http://dx.doi.org/10.5392/JKCA.2012.12.08.077

IMA 시스템에서의 Fault Tolerance를 위한 HM 시스템 설계

HM System Design for Fault Tolerance on the IMA System

고영관*, 이승훈*, 박세영*, 반창봉**, 강대일**, 정재엽**, 이철훈*
충남대학교 컴퓨터공학과*, LIG 넥스원**

Young-Kwan Ko(Yg1985@cnu.ac.kr)*, Seung-Hoon Lee(sh2011@cnu.ac.kr)*,
Se-Young Park(psy6693@gmail.com)*, Chang-Bong Ban(bancb77@lignex1.com)**,
Dai-Il Kang(daeil.kang@lignex1.com)**, Jae-Yeop Jeong(jyjeong@lignex1.com)**,
Cheol-Hoon Lee(clee@cnu.ac.kr)*

요약

항공전자 시스템은 중요한 임무를 수행하는 다양한 전자 장치들로 구성되어지며 항공기의 성능을 위해 전자 장치의 수가 점차적으로 증가되고 있다. 이에 따라 개발 비용, 유지보수 비용, 운영비용 등이 증가되었고, 이를 해결하기 위해 항공전자 시스템의 기술 패러다임은 시스템을 독립적으로 관리하는 연방형 항공전자(Federated Avionics) 시스템에서 모듈 통합형 항공전자(IMA: Integrated Modular Avionics) 시스템으로 발전하고 있다. 모듈 통합형 항공전자 시스템은 연방형 시스템과는 달리 항공기의 각 기능들을 IMA 시스템에서 통합 처리하기 때문에 fault 발생 시 시스템 운용에 큰 영향을 미치게 되므로 고장 허용 기술이 필수 사항으로 적용되고 있다. 본 논문에서는 ARINC 653 기반의 모듈 통합형 항공전자 시스템에서 발생할 수 있는 fault를 정의하고 fault 발생 시 시스템이 지속적으로 정상 동작할 수 있도록 고장 허용 기법 설계 및 ARINC 653 표준에 따른 HM(Health Monitoring) 시스템 설계 방법을 제안한다.

■ 중심어 : | 모듈 통합형 항공전자 시스템 | 고장 허용 | 헬스 모니터링 시스템 |

Abstract

Avionics system are composed of multiple electronic device that performs important missions the number of electronic devices for the performance of aircraft has been gradually increasing. As a result, cost of development, maintenance and operating have increased. To solve this problem, technology paradigm of avionics has been shifting from federated avionics systems that manage to each system independently to IMA(Integrated Modular Avionics) systems. Unlike federated systems, fault tolerance becomes an essential technology in IMA systems. Because each aircraft features integrated in the an IMA system, a fault can jeopardize the entire system. In this paper, we define faults which can occur on the ARINC 653 based IMA system first, and design the ARINC 653 compliant HM(health Monitoring) system for the system can continue to operate be normal when occur a fault.

■ keyword : | Integrated Modular Avionics System | Fault Tolerance | Health Monitoring System |

* 본 연구는 2012년 LIG넥스원의 "항공기 임무컴퓨터 Fault Tolerance 시스템 연구" 과제로 수행되었습니다.

접수번호 : #120702-009
접수일자 : 2012년 07월 02일
심사완료일 : 2012년 08월 20일
교신저자 : 이철훈, e-mail : clee@cnu.ac.kr

I. 서 론

1903년 항공기가 개발된 이래 1910년대 무선 통신을 시작으로 항공전자 시스템은 항공기의 발달과 함께 발전해 왔다. 항공전자 시스템은 항법 시스템, 피아식별, 조종사-항공기 간의 효율적인 인터페이스 제공, 각종 정보 획득 및 외부환경 감지 등 요구되는 비행 임무가 다양해짐에 따라 항공기에 탑재되는 전자 장비들이 크게 증가하였다. 이에 항공기의 전체 무게 및 부피, 시스템의 개발, 성능개량, 유지보수 비용 등이 증가하는 문제가 발생하였다. 이러한 문제를 해결하기 위해 미국, 유럽을 비롯한 항공 선진국에서는 항공전자 시스템의 향상을 목적으로 항공전자 기능의 모듈 통합화 설계를 적용한 모듈 통합형 항공전자 시스템을 개발하였다[1-3].

항공전자 시스템은 과거 연방형(Federated) 시스템에서 IMA 시스템으로 발전됨에 따라 모듈 통합화 설계가 가능하게 되었고, 모듈 통합화에 따른 항공기 무게 및 부피 감소, 전력 소모 감소, 개발 비용 및 유지보수 비용의 감소 효과를 얻게 되었다. 또한 세부 임무를 수행하는 기능 단위 별 서브시스템 간에 여분 모듈을 공유함으로써 시스템 가용도가 증가 하게 되었다. 이와 같은 장점으로 IMA 시스템에 대한 많은 연구가 진행되었고, ARINC 653과 같은 항공 소프트웨어 표준이 등장하였다. ARINC 653은 항공전자 시스템을 위한 실시간 운영체제와 응용 간의 인터페이스를 규정하고 있다. 현재 ARINC 653 표준을 준수하는 VxWorks 653, QPlus-Air, LynxOS 와 같은 실시간 운영체제들이 개발되어 다양한 항공기에 활용되고 있다[4].

