데이터베이스 기반의 저궤도 관측위성용 텔레메트리 데이터 처리 방안에 대한 연구

이재승*, 신현규*, 천이진*, 윤정오**
*한국항공우주연구원 위성비행소프트웨어팀
**경운대학교 항공정보통신공학과
e-mail:jaeseung@kari.re.kr

A Study on Telemetry Data Processing based on Database Tables for LEO Satellites

Jae-Seung Lee*, Hyun-Kyu Shin*, Yee-Jin Cheon*, Jeong-Oh Yun**
*Satellite Flight Software Team, Korea Aerospace Research Institute
**Dept of Aviation Information & Communications, Kungwoon University

요 약

위성의 상태를 모니터링하고 임무수행 준비 및 결과를 분석하기 위해 위성에서는 주기적으로 텔레메트리 프레임을 생성하여 지상으로 전송한다. 텔레메트리 프레임을 통해 많은 데이터가 전송될수록 정확한 위성의 상태 분석이 가능하고 위성 운용을 용이하게 할 수 있다. 그러나 위성에서 지상으로 전송할 수 있는 텔레메트리의 전송속도는 하드웨어의 성능에 따라 제한되며, 특히 저궤도 위성의 경우에는 지상과 교신이 가능한 시간이 짧다는 제약으로 인해 한정된 시간 안에 정해진 전송속도로 보낼 수 있는 데이터의 양에는 한계가 있다. 이러한 제약조건 하에서 최대한 많은 정보를 효율적으로 전송할 수 있도록 위성의 텔레메트리를 생성할 때 비트 정보들을 모아 하나의 바이트로 묶어서 텔레메트리 크기를 최소화하는 방법을 이용한다.

위성비행소프트웨어는 태스크 스케쥴링, 열제어, 전력제어, 자세제어, 원격명령처리, 원격측정데이터 처리 등의 기능별로 모듈화 되어있다. 각 모듈마다 텔레메트리로 전송되는 데이터들이 존재하고 비트 정보들을 모으는 기능도 해당하는 모듈에서 각각 담당한다. 따라서 각 모듈들이 독립적이지 못하고 텔레메트리 처리를 담당하는 모듈과 다른 모듈들 간의 커플링(coupling)이 존재하게 되어 하나의 텔레메트리 데이터 변경이 여러 모듈에 영향을 미치게 된다.

본 논문에서는 모듈들 간의 커플링을 최소화하고 텔레메트리의 변경사항이 위성비행소프트웨어 코드 자체에는 영향을 주지 않도록 하기위한 데이터베이스 테이블을 이용한 텔레메트리 처리 방안에 대하여 설명한다.

1. 서론

위성의 상태를 모니터링하고 위성운영 및 임무수행을 분석하기 위해 위성은 주기적으로 텔레메트리 프레임에 위성의 상태데이터를 실어 지상으로 전송한다. 위성의 상태 정보가 많으면 많을수록 위성 운영에 효과적이지만 위성의 데이터 전송속도는 해당 하드웨어의 성능에 따라 제한되기 때문에 위성에서 생성되는 텔레메트리 프레임의 크기는 전송속도에 따라 일정한 값으로 정해진다. 또한 저궤도 위성의 경우 지상과 교신이 가능한 시간이 짧기 때문에 제한된 텔레메트리 프레임 크기를 최대한 효율적으로 활용해야 한다. 이를 위해 위성비행소프트웨어에서는 8-비트보다 작은 비트 정보들을 모아서 하나의 바이트로 묶어주는 Bit-Packing 작업을 수행하여 텔레메트리 크기를 최소화 한다.

위성비행소프트웨어와 같이 많은 개발 인원이 함께 수행해야하는 방대한 양의 코드 개발이 필요한 대규모 프로젝트에서는 전체 소프트웨어를 여러 가지 기준에 따라 분류하여 나누어진 소프트웨어 간에는 서로의 존속성을 줄여 독립적인 개발이 가능하도록 하는 방법이 사용된다. 현재까지 개발된 위성비행소프트웨어의 경우에도 각 기능별 분류를 통해 모듈화 설계를 적용하였으며 각 모듈별로 해당 개발자가 존재한다. 그러나 실제적으로 완전하게 독립적인 모듈화가 이루어지지 못하였기 때문에 개별 모듈만으로 개발, 시험, 검증을 수행하기에는 어려움이 있었다. 특히 텔레메트리 프레임의 효율적 사용을 위한 Bit-Packing의 경우, 비트 정보들은 여러 모듈에 걸쳐서 존재할 수 있으며 이러한 Bit-Packing의 수행도 각 모듈에서 별도로 수행된다. 따라서 관련된 텔레메트리 정보나 해당하는 소스코드의 변경이 있을 경우 다른 모듈에도 영향을 주게 된다.

