelligent Systems*, pp. 69-75, September/October, 1998.

[4] Ian Sommerville, *Software Engineering*, 5th edition, Addison-Wesley, 1996.

[5] Consultative Committee for Space Data Systems, *Advanced Orbiting Systems, Networks and Data Links: Architectural Specification*, CCSDS 701.0-B-2, November, 1992.

[6] Consultative Committee for Space Data Systems, *Packet Telemetry*, CCSDS 102.0-B-3, November, 1992.

[7] Consultative Committee for Space Data Systems, Telecommand CCSDS 201, 202, 203, November, 1992.

[8] *Mil-Std-1553 Designer's Guide*, DDC ILC Data Service Co., 1992.

[9] Jong-In Lee, John Keller, et al., KOMPSAT Flight Software Requirements Specification, September 1997.

[10] 이종인, 천이진, 외 4 인, "아리랑위성 탑재소프트웨어 소개," 한국정보과학회 학술발표논문집, 제 25 권 1 호, pp. 741-743, April 1998.

이 종 인

1983 년 서울대학교 계산통계학과(이학사). 1985 년 KAIST 전산학과(공학석사). 1999 년~현재 KAIST 전산학과 박사과정. 1985 년~1990 년 삼성전자 정보통신연구소. 1990 년~1994 년 KAIST 인공위성연구센터. 1994 년~현재 한국항공우주연구소 위성응용연구그룹. 관심분야는 위성소프트웨어, embedded real-time software, safety-critical software.

백 홍 열

1975 년 서울대학교 응용물리학과(공학사). 1983 년 Cornell 대학원 응용물리학과(공학석사). 1985 년 Cornell 대학원 응용물리학과(공학박사). 1975 년～1995 국방과학연구소. 1995～현재 한국항공우주연구소 위성응용연구그룹장. 관심분야는 전자광학카메라, 항공우주체계 및 소프트웨어.