IMA 시스템은 연방형 시스템과는 달리 각 임무 수행을 통합 처리하기 때문에 fault 발생 시 fault의 영향이 전체 시스템에 전파될 수 있기 때문에 시스템의 정상동작을 보장할 수 없게 된다. 따라서 모듈 시스템은 fault를 허용하고, 올바른 fault 처리를 통해 지속적으로 정상적인 시스템 운용을 하기 위한 고장 허용 기술이 필요하다. 고장 허용 기술은 고장 탐지(Fault Detection)를 통해 fault를 발견하고, 고장 분리 및 고장 봉쇄(Containment)를 통해 fault가 발생된 영역으로부터 다른 영역으로 fault의 영향이 미치는 것을 미연에 방지한다. 또한 고장 복구 정책을 통해 fault 발생 시 미리 정의된 복구 행동을 통해 시스템이 지속적으로 정상 동작되는 것을 보장한다. 하지만 IMA 시스템에서는 시스템의 복잡성이 증가함에 따라 고장 허용 기술의 설계 및 유지 보수가 어렵다는 문제가 있다[5][6].

이에 본 논문에서는 복잡해진 IMA 시스템의 지속적인 정상 동작을 보장하는 고장 허용 기술을 보다 쉽게 설계하고 관리하기 위해 Fault의 발생을 알리고 복구 행동을 규정하는 HM 시스템을 설계한다. 제안된 HM 시스템에서는 IMA 시스템의 시스템 상태와 발생될 수 있는 각 fault들이 정의된 HM 테이블 설계를 통해 보다 간편한 설계 및 유지보수가 가능하다.

본 논문의 구성은 다음과 같다. 관련연구에서 항공전자 시스템 및 시스템 코어와 응용 간의 인터페이스를 정의한 ARINC 653에 대해 소개하고, 3장에서는 IMA 시스템의 각 시스템 상태와 발생될 수 있는 fault를 정의하고 고장 허용을 위해 HM 시스템을 설계하여 fault 처리 방안을 기술하였으며, 마지막으로 4장에서는 결론 및 향후 연구과제에 대해 기술하였다.

II. 관련 연구

본 절에서는 항공전자 시스템의 시대별 발전에 따른 독립형 시스템, 연방형 시스템 및 IMA 시스템을 설명하고, IMA 시스템에서 사용하고 있는 항공 소프트웨어 표준인 ARINC 653 표준에 대해 소개한다.

1. 항공전자 시스템

항공전자 시스템은 2차 세계대전을 기점으로 다양한 임무를 수행하기 위해 항공기에 탑재되는 장비들이 비약적으로 증가함에 따라 항공전자 시스템의 신뢰성, 가용성 등의 향상을 위해 독립형 시스템부터 IMA 시스템에 이르기까지 발전을 거듭해 왔다.

1.1 독립형(Independent) 시스템

1950년대까지 항공전자는 조종사의 임무 기능 단위

별로 개별 임무장비들이 각각 장착된 독립형 아날로그 시스템으로 구성되었다. 그러나 장착 장비들이 증가하면서 조종석은 제어 및 시현 장비로 넘쳐나고 개별 장비 간 연결 케이블의 증가로 인해 항공기 중량이 과도하게 되어 항공기의 유지보수, 운용비가 증가하게 되는 문제가 발생한다.

1.2 연방형 시스템

개별 시스템에 요구되는 장비 개수 및 케이블을 줄이기 위해 시스템 간에 정보를 공유하고 정보를 통합 처리할 수 있는 임무 컴퓨터(Mission Computer)와 통합 처리된 정보를 시현하는 다기능 시현기(Multi Function Display)를 중심으로 직렬 디지털 버스 구조를 채택한 연방형 시스템이 출현하게 되었다. 오늘날까지도 널리 적용되고 있는 표준 데이터 버스인 MIL-STD-1553과 같이 잘 정의된 인터페이스와 제작사들의 풍부한 개발 경험을 기초로 하여 만들어진 개별 장비들이 다양하게 공급되고, 서브시스템의 경계가 분명하여 시스템 설계 및 제작이 용이한 장점이 있으나, 개별 서브시스템 간에 일부 기능이 중복될 수 있기 때문에 전체 시스템의 비용 및 중량이 불가피하게 증가하는 단점이 있다.

1.3 Module 통합형 시스템

연방형 시스템은 기능 중복으로 인해 중량 및 전력, 공간 낭비가 발생하고 항공기를 개발하는 제작 업체 간 표준화가 이루어지지 않아 시스템의 유지보수 및 성능 개량에 어려움이 있다. 이에 미국과 유럽을 비롯한 항공 선진국에서는 항공전자의 연구 개발 사업을 통해 연방형 시스템보다 향상된 시스템을 모색하기 시작하였고, 그 결과 표준 모듈을 통합 관리하기 위해 표준 데이터 네트워크로 통신하는 이른바 모듈형 시스템이 출현하게 되었다. 모듈형 시스템을 통해 일반적으로 중량, 전력, 공간, 비용 등을 줄일 수 있을 것으로 판단되나 이를 위해서는 연방형 시스템에 존재하는 서브시스템간의 경계를 허물고 공통 여유 자원 풀을 활용하여 서브시스템 간에 공유함으로써 시스템의 가용도를 높이는 통합형 시스템으로 발전해야 한다. 통합형 시스템은 고장 허용 및 재구성을 통해 운용상의 융통성을 제공할 수 있는 부가적인 장점도 있으므로 군용 및 민간 항공기에서 이 개념을 바탕으로 한 시스템으로 발전하고 있다[7].

