저궤도 관측위성의 히터제어를 위한 위성비행소프트웨어 설계

이재승*, 신현규**, 최종욱***, 천이진****

Design of Flight Software for Heater Control in LEO Satellites

Jae-Seung Lee*, Hyun-Kyu Shin**, Jong-Wook Choi***, Yee-Jin Cheon****

Abstract

LEO satellites have many heaters for thermal control, such as bus module heaters, payload heaters and battery internal heaters. Some of these heaters are controlled by thermisters, and others can be controlled by flight software.

These heaters are divided into various types of group according to the location, telemetry variables, flight software logic, power distribution, etc. Thus, it is difficult to find out which heaters are inlcuded in a certain group and modify heater control logic for a new/other software developers.

This document describes about the general/special control logic for satellite heaters and groups/arrays for heaters.

초 록

저궤도 관측위성에는 버스 히터, 탑재체 히터, 배터리 내부 히터 등 다양한 히터들이 각각의 해당 영역에 대한 열제어를 위해 존재한다. 이러한 히터들의 제어는 서미스터에 의해 수행되거나 비행소프트웨어에 의해 제어될 수 있다.

각 히터들은 설치된 위치, 텔레메트리로 전송하기 위한 분류, 사용되는 서브시스템 등에 따라서 여러 형태의 그룹으로 나눌 수 있으며, 비행소프트웨어에서는 히터제어를 위한 정보들을 다양한 배열에 저장하는데 각 히터마다 고유의 인덱스를 부여하여 구분하는 방법을 사용할 수 있다. 각 히터들이 분류하는 방식에 따라 서로 다른 그룹에 속하기도 하고 비행소프트웨어 로직에서 사용되는 히터정보가 어느 히터, 또는 어느 그룹의 정보인지를 판별하는데 어려움이 있을 수 있다.

본 문서에서는 저궤도 관측위성의 일반적인 히터 제어를 위한 비행소프트웨어의 설계에 대해 기술하고, 히터들의 그룹 및 배열의 활용과 특별한 관리가 필요한 히터들의 제어방식에 대하여 설명한다.

키워드 : 비행소프트웨어(flight software), 히터(heater), 열제어(thermal control)

접수일(2010년 12월 21일), 수정일(1차 : 2011년 5월 4일, 2차 : 2011년 6월 15일, 게재 확정일 : 2011년 7월 1일)

* 위성비행소프트웨어팀/jslee@kari.re.kr ** 위성비행소프트웨어팀/hkshin@kari.re.kr
*** 위성비행소프트웨어팀/jwchoi@kari.re.kr **** 위성비행소프트웨어팀/yjcheon@kari.re.kr

1. Introduction

저궤도 관측위성에는 버스 히터, 탑재체 히터, 배터리 내부 히터 등 다양한 히터들이 각각의 해당 영역에 대한 열제어를 위해 존재한다. 이러한 히터들의 제어는 서미스터에 의해 수행되거나 비행소프트웨어에 의해 제어될 수 있다.

각 히터들은 설치된 위치, 텔레메트리로 전송하기 위한 분류, 사용되는 서브시스템 등에 따라서 여러 형태의 그룹으로 나눌 수 있으며, 비행소프트웨어에서는 히터제어를 위한 정보들을 다양한 배열에 저장하는데 각 히터마다 고유의 인덱스를 부여하여 구분하는 방법을 사용할 수 있다. 각 히터들이 분류하는 방식에 따라 서로 다른 그룹에 속하기도 하고 비행소프트웨어 로직에서 사용되는 히터정보가 어느 히터, 또는 어느 그룹의 정보인지를 판별하는데 어려움이 있을 수 있다.

본 문서에서는 저궤도 관측위성의 일반적인 히터 제어를 위한 비행소프트웨어의 설계에 대해 기술하고, 히터들의 그룹 및 배열의 활용과 특별한 관리가 필요한 히터들의 제어방식에 대하여 설명한다.

2. Heater Groups and Arrays

비행소프트웨어에서는 제어목적에 따라 히터들을 각각의 그룹으로 분류하고 있으며 각 히터 정보를 저장하는 변수는 배열의 형태로 선언되어 배열의 인덱스를 통해 히터들을 구분하고 있다. 각각의 기준에 따라 분류된 그룹별로 포함되는 히터들이 상이하기 때문에 히터제어 로직의 변경 또는 검토 시 혼동하기 쉽다.

