UCA와 필드버스를 이용한 전력 설비 자동화 통신망 구축 방안 연구

論 文
49D-6-1

A Study on the Implementation of Automation Network for Power Utility using UCA and Fieldbus

崔 仁 鎬* · 黃 仁 輝** · 洪 承 鎬***
(In-Ho Choi · In-Hui Hwang · Seung-Ho Hong)

Abstract - The UCA is a standard-based approach to utility communications which provides for wide scale integration of functional areas including customer interface, distribution, transmission, power plant, control center, and corporate information systems. Field devices in the power facilities require real-time communications, and they must be interconnected into fieldbus. This study introduces a method of implementing gateway that interfaces UCA and fieldbus. In this study, PROFIBUS and FOUNDATION Fieldbus are selected as candidate fieldbus protocols. Protocol interface in the gateway can be accomplished by mapping objects and services of UCA and fieldbus. Most of the UCA's CASM objects and services are directly mapped into those of the application and user layers of fieldbus. However, some CASM services are not supported in the application and user layers of fieldbus. This study presents the method of implementing those services. In order to show that the implementation of interface is possible, this study presents pseudo-codes of interface program that maps the objects and services of UCA and fieldbus.

Key Words : Power system, UCA, Fieldbus, Gateway

1. 서 론

지난 수년간 컴퓨터와 통신 기술이 급격히 발전함에 따라 기술 선진국에서는 전력 설비에 정보화 기술을 기반으로 하는 자동화 시스템의 도입을 매우 활발히 추진하고 있다. 기존의 시스템은 특정 설비 제조 업체가 제공하는 폐쇄적인 통신 프로토콜을 사용함에 따라 설비들 간의 통신에 많은 어려움이 있었다. 이러한 문제점을 해결하기 위하여 미국의 EPRI(Electric Power Research Institute)에서는 1988년부터 UCA(Utility Communications Architecture) 개발 사업을 시작하였다. UCA[1-7]는 발전소를 비롯하여, 변전소, 송배전소의 모든 전력 설비들 간에 실시간 통신뿐만이 아니라, 전력 회사와 고객까지를 포함하여 전력과 관련된 모든 통신 요구 사항을 만족시키기 위하여 개발된 전력 설비용 개방형 통신 프로토콜이다.

UCA는 (i) 발전소의 DCS(Distributed Control System), EMS(Energy Management System), SCADA(Supervisory Control and Data Aquisition) 시스템 등의 전력 설비들 간에 실시간 데이터 교환을 지원하기 위한 통신 프로토콜인 TASE.2(Telecontrol Application Service Element 2, 또는 ICCP: Inter-Control Center Communications Protocol)[5-7]와 (ii) 발전소, 변전소, 송배전소 등의 전력 설비의 필드에 설치된 Remote Terminal Unit, Switch, Voltage Regulator/Tap Changer, Recloser 및 Capacitor Bank Controllers 등의 많은 필드 장치들을 모델링하기 위한 GOMSFE(Generic Object Models for Substation and Feeder Equipment)[4]라는 장비 객체 모델과 이들에 공통으로 사용되는 통신 서비스를 정의하는 CASM(Common Application Service Models)[3]으로 구성된다.

그림 1에는 전력회사, 오퍼레이션 제어센터, 발전소, 송전소, 배전소, 고객 인터페이스 등의 모든 전력 관련 설비가 UCA 통합 네트워크를 통하여 구축된 통합 시스템 구성도의 예가 나타나 있다. 그림 1에서 발전소, 변전소, 송배전소와 같은 전력 설비의 필드에 설치된 각종 센서, 제어기 및 컴퓨터들은 매우 열악한 환경에서 동작되며, 또한 주어진 시간 내에 데이터 전송이 완료되어야 하는 실시간 통신 요구 조건을 만족하여야 한다. 따라서 이러한 필드 장비들은 플랜트 내의 기간통신망인 LAN 또는 WAN에 직접 접속될 수가 없다. UCA에서는 하부 통신망으로 특정 프로토콜을 사용하도록 지정하지 않고 있으나, 필드 장비들간의 실시간 통신을 위하여서는 자동화 시스템 전용 산업통신망인 필드버스를 사용하여야 한다. 발전 설비를 원격으로 실시간 모니터링 및 제어하기 위하여서는 UCA와 필드버스간에 연동이 이루어져야 하며, 본 연구에서는 이를 위하여 UCA와 필드버스의 인터페이스 방안을 제시한다.

필드버스가 UCA에 바로 접속될 수 있으면 통합 정보망의 구축은 용이해질 것이다. 그러나 현존하는 필드버스에서는 UCA에서 실시간 데이터 수집과 제어를 위해 응용계층 프로토콜로 채택한 MMS(Manufacturing Message

* 準 會 員 : 漢陽大 制御計測工學科 碩士課程
** 正 會 員 : 漢陽大 制御計測工學科 博士課程
*** 正 會 員 : 漢陽大 電子컴퓨터工學部 副敎授 · 工博
接受日字 : 1999年 12月 22日
最終完了 : 2000年 5月 18日

Specification)[8-9]를 지원하지 않는다. 또한 필드버스의 사용자 계층에서 정의된 기능 블록(Function Blocks)[14]은 일반적인 자동화 환경에서 사용되는 객체 모델과 서비스 방식을 지정하고 있으며, 이러한 모델들은 전력 설비라는 특정 응용 분야에서 사용되는 필드 장비를 위하여 정의된 GOMSFE/CASM의 장비 객체 모델 및 서비스와는 다른 구조를 가진다. 본 연구에서는 필드버스와 UCA의 통합 운용 환경을 구축하기 위한 기초 연구로 필드버스-UCA 게이트웨이 구성 방안을 제시한다.

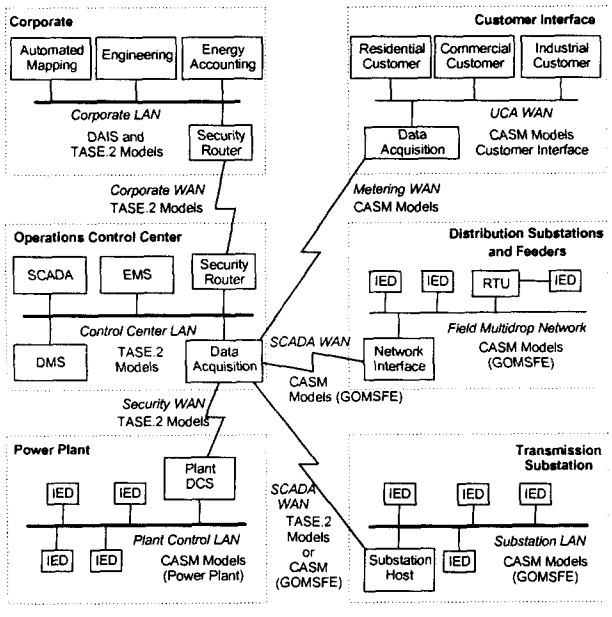


그림 1 UCA 통합 네트워크의 구성도의 예

Fig. 1 Example of UCA integrated network

UCA를 사용하는 전력설비에서 사용자는 UCA의 사용자 계층에서 제공하는 서비스를 이용하여 필드 디바이스의 구동을 위한 응용 프로세스를 구현한다. 따라서 UCA의 하부 계층으로 필드버스를 도입하는 경우 사용자가 UCA 환경에서 구현한 응용 프로세스는 필드버스의 응용계층과 사용자 계층 프로토콜에 자동적으로 인터페이스 되어야 한다. 즉, UCA 사용자에게는 하부계층의 통신망 프로파일에 관계없이 UCA 사용자계층에서 제공하는 서비스만을 이용하여 응용 프로세스를 구현할 수 있는 환경이 제공되어야 한다. 이를 위하여서는 UCA의 사용자계층과 필드버스의 응용계층 및 사용자계층간에 인터페이스 기술이 확보되어야 한다.

