JANT 한국항행학회논문지
Journal of Advanced Navigation Technology

기반 기술

J. Adv. Navig. Technol. 27(4): 429-435, Aug. 2023

항공전자 시스템에서 ARINC653 기반의 FACE를 준수하는 IOS 및 TS 세그먼트 구조 설계

A design of FACE-compliant IOS and TS segments architecture based on ARINC653 in avionics system

이 두 환*· 남 영 욱· 조 경 연 · 윤 지 용
엘아이지넥스원

Doo-Hwan Lee* · Young-Uk Nam · Kyeong-Yeon Cho · Ji-Yong Yoon
LIG Nex1, Daejeon, 34127, Korea

[요 약]

항공전자 시스템의 복잡성이 높아짐에 따라 소프트웨어 컴포넌트의 이식성 및 재사용성이 강조되었다. 본 논문에서는 ARINC 653 요구사항을 만족하는 VxWorks 653 운용 환경에서 FACE(The Future Airborne Capability Environment)표준에 적합한 IOSS(Input Output Service Segment) 및 TSS(Transport Service Segment)에 대한 구조 설계 방안을 설명한다. IOSS 및 TSS는 각각 다른 파티션에서 독립적으로 동작하게 하여 시/공간 분리 및 다른 소프트웨어의 영향성을 최소화 하였고, 이식성 및 재사용성을 높이기 위해 디자인 패턴 중 전략 패턴을 적용하였다. 또한, IOSS는 Distributed IO Service 구조를 적용하여 외부 인터페이스 서비스를 제공하고, 외부 인터페이스 중 FACE를 적용한 장비의 ARINC 664 P7 인터페이스는 TSS에 배치하여 데이터 이동 경로를 최적화 하였다.

[Abstract]

The increasing complexity of avionics systems has emphasized the portability and reusability of software components. In this paper, a structural design method for IOSS (Input Output Service Segment) and TSS (Transport Service Segment) complying with the FACE (The Future Airborne Capability Environment) standard in the VxWorks 653 operating environment that satisfies ARINC 653 requirements is described. IOSS and TSS operate independently in different partitions to minimize time/space separation and the influence of other software, and to increase portability and reusability, strategy patterns among design patterns are applied. In addition, IOSS provides external interface service by applying distributed IO service structure, and among external interfaces, the ARINC 664 P7 interface of FACE-compliant equipment is placed in TSS to optimize the data movement path.

Key word : ARINC 653, Avionics Software, FACE, Open System Architecture**,** Portability.

http://dx.doi.org/10.12673/jant.2023.27.4.429

Received 25 July 2023; **Revised** 10 August 2023
Accepted (Publication) 22 August 2023 (30 August 2023)

***Corresponding Author : Doo-Hwan Lee**

Tel:*** _ **** _ ****
E-mail: doohwan.lee2@lignex1.com

Ⅰ. 서 론

항공전자 시스템 구조는 분산형, 연방형, 모듈 통합형 구조에서 차세대 통합 아키텍처로 발전 중에 있다[1]. 분산형 구조는 개별 기능의 독립된 항전 장비가 다수 있었으며, Point-to-Point 방식의 배선 방식이었다. 이후 LRU(Line Replace Unit) 단위 시스템 구성으로 복수의 항전 기능을 통합하였고, 시스템 버스를 도입한 연방형 구조로 발전하였으며, F-15K, KF-16, T/A-50, Euro Fighter가 그 예이다. 이후 항전 처리 기능을 하나의 샤시에 통합하고 고속 통신을 이용한 센서 테이터 취합 및 데이터 처리를 위한 자원 공유를 도입한 모듈 통합형 구조로 발전하였으며, F-22, F-35, Rafale 등에 적용되었다. 앞으로는 구성 요소와의 인터페이스를 표준화하여 개발 및 유지보수가 용이한 통합 아키텍처로 발전 중에 있다. 이에 따라 소프트웨어 또한 개방형 소프트웨어 아키텍처가 필요해 졌고, 이식성 및 재사용성을 높이기 위해 FACE와 같은 개방형 소프트웨어 아키텍처를 적용하게 되었으며, 관련 연구 또한 활발히 진행되고 있다[2].

본 논문은 ARINC 653을 만족하는 VxWorks 653 운영 체제 환경에서 이식성 및 재사용성을 높이기 위해 개발된 FACE 아키텍처 중 외부 장비와 연동을 수행하는 IOSS(Input Output Service Segment) 및 TSS(Transport Service Segment)에 대한 설계 방안을 제시한다.