위성비행소프트웨어와 같은 대규모 프로젝트에서 각 모듈간의 존속성을 없애기 위한 다양한 소프트웨어 개발 방안들이 연구되어져 왔으며, 대부분의 연구결과 이러한 커플링 현상을 완전히 없앨 수는 없으며 최대한 줄일 수 있는 방안들이 제시되어져 왔다. 이러한 방안들에 대한 비교 및 분석을 통해 향후 개발되는 위성비행소프트웨어의

개발에 적용하기 위한 노력이 시도되고 있다.

본 논문에서는 이러한 노력의 일환으로 Bit-Packing을 수행하기 위해 해당하는 비트정보들을 데이터베이스화 하여 테이블을 이용한 방안을 제시한다. 테이블에 입력된 데이터베이스를 이용함으로서 관련된 텔레메트리 정보나 해당하는 소스코드의 변경이 있을 경우 소스코드에는 영향이 없이 데이터베이스 수정만으로 해당 변경사항이 가능하고, 여러 모듈들에 흩어졌던 비트 정보들도 서로 참조하지 않고 데이터베이스의 내용만을 이용하므로 각 모듈간 커플링 효과도 줄일 수 있다.

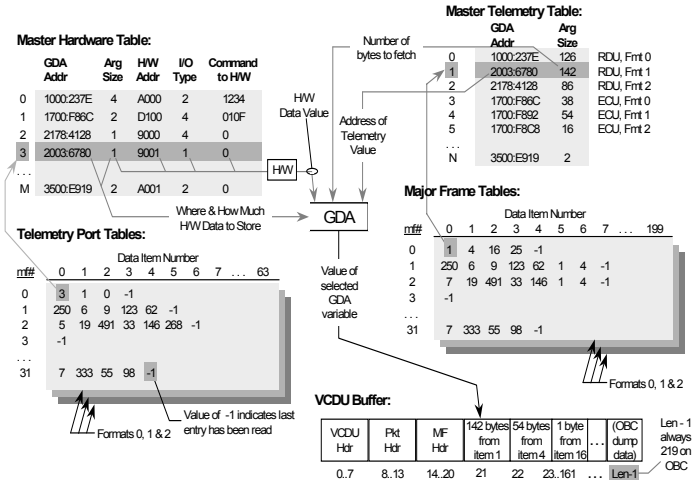
2절에서는 테이블을 이용한 텔레메트리 프레임 생성 방법에 대해 간단히 설명하고 3절에서는 Bit-Packing을 위해 테이블을 이용하는 방안에 대해 소개한다.

2. 테이블을 이용한 텔레메트리 프레임 생성

기존에 개발된 저궤도 관측위성의 비행소프트웨어에서는 텔레메트리 프레임을 생성하기 위하여 아래의 테이블들을 이용하였다.([1],[2])

- MTT(Master Telemetry Table) : 모든 텔레메트리 항목들에 대한 억세스 정보(Address, Size)
- MFT(Major Frame Table) : 매 초마다 생성해야 할 프레임에 저장되어야할 텔레메트리 항목들에 대한 정보(해당 텔레메트리에 대한 MTT 인덱스)
- MHT(Master Hardware Table) : 하드웨어에서 획득되어야하는 텔레메트리 항목들의 하드웨어 정보(Address, Size, H/W Address, I/O Type, Command)
- TPT(Telemetry Port Table) : 매 초마다 생성되는 텔레메트리 프레임의 데이터 중에서 하드웨어에서 획득되어야하는 데이터가 있을 경우 해당 데이터에 대한 MHT 인덱스 정보

다음의 그림 1에 이러한 테이블들을 이용하여 텔레메트리 프레임을 생성하는 방법을 나타내었다.



(그림 1) Telemetry Frame Generation

MHT와 TPT는 하드웨어 데이터 획득을 위한 테이블로서 텔레메트리 프레임 생성에 앞서 해당 프레임의 TPT에 항목들이 존재할 경우 인덱스가 가르키는 MHT를 참조하여 필요한 하드웨어 데이터를 먼저 획득한다. 그리고 MFT에서 현재 프레임에 해당하는 행에 있는 MTT 인덱스를 하나씩 읽어서 인덱스가 가르키는 MTT의 정보를 참조하여 해당하는 주소에서 정해진 크기의 데이터를 가져와서 텔레메트리 프레임에 저장하는 방식을 이용한다.