본 연구에서는 현존하는 여러 종류의 필드버스들 가운데 유럽지역에서 개발되어 현재 전세계적으로 가장 높은 시장 점유율을 보이고 있는 PROFIBUS[10-12]와 최근에 미주지역에서 개발이 완료되어 앞으로 널리 사용될 것으로 예상되며, 가장 빠른 시장 증가율을 보이고 있는 FOUNDATION Fieldbus[13-14]를 대상으로 UCA와 필드버스의 인터페이스 방안을 제시한다. PROFIBUS와 FOUNDATION Fieldbus는 모두 응용계층 프로토콜로 FMS(Fieldbus Message Specification)를 사용하고 있다. 또한, PROFIBUS는 공정제어를 위한 사용자계층 프로토콜로 PROFIBUS-PA를 정의하고 있으며, FOUNDATION Fieldbus는 사용자계층에서 기능 블록, 시스템 관리 프로토콜 등을 정의하고 있다. 본 논문의 2장에서는 UCA와 필드버스의 인터페이스 기능을 수행하는 게이트웨이의 물리적 구성 방안에 대하여 기술하고, 3장에서는 게이트웨이 내에서 UCA와 필드버스 프로토콜의 인터페이스 방안에 대하여 기술한다. 4장에서는 인터페이스 S/W를 실제로 구현하기 위하여 UCA와 필드버스의 통신 객체 및 서비스를 매핑하는 방법에 대하여 기술하며, 이를 실제로 구현할 수 있는 의사 코드(Pseudo-code)의 예를 제시한다. 마지막으로 5장에서는 본 논문의 결론과 추후 연구사항이 기술된다.

2. 게이트웨이 구성 방안

UCA와 필드버스간에 인터페이스 구축은 전력설비의 규모가 소규모인가 또는 대규모인가에 따라 두 가지 경우의 물리적 구성으로 구분할 수 있다. 먼저 변전소, 송배전소와 같은 소규모 전력설비의 경우에는 필드 디바이스들이 필드버스에 연결되고, 필드버스는 바로 WAN에 접속되어 원격지의 제어센터와 통신 기능을 수행하여야 한다. 그림 2에는 소규모 전력 설비에서 UCA 및 필드버스 통신망의 물리적 구성이 나타나 있다. 제어센터의 LAN은 WAN과의 접속을 위해 LAN-WAN 게이트웨이를 필요로 하며, 전력설비의 필드버스는 WAN과의 접속을 위해 WAN-필드버스의 게이트웨이를 필요로 한다.

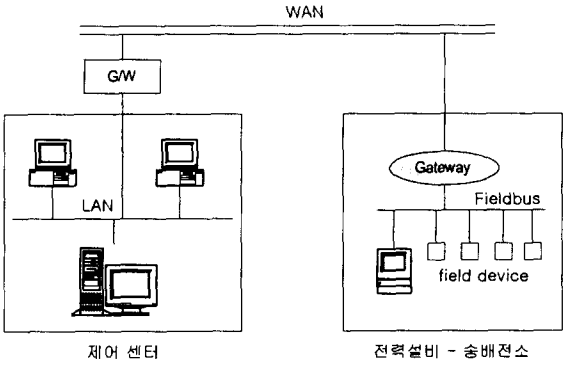


그림 2 소규모 전력설비에서 통신망의 물리적 구성

Fig. 2 Physical configuration of communication network in the small size facility

제어센터의 UCA와 전력설비에서 사용되는 필드버스의 구조 및 프로토콜간에 공통점이 많다면 이들간의 인터페이스는 그만큼 용이해질 것이다. 그러나 이들간의 통신망 구조 및 계층별 사용 프로토콜은 크게 다르다. 제어센터에서는 전력설비들 간에 실시간 통신을 위하여 사용자계층에서 UCA의 TASE.2 프로토콜을 필요로 하며, TASE.2는 MMS 프로토콜 상에서 동작된다. 그러나 전력설비의 필드버스에는 필드에 설치된 디바이스들을 실시간으로 제어하기 위한 자체 사용자계층 프로토콜을 사용하며, 이 사용자계층 프로토콜은 FMS 프로토콜을 응용계층으로 사용한다. 따라서 이들간의 인터페이스는 그림 3에 나타난 바와 같이 WAN-필드버스 게이트웨이를 거쳐 수행되어야 한다.

전력 설비에 UCA를 도입하는 경우 전력 설비의 필드버스 디바이스의 실시간 제어 기능은 CASM 프로토콜을 통하여 수행된다. UCA를 사용하는 제어센터에서 CASM은 MMS 상에서 동작되나, 필드버스에서는 CASM에 대한 통신 서비스가 MMS 대신에 필드버스의 응용계층인 FMS 서비스 및 필드버스 사용자계층 서비스를 이용하여 수행되어야 할 것이다. 따라서 UCA와 필드버스의 게이트웨이를 구현하는 경우에 CASM과 FMS/사용자계층의 매핑 작업이 필연적으로 요구된다.

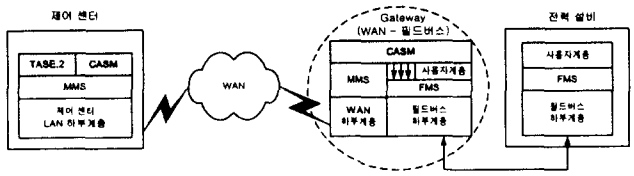


그림 3 소규모 전력설비에서 통신망 프로토콜간의 인터페이스

Fig. 3 Protocol interface in the small size facility

제어센터의 사용자계층에는 CASM 이외에 TASE.2가 탑재되며, 제어센터와 전력설비간의 실시간 데이터 교환을 위하여 TASE.2와 CASM의 인터페이스 작업이 요구된다. TASE.2와 CASM의 인터페이스는 게이트웨이 내에서도 수행될 수 있으나, 이러한 경우에는 게이트웨이에서 과다한 연산작업을 초래하게 된다. 이는 UCA-필드버스 게이트웨이를 실제로 개발하는 경우 고성능의 컴퓨터를 필요로 하게 되어 게이트웨이의 가격 상승 요인으로 작용할 것이다. 따라서 게이트웨이 내에서 연산작업이 최소화되도록 하기 위하여 게이트웨이에는 CASM만 탑재하고 TASE.2와 CASM의 인터페이스는 제어센터의 컴퓨터 내에서 처리되도록 하는 것이 바람직하다. TASE.2와 CASM은 모두 기본적으로 MMS 상에서 동작되므로 제어 센터의 컴퓨터 내에서 TASE.2와 CASM의 인터페이스 기능은 쉽게 구현될 수 있다.

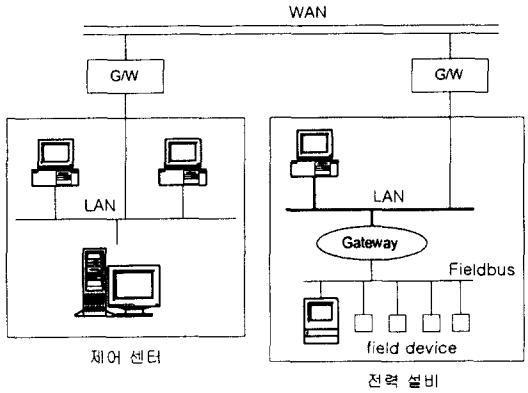


그림 4 대규모 전력설비에서 통신망의 물리적 구성

Fig. 4 Physical configuration of communication network in the large size facility

발전소와 같은 대규모 전력설비에는 전력설비 내에 기간 통신망으로 LAN이 설치된다. 이러한 경우에 필드 디바이스들간에 통신을 담당하는 필드버스는 전력설비 LAN과 접속되어야 하며, 원격지의 제어센터와 전력설비간의 통신은 전력설비 LAN과 제어센터 LAN을 연결하는 WAN을 통하여 수행되어야 한다. 그림 4에는 대규모 전력 설비에서 통신망의 물리적 구성이 나타나 있다. 제어센터 LAN과 전력설비 LAN은 WAN과의 접속을 위해 LAN-WAN 게이트웨이를 필요로 하며, 전력설비의 LAN은 필드버스와의 접속을 위해 LAN-필드버스의 게이트웨이를 필요로 한다.

