論文 J. of The Korean Society for Aeronautical and Space Sciences 44(3), 256-265(2016)
DOI:http://dx.doi.org/10.5139/JKSAS.2016.44.3.256
ISSN 1225-1348(print), 2287-6871(online)

비행조종컴퓨터 소프트웨어 기반 고장허용 설계 기법 연구

윤형식*, 김연균

A Study on Software Based Fault-Tolerance Techniques for Flight Control Computer

Hyung-Sik Yoon* and Yeon-Gyun Kim
Agency for Defense Development

ABSTRACT

Software based fault tolerance techniques are designed to allow a system to tolerate software faults in the system. Fault tolerance techniques are divided into two groups : software based fault tolerance techniques and hardware based fault tolerance techniques. We need a proper design method according to characteristics of the system. In this paper, the concepts of software based fault tolerance techniques for Dual Flight Control Computer are described. For software based fault tolerance design, we classified software failure, designed a way for failure detection and the way of recovery. Eventually the effectiveness of software based fault tolerance techniques was verified through the Software Test Environment(STE).

초 록

소프트웨어 기반의 고장허용이란 장비의 일부분에 소프트웨어 고장이 발생하더라도 허용할 수 있도록 장비를 설계하는 것을 의미한다. 고장허용을 위한 설계 방법은 크게 하드웨어 기반 고장허용 설계 방법과 소프트웨어 기반 고장허용 설계 방법이 있으며, 시스템의 특징에 따라 적절한 방법의 고장허용 설계 방법 선택이 필요하다. 본 논문에서는 하드웨어적으로 이중화로 구성된 비행조종컴퓨터의 소프트웨어 기반 고장허용 설계 기법에 대하여 기술하였다. 소프트웨어 기반의 고장허용 설계를 위하여 소프트웨어 고장을 분류하고, 고장에 대한 검출 방법을 설계한 후, 고장발생시 복구 방법을 설계하였다. 설계된 방법의 유효성을 확인하기 위하여 전용 소프트웨어 시험 환경을 통해 설계된 소프트웨어 기반 고장허용 설계의 타당성을 검증하였다.

Key Words : Software fault tolerance techniques(소프트웨어 고장허용 기법), dual Flight control computer(이중화 비행조종컴퓨터), fault detection(고장 검출), Fault Recovery(고장 복구)

† Received : September 10, 2015 Revised : February 23, 2016 Accepted : February 25, 2016
* Corresponding author, E-mail : yjw0210@empas.com

Ⅰ. 서 론

항공기에 탑재되어 비행조종시스템의 다양한 센서들로부터 입력 신호를 이용하여 항공기의 조종면을 제어하는 출력신호를 생성하는 기능을 수행하는 비행조종컴퓨터의 비행 중 고장 발생은 항공기의 손실에 직접적인 영향을 준다. 이러한 항공기의 비행 안전에 필수적인 비행조종컴퓨터는 고 신뢰성 및 가용성에 대한 요구도를 만족하기 위하여 장비의 설계시 다양한 기법이 적용되고 있으며, 고장허용 설계 방법도 그 중 하나이다[1]. 고장허용이란 장비의 일부분에 고장이 발생하더라도 그 영향이 장비의 고장으로 발전하는 것을 막아 주어진 임무를 지속적으로 수행할 수 있도록 설계하는 방법이다. 고장허용을 위한 설계 방법은 크게 하드웨어 기반 고장허용 설계 방법과 소프트웨어 기반 고장허용 설계 방법이 있으며, 시스템의 특징과 요구조건에 따라 적절한 방법의 고장허용 설계 방법을 선택하는 것은 매우 중요하다. 시스템 개발시에 적절한 고장허용 설계 기법을 적용하기 위한 일반적인 수행 절차는 다음과 같다. 첫 번째 단계에서는 해당 장비의 운용 수명동안 예상되는 치명적인 고장의 원인을 분류하고, 장비의 신뢰성 요구도 등을 고려하여 고장허용 설계를 적용할 범위를 결정한다. 고장허용 설계 범위가 결정되면 두 번째 단계에서는 고장 원인별로 고장검출을 위한 고장 진단 알고리즘을 설계한다. 그리고 세 번째 단계에서는 고장 검출 후에도 장비를 계속 정상운용이 가능한 안전한 상태로 전환하기 위하여 검출된 고장에 따라 적절한 고장 복구 설계를 수행한다. 네 번째 단계에서는 설계된 고장허용 설계의 타당성 검증을 위해서 고장 주입 시험을 이용하여 평가한다. 본 논문에서는 먼저 하드웨어 및 소프트웨어 기반의 고장허용 설계 개념을 살펴본다[2-7]. 그리고 하드웨어적으로 이중화로 구성된 비행조종컴퓨터에 적용한 단일 버전 소프트웨어 기반의 소프트웨어 고장허용 설계에 대한 내용을 기술하였다.