2. ARINC(Aeronautical Radio, Incorporated) 653

ARINC는 미국 소유의 비영리 단체이며 5개 분야(항공, 공항, 국방, 정부, 수송)를 주 업무 영역으로 두고 있다. 또한 최초로 경찰차와 철도에 컴퓨터 네트워크를 적용하였으며, LRU(Line Replaceable Units)에 대한 표준을 확립한 단체로 지상 기지국과 항공기 간의 통신 서비스 및 항공전자 표준 규격을 정의하고 있다. 이중 ARINC 653은 IMA 시스템을 위하여 코어 운영체제와 응용 간의 APEX(Application /Executive) 인터페이스에 대한 표준을 정의하고 있다. 또한 ARINC 653은 파티셔닝(Partitioning) 이라는 핵심적인 기능을 사용하여 각 응용에 대해 독점적인 자원 사용을 보장함으로써 각각의 응용 사이의 시간 및 공간적 격리를 제공한다. 이러한 방법으로 파티셔닝은 각 파티션에 대하여 공간 분할(Spatial Partitioning)과 시간 분할(Temporal Partitioning)이 이루어진다. 공간 분할은 각 파티션이 서로의 물리적 메모리 자원에 영향을 끼치지 못하는 것을 말하고, 시간 분할은 파티션에 할당된 시간자원 역시 다른 파티션에서 간섭할 수 없다는 것을 의미한다. 이러한 파티셔닝 개념은 항공전자 시스템과 같이 중요한 임무를 수행하는 환경에서 하나의 응용 fault가 전체 시스템에 악영향을 미치는 것을 방지하여 높은 신뢰성을 제공할 수 있다[8][9].

3. VxWorks 653

VxWorks 653은 Wind River사에서 개발한 실시간 운영체제로 ARINC 653 표준을 준수한다. VxWorks 653은 ARINC 653 APEX, VxWorks 및 POSIX Partition에 대한 동시 지원이 가능하고, HM 아키텍처를 제공함으로써 시스템 안정성을 향상시킨다. 또한 DO-297 IMA 공급자 역할 및 분리를 지원하고, ARINC 653 규격 준수 검증 및 DO-178B Qualified &

Verification Tool 제공으로 개발 및 인증 부분에 강점을 가지고 있다. [그림 1]은 ARINC 653 표준을 따르는 VxWorks 653에서 모듈과 파티션의 관계를 나타내는 구조도이다[10][11].

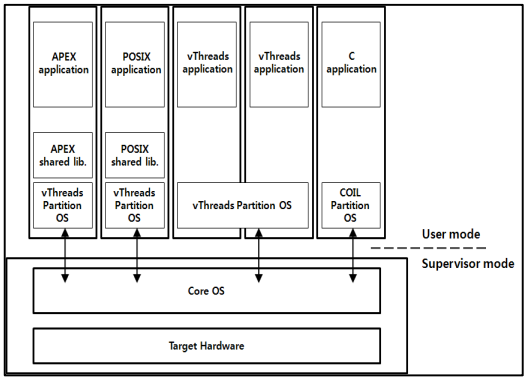


그림 1. VxWorks 653 구조도

VxWorks 653은 코어 OS가 여러 개의 파티션을 관리하고 각 파티션들은 파티션 OS를 통해 프로세스 응용을 관리한다. 현재 VxWorks 653은 Boeing/GE, KC-767, AirBus A400M 등 45종 이상의 항공기에서 180개 이상의 서브시스템에 전 세계적으로 사용되고 있다.

Ⅲ. IMA 시스템을 위한 HM 시스템 설계

IMA 시스템은 연방형 시스템과는 달리 모듈 통합에 의해 각 모듈들을 통합하여 관리하기 때문에 fault 발생 시 전체 시스템에 영향을 미치게 되어 정상적인 운용을 보장할 수 없게 된다. 이에 IMA 시스템은 고장 허용 정책을 적용하여 고장을 허용하고 시스템 재구성 및 롤백(roll-back)을 통한 복구 행동이 필요하다. ARINC 표준에 정의된 고장 허용 정책으로는 fault가 발생된 것을 알리는 fault detection, fault의 위치를 식별할 수 있는 fault isolation, 전체 시스템으로 fault의 영향을 방지하는 fault containment, fault를 복구하기 위한 fault recovery가 필요하다[12].

1. 시스템 상태

본 논문에서는 발생된 fault가 다른 영역에 영향을 미치는 것을 방지하기 위해 시스템에 존재하는 각 상태에 대해 정의하였다. ARINC 653 표준을 따르는 VxWorks 653 기반 IMA 시스템의 운용 시점에 따라 Module, Partition, Process 모드로 구분하였고, 각 모드에 따라 [표 1]에 정의된 시스템 상태를 가질 수 있다.