본 논문의 구성은 다음과 같다. 2장에서 ARINC 653 및 FACE 아키텍처에 대한 관련 기술을 설명하고 3장에서는 FACE에 적합한 IOSS 설계 방안 및 FACE를 적용한 장비간 ARINC 664 P7 통신을 적용할 경우 TSS 설계 방안을 제시한다. 4장의 경우 설계한 IOSS 및 TSS를 검증하기 위한 시스템 통합 환경 및 결과를 제시하여 해당 설계가 해당 시스템에 적절하게 설계되었는지를 확인할 수 있다. 마지막으로 5장에서 결론을 맺는다.

Ⅱ. ARINC 653 및 FACE 아키텍처

2-1 ARINC 653

ARINC 600 시리즈는 디지털 항공전자의 물리적인 패키징, 장비 장착, 데이터 통신 표준 등 사양을 포함하고 있다. 특히, ARINC 653의 경우 운영 체제(OS; Operating System)와 응용 소프트웨어와의 일반적인 APEX(APplication/EXecutive) 통신을 정의하며, 시/공간 파티셔닝, 에러 탐지 및 보고하는 Heath Monitoring, 포트를 통한 통신에 대한 사양을 정의한다. 또한, 멀티 코어 적용으로 다른 파티션 코드, 입출력(IO; Input Output), 데이터 저장 영역을 침범하지 않고, 할당된 시간에 공유 리소스를 사용할 수 있으며, 하나의 파티션 고장이 다른 파티션에 영향을 주지 않는, 즉 시/공간에 대한 강건한 파티션 처리가 강조되었다[3].

ARINC 653을 만족하는 실시간 운영 체제로는 대표적으로 VxWorks 653 및 LynxOS-178이 있다. VxWorks 653은 Wind River사에서 개발한 실시간 운용체제로서 항공 우주 분야에서 널리 사용되며, 높은 안정성과 신뢰성, 분리된 파티션 운영 등의 기능을 제공한다. LynxOS-178은 Lynx Software Technologies사에서 개발된 운영 체제로 ARINC653, DO-178C, POSIX 등의 규격을 준수하고 있다.

본 논문에서는 PowerPC T2080 프로세서에 적용 가능한 VxWorks 653 OS를 적용하였다. VxWorks 653 구조를 간략하게 설명하면, 보드 장치 위에 멀티 코어 프로세서가 있으며 하드웨어 접근을 지원해 주는 가상 버스 위에 Hypervisor 역할을 수행하는 MOS(Module OS)가 위치하고 그 위에 각 Partition 별로 Guest OS인 POS(Partition OS)가 위치하여 어플리케이션이 동작할 수 있도록 한다.

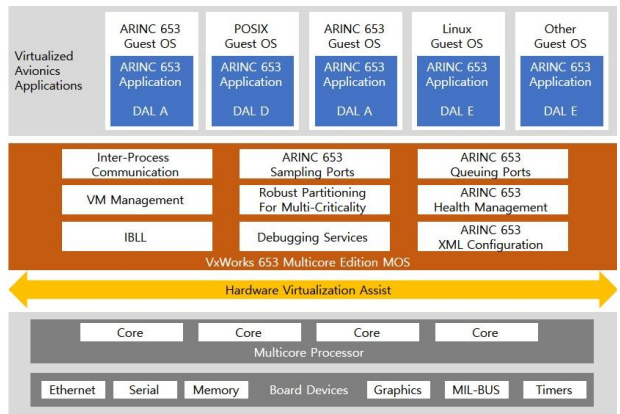


그림 1. VxWorks 653 구조
Fig. 1. VxWorks 653 Architecture

2-2 FACE 아키텍처

FACE는 OS 및 하드웨어 변경에 따른 영향성 최소화 및 소프트웨어 재사용성 확보를 위해 미 해군 및 미국 주요 방산업체가 참여해서 개발하였으며, 소프트웨어 개발 및 인터페이스 표준화를 목적으로 한다. 즉, 소프트웨어를 다른 플랫폼에 배치할 때 영향을 최소화하기 위한 환경을 제공하는 것이며 기존의 플랫폼 종속적인 항공전자 소프트웨어의 이식성을 향상시켜 여러 항공기 플랫폼에 적용하는 것이다. Layered Architecture 기반으로 컴퓨팅 플랫폼에 대한 접근 및 제어를 수행하는 OSS(Operating System Segment), 연동 대상 장비와의 통신 관리를 수행하는 IOSS(I/O Services Segment), 연동 대상 장비를 제어/관리하는 PSSS(Platform Specific Service Segment), 플랫폼 하드웨어에 종속되지 않는 임무 레벨의 기능을 제공하는 PCS(Portable Components Segment), PCS와 PSSS 간 통신을 위한 표준 인터페이스를 제공하는 TSS(Transport Service Segment)로 구성[4]-[6]된다.