본 절에서는 히터제어 로직을 용이하게 파악하고 그룹 및 배열 인덱스의 정의를 명확히 하기 위하여 비행소프트웨어에서 각 히터들의 정보를 가지고 있는 변수들의 그룹 정보 및 인덱스 정보가 어떻게 연결되고 관리되는지에 대하여 설명한다.

2.1 Heater Groups

저궤도 관측위성에서는 일반적으로 각 히터들의 Primary와 Redundant가 존재하며 소프트웨어 요구사항 규격서에 다음과 같은 형식으로 해당 분류를 정의해 준다.

- Primary Heater Groups : Bus-P, Bus T/S-P, Payload-P, etc.
- Red. Heater Groups : Bus-R, Bus T/S-R, Payload-R, etc.

위의 그룹에 속한 히터들 중 서미스터를 이용하여 하드웨어적으로 제어되는 히터들을 제외한 히터들에 대해 비행소프트웨어에서 S/W 제어를 수행하며 주로 다음과 같은 부분체에 속한 히터들을 제어하게 된다.

- Battery, Star Tracker Assembly, Reaction Wheel Assembly, Battery Internal, GPS, Payload, Camera Assembly, STA (Star Tracker) Optical Head, etc.

지상에서 히터들의 제어상태를 확인하기 위하여 각 히터 및 히터그룹의 ON/OFF 정보가 텔레메트리로 전송되어 진다. 이 정보들은 16비트 크기의 텔레메트리의 각 비트별로 해당하는 히터가 설정되어 있으며 히터의 수에 따라 여러 텔레메트리로 나누어져서 전송된다. 만약 16비트 텔레메트리 4개가 사용된다고 가정하면 이에 해당하는 g1 ~ g4, 4개의 그룹 정보를 저장하고 있는 변수들(지상으로의 텔레메트리 전송에 사용되는 해당 히터들로부터 정해진 인터페이스를 통해 전송받은 값을 저장하는 변수와 히터 제어를 위해 사용되는 변수)과 각 변수들의 비트 정보가 할당되는 예를 다음의 그림 1에 나타내었다.

그림 1의 각 그룹 히터 정보에 대해 지상으로 전송된 텔레메트리를 분석하기 위한 Mnemonic이 다음과 같은 방식으로 설정될 수 있다.

- g1 : Heater Board #1, TLM #1 ON/OFF Status
- g2 : Heater Board #1, TLM #2 ON/OFF

그림 1. 전송되는 텔레메트리에 따른 g1~g4 히터그룹 분류

Status
- g3 : Heater Board #2, TLM #1 ON/OFF Status
- g4 : Heater Board #2, TLM #2 ON/OFF Status

그림 1에서 P1, P2, P3, P4, R4로 표시된 히터들의 그룹은 각 히터들에 대한 전력 분배 설계에 따라 분류된 그룹을 나타낸다. 위성에 장착된 모든 히터들의 전원 ON/OFF와 관련한 Primary 및 Redundant 그룹은 각 위성에서 사용되는 히터의 수에 따라 더 많거나 작은 그룹으로 분류될 수 있다. 본 논문에서는 그림 1에 표시된 그룹들은 S/W 제어와 관련된 그룹으로 가정한다.

이 그룹 스위치의 ON/OFF 제어에 따라 각 그룹에 속한 히터들의 전원도 당연히 ON/OFF 되지만 실제 운영 시에는 안전을 위하여 그룹스위치 제어와 함께 각 히터들의 ON/OFF도 각각 수행해 주도록 로직이 설계되어야 한다.

실제 위성 시스템의 설계에서는 PCDU(Power Control and Distribution Unit)에서 설계된 각 히터 전원의 인터페이스에 따라 그룹이 정해지지만 비행소프트웨어에서는 지상으로 전송되는 텔레메트리 정보에 따라 별도의 그룹으로 관리하므로 관련 히터제어 로직의 이해를 위해 그림 1과 같은 그룹별 히터 분류를 명시할 필요가 있다.

2.2 Arrays for Heater Status

저궤도 관측위성에서 S/W 제어를 수행하는 히터는 수십여개 이상이며, 히터제어를 위하여 각각의 히터에 대한 정보를 별도의 변수로 할당하기에는 너무 많은 변수들이 필요하므로 동일한 종류의 정보들은 하나의 배열로 선언하고 배열의 인덱스를 각 히터별로 할당하여 사용하는 방법을 이용한다. 비행소프트웨어의 히터제어 로직을 검토하거나 변경이 요구될 경우 배열에서 해당 히터들을 확인하기 위해서는 각 인덱스에 해당하는 히터들이 무엇인지를 파악해야 한다. 본 절에서는 히터제어에 사용되는 배열들과 각 배열의 인덱스를 이용하여 히터를 구분하는 방법에 대하여 설명한다.