3. 비트 패킹 방법

기존에 사용된 비트정보를 하나의 텔레메트리로 묶어주는 방식은 아래의 그림 2와 같이 단순하게 각 모듈에서 해당 기능을 코드로 직접 구현하였다.

```
typedef struct
{
  UINT8 element8:1;
  UINT8 element7:1;
  UINT8 element6:1;
  UINT8 element5:1;
  UINT8 element4:1;
  UINT8 element3:1;
  UINT8 element2:1;
  UINT8 element1:1;
} SW_falgs_struct;

SW_flags_struct    SW_flags;

SW_flags.element1 = (UINT8)SW_heater1_onoff;
SW_flags.element2 = (UINT8)SW_heater2_onoff;
                          …
```

(그림 2) Example of Previous Bit-Packing Method

그림 2와 같은 코드가 각 모듈마다 존재하여 Bit-Packing 기능이 구현되어져 있으며, 해당 텔레메트리 정보가 변경되면 소스코드 자체가 수정되어야 한다. 또한 여러 모듈에 흩어져 있는 비트 정보의 Bit-Packing의 경우 다른 모듈의 변수를 사용해야 하므로 모듈간 의존성을 증가시킨다.

이러한 단점들을 보완하기 위하여 Bit-Packing에 사용되는 정보들을 2절에서와 같은 방식으로 테이블로 작성하여 사용할 수 있다. Bit-Packing된 텔레메트리 정보는 지상에서 수신하여 다시 각 비트 정보로 재구성해서 위성관제 요원에게 보여줘야 하기 때문에 실제 위성개발 시 패킹된 텔레메트리를 구성하는 각 비트에 대한 정보가 데이터베이스로 정의되어져 있다. 그러므로 Bit-Packing을 위한 테이블 작성에 이미 작성된 데이터베이스를 활용하므로 이를 위한 추가적인 개발노력은 필요하지 않다. 기존의 위성개발 시에도 작성되었던 비트 정보에 대한 데이터베이스는 다음의 그림 3과 같은 형태이다.

Destination(MOM)	Mnemonic	Source	Mom Type	Source Type	Bit Position	Bit Size
ACS_aenflgs1	TAWNULSPSW	ACS_automatic_null_space_onoff	UINT16	UINT8	0	1
	TAMINJERK	ACS_min_jerk_man_enable	UINT16	UINT8	1	1
	TAGEODLVLH	ACS_LVLH_geodetic_flag	UINT16	UINT8	2	1
	TATACHMFLAG	ACS_tach_M_prcs	UINT16	UINT8	3	1
	TAMOMFLAG	ACS_imm_enable	UINT16	UINT8	4	1
	TAROLLEN	ACS_imom_x_enable	UINT16	UINT8	5	1
	TAPITCHEN	ACS_imom_y_enable	UINT16	UINT8	6	1
	TAYAWEN	ACS_imom_z_enable	UINT16	UINT8	7	1

(그림 3) Example of Bit-Packing Database

그림 3과 같은 데이터베이스를 기반으로 위성비행소프트웨어에서는 다음의 그림 4와 같은 형태의 테이블을 작성할 수 있다. 현재 향후 위성비행소프트웨어의 개발을 위하여 수행하는 기능에 따라 서비스를 분류하고 서비스 간의 커플링을 최소화하기 위한 설계 개념을 도입하고 있다.([3]) 이러한 설계 개념의 일환으로 매퍼(Mapper)를 활용할 예정이며, 매퍼를 사용할 경우 모든 변수들의 주소를 개발자가 원하는 영역으로 지정할 수 있다. 따라서 아래의 그림 4에서 데이터베이스의 변수들이 모두 주소값으로 변환된 것을 알 수 있다.