표 1. 시스템 상태

System Mode	System Status	Definitions
Module Mode	Module Init	Module 초기화 중인 상태
	Module HM	Module HM handler가 수행중인 상태
	module Func	ISR or Partition과 관계된 Task가 동작중인 상태
	Partition Switching	Partition 스케쥴링으로 인한 Partition간 스위치 중인 상태
	unknown system status	시스템의 상태를 알 수 없는 상태
Partition Mode	Partition Init	Partition이 초기화 중인 상태
	Partition HM	Partition HM handler가 수행중인 상태
	System Call	Partition과 관계된 Core OS Task가 수행중인 상태
Process Mode	Partition Process Management	Partition OS Kernel or Partition OS Interrupt 상태
	Process Execution	Partition OS Task가 수행중인 상태

2. Fault 정의

IMA 시스템에서 발생할 수 있는 fault는 하드웨어 fault에 의한 하드웨어 fault와 잘못된 소스 코드 및 메모리 접근 등에 의해 발생하는 소프트웨어 fault로 구분된다.

2.1 하드웨어 Fault

하드웨어 fault는 IMA 시스템의 모듈 구성에 따라 발생할 수 있는 fault가 결정되어 지고, BIT(Built In Test)를 통해 하드웨어의 fault 식별이 이루어진다. 하드웨어의 fault 식별은 모듈에 전원이 인가된 후 OS가 부팅되기 전 각 하드웨어의 정상 유무를 검사하기 위한 PBIT(Power-on Built In Test)와 운영자에 의해 하드

웨어의 정상 유무를 검사하는 IBIT (Initiated Built In Test), 시스템 운용 중 프로세스에서 지속적으로 검사를 수행하는 CBIT (Continuous Built In Test)를 통해 이루어진다[13].

본 논문에서는 하드웨어 fault의 경우 하드웨어의 명세 및 모듈 구성에 따라 발생할 수 있는 fault가 다르기 때문에 하드웨어 fault에 대해서는 다루지 않는다.

2.2 소프트웨어 Fault

소프트웨어 fault는 OS에서 발생될 수 있는 fault와 ARINC 표준에 정의된 프로세스 수행 중 발생될 수 있는 fault로 분류되어 진다. 소프트웨어 fault는 ARINC 653 표준을 따르는 VxWorks에서 Event Injection에 의해 fault 식별이 이루어진다. [표 2]는 ARINC 653 표준에 정의된 fault를 나타낸 표이다. ARINC 653 fault는 프로세스 및 OS 운용 도중 발생 될 수 있는 fault로 메모리 접근 오류나 연산 오류 등 시스템에서 발생될 수 있는 일반적인 fault를 정의하고 있다.

표 2. ARINC 653 Fault

	Fault Symbolic	발생원인
ARINC 653 Fault	Module Init Error	Module OS 초기화를 성공하지 못했을 때 발생
	Module Configuration Error	Module 초기화 과정 중 Module Configuration XML 파일을 정상적으로 불러오지 못했을 때 발생
	Partition Init Error	Partition OS 초기화를 성공하지 못했을 때 발생
	Partition Configuration Error	Partition OS 초기화 과정 중 Partition Configuration XML 파일을 정상적으로 불러오지 못했을 때 발생
	Segmentation Error	Out-of-Bound와 같은 메모리 참조 오류
	Protection Memory Violation	사용할 수 없는 메모리 영역을 Write 할 경우 발생
	Time Duration Exceeded	Process의 정해진 Deadline을 넘어설 경우 발생
	Invalid Os Call	잘못된 인자 값으로 함수를 호출한 경우 발생
	Divide by Zero	연산 과정 중 0으로 나눈 경우 발생
	Variable Range Overflow	변수의 정의된 범위보다 초과한 값을 저장한 경우 발생
	Floating Point Error	잘못된 Floating Point 연산 및 형 변환 시 발생
	Stack Overflow	과도한 배열을 할당한 경우 및 재귀함수를 잘못 쓴 경우 발생
	Parity Error	메모리 검사 시 데이터에 이상이 있을 경우 발생

[표 3]은 VxWorks 653에 정의된 fault를 나타낸 표이다. VxWorks 653 Defined Fault는 VxWorks 653 OS에 정의된 아키텍처 및 API를 위반한 경우 발생되는 fault로 HM Internal Error 및 코어 OS PORT Internal Error가 발생된 경우 등 VxWorks 653 OS 내부에서 발생될 수 있는 fault를 정의하고 있다.