비행소프트웨어의 히터제어 로직은 크게 Bus 히터 제어부분과 Payload 히터 제어부분으로 나누어져 있으며 히터 정보를 저장하는 배열들도 Bus와 Payload로 나누어져 있다.

Bus와 Payload 히터 제어를 위한 로직은 각각의 히터 자체 특성에 의한 차이만 있을 뿐 기본적인 제어 개념과 설계 방법은 유사하지만, Bus와 Payload 히터의 제어 주기가 다르고 히터제어 로직 수행 시 히터정보의 혼선으로 인한 위성의 오동작을 방지하기 위하여 Bus 및 Paylaod 히터의 관련 변수들을 별도의 배열로 선언하였다. 또한 "heater_on[]" 및 “heater_off[]”로 선언된 배열에는 ON/OFF Status와 함께 해당 히터의 ON/OFF 명령을 전송하는데 사용되는 값을 포함하고 있기 때문에 ON과 OFF에 따라 별도의 배열로 할당되어져 있다.

먼저 Bus 히터 정보를 저장하기 위해 아래와 같은 배열 및 구조체를 선언하여 활용할 수 있다.

- UINT16 Bus_heater_on[N], UINT16 Bus_heater_off[N]
  : N개 Bus 히터들의 ON/OFF 명령정보
- UINT16 Bus_temp_upper_limit[N], Bus_temp_lower_limit[N]
  : N개 Bus 히터들 각각을 온도 센서 정보에 따라 ON/OFF하기 위한 upper limit과 lower limit 온도 값 (모드에 따른 Key Parameter Database 값을 저장)

- struct Bus_heater_data_struct
  {
  UINT8 sensor_select;
  UINT8 sensor_error;
  UINT8 control_flag;
  UINT16 temp;
  } Bus_heater[N]
  : N개 Bus 히터들에 대한 온도, 에러, 제어 정보를 저장하기 위한 구조체

위에서 선언된 Bus_heater[], Bus_heater_on[], Bus_heater_off[], Bus_temp_upper_limit[], Bus_temp_lower_limit[] 배열의 인덱스별로 나타내는 히터를 다음의 그림2와 같은 방식으로 모든 변수에 동일하게 설정하여 사용하면 제어 로직에 적용하기 편리하다.

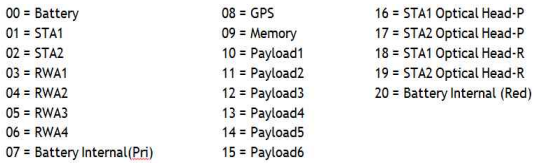

그림 2. 버스히터 제어용 배열의 인덱스 설정 예

Payload 히터 정보를 저장하기 위해서는 아래와 같이 배열들을 선언하여 활용할 수 있으며, Bus 히터의 경우와 거의 동일하지만 탑재체는 위성의 임무 수행과 밀접한 관계가 있으므로 보다 정확한 히터제어를 위하여 float 값의 임계값을 사용할 수도 있다.

- UINT16 Payload_heater_on[M], Payload_heater_off[M]
  : M개 탑재체 히터들에 대한 ON/OFF 명령정보
- float Payload_temp_upper_limit[M], Payload_temp_lower_limit[M]
  : M개 탑재체 히터들 각각을 온도 센서 정보에 따라 ON/OFF하기 위한 upper limit과 lower limit 온도 값 (모드에 따른 Key Parameter Database 값을 저장)
- struct Payload_heater_data_struct
  {
  UINT8 sensor_error;
  UINT8 control_flag;
  float temp;
  } Payload_heater[M];
  : M개의 탑재체 히터 각각에 대한 온도, 에러, 제어 정보를 저장하기 위한 구조체

버스 히터제의 경우와 동일한 방법으로 탑재체 히터제어를 위한 Payload_heater[], Payload_heater_on[], Payload_heater_off[], Payload_temp_upper_limit[], Payload_temp_lower_limit[] 배열의 인덱스별로 나타내는 히터를 다음의 그림3과 같이 설정하여 탑재체 히터제어 로직에 활용할 수 있다.