제어센터와 발전소의 LAN에 접속된 장비들은 실시간 통신을 위하여 사용자계층에서 UCA의 TASE.2 프로토콜을 필요로 한다. 또한, 전력설비의 필드버스에 설치된 디바이스들을 실시간으로 제어하기 위하여서는 UCA의 CASM 프로토콜이 사용되어야 한다. 따라서 이들간에 인터페이스는 그림 5에 나타난 바와 같은 과정을 거쳐 수행된다. 그림 5에서 제어센터의 LAN과 전력설비의 LAN은 게이트웨이를 통하여 WAN과 연결된다. 전력설비에서 전력설비 LAN과 필드버스간에 메시지 변환은 LAN-필드버스 게이트웨이를 통하여 수행된다. 그림 3과 5에서 나타난 바와 같이 UCA/필드버스 게이트웨이를 구성하는 경우 전력설비의 규모에 상관없이 동일한 구조의 게이트웨이가 사용될 수 있을 것이다. 단지 소규모와 대규모 전력설비의 게이트웨이에서 차이가 나는 부분은 LAN 또는 WAN 쪽 연결부의 하부계층 통신 프로파일이다.

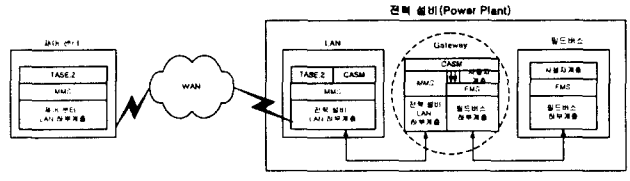


그림 5 대규모 전력설비에서 통신망 프로토콜간의 인터페이스

Fig. 5 Protocol interface in the large size facility

3. 프로토콜 인터페이스 방안

UCA의 CASM 프로토콜은 실시간 데이터 수집 및 교환을 위해 MMS를 응용계층으로 사용한다. 그러나 필드버스에서는 응용계층 이외에 추가적으로 사용자 계층을 정의하고 있기 때문에 UCA를 필드버스에 인터페이스시키는 경우 필드버스의 응용계층 또는 사용자계층의 어느 한쪽으로만 매핑하는 것은 사실상 불가능하다. 따라서 UCA와 필드버스의 인터페이스는 다음의 세부분으로 구분하여 수행되어야 한다.

- 필드버스의 응용계층으로 직접 매핑이 가능한 부분
- 응용계층에는 직접 매핑되지 않으나, 사용자계층에 직접 매핑 가능한 부분
- 응용계층과 사용자계층 양쪽 어느 계층으로도 매핑되지 않는 부분

필드버스는 응용계층에서 FMS를 사용하는데 이 FMS는 MMS에서 제공하는 통신서비스의 일부분이 지원되지 않는다. 따라서 FMS 또는 사용자계층으로 직접 매핑이 가능한 부분에 대하여서는 먼저 객체모델과 통신서비스들에 대한

매핑테이블을 작성하고, 매핑테이블에 의한 인터페이스 프로그램 코딩 작업을 수행함으로써 인터페이스 S/W를 구현할 수 있으나, 필드버스의 FMS와 사용자계층 양쪽 어느 계층으로도 매핑되지 않는 부분에 대하여서는 인터페이스를 위한 추가 서비스를 직접 구현하여야 한다. 다음의 3.1절과 3.2절에는 FOUNDATION Fieldbus와 PROFIBUS-PA에 대하여 응용계층으로 매핑되어야 할 부분, 사용자계층으로 매핑되어야 할 부분 및 추가적으로 구현되어야 할 부분을 분류하여 표로 제시하였다.

3.1 UCA와 FOUNDATION Fieldbus의 인터페이스

UCA CASM의 "디바이스 모델" 가운데 FMS 및 사용자계층과 매핑되어야 하는 것들에는 LogicalDevice, DataObject, DataSets, FunctionalComponent 등이 있다. 이러한 디바이스 모델들은 CASM의 서버 모델 가운데 하나인 Data Access Service Model을 통하여 데이터를 교환한다. 또한, CASM의 "서버 모델" 가운데 FMS와 매핑되는 것들에는 Server, Association, Report, Device Control, Multicast 등이 있다. 본 절에서는 CASM의 디바이스 모델 및 서버 모델에 대하여 FOUNDATION Fieldbus의 응용계층인 FMS 및 사용자계층 간의 인터페이스 관계를 표로서 제시하였다. 본 절의 표에서 사용자 계층으로 매핑되는 부분은 다시 시스템관리로 매핑되는 부분과 기능블록으로 매핑되는 부분으로 구분된다. 표에서 이탤릭으로 표시된 부분은 객체 모델 및 통신서비스가 직접 매핑되지 않아 추가적으로 구현되어야 할 부분을 나타낸다.

표 1에는 CASM의 디바이스 모델인 LogicalDevice 모델을 FOUNDATION Fieldbus의 응용계층인 FMS에 매핑시키는 관계가 나타나 있다. 표1에서 보는 바와 같이 LogicalDevice 모델의 경우 LogicalDevice 객체 클래스는 FMS의 Domain 객체로 매핑되어야 하며, LogicalDevice 모델의 서비스인 GetDataObjectList, GetDataSetsList, CreateLogicalDevice 서비스는 각각 FMS에서 제공하는 GetOD, GetOD, InitiateDownloadSequence 서비스로 매핑되어야 한다. 그러나 FMS에서는 DeleteLogicalDevice 서비스를 지원하지 않기 때문에 이를 위하여서는 DeleteDomain 서비스가 추가로 구현되어야 한다. 표 1에 제시된 UCA와 FOUNDATION Fieldbus의 객체 및 응용계층과 사용자계층에서 제공하는 서비스는 해당 규격서에 상세히 기술되어 있으므로 규격서를 참조하기 바란다.

표 1 디바이스 모델 - LogicalDevice 모델의 매핑
Table 1 Mapping of Device Model - LogicalDevice Model

UCA CASM 디바이스 모델		FOUNDATION Fieldbus		
모델 종류	객체 클래스	응용 계층	사용자 계층	
	서비스	FMS	SM	FB
Logical Device	LogicalDevice	Domain		
	GetDataObjectList	GetOD		
	GetDataSetsList	GetOD		
	CreateLogicalDevice	InitiateDownloadSequence		
	DeleteLogicalDevice	*DeleteDomain*		

표 2에는 DataObject 모델의 경우에 대한 CASM과 FOUNDATION Fieldbus 간에 매핑 관계가 나타나 있다. DataObject 모델의 경우 DataObject, DeviceIdentify, DeviceModel 객체들은 FMS의 Named Variable 객체로 직접 매핑된다. 그러나, FunctionalComponents의 객체 가운데 일부는 FMS 보다는 사용자계층의 AI(Analog Input), AO(Analog Output), DI(Discrete Input), DO(Discrete Output), Alert Analog 블록에 직접 매핑되는 것이 유리하다. 표 3에는 FunctionalComponents 기능 그룹들을 필드버스의 응용계층 또는 사용자계층에 매핑하는 방안이 나타나 있다. DataObject 모델의 GetDataObjectValues, SetDataObjectValues, SetDataObjectValues(unconfirmed), GetDataObjectAttributes 서비스들은 각각 FMS의 변수 접속 및 이벤트관리 기능에서 제공하는 Read, Write, InformationReport, ReadWithType 서비스들과 직접 매핑된다. 그러나 CreateDataObject 및 DeleteDataObject에 해당하는 서비스는 FMS에서 제공하지 않기 때문에 DefineNamedVariable, DeleteVariableAccess 서비스를 추가로 구현하여야 한다.

표 2 디바이스 모델 - DataObject 모델의 매핑
Table 2 Mapping of device model - dataobject model

UCA CASM 디바이스 모델		FOUNDATION Fieldbus		
모델 종류	객체 클래스	응용 계층	사용자 계층	
	서비스	FMS	SM	FB
Data Object	DataObject	Named Variable		
	DeviceIdentify	Named Variable		
	DeviceModel	Named Variable		
	FunctionalComponents	표 3 참조		AI/AO/ DI/DO Alert-Analog
	GetDataObjectValues	Read		
	SetDataObjectValues	Write		
	SetDataObjectValues (unconfirmed)	InformationReport		
	GetDataObjectAttributes	ReadWithType		
	CreateDataObject	*DefineNamedVariable*		
	DeleteDataObject	*DeleteVariableAccess*		

표 4~14까지는 CASM의 디바이스 모델과 서버 모델 가운데 DataSets, Server, Association, Report, Device Control, Multicast 모델을 FOUNDATION Fieldbus의 응용계층과 사용자계층의 객체 및 통신서비스에 인터페이스시키는 관계가 나타나 있다. 표 4~14에서 CASM과 FOUNDATION Fieldbus의 객체와 서비스의 매핑 관계에 대한 설명은 표 1과 2에서 기술한 것과 동일하게 설명될 수 있으므로 생략한다.