OSS는 모든 Segment에서 OS API를 이용해 접근 가능하며,

IOSS와 PSSS간에는 I/O Service Interface를 통해 서로 데이터를 공유한다. PSSS, TSS, PCS간 통신은 Transport Service Interface를 통해 수행한다.

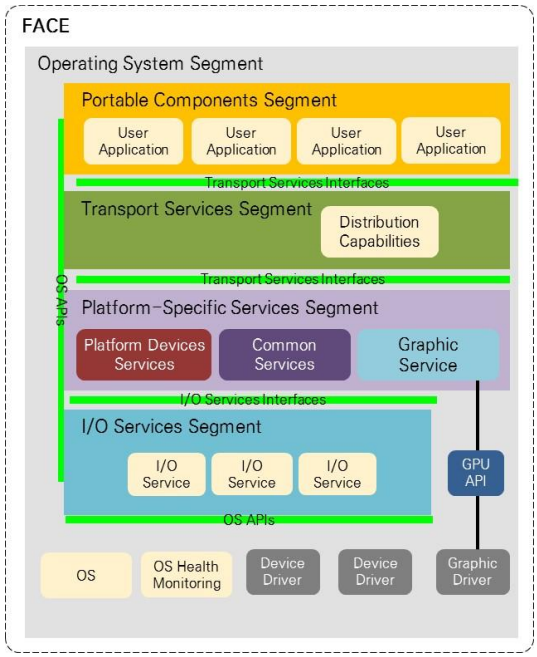


그림 2. FACE 아키텍처
Fig. 2. FACE Architecture

Ⅲ. IOS 및 TS 세그먼트 구조 설계

3-1 IOS 세그먼트 구조 설계

IOS 세그먼트는 연동 대상 하드웨어와의 통신을 관리하고, PSSS 컴포넌트가 사용하는 IO 프로토콜을 추상화 한다. 또한, 비 표준 IO 디바이스 드라이버 접근을 위한 공통 API를 제공한다.

FACE 표준에는 IO Service 구조 설계를 Distributed/Non-Distributed 방식을 제시[6]하고 있으며, Non-Distributed IO Service는 PSSS에서 FACE IO API, 즉 IO Service Interface를 통해 IO Service를 직접 제어하며, IO Service는 PSSS와 동일한 파티션에 포함하여 구현된다. Distributed IO Service는 하나의 파티션에서 IO Service를 제공하며 데이터 이동 기능을 제공하기 위해 라이브러리 형태로 FACE IO API를 제공하며, IPC(Inter-Process Communication)을 사용하여 통신한다.

Non-Distributed IO Service 구조는 PSSS에서 직접 호출하여 속도 측면에서는 장점이 있으나 하나의 파티션에 여러 세그먼트를 할당해야 하고 이에 따라 독립된 구조로 설계 및 구현할 수 없다. 따라서 본 논문은 이식성, 확장성 및 재사용성을 위해 독립된 구조의 Distributed IO Service구조를 채택하여 하나의 파티션에서 하드웨어를 제어하는 방식으로 구현하였고, 각 세그먼트간 인터페이스를 위해서 VxWorks 653에서 제공하는 라이브러리를 링크하여 FACE API를 사용하였다.

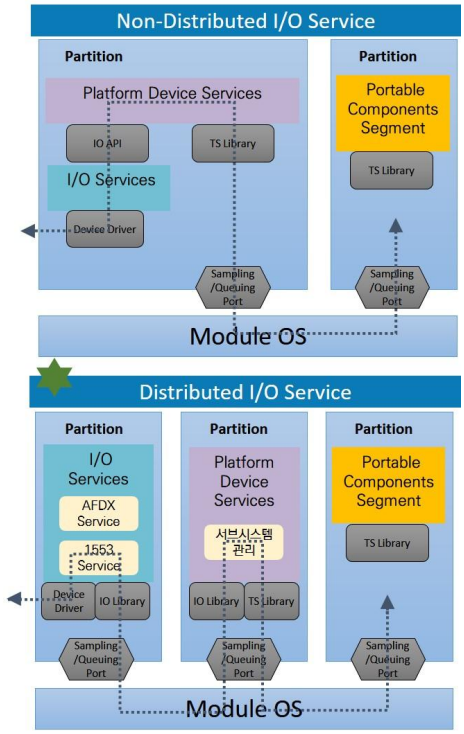


그림 3. FACE I/O Service 구조 설계 방식
Fig. 3. FACE I/O Service Architecture Design Approach

IO Service 클래스 설계는 인터페이스 종류가 추가/삭제 될 경우 용이하도록 디자인 패턴 중 전략 패턴을 적용하였다. 즉, 장비마다 적용되는 인터페이스 종류에 따라 CIOServiceManager 클래스에서 CIOService 객체를 생성하여 운용하도록 설계하였다.