Ⅱ. 본 론

2.1 하드웨어 기반 고장허용 설계

고장허용 설계를 위한 가장 일반적이고 오래된 개념은 물리적 하드웨어 여분을 사용한 하드웨어 기반 다중화 기법이다[8-10]. 물리적으로 동일한 여분의 모듈을 중복해서 사용하는 하드웨어 기반 고장허용 설계는 구현 방식에 따라 수동방식(Passive), 능동방식(Active) 그리고 혼용방식(Hybrid)으로 구분한다. 수동방식 고장허용 설계는 장비의 일부에 발생된 결함을 단순히 마스킹하여 결함이 장비의 고장으로 전파되는 것을 방지하는 방법으로 보팅 알고리즘을 기본으로 사용한다. 보팅 알고리즘은 일반적으로 다수결 보팅 방법을 사용하며, 이를 위해서 3중화 이상의 하드웨어 여분을 요구한다. 능동방식 고장허용 설계는 장비의 일부에 발생된 결함이 검출되면 이를 배제함으로서 장비의 고장으로 전파되는 것을 방지하는 방법으로 자체 진단 시험을 기반으로하여 결함 발생 모듈을 검출하고 정상적인 모듈을 선택함으로서 결함 회복을 수행한다. 혼용방식 고장허용 설계는 수동방식과 능동방식의 장점을 조합한 방식으로 기본적으로 수동방식으로 결함을 탐지하고 결함 확인시 능동방식의 결함 검출, 배제 및 회복 단계를 수행한다. 그러나 하드웨어적인 다중화 설계를 이용한 고장허용 설계는 설계가 쉬운 장점이 있으나, 하드웨어 다중화로 인한 시스템의 크기, 무게 및 비용이 증가하는 단점이 있어 비행조종컴퓨터 개발 시 적용에 제한이 있다.

2.2 소프트웨어 기반 고장허용 설계

장비 운용 중 발생하는 소프트웨어 고장은 하드웨어 고장과는 달리 운용 시간에 따라 성능 저하에 의한 고장은 발생하지 않지만 입력 등에 의한 외부 연동 환경의 영향으로 인해 발생되는 오류이다. 소프트웨어 고장의 주요 원인으로는 부정확한 초기 요구도, 설계 단계에서의 오류 및 코딩 오류 등이 있으며, 이러한 소프트웨어 고장으로 인해 발생되는 고장허용을 위한 소프트웨어 기반 고장허용 설계 기술은 구현 방식에 따라 다중 버전 소프트웨어 기반의 고장허용 설계 및 단일 버전 소프트웨어 기반의 고장허용 설계 기술로 구분한다[2-7]. 다중 버전 소프트웨어 기반의 고장허용 설계 기술은 동일한 요구도를 구현하기 위하여 다양한 버전의 소프트웨어를 독립적으로 개발하여 동일한 소프트웨어 설계 오류에 의한 고장을 방지하는 방법으로 N Version Programing, Recovery Blocks, N Self Checking Programming 및 Consensus Recovery Blocks 등이 있으며, 이는 하드웨어 기반의 고장허용 설계 기법과 유사하다. 다중 버전 소프트웨어 기반의 고장허용 설계 기술은 소프트웨어 설계 오류를 검출하고 정상 출력을 사용할 확률이 증가하는 장점이 있으나, 개발 기간 및 비용 증가와 다양한 버전 소프트웨어 개발로 인한 설계 오류가 추

가로 발생할 수 있는 단점이 있다. 단일 버전 소프트웨어 기반의 고장허용 설계 기술은 하나의 버전으로 구현된 소프트웨어에서 다양한 진단 방법을 이용해서 고장을 검출하여 고장 영향성을 없애고 시스템을 안전한 상태로 유지하는 고장 복구 설계 기법으로, 개발기간 및 비용 등을 고려하여 많이 사용되는 고장허용 설계 기법이다[3]. 본 논문에서는 단일 버전 소프트웨어 기반의 고장허용 설계 기술에 대하여 기술한다.

2.3 고장 분류

장비에 존재하는 다양한 고장들에 대하여 고장허용 설계를 위한 첫 번째 단계는 발생 가능한 고장들에 대한 유형 분석이다. 장비의 고장을 발생시키는 원인들은 하드웨어 고장, 외부 연동장비에 의한 고장 및 소프트웨어 설계 오류 등이 있으며, 일반적으로 고장 모드 영향 및 치명도 분석(FMECA, Failure Modes, Effects and Criticality Analysis)을 통해 장비의 고장 분석을 수행한다. 장비에 존재하는 일반적인 고장의 분류는 Table 1과 같다[3]. 무인항공기 탑재용 이중화 비행조종컴퓨터는 FMECA를 통해 분류된 고장 항목들에 대하여 고장허용 설계 범위를 결정하기 위하여 장비의 신뢰성 요구도, 개발 비용, 개발 기간 및 CPU 부하(run time overhead)등을 고려하여 설계하였다. 본 논문에서 무인항공기 탑재용 이중화 비행조종컴퓨터의 고장허용 설계를 위하여 분류한 고장은 장비의 내부 원인과 외부 원인으로 구분하였으며, 각 세부 고장 내용은 Table 2와 같다.

Table 1. Fault Classification

분류	고장
Activity	Latent/Active
Duration	Transient/Permanent
Perception	Symmetric/Asymmetric
Cause	Random/Generic
Intent	Benign/Malicious
Count	Single/Multiple
Time (multiple faults)	Coincient/Distinct
Cause (multiple faults)	Independent/ Common mode

Table 2. Selected Fault Types

고장 분류		고장 유형
장비 외부 원인	외부 장비 고장	비유효 입력: range, rate of change, 부동 소수점 오류
		요구도 미 정의 입력 오류
	연동 통신 고장	신호 전달 오류
		통신 두절에 의한 연동 오류
장비 내부 원인	하드웨어 고장	CPU 모듈 고장
		메모리 모듈 고장
		통신 소자 모듈 고장
	소프트웨어 설계 오류	코드 설계 오류 : overflow, underflow, divide by zero
		제어 흐름 오류
		Memory overhead
		Timing 오류
		동기화 오류

2.4 고장 검출 설계

소프트웨어 설계 오류에 의한 고장을 검출하기 위하여 고장 유형에 따라 적절한 고장검출 설계 기법이 적용되고 있다. 일반적인 단일 버전의 소프트웨어에서 고장 검출을 위한 방법은 다음과 같다.