표 3 FunctionalComponents 기능그룹들의 매핑
Table 3 Mapping of functionalcomponents

UCA CASM		FOUNDATION Fieldbus		
Functional Component	공통 클래스	응용계층	사용자계층	
		FMS	SM	FB
AX	Tag		PD_Tag	FB Tag
EV	ECB	Named Variable		
LG	LCB	Named Variable		
RP	RCB	Named Variable		
CF	AC	Named Variable		
	ARC	Named Variable		
CO	BOP	Named Variable		
	BOPS	Named Variable		
	BOL			DO Block
	BOLS	Named Variable		
DC	d	Named Variable		
	EqRtg	Named Variable		
	ConCkt	Named Variable		
MX	AI			AI Block/ Alert Analog Block
	Acl	Named Variable		
	III	Named Variable		
	PfgIII	Named Variable		
SP	AO	Named Variable		AO Block
	AOS			
	AISP			AO Block
ST	SI			DI Block
	SIT			DI Block
	SIG			DI Block
AS	Association	Context Management		

표 4 디바이스 모델 - DataSet 모델의 매핑
Table 4 Mapping of device model - dataSet model

UCA CASM 디바이스 모델		FOUNDATION Fieldbus		
모델 종류	객체 클래스	응용계층	사용자 계층	
	서비스	FMS	SM	FB
Data Sets	DataSet	NamedVariableList		
	GetDataSetElementNames	*GetNamedVariableList Attributes*		
	GetDataSetValues	Read		
	CreateDataSet	DefineVariableList		
	DeleteDataSet	DeleteVariableList		

표 5 서버 모델 - Server 모델의 매핑
Table 5 Mapping of server model - server model

UCA CASM 서버 모델		FOUNDATION Fieldbus		
모델 종류	객체 클래스	응용계층	사용자계층	
	서비스	FMS	SM	FB
Server	Server	VFD		
	GetLogicalDeviceList	GetOD		
	GetCapabilities	*GetCapabilityList*		

표 6 서버 모델 - Association 모델의 매핑
Table 6 Mapping of server model - association model

UCA CASM 서버 모델		FOUNDATION Fieldbus		
모델 종류	객체 클래스	응용계층	사용자 계층	
	서비스	FMS	SM	FB
Association	Associatoin	Association		
	Initiate	Initiate		
	Conclude	*Conclude*		
	Abort	Abort		

표 7 서버 모델 - Reporting Service(Event) 모델의 매핑
Table 7 Mapping of server model - reporting service (event) model

UCA CASM 서버 모델		FOUNDATION Fieldbus		
모델 종류	객체 클래스	응용계층	사용자 계층	
	서비스	FMS	SM	FB
Report	EventControlBlock	NamedVariable		
	EventInfo	NamedVariable		
	GetEventControlBlock	Read		
	SetEventControlBlock	Write		
	CreateEventControlBlock	*DefineNamedVariable*		
	DeleteEventControlBlock	*DeleteVariableAccess*		
	GetEventControlBlockNames	GetOD		
	EventNotification	EventNotification		
	EnrollforNotification	*DefineEventEnrollment*		
	DisEnrollforNotification	*DeleteEventEnrollment*		

표 8 서버 모델-Reporting Service(Report) 모델의 매핑
Table 8 Mapping of server model-reporting service (report) model

UCA CASM 서버 모델		FOUNDATION Fieldbus		
모델 종류	객체 클래스	응용계층	사용자 계층	
	서비스	FMS	SM	FB
Report	ReportControlBlock	NamedVariable		
	GetReportControlBlock	Read		
	SetReportControlBlock	Write		
	CreateReportControlBlock	*DefineNamedVariable*		
	DeleteReportControlBlock	*DeleteVariableAccess*		
	GetReportControlBlockNames	GetOD		
	Report	InformationReport		

표 9 서버 모델 - Reporting Service(Log) 모델의 매핑
Table 9 Mapping of server model - reporting service (log) model

UCA CASM 서버 모델		FOUNDATION Fieldbus		
모델 종류	객체 클래스	응용계층	사용자 계층	
	서비스	FMS	SM	FB
Report	LogControlBlock	NamedVariable		
	GetLogControlBlock	Read		
	SetLogControlBlock	Write		
	CreateLogControlBlock	*DefineNamedVariable*		
	DeleteLogControlBlock	*DeleteVariableAccess*		
	GetLogControlBlockNames	GetOD		
	QueryLog	*ReadJournal*		
	EmptyLog	*InitializeJournal*		

표 10 서버 모델 - Device control service(direct control) 모델의 매핑

Table 10 Mapping of server model - device control service(direct control) model

UCA CASM 서버 모델		FOUNDATION Fieldbus		
모델 종류	객체 클래스	응용계층	사용자 계층	
	서비스	FMS	SM	FB
Device Control	DirectControl	NamedVariable		
	GetDirectControlObjectNames	GetOD		
	DirectControlOperation	Write		

표 11 서버 모델 - Device control service(select before operate) 모델의 매핑

Table 11 Mapping of server model - device control service(select before operate) model

UCA CASM 서버 모델		FOUNDATION Fieldbus		
모델 종류	객체 클래스	응용계층	사용자 계층	
	서비스	FMS	SM	FB
Device Control	SBO	NamedVariable		
	GetSBOObjectNames	GetOD		
	Select	Read		
	Deselect	Write		

표 12 서버 모델 - Device control service(time activated control) 모델의 매핑

Table 12 Mapping of server model - device control service(time activated control) model

UCA CASM 서버 모델		FOUNDATION Fieldbus		
모델 종류	객체 클래스	응용계층	사용자 계층	
	서비스	FMS	SM	FB
Device Control	TimeActivatedControl	NamedVariable		
	GetTimeActivatedControlObjectNames	Read		
	TimeActivatedAcknowledgementReport	InformationReport		
	TimeActivatedErrorReport	InformationReport		

표 13 서버 모델 - Device control service(device tagging) 모델의 매핑

Table 13 Mapping of server model - device control service(device tagging) model

UCA CASM 서버 모델		FOUNDATION Fieldbus		
모델 종류	객체 클래스	응용계층	사용자계층	
	서비스	FMS	SM	FB
Device Control	Tag		PD_Tag	FB Tag
	-			

표 14 서버 모델 - Multicast 모델의 매핑

Table 14 Mapping of server model - multicast model

UCA CASM 서버 모델		FOUNDATION Fieldbus		
모델 종류	객체 클래스	응용계층	사용자 계층	
	서비스	FMS	SM	FB
Multicast	-			
	SetDataObjectValues (unconfirmed)	InformationReport		

CASM 서버 모델 가운데 local time을 나타내는 Time 모델의 객체는 DataObject의 특별한 경우라 할 수 있다. 따라서 기본적으로 Time 모델은 표 2에 나타난 DataObject 모델의 서비스를 사용한다. 그러나 다중 서버와 디바이스의 local time 파라미터를 동기화 시키기 위하여서는 표 15와 같은 추가 서비스가 제공되어야 한다. 이들은 다른 CASM 서비스처럼 서버-클라이언트 관계에 의한 두 시스템간의 서비스가 아니라, 서버 모델이나 디바이스 모델 내에서 사용되는 time을 동기화 시키기 위하여 내부적으로 사용되는 서비스이므로 필드버스의 FMS나 사용자 계층과는 매핑은 불필요하며, 필드 디바이스의 응용 프로세스에 추가적으로 구현되어야 한다.