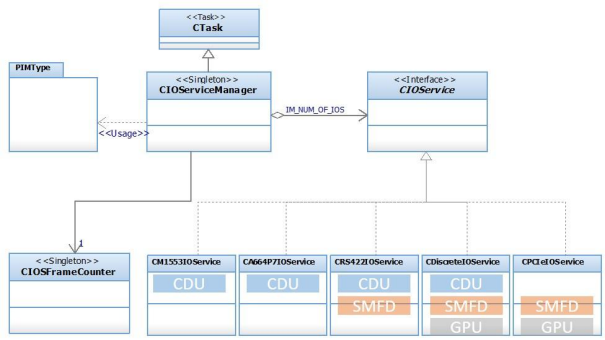


그림 4. FACE IO Service 클래스 다이어그램
Fig. 4. FACE IO Service Class Diagram

IO Service는 특히 재사용성을 극대화하기 위해 .xml 파일 형식의 Parameter Data를 통해 초기화 시 해당 파일을 파싱하여 인터페이스 종류, 채널, 형식 등 속성을 설정하여 코드 수정

없이 자동으로 운용되도록 설계하였다. 즉, 하나의 파티션에서 독립적으로 IO Service를 구현함으로써 시간 및 메모리에 대한 파티셔닝이 가능하고, 별도의 컴포넌트로 구현할 수 있다.

아래 그림은 MIL-STD-1553B 인터페이스 운용을 위한 속성을 정의하고 XML 파일로 해당 속성을 정의한 결과이다.

MIL-STD-1553B		
XML IOType Information	Parameter Type	Note
Message Name	char[15]	
Channel Number	unsigned short	
Message Number	unsigned short	
Message Type	unsigned short	Msg Type (ex, BC-RT RT-BC RT-RT...) E1553_MSG_NONE = 0, E1553_MSG_BC_RT = 1, E1553_MSG_RT_BC = 2, E1553_MSG_RT_RT = 3, E1553_MSG_MODE_CODE
Transmit RT Address	unsigned short	
Transmit Sub Address	unsigned short	
Receive RT Address	unsigned short	
Receivce Sub Address	unsigned short	
T/R bit	unsigned short	0 : RT receive(BC-RT transaction) 1 : RT Transmit(RT-BC transaction)
Word Count	unsigned short	
Frame Mask	unsigned int	
AP Message	bool	
Message Option	unsigned short	Default Bus, Retry Option, etc.

[IOSConfig.xml]

```
<?xml version="1.0" encoding="utf-8"?>
<IOSConfig>
  <Connection>
    <Name>ICD-CDU-EGI1-0010_DES</Name>
    <IOType>MIL_STD_1553B</IOType>
    <M1553_CHNum>0</M1553_CHNum>
    <M1553_MsgNum>0</M1553_MsgNum>
    <M1553_MsgType>1</M1553_MsgType>
    <M1553_TrnsRTAddr>0</M1553_TrnsRTAddr>
    <M1553_TrnsSubAddr>0</M1553_TrnsSubAddr>
    <M1553_RecvRTAddr>5</M1553_RecvRTAddr>
    <M1553_RecvSubAddr>1</M1553_RecvSubAddr>
    <M1553_TRBit>0</M1553_TRBit>
    <M1553_WC>4</M1553_WC>
    <M1553_FrameMask>131074</M1553_FrameMask>
    <M1553_APMsg>0</M1553_APMsg>
    <M1553_MsgOption>12353</M1553_MsgOption>
  </Connection>
  <Connection>
    <Name>ICD-CDU-EGI1-0020_DES</Name>
    <IOType>MIL_STD_1553B</IOType>
    <M1553_CHNum>0</M1553_CHNum>
    <M1553_MsgNum>1</M1553_MsgNum>
    <M1553_MsgType>1</M1553_MsgType>
    <M1553_TrnsRTAddr>0</M1553_TrnsRTAddr>
    <M1553_TrnsSubAddr>0</M1553_TrnsSubAddr>
    <M1553_RecvRTAddr>5</M1553_RecvRTAddr>
    <M1553_RecvSubAddr>2</M1553_RecvSubAddr>
    <M1553_TRBit>0</M1553_TRBit>
    <M1553_WC>11</M1553_WC>
    <M1553_FrameMask>536879104</M1553_FrameMask>
    <M1553_APMsg>0</M1553_APMsg>
    <M1553_MsgOption>12353</M1553_MsgOption>
  </Connection>
```

그림 5. FACE IO Service 파라미터 데이터(MIL-STD-1553B)
Fig. 5. FACE IO Service Parameter Data(MIL-STD-1553B)