(1) 모니터 삽입 방법 : 소프트웨어 코드에 고장 모니터 기능을 위한 논리적 구문을 추가로 삽입하여 소프트웨어 운용시 주기적으로 상태를 확인하면서 소프트웨어 내부 고장 여부를 판단하는 방법

(2) 제어 흐름 확인(control flow check) : 소프트웨어 코드의 각 노드에 미리 정의된 값인 deterministic signature를 삽입하고, 운용중 계산된 run time signature와 비교하여 고장 여부를 판단하는 방법

(3) 중복 확인(replication check) : 소프트웨어의 운용 로직에서 계산된 출력신호와 여분의 모듈에서 계산된 출력신호에 대하여 상호 비교를 통해 고장 여부를 판단하는 방법.

(4) 시간성 확인(timing check) : 실시간 시스템 소프트웨어 운용을 위한 시간적 제한 요구도를 판단하기 위한 방법으로 watchdog timer 등의 방법을 이용

(5) 반전 확인(reversal check) : 소프트웨어로부터 계산된 출력 값을 이용하여 입력을 역으로 계산하고 실제 입력값과 비교하여 고장 여부를 판단하는 방법

(6) 코딩 확인(Coding Check) : 신호 전송 오류를 판단하기위한 방법으로 checksum, crc 등의 방법을 이용

(7) 합리성 확인(reasonableness check) : 입력 데이터의 범위, 변화율, 순차성 등의 오류 여부를

확인하기 위하여 초기에 정의된 제한값을 이용하여 설계범위 위반 여부를 판단하는 방법

(8) 데이터 구조 확인(Data structural check) : 버퍼 overflow/underflow, divide by zero 등의 소프트웨어 설계오류에 의한 고장 여부를 판단하기 위하여 설계된 데이터의 구조를 확인하는 방법

2.5 고장 복구 설계

고장 복구는 장비의 고장을 검출한 이후에 중단없이 정상 운용을 지속적으로 수행하기 위하여 빠르게 장비를 다시 안전한 상태로 전환하는 것을 의미한다. 장비의 일부에 고장 발생시 단순한 고장 복구 방법으로는 장비의 전원 재인가를 통해 이전 상태로 리셋한 후 재수행 하는 방법, 고장을 무시하고 계속 운용을 하는 방법 및 오류 발생 부분을 반복 수행하는 방법들이 있다. 그러나 비행조종컴퓨터는 비행 중 장비를 재수행하거나 오류 부분을 반복적으로 수행할 경우 장비의 운용에 있어서 실시간 요구도를 만족하지 못해 항공기 운용에 치명적인 위험이 발생할 수 있다. 따라서 비행조종컴퓨터는 고장 검출시 재 수행 없이 빠르게 안전한 상태로 복구하기 위한 설계 방법이 요구된다. 고장 복구 설계의 기본 개념은 Fig. 1의 체크 포인팅과 롤백(checkpointing and Rollback) 방법으로, 시스템의 정상 운용 중 안전한 상태를 계속 확인하고 저장하면서 고장 발생 시에도 장비를 그 위치에서 재수행할 수 있도록 한다[5]. 그러나 고장복구 로직을 실행하기 위해서는 실시간으로 고장 발생시 복구를 위한 위치를 확인하고 고장이 발견되는 순간 복구 위치에서 시스템을 재수행하기 위한 소프트웨어 실행 시간의 과부하를 고려하여야 한다. 고장 복구 설계를 위한 소프트웨어 고장복구 방법을 이용한 상태복구 방법은 다음과 같다.

(1) 백워드(backward) : 시스템을 고장이 발생하기 이전의 정상상태로 되돌리는 방법으로 주기적으로 checkpoint를 사용하여 정상적인 상태를 모니터링하면서 고장 발생시 이전 정상상태로 시스템을 복구하는 방법

(2) 포워드(forward) : 시스템에 고장 발생시 이전 상태를 무시하고 정상적인 다음 상태를 판단하여 시스템을 다음 상태에서 재수행 하는 방법

또한 이중화 장비의 경우 일시적인 고장과 영구적 고장에 따라 여분의 정상 채널 데이터를 이용하는 하드웨어 고장복구 방법을 이용한 데이터를 복구하는 방법은 다음과 같다.

(1) 일시적 고장 : 채널에 일시적 고장이 발생시 CCDL을 통해 채널간 상호 교환되는 정상적인 채널로부터 수신 데이터를 이용하여 고장발생 채널을 정상적으로 수행하는 방법

(2) 영구적 고장 : 채널에 발생한 고장이 일정 시간 이상으로 유지되는 경우 영구적 고장으로 판단하고, 정상적인 상대 채널에 영향을 미치거나 외부 장치에 영향을 미치는것을 방지하기 위하여 고장 채널에서 하드웨어적으로 외부 입출력 신호를 모두 단절시키며, 외부 시스템은 이를 통해 채널의 고장 여부를 판단하는 방법

2.6 이중화 비행조종컴퓨터 소프트웨어 고장허용 설계 기술

본 논문에서는 시스템의 신뢰성 요구도를 만족하면서 개발비용, 기간 및 항공기의 무게 등 다양한 요구조건을 고려하여 최소한의 하드웨어 다중화 수준인 이중화 구조의 비행조종컴퓨터 고장허용 설계 기법에 대한 내용을 기술하였다. Fig. 2는 이중화 구조의 비행조종컴퓨터의 기본 구조로서, 물리적으로 분리된 동일한 이중 채널로 구성된 비행조종컴퓨터의 각 채널은 프로세스 모듈, 입출력 모듈 및 전원 모듈로 구성되어 시리얼 통신인 CCDL을 통해 채널간 상태 및 입출력 데이터를 서로 교환한다. 이중화 비행조종컴퓨터에 탑재를 위한 비행운용 프로그램은 개발 비용