표 3. VxWorks 653 Defined Fault

	Fault Symbolic	발생원인
VxWorks 653 Defined Fault	Can't Invoke VxWorks 653 OS	Module 운영 중 Exception 발생 시 정의되어 있지 않은 Exception이 발생된 경우
	Configuration Error in the HM	HM Handler가 Configuration XML 파일을 참조할 때 XML 파일이 잘못된 경우 발생
	over time limit	Partition의 정해진 Deadline을 넘어서 실행되는 경우 발생
	Partition Mode Change	Partition Switching 도중 Fault가 발생된 경우
	APEX Internal Error Code	VxWorks 653에서 제공하는 APEX API 수행 도중 Fault를 반환한 경우 발생
	HM Internal Error Code	Event Injection에 의해 HM Handler가 수행 도중 Fault를 반환한 경우 발생
	Core OS PORT Internal Error Code	Partition간 통신을 위해 PORT를 사용 도중 Fault가 발생된 경우
	System Clock Ticks were Lost	전체 시스템 Clock인 Tick 값이 잘못 되었거나, Partition Clock인 Pseudo Tick이 잘못된 경우 발생
	HM Event Processing Error	HM_EVENT 처리 도중 Fault가 발생된 경우
	HM Queue overflow	HM Queue에 선언된 Threshold 값을 넘어설 경우 발생
	HM Could not execute within the required time constraint	Event가 Injection되어 Module HM 수행 시 HM의 Deadline을 넘긴 경우 발생

3. Fault Tolerance를 위한 HM 시스템 설계

3.1 HM 시스템 동작 알고리즘

ARINC 653 기반의 VxWorks 653은 fault 발생 시 고장 허용을 위해 HM 아키텍처를 제공한다. [그림 2]는 시스템 운용 도중 고장 허용을 위한 HM 아키텍처를 보여준다.

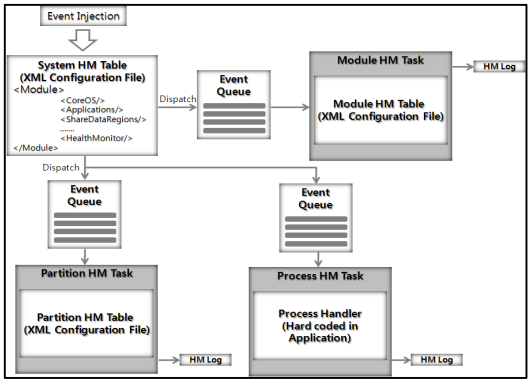


그림 2. HM Architecture

VxWorks 653에서 fault가 발생되면 HM 아키텍처에 의해 이벤트가 주입되고, System HM 테이블에서 각 이벤트의 종류에 따라 처리해야 할 Level (Module, Partition, Process Level)을 결정하여 해당 Level의 Event Queue에 Dispatch 된다. Module Event Queue에 Dispatch된 이벤트에 대해 Module HM Handler를 호출하여 Module HM Table에 정의된 복구 행동에 의해 처리된다. 마찬가지로 Partition Event Queue에 Dispatch된 이벤트에 대해 Partition HM Handler를 호출하여 Partition HM Table에 정의된 복구 행동에 의해 처리된다. Process Event Queue에 Dispatch된 이벤트는 Module, Partition에서 발생된 fault와 다르게 응용 내의 hard coded된 코드를 호출하여 처리하게 된다. Module, Partition, Process에서 발생된 각 fault에 대해 HM 시스템은 로그를 저장하게 된다.

3.2 System HM 테이블

System HM 테이블은 각 fault 별 시스템 상태에 따른 처리 Level을 정의한다. Module Mode에서 발생된 fault는 Module Level에서 처리될 수 있고, Partition Mode에서 발생된 fault의 특성에 따라 Partition Level에서 처리할 수 있다면 Partition Level에서 처리하고, Partition Level에서 처리 할 수 없는 경우 상위 Level인 Module Level에서 처리해야 한다. Process Level에서 발생된 fault의 경우 Process Error Handler가 등록하여 처리하게 된다. Process Error Handler가 등록 되지 않은 경우 Partition Level에서 처리하게 되고, Partition Level에서 처리하지 못할 경우 Module Level에서 처리해야 한다. [그림 3]은 본 논문에서 설계된 System HM 테이블로 각 시스템 상태별 fault의 처리 레벨을 정의한 그림이다.

		module Mode					Partition Mode			Process Mode	
		module init	module HM	module function	partition switching	unknown system status	partition init	partition HM	system call	partition process management	process execution
	Error ID	State 1	State 2	State 3	State 4	State 5	State 6	State 7	State 8	State 9	State 10
Module Init Error	1	ML									
Module Configuration Error	2	ML									
Partition Init Error	3						PL				
Partition Configuration Error	4						PL				
Segmentation Error	5	ML	ML	ML	ML		PL	PL	PL	PL	Process Error Handler
Protection Memory Violation	6	ML	ML	ML	ML		PL	PL	PL	PL	Process Error Handler
Time duration exceeded	7		ML	ML	ML			PL	PL	PL	Process Error Handler
Invalid OS Call	8										Process Error Handler
Divide by Zero	9	ML	ML	ML	ML		PL	PL	PL	PL	Process Error Handler
Variable range overflow	10	ML	ML	ML	ML		PL	PL	PL	PL	Process Error Handler
Floating Point Error	11	ML	ML	ML	ML		PL	PL	PL	PL	Process Error Handler
Stack overflow	12										Process Error Handler
Parity Error	13			ML							
can't invoke VxWorks653 OS	14	ML	ML	ML	ML		ML	ML	ML	ML	ML
configuration Error in the HM	15		ML					ML		ML	
over time limit	16		ML		ML						
partition mode change	17				ML					PL	
APEX internal error code	18										PL
HM internal error code	19		ML					PL		PL	
Core OS port internal error code	20										PL
system clock ticks were lost	21	ML	ML	ML	ML		ML	ML	ML	ML	ML
HM event processing error	22		ML					ML		PL	
HM queue overflow	23		ML					ML		PL	
data loss after reformatting event	24		ML					PL		PL	
HM could not excute within the required time constrain	25		ML					ML		PL	

그림 3. System HM Table

3.3 Module HM 테이블

Module HM 테이블은 System HM 테이블로부터 Dispatch된 이벤트에 대한 처리 행동을 정의하고 있는 테이블이다. [표 4]는 Module Level에서 처리되는 행동을 정의하였다.