표 15 서버 모델 - Time 모델의 매핑

Table 15 Mapping of server model - time model

UCA CASM 서버 모델		FOUNDATION Fieldbus		
모델 종류	오브젝트 클래스	응용계층	사용자계층	
	서비스	FMS	SM	FB
Time				
	Time Synchronization Indication			
	Configure Time Synchronization			
	Request Time Synchronization			
	Request Time Synchronization Indication			
	Perform Time Synchronization			

3.2 UCA와 PROFIBUS-PA의 인터페이스

PROFIBUS-PA와 FOUNDATION Fieldbus는 모두 응용계층으로 FMS를 사용하고 있으며, 사용자계층 역시 매우 유사한 구조를 가지고 있다. 따라서 UCA와 PROFIBUS-PA 간의 인터페이스에서 FMS와 인터페이스 되는 부분은 3.1절에서 기술된 FOUNDATION Fieldbus의 경우와 동일하며, 본 절에서는 사용자계층에서 인터페이스되는 부분에 대하여서만 기술하기로 한다. 표 16-17에는 CASM의 디바이스 및 서버 모델과 PROFIBUS-PA의 사용자계층간에 매핑 방안이 제시되어 있다.

표 16 Functionalcomponents 기능그룹들의 매핑
Table 16 Mapping of functionalcomponents

UCA CASM		PROFIBus	
Functional Component	공통 클래스	응용계층 FMS	사용자계층 PA
AX	Tag		PD_TAG / AF_TAG
EV	ECB	Named Variable	
LG	LCB	Named Variable	
RP	RCB	Named Variable	
CF	AC	Named Variable	
	ARC	Named Variable	
CO	BOP	Named Variable	
	BOPS	Named Variable	
	BOL		Discrete Structure Block Object
	BOLS		Discrete Structure Block Object
DC	d	Named Variable	
	EqRtg	Named Variable	
	ConCkt	Named Variable	
MX	AI		Discrete Structure Floating Point Structure Bitstring Structure Block Object
	Acl		Discrete Structure Floating Point Structure Bitstring Structure Block Object
	III		Bitstring Structure Block Object
	PfgIII	Named Variable	
SP	AO		Discrete Structure Floating Point Structure Block Object
	AOS		Discrete Structure Floating Point Structure Block Object
	AISP		Floating Point Structure Block Object
ST	SI		Bitstring Structure Block Object
	SIT		Bitstring Structure Block Object
	SIG		Bitstring Structure Block Object
AS	Associa-tion	ContextManagement	

표 17 Device control service(device tagging) 서버 모델의 매핑
Table 17 Mapping of device control service(device tagging) server model

UCA CASM 서버 모델		PROFIBUS	
모델 종류	객체 클래스 서비스	응용계층 FMS	사용자계층 PA
Device Control	Tag		PD_TAG / AF_TAG
	-		

4. 매핑테이블 및 추가 서비스 구현

3장에서는 CASM에서 지원하는 서버 모델과 디바이스 모델들에 대하여 이들이 각각 FOUNDATION Fieldbus와 PROFIBUS-PA의 응용계층 및 사용자계층의 어떤 객체와 어떤 서비스에 매핑되어야 할 것인가에 대하여 기술하였다. 본 절에서는 UCA의 CASM 프로토콜과 필드버스의 응용계층 및 사용자계층간에 매핑이 실제로 어떻게 이루어지는가를 제시하기 위하여 3장에서 제시한 매핑 관계에 대한 구체적인 매핑테이블 작성 및 추가 서비스 구현 방안에 대하여 기술한다. 4.1절에서는 CASM과 FMS의 인터페이스 방안이 기술되며, 4.2절에는 CASM과 필드버스의 사용자계층 인터페이스 방안이 기술된다.

4.1 UCA CASM과 FMS의 인터페이스

본 연구에서는 UCA 규격에 제시되어 있는 CASM과 MMS간에 매핑방법을 기반으로 하여, CASM과 FMS 간에 직접 매핑이 가능한 부분의 인터페이스 방안을 제시한다. 또한, CASM과 FMS간에 직접 매핑이 이루어지지 않는 부분에 대하여서는 추가적으로 서비스를 구현하는 방법을 제시한다. 추가적인 서비스의 구현은 MMS에서 제공하는 통신서비스 방식을 따르도록 하였다. 본 논문에서는 직접 매핑 되는 부분과 추가로 구현되어야 하는 부분에 대하여 각각 대표적인 경우를 선정하여 인터페이스 방법을 기술하며, 나머지 CASM의 서버 모델과 디바이스 모델에 대한 서비스 매핑 및 데이터형 모델 매핑은 참고문헌 [15]를 참고하기 바란다.

4.1.1 CASM-FMS 서비스 직접 매핑

CASM에 직접 매핑되는 FMS 서비스는 FMS가 CASM의 통신서비스를 다른 추가적인 조작 없이 직접 제공해 주는 경우이다. 즉, FMS와 MMS간에 통신서비스가 일치되는 경우이다. 본 논문에서 CASM과 FMS의 직접 매핑은 다음의 두 단계의 매핑 테이블을 통하여 기술된다.

- CASM과 FMS간에 객체 클래스의 속성에 대한 매핑 및 해당 객체 클래스에서 사용하는 서비스들의 매핑
- 해당 객체 클래스에서 사용되는 서비스에 대해 서비스 프리미티브의 속성들에 대한 매핑

표 18 DataObject 모델의 CASM-FMS 매핑
Table 18 CASM-FMS mapping of dataobject model

Object Class	FMS Class	Server	Client
DataObject	Named Variable	m	m
Attribute	FMS Object / Attribute		
DOReference	Simple Variable Name	m	m
Ancestry	visible-string	m	o
Scope	DOMAIN, VFD, CM		
DataType	Type Specification	m	m
Value	Data	m	m
AccessControl	Access Method		
Deletable	Deletable		
LocalResource	Address		
Services	FMS Services		
GetDataObjectValues	Read	m	m
SetDataObjectValues	Write	o	o
SetDataObjectValues (Unconfirmed)	InformationReport	o	o
GetDataObjectAttributes	ReadWithType	m	m
CreateDataObject	*DefineNamedVariable*	o	o
DeleteDataObject	*DeleteVariableAccess*	o	o

다음에는 직접 매핑의 대표적인 예로 표 2에 주어진 DataObject 디바이스 모델에 대한 CASM-FMS 매핑 방법을 기술한다. DataObject 모델은 필드 장비 데이터와 모델 내에서 특별히 요구되는 데이터들에 대한 구성과 이름을 표현하는 실제 디바이스의 추상적인 요소이다. 표 18에서는 CASM과 FMS의 DataObject의 클래스 속성의 매핑과 DataObject의 서비스에 대한 FMS 서비스의 매핑관계를 보여준다. 표 18, 19에서 "m"(mandatory)으로 표시된 항목은 표준화된 객체 모델에서 필수적으로 구현되어야 함을 의미하며, "o"(optional)로 표시된 항목은 그 항목이 구현이 되거나 또는 그렇지 않아도 되는 선택적 사항임을 의미한다. "c"(conditional)로 표시된 항목은 규격에서 참조되는 특정한 조건하에서 지원됨을 의미한다.

CASM의 DataObject 객체 클래스는 FMS의 Named Variable 객체에 대응된다. DataObject의 객체 속성인 DOReference, Ancestry, Scope, DataType, Value, AccessControl, Deletable, LocalResource는 각각 FMS의 Named Variable이 가져야할 속성인 Simple Variable Name, visible-string, (DOMAIN, VFD, CM), Type Specification, Data, Access Method, Deletable, Address와 직접적으로 매핑된다. 이 가운데 DOReference, DataType, Value 속성들은 서버와 클라이언트에 모두 반드시 구현되어야 하며, Ancestry 속성은 서버에 반드시 구현되어야 한다.

CASM에서 DataObject 모델에 제공되는 서비스에는 GetDataObjectValues, SetDataObjectValues, SetDataObject Values(Unconfirmed), GetDataObjectAttributes가 있으며, 이들은 각각 FMS의 Read, Write, InformationReport, ReadWithType 서비스와 직접 매핑된다. 그러나 FMS는 CASM에서 요구하는 CreateDataObject과 DeleteData-Object 서비스를 제공하지 않는다. 따라서 이러한 서비스들은 DefineNamedVariable과 DeleteNamedVariable이라는 이름의 새로운 서비스로 구현되어야 한다. DataObject 모델에 제공되는 서비스들 가운데 GetDataObjectValues와 GetDataObjectAttributes는 반드시 서버와 클라이언트에 모두 구현되어야 한다.