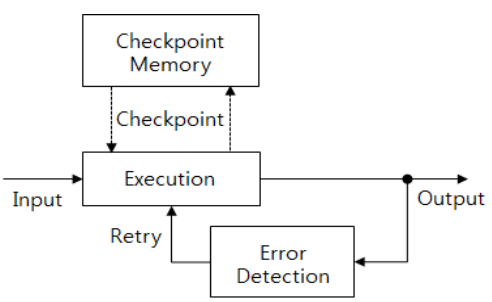


Fig. 1. Logical Representation of Checkpoint and Restart

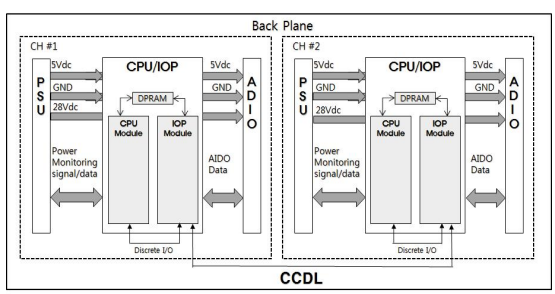


Fig. 2. Structure of Dual Flight Control Computer

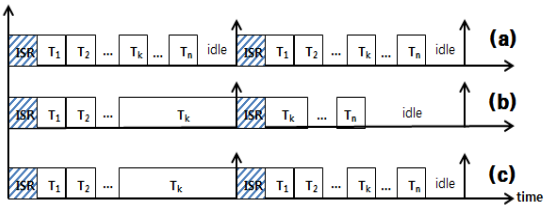


Fig. 3. Frame Overrun Concept

및 기간을 고려하여 단일 버전의 비행운용 프로그램을 구현하였으며, 신뢰성 향상을 위하여 소프트웨어 기반의 고장허용 설계를 구현하였다. 소프트웨어 기반의 고장허용 설계를 위하여 고장 검출 설계는 기본적으로 모니터 삽입 방식을 기반으로 고장 종류에 따라 적절한 고장 검출 방법을 채택하여 설계하였다. 고장 복구 설계는 고장 검출시 장비의 실시간 동작을 유지하기 위하여 신속한 정상 운용이 필요하므로 적응형 고장 복구 방식으로 설계하였다. 적응형 고장 복구 설계를 위하여 고장 종류, 고장 발생시 항공기 운용 모드 및 고장 지속시간을 고려하여 데이터 복구를 위한 적절한 방법을 변경하면서 적용할 수 있도록 설계하였다. 또한 상태 복구 방법도 고장 종류에 따라 백워드 또는 포워드 방식을 선택하여 적용하였다. 이중화 비행조종컴퓨터의 고장 유형에 따른 고장 검출 방법 및 고장 복구 설계에 대한 내용은 Table 3과 같으며, 상세 설계 내용은 다음과 같다.

(1) Timing 오류 고장허용 설계 : 비행조종컴퓨터는 실시간 시스템으로 실시간 주기 스케쥴링에 따라 태스크들을 수행하기 위하여 타이머 인터럽트를 이용하고 반복적으로 주기 태스크들을 수행하도록 설계한다. Fig. 3(a)는 타이머 인터럽트 핸들러(ISR)에 의해 주기적으로 반복되는 프레임의 개념을 보여주는 그림이다. 그러나 소프트웨어 설계 오류에 의해 태스크들을 정상적으로 수행 주기내에 종료하지 못하는 타이밍 오류를 발생시키는 고장을 프레임 오버런(Frame Overrun)이라 한다. Fig. 3(b)는 프레임 오버런 고장 발생시에 수행 주기 동안 종료되지 못한 태스크들은 다음 주기에 계속 수행되는 프레임 개념을 보여주는 그림으로, 이 경우 비행조종컴퓨터는 실시간 처리 요구도를 만족할 수 없고, 프레임 오버런이 발생한 채널과 발생하지 않은 채널의 실행 태스크 차이로 인해 시스템에 심각한 고장을 유발할 수 있다. 프레임 오버런의 고장 검출을 위하여 상태변수 모니터 삽입 방법을 이용하였다. 매 주기마다 정상적으로 모든 태스크가 완료되면 프레임 오버런 상태변수를 완료로 설정하고, 주기 타이머 인터럽트 핸들러 내부의 상태변수 모니터에서 프레임 오버런 상태변수를 확인함으로 이전 프레임의 태스크들이 정상적으로 완료되었는지를 확인하여 프레임 오버런 고장 발생을 검출한다. 고장 검출시 복구 방법은 Fig. 4와 같이 정상적인 프레임 종료시 주기 타이머 인터럽트 핸들러에서 정상적인 프레임 시작을 위한 프로그램 카운터를 스택 포인터에 저장 하고 프레임 오버런 발생시 주기 타이머 인터럽트 핸

Table 3. Fault Detection and Recovery Design by Failure Modes

고장종류	검출방법	복구방법
Timing 오류 (Frame Overrun 발생)	상태변수 모니터	상태 : 포워드 방식 데이터 : 정상채널 이용
비유효 입력 데이터 오류 (부동소수점 처리 오류)	오류 체크 모니터	상태 : 포워드 방식 데이터 : 정상채널 이용, 백워드 방식
채널간 동기화 오류 (태스크 스텝 불일치)	중복 확인 제어 흐름 확인	상태 : 포워드 방식 데이터 : 정상채널 이용
코드 설계오류 (오버플로우, 언더플로우, divide by zero 발생)	데이터 구조 확인	상태 : 백워드 방식 데이터 : 정상채널 이용, 백워드 방식
데이터 비정상 오류 (시리얼 통신 오류)	코딩 확인	상태 : 포워드 방식 데이터 : 정상채널 이용, 백워드 방식
제어 흐름 오류 (요구도 위반 오류)	제어 흐름 확인 합리성 체크	상태 : 백워드 방식 데이터 : 정상채널 이용, 백워드 방식
하드웨어 고장 (CPU, 메모리 고장)	중복 확인	상태 : 백워드 방식 데이터 : 정상채널 이용