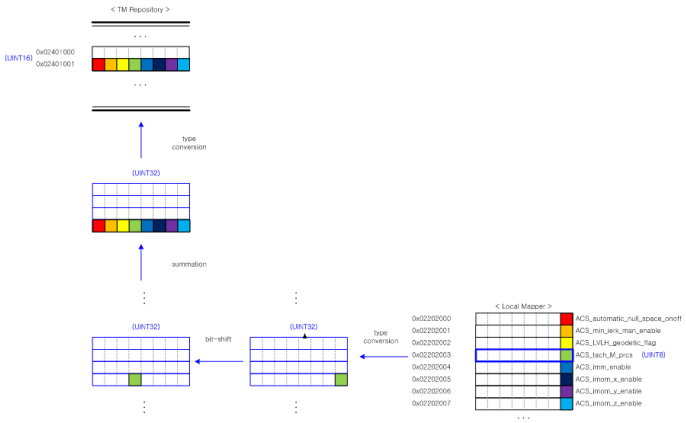
```
typedef struct
{
   UINT32 src_addr;
   UINT8  src_type
   UINT8  start_bit;
   UINT8  bit_size;
   UINT8  dest_type;
   UINT32 dest_addr;
} TM_build_mom_strct_type;

TM_build_mom_strct_type TM_build_mom_table[TM_BUILD_MOM_MAX_SIZE][TM_MOM_BIT_MAX_SIZE] =
{
   /* ACS_AENFLGS1 */
   {
      {0x02202000,  TYPE_UINT8,   0,   1,   TYPE_UINT16,   0x02401000}, /* TAWNULSPSW  */
      {0x02202001,  TYPE_UINT8,   1,   1,   TYPE_UINT16,   0x02401000}, /* TAMINJERK   */
      {0x02202002,  TYPE_UINT8,   2,   1,   TYPE_UINT16,   0x02401000}, /* TAGEODLVLH  */
      {0x02202003,  TYPE_UINT8,   3,   1,   TYPE_UINT16,   0x02401000}, /* TATACHMFLAG */
      {0x02202004,  TYPE_UINT8,   4,   1,   TYPE_UINT16,   0x02401000}, /* TAMOMFLAG   */
      {0x02202005,  TYPE_UINT8,   5,   1,   TYPE_UINT16,   0x02401000}, /* TAROLLEN    */
      {0x02202006,  TYPE_UINT8,   6,   1,   TYPE_UINT16,   0x02401000}, /* TAPITCHEN   */
      {0x02202007,  TYPE_UINT8,   7,   1,   TYPE_UINT16,   0x02401000}, /* TAYAWEN     */
      {         0,           0,   0,   0,            0,             0}, /* End of MOM  */
   },

   /* ACS_TAGRAFLAGS */
   {
      {0x02102000,  TYPE_UINT8,   0,   1,   TYPE_UINT16,   0x02401002},
      ...
      {         0,           0,   0,   0,            0,             0}, /* End of MOM  */
   },
   ...
};
```

(그림 4) Example of Table for Bit-Packing

그림 4와 같은 테이블은 자동화 툴에 의하여 데이터베이스를 입력으로 자동적으로 생성되므로 개발자의 입력에 의한 오류를 제외시킬 수 있다. 작성된 테이블은 2절에서 설명한 텔레메트리 프레임 생성 방법과 동일한 방식으로 테이블에서 차례대로 비트 정보에 대한 억세스 정보를 참조하여 해당 데이터를 읽어와서 하나의 텔레메트리로 Bit-Packing을 수행하게 된다. 다음의 그림 5에는 각 비트 정보를 가지고 텔레메트리로 패킹되는 과정을 보여준다. 이 과정에서 실제 코드 구현 시 해당 비트 정보 및 텔레메트리의 데이터 타입이 다양할 수 있기 때문에 모두 UINT32로 변환한 뒤 Bit-Packing을 수행한 후 결과가 저장될 텔레메트의 데이터 타입으로 최종적으로 변환되는 것을 확인할 수 있다.



(그림 5) Example of Bit-Packing

4. 결론

본 논문에서는 제한된 텔레메트리 데이터 전송속도를 효율적으로 사용하기 위한 Bit-Packing 과정에서 데이터베이스를 기반으로 한 테이블을 활용하는 방안을 설명하였다. 기존의 위성비행소프트웨어에서는 소스코드에 직접 해당기능을 구현하여 텔레메트리 변경 시 여러 모듈들에 영향을 줄 수 있고 코드를 직접 변경해야만 했지만 본 논문에서 소개한 테이블을 활용할 경우 이러한 단점들을 보완하고 각 모듈 간 커플링을 줄일 수 있다.

현재 위성비행소프트웨어에서 커플링을 최대한 줄이기 위한 연구가 진행되고 있으며, 새롭게 도입된 매퍼 개념을 적용한 텔레메트리 프레임 생성 방안에 대한 설계가 계속 진행되고 있다.

참고문헌

[1] 이재승, “다목적실용위성 2호의 데이터 획득을 위한 자동적인 포맷 테이블 생성”, 한국정보과학회 가을 학술발표논문집(Ⅲ), 제 28권 2호, pp. 508-510, 2001

[2] 이재승, “다목적실용위성 2호의 데이터 획득을 위한 자동적인 포맷 테이블 생성”, 한국산업정보학회 춘계학술대회, pp. 70-74, 2003

[3] F.B. Abreu, “Coupling and Cohesion as Modularization Drivers”, CMSR, IEEE Computer Society, pp. 47-57, 2001