표 19에는 DataObject 모델에서 사용되는 서비스 가운데 GetDataObjectValues 서비스에 대하여 CASM과 FMS간에 서비스 프리미티브의 속성들에 대한 매핑 테이블이 나타나 있다. CASM의 GetDataObjectValues 서비스는 FMS의 Read 서비스에 매핑된다. GetDataObjectValues 서비스는 Request 프리미티브와 Response 프리미티브를 통하여 수행된다. Request 프리미티브의 SpecificationWithResult 파라미터는 BOOL 값을 갖는데 응답 프리미티브가 발생 시 response+ 파라미터에 AccessSpecification Parameter가 요구되는지를 나타낸다. 즉 TRUE 값을 가지면 Response+에 AccessSpecification Parameter를 포함하고 FALSE 값을 가지면 포함하지 않는다. CASM의 ListOfDataObjects는 FMS Read 서비스의 AccessSpecification Parameter에 대응되는데 FMS에서는 변수 접근시 사용자가 객체사전에 OD(Object Description)에서 정의한 방식, 즉 Index, Variable Name, Variable List Name 중 1 가지로 접근 가능하다. Response+의 Value 속성은 읽어온 FMS의 Data에 매핑된다. Response-는 미리 지정된 에러 코드 중 적당한 것이 선택되어 응답된다.

표 19 GetDataObjectValues 서비스의 CASM-FMS 매핑
Table 19 CASM-FMS mapping of getDataobjectValues service

Service: GetDataObjectValues	FMS Services/Argument/Results	Server	Client
Request SpecificationWithResult	Read Request SpecificationWithResult	m	o
ListOfDataObjects	AccessSpecification Parameter = Index, Variable Name, Variable List Name	m	m
Response+	Read Response		
ListOfDataObjects	AccessSpecification Parameter = Index, Variable Name, Variable List Name	c	c
Values	Data	m	m
Response-	error code 참조		

표 18의 SetDataObjectValues, SetDataObjectValues(Unconfirmed), GetDataObjectAttributes 서비스도 표19에 나타난 GetDataObjectValues 서비스와 같은 방식으로 CASM-FMS간에 매핑 관계를 설정할 수 있으며, 이에 대한 자세한 사항은 참고문헌 [15]를 참조하기 바란다. CreateDataObject 서비스는 FMS에 대응되지 않는 서비스로서 추가로 구현되어야 할 서비스이며, 본 연구에서는 DefineNamedVariable이라는 이름으로 구현된다. 추가 서비스의 구현 방법은 4.1.2절에 기술하였다. 표 20에는 CreateDataObject 서비스의 CASM-FMS 매핑 관계가 나타나 있으며, 표에서 보는바와 같이 CreateDataObject의 요구 프리미티브의 속성인 DOReference, LocalResource, Type는 각각 추가로 구현되는 FMS 서비스의 ObjectName, Address, TypeSpecification에 매핑된다. CreateDataObject 서비스는 응답시 요구하는 값이 없으므로 Null 값을 가지며, Response-는 미리 지정된 에러 코드 중 적당한 것이 선택되어 응답된다. 추가로 구현되어야 하는 DeleteDataObject 서비스도 표 20에 나타난 CreateDataObject 서비스와 같은 방식으로 CASM-FMS간에 매핑 관계를 설정할 수 있으며, 이에 대한 자세한 사항 역시 참고문헌 [15]를 참조하기 바란다.

표 20 CreateDataObject 서비스의 CASM-FMS 매핑
Table 20 CASM-FMS mapping of createdataobject service

Service: CreateDataObject	FMS Services/Argument/Results	Server	Client
Request	DefineNamedVariable-Request		
DOReference	ObjectName	m	m
LocalResource	Address	m	m
Type	TypeSpecification	m	o
Response+	DefineNamedVariable-Response		
Null	Null		
Response-	error code 참조		

표 2에 나타난 바와 같이 DataObject 디바이스 모델은 DataObject 클래스 이외에 DeviceIdentify, DeviceModel, FunctionalComponents 클래스들을 포함한다. DeviceIdentity는 「structure」 Data type을 가진 DataObject에 매핑된다. DeviceIdentity는 항상 「DI」로 이름 지어진 NamedVariable이 되야 하며 FMS의 「record」 Data type을 가진다. DeviceModel은 「structure」 Data type을 가진 DataObject에 매핑되며 FMS의 「record」 Data type을 가진다. FucntionalComponent 역시 「structure」 Data type을 가진 DataObject에 매핑되며 FMS의 「record」 Data type을 가진다.

표 21 GetDataObjectValues 서비스 구현에 대한 CASM-FMS 매핑의 의사코드

Table 21 Pseudo-code of CASM-FMS mapping implementation for getdataobjectValues service

```
// UCA, Fieldbus Data structure를 선언
   DefineDataStructure();
// 데이터를 수신할 때까지 기다림
While( !DataReceived() )
  WaitforPDU();
// 패킷 헤더와 서비스 확인
CheckHeader();
CheckService();
// UCA에서 필드버스로 변환하는 경우
if ( SenderNetwork == UCA ){
  // 요청된 UCA 서비스에 대해 해당 FMS 서비스 선정
  FMS_Service = Check_UCAServiceType();
  if ( FMS_Service == "Read" ){
    // CASM 프리미티브 인수를 FMS 인수로 전달
    FieldbusObj.Read = UCAObj.GetDataObjectValue;
    // 메시지 전송을 위하여 하부 계층 서비스 호출
    CallLowerLayerService(message);
  }
  // "Read"서비스가 아닐 경우 해당 프로세스 처리
  else {
    Results = ProcessData();
    ReturnToSender(Results);
  }
}
// 필드버스에서 UCA로 변환하는 경우
else ( SenderNetwork == FMS ){
  // FMS 서비스에 대해 해당 UCA 서비스 선정
  UCA_Service = Check_FMSServiceType();
  if (UCA_Service == "GetDataObjectValue" ){
    // FMS 프리미티브 인수를 CASM 인수로 전달
    UCAObj.GetDataObjectValue = FieldbusObj.Read;
    // UCA PDU 생성후 해당 UCA 서비스 수행
    CreateUCAPDU(message);
    CallUCAService();
  }
// "GetDataObjectValue"서비스가 아닐 경우 해당 프로세스 처리
  else {
    Results = ProcessData();
    ReturnToSender(Results);
  }
}
// 처리결과를 송신부로 전송
ReturntoSender(Results);
```

표 21은 본 논문에서 제시하는 UCA CASM과 필드버스 FMS간의 매핑이 실제로 구현될 수 있음을 보이기 위하여, CASM의 서비스 중 표 19에 나타난 GetDataObjectValues 서비스의 매핑관계를 구현하기 위한 의사코드(Pseudo-code)를 제시하였다. 다른 CASM-FMS 서비스간의 매핑도 같은 방법으로 구현될 수 있다.

4.1.2 CASM-FMS 매핑을 위한 추가 서비스 구현

CASM에 직접 매핑되지 않는 FMS 서비스는 FMS가 CASM에서 요구하는 서비스를 제공해주지 못하는 경우이며, 이러한 서비스는 추가로 구현되어야 한다. 본 연구에서는 CASM과 MMS간에 매핑되는 서비스를 참고로 하여 MMS에서 제공하는 서비스를 FMS에 추가시키는 방법을 제시한다. 본 논문에서 CASM과 FMS의 매핑을 위한 추가 서비스의 구현은 다음의 세 단계를 통하여 기술된다.

- 서비스 프리미티브에 따른 서비스 파라미터의 정의
- ASN.1 정의
- 클라이언트-서버 관계에 의한 서비스 구현의 의사코드 작성

다음에는 추가 서비스 구현의 대표적인 예로 4.1.1절에서 제시된 DefineNamedVariable에 대한 서비스 구현 방법을 기술한다. DefineNamedVariable 서비스는 CASM의 CreateDataObject 서비스와 매핑된다.

표 22 Definenamedvariable 서비스의 프리미티브 정의

Table 22 Primitive definition of definenamedvarable service

Parameter Name	.req .ind	.res .con
Argument	M	
VariableName	M	
address	M	
TypeSpecification	U	
Result(+)		S
Result(-)		S
ErrorType		M

표 22에는 DefineNamedVariable 서비스의 프리미티브가 정의되어 있다. .req와 .ind는 각각 request와 indication 프리미티브를 표시하며, .res와 .con은 각각 response와 confirm 프리미티브를 나타낸다. 표 22에서 M(Mandatory), U(User Option), S(Selection) 는 각각 해당 파라미터 선택여부가 필수, 사용자선택, 양자택일임을 의미한다.