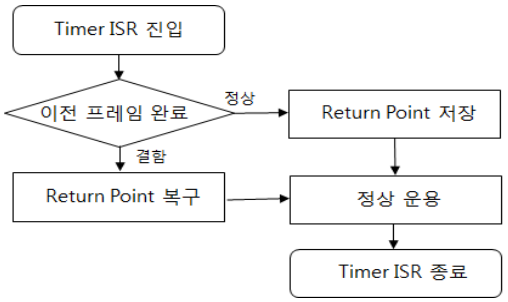


Fig. 4. Frame Overrun Recovery Procedure

들러에서 리턴할 프로그램 카운터를 스택 포인터에 저장하였던 안전한 실행 위치로 이동하는 포워드 복구 방식을 이용하여 장비를 중단없이 채널간 동기화를 유지하면서 실행할 수 있도록 한다. Fig. 3(c)는 고장허용 설계에 의해 프레임 오버런이 발생하여도 다음 프레임에서 정상적인 주기 실행을 수행하는 개념을 보여주는 그림이다.

(2) 비유효 입력 데이터 오류 고장허용 설계 : 부동 소수점 실수 연산 처리를 지원하는 프로세스는 IEEE Std 754 표준 포맷에 따라 부동 소수점 데이터가 전송되어야 하나, 송신 장비의 오류 등으로 인해 부동 소수점 포맷에 위반되는 데이터가 전송되면, 수신 장비에서는 심각한 오류가 발생될 수 있으며, 이를 비유효 입력 데이터 오류라 한다. IEEE Std 754 표준에 따라 정의되지 않는 비유효한 데이터의 종류로는 무한(infinities) 및 비정의(NaN,Not a Number)오류 등이 있으며, 이러한 데이터가 제어 알고리즘의 입력으로 사용될 때 계산 결과가 발산되거나 예상하지 못한 결과를 발생시켜 시스템의 심각한 고장을 유발할 수 있다. 이러한 비유효 입력 데이터 오류를 검출하기 위하여 오류 체크 모니터 삽입 방법을 이용하였다. 비유효 입력 데이터를 검출하기 위하여 실수형 입력 데이터의 경우 수신 데이터를 버퍼에 저장하는 단계에서 모니터를 이용하여 오류 체크를 수행한다. 오류 체크 모니터는 C언어 표준 라이브러리 헤더 math.h의 매크로 함수인 int fpclassify(real-floating x)를 이용하여 입력 데이터 실수 값이 어떤 분류의 오류에 해당하는지 판단하여 검출할 수 있다. 부동 소수점 실수 연산 처리 및 유효 데이터 영역에 대한 오류 확인 모니터 함수는 Fig. 5와 같다. 모니터에서 오류를 검출한 경우 비유효 입력 데이터가 한 채널에서 일시적으로 발생된 경우에는 정상 채널로부터 수신된 채널의 입력 데이터를 CCDL을 통해 수신받아 사용하고, 두 채널에서 동시에 일시적인 오류이거나 한 채널만 운용 가능한 경

```
#include <math.h>
#define FP_NAN        1
#define FP_INFINITE   2
float Invalid_Check(float data_pre, float data,
      float min, float max)
{
if ( fpclassify(data) == FP_NAN ||
     fpclassify(data) == FP_INFINITE )
{ data = val_pre; }
else {
        if (data < min)
        { data = min;  }
        else if (data > max)
        { data = max; }
        else
        {               }
}
return data;
}
```

Fig. 5. Floating Point Error Monitor

우에는 유효한 입력 데이터가 수신되었던 이전 프레임의 데이터를 사용하는 백워드 방식을 설계하였다. 이때 상태 복구방법은 상기 방법으로 선택된 데이터를 이용하여 정상 실행을 계속 수행하는 포워드 방식을 이용하였다.

(3) 채널간 동기화 오류 고장허용 설계 : 이중화 채널에서 수행되는 태스크들은 각 채널 사이에 동기화가 유지되면서 수행되어져야 하며, 동기화가 되지않는 경우 채널간 사용되는 데이터의 불일치로 인해 심각한 장비의 고장이 발생한다. 채널간 동기화 오류 검출을 위해서 동기화 제어 흐름 확인 방법을 이용하였다. 채널사이에 수행되는 태스크의 스텝별로 미리 정의된 동기화 상태변수를 설정하여, 주기적으로 스텝 단계 동기화 상태변수를 채널간 상호 전송하고, 수신된 상대 채널의 스텝별 동기화 상태변수와 자신의 스텝별 동기화 상태변수를 비교하여 차이가 발생하는 경우 고장으로 처리 한다. 채널간 동기화 오류가 검출되면 사전에 정의된 허용 시간 동안 채널간 동기화 알고리즘을 수행한다. 그러나, 허용 시간 내에 동기화가 이루어지지 않고 지속적인 동기화 오류가 유지되는 경우 고장 채널의 판별을 위하여 미리 설계된 자체 점검 모듈을 실행하는 중복확인 방법을 수행하여 고장 채널을 판단한다. 자체점검 모듈은 CPU 코어 결함을 검출하기 위하여 프로그램에서 사용되고 있는 명령어 집합들에 대하여 실행 결과와 기댓값의 비교를 수행하는 명령어 자체시험을 수행한다. 만약 고장 채널이 자체점검 모듈 수행으로 분리되지 않