3-2 TS 세그먼트 구조 설계

TS 세그먼트는 PCS, PSSS 간 통신을 위한 표준 인터페이스를 제공하며, T Service 및 TS API로 구성하였다. FACE 아키텍처 세그먼트 간의 데이터 이동은 IOSS에서 외부로부터 입력을 받아 그것을 처리하는 PSSS 계층으로 송신하고 PSSS는 해당 데이터를 FACE 데이터 모델을 바탕으로 데이터 가공하여 TSS를 통해 PCS로 송신하는 과정을 거친다. 이러한 불필요한 데이터 이동 및 중복된 데이터 가공하는 부분을 최적화 하기 위해 장비간 ARINC 664 P7 통신하는 부분을 TSS로 배치하였고, TS API는 IO API와 마찬가지로 VxWorks 653에서 제공하는 라이브러리를 링크하여 사용하도록 구현하였다.

또한, 메모리 간섭에 의한 영향성을 없애기 위해 물리적인 메모리는 오직 IO Service 파티션에서 만 접근하도록 설계하였다. 따라서 ARINC 664 P7 인터페이스를 수행하는 T Service 파티션은 IO Service 파티션의 POS와 IPC통신을 수행하고 IO Service 파티션에서 ARINC 664 P7 Core가 실행되는 FPGA를 제어하는 것으로 설계하였다. 즉, PSSS에서 PCS가 필요한 데이터를 가공하여 TS API를 통해 TS 메시지 데이터 구조체에 맞도록 구성하여 송신하면 ARINC P7 Service에서 해당 데이터를 TS API를 통해 해당 데이터를 수신한다. 수신한 데이터를 PCS에 송신 및 APEX 포트를 통해 IO Service 파티션에 송신한다. IO Service 파티션의 POS Task에서 해당 데이터를 수신하여 외부와 통신한다. 해당 데이터를 받는 장비는 역순으로 데이터를 가공 없이 PCS에서 운용하도록 한다.

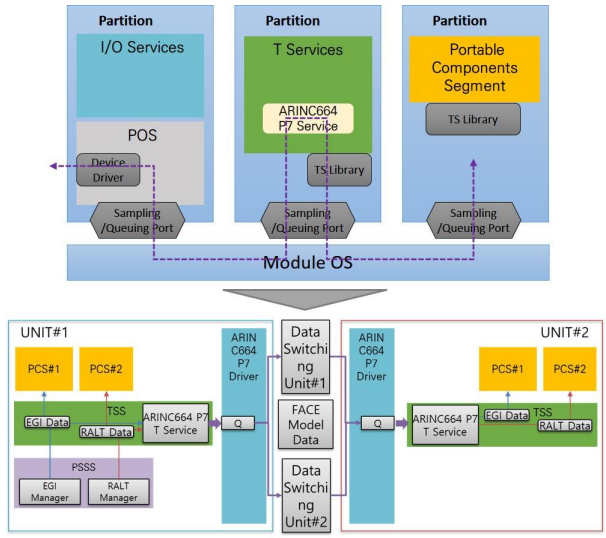


그림 6. ARINC 664 P7 데이터 흐름
Fig. 6. ARINC 664 P7 Data Flow

T Service 클래스 다이어그램 설계는 그림 7과 같다. CARINC 664P7Service 객체가 초기화 수행 시 TSConfig.xml 파일을 파싱하여 CARINC 664P7TSChannelHander 및 CARINC 664P7TSMsg 객체를 통해 채널 및 메시지를 초기화하여 운용한다.

TSConfig.xml은 PCS 및 PSSS와 통신을 수행하는 APEX 포터에 대한 이름 및 장비 외부와 통신하기 위해 IOSS의 POS Task와 통신하는 ARINC 664 P7 포트에 대한 정보를 가지고 있으며 해당 파일의 정보만 변경하고 소스 코드는 재사용할 수 있도록 설계하였다.

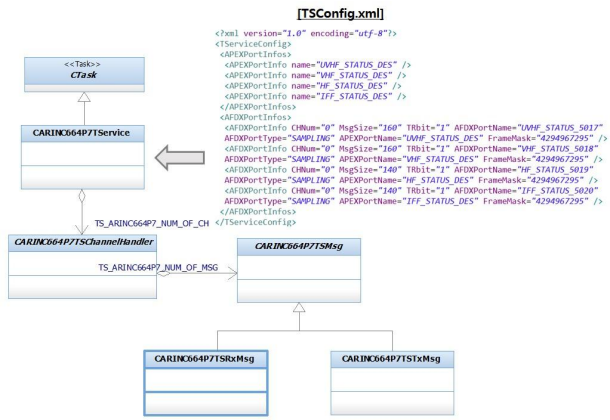


그림 7. FACE T Service 클래스 다이어그램
Fig. 7. FACE T Service Class Diagram

Ⅳ. 시스템 통합

4-1 Core 및 Partition 할당

해당 시스템은 Power PC T2080 프로세서를 사용하였고, 해당 프로세서는 물리적으로 4개의 Core를 가지고 있다. 따라서 시스템 통합은 코어 및 파티션에 대한 시간 관리 및 물리적 메모리/장치 관리가 매우 중요하다[7].