- 인수(Argument): 이 파라미터는 DefineNamedVariable 서비스 요구에 대한 다음과 같은 특정 파라미터들을 전송한다.
  - 변수이름(VariableName): 변수이름은 VFD에서 변수 객체를 유일하게 식별하는 이름을 지정한다.
  - 주소(address): 변수 객체의 주소 속성을 나타낸다. 이것은 형 지정 파라미터(TypeSpecification)에 의해 기술되는 변수에 대한 기본 또는 시작 주소를 나타낸다. 만약 여러 개의 주소가 이 형을 표현하기 위해 필요하다면 그들은 VFD의 인접한 위치들에 할당된다.
  - 형 지정(TypeSpecification) : 이 파라미터는 선택사항이다. 이것은 정의되는 변수(Named Variable) 객체의

형 속성을 정의한다.

- 결과(Result+): 결과(+) 파라미터는 서비스 요구의 성공을 나타낸다. 성공적인 결과는 서비스 특정 파라미터를 제공하지 않는다.
- 결과(Result-): 결과(-) 파라미터는 서비스 요구의 실패를 나타낸다. 실패에 대한 원인은 미리 지정된 에러 코드를 통하여 제공된다.

표 23은 표 22에 나타난 DefineNamedVariable 서비스에 대한 파라미터를 ASN.1(Abstract Syntax Notation-1) 정의에 따라 기술한 내용이다. 표에서 각 셀은 서비스의 프리미티브에 따라 request(indication)와 response(confirm)으로 구분하였으며 각 프리미티브의 속성은 위의 파라미터 도표의 속성과 동일하다. 표 24는 본 논문에서 제시하는 DefineNamedVariable 서비스가 실제로 구현될 수 있음을 보이기 위하여 제시한 의사코드(Pseudo-code)이다. 다른 추가 서비스들도 동일한 방법으로 구현될 수 있다.

표 23 DefineNamedVariable의 ASN.1 정의
Table 23 ASN.1 definition of definednamedvariable

ASN.1 정의		
DefineNamedVariable-Request	::= SEQUENCE {	
variableName		[0] ObjectName
address		[1] Address
typeSpecification		[2] TypeSpecification OPTIONAL
}		
DefineNamedVariable-Response	::= NULL	

4.2 CASM과 필드버스 사용자계층의 인터페이스

본 절에서는 CASM의 디바이스 모델과 서버 모델 가운데 필드버스 사용자계층으로 매핑되는 부분에 대한 매핑 방법을 기술한다. 필드버스 사용자계층은 필드버스 응용계층인 FMS의 데이터형과 범위를 그대로 쓰기 때문에 실질적으로 데이터형과 범위의 매핑은 FMS에 대해 이루어진다. 표 3에 기술된 바와 같이 CASM이 필드버스의 사용자계층에 직접 매핑되는 부분은 DataObject 디바이스 모델 가운데 FC(FunctionalComponent) 객체의 일부인 AX(Access), CO(Controls), MX(Measurements), SP(Setpoints), ST(Status)와 Device Control Service(Device Tagging) 모델의 Tag 객체뿐이다. 본 절에서는 이 가운데 MX객체의 하나인 AI(Analog Input)에 대하여 CASM과 사용자계층의 매핑 방법을 기술한다.

또한, 이러한 매핑이 실제로 구현될 수 있음을 보여주기 위한 의사코드를 제시한다. 다른 객체들도 동일한 방법으로 매핑 관계를 설정할 수 있으며, 이들에 대한 상세한 사항은 참고문헌[15]에 나타나 있다.

FC 객체는 필드 디바이스의 공통적인 기능을 한데 묶기 위한 구조화된 DO이다. 이 가운데 MX는 측정치, 또는 각 서비스의 입력값으로 사용되는 아날로그 데이터들을 나타내기 위해 사용된다. MX 가운데 사용자계층과 매핑되는 객체에는 AI가 있다. 표 25에는 CASM과 FOUNDATION Fieldbus의 사용자계층 간에 AI 매핑 관계가 나타나 있다.

AI는 아날로그 입력을 나타내며, CASM에서 AI의 실제 출력값을 나타내는 i(IntegerValue), f(FloatingPointValue)

표 24 DefineNamedVariable 추가 서비스 구현에 대한 의사코드
Table 24 Pseudo-code for implementation of definenamedvariable service

```
// 연결이 설정되었는지 판단
While ( communication != ENABLE )
  Waitfor_InitiateService();
// 메시지 수신 때까지 기다림
While ( Check_buffer() ){
  primitive = RetrieveHeader();
  data = RetrieveData();
  // 수신된 프리미티브가 confirm인지 확인
  if ( primitive == "confirm" ){
     // 응답결과가 (+)이면 서비스 종료
          if ( result == "yes" )
             Terminate_Service();
     // 응답결과가 (-)이면 오류처리
     else  CallError();
  else{
  // 데이터 블록 생성 - 변수이름, 주소, 형 지정
          make_VariableName;

          make_Address;
          make_TypeSpecification;
  // 데이터 블록을 송신버퍼에 저장 후 request 프리미티브 전송
          SendDataToBuffer();

          SendPrimitive("request");
  }
  // 수신된 프리미티브가 indication인지 확인
  if ( primitive == "indication" ){
  // 서비스 수행 - 요청된 주소에 객체 검색
          Check_Object();
  // 해당 주소에 변수가 없다면 변수 생성
          if ( Object == "absence" )
          {
             make_Object;
  // 데이터 블록을 송신 버퍼에 저장 후 응답(+) 프리미티브 전송
             SendDataToBuffer();
             SendPrimitive("response+");
          }
  // 응답(-) 프리미티브 전송
          else  SendPrimitive("response-");
  }
} // end of While
```

표 25 AI 클래스의 CASM-FOUNDATION Fieldbus 기능 블록 매핑
Table 25 CASM-FOUNDATION fieldbus function block mapping of AI class

DataObject Class Name	FB Class / Object
AI{	AI Block
i	OUT with SIMULATE = disabled
f	OUT with SIMULATE = disabled
q	Alarms
	Alert Analog Block
t}	TimeStamp

컴포넌트는 각각 FOUNDATION Fieldbus의 기능블록 가운데 AI 블록의 OUT으로 매핑된다. 이때 매핑되는 AI 블록에서 SIMULATE는 disabled로 설정된다. 또한 오브젝트 값에 대한 유효함, 통신오류, 유효범위초과 등을 나타내는 q(Quality)는 AI 블록의 Alarms로 매핑된다. 데이터가 최종적으로 갱신된 시각을 나타내는 t(TimeStamp)는 AI 블럭에 시각을 나타내는 별도의 파라미터가 없기 때문에 Alert Analog 블록의 TimeStamp 파라미터로 매핑된다. 즉, AI는 FOUNDATION Fieldbus 기능 블록의 AI 블록과 AlertAnalog 블록을 연결한 블록에 매핑된다.

표 26 AI 클래스의 CASM-PROFIBUS-PA 매핑
Table 26 CASM-PROFIBUS-PA mapping of AI class

DataObject Class Name	PA Object
AI(i f q t)	 Discrete Structure Value Floating Point Structure Value Status with Block Object BYTE #2 - 02 BYTE #3 - 01 BYTE #4 - 01 Bitstring Structure Value

표 26에는 CASM과 PROFIBUS-PA의 사용자계층 간에 AI 매핑 관계가 나타나 있다. AI는 실제적인 데이터 값이 정수형인지 실수형인지에 따라 Discrete Structure 또는 Floating Point Structure로 매핑되고, q는 해당 Discrete 또는 Floating Point Strucutre의 Status에 매핑된다. 이때 Block Object는 BYTE #3 = 01(Input), BYTE #4 = 01(Analog Input)으로 설정된다. 또한 TimeStamp를 나타내기 위한 t는 추가적인 Bitstring Structure를 정의하여 그 Bitstring Structure로 매핑된다.