표 4. Module Actions

Module Actions	Definitions
SH(Shutdowon)	Module 작동 중지
RS(Reset)	Module의 재실행
IG(Ignore)	발생된 Fault를 무시함
UEH(User Error Handler)	특정 복구 행동을 할 수 있도록 사용자 핸들러 등록

이와 같은 처리 행동으로 Module Level에서 발생된 fault에 대해 복구가 가능하다. 그러나 SH의 경우 반드시 여분 모듈로의 교체가 이루어져야 시스템이 정상적으로 동작할 수 있다. 또한 fault 발생 시 시스템에 영향을 끼치지 않는 fault의 경우 IG 처리 행동을 통해 시스템의 유연성을 높일 수 있다. [그림 4]는 Module HM 테이블로 각 시스템 상태별 fault 처리 행동을 정의한 그림이다.

3.4 Partition HM 테이블

Partition HM 테이블은 System HM 테이블로부터 Dispatch된 이벤트에 대한 처리 행동을 정의하고 있다. [표 5]는 Partition Level에서 처리되는 행동을 정의하였다.

표 5. Partition Actions

Module Actions	Definitions
ID(Idle)	Partition을 유휴 상태로 변경
CS(Cold Start)	데이터 초기화를 통한 파티션 재 시작
WS(Warm Start)	기존의 데이터를 초기화 하지 않고 특정 지점부터 시작
IG(Ignore)	발생된 Fault를 무시함
UEH(User Error Handler)	특정 복구 행동을 할 수 있도록 사용자 핸들러 등록

[표 5]에 나타난 처리 행동으로 Partition Level에서 발생된 fault에 대해 복구가 가능하다. 그러나 ID의 경우 유휴 상태에 빠진 파티션의 임무를 수행하기 위하여 동일한 작업을 수행하는 파티션으로의 교체가 이루어져야 시스템이 정상적으로 동작할 수 있다. [그림 5]는 Partition HM 테이블로 각 시스템 상태 별 fault 처리 행동을 정의한 그림이다.

		module Mode					Partition Mode			Process Mode	
		module init	module HM	module function	partition switching	unknown system status	partition init	partition HM	system call	partition process management	process execution
	Error ID	State 1	State 2	State 3	State 4	State 5	State 6	State 7	State 8	State 9	State 10
Module Init Error	1	SH									
Module Configuration Error	2	SH									
Partition Init Error	3										
Partition Configuration Error	4										
Segmentation Error	5	SH	SH	SH	SH						
Protection Memory Violation	6	SH	SH	SH	SH						
Time duration exceeded	7		UEH	UEH	UEH						
Invalid OS Call	8										
Divide by Zero	9	IG	IG	IG	IG						
Variable range overflow	10	IG	IG	IG	IG						
Floating Point Error	11	IG	IG	IG	IG						
Stack overflow	12										
Parity Error	13			SH							
can't invoke VxWorks653 OS	14	RS	UEH	UEH	UEH		UEH	UEH	UEH	UEH	UEH
configuration Error in the HM	15		SH					SH		SH	
over time limit	16		RS		RS						
partition mode change	17				RS						
APEX internal error code	18										
HM internal error code	19		RS								
Core OS port internal error code	20										
system clock ticks were lost	21	RS	RS	RS	RS		RS	RS	RS	RS	RS
HM event processing error	22		UEH					UEH			
HM queue overflow	23		UEH					UEH			
data loss after reformatting event	24		RS								
HM could not excute within the required time constrain	25		UEH					UEH			

Module Actions
SH = Shutdown
RS = Reset
IG = Ignore
UEH = User Error Handler

그림 4. Module HM Table

		module Mode					Partition Mode			Process Mode	
		module init	module HM	module function	partition switching	unknown system status	partition init	partition HM	system call	partition process management	process execution
	Error ID	State 1	State 2	State 3	State 4	State 5	State 6	State 7	State 8	State 9	State 10
Module Init Error	1										
Module Configuration Error	2										
Partition Init Error	3						ID				
Partition Configuration Error	4						ID				
Segmentation Error	5						ID	ID	ID	ID	
Protection Memory Violation	6						ID	ID	ID	ID	
Time duration exceeded	7							UEH	UEH	UEH	
Invalid OS Call	8										
Divide by Zero	9						IG	IG	CS	IG	
Variable range overflow	10						IG	IG	CS	IG	
Floating Point Error	11						IG	IG	CS	IG	
Stack overflow	12										
Parity Error	13										
can't invoke VxWorks653 OS	14										
configuration Error in the HM	15										
over time limit	16										
partition mode change	17									CS	
APEX internal error code	18										UEH
HM internal error code	19							CS		CS	
Core OS port internal error code	20										WS
system clock ticks were lost	21										
HM event processing error	22									UEH	
HM queue overflow	23									UEH	
data loss after reformatting event	24							CS		CS	
HM could not excute within the required time constrain	25									UEH	