코어 할당은 파티션 및 프로세스의 실행 오버헤드를 고려하여 상호작용이 많은 파티션은 동일 코어에 배치하였고, 병렬 실행 가능하고 간섭이 없는 파티션은 분리 하였다. 파티션 할당에 대해서는 플랫폼, 디바이스 의존적 컴포넌트, 비의존적 컴포넌트의 파티션 분리를 수행하였고, BSP(Board Support Package), 디바이스 드라이버와 IOSS는 동일 파티션으로 배치하였다.

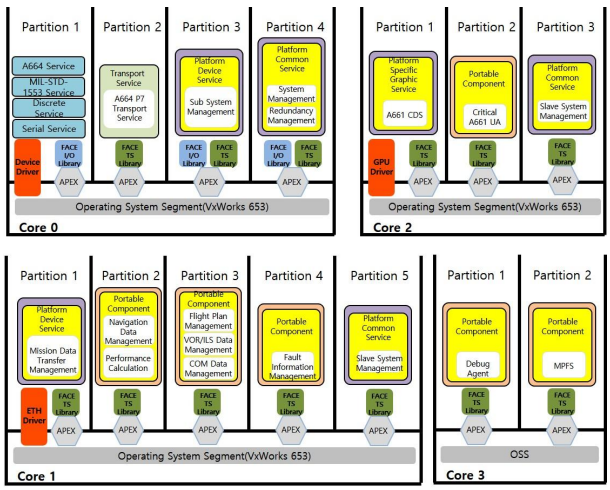


그림 8. Core 및 Partition 할당
Fig. 8. Core And Partition Allocation

시스템은 MIL-STD-1553B Minor 프레임 주기가 50Hz(20ms)이기 때문에 파일 시스템을 제외한 운용 프로그램은 20ms 주기로 실행하도록 구성하였다. 또한, Core0에는 하드웨어를 직접 접근하는 IO Service, IO Service와 상호작용이 많은 T Service 및 각 장비 관리하는 파티션을 배치하였다. Core1은 각 POS를 배치하였고, Core2는 그래픽 처리를 수행하는 ARINC 661 CDS(Cockpit Display System) 및 UA(User Application)를 배치하였다. 마지막으로 Core3는 파일 시스템과 Debug Agent를 배치하여 시스템을 구성하였다. 파일 시스템은 IO Service 및 T Service 파티션에서 사용하는 설정파일인 IOSConfig.xml, TSConfig.xml 파일을 관리한다. VxWorks 653에서 제공하는 MPFS(Multi Partition File System)를 사용하였고 Server는 MPFS 파티션, 다른 파티션에서는 Client로 운용하였다.

표 1. Core 및 Partition 할당(시간 정보 포함)
Table. 1. Core And Partition Allocation(Including time information)

Core	Partition	Periodic	Duration	Note
Core0	IO Service	50 Hz	11ms	Placed on Core0 because it accesses the hardware
	T Service	50 Hz	3ms	Placed on Core0 because it interacts a lot with IOSS
	Sub System Management	50 Hz	5ms	Placed on Core0 like IOS by performing IO Interface
	System & Redundancy Management	50 Hz	1ms	-
Core1	Mission Data Transfer Management	50 Hz	6ms	PCS is placed on Core1
	Navigation Management	50 Hz	4ms	
	Flight Plan & Communication Management	50 Hz	4ms	
	Fault Information Management	50 Hz	5ms	
	System Management(Slave)	50 Hz	1ms	-
Core2	Display Management (CDS)	50 Hz	13ms	ARINC 661 CDS and UA are placed on the same Core
	Display Management(UA)	50 Hz	6ms	
	System Management(Slave)	50 Hz	1ms	
Core3	MPFS	500 Hz	1ms	File system
	Debug Agent	500 Hz	1ms	If necessary, register and use the Debug Agent.

4-2 시스템 통합 결과

시스템 통합은 SIL에서 수행하였다. 사용자 입력 또는 항법/통신/식별 등 항공전자 모의 데이터는 MIL-STD-1553B BC(Bus Controller)로 동작하는 CDU(Cockpit Display Unit)와 연동을 수행하였고, CDU, SMFD(Smart Type Multi-Function Display) 및 GPU(Graphic Processing Unit)는 DSU(Data Switching Unit)을 통해 ARINC 664 P7 통신을 수행하였다.