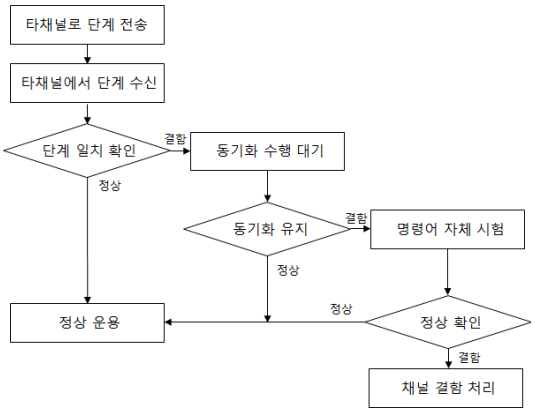
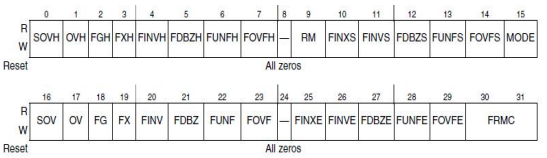


Fig. 6. Sync. Error Recovery Procedure

는 경우 정상 운용시 이중 채널에서 미리 정의된 채널만 수행하고 타채널을 배제하여 장비의 고장 전파를 방지한다. Fig. 6은 채널 동기화 오류발생시 복구 절차이다.

(4) 코드 설계 오류 고장허용 설계 : 코드 실행 중 버퍼의 오버플로우 또는 언더플로우가 발생하거나, 분모에 0이 입력되는 divide by zero 연산 오류가 발생하는 경우 연산 결과를 예측할 수 없을뿐만 아니라 프로세스가 실행을 멈추는 심각한 고장이 발생할 수도 있다. 이러한 종류의 코드 설계 오류에 의한 고장을 검출하기 위해 코드 설계 오류가 발생할 때 프로세스에서 제공하는 소프트웨어 인터럽트인 트랩 사용하여 예외처리 핸들러에서 프로세스 상태 레지스터의 값을 확인하는 데이터 구조 확인 방법으로 고장검출을 수행하였다. 비행조종컴퓨터 탑재 프로세스인 MPC5674F에서 지원되는 예외처리 인터럽트인 IVOR33을 이용하며 고장 발생시 SPEFSCR 레지스터의 상태 비트를 확인하여 고장을 확인한다. Fig. 7은 SPEFSCR 레지스터의 구조를 나타낸다. SPEFSCR 레지스터를 이용한 데이터 구조 확인으로 코드 설계 오류를 검출하면 해당 데이터의 복구를 위해 한 채널에서 일시적인 오류로 판단되는 경우에는 정상 채널로부터 수신된 채널의 입력 데이터를 사용하여 실행을 지속하며, 두 채널에서 동시에 일시적인 오류이거나 한 채널만 운용 가능한 경우에는 유효한 입력 데이터가 수신되었던 이전 프레임의 데이터 또는 미리 정의된 safety value 를 사용하며, 시스템은 예외처리 인터럽트가 발생하기 이전 상태로 복구하는 백워드 방식을 사용한다.

(5) 데이터 비정상 오류 고장허용 설계 : 외부에서 시리얼 통신을 통해 입력되는 데이터가 전송 중 노이즈 등에 의한 오류로 인해 데이터의



FDBZ : Embedded Floating-Point Divide by Zero
FUNF : Embedded Floating-Point Underflow
FOVF : Embedded Floating-Point Overflow

Fig. 7. SPE/EFPU Status and Control Register

변경이 발생하는 경우 이를 입력 데이터로 사용할 때 비정적인 연산 결과로 인한 심각한 오류가 발생될 수 있다. 데이터의 비정상 오류를 검출하기 위하여 CRC, Checksum, heartbeat을 송신하는 신호에 삽입하고 수신측에서 이를 확인하는 코딩 확인 방법을 사용한다. 코딩 확인 방법으로 오류를 검출하면 비유효 입력 데이터가 한 채널에서 일시적인 오류로 판단되는 경우에는 정상 채널로부터 수신된 채널의 입력 데이터를 사용하고, 두 채널에서 동시에 일시적인 오류이거나 한 채널만 운용 가능한 경우에는 유효한 입력 데이터가 수신되었던 이전 프레임의 데이터를 사용하기위한 백워드 방식을 사용하고, 시스템의 상태는 정상 실행을 계속 수행하기 위하여 포워드 방식을 사용한다.

(6) 제어 흐름 오류 고장허용 설계 : 운용 개념에서 정의된 요구도를 위반하는 데이터의 입력에 의해 비행체의 모드 변경에 대한 제한, 입력 데이터 범위 제한 등의 요구도 위반 오류가 발생할 수 있으며, 이 경우 설계 의도와 다른 비정상적인 실행을 수행하는 제어 흐름 오류가 발생한다. 이러한 요구도 위반에 의한 제어 흐름 오류를 검출하기 위해 요구도에서 정의되지 않은 명령이 인가되는 경우는 제어 흐름 확인 중 다중 분기(Multi-way branch) 확인 방법을 사용한다. 다중 분기 확인 방법은 새로운 임무모드 인가시 이전 임무모드의 상태를 확인하고 요구도를 기반으로 사전에 정의한 전이를 위한 상태 확인을 통해 적절한 임무를 수행하며, 전이가 불가능하다고 판단된 경우는 오류로 분류하여 이전 임무모드를 계속 수행할 수 있도록 설계한다. Fig. 8은 다중 분기 확인 방법을 이용한 제어 흐름 오류 발생시 복구 절차이다. 입력 데이터 범위 오류에 대하여 고장 판단은 합리성 확인 방법을 사용한다. 오류를 검출하면 비유효 입력 데이터가 한 채널에서 일시적인 오류로 판단되는 경우에는 정상 채널로부터 수신된 채널의 입력 데이터를 사용하고, 두 채널에서 동시에 일시적인 오류이거나 한 채널만 운용 가능한 경우에는 유효한 입력 데이터가 수