00 = Camera A (Pri)	04 = Camera A (Red)
01 = Camera B (Pri)	05 = Camera B (Red)
02 = Camera C (Pri)	06 = Camera C (Red)
03 = Camera D (Pri)	07 = Camera D (Red)

그림 3. 탑재체히터 제어용 배열의 인덱스 설정 예

3. Heater Control Logic

비행소프트웨어서 제어하는 히터들에는 버스쪽 히터들과 탑재체쪽 히터들이 포함되어있으며 히터제어 로직은 버스와 탑재체 히터들에 대해 분리되어 설계되어졌다. 기본적인 제어개념은 동일하므로 본 절에서는 버스히터 제어를 위한 비행소프트웨어 로직을 기준으로 설명한다.

3.1 General Heater Control Logic

버스히터의 제어를 위한 비행소프트웨어의 로직은 거의 대부분의 히터들에 대해 동일하게 적용되며, 일부 특성이 다른 히터의 경우와 시스템 설계에 따라 특정한 제어방식이 요구되는 경우에는 별도의 로직을 설계하여 구현하게 된다.

본 절에서는 저궤도 관측위성에서 일반적으로 적용될 수 있는 히터제어용 비행소프트웨어 설계에 대하여 설명한다.

첫 번째로 히터제어관련 변수들의 초기화 로직이 당연히 수행되어야 하며 비행소프트웨어 부트업 시 한번만 수행된다. 그리고 히터의 제어주기에 따라 수행되어야 할 히터제어 로직을 호출하는 메인함수가 존재한다. 또한 버스히터와 탑재체 히터를 별도로 제어하므로 각각의 히터제어를 총괄하는 메인함수도 별도로 구성할 수 있다.

히터제어 로직의 시작은 제어할 히터의 온도센서 데이터를 읽어 유효성을 판단하고 히터제어에 사용할 센서 값 선택 등의 온도 데이터 처리를 수행하는 것이다. 이러한 온도 데이터 처리를 위해서는 I/O 모듈로부터 읽어온 온도 데이터를 2.2절에서 설명한 각 히터의 해당 배열에 저장하는 작업이 선행되어야 한다.

각 히터의 상태를 점검하여 온도에 따라 해당 히터의 ON/OFF 명령을 전송하거나 현 상태를 유지하도록 한다. 이러한 히터제어 로직의 수행이 완료되면 각 히터의 상태 데이터를 지상으로 전송하기 위한 텔레메트리 값을 설정하게 된다.