Partition Actions
ID = Idle
CS = Cold Start
WS = Warm Start
IG = Ignore
UEH = User Error Handler

그림 5. Partition HM Table

4. 모의 시나리오

본 절에서는 시스템의 상태에 따라 Fault 발생 시 HM 아키텍처에 의해 처리되는 모의 시나리오를 작성하였다. 시나리오로는 Partition Switching 도중 발생하는 Segmentation Error에 대한 처리 흐름을 나타내었다.

Segmentation Error는 배열의 끝을 넘어섰을 경우나 out-of-bound와 같은 메모리 참조 오류로 인해 발생할 수 있는 fault로 시스템 모든 상태에서 발생될 수 있다. [그림 6]은 Partition Switching 도중 Segmentation Error가 발생하여 [그림 2]의 HM 아키텍처를 통해 fault을 처리하는 과정을 보여준다.

먼저 fault가 발생되면 시스템은 Fault Detection을 통해 발생된 fault 정보를 가지고 이벤트를 주입하게 된다. 이벤트가 주입되면 System HM Table을 통해 Segmentation Error는 Module Level로 처리하기 위해 Module Event Queue로 dispatch된다. 다음으로 Event Queue에 Dispatch된 이벤트는 Module HM Handler를 호출하여 fault 처리 행동을 하게 된다. 정의된 복구 행동인 Shutdown 정책을 수행함으로써 fault가 발생된 모듈을 종료하고 모듈에서 수행하고 있던 기존 항공기 임무들을 여분의 공통 모듈로 로딩 하여 시스템이 정상적으로 동작할 수 있게 해준다.

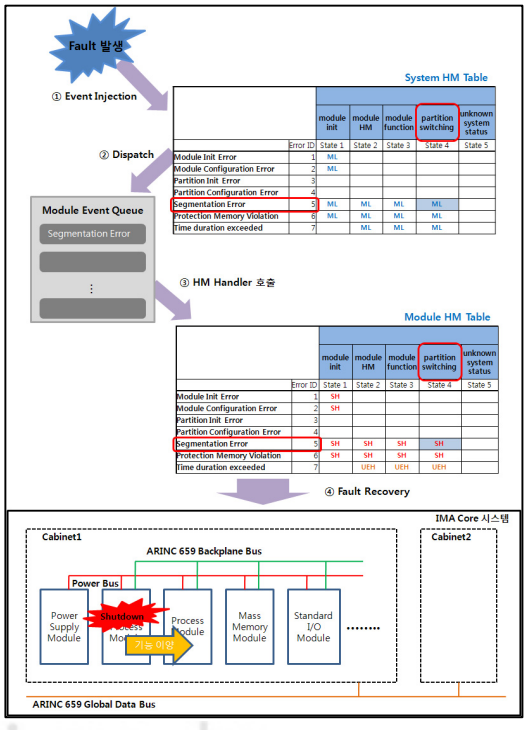


그림 6. Segmentation Fault Recovery 시나리오

Ⅳ. 결론

최근 항공전자 시스템은 연방형 시스템에서 IMA 시스템으로 발전함에 따라 모듈 통합형 설계가 가능하게 되었다. 모듈 통합화 설계에 따라 개발 비용, 유지보수 비용의 감소 및 항공기의 무게 감소 등의 효과를 얻게 되었고 세부 임무를 수행하는 기능 단위별 서브시스템 간에 여분 모듈을 공유함으로써 시스템 가용도가 증가하게 되었다. 하지만, IMA 시스템은 각 모듈에 대해 통합 처리가 이루어지기 때문에 fault 발생 시 fault의 영향이 쉽게 전파될 수 있어, 전체 통합 시스템에 문제가 발생하게 되는 문제가 발생되고, 또한 시스템의 복잡성이 증가됨에 따라 고장 허용 기술의 설계가 어려워지는 문제가 발생한다.

이러한 문제점을 해결하기 위해 본 논문에서는 IMA 시스템에서의 고장 허용을 위한 HM 시스템을 설계하였다. HM 시스템에서는 시스템 상태로 Module, Partition, Application 상태를 분류하여 상위 레벨에서 발생된 fault가 하위 레벨로 전파되는 것을 방지하고, 시스템 상태 별 각 fault를 구성 테이블에 정의함으로써 보다 쉽게 고장 허용 기술을 설계하고, 유지 보수를 가능하게 한다. 이를 통해 IMA 시스템이 탑재된 항공기의 신뢰성 및 안정성을 향상 시킬 수 있다.

향후 연구 과제로는 본 논문에서 설계한 HM 시스템이 적용된 IMA 시스템 항공기를 구축하고 테스트를 통해 시스템의 신뢰성 및 안정성에 대해 검증이 필요하다.

참 고 문 헌

[1] 한상호, “항공전자 부품의 인증동향”, 항공우주산업기술동향, 제8권, 제1호, pp.131-139, 2010.