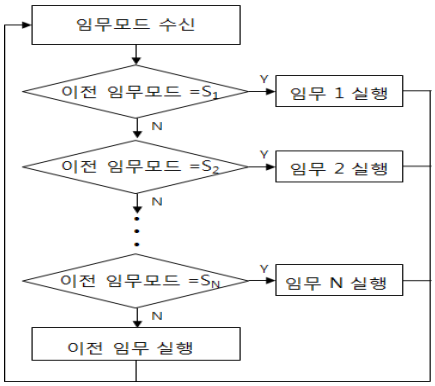


Fig. 8. Control Flow Error Recovery Procedure (Multi-Way Branch Check)

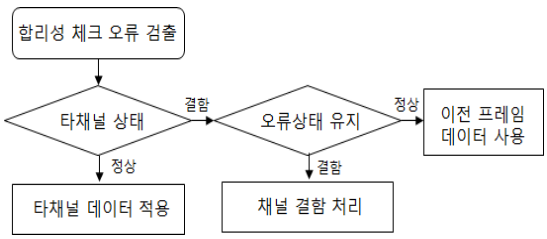


Fig. 9. Control Flow Error Recovery Procedure (reasonableness check)

신되었던 이전 프레임의 데이터를 사용하기위한 백워드 방식을 사용하고 오류가 지속되는 경우 외부 연동장비의 고장 처리를 수행한다. Fig. 9는 합리성 확인 방법을 이용한 제어 흐름 오류 발생 시 복구 절차이다. 시스템의 상태는 제어 흐름 오류 발생 이전 상태에서 정상 실행을 계속 수행하기 위하여 백워드 방식을 사용한다.

2.7 시험 및 평가

본 논문에서 제안한 이중화 비행조종컴퓨터의 소프트웨어 기반 고장허용 설계에 대한 성능 검증을 위하여 자체적으로 개발한 Fig. 10의 소프트웨어 시험환경을 사용하였다. 소프트웨어 시험환경은 외부 입력 신호를 모사하고 출력 신호를 수신하기 위한 입출력 신호 시뮬레이터와 소프트웨어 내부 변수를 프로그램 실행 중 읽고 쓰기 위한 프로세스 모니터로 구성하였다. 프로세서 모니터는 소프트웨어의 내부 데이터를 실시간으로 모니터링하기 위하여 이중으로 구성된 비행조종컴퓨터의 각 채널에 대하여, 상용 NEXUS[11] 에뮬레이터를 이용하여 호스트 컴퓨터로부터 수신받은 명령을 실시간으로 처리하기 위한 SBC(Single Board Computer)로 개발하였다[12]. 고장허용 설계 기능의 시험을 위하여 비행조종컴

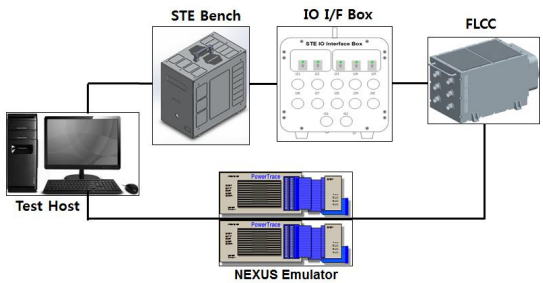


Fig. 10. Software Test Environment

Table 4. Fault Injection Methods

고장종류	고장주입 방법	결과확인
Timing 오류	시험 코드 삽입	- 채널 정상 상태 유지 - 20ms 이내 고장 복구
비유효 입력 데이터 오류	외부 입력 고장 주입	
채널간 동기화 오류	채널 일시 정지	
코드 설계오류	내부 상태 고장 주입	
데이터 비정상 오류	외부 입력 고장 주입	
제어 흐름 오류	외부 입력 고장 주입	

퓨터의 운용 중 정의된 고장을 주입하고 이때 고장 허용 설계에서는 주입된 각각의 고장을 판단하고 고장의 종류에 따라 설계된 고장허용 방법으로 장비를 정상 운용할 수 있는 상태로 전환되는것을 확인하여야 한다. 장비의 운용 중 장비의 외부에서 발생되는 고장 주입을 위하여 소프트웨어 시험환경의 입출력 신호 시뮬레이터를 이용하여 사전에 정의된 비정상적인 입력을 인가하고, 장비의 상태를 확인한 결과 장비는 설계된 요구도에 따라 발생된 고장을 보고하고, 정상적인 동작을 지속적으로 수행하는지 확인하였다. 장비의 내부에서 발생되는 고장 주입은 소프트웨어 고장 주입 방식[13-15]을 기반으로 구현하였다. 이를 위하여 자체 개발한 시험 환경의 기능인 프로세스 모니터를 이용하여 프로그램 운용 중에 소프트웨어 내부 상태 변수에 직접 고장 상태를 입력하고 장비가 설계된 요구에 따라 정상적인 동작을 수행하는지 확인하였다. 고장 종류별 주입 방법은 Table 4와 같다.