이상에서 설명한 기능들이 수행되면서 이루어지는 저궤도 관측위성의 일반적인 비행소프트웨어의 히터제어 로직의 설계 내용을 그림 4에 나

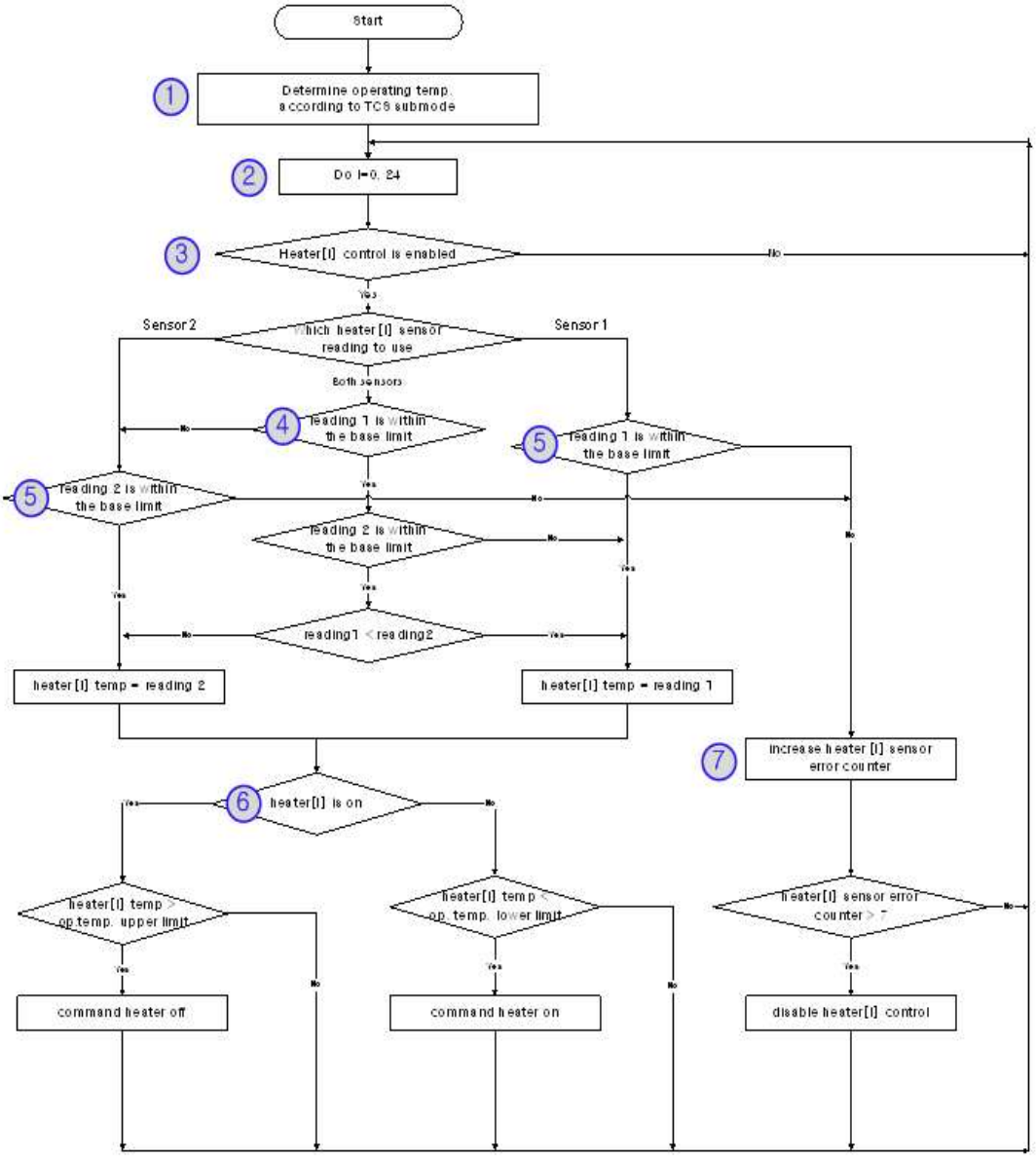

그림 4. 저궤도 관측위성에서의 히터제어 로직 설계

타내었다.

그림 4에 제시된 히터제어 로직은 앞에서 언급했던 특별한 제어방식이 요구되는 히터의 경우는 논외로 하였으며 제어를 수행하는 히터 개수가 24개인 경우를 가정하였다.

그림 4의 로직을 순서에 따라 각 단계별로 아래에 설명하였다.

① 위성은 지상에서의 시험을 위한 Test Mode, 임무를 수행하는 동안 유지해야하는

Science Mode, 이상상황이 발생하여 초기화가 이루어진 후 지상과의 교신을 기다리기 위한 Survival Mode 등과 같은 여러 모드들을 정의하여 비행소프트웨어 설계 및 위성운영에 적용한다. 이러한 위성의 모드에 따라 각 히터를 ON/OFF하기 위한 제어온도가 다르게 설정되어 있으므로 히터제어 로직을 수행하기 전에 먼저 현재의 모드를 확인하고 각 모드에 해당하는 제어온도 값을 결정해야 한다.

② 제어해야 하는 히터 개수를 24개로 가정하였으므로 동일한 제어로직을 24번 반복해서 각 히터별로 수행한다.

③ 센서나 히터 자체의 문제로 인해 제어가 불가능한 히터가 발생하거나 위성의 모드에 따라 사용하지 않는 히터들은 항상 OFF 상태로 유지되어야 하므로 해당 히터의 control flag을 "DISABLE" 상태로 설정하여 제어로직이 수행되지 않도록 한다.

④ 각 히터에는 온도센서가 2개씩 존재하여 하나의 센서에 문제가 발생하더라도 히터제어를 수행할 수 있도록 해준다.
2개의 센서가 모두 정상적으로 작동할 경우에는 각 센서값이 운영온도 범위 내에 존재하는지 여부를 판단한다. 2개 모두 범위 안의 온도값을 가질 경우 낮은 온도 값을 기준으로 히터제어를 수행한다.

⑤ 정상적으로 동작하지 않는 센서가 존재할 경우 나머지 센서값이 운영온도 범위 내에 있는지 판단하여 범위내의 온도값을 가질 경우 해당 값을 기준으로 히터제어를 수행한다. 운영범위를 벗어났을 경우에는 히터제어에 사용할 수 있는 온도값이 없으므로 에러를 처리하기 위한 ⑦번의 단계를 수행한다.