표 27에는 CASM의 FC 가운데 AI와 FOUNDATION Fieldbus의 기능 블록의 매핑에 대한 의사코드 작성의 예를 제시하였다. UCA의 CASM과 필드버스 사용자계층의 매핑을 실제적으로 구현하는 경우, 사용자계층에서는 CASM과의 객체 매핑만 수행하게 되므로, UCA 또는 필드버스로부터 수신한 데이터를 필드버스 또는 UCA의 데이터형으로 바꾸어 전송하고 작업을 종료하거나 처리 결과를 반환하는 형태로 구현된다.

5. 결론 및 추후 연구 사항

발전소, 변전소, 송배전소 등의 필드에 설치되는 장비들은 매우 열악한 환경에서 동작되며, 또한 주어진 시간 내에 데이터 전송의 완료되어야 하는 실시간 통신 요구조건을 만족하여야 한다. 따라서 필드 디바이스들간의 통신을 위해서는 자동화 시스템 전용 산업통신망인 필드버스를 사용하여야 한다. 본 연구에서는 발전 설비에서 원격 실시간 모니터링,

표 27 AI 객체 매핑 구현에 대한 의사코드
Table 27 Pseudo-code for mapping of AI object class

```
// UCA, Fieldbus Data structure를 선언
DefineDataStructure();
// 데이터를 수신할 때까지 기다림
While( !DataReceived() )
  WaitforPDU();
// 패킷 헤더와 서비스 확인
CheckHeader();
CheckService();
// UCA에서 필드버스로 변환하는 경우
if ( SenderNetwork == UCA )
{
// 오브젝트 매핑
  // 수신한 데이터가 AI 오브젝트인 경우
  if ( UCAObjType == "AI")
  {
    if ( UCAObjDataType == "INTEGER")
          FieldbusObj.AIBlock.OUT = UCAObj.AI.i;
    else if ( UCAObjDataType == "FLOAT")
          FieldbusObj.AIBlock.OUT = UCAObj.AI.f;
    FieldbusObj.AIBlock.Alarms = UCAObj.AI.q;
    FieldbusObj.AlertAnalogBlock.TimeStamp = UCAObj.AI.t;
    // AI block의 SIMULATE 파라미터를 disabled로 설정
    FieldbusObj.AIBlock.SIMULATE = disabled;
    // 매핑을 끝낸 데이터를 필드버스 측으로 송신
    CreateFieldbusPDU( FieldbusObj );
    Results = SendPDU( "FIELDBUS", FieldbusObj );
  }
  // AI 오브젝트가 아닌 경우 다른 과정 수행
  else
   Results = ProcessData();
}
// 처리결과 송신
ReturntoSender(Results);
```

상태 진단, 고장 예측 및 실시간 제어를 위한 통합 통신망 구축에 있어서 가장 먼저 해결하여야 할 사안인 UCA와 필드버스의 인터페이스 방안을 제시하였다.

본 연구에서는 현존하는 여러 종류의 필드버스들 가운데 유럽지역에서 개발되어 현재 전세계적으로 가장 높은 시장 점유율을 보이고 있는 PROFIBUS와 최근에 미주지역에서 개발이 완료되어 앞으로 널리 사용될 것으로 예상되며, 가장 빠른 시장 증가율을 보이고 있는 FOUNDATION Fieldbus를 개발 대상으로 UCA와 필드버스의 인터페이스 방안을 제시하였다. UCA와 필드버스를 통합하는 경우 사용자가 UCA 환경에서 구현한 응용 프로세스는 필드버스의 응용계층과 사용자계층 프로토콜에 자동적으로 인터페이스 되어야 한다. 본 연구를 통하여 제시한 UCA-필드버스 인터페이스 방안은 전력 설비 사용자가 하부계층의 통신망 프로파일에 관계없이 UCA 사용자계층에서 제공하는 서비스만을 이용하여 응용 프로세스를 구현할 수 있는 환경을 제공한다.

필드버스가 UCA 통합 네트워크에 바로 접속될 수 있으면 통합 정보망의 구축은 용이해질 것이다. 그러나 필드버스와 UCA는 서로 상이한 통신망 구조를 가지고 있다. 또한, UCA의 CASM 프로토콜은 응용계층으로 MMS를 사용하고 있으나, 필드버스는 응용계층 이외에 추가적으로

사용자계층을 정의하고 있기 때문에 UCA를 필드버스의 응용계층 또는 사용자계층의 어느 한쪽으로만 매핑하는 것은 사실상 불가능하다. 따라서 본 연구에서는 UCA와 필드버스의 인터페이스를 (i) 필드버스의 응용계층으로 직접 매핑이 가능한 부분, (ii) 응용계층에는 직접 매핑되지 않으나, 사용자계층에 바로 매핑 가능한 부분, (iii) 응용계층과 사용자계층 양쪽 어느 계층으로도 매핑되지 않는 부분으로 구분하여 매핑방안을 제시하였다. 또한, 이러한 매핑이 실제로 구현될 수 있음을 보이기 위한 의사코드(Pseudo-code)를 제시하였다.

본 연구의 결과는 국내의 발전 설비에 UCA를 이용한 통합 자동화 정보망을 구축하는데 바로 활용될 수 있을 것이다. UCA는 또한 전력 설비뿐만이 아니라 수도, 가스 설비 등의 모든 공공 설비에 활용될 수 있어 국내 여러 산업에 대한 파급 효과가 매우 클 것으로 기대된다. 본 연구에서는 UCA와 필드버스를 통합하기 위한 기본 방안만을 제시하였으며, 추후 실제로 UCA와 필드버스를 접속시키는 게이트웨이를 실제로 개발하기 위한 후속 연구가 수행되어야 할 것이다.

감사의 글

본 연구는 1998년도 한국전력공사의 지원에 의하여 기초전력공학 공동연구소 주관으로 수행된 연구로서, 관계부처에 감사드립니다.(과제번호 98-080)

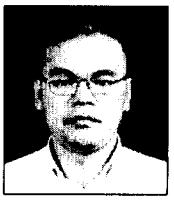
참 고 문 헌

[1] Introduction to UCA version 2.0
[2] UCA Profile Specification, Version 2.0
[3] UCA Common Application Service Models(CASM)
[4] Generic Object Models for Substation and Feeder Equipment(GOMSFE)
[5] IEC 870-6-503: TASE.2 Service and Protocol, Version 1996-08
[6] IEC 870-6-802: TASE.2 Object Models, Version 1996-08
[7] IEC 870-6-702: TASE.2 Application Profile, Version 1996-08
[8] ISO/IEC 9506 Manufacturing Message Specification
[9] KS 산업 자동화 시스템-생산 메시지 시방, 국립기술품질원
[10] DIN 19245 PROFIBUS Standard
[11] PROFIBUS-PA Protocol Specification Version : 1.0, 1.2.1995
[12] PROFIBUS-PA Profile for Process Control Devices Class B
[13] FOUNDATIONTM Communication Specification Release 1.0
[14] FOUNDATIONTM Function Block Application Process: Part 1 and 2 Release
[15] 홍승호 외 4인, UCA(Utility Communications Architecture)를 이용한 전력설비 자동화 통신망 구축방안: 최종보고서, 기초전력공학공동연구소(과제번호 98-080), 1999. 10.

저 자 소 개

최 인 호 (崔 仁 鎬)
1973년 4월 6일 생. 1999년 울산대 제어계측공학과 졸업. 1999년~현재 한양대 제어계측공학과 석사과정
Tel : 0345-400-4084
E-mail : adonis73@hymail.hanyang.ac.kr

황 인 휘 (黃 仁 輝)
1973년 11월 15일 생. 1996년 한양대 제어계측공학과 졸업. 1998년 동 대학원 제어계측공학과 졸업(석사). 1999년~현재 동 대학원 제어계측공학과 박사과정
Tel : 0345-406-8967
E-mail : inie@hymail.hanyang.ac.kr



홍 승 호 (洪 承 鎬)
1956년 5월 31일 생. 1982년 연세대 기계공학과 졸업. 1985년 Texas Tech. University 기계공학과(석사). 1989년 Pennsylvania State University 기계공학과 졸업(공박). 1989년~1992년 한국 전자통신연구소 선임연구원. 1992년~현재 한양대 전자컴퓨터공학부 부교수
Tel : 0345-400-5213, Fax : 0345-406-4132
Email : shhong@email.hanyang.ac.kr