고장허용 설계의 중요한 판단 지표는 비행조종컴퓨터가 고장주입 이후에도 장비의 정상상태 유지 여부와 상태 복구 시간이다. 비행조종컴퓨터는 치명적인 고장 발생시 상태가 복구되지 않으면 해당 채널을 배제하도록 펌웨어를 이용하여 워치도그 타이머 및 채널 유효 판단 모듈을 구현

하였다. 시험 수행시 고장허용 설계가 포함되지 않은 비행조종컴퓨터에 고장을 주입한 경우 해당 채널이 정상적으로 배제되는것을 확인하였으며, 고장허용 설계를 포함하여 동일한 고장을 주입한 경우 채널이 정상적으로 운용되는것을 확인하였다. 또한 상태 복구 시간은 고장 발생으로 인한 영향성을 장비에 최소화할 수 있도록 신속하게 상태를 정상으로 복구하기 위하여 장비의 최소 운용 주기인 20ms 이내에 상태를 정상으로 복구하는것을 내부 상태 변수를 통해 확인하였으며, 이를 통해 고장허용 설계가 정상적인 동작을 수행하는 것을 확인하였다.

Ⅲ. 결 론

신뢰성 향상을 위해 비행조종컴퓨터 개발 시 다양한 설계 기법들이 적용되고 있으며, 고장허용 설계 기법도 그 중 하나이다. 기존의 고장허용 설계 기법은 하드웨어적인 다중화 구조 설계와 보팅 알고리즘을 이용한 하드웨어 기반 고장허용 설계 기법이 주로 적용되었다. 그러나 하드웨어 기반 고장허용 설계는 설계가 쉬운 장점이 있지만, 하드웨어 부피, 무게 및 비용 등의 많은 제약이 있어, 소프트웨어 기반의 고장허용 설계 기법의 적용을 위한 연구들이 많이 이루어지고 있다. 본 논문에서는 신뢰성 향상을 위해 하드웨어적으로 이중으로 구성된 비행조종컴퓨터의 소프트웨어 기반 고장허용 설계를 위하여 소프트웨어 설계 오류에 의해 발생 가능한 고장 모드를 분석하고, 고장 모드에 대한 검출 및 복구 설계 기법에 관한 연구를 수행하였다. 소프트웨어 기반 고장허용 설계를 위한 고장검출 단계에서는 모니터 삽입, 중복 확인, 제어 흐름 확인, 데이터 구조 확인, 코딩 확인 및 합리성 체크 방법 등을 사용하였으며, 고장 검출 후 고장 복구를 위해서는 데이터의 치명적인 오류인 경우 시스템을 고장 이전 상태로 되돌리는 백워드 방식을, 시스템의 상태가 완료된 경우에는 고장 발생시 이전 상태를 무시하고 정상적인 다음 상태에서 재수행하는 포워드 방식을 이용하였다.

설계된 소프트웨어 기반 고장허용 설계는 이중화 비행조종컴퓨터에 탑재하여 자체 개발한 소프트웨어 시험 환경을 이용하여 고장 주입 시험을 수행하였으며, 주입된 오류에 의해 발생한 고장들에 대하여 소프트웨어 기반 고장허용 알고리즘을 통해 고장 검출 및 복구 단계를 수행하고 비행조종컴퓨터가 정상동작을 유지하는것을 확인하는 것으로 고장허용이 되는것을 확인하였다.

References

1) C. B. Feldstein and J. C. Muzio, "Development of A Fault Tolerant Flight Control System", Proc. of the 23rd Digital Avionics Systems Conference, 2004.

2) L. L. Pullum, "Software Fault Tolerance Techniques and Implementation", Artech House, 2001.

3) W. Torres-Pmales, "Software Fault-Tolerance : A Tutorial", NASA Technical Memorandom TM-2000-210616, 2000

4) Z. Xie, H. Sun and K. Saluza, "A Survey of Software Fault Tolerance Techniques", University of Wisconsin-Madison/Department of Electrical and Computer Engineering 1415 Engineering Drive, Madison WI 53706 USA

5) T. Anderson, P. A. Barrett, D. N. Halliwell and M. R. Moulding, "Software fault tolerance: an evaluation", IEEE Transactions on Software Engineering, Vol. SE-11, No. 12, pp.1502-1510, 1985.

6) G. K. Saha, "Approaches to Software Based Fault Tolerance - A Review", Computer Science Journal of Moldova, Vol. 13, No. 3(39), 2005.

7) A. Avizienis, "The N-version approach to fault-Tolerant software", IEEE Trans. on Software Eng., Vol. 11. No.12. pp. 1491-1501, Dec. 1985

8) A. Avizienis and J. P. J. Kelly, "Fault Tolerance by Design Diversity: Concepts and experiments", IEEE Computer, 17, pp.67-80, 1984.

9) R. W. Butler, "A Primer on Architectural Level Fault Tolerance", NASA Technical Report TM-2008-215108, 2008

10) B. W. Johnson, "fault-tolerant microprocessor-based systems", IEEE Micro, Vol. 4, No. 6, pp.6-21, 1984.

11) IEEE-ISTO, “The Nexus 5001 Forum Standard for a Global Embedded Processor Debug Interface”, 2003

12) H. S. Yoon and J. P. Han, "Development of Verification Environment for Flight Safety Critical Software using NEXUS", Journal of the Korean Society for Aeronautical & Space Sciences, Vol. 40, No.6, 2012, pp.548-554

13) R. R. Some, W. S. Kim, G. Khanoyan, L. Callum, A. Agrawal, and J. J. Beahan, "A software-implemented fault injection methodology for design and validation of system fault tolerance", In Dependable System and Networks, 2001. Proeedings. The International Conference on, pp501-506, 2001.

14) A. Johansson, "Software implemented fault injection used for software evaluation", in Building Reliable Component-Based System, I. Crnkovic and M. Larsson, Eds., ed: Artech House, 2002.

15) A. Tai, M. Hecht and H. Hecht, "A New Method for the Verification of Fault Tolerant Software", Proc. EASCON 87, Washington, DC, October, 1987