⑥ 현재 히터의 상태가 ON일 경우 해당 센서의 온도값이 상한값 이상이면 히터를 OFF하기 위한 명령을 전송하며, 반대로 히터의 상태가 OFF일 경우 해당 센서의 온도값이 하한값 이하이면 히터를 ON하기 위한 명령을 전송하여 관련된 위성파트의 온도가 정해진 범위내에 유지되도록 해 준다.

⑦ 2개의 센서값이 모두 히터제어에 사용할 수 없을 경우 에러로 처리되며 에러 발생회수를 증가시킨다. 만약 에러가 임계값(그림 4의 설계상에는 이 값을 "7"로 설정하였다.)을 초과하여 발생하게 되면 해당 히터는 사용이 불가능하다고 판단하여 히터의 control flag을 "DISABLE"로 설정하여 이후부터 해당히터에 대해서는 제어로직을 수행하지 않도록 한다.

3.2 FET(Field Effect Transistor) Short Error

3.1절에서 설명한 일반적인 히터 제어방식은 그림 4의 로직을 사용하지만 위성의 임무수행에 미치는 영향이 크거나 고도의 안정성이 요구될 경우 특별한 제어 로직으로 설계되어야 하는 히터가 존재한다. 이러한 히터의 한 예로서 서브시스템의 요구에 의해 FET Short(Open) Error가 발생했을 경우에 이를 대처하기 위해 수행되는 로직이 별도로 추가되기도 한다. FET Short Error는 직전에 전송된 히터 제어명령(ON/OFF)과 현재의 히터 상태가 정해진 회수 이상 연속적으로 다를 경우 발생하게 되며 에러 발생 시에는 해당 히터가 속한 그룹 및 그룹의 모든 히터 전원을 끄고 더 이상 제어를 할 수 없으므로 제어 플래그를 DISABLE한다. 만약 Primary에서 발생하였을 경우에는 Redundant 히터가 속한 그룹 및 그룹의 모든 히터들의 전원을 켜서 계속적으로 온도제어가 가능하도록 한다.

다음의 그림 5에는 STA Optical 히터에서 FET Short Error 발생 시 이러한 제어로직이 요구된다고 가정했을 때 수행되는 로직을 나타내었다.

4. 결 론

본 논문에서는 저궤도 관측위성에서의 버스 히터제어 로직에 대하여 설명하였다. 일반적으로 위성에서 사용되는 히터제어 로직과 함께 특수한 제어방법이 요구되는 경우의 예를 함께 제시하였다.

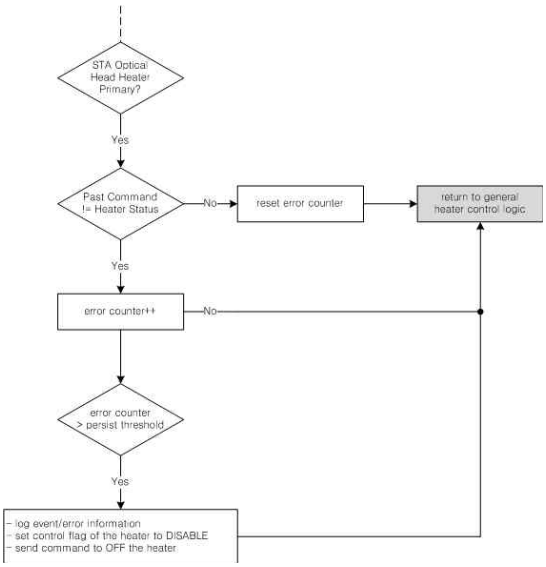

그림 5. FET Short Error에 대한 히터제어 로직 예

향후 히터 제어로직의 검토 및 수정이 필요할 경우 설계된 비행소프트웨어의 이해 및 설계 변경을 용이하게 하기 위한 배열을 이용한 히터정보의 저장방법 및 실제 하드웨어적인 시스템 설계와의 연계를 위한 배열 인덱스의 정보 활용 방안에 대하여 기술하였다.

본 논문의 히터제어 방안은 향후 저궤도 관측위성의 히터 제어로직의 기본설계에 지속적으로 활용할 수 있을 것으로 예상된다.

참 고 문 헌

1. 권동영, “지구 저궤도 위성의 히터 채널 및 열센서의 전기 종합 설계”, 한국항공우주학회 추계학술발표회, 2010, pp.1096-1099
2. 박영복, “위성 열제어시스템 로직 검증 시험”, 한국우주과학회보, 제18권 제2호, 2009, pp. 84-85