[2] 한상호, “항공전자기술의 분류”, 항공우주기술, 제1권, 제1호, pp.188-200. 2002.

[3] 홍승범, 지민석, 홍교영, 김영인, “항공전자통합구조를 위한 광통신 데이터 버스의 연구”, 한국콘텐츠학회, 2009.

[4] 고진영, 박한준, 손진규, “모듈 통합형 항공전자 기술의 개발 동향 연구”, 한국항공우주학회, pp.1749-1753, 2011.

[5] Airlines Electronic Engineering Committee, *"DESIGN GUIDANCE FOR INTEGRATED MODULAR AVIONICS"*, ARINC Report 651, ARINC, 1997.

[6] 서준호, 박한준, 손진규, “모듈 통합형 항공전자 시스템 Fault Tolerance 설계 연구”, 한국항공우주학회, pp.1710-1714, 2011.

[7] 송찬호, “항공전자 기술 발전 동향” IT SOC magazine, 제34권, pp.24-31, 2009.

[8] Airlines Electronic Engineering Committee, *"AVIONICS APPLICATION SOFTWARE STANDARD INTERFACE"*, ARINC Specification 653, ARINC, 2010.

[9] A. Wilson, "The evolving ARINC 653 standard and it's application to IMA," WindRiver, 2007.

[10] Wind River, *"Wind River VxWorks 653 Platform 2.3"*, pp.1-10, 2010.

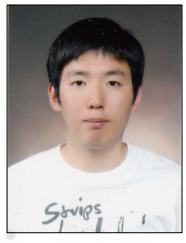
[11] Wind River, *"VxWorks 653 PROGRAMMER's GUIDE 2.3"*, pp.1-134., 2010.

[12] K. A. Seeling, "Reconfiguration in an Integrated Avionics Design," Digital Avionics Systems Conference, 1996.

[13] D. R. Benizuez and D. C. Witteried "Built-In-Test Adequacy-Evaluation Methodology for an Air-Vehicle System," Proceedings Annual Reliability and Maintainability Sysposium, 1995.

저 자 소 개

고 영 관(Young-Kwan Ko) 준회원



- 2011년 8월 : 충남대학교 컴퓨터공학과(공학사)
- 2011년 8월 ~ 현재 : 충남대학교 컴퓨터공학과(공학석사과정)

<관심분야> : 실시간 시스템, IMA 시스템, Fault Tolerance,

이 승 훈(Seung-Hoon Lee) 준회원

- 2011년 2월 : 충남대학교 컴퓨터공학과(공학사)
- 2011년 2월 ~ 현재 : 충남대학교 컴퓨터공학과(공학석사과정)

<관심분야> : IMA 시스템, 임베디드 시스템, 운영체제

박 세 영(Se-Young Park) 준회원

- 2010년 2월 : 충남대학교 컴퓨터공학과(공학사)
- 2010년 2월 ~ 현재 : 충남대학교 컴퓨터공학과(공학석사과정)

<관심분야> : 실시간 시스템, 임베디드 시스템, 운영체제

반 창 봉(Chang-Bong Ban) 정회원

- 1996년 2월 : 중앙대학교 전기전자제어공학부(공학사)
- 2000년 2월 : 중앙대학교 전자전기공학부(공학석사)
- 2004년 2월 ~ 현재 : LIG넥스원 항공연구센터 수석연구원

<관심분야> : IMA 시스템, 항공전자 시스템, 임무컴퓨터

강 대 일(Dai-Il Kang) 준회원

- 1993년 2월 : 창원대학교 전자공학과(공학사)
- 1997년 2월 : 창원대학교 전기전자제어공학과(공학석사)
- 2011년 9월 ~ 현재 : LIG넥스원 항공연구센터 선임연구원

<관심분야> : IMA 시스템, 실시간 운영체제, 멀티코어

정 재 엽(Jae-Yeop Jeong) 정회원

- 2000년 2월 : 충남대학교 컴퓨터공학부(공학사)
- 2007년 2월 : 충남대학교 컴퓨터공학과(공학석사)
- 2009년 1월 ~ 현재 : LIG넥스원 항공연구센터 선임연구원

<관심분야> : IMA 시스템, 실시간 운영체제, 임무컴퓨터

이 철 훈(Cheol-Hoon Lee) 정회원

- 1983년 2월 : 서울대학교 전자공학과(공학사)
- 1988년 2월 : 한국과학기술원 전기 및 전자공학과(공학석사)
- 1992년 2월 : 한국과학기술원 전기 및 전자공학과(공학박사)
- 1983년 3월 ~ 1983년 2월 : 삼성전자 컴퓨터사업부 연구원
- 1992년 3월 ~ 1994년 2월 : 삼성전자 컴퓨터사업부 선임 연구원
- 1994년 2월 ~ 1995년 2월 : Univ. of Michigan 객원 연구원
- 1995년 2월 ~ 현재 : 충남대학교 컴퓨터 공학과 교수
- 2004년 2월 ~ 2005년 2월 : Univ. of Michigan 초빙 연구원

<관심분야> : 실시간시스템, 운영체제, 컴퓨터 구조, Fault Tolerance 컴퓨팅, 병렬 컴퓨팅, IMA 시스템