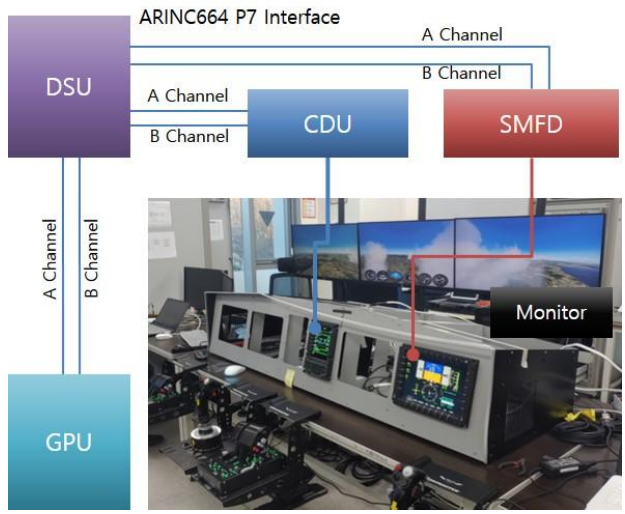
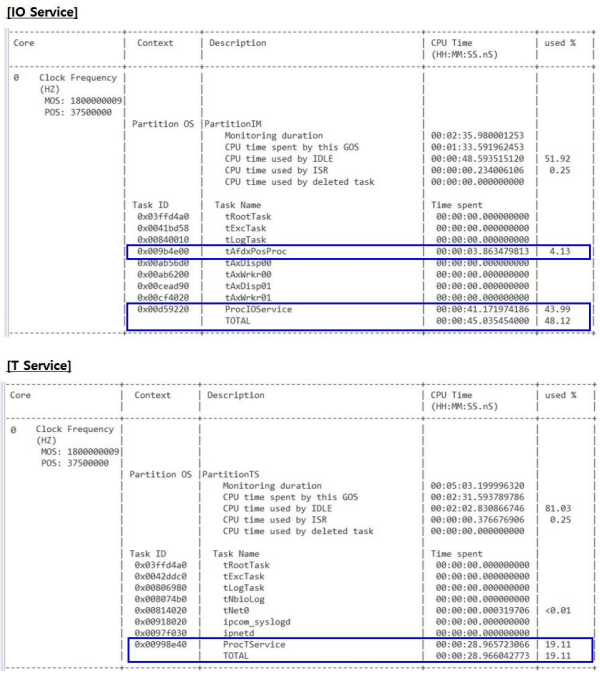


그림 9. SIL 시험 결과
Fig. 9. SIL Test Result

아래 시험 결과는 VxWorks 653의 Debug Agent를 통해 Time Monitor Performance를 사용하여 측정한 결과이다. TOTAL 값은 파티션이 할당된 시간에서 실제 사용하는 시간의 백분율을 의미하며, IO Service 및 T Service는 50%이내의 시간을 사용하는 것을 알 수 있다.



[IO Service]

Core	Context	Description	CPU Time (HH:MM:SS.nS)	used %
0 Clock Frequency (HZ) MOS: 1800000009 POS: 37500000				
	Partition OS	PartitionIM		
		Monitoring duration	00:02:35.980001253	
		CPU time spent by this GOS	00:01:33.591962453	
		CPU time used by IDLE	00:00:48.593515120	51.92
		CPU time used by ISR	00:00:00.234006106	0.25
		CPU time used by deleted task	00:00:00.000000000	
	Task ID	Task Name	Time spent	
	0x03ffd4a0	tRootTask	00:00:00.000000000	
	0x0041bd58	tExcTask	00:00:00.000000000	
	0x00840010	tLogTask	00:00:00.000000000	
	0x009b4e00	tAfdxPosProc	00:00:03.863479813	4.13
	0x00ab56d0	tAxDisp00	00:00:00.000000000	
	0x00ab6200	tAxWrkr00	00:00:00.000000000	
	0x00cead90	tAxDisp01	00:00:00.000000000	
	0x00cf4020	tAxWrkr01	00:00:00.000000000	
	0x00d59220	ProcIOService	00:00:41.171974186	43.99
		TOTAL	00:00:45.035454000	48.12

[T Service]

Core	Context	Description	CPU Time (HH:MM:SS.nS)	used %
0 Clock Frequency (HZ) MOS: 1800000009 POS: 37500000				
	Partition OS	PartitionTS		
		Monitoring duration	00:05:03.199996320	
		CPU time spent by this GOS	00:02:31.593789786	
		CPU time used by IDLE	00:02:02.830866746	81.03
		CPU time used by ISR	00:00:00.376676906	0.25
		CPU time used by deleted task	00:00:00.000000000	
	Task ID	Task Name	Time spent	
	0x03ffd4a0	tRootTask	00:00:00.000000000	
	0x0042ddc0	tExcTask	00:00:00.000000000	
	0x00806980	tLogTask	00:00:00.000000000	
	0x008074b0	tNbioLog	00:00:00.000000000	
	0x00814020	tNet0	00:00:00.000319706	<0.01
	0x00918020	ipcom_syslogd	00:00:00.000000000	
	0x0097f030	ipnetd	00:00:00.000000000	
	0x00998e40	ProcTService	00:00:28.965723066	19.11
		TOTAL	00:00:28.966042773	19.11

그림 10. CPU 시간 측정 결과
Fig. 10. CPU Time Measurement Result

Ⅴ. 결 론

본 논문에서는 개방형 항공전자 소프트웨어 아키텍처인 FACE 및 ARINC653에 대해 소개하고 FACE를 준수하는 IOSS 및 TSS에 대한 이식성/재사용성을 위한 구조 및 최적화 설계 방안을 기술하였다. 또한, 해당 설계가 ARINC 653 운용 환경에서 FACE를 적용하여 기능 및 성능이 만족한다는 것을 확인하였고, SIL을 통해 통합된 환경에서 본 논문의 설계가 정상 운용되는 것을 확인하였다. 추후 FACE 인증을 위해서는 데이터 모델링, FACE 적합성 시험 및 FACE에서 요구하는 산출물 작성이 필요하다.

Acknowledgments

본 논문은 산업통상자원부 재원으로 한국산업기술평가관리원(KEIT)의 지원을 받아 수행된 연구 결과입니다. [사업명: 항공우주부품기술개발사업-수출유망부품 및 핵심기술개발 / 과제명: 중소형 항공기급 개방형 항공전자 시스템 아키텍처 및 소프트웨어 개발 / 과제 고유번호: 20005378]

References

[1] Wang, Tiger, and G. U. Qingfan, “Research on distributed integrated modular avionics system architecture design and implementation,” in *2013 IEEE/AIAA 32nd Digital Avionics Systems Conference (DASC)*, New York: NY, pp.1-53, 2013.

[2] T. Joyce L, “A comparison of avionics open system architectures,” *ACM*, ACM SIGAda Ada Letters 36.2 pp. 22-26, 2017.

[3] Aeronautical Radio. Incorporated, Avionics Application software package Standard Interface Part 1 − Required Services. ARINC Specification 653P1-3, *Aeronautical Radio. Incorporated*, pp.11-49, Nov. 2010.

[4] Bloom, Gedare, and Joel Sherrill, "Harmonizing ARINC 653 and realtime POSIX for conformance to the FACE technical standard," in *2020 IEEE 23rd international symposium on real-time distributed computing (ISORC)*, Tennessee: TN, pp.98-105, 2020.

[5] The Open Group, Technical Standard Future Airborne Capability Environment (FACE™), Edition 2.1, *The Open Group*, pp.3-155, May 2014.

[6] The Open Group, Reference Implementation Guide for FACE™ Technical Standard, Edition 2.1, *The Open Group*, pp.4-101, Apr. 2016.

[7] Hyun-Chul Jo, Joo-Kwang Park, Hyun-Wook Jin, Hyung-Sik Yoon and Sang-Hun Lee, “Portable and configurable implementation of ARINC-653 temporal partitioning for small civilian UAVs,” *IEEE Access*, Vol.7, pp.142478-142487, Sep. 2019.

이 두 환 (Doo-Hwan Lee)
2012년 2월 : 충남대학교 전기공학과 (공학사)
2011년 10월 ~ 현재 : 엘아이지넥스원 선임연구원
※관심분야 : 항공전자, 소프트웨어 공용화 설계, 시스템 소프트웨어

남 영 욱 (Young-Uk Nam)
2003년 2월 : 부산대학교 항공우주공학과 (석사)
2012년 ~ 현재 : 엘아이지넥스원 수석연구원
※관심분야 : 항공전자 OFP, 개방형 소프트웨어 구조

조 경 연 (Kyeong-Yeon Cho)
2018년 2월 : 충남대학교 컴퓨터공학과 (석사)
2019년 8월 ~ 현재 : 엘아이지넥스원 선임연구원
※관심분야 : 항공전자 소프트웨어, 시스템 소프트웨어

윤 지 용 (Ji-Yong Yoon)
2006년 2월: 경북대학교 전자공학부
2005년 12월 ~ 2007년 10월: 엘지필립스LCD 사원
2007년 10월 ~ 현재: 엘아이지넥시원 수석연구원
※관심분야: 항공전자, 소프트웨어 공용화 설계, 시스템 